DarkSUSY DarkSUSY_main

Manual and description of routines

with full routine headers

Documentation harvested from documentation and source directories with harvestdoc.pl
Fri Dec 9 22:05:03 2022

`http://www.darksusy.org`

Joakim Edsjö[a*], Torsten Bringmann[b†], Paolo Gondolo[c‡], Piero Ullio[d§], Lars Bergström[a¶], Mia Schelke[‖], Edward A. Baltz[**] and Gintaras Duda[††]

[a] *The Oskar Klein Centre for Cosmoparticle Physics, Department of Physics, Stockholm University, AlbaNova, SE-106 91 Stockholm, Sweden.*

[b] *Department of Physics, University of Oslo, Box 1048, NO-0371 Oslo, Norway*

[c] *Department of Physics, University of Utah,115 South 1400 East, Suite 201, Salt Lake City, UT 84112-0830, USA.*

[d] *SISSA and INFN, Sezione di Trieste, via Bonomea 265, 34136 Trieste, Italy.*

[*] E-mail address: edsjo@fysik.su.se
[†] E-mail address: torsten.bringmann@fys.uio.no
[‡] E-mail address: paolo.gondolo@utah.edu
[§] E-mail address: ullio@he.sissa.it
[¶] E-mail address: lbe@fysik.su.se
[‖] E-mail address: schelke@...
[**] E-mail address: eabaltz@physics.columbia.edu
[††] E-mail address: gkduda@creighton.edu

Abstract

DarkSUSY is a package for numerical dark matter calculations including, but not limited to, supersymmetric dark matter. This manual describes the theoretical background as well as details about the actual routines. Everything is not covered, but it should hopefully prove useful if you need more information than in our published articles.

Disclaimer. This manual is work in progress.

We try to keep it clear and up-to-date with the code, but there will be cases where changes/improvements in the code, for one reason or another, will not have propagated into the manual. Hence, check the actual code if you want to be certain about how a given process/feature is implemented.

Contents

Part I

Prelude

Chapter 1

Introduction

DarkSUSY is a set of Fortran routines to allow advanced numerical calculations connected to dark matter physics. In part I of this Manual we explain the basic structure of the code, and provide an introduction on how to get quickly started and use it. Part II is devoted to a description of the DarkSUSY *core library*, which contains all routines that are completely independent of the underlying DM particle physics. In part III, we describe the additional libraries, or *particle modules*, for specific DM models that are currently implemented (including supersymmetric dark matter in the Minimal Supersymmetric Standard Model, the MSSM). The modular structure of the code allows to easily link the core library to any of those particle modules, as well as to add user-defined new modules in a straight-forward way.

In this manual we will mainly cover the more techincal aspects of DarkSUSY, i.e. how to call different subroutines, both particle-physics dependent and not, and how to change various switches and options in more advanced routines. We will only briefly review the necessary physics involved when needed and refer the reader to [1, 2] and the original papers behind DarkSUSY (see Section 5) for more details. If you use DarkSUSY please consider the original physics work behind and give proper credit to [2] and the relevant references in Section 5. If you use non-standard options, e.g. a different propagation model for antiprotons, please remember to give proper credit to that model.

Chapter 2

Quick start guide

2.1 Installation

To get started, first download DarkSUSY from `www.darksusy.org` and unpack the tar file. To compile, run the following in the folder where you unpacked it

```
./configure
make
```

You will then have compiled the main DarkSUSY library, as well as all supplied particle physics modules. To test whether the installation was successful, type

```
cd examples/test
./dstest
```

The program will take up to a minute to run and reports if there are any problems.*
After about a minute's runtime, you should get an output of the form

```
====================================

    Summary of performed DarkSUSY tests

 ====================================

    Number of errors reported by dstest_mssm:           0

    Number of errors reported by dstest_silveira_zee:   0

    Number of errors reported by dstest_genWIMP:        0

====================================
```

If you get something else then 0 errors, you should check the output more carefully. The way the test program runs is that for each observable it compares the result with a pre-calculated value.

*Strictly speaking, this is only a test of the default particle physics module, mssm. To see what happens behind the scenes, you can run in verbose mode by replacing 'testlevel/2/' with 'testlevel/1/' at the beginning of dstest.f. Then type make and run dstest again. The program is also extensively commented, and can be used to learn about which DarkSUSY routines to call.

If the difference is larger than 0.3% an error is issued. You normally would not expect larger differences than this due to just numerical errors.

Even if you now have DarkSUSY running, it is more fun to start doing some calculations on your own. Possible next steps are explained in the next Sections.

Happy running!

2.1.1 Options for install

Even if the above install usually works, sometimes you want to use special options, like a specific compiler. Most options are specified at the time of configure and then propagated through to the DarkSUSY makefiles. Examples of options are

- To specify that you want to compile with gfortran, you can e.g. run configure with the following options (on one line)

```
./configure CC=gcc CFLAGS=-g CXX=g++ CXXFLAGS=-g FC=gfortran
FCFLAGS="-O -ffixed-line-length-none -fopenmp"
```

For your convenience, this particular choice is a available as a script conf.gfortran that can be invoked instead of the string above.

2.1.2 System requirements

DarkSUSY should run on most Linux/Unix systems including Mac OS X. You need to have a Fortran, C and C++ compiler available and the typical developer tools (like make and ranlib). You also need to have perl installed. If you are creating new particle physics modules and want to use the tools available to automatically create makefiles for you, you also need to have autoconf installed.

External, or 'contributed', code residing in /contrib typically needs further packages installed, and relies on more specific system requirements. As of DarkSUSY version 6.2, the additional requirements include curl (+libcurl!), autoconf, aclocal and cmake. Note that these requirements may change, *please consult the webpage for the most up-to date list*! Experience has shown that some of the contributed packages are notoriously challenging to build on some systems, such that the standard make command does not even proceed to the actual DarkSUSY code. In case you cannot resolve these errors, the make system offers a leightweight version of DarkSUSY that does not depend on contributed packages at all. For this, just do make distclean followed by configure and then

```
make darksusy_light
```

instead of the usual make. No contributed code will be built in this case (and particle modules heavily relying on these external packages, like the mssm module, will be disabled).

2.2 Example programs

DarkSUSY is primarily a library that is intended to be used with your own main programs. However, to get you started, we supply a few sample programs in the /examples directory. These can be used as they are, but they are also extensively commented to help you understand which routines you are supposed to call for the most typical calculations. The most instructive general-purpose programs in /examples, apart from the already mentioned /test/dstest, are

dsmain_wimp.F An example main program to calculate various DM observables. It can be the starting point if you want to write your own programs (see below). Note that you can run *the same code* with different particle modules that provide a WIMP DM candidate (the default is the mssm module); simply do
./> make -B dsmain_wimp DS_MODULE=⟨MY_MODULE⟩
and then ./dsmain_wimp in the /examples directory. Choosing ⟨MY_MODULE⟩=generic_wimp, e.g., will produce results for a generic WIMP DM candidate (see Chapter 38 for details on the implementation).

dsmain_decay.f A main program that calculates the same observables, where relevant, as dsmain_wimp.F – but for a generic decaying DM candidate (see Chapter 36 for details on the implementation) rather than a WIMP.

2.2.1 Auxiliary example programs

In the folder examples/aux/ we list a set of instructive auxiliary example programs that illustrate more specific usage of DarkSUSY. These example programs are typically set up for specific particle physics modules (e.g. generic_wimp) but, as explained below in more detail, it is straightforward to use them for other particle physics modules as well. Below we describe some of the currently available auxiliary example programs – but *note that the actual list of example programs in examples/aux/ is growing with almost every, even minor, release.* We thus strongly advise to check the actual list of these programs, and their headers for a quick summary.

flxconv.f This program can be used to convert between different fluxes from the Sun and the Earth. It can be use to convert a limit on a muon flux to a limit on the scattering cross section, or to a limit on the annihilation rate (or vice versa).

DMhalo_predef.f This program illustrates how to use the default version of dsdmsdriver (provided with the DarkSUSY release) to load additional profiles into the currently active halo database by using its pre-defined halo parameterizations.

DMhalo_table.f This program demonstrates how to load a halo profile from a table in an accompanying data file.

DMhalo_new.f This program demonstrates how to correctly extend dsdmsdriver when adding a new profile parameterization in order to consistently make it available to all DarkSUSY routines that rely on the DM density (in this concrete example, we add the spherical Zhao profile [3], aka $\alpha\beta\gamma$ profile).

DMhalo_bypass.f Here, we demonstrate a work-around of completely bypassing the default dsdmsdriver setup when switching to a user-provided new DM density profile. While easier to implement than DMhalo_new.f, such an approach has the significant drawback that the advanced DarkSUSY system of automatic tabulation of quantities related to DM rates cannot easily be exploited.

generic_wimp_oh2.f This program calculates the annihilation cross section needed to produce the correct relic density (as measured by Planck within errors). It does this for a range of WIMP masses so that you can plot e.g. the required annihilation cross section versus mass. It also let's you incorporate threshold effects (either as a hard cut, default), or with a more sophisticated sub-threshold treatment, that also illustrates the use of replaceable functions.

ScalarSinglet_RD.f This program calculates the couplings required to get the correct relic density in the Silveira-Zee (scalar singlet) model.

ucmh_test.f This program is an example of how the ultra-compact mini halo routines can be used.

wimpyields.f This is a simple program that sets up a generic WIMP with a given mass that annihilates to a given channel and then calculates the yields of different particles from the hadronization/decay of the annihilation products.

caprates.f This is a simple program that scans a range of WIMP masses and calculates the capture rates in the Sun via spin-independent and spin-dependent scattering.

caprates_ff.f This program is similar to caprates.f, but a bit more advanced as it performs the capture rate calculation with the most complete numerical setup and also calculates the capture rate on individual elements in the Sun.

2.2.2 Making your own example programs

The simplest way to create your own example program is to copy one of the Fortran example files in examples/aux/ to a directory of your choice, then change the name of that file and modify it. You also need to copy the makefile from examples/aux/ to the same directory, and make sure that you update the name of the file that you just changed.[†] Running 'make' then compiles your own new example program without touching the release version of the DarkSUSY code. Always keeping your own code, or your modifications of DarkSUSY, separate like this is the recommended way to proceed because it facilitates debugging and minimizes the risk of introducing errors. Don't modify any of the DarkSUSY makefiles directly as they are overwritten every time you run configure.

If you want to *compile with a different particle physics module*, you will in general need to do go through three simple steps:

1. change the corresponding block in the makefile, i.e. simply set the variable DS_MODULE to a different value (you may also have to add additional libraries, e.g. lisajet for the mssm module)
2. update the model setup routines to the new particle module (see the various example programs; for more details, have a look at the respective Section of this manual and the header of setup routines starting with dsgivemodel).
3. make sure that your main program only calls routines that are supported for the new particle module. (If it doesn't, it will not compile, stating the functions that are not supported)

An explicit demonstration of how all these three steps are taken care of in one single example is provided in dsmain_wimp.F. Note that the pre-compiler directives in that example are only necessary if you want to be able to compile the *same* code with two (or more) different particle modules.

2.3 Modifying individual subroutines or functions

If you want to modify some existing DarkSUSY function or subroutine, *don't do it,* instead add your own routine as a replaceable function (see also Section 3.6), by running the script scr/make_replaceable.pl on the routine you wish to have a user-replaceable version of. You will then find that version in the corresponding user_replaceables folder, where you can edit it to your liking. Following these steps, it is guaranteed that the newly created user-replaceable function is properly included in the library where the original DS function used to be, with all makefiles being automagically updated.

An alternative – and often even simpler – way of using user-replaceable functions is to leave the DS libraries untouched, and to instead link to the user-supplied function only when making the main program (this option is indicated in the left-most part of Fig. 3.1). For an explicit example, see (the makefile for) generic_wimp_oh2.f.

[†]If you want, you can delete all the blocks for the other (not needed) example programs, in order to have the makefile look clearer and easier to understand.

Chapter 3

Guiding principles of the code

DarkSUSY (as of version 6) has a new structure compared to earlier versions of the code. The most striking difference is that we have split the particle physics model dependent parts from the rest of the code. This means in practice that we have separated DarkSUSY into one set of routines, ds_core, which contains no reference to any specific particle model, as well as distinct sets of routines for each implemented model of particle physics. For supersymmetry, for example, all routines that require model-dependent information now reside in the mssm module. The advantage with this setup is that ds_core and the particle physics modules may be put in separate FORTRAN libraries, which implies that DarkSUSY can now have several particle physics modules side by side. The user then simply decides at the linking stage, i.e. when making the main program, which particle physics module to include.

For this to work, the main library communicates with the particle physics modules via so-called *interface* functions (or subroutines), with pre-defined signatures and functionalities. Note that a particle physics module does not have to provide all of these predefined functions: which of them are required is ultimately determined only when the user links the main program to the (ds_core and particle module) libraries. Assume for example that the main program wants to calculate the gamma-ray flux from DM (a functionality provided by ds_core). This is only possible if the particle module provides an interface function for the local cosmic ray source function; if it does not, the main program will not compile and a warning is issued that points to the missing interface function. If, on the other hand, interface functions required by direct detection routines would be missing in this example, this would not create any problems at either runtime or the compile stage.

In addition, we have added the concept of *replaceable functions*, which allows users to replace essentially any function in DarkSUSY with a user-supplied version. DarkSUSY ships with dedicated tools to assist you setting up both replaceable functions and new particle physics modules. Fig. 3.1 illustrates these concepts by showing how a typical program would use DarkSUSY; below, we describe each of them in more detail.

3.1 The main DarkSUSY library ds_core

As introduced above, the main library is in some sense the heart of the new DarkSUSY, offering all the functionality that a user typically would be interested in without having explicitly to refer to specific characteristics of a given particle physics model (after initialization of such a model). The main library thus contains routines for, e.g., cosmic ray propagation, solar models, capture rates for the Sun/Earth, a Boltzmann solver for the relic density calculation, yield tables from annihilation/decay etc. None of the routines in the main library contains any information about the particle physics module. Instead any information needed is obtained by calling a required function that resides in the particle physics module which is linked to (see below).

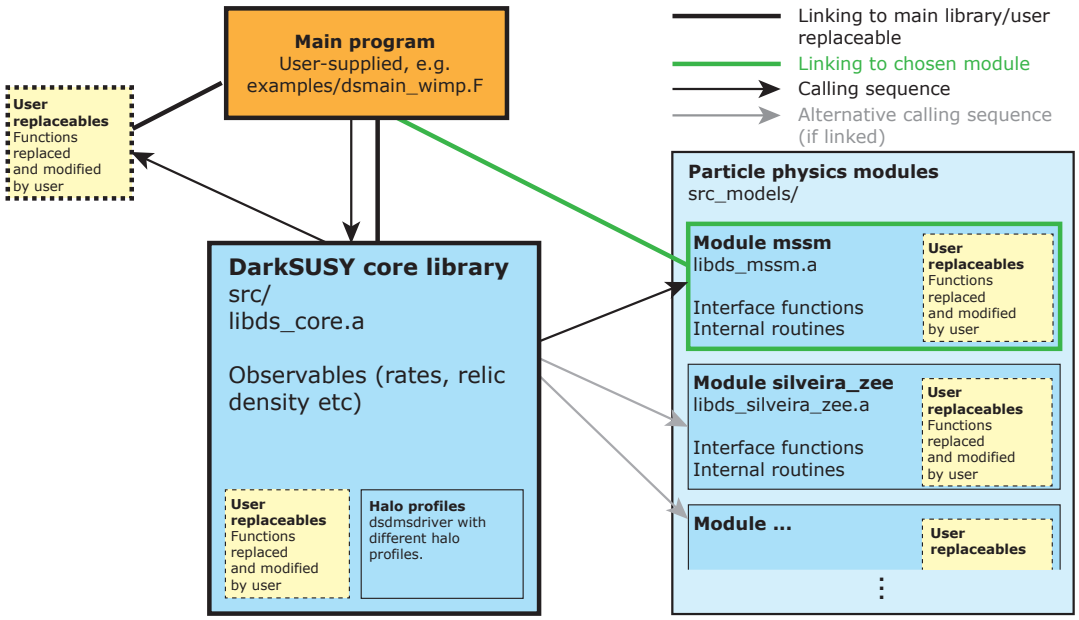

Figure 3.1: Conceptual illustration of how to use DarkSUSY. The main program links to both the main library, ds_core, and to *one* of the available particle physics modules. User-replaceable functions are optional and may be linked to directly from the main program, or indirectly by including them in the various libraries. See examples/dsmain_wimp.F for an example of a main program that demonstrates typical usage of DarkSUSY for different particle physics modules.

The source code for all functions and subroutines in the main library can be found in the src/ directory of the DarkSUSY installation folder, with subdirectory names indicating subject areas as summarized in Table 3.1.

3.2 Particle physics modules

The particle physics modules contain the parts of the code that depend on the respective particle physics model. Examples are cross section calculations, yield calculations etc. The routines in the particle physics module have access to all routines in ds_core, whereas the reverse is in general not true (with the exception of a very limited set of interface functions that each particle module provides).

DarkSUSY 5 and earlier was primarily used for supersymmetric, and neutralino DM, and those parts of the code now reside in the MSSM module mssm. However, many people used DarkSUSY even before for e.g. a generic WIMP setup, which was doable for parts of the code, but not all of it. We now provide a generic WIMP module generic_wimp that can be used for these kinds of calculations in a much more general way. In a similar spirit, we also provide a module generic_decayingDM for phenomenological studies of decaying DM scenarios. As an example for an actual particle physics model other than supersymmetry, DarkSUSY furthermore includes a module silveira_zee which implements the DM candidate originally proposed by Silveira and Zee [4] and which now is often referred to as Scalar Singlet DM [5, 6, 7]. We also include an empty model, empty, which is of course not doing any real calculations, but contains (empty versions of) all interface functions that the core library is aware of – which is very useful for debugging and testing purposes. Designing new particle modules is a straight-forward exercise, see below, and it is generally a good idea to start with the most similar module that is already available.

Subdirectory name	Description
an_yield	Annihilation yields in the halo – yields from simulations
aux	General routines
aux_xcernlib	CERN routines needed by DarkSUSY
aux_xdiag	Diagonalization routines
aux_xnswclib	
aux_xquadpack	CMLIB routines needed by DarkSUSY
cr_aux	Cosmic rays – general
cr_axi	Cosmic rays – diffusion routines for axisymmetric distributions
cr_gamma	Cosmic rays – Gamma fluxes
cr_nu	Cosmic rays – Neutrino fluxes
cr_ps	Cosmic rays – point sources
dd	Direct detection
dd_crdm	Direct detection – cosmic ray upscattered DM
dmd_astro	Astrophysical source functions
dmd_aux	Auxiliary functions for dark matter distribution routines
dmd_mod	Dark matter distributions
dmd_vel	Dark matter velocity distributions
fi	Freeze-In
ini	Initialization routines
kd	Kinetic decoupling
rd	Relic density
se_aux	Auxiliary routines for WIMP annihilation in the Sun/Earth
se_mod	Sun and Earth models
se_nu	Capture and annihilation in the Sun/Earth
se_yield	Yields from annihilation in the Sun/Earth
si	Self-interactions
ucmh	Ultra-compact mini-halos

Table 3.1: Organization of the main library ds_core: all functions and subroutines reside in the src/ folder of the DarkSUSY installation, with the names of the subdirectories indicating the subject area. (Note: this table is automatically generated from the actual directory structure in src/.)

A list of the currently available particle physics modules is given in table 3.2, and the source code for the particle modules can be found in the src_models/ directory of the DarkSUSY installation folder, each of the subdirectories (e.g. src_models/mssm/, src_models/generic_wimp/) typically reflecting a (sub)subdirectory structure analogous to what is shown in Table 3.1 for the core library. Many dark matter models, furthermore, constitute only relatively simple extensions to the standard model, inheriting most of its structure. For convenience, we therefore also provide various auxiliary routines, in src_models/common/sm, that each particle module automatically has access to, and which return basic standard model quantities like, e.g., the masses of standard model particles and their running (so additional BSM effects have to be implemented in the respective particle module).

Particle physics modules

Module No.	Short name	Description
1	**common**	Basic principles and common routines
2	**empty**	The empty model
3	**generic_decayingDM**	Generic decaying dark matter
4	**generic_fimp**	Generic FIMPs
5	**generic_wimp**	Generic WIMPs
6	**mssm**	The Minimal Supersymmetric Standard Model
7	**silveira_zee**	Silveira-Zee (Scalar Singlet)
8	**vdSIDM**	Self-Interacting Dark Matter

Table 3.2: List of particle physics modules currently available in src_models. (Note: this table is automatically generated from the actual content of src_models.)

3.2.1 Using a particle physics module

How to use a particle physics module obviously depends on how it is implemented, and in principle there are no formal requirements on how this should be done – as long as the provided interface functions are correctly set up, see Section 3.4 further down. The one subroutine that *is* required to exist, however, is

```
dsinit_module
```

This is called directly from dsinit, and must be the first call to the particle module as it initializes all general settings and relevant common block variables.

While there is no required structure otherwise, there is a typical workflow associated to using a particle physics module in a main program. First, one needs to initialize a given model by specifying its model parameters. This is done by routines like

```
dsgivemodel...
```

A call to dsgivemodel_decayingDM, e.g., allows to enter the defining parameters for a DM model in the generic_decayingDM module, while dsgivemodel_25 allows to enter the parameters for pMSSM model with 25 parameters in the mssm module. The next step is then typically a call to

```
dsmodelsetup
```

in order to transfer the model parameters to common blocks and calculate basic quantities like masses and couplings. Once a model is set up like this, a main program can use the full functionality of DarkSUSY supported by the respective module.

3.2.2 Adding a new particle physics module

To create a new particle physics module, the easiest way is to start from an existing one as a template and create a new one from that one. To help you in this process we provide a script scr/make_module.pl that takes two arguments, the module you want to start from and the new one you want to create (for further instructions, just call the script without arguments). It will then copy the module to a new one, change its name throughout the module and make sure that it is compiled by the makefiles and included properly when requested by the main programs. If you specify the option -i only interface functions will be copied (which creates a cleaner starting point, but also will most likely not compile without modifications). When creating a new module this way, the best is to copy from a module that is as similar as possible to your new model. If

you want a clean setup, you can always copy from the empty module. If you want something more phenomenological that has a basic framework for calculating observables, starting from the generic_wimp or generic_decayingDM might be a good idea. A general advice is to view the modules we provide as a starting point as inspiration for your new modules.

Even though a particle physics module does not need to include all interface functions (which ones are needed only depends on the observables you try to calculate in your main programs), it needs to provide an initialization routine dsinit_module.f. This routine should set a global variable moduletag to the name of the module so that routines that need to check if the correct module is loaded can do so. When using the script scr/make_module.pl this routine is always created and moduletag set as it should.

Note that the script scr/make_module.pl only provides the framework in the configure script and makefiles (or rather makefile.in's) to make sure your module is compiled. It does *not* create a main program that uses your module, that is up to you. A good starting point for that can either be e.g. dsmain_wimp.F in examples that already contain different blocks for different modules (controlled by pre-compiler directives, see the code for more details). Another option is to use any of the example programs in examples/aux as a starting point.

To make upgrading to new DarkSUSY versions as smooth as possible, we advice to create your own folder with your main programs, using e.g. the makefile in examples/aux as a starting point. Don't modify any of the DarkSUSY makefiles directly as they are overwritten every time you run configure, see Section 2.2.2 for more details.

Note that for the scr/make_module.pl script to work you need to have autoconf installed as the script adds your new module to configure.ac and runs autoconf to create a new configure script.

3.3 Halo models

Several routines in the core library need to know which DM halo should be adopted for the calculations. With this DarkSUSY version, we introduce a new and flexible scheme that avoids pre-defined hardcoded functions to describe the DM density profiles, and allows to consistently define different DM targets at the same time. For convenience, we still provide several pre-defined options for such halo parameterizations, and the user can choose between, e.g., the Einasto [8] and the Navarrow-Frenk-White profile [9], or read in any tabulated axi-symmetric (or spherically symmetric) profile. On a *technical* level, the halo models are handled by the dsdmsdriver routine which contains a database of which halo profiles the user has set up, and consistently passes this information to all parts in the code where it is needed.*

We provide detailed hands-on examples on how to use this system by a set of example programs DMhalo_*.f, see Section 2.2.1. For further details, we refer to Section 21.1 of this manual.

3.4 Interface functions

Interface functions are functions that routines in ds_core might need to call, and which therefore every particle physics module should contain if that particular observable is requested. Examples of interface functions include dsddsigma that returns the equivalent DM nucleus cross section, dscrsource that returns the source term for DM-induced cosmic rays (relevant for indirect detection), dsanxw that returns the invariant annihilation rate (in the case of WIMPs), etc. All interface functions are found by looking in the empty module, and always contain the keyword 'interface' in the function or subroutine header.

*From a technical point of view, it is actually not dsdmsdriver itself which acts as an interface to the rest of the code, but the set of wrapper routines collected in src/dmd_astro (which all call dsdmsdriver in a specific way). Only those routines are called by cosmic-ray flux routines and other functions in src, never dsdmsdriver directly. The routines in src/dmd_astro therefore provide examples of functions that *cannot be replaced* individually in a consistent way, but only as a whole set (along with dsdmsdriver, in case the user wants to change the structure of the driver itself).

If a particle physics module contains the full set of interface functions, all routines in the main DarkSUSY code should work. However, this is not needed to set up a paticle physics module. E.g. a generic WIMP model does not need to have decay rates set up. If a routine in the main DarkSUSY routines is called that rely on this interface being present, an error will be thrown when trying to compile your code. This of course applies to all interface functions, not just unphysical ones. E.g. if you are only interested in the relic density for a new particle physics module, you do not need to set up scattering rates, cosmic ray source functions, etc.

In Tab. 3.3, we provide the complete list of interface functions currently implemented in DarkSUSY, along with a brief description. For more details, consult the headers of these files.

3.5 Commonly used functions

Routines that we believe are particularly useful for most users to call, we denote as *commonly used functions*. A list of the commonly used functions in the main DarkSUSY library in src is shown in Table 3.4. The routines are described in more detail in their corresponding chapters in part II of this manual, as indicated in the table 3.4.

3.6 Replaceable functions

The concept of replaceable functions introduces the possibility to replace a DarkSUSY routine with one of your own. The way it works is that the user-provided function will be linked when you make your main program instead of the DarkSUSY default one. If, for example, you want to replace the yields from a typical final state of DM annihilation or decay (like $\bar{b}b$) with a new function of your own – e.g. because you are interested in comparing the tabulated Pythia [10] yields with those provided by PPPC [11] – you create your own version of the routine dsanyield_sim and let DarkSUSY use this one instead. To help you with this setup, we provide tools that can create (or delete) a replaceable function from any DarkSUSY function and set up the makefiles to use this user-supplied function instead. We also provide a simply way of managing large 'libraries' of user-supplied functions, via a list imported by the makefiles, where the user can determine on the fly which user-replaceable functions should be included and which ones should not. Note that both routines in ds_core and in any of the particle physics modules can be replaced in this way, including interface functions. There is also a possibility for the particle physics modules to replace a function in the core library. This is not used often, but is used e.g. for the function dsrdxi.f (which gives a possibility to have different dark matter and background temperatures in the early Universe).

There is also a simple way to bypass this system – indicated in the left-most part of Fig. 3.1 – by directly linking to the user-supplied function only when making the main program. For an explicit example, see (the makefile for) generic_wimp_oh2.f.

Routine	Appears in modules	Used by	Description
dsacbnd	2, 6		Check collider bounds
dsanwx_finiteT	2, 4, 7	fi	Self-annihilation invariant rate at finite temperature
dsanwx	2, 4, 5, 6, 7, 8	fi, rd	Self-annihilation invariant rate
dscrsource_line	2, 3, 5, 6, 7, 8	cr_axi, cr_gamma, cr_nu	Source term for monochromatic contributions from dark matter
dscrsource	2, 3, 5, 6, 7, 8	cr_axi, cr_gamma, cr_nu, ucmh	Source term for dark matter induced cosmic rays
dsddgpgn	2, 5, 6, 7, 8	se_nu	Four fermion couplings.
dsddsigma	1, 2	dd, dd_crdm, se_nu	UNpolarized *equivalent* WIMP nucleus cross section including form factors
dsdmspin	2, 3, 5, 6, 7, 8	se_nu	dark matter spin
dsidnumberset	1		
dsidnumber	1	dd_crdm	
dsinit_module	2, 3, 4, 5, 6, 7, 8	ini	Intialzation of module
dskdm2simp	2, 5, 6, 7, 8	kd	Scattering amplitude squared, expanded in powers of energy
dskdm2	2, 5, 6, 7, 8	kd	Full scattering amplitude squared, averaged over momentum transfer
dskdparticles	2, 5, 6, 7, 8	kd, rd	Initalization of kinetic decoupling for module
dsmodelsetup	2, 3, 4, 5, 6, 7, 8		Sets up a new particle physics model
dsmwimp	2, 3, 4, 5, 6, 7, 8	cr_aux, cr_axi, dd, dd_crdm, fi, kd, rd, se_nu	Dark matter mass
dsrdparticles	2, 4, 5, 6, 7, 8	fi, rd	Particles included in relic density calculation (coannihilations, resonances and thresholds)
dsseyield	2, 5, 6, 7, 8	ini, se_nu, se_yield	Yields from annihilation in the Sun/Earth
dssigmav0tot	2, 5, 6, 7, 8	cr_axi, se_nu	Total annihilation cross section at $v = 0$
dssisigtm	2, 8	si	Self-interaction momentum-transfer cross section

Table 3.3: Table of interface functions, which modules that contain them (with numbering from table 3.2, where they are used and a short description of them. (Note: this table is automatically generated from scanning through the particle physics modules in src_models. The description is taken from the empty module routine headers.)

Routine	Description	Chapter
dsanyield_sim	Simulated particle yields	6
dsdbdphidtaxi	Local galactic differential antideuteron flux from dark matter	13
dsepdphidpaxi	Local galactic differential positron flux from dark matter	13
dspbdphidtaxi	Local galactic differential antiproton flux from dark matter	13
dscrgaflux_line_v0ann	Flux of monoenergetic gamma-rays from annihilation in the halo, in the limit of zero relative velocity	14
dscrgaflux_v0ann	Flux of gamma-rays from annihilation in the halo, in the limit of zero relative velocity	14
dsgafluxsph	gamma-rays from decay/annihilation from halo specified by halo label	14
dscrmuflux_v0ann	Neutrino-induced muon flux from WIMP annihilation in the halo	15
dsdddrde	Differential WIMP-nucleus recoil rates	17
dsddhelp	help with options for scattering cross sections	17
dsddDMCRcountrate	experimental sensitivity to cosmic-ray uppscattered dark matter	18
dsdfactor	D-factor (l.o.s. integral for decaying DM)	20
dsjfactor	J-factor (l.o.s. integral for annihilating DM)	20
dsdmdselect_halomodel	select between halo models in the halo repository	21
dsdmdset_halomodel	add an existing halo model to halo repository	21
dsfi2to2oh2	Calculate the relic density Ωh^2 from freeze-in	23
dsinit	Initialize DarkSUSY (should always be called)	25
dskdmcut	Cutoff mass in linear power spectrum	26
dskdtkd	Kinetic decoupling temperature	26
dsrdomega	Calculate the relic density Ωh^2 of a dark matter particle	27
dsrdomega_aDM	Calculate the relic density Ωh^2 of an asymmetric	27
dsrdomega_cBE	Calculate the relic density Ωh^2 of a dark matter particle, using coupled Boltzmann equations	27
dssenu_rates	Rates of neutrinos, neutrino-induced leptons and hadronic showers from DM annihilations in the Sun/Earth	30
dssisigtmav	velocity-averaged momentum-transfer cross section for dark matter self-interactions	32

Table 3.4: List of commonly used functions and subroutines. (Note: this table is automatically generated from the routines in src that have the commonly used tag in their header.)

Chapter 4

Comparison to previous DarkSUSY versions

For those DarkSUSY users who are familiar with previous version of the code, the main structural difference introduced with DarkSUSY 6.0 is a highly modular setup that allows to fully disentangle the astrophysics-related parts from those that rely on a specific particle physics model (see Chapter 3 for a detailed description). There are also many new or significantly improved physics capabilities introduced after version 5 and 4 [1] – including electroweak and strong corrections to DM annihilation, improved routines to solve the Boltzmann equation for chemical and kinetic decoupling, and a new framework to handle the propagation of cosmic rays. For a detailed list of those new features, we refer to the publication describing this release of the code [2].

One technical aspect that has changed are the particle codes (k-variables), which are now treated as (module-)internal codes. They can be used by the particle physics module if the module so wishes, but the interface functions and routines in ds_core instead use PDG codes when referring to particles.

In the course of re-organizing the code, it also became necessary to re-name some of the basic functions and subroutines that existed in earlier versions and which users may have become familiar with. For convenience, we therefore list below, in Table 4.1, the most commonly used functionalities in version 4 and 5 that have changed in name or usage with the present version of the code.

routine name up to DarkSUSY 5	new routine name	comments
dshmset	dsdmdset_halomodel	Setting up and refering to DM halos has fundamentally changed. See Section 3.3 for an overview and all Chapters with names containing src_dmd in Part II of the manual for more details.
dshmj	dsjfactor	Line-of-sight integrals now take a halo label as input, and can be computed for various objects at the same time.
dssusy	[dsmodelsetup]	Setting up a model (calculating the mass spectrum, relevant 3-particle vertices etc.) now depends on the particle module implementation.

dsmhtkd dsmhmcut	dskdtkd dskdmcut	The 'microhalo' routines are now more properly referred to as 'kinetic decoupling' routines.
dshmrescale_rho	—	A mismatch between local halo density and DM abundance for a given DarkSUSY module is no longer hidden as a rescaling factor in a common block. Instead, such factors typically enter as explicit parameters in direct and indirect detection routines.
dshaloyield	[dsanyield]	total cosmic ray yield from *neutralino* annihilation
dshayield	dsanyield_sim	simulated cosmic ray yields from individual annihilation/decay channels to SM particles. Note that internal channel codes are now replaced with PDG codes of the final state particles as input.
dshrgacontdiff	dsgafluxsph	Gamma-ray flux routines now work seamlessly together with both the new halo setup and the modular particle physics structure.
dshrgaline	dscrgaflux_line_v0ann dscrgaflux_dec	Gamma-ray line routines now return both number, energies and widths of all such signals that are present in the current particle setup.
dshrpbardiff dshrdbardiff dsepdiff	dspbdphidtaxi dsdbdphidtaxi dsepdphidpaxi	Cosmic-ray propagation routines have been re-written from scratch. They are now much more flexible and can be used for any axisymmetric halo/diffusion model.
dsntrates	dssenu_rates	Neutrino rates from inside the sun or earth
dshrmuhalo	dscrmuflux_v0ann dscrmuflux_v0ann	Neutrino-induced muon flux from the halo (for annihilating and decaying DM, respectively).

Table 4.1: 'Translation table' for how the most commonly used functionalities in DarkSUSY version 5 and earlier have changed with the present version of the code. Routines in parentheses, e.g. [dsanyield], are no longer part of the DarkSUSY main library and only provided by specific particle physics modules. For more detailed descriptions of the new routines and functions, see the headers of the respective files.

Chapter 5

Original articles

A large part of the routines in the present DarkSUSY version have been implemented in the context of original research work. Therefore, when using those routines, please give proper credit not only to the main DarkSUSY paper [1, 2] but also to the relevant articles in the following list:

Relic density – freeze-out P. Gondolo and G. Gelmini, Nucl. Phys. **B360** (1991) 145 [12]; J. Edsjö and P. Gondolo, Phys. Rev. **D56** (1997) 1879 [13]; J. Edsjö, M. Schelke, P. Ullio and P. Gondolo, JCAP **04** (2003) 001 [14]. For coupled Boltzmann equations: T. Binder, T. Bringmann, M. Gustafsson and A. Hryczuk, Eur. Phys. J. **C 81** (2021) 577 [15].

Relic density – freeze-in T. Bringmann, S. Heeba, F. Kahlhoefer and K. Vangsnes, arXiv: 2111.14871 (2021) [16].

Kinetic decoupling and microhalos T. Bringmann, NJP **11** (2009) 10527 [17].

Continuous gamma-rays L. Bergström, J. Edsjö and P. Ullio, Phys. Rev. **D58** (1998) 083507 [18].

Neutrino telescopes L. Bergström, J. Edsjö and P. Gondolo, Phys. Rev. **D58** (1998) 103519 [19].

Positrons E.A. Baltz and J. Edsjö, Phys. Rev. **D59** (1999) 023511 [20].

Antiprotons L. Bergström, J. Edsjö and P. Ullio, ApJ **526** (1999) 215 [21].

QCD corrections to DM annihilation T. Bringmann, A. J. Galea, P. Walia, Phys.Rev. **D93** (2016) 043529 [22].

Cosmic-ray upscattered dark matter T. Bringmann and M. Pospelov, Phys. Rev. Lett. **122** (2019) 17, 171801 [23]; K. Bondarenko *et al.*, JHEP **03** (2020) 118 [24].

General MSSM, direct detection L. Bergström and P. Gondolo, Astrop. Phys. **5** (1996) 263 [25].

Gamma lines from supersymmetric DM L. Bergström and P. Ullio, Nucl. Phys. **B504** (1997) 27 [26]; P. Ullio and L. Bergström, Phys. Rev. **D57** (1998) 1962 [27].

Internal bremsstrahlung in the MSSM (γ and e^+) T. Bringmann, L. Bergström and J. Edsjö, JHEP **0801** (2008) 049 [28]; L. Bergström, T. Bringmann and J. Edsjö, Phys.Rev. **D78** (2008) 103520 [29].

MSSM Electroweak corrections to indirect detection yields T. Bringmann and F. Calore, Phys.Rev.Lett. **112** (2014) 071301 [30]; T. Bringmann, F. Calore, A. J. Galea and M. Garny, JHEP **1709** (2016) 041 [31].

Part II

Main DarkSUSY routines in src/

Chapter 6

an_yield: Annihilation yields in the halo – yields from simulations

6.1 Annihilation in the halo, yields – theory

Here we calculate yields of different particles from annihilation of dark matter particles in the halo.

6.1.1 Monte Carlo simulations

We need to evaluate the yield of different particles per WIMP annihilation. The hadronization and/or decay of the annihilation products are simulated with PYTHIA [10] 6.426. The simulations are done for a set of 18 WIMP masses, m_χ = 10, 25, 50, 80.3, 91.2, 100, 150, 176, 200, 250, 350, 500, 750, 1000, 1500, 2000, 3000 and 5000 GeV. We tabulate the yields and then interpolate these tables in DarkSUSY.

The simulations are here simpler than those for annihilation in the Sun/Earth since we don't have a surrounding medium that can stop the annihilation products. We here simulate for 8 'fundamental' annihilation channels $c\bar{c}$, $b\bar{b}$, $t\bar{t}$, $\tau^+\tau^-$, W^+W^-, Z^0Z^0, gg and $\mu^+\mu^-$. Compared to the simulations in the Earth and the Sun, we now let pions and kaons decay and we also let antineutrons decay to antiprotons. For each mass we simulate 2.5×10^6 annihilations and tabulate the yield of antiprotons, positrons, gamma rays (not the gamma lines), muon neutrinos and neutrino-to-muon conversion rates and the neutrino-induced muon yield, where in the last two cases the neutrino-nucleon interactions has been simulated with PYTHIA as outlined in section 31.1.1

With these simulations, we can calculate the yield for any of these particles for a given particle physics model. In `src`, we only include the channels that are actually simulated. In the different particle physics modules in `src_model`, the summation over all possible final states and possibly more complex channels, like scalars decaying to other particles is done.

Note that simulations are typically done without specifying a particular polarization state of the final state particles. This is however not entirely correct as the possible polarization states will depend on the particle physics model we have.

Even if the simulations are not performed in this more general way yet, we have set up the structure here to eventually provide the yields in this form. In particular the most general routine to calculate the yields from any of the simulated channels is **dsanyield_sim_ls**. Apart from the mass, energy and yield type, this routine also takes are arguments the PDG codes of the final state particles and the polarization state of the final state. In particular, you are required to provide the

quantum numbers

j = the quantum number for the total angular momentum
P = the parity of the final state
l = the quantum number for the orbital angular momentum in the final state
s = the quantum number for the total spin in the final state

We want to move in a direction where this routine is the one that should be used. While getting there, we also provide a simpler routine which just gives the polarization state in terms of helcity and polarization, **dsanyield_sim**. This routine takes the PDG code of the final state and the polarization as

- Left-handed or right-handed polarization for fermions.
- Longitudinal or transverse polarization for vector bosons.

This routine for simplicity assumes that the final state particles have the same polarization state.

6.2 Routine headers – fortran files

dsandec.f

```
*****************************************************************
*** suboutine dsandec decomposes yieldcode yieldk to flyxtype
*** fltype and fi
*****************************************************************

      subroutine dsandec(yieldk,fltyp,fi)
```

dsanifind.f

```
*********************************************************
*** routine to find the index of an entry              ***
*** the closest lowest hit is given                    ***
*** author: joakim edsjo (edsjo@physics.berkeley.edu)  ***
*** date: 98-01-26
*********************************************************

      subroutine dsanifind(value,array,ipl,ii,imin,imax)
```

dsanreadfiles.f

```
*****************************************************************************
***   subroutine dsanreadfiles loads (from disk) the the requested data
***   files. yieldk is
***   the yield type (51,52 or 53 (or 151, 152, 153)) for
***   positron yields, cont. gamma or muon neutrino yields respectively.
***   yieldk is used to check that the provided data file is of the
***   correct type.  if yieldk=51,52 or 53 integrated yields are loaded and
***   if yieldk =151, 152 or 153, differential yields are loaded.
***   Author: Joakim Edsjo
***   edsjo@fysik.su.se date: 96-10-23 (based on dsmuinit.f version
***   3.21)
***   modified: 98-01-26
***   modified: 09-10-20 pat scott pat@fysik.su.se
***   modified: 04-05-09 Joakim Edsjo, edsjo@fysik.su.se
***   modified: April, 2016, Joakim Edsjo, to read new simulation yields
***     (including dbars)
*****************************************************************************

      subroutine dsanreadfiles(yieldk)
```

dsanyield_contrib_A.f

```
************************************************************************
*** function dsanyield_contrib_A returns the yield above threshold
*** (or differential at that energy, dN/dE). As dsanyield_sim but based
*** on the tables provided by
***
*** S. Amoroso et al. arXiv: 1812.07424 [hep-ph], JCAP05(2019)007
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Amoroso'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e     = kinetic energy where the yield is calculated (in GeV)
***   - pdg   = PDG code of one of the annihilation final state particle.
***             The following channels are available:
***
***              pdg   Channel
***             -----  -------
***                1   d d-bar
***                2   u u-bar
***                3   s s-bar
***                4   c c-bar
***                5   b b-bar
***                6   t t-bar
***               11   e- e+
***               13   mu- mu+
***               15   tau- tau+
***               21   gluon gluon
***               23   Z0 Z0
***               24   W+ W-
***               25   h h
***
***   - yieldpdg =  PDG code for the yield type; currently the following is available:
***
***          yieldpg       yield type
***          -------       ----------------
***          22            cont. photons (gamma rays)
***          -11           positrons
***          12 or -12     nu_e *and* nu_e-bar
***          14 or -14     nu_mu *and* nu_mu-bar
***          16 or -16     nu_tau *and* nu_tau-bar
***
***    - var = 1..12, referring to the variation of fragmentation and shower evolution
***                   parameters, as specified in 1812.07424. Currently only
***                   var = 1 (central prediction for the spectra) is available.
***
***    - diff: currently only differential yield at egev (diff=1) is available
***
***
*** Output:
***   - istat = 0 if no warnings/errors are reported
***           = 1 if the requested channel is not simulated
***           = 2 if mwimp is below the lowest simulated mass
***                               or above the highest simulated mass
***           = 3 if (none of the above,but) e is below the lowest simulated energy
***
***   - dsanyield_sim, yield in units of
***                     - (annihilation)**-1          for diff = 0
***                     - gev**-1 (annihilation)**-1  for diff = 1
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
```

```
*** Date: 2020-05-08
**************************************************************************

      real*8 function dsanyield_contrib_A(mwimp,e,pdg,yieldpdg,var,diff,istat)
```

dsanyield_contrib_A_read.f

```
**************************************************************************
*** subroutine dsanyield_contrib_A_read reads in yield tables provided by
***
*** S. Amoroso et al. arXiv: 1812.07424 [hep-ph], JCAP05(2019)007
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Amoroso'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2020-05-25
**************************************************************************

      subroutine dsanyield_contrib_A_read()
```

dsanyield_contrib_B.f

```
**************************************************************************
*** function dsanyield_contrib_B returns the yield above threshold
*** (or differential at that energy, dN/dE). As dsanyield_sim but based
*** on the tables provided by
***
*** C. W. Bauer, N. L. Rodd and B. R. Webber, JHEP 06 (2021) 121 [arXiv:2007.15001]
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Bauer'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e     = *kinetic* energy where the yield is calculated (in GeV)
***   - pdg   = PDG code of one of the annihilation final state particle.
***             The following channels are available:
***
***              pdg   Channel
***             -----  -------
***                1   d d-bar
***                2   u u-bar
***                3   s s-bar
***                4   c c-bar
***                5   b b-bar
***                6   t t-bar
***               11   e- e+
***               12   nue nuebar
***               13   mu- mu+
***               14   numu numubar
***               15   tau- tau+
***               16   nutau nutaubar
***               21   gluon gluon
***               23   Z0 Z0
***               24   W+ W-
***               25   h h
***
***              NB: pdg=1,2,4 return the *same* yield =(d+u+s)/3
***
```

```
***   - yieldpdg =  PDG code for the yield type; currently the following is available:
***
***          yieldpg       yield type
***          -------       ----------------
***          22            cont. photons (gamma rays)
***          -11           positrons
***          -2212         antiprotons
***          12 or -12     nu_e *and* nu_e-bar
***          14 or -14     nu_mu *and* nu_mu-bar
***          16 or -16     nu_tau *and* nu_tau-bar
***
***    - diff: currently only differential yield at egev (diff=1) is available
***
***
*** Output:
***   - istat = 0 if no warnings/errors are reported
***           = 1 if the requested channel is not simulated
***           = 2 if mwimp is below the lowest simulated mass
***                            or above the highest simulated mass
***           = 3 if (none of the above,but) e is below the lowest simulated energy
***
***   - dsanyield_sim, yield in units of
***                    - (annihilation)**-1          for diff = 0
***                    - gev**-1 (annihilation)**-1  for diff = 1
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-01-26
**************************************************************************

      real*8 function dsanyield_contrib_B(mwimp,e,pdg,yieldpdg,diff,istat)
```

dsanyield_contrib_B_read.f

```
**************************************************************************
*** subroutine dsanyield_contrib_B_read reads in yield tables provided by
***
*** C. W. Bauer, N. L. Rodd and B. R. Webber, JHEP 06 (2021) 121 [arXiv:2007.15001]
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Bauer'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-01-26
**************************************************************************

      subroutine dsanyield_contrib_B_read()
```

dsanyield_contrib_H.f

```
**************************************************************************
*** function dsanyield_contrib_H returns the yield above threshold
*** (or differential at that energy, dN/dE). As dsanyield_sim but based
*** on yields returns by Hazma
***
*** A. Coogan et al. arXiv: 2207.07634 [hep-ph]
*** (with special thanks to F. Kahlhoefer for producing the tables
***  underlying the DarkSUSY implementation)
***
*** for (sub-) GeV dark matter.
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Hazma'). See also that
*** subroutine for various options that can be set for these tables.
```

```
*** Please remember to cite the above reference on top of the DarkSUSY
*** article if you use these yield tables.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e     = kinetic energy where the yield is calculated (in GeV)
***   - pdg   = PDG code of one of the annihilation final state particle.
***             The following channels are available:
***
***              pdg   Channel
***             -----  -------
***               11   e- e+
***               13   mu- mu+
***               21   hadrons
***
***             NOTE: While channel 21 is reserved for "gluon gluon" final
***                   states in other yield implementations, it here refers
***                   to the cumulative contribution from all hadronic final
***                   states (with relative branching fractions as in model
***                   with U(1) kinetic mixing).
***
***   - yieldpdg =  PDG code for the yield type; currently the following is available:
***
***          yieldpg       yield type
***          -------       ----------------
***          22            cont. photons (gamma rays)
***          -11           positrons
***          12 or -12     nu_e *and* nu_e-bar
***          14 or -14     nu_mu *and* nu_mu-bar
***
***    - diff: currently only differential yield at egev (diff=1) is available
***
***
*** Output:
***   - istat = 0 if no warnings/errors are reported
***           = 1 if the requested channel is not simulated
***           = 2 if mwimp is below the lowest simulated mass
***                            or above the highest simulated mass
***           = 3 if (none of the above,but) e is below the lowest simulated energy
***
***   - dsanyield_sim, yield in units of
***                    - (annihilation)**-1          for diff = 0
***                    - gev**-1 (annihilation)**-1  for diff = 1
***
*** NB: positron yield from e- e+ channel is zero because the monochromatic
***     positron "line" is handled separately (in dscrsource_line rather than
***     dscrsource). Monochromatic photon lines are included in the hadron
***     channel, though.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-12-06
**************************************************************************

      real*8 function dsanyield_contrib_H(mwimp,e,pdg,yieldpdg,diff,istat)
```

dsanyield_contrib_H_read.f

```
**************************************************************************
*** subroutine dsanyield_contrib_H_read reads in yield tables based on Hazma
***
*** A. Coogan et al. arXiv: 2207.07634 [hep-ph]
*** (with special thanks to F. Kahlhoefer for producing the tables
***  underlying the DarkSUSY implementation)
***
*** In order to activiate these yield tables instead of the default ones
```

```
*** you need to call dsanyield_set('yieldtables','Hazma'). See also that
*** subroutine for various options than can be set for these tables.
*** Please remember to cite the above reference on top of the DarkSUSY
*** article if you use these yield tables.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-12-06
************************************************************************

      subroutine dsanyield_contrib_H_read()
```

dsanyield_contrib_J.f

```
************************************************************************
*** function dsanyield_contrib_J returns the yield above threshold
*** (or differential at that energy, dN/dE). As dsanyield_sim but based
*** on the tables provided by
***
*** A. Jueid et al. arXiv: 2202.11546 [hep-ph]
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Jueid'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e     = kinetic energy where the yield is calculated (in GeV)
***   - pdg   = PDG code of one of the annihilation final state particle.
***             The following channels are available:
***
***              pdg   Channel
***             -----  -------
***                1   d d-bar
***                2   u u-bar
***                3   s s-bar
***                4   c c-bar
***                5   b b-bar
***                6   t t-bar
***               11   e- e+
***               13   mu- mu+
***               15   tau- tau+
***               21   gluon gluon
***               22   gamma gamma
***               23   Z0 Z0
***               24   W+ W-
***               25   h h
***
***   - yieldpdg =  PDG code for the yield type; currently the following is available:
***
***          yieldpg       yield type
***          -------       ----------------
***          22            cont. photons (gamma rays)
***          -11           positrons
***          -2212         antiprotons
***          12 or -12     nu_e *and* nu_e-bar
***          14 or -14     nu_mu *and* nu_mu-bar
***          16 or -16     nu_tau *and* nu_tau-bar
***
***    - var = 1..12, referring to the variation of fragmentation and shower evolution
***                   parameters, as specified in 2202.11546. Currently only
***                   var = 1 (central prediction for the spectra) is available.
***
***    - diff: currently only differential yield at egev (diff=1) is available
***
```

```
***
*** Output:
***   - istat = 0 if no warnings/errors are reported
***           = 1 if the requested channel is not simulated
***           = 2 if mwimp is below the lowest simulated mass
***                            or above the highest simulated mass
***           = 3 if (none of the above,but) e is below the lowest simulated energy
***
***   - dsanyield_sim, yield in units of
***                    - (annihilation)**-1          for diff = 0
***                    - gev**-1 (annihilation)**-1  for diff = 1
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-03-03
************************************************************************

      real*8 function dsanyield_contrib_J(mwimp,e,pdg,yieldpdg,var,diff,istat)
```

dsanyield_contrib_J_read.f

```
************************************************************************
*** subroutine dsanyield_contrib_J_read reads in yield tables provided by
***
*** A. Jueid et al. arXiv: 2202.11546 [hep-ph]
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Jueid'). Please
*** remember to cite the above reference on top of the DarkSUSY paper
*** if you use these yield tables.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-03-03
************************************************************************

      subroutine dsanyield_contrib_J_read()
```

dsanyield_contrib_P.f

```
************************************************************************
*** function dsanyield_contrib_P returns the yield above threshold
*** (or differential at that energy, dN/dE). As dsanyield_sim but based
*** on the tables provided by
***
*** T. Plehn et al. arXiv: 1911.11147 [hep-ph]
***
*** for LIGHT QUARK FINAL STATES from (sub-) GeV dark matter.
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Plehn'). See also that
*** subroutine for various options that can be set for these tables.
*** Please remember to cite the above reference on top of the DarkSUSY
*** article if you use these yield tables.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e     = kinetic energy where the yield is calculated (in GeV)
***   - pdg   = PDG code of one of the annihilation final state particle.
***             The following channels are available:
***
***              pdg   Channel
***             -----  -------
***                1   d d-bar
***                2   u u-bar
***                3   s s-bar
***
```

```
***             NOTE: The output for each of these spectra is identical, and set to
***                   1/3 of the total yield in the specific underlying model that
***                   is assumed. See dsanyield_set for available models (default
***                   is 'B-L').
***
***   - yieldpdg =  PDG code for the yield type; currently the following is available:
***
***          yieldpg       yield type
***          -------       ----------------
***          22            cont. photons (gamma rays)
***          -11           positrons
***          -2212         antiprotons
***          12 or -12     nu_e *and* nu_e-bar
***          14 or -14     nu_mu *and* nu_mu-bar
***          16 or -16     nu_tau *and* nu_tau-bar
***
***    - diff: currently only differential yield at egev (diff=1) is available
***
***
*** Output:
***   - istat = 0 if no warnings/errors are reported
***           = 1 if the requested channel is not simulated
***           = 2 if mwimp is below the lowest simulated mass
***                            or above the highest simulated mass
***           = 3 if (none of the above,but) e is below the lowest simulated energy
***
***   - dsanyield_sim, yield in units of
***                     - (annihilation)**-1          for diff = 0
***                     - gev**-1 (annihilation)**-1  for diff = 1
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2020-06-12
**********************************************************************

      real*8 function dsanyield_contrib_P(mwimp,e,pdg,yieldpdg,diff,istat)
```

dsanyield_contrib_P_read.f

```
**********************************************************************
*** subroutine dsanyield_contrib_P_read reads in yield tables based on
***
*** T. Plehn et al. arXiv: 1911.11147 [hep-ph]
***
*** In order to activiate these yield tables instead of the default ones
*** you need to call dsanyield_set('yieldtables','Plehn'). See also that
*** subroutine for various options than can be set for these tables.
*** Please remember to cite the above reference on top of the DarkSUSY
*** article if you use these yield tables.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2020-06-15
**********************************************************************

      subroutine dsanyield_contrib_P_read()
```

dsanyield_dbset.f

```
****************************************************************
*** suboutine dsanyield_dbset sets options for antideuteron
*** yields
*** Input:
***    yieldk = 59: old spherical coalescence model (should only
***                 be used for comparison, obsolete
***             61: Pythia 6 MC results (default)
```

```
***             62: Pythia 8 MC results (not yet implemented)
***             63: Herwig++ MC results (not yet implemented)
***              0: default, i.e. the same as 61
***           -100: do not change yieldk
***    Note: integrated or differential yields are chosen
***    explicitly when calling dsanyield_sim.
***
***    p0bar: coalescence momentum chosen. This is the *deviation*
***           from the best fit value from Aleph data (for MC yields)
***           Hence, the used coalescence momentum is given by
***               p0 = p0_best-fit + p0bar*sigma_p0
***           p0bar is hence the dimenionless number of sigma away (+ or -)
***           from the best fit value. The best fit value is determined
***           from comparisons with Aleph measurements and is determined
***           for each MC separately. sigma_p0 is defined as
***           p0_high - p0_best-fit for p0>p0_best-fit and
***           p0_best-fit-p0_low for p0<p0_best-fit
***           Default: 0.0 (i.e. best fit p0 will be used)
***           If -100.d0, p0bar is not changed in this routine.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: April, 2016
*****************************************************************

      subroutine dsanyield_dbset(yieldk,p0bar)
```

dsanyield_init.f

```
*****************************************************************************
***   subroutine dsaninit initializes and loads (from disk) the common
***   block variables needed by the other halo yield routines.
***   author: joakim edsjo
***   edsjo@physto.se date: 96-10-23 (based on dsmuinit.f version
***   3.21)
***   modified: 98-01-26
***   modified: 09-10-20 pat scott pat@fysik.su.se
***   modified: May, 2014, Joakim Edsjo
***   modified: April, 2016, Joakim Edsjo, to read new simulations tables
***     (including dbars)
*****************************************************************************

      subroutine dsanyield_init
```

dsanyield_set.f

```
      recursive subroutine dsanyield_set(key,value)
c...  desc : Set parameters for yield routines
c...  c - character string specifying choice to be made
c...  author: torsten bringmann 2020-06-12
```

dsanyield_sim.f

```
*****************************************************************************
*** function dsanyield_sim calculates the yield above threshold
*** (or differential at that energy) for the requested annihilation channel
*** and the fluxtype given by yieldpdg. This routine assumes that annihilation
*** takes place to two final state particles where the second is the
*** antiparticle of the first. For polarized states, the polarization
*** is assumed to be the same for both the particles.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e = kinetic energy where the yield is calculated (in GeV)
```

```
***         Note: e is kinetic energy per particle (i.e. not per nucleon for
***         anti-deuterons)
***
***   - pdg = pdg codes of annihilation final state particle.
***   Only the pdg code of the first particle is given, the second one
***   is assumed to have pdg code -pdg.
***   Only channels for which simulation data from Pythia simulations exist
***   are available here. More complex channels
***   like channels containing Higgs bosons etc that decay to standard model
***   particles are treated in each respective particle physics module
***   in src_model.
***   The currently implemented channels are
***
***      pdg  Channel        Internal number (only listed temporarily)
***     ----  -------        ---------------
***        1  d d-bar        1
***        2  u u-bar        2
***        3  s s-bar        3
***        4  c c-bar        4
***        5  b b-bar        5
***        6  t t-bar        6
***       13  mu- mu+        10
***       15  tau- tau+      11
***       21  gluon gluon    7
***       23  Z0 Z0          9
***       24  W+ W-          8
***       25  hh
***
***   Note: If a channel that is not simulated is asked for, the yield
***   0 is returned and a warning is issued (istat bit 3 set)
***
***   hel: helicity state of final state. Currently, only states where the
***        two final state particles are in the same helicity state (e.g.
***        both left, both right, both transverse, both longitudinal)
***        is included.
***   Possible values for hel:
***     'L': left-polarized fermion for fermions, longitudinal for vectors
***     'R': right-polarized fermion for fermions
***     'T': transverse for vectors
***     'O': unpolarized yields (unphysical, but included for comaprison)
***
***   - yieldpdg, PDG code for the yield type; currently the following is implemented:
***
***           yieldpg       yield type
***           -------       ----------------
***           22            cont. gamma rays
***           -11           positrons
***           -2212         antiprotons
***           -1000010020   anti-deuteron
***           111           pi0
***           12 or -12     nu_e and nu_e-bar
***           14 or -14     nu_mu and nu_mu-bar
***           16 or -16     nu_tau and nu_tau-bar
***           130072        muons from nu at creation
***           130073        muons from nu, as seen by a detector in ice
***                         (i.e. integrating 130072 over the mean muon path)
***
***    - diff: dictates whether differential source term at egev (diff=1)
***            or integrated source term above egev (diff=0) is returned
***
***
*** Output:
***   - istat (=0 if no warnings/errors are reported)
***     bit 0 is set if extrapolations below the lowest simulated mass
***       or above the highest simulated mass is needed
```

```
***     bit 3 is set if the requested channel is not simulated,
***       yield zero is returned
***     bit 4 is set if the requested polarization state is not available,
***       the yield of the simulated polarization yield (typically
***       unpolarized) is returned instead
***   - dsanyield_sim, yield in units of
***   units: (annihilation)**-1  integrated
***   units: gev**-1 (annihilation)**-1 differential
***
*** If this routine is called outside of the kinematical regions
*** where tables exist, the following is done:
***   - for lower energies, than the lowest simulated ones,
***     extrapolations are used
***   - for masses below the lowest simulated, extrapolations
***     are used
***   - for energies above the highest simulated, the results
***     for the highest energy simulated are used
***
*** Note: at initialization of DarkSUSY, dsanyield_init should be called
*** to initialize these routines (done automatically in dsinit). This is
*** only needed once per run.
***
***
***   type : commonly used
***   desc : Simulated particle yields
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Modifications: 2010-11-05 (JE): better extrapolation below lowest
***   simulated energy (now dN/dz is taken as constant below)
*** mod tb -- added FSR from e+e-, linked light quarks to cc (prelim fix)
*** mod tb -- added pdg yield codes
*** mod je -- new simulation yields (including light quarks and anti-deuterons)
***           and internal channel numbers
*** mod tb -- added tables by Amoroso et al [1812.07424]
***                          Plehn et al [1911.11147]
***                          Bauer et al [2007.15001]
***                          Hazma [2207.07634]
**************************************************************************

      real*8 function dsanyield_sim(mwimp,e,pdg,hel,yieldpdg,diff,istat)
```

dsanyield_sim_ls.f

```
**************************************************************************
*** NOTE: This routine is not fully functional yet, but shows what we intend
*** to have eventually. The goal is to have a much more general structure
*** regarding which channels and polarization states that are available.
*** This routine should eventually be able to return the yield for different
*** final state particles and polarization states, expressed as the final
*** state's quantum numbers j,P,l and s.
***
*** function dsanyield_sim_ls calculates the yield above threshold
*** (or differential at that energy) for the requested annihilation channel
*** and the fluxtype given by yieldpdg. This routine assumes that annihilation
*** takes place to two final state particles.
***
*** Inputs:
***   - mwimp = WIMP mass in GeV
***   - e = kinetic energy where the yield is calculated (in GeV)
***
***   - pdg1, pdg2 = pdg codes of annihilation final state particles.
***   Only channels for which simulation data from Pythia simulations exist
***   are available here. More complex channels
***   like channels containing Higgs bosons etc that decay to standard model
***   particles are treated in each respective particle physics module
```

```
***   in src_model.
***   The currently implemented channels are
***
***     pdg1  pdg2  Channel        Internal number (only listed temporarily (old new))
***     ----  ----  -------        ---------------
***        1    -1  d d-bar        -   1
***        2    -2  u u-bar        -   2
***        3    -3  s s-bar        -   3
***        4    -4  c c-bar        1   4
***        5    -5  b b-bar        2   5
***        6    -6  t t-bar        3   6
***       15   -15  tau- tau+      4  11
***       24   -24  W+ W-          5   8
***       23    23  Z0 Z0          6   9
***       13   -13  mu- mu+        7  10
***       21    21  gluon gluon    8   7
***
***   Note: If a channel that is not simulated is asked for, the yield
***   0 is returned and a warning is issues (istat bit 3 set)
***
***   For the final state polarization, we need a few arguments to
***   describe it fully.
***     twoj: total angular momentum quantum number of final state particles
***         times 2.
***     p: parity quantum number:
***     twol: orbital angular momentum quantum number of final state times 2
***     twos: spin quantum number of final state times 2
***
***   - yieldpdg, PDG code for the yield type; currently the following is implemented:
***
***           yieldpg       yield type
***           -------       ----------------
***           22            cont. gamma rays
***           -11           positrons
***           -2212         antiprotons
***           -1000010020   anti-deuteron
***           111           pi0
***           12 or -12     nu_e and nu_e-bar
***           14 or -14     nu_mu and nu_mu-bar
***           16 or -16     nu_tau and nu_tau-bar
***           130072        muons from nu at creation
***           130073        muons from nu, as seen by a detector in ice
***                         (i.e. integrating 130072 over the mean muon path)
***
***    - diff: dictates whether differential source term at egev (diff=1)
***            or integrated source term above egev (diff=0) is returned
***
***
*** Output:
***   - istat (=0 if no warnings/errors are reported)
***     bit 0 is set if extrapolations below the lowest simulated mass
***       or above the highest simulated mass is needed
***     bit 3 is set if the requested channel is not simulated,
***       yield zero is returned
***     bit 4 is set if the requested polarization state is not available,
***       the yield of the simulated polarization yield (typically
***       unpolarized) is returned instead
***   - dsanyield_sim_ls, yield in units of
***   units: (annihilation)**-1  integrated
***   units: gev**-1 (annihilation)**1 differential
***
*** If this routine is called outside of the kinematical regions
*** where tables exist, the following is done:
***   - for lower energies, than the lowest simulated ones,
***     extrapolations are used
```

```
***   - for masses below the lowest simulated, extrapolations
***     are used
***   - for energies above the highest simulated, the results
***     for the highest energy simulated are used
***
*** Note: at initialization of DarkSUSY, dsaninit should be called
*** to initialize these routines (done automatically in dsinit). This is
*** only needed once per run.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Modifications: 2010-11-05 (JE): better extrapolation below lowest
***   simulated energy (now dN/dz is taken as constant below)
*** mod tb -- added pdg yield codes
************************************************************************

      real*8 function dsanyield_sim_ls(mwimp,e,pdg1,pdg2,twoj,p,
     &  twol,twos,yieldpdg,diff,istat)
```

dsanyieldget.f

```
************************************************************************
*** function dsanyieldget gives the information in the differential and
*** integrated arrays phiit and phidiff for given array indices.
*** compared to getting the results directly from the array, this
*** routine performs smoothing if requested.
*** the smoothing is controlled by the parameter ansmooth in the
*** following manner.
*** ansmooth = 0 - no smoothing
***            1 - smoothing of zi-1,zi and zi+1 bins if z>.3
***            2 - smoothing of zi-2,zi-1,zi,zi+1 and zi+2 if z>0.3
*** Author: Joakim Edsjo, edsjo@fysik.su.se
************************************************************************

      real*8 function dsanyieldget(zi,mxi,ch,fi,ftype,istat)
```

dsanyieldgetdiff.f

```
************************************************************************
*** function dsanyieldgetdiff returns the differential yields for the given
*** choice of arguments. Compared to reading the arrays directly, this
*** routine interpolates in the coalescence momenta for dbars.
*** The routine is called by dsanyieldget.f that also smoothes the arrays.
*** The coalescence momentum used is determined by dbp0bar in common block.
*** If zero, the best-fit p0 dbp0fit will be used. If non-zero, it indicates
*** the deviation in units of the 1sigma uncertainty on p0.
*** For the actual p0 values, they are in MeV and tabulations exist
*** up to 300 MeV.
*** kind is used (currently) for dbar yields. 1=yield, 2=error on yield
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: November, 2014
************************************************************************

      real*8 function dsanyieldgetdiff(zi,mxi,ch,fi,kind)
```

dsanyieldgetint.f

```
************************************************************************
*** function dsanyieldgetint returns the integrated yields for the given
*** choice of arguments. Compared to reading the arrays directly, this
*** routine interpolates in the coalescence momenta for dbars.
*** The routine is called by dsanyieldget.f that also smoothes the arrays.
*** The coalescence momentum used is determined by dbp0bar in common block.
*** If zero, the best-fit p0 dbp0fit will be used. If non-zero, it indicates
```

```
*** the deviation in units of the 1sigma uncertainty on p0.
*** For the actual p0 values, they are in MeV and tabulations exist
*** up to 300 MeV.
*** kind is used (currently) for dbar yields. 1=yield, 2=error on yield
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: November, 2014
**************************************************************************

      real*8 function dsanyieldgetint(zi,mxi,ch,fi,kind)
```

Chapter 7

aux: General routines

7.1 General routines

In **aux/**, we collect routines that are of general interest to many other routines in DarkSUSY. E.g., we have routines to find elements in arrays (used for interpolation), Bessel functions, error functions, spline routines, etc.

7.2 Routine headers – fortran files

dsabsq.f

```
c-----------------------------------------------------------------------
c  abs squared of a complex*16 number.
c  called by dwdcos.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
      function dsabsq(z)
```

dsbessei0.f

```
**********************************************************************
      real*8 function dsbessei0(x)
**********************************************************************
c     exp(-|x|) i_0(x)
```

dsbessei1.f

```
      real*8 function dsbessei1(x)
c     exp(-|x|) i1(x)
```

dsbessek0.f

```
**********************************************************************
*** function dsbessek0 returns the value of the modified bessel       ***
*** function of the second kind of order 0 times exp(x)             ***
*** works for positive real x                                       ***
*** coefficients from abramovitz and stegun.                        ***
*** e-mail: edsjo@physto.se                                  ***
*** date: 98-04-29                                                  ***
**********************************************************************
```

```
      real*8 function dsbessek0(x)
```

dsbessek1.f

```
************************************************************************
*** function dsbessek1 returns the value of the modified bessel       ***
*** function of the second kind of order 1 times exp(x).            ***
*** works for positive real x                                       ***
*** coefficients from abramovitz and stegun.                        ***
*** e-mail: edsjo@physto.se                                   ***
*** date: 98-04-29                                                  ***
************************************************************************

      real*8 function dsbessek1(x)
```

dsbessek2.f

```
************************************************************************
*** function bessk2 returns the value of the modified bessel        ***
*** function of the second kind of order 2 times exp(x)             ***
*** works for positive real x                                       ***
*** recurrence relation                                             ***
*** e-mail: gondolo@mppmu.mpg.de                                    ***
*** date: 00-07-07                                                  ***
************************************************************************

      real*8 function dsbessek2(x)
```

dsbessjw.f

```
c....Various bessel functions
c....from P. Ullio

      real*8 function dsbessjw(n,x)
```

dscharadd.f

```
************************************************************************
*** dscharadd takes a string str, adds a string add to it
*** and returns the concatenated string with spaces removed.
*** Author: Joakim Edsjo, edsjo@physto.se
*** Date: 2004-01-19
************************************************************************

      subroutine dscharadd(str,add)
```

dsdatafile.f

```
************************************************************************
*** subroutine dsdatafile generates a filename for files stored
*** in the data/ dirctory in the DarkSUSY root.
*** This routine is useful to get absolute filenames to be used
*** on a given system.
*** Output: filename, full absolute path to data file
*** Input: filename_in (name of file in data/)
************************************************************************
      subroutine dsdatafile(filename_out,filename_in)
```

dsdilog.f

```
c=====================================================================
c
c   dsdilogarithm function
c   argument should be between -1 and 1
c
c   author: lars bergstrom (lbe@physto.se)
c
c---------------------------------------------------------------------

      real*8 function dsdilog(x)
```

dsf_int.f

```
      real*8 function dsf_int(f,a,b,eps)
c-----------------------------------------------------------------------
c  integrate function f between a and b
c  input
c    integration limits a and b
c  called by different routines
c  author: joakim edsjo (edsjo@physto.se) 96-05-16
c          2000-07-19 paolo gondolo added eps as argument
c          2007-07-27 pat scott added j > 6 condition to prevent
c           spurious early convergence, removing os initial
c           definition.
c          2016-11-17, Joakim Edsjo, added error handling to stop if
c           integrand is NaN
c  based on paolo gondolos wxint.f routine.
c=======================================================================
```

dsf_int2.f

```
      real*8 function dsf_int2(f,a,b,eps)
c-----------------------------------------------------------------------
c  integrate function f between a and b
c  input
c    integration limits a and b
c  called by different routines
c  author: joakim edsjo (edsjo@physto.se) 96-05-16
c          2000-07-19 paolo gondolo added eps as argument
c          2007-07-27 pat scott added j > 6 condition to prevent
c           spurious early convergence, removing os initial
c           definition.
c  based on paolo gondolos wxint.f routine.
c  the same routine as dsf_int but used when double integration is needed.
c=======================================================================
```

dsfac.f

```
**********************************************************************
*** returns the factorial of an integer, based on NUMERICAL RECIPES
***
*** torsten bringmann (troms@physto.se), 2008-12-03
*** Modified: Joakim Edsjo, edsjo@fysik.su.se, 2011-05-25
**********************************************************************

      REAL*8 FUNCTION dsfac(n)
```

dsgamma.f

```
!=====================================================================
```

```
! Calculates gamma(x) to double precision.
!     gamma(x) = \int_0^\infty dt t^{x-1} e^{-t}
! Returns a large negative number where gamma is undefined
! (x=0,-1,-2,...).
!
!
!    Created by Chris Savage (savage@fysik.su.se)
!    2011/05/23
!
!=======================================================================
!
      REAL*8 FUNCTION dsgamma(x)
```

dshealpixint.f

```
*****************************************************************************
*** Subroutine dshealpixint integrates a function f(l,b) on the unit sphere ***
*** by means of pixelization in the HEALpix scheme, where l and b are       ***
*** longitude and latitude in rad, respectively. The integration is         ***
*** optimized for functions f centered on (l,b)=0.0d0 (but not necessarily  ***
*** spheically symmetric).                                                  ***
***                                                                         ***
***  Input:                                                                 ***
***    f      - function to be integrated over; must be declared EXTERNAL   ***
***             in calling main program / subroutine                        ***
***    nmin   - HEALPIX levels to start integration (at least 5 recomm.)    ***
***    nmax   - HEALPIX level where integration is stopped even if accuracy ***
***             goal is not met                                             ***
***    eps    - relative accuracy goal                                     ***
***    epsabs - absolute accuracy goal                                     ***
***                                                                         ***
***  Output:                                                                ***
***    res   - integration result                                          ***
***    niter - number of iterations (HEALPIX levels) actually used         ***
***    ierr  - 1 if nmax is reachd before accuracy goal, else 0            ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no (based on previous DS version)     ***
*** date 2018-11-28                                                         ***
*****************************************************************************
      subroutine dshealpixint(f,nmin,nmax,eps,epsabs,res,niter,ierr)
```

dshiprecint3.f

```
      subroutine dshiprecint3(fun,lowlim,upplim,result)
No header found.
```

dshunt.f

```
************************************************************************
      subroutine dshunt(xx,n,x,indx)
*** returns the lowest index indx for which x>xx(indx).
*** if x<= xx(i) it returns 0,
*** if x>xx(n) it returns indx=n
************************************************************************
```

dsi_trim.f

```
      function dsi_trim(s)
No header found.
```

dsidtag.f

```
      character*20 function dsidtag()
No header found.
```

dsinterpolatetable.f

```
************************************************************************
*** Function dsInterpolateTable takes tabulated values of a real function,  ***
*** f(x_i), and returns an interpolated value for f(x).                     ***
*** NB: This routine assumes that the x-values are ordered as x_1<x_2<... ! ***
***                                                                         ***
***  Input:                                                                 ***
***    x        - independent variable                                      ***
***    functab  - Array containing f(x_i)                                   ***
***    xtab     - 1D array containing x_i                                   ***
***    Npoints  - size of functab, xtab                                     ***
***    tabID    - integer from 1-10                                         ***
***    how      - linear interpolation (how=1)                              ***
***               log-interpolation  (how=2)                                ***
***                                                                         ***
***  Output: f(x)                                                           ***
***                                                                         ***
***  Note: 'tabID' does not affect the result, but increases performance in ***
***        case of subsequent calls to dsInterpolateTable with similar      ***
***        values of x. If dsInterpolateTable is simultaneously used on     ***
***        different tables, each of those tables should be assigned a      ***
***        unique (and different) tabID.                                    ***
***        A call with tabID<0 will enforce a "re-set" of this label, to    ***
***        to ensure that the behaviour is identical to the first function  ***
***        call with abs(tabID).                                            ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2020-10-25                                                         ***
************************************************************************
      real*8 function dsInterpolateTable(x,functab,xtab,Npoints,tabID,how)
```

dsinterpolatetable2D.f

```
************************************************************************
*** Function dsInterpolateTable2D takes tabulated values of a 2D real       ***
*** function, f(x_i,y_j), and returns an interpolated value  for f(x,y).    ***
***                                                                         ***
***  Input:                                                                 ***
***    x, y     - independent variables                                     ***
***    functab  - Array containing f(x_i,y_j)                               ***
***    Nx, Ny   - size of functab (Nx x Ny)                                 ***
***    Nymax    - 1D array containing the number of relevant bins in y,     ***
***               for a given value of x_i  (length: Nx)                    ***
***    xtab     - 1D array containing x_i  (length: Nx)                     ***
***    ytab     - 2D array containing y_ij (length: Nx x Ny)                ***
***    tabID    - integer from 1-15 (see below)                             ***
***    how      - linear interpolation (how=1)                              ***
***               log-interpolation  (how=2)                                ***
***                                                                         ***
***  Output: f(x,y)                                                         ***
***                                                                         ***
*** ASSUMPTIONS about input values (NOT tested at runtime, to increase      ***
*** performance!)                                                           ***
***     - xtab is ordered as x(1) < x(2) < ... < x(Nx)                      ***
***     - for each value of x(i), there are at least a few tabulated values ***
***       of y, with y(i,1) < y(i,2) < ... < y(i,Nymax(i))                  ***
***     - Nymax(i) <= Ny                                                    ***
***     - any values for f(x_i,y_j), as well as y(i,j),                     ***
```

```
***       with y_ij > y(i,Nymax(i)) can be disregarded                      ***
***       (as indicated, Nymax(i) can be chosen arbitrary -- but            ***
***        performance is best for a uniform Nymax(i) = Ny)                 ***
***                                                                         ***
*** (Typical usage would make use of a wrapper function, creating a         ***
***  table satisfying the above assumptions. For an example, see            ***
***  dsanyield_contrib_A.                                                   ***
***  WARNING: The overhead required to use the routine in this way comes    ***
***           with significant numerical costs. For frequent calls (e.g.    ***
***           inside integrands) it is much better to copy the code below   ***
***           and adapt it by hand to the concrete table to be interpolated)***
***                                                                         ***
***  Note: 'tabID' does not affect the result, but increases performance    ***
***        in case of subsequent calls to dsInterpolateTable with similar   ***
***        values of x. If dsInterpolateTable is simultaneously used on     ***
***        different tables, each of those tables should be assigned a      ***
***        unique (and different) tabID.                                    ***
***        A call with tabID<0 will enforce a "re-set" of this label, to    ***
***        to ensure that the behaviour is identical to the first function  ***
***        call with abs(tabID).                                            ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2020-05-26                                                         ***
*******************************************************************************
      real*8 function dsInterpolateTable2D(x,y,functab,Nx,Ny,Nymax,xtab,ytab,
     &                                     tabID,how)
```

dsisnan.f

```
************************************************************************
*** function to check if a real*8 number is NaN, returns true if it is
************************************************************************
      logical function dsisnan(a)
```

dslowcase.f

```
      subroutine dslowcase(a)
No header found.
```

dsmoderf.f

```
      real*8 function dsmoderf(x)
c ================================================================
c error function
c modified by l. bergstrom 98-09-15
c modified by p. gondolo 2000-07-19
c see test output below
c used for damour-krauss calculations
c ================================================================
```

dsrnd1.f

```
      function dsrnd1(idum)
c-----------------------------------------------------------------------
c  uniform deviate between 0 and 1.
c  input:
c    idum - seed (integer); enter negative integer at first call
c=======================================================================
```

dsrndlin.f

```
      function dsrndlin(idum,a,b)
```

```
No header found.
```

dsrndlog.f

```
      function dsrndlog(idum,a,b)
No header found.
```

dsrndsgn.f

```
      function dsrndsgn(idum)
No header found.
```

dsspline.f

```
      SUBROUTINE dsspline(x,y,n,yp1,ypn,y2)
c spline routine, double precision
```

dssplint.f

```
      SUBROUTINE dssplint(xa,ya,y2a,n,x,y)
c spline routine, double precision
```

dswrite.f

```
      subroutine dswrite(level,opt,message)
c-----------------------------------------------------------------------
c  handle writing onto standard output in darksusy
c  input:
c    level - print message if prtlevel is >= level
c    opt   - print (1) the model tag or not (0)
c    message - string containing the text to print
c  common:
c    'dsio.h' - i/o units numbers and prtlevel
c  author: paolo gondolo 1999
c=======================================================================
```

dszarg.f

```
**********************************************************************
*** dszarg gives the argument (or phase) of a complex number [radians].
**********************************************************************
      real*8 function dszarg(z)
```

erfinv.f

```
! NSWC function erfinv
! US Naval Surface Warfare Center Mathematical Library
! http://www.ualberta.ca/CNS/RESEARCH/Software/NumericalNSWC/site.html
! Apparently no license, so anything goes.  Probably they prefer you don't use it
! to design neutralino missiles.
! Added to DarkSUSY by Pat Scott (p.scott@imperial.ac.uk)
! Oct 11 2014

      function erfinv (p, q)
!
!**************************************************************************
!
!! ERFINV: evaluation of the inverse error function
!
!     for 0 <= p <= 1,  w = erfinv(p,q) where erf(w) = p. it is
```

```
!     assumed that q = 1 - p. if either inequality on p is violated
!     or p + q /= 1, then erfinv(p,q) is set to a negative value.
!
!
!     reference. mathematics of computation,oct.1976,pp.827-830.
!                  j.m.blair,c.a.edwards,j.h.johnson
!
```

lngamma.f

```
!====================================================================
! Calculates ln(gamma(x)) to double precision.
!     gamma(x) = \int_0^\infty dt t^{x-1} e^{-t}
! The argument must be positive.
!
!
!    Created by Chris Savage (savage@fysik.su.se)
!    2011/05/23
!
!====================================================================
!
! Calculated using Lanczos' approximation.  Valid only for x > 0.
!
! Lanczos's approximation:
!   Gamma(z+1) = (z+r+1/2)^{z+1/2} * EXP[-(z+r+1/2)] * sqrt(2*pi)
!                * (B_0 + \Sum_{k=0}^{N} B_k/(z+k) + eps)
! where r is some constant s.t. z+r+1/2 > 0.
! Coefficients in the series expansion are dependent on r and the
! number of terms at which the series is truncated.  The determination
! of these coefficients is too complicated to show here.
!
! Note the formula above is valid for complex cases as well, as long as
! Re(x) > 1.  The following identity also applies for the complex case:
!   Gamma(1-z) = pi*z / Gamma(1+z) / sin(pi*z)
!
! The coefficients are calculated as described in Section 6.7 of
! Glendon Pugh's thesis (2004), which provides an extensive discussion
! of the Lanczos approximation.  The coefficients are given for the
! following modified form of the formula:
!   Gamma(z+1) = 2 * sqrt(e/pi) * ((z+r+1/2)/e)^(z+1/2)
!                * (D_0 + \Sum_{k=0}^{N} D_k/(z+k) + eps)
! The quantities N and r are chosen from Table C.1 to achieve the
! desired precision:
!   Single precision:  N=4,  r=4.340882
!   Double precision:  N=10, r=10.900511  (see Table 8.5)
!   Quad precision:    N=21, r=22.618910  (see Table 9.4)
!
      REAL*8 FUNCTION lngamma(x)
```

modcosint.f

```
!Returns modified cosine integral MCi(x) = 1/x^2 * (Ci(x) - lnx - EULER), along with 1/x^2 * Si(x)
!Adapted from Numerical Recipes.
!For x < 0 the routine returns MCi(-x) and the user must supply the -i pi/x^2 him/herself
!si returns a very large value for x=0, instead of crashing or outputting Inf
!Author: Pat Scott (p.scott@imperial.ac.uk)
!Date: Some time in 2010/11.
!Added to DarkSUSY: Oct 11 2014

      subroutine ModCosInt(x,ci,si)
```

safelog1m.f

```
      !Calculates ln(1-x) with a Taylor expansion if x < 1e-3, otherwise uses the F90 native ln function

      double precision function safelog1m(x)
```

sinc.f

```
!====================================================================
! Calculates sinc(x) = sin(x)/x, to double precision.
!
! Author: Pat Scott (p.scott@imperial.ac.uk)
! Date: Some time in 2010/11.
! Added to DarkSUSY: Oct 13 2014
!
!====================================================================


      double precision function sinc(x)
```

zeroin.f

```
! Root finding by Brent's method.
! Routine courtesy of Netlib GO (Golden Oldies)
! http://www.netlib.org/go/
! Apparently no license, so anything goes.
! Added to DarkSUSY by Pat Scott (p.scott@imperial.ac.uk)
! Oct 11 2014
!
c  To get d1mach, mail netlib
c       send d1mach from core
      real*8 function zeroin(ax,bx,f,tol)
```

dshealpixave.F

```
**************************************************************************
*** Subroutine dshealpixave sums over the value of EXTERNAL function f(l,b) ***
*** at the center of all HEALPIX pixels of level n, divided by their size.  ***
*** longitude and latitude in rad, respectively. The integration is         ***
***                                                                        ***
***  Input:                                                                ***
***    f   - EXTERNAL function of longitude l and latitude b (both in rad)  ***
***    n   - HEALPIX level                                                 ***
***    how - sum over all pixels at level n (how=0), or only those that    ***
***          are neighbouring or inside those that gave a non-zero value   ***
***          at the revious call (How=1; this assmues that the previous    ***
***          call was done with HEALPIX level n-1                          ***
***                                                                        ***
***  Output:                                                               ***
***    ave   - average value of f on unit sphere                           ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-11-28                                                        ***
**************************************************************************
      subroutine dshealpixave(f,n,how,ave)
#include "dscontribstat.F"
#if defined HEALPIX_INSTALLATION && HEALPIX_INSTALLATION == succeeded
```

Chapter 8

aux_xcernlib: CERN routines needed by DarkSUSY

8.1 Routine headers – fortran files

bsir364.f

```
*
* $Id: bsir364.F,v 1.1.1.1 1996/04/01 15:02:07 mclareni Exp $
*
* $Log: bsir364.F,v $
* Revision 1.1.1.1  1996/04/01 15:02:07  mclareni
* Mathlib gen
*
*
      FUNCTION DBSIR3(X,NU)
```

dbzejy.f

```
      SUBROUTINE DBZEJY(A,N,MODE,REL,X)

C     Computes the first n positive (in the case Jo'(x) the first n
C     non-negative) zeros of the Bessel functions
C               Ja(x), Ya(x), Ja'(x), Ya'(x),
C     where a >= 0 and ' = d/dx.
C
C     Based on Algol procedures published in
C
C     N.M. TEMME, An algorithm with Algol 60 program for the compu-
C     tation of the zeros of ordinary Bessel functions and those of
C     their derivatives, J. Comput. Phys. 32 (1979) 270-279, and
C
C     N.M. TEMME, On the numerical evaluation of the ordinary Bessel
C     function of the second kind, J. Comput. Phys. 21 (1976) 343-350.
```

ddilog.f

```
CDECK  ID>, DDILOG.
      DOUBLE PRECISION FUNCTION DDILOG(X)
C
C          FROM CERN PROGRAM LIBRARY
C
```

dgadap.f

```
c#########################################################################
c
c   one- and two-dimensional adaptive gaussian integration routines.
c
c ************************************************************************

      subroutine dgadap(a0,b0,f,eps0,sum)
c
c   purpose           - integrate a function f(x)
c   method            - adaptive gaussian
c   usage             - call gadap(a0,b0,f,eps,sum)
c   parameters  a0    - lower limit (input,real)
c               b0    - upper limit (input,real)
c               f     - function f(x) to be integrated. must be
c                       supplied by the user. (input,real function)
c               eps0  - desired relative accuracy. if sum is small eps
c                       will be absolute accuracy instead. (input,real)
c               sum   - calculated value for the integral (output,real)
c   precision         - single (see below)
c   req'd prog's      - f
c   author            - t. johansson, lund univ. computer center, 1973
c   reference(s)      - the australian computer journal,3 p.126 aug. -71
c
c  made real*8 by j. edsjo 97-01-17
```

dgadap2.f

```
c#########################################################################
c
c   one- and two-dimensional adaptive gaussian integration routines.
c
c ************************************************************************

      subroutine dgadap2(a0,b0,f,func2,eps0,sum)
**********************************************************************
**********************************************************************
c CLONE of dgadap, with external func2 as additional input
c Instead of f(x), this routine thus integrates f(func2,x)
c
c  Author: torsten.bringmann@fys.uio.no, 2018-04-26
**********************************************************************
**********************************************************************
c
c   purpose           - integrate a function f(x)
c   method            - adaptive gaussian
c   usage             - call gadap(a0,b0,f,eps,sum)
c   parameters  a0    - lower limit (input,real)
c               b0    - upper limit (input,real)
c               f     - function f(x) to be integrated. must be
c                       supplied by the user. (input,real function)
c               eps0  - desired relative accuracy. if sum is small eps
c                       will be absolute accuracy instead. (input,real)
c               sum   - calculated value for the integral (output,real)
c   precision         - single (see below)
c   req'd prog's      - f
c   author            - t. johansson, lund univ. computer center, 1973
c   reference(s)      - the australian computer journal,3 p.126 aug. -71
c
c  made real*8 by j. edsjo 97-01-17
```

drkstp.f

```
      SUBROUTINE DRKSTP(N,H,X,Y,SUB,W)
No header found.
```

dstred2.f

```
CDECK  ID>, DSTRED2.
      SUBROUTINE DSTRED2(NM,N,A,D,E,Z)
C     FROM CERN PROGRAM LIBRARY
```

eisrs1.f

```
CDECK  ID>, EISRS1.
      SUBROUTINE EISRS1(NM,N,AR,WR,ZR,IERR,WORK)
C     ALL EIGENVALUES AND CORRESPONDING EIGENVECTORS OF A REAL
C     SYMMETRIC MATRIX
C     FROM CERN PROGRAM LIBRARY
C
```

gpindp.f

```
      real*8 function gpindp(a,b,epsin,epsout,func,iop)
c
c     parameters
c
c     a       = lower boundary
c     b       = upper boundary
c     epsin   = accuracy required for the approxination
c     epsout  = improved error estimate for the approximation
c     func    = function routine for the function func(x).to be de-
c               clared external in the calling routine
c     iop     = option parameter , iop=1 , modified romberg algorithm,
c                                          ordinary case
c                                  iop=2 , modified romberg algorithm,
c                                          cosine transformed case
c                                  iop=3 , modified clenshaw-curtis al
c                                          gorithm
c
c     parameters in common block / gpint /
c
c     tend    = upper bound for value of integral
c     umid    = lower bound for value of integralc
c     n       = the number of integrand values used in the calculation
c     line    = line no in romberg table (related to n through
c               n-1=2**(line-1) , applicable only for iop=1 or 2)
c     iout    = element no in line (applicable only for iop=1 or 2)
c     jop     = option parameter , jop=0 , no printing of intermediate
c                                          calculations
c                                  jop=1 , print intermediate calcula-
c                                          tions
c     kop     = option parameter , kop=0 , no time estimate
c                                  kop=1 , estimate time
c     t       = time used for calculation in msec.
c
c     integration parameters
c
c     nupper  = 9 , corresponds to 1024 sub-intervals for the unfolded
c               integral.the max.no of function evaluations thus beeing
c               1025.the highest end-point approximation is thus using
c               1024 intervals while the highest mid-point approxima-
c               tion is using 512 intervals.
c
```

```
c     input/output parameters
c
```

gpindp2.f

```
      real*8 function gpindp2(a,b,epsin,epsout,func,func2,iop)
*********************************************************************
*********************************************************************
c CLONE of gpindp, with external func2 as additional input
c Instead of func(x), this routine thus integrates func(func2,x)
c
c  Author: torsten.bringmann@fys.uio.no, 2018-04-26
*********************************************************************
*********************************************************************
c
c     parameters
c
c     a       = lower boundary
c     b       = upper boundary
c     epsin   = accuracy required for the approxination
c     epsout  = improved error estimate for the approximation
c     func    = function routine for the function func(x).to be de-
c               clared external in the calling routine
c     iop     = option parameter , iop=1 , modified romberg algorithm,
c                                          ordinary case
c                                  iop=2 , modified romberg algorithm,
c                                          cosine transformed case
c                                  iop=3 , modified clenshaw-curtis al
c                                          gorithm
c
c     parameters in common block / gpint /
c
c     tend    = upper bound for value of integral
c     umid    = lower bound for value of integralc
c     n       = the number of integrand values used in the calculation
c     line    = line no in romberg table (related to n through
c               n-1=2**(line-1) , applicable only for iop=1 or 2)
c     iout    = element no in line (applicable only for iop=1 or 2)
c     jop     = option parameter , jop=0 , no printing of intermediate
c                                          calculations
c                                  jop=1 , print intermediate calcula-
c                                          tions
c     kop     = option parameter , kop=0 , no time estimate
c                                  kop=1 , estimate time
c     t       = time used for calculation in msec.
c
c     integration parameters
c
c     nupper  = 9 , corresponds to 1024 sub-intervals for the unfolded
c               integral.the max.no of function evaluations thus beeing
c               1025.the highest end-point approximation is thus using
c               1024 intervals while the highest mid-point approxima-
c               tion is using 512 intervals.
c
c     input/output parameters
c
```

mtlprt.f

```
*
* Dummy routine to easily integrate bsir364.f with DarkSUSY
* Author: Joakim Edsjo, edsjo@physto.se
* Date: September 13, 2000
*
```

```
      SUBROUTINE MTLPRT(NAME,ERC,TEXT)
```

tql2.f

```
CDECK  ID>, TQL2.
      SUBROUTINE TQL2(NM,N,D,E,Z,IERR)
C     FROM CERN PROGRAM LIBRARY
```

wgamma.f

```
c Function wgamma to calculate complex gamma function in double precision.
c Taken from CERN program library.
c
c Added by torsten.bringmann@fys.uio.no (as well as all functions it depends on)

*
* $Id: cgamma64.F,v 1.1.1.1 1996/04/01 15:01:55 mclareni Exp $
*
* $Log: cgamma64.F,v $
* Revision 1.1.1.1  1996/04/01 15:01:55  mclareni
* Mathlib gen
*
*
      FUNCTION WGAMMA(Z)
*
* $Id: imp64.inc,v 1.1.1.1 1996/04/01 15:02:59 mclareni Exp $
*
* $Log: imp64.inc,v $
* Revision 1.1.1.1  1996/04/01 15:02:59  mclareni
* Mathlib gen
*
*
* imp64.inc
*
```

wlgama.f

```
c Function wlgama to calculate lograithm of complex gamma function in double precision.
c Taken from CERN program library.
c
c Added by torsten.bringmann@fys.uio.no (as well as all functions it depends on)


*
* $Id: clogam64.F,v 1.1.1.1 1996/04/01 15:01:55 mclareni Exp $
*
* $Log: clogam64.F,v $
* Revision 1.1.1.1  1996/04/01 15:01:55  mclareni
* Mathlib gen
*
*
      FUNCTION WLGAMA(Z)
*
* $Id: imp64.inc,v 1.1.1.1 1996/04/01 15:02:59 mclareni Exp $
*
* $Log: imp64.inc,v $
* Revision 1.1.1.1  1996/04/01 15:02:59  mclareni
* Mathlib gen
*
*
* imp64.inc
*
```

Chapter 9

aux_xdiag: Diagonalization routines

9.1 XDIAG

This folder contains a set of routines to perform numerical diagonalization of complex matrices. Even though analytical routines can be used for up to 5×5 matrics, they tend to be numerically unstable for matrices occuring in some particle physics models (for example supersymmetry). Hence, we use numerical routines to perform the diagonalization instead.

9.2 Routine headers – fortran files

DSCEigensystem.f

```
* CEigensystem.F
* diagonalization of a complex n-by-n matrix using the Jacobi algorithm
* code adapted from the "Handbook" routines for complex A
* (Wilkinson, Reinsch: Handbook for Automatic Computation, p. 202)
* this file is part of the Diag library
* last modified 27 Sep 07 th
* adapted to darksusy 04 June 08 pg
*
**********************************************************************
** CEigensystem diagonalizes a general complex n-by-n matrix.
** Input: n, A = n-by-n matrix
** Output: d = vector of eigenvalues, U = transformation matrix
** these fulfill diag(d) = U A U^-1.
*
      subroutine CEigensystem(n, A,ldA, d, U,ldU, sort)
```

DSHEigensystem.f

```
* HEigensystem.F
* diagonalization of a Hermitian n-by-n matrix using the Jacobi algorithm
* code adapted from the "Handbook" routines for complex A
* (Wilkinson, Reinsch: Handbook for Automatic Computation, p. 202)
* this file is part of the Diag library
* last modified 27 Sep 07 th
* adapted to darksusy 04 June 08 pg
*
**********************************************************************
** HEigensystem diagonalizes a Hermitian n-by-n matrix.
```

```
** Input: n, A = n-by-n matrix, Hermitian
** (only the upper triangle of A needs to be filled).
** Output: d = vector of eigenvalues, U = transformation matrix
** these fulfill diag(d) = U A U^+ = U A U^-1 with U unitary.
*
      subroutine HEigensystem(n, A,ldA, d, U,ldU, sort)
```

DSSEigensystem.f

```
* SEigensystem.F
* diagonalization of a complex symmetric n-by-n matrix using
* the Jacobi algorithm
* code adapted from the "Handbook" routines for complex A
* (Wilkinson, Reinsch: Handbook for Automatic Computation, p. 202)
* this file is part of the Diag library
* last modified 27 Sep 07 th
* adapted to darksusy 04 Jun 08 pg
*
************************************************************************
** SEigensystem diagonalizes a complex symmetric n-by-n matrix.
** Input: n, A = n-by-n matrix, complex symmetric
** (only the upper triangle of A needs to be filled).
** Output: d = vector of eigenvalues, U = transformation matrix
** these fulfill diag(d) = U A U^T = U A U^-1 with U U^T = 1.
*
      subroutine SEigensystem(n, A,ldA, d, U,ldU, sort)
```

DSSVD.f

```
* SVD.F
* singular value decomposition of an m-by-n matrix
* this file is part of the Diag library
* last modified 27 Sep 07 th
* adapted to darksusy 04 June 08 pg
*
************************************************************************
** SVD performs a singular value decomposition.
** Input: m, n, A = m-by-n matrix.
** Output: d = nm-vector of singular values,
** V = nm-by-m left transformation matrix,
** W = nm-by-n right transformation matrix, nm = min(m, n),
** these fulfill diag(d) = V^* A W^+.
*
      subroutine SVD(m, n, Ao,ldA, d, Vo,ldV, Wo,ldW, sort)
```

DSTakagiFactor.f

```
* TakagiFactor.F
* computes the Takagi factorization of a complex symmetric matrix
* code adapted from the "Handbook" routines
* (Wilkinson, Reinsch: Handbook for Automatic Computation, p. 202)
* this file is part of the Diag library
* last modified 27 Sep 07 th
* adapted to darksusy 04 June 08 pg
*
************************************************************************
** TakagiFactor factorizes a complex symmetric n-by-n matrix
** Input: n, A = n-by-n matrix, complex symmetric
** (only the upper triangle of A needs to be filled).
** Output: d = vector of diagonal values, U = transformation matrix
** these fulfill diag(d) = U^* A U^+ with U unitary.
*
      subroutine TakagiFactor(n, A,ldA, d, U,ldU, sort)
```

Chapter 10

aux_xnswclib: src/aux_xnswclib

10.1 Routine headers – fortran files

dsnswc.f

```
C ***********************************************
C Full nswc lib in /internal. IN this file, keep only routines
C currently needed by DS
C created on 02/02/2021 by TB
C ***********************************************
      INTEGER FUNCTION IPMPAR (I)
C-----------------------------------------------------------------------
C
C     IPMPAR PROVIDES THE INTEGER MACHINE CONSTANTS FOR THE COMPUTER
C     THAT IS USED. IT IS ASSUMED THAT THE ARGUMENT I IS AN INTEGER
C     HAVING ONE OF THE VALUES 1-10. IPMPAR(I) HAS THE VALUE ...
C
C  INTEGERS.
C
C     ASSUME INTEGERS ARE REPRESENTED IN THE N-DIGIT, BASE-A FORM
C
C               SIGN ( X(N-1)*A**(N-1) + ... + X(1)*A + X(0) )
C
C               WHERE 0 .LE. X(I) .LT. A FOR I=0,...,N-1.
C
C     IPMPAR(1) = A, THE BASE.
C
C     IPMPAR(2) = N, THE NUMBER OF BASE-A DIGITS.
C
C     IPMPAR(3) = A**N - 1, THE LARGEST MAGNITUDE.
C
C  FLOATING-POINT NUMBERS.
C
C     IT IS ASSUMED THAT THE SINGLE AND DOUBLE PRECISION FLOATING
C     POINT ARITHMETICS HAVE THE SAME BASE, SAY B, AND THAT THE
C     NONZERO NUMBERS ARE REPRESENTED IN THE FORM
C
C               SIGN (B**E) * (X(1)/B + ... + X(M)/B**M)
C
C               WHERE X(I) = 0,1,...,B-1 FOR I=1,...,M,
C               X(1) .GE. 1, AND EMIN .LE. E .LE. EMAX.
C
C     IPMPAR(4) = B, THE BASE.
C
```

```
C  SINGLE-PRECISION
C
C     IPMPAR(5) = M, THE NUMBER OF BASE-B DIGITS.
C
C     IPMPAR(6) = EMIN, THE SMALLEST EXPONENT E.
C
C     IPMPAR(7) = EMAX, THE LARGEST EXPONENT E.
C
C  DOUBLE-PRECISION
C
C     IPMPAR(8) = M, THE NUMBER OF BASE-B DIGITS.
C
C     IPMPAR(9) = EMIN, THE SMALLEST EXPONENT E.
C
C     IPMPAR(10) = EMAX, THE LARGEST EXPONENT E.
C
C-----------------------------------------------------------------------
C
C     TO DEFINE THIS FUNCTION FOR THE COMPUTER BEING USED, ACTIVATE
C     THE DATA STATMENTS FOR THE COMPUTER BY REMOVING THE C FROM
C     COLUMN 1. (ALL THE OTHER DATA STATEMENTS SHOULD HAVE C IN
C     COLUMN 1.)
C
C     IF DATA STATEMENTS ARE NOT GIVEN FOR THE COMPUTER BEING USED,
C     THEN THE FORTRAN MANUAL FOR THE COMPUTER NORMALLY GIVES THE
C     CONSTANTS IPMPAR(1), IPMPAR(2), AND IPMPAR(3) FOR THE INTEGER
C     ARITHMETIC. HOWEVER, HELP MAY BE NEEDED TO OBTAIN THE CONSTANTS
C     IPMPAR(4),...,IPMPAR(10) FOR THE SINGLE AND DOUBLE PRECISION
C     ARITHMETICS. THE SUBROUTINES MACH AND RADIX ARE PROVIDED FOR
C     THIS PURPOSE.
C
C-----------------------------------------------------------------------
C
C     IPMPAR IS AN ADAPTATION OF THE FUNCTION I1MACH, WRITTEN BY
C     P.A. FOX, A.D. HALL, AND N.L. SCHRYER (BELL LABORATORIES).
C     IPMPAR WAS FORMED BY A.H. MORRIS (NSWC). THE CONSTANTS ARE
C     FROM BELL LABORATORIES, NSWC, AND OTHER SOURCES.
C
C-----------------------------------------------------------------------
```

Chapter 11

aux_xquadpack: CMLIB routines needed by DarkSUSY

11.1 Routine headers – fortran files

d1mach.f

```
      real*8 function d1mach(i)
```

dqagp.f

```
      subroutine dqagp2(f,func2,a,b,npts2,points,epsabs,epsrel,result,abserr,
     *   neval,ier,leniw,lenw,last,iwork,work)
************************************************************
************************************************************
c CLONE of dqagp, with external func2 as additional input
c Instead of f(x), this routine thus integrates f(func2,x)
c
c  Author: torsten.bringmann@fys.uio.no, 2018-04-27
************************************************************
************************************************************
c***begin prologue  dqagp
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a2a1
c***keywords  automatic integrator, general-purpose,
c             singularities at user specified points,
c             extrapolation, globally adaptive
c***author  piessens,robert,appl. math. & progr. div - k.u.leuven
c           de doncker,elise,appl. math. & progr. div. - k.u.leuven
c***purpose  the routine calculates an approximation result to a given
c            definite integral i = integral of f over (a,b),
c            hopefully satisfying following claim for accuracy
c            break points of the integration interval, where local
c            difficulties of the integrand may occur (e.g.
c            singularities, discontinuities), are provided by the user.
c***description
c
c        computation of a definite integral
c        standard fortran subroutine
c        double precision version
```

```
c
c        parameters
c         on entry
c            f      - double precision
c                     function subprogram defining the integrand
c                     function f(x). the actual name for f needs to be
c                     declared e x t e r n a l in the driver program.
c
c            a      - double precision
c                     lower limit of integration
c
c            b      - double precision
c                     upper limit of integration
c
c            npts2  - integer
c                     number equal to two more than the number of
c                     user-supplied break points within the integration
c                     range, npts.ge.2.
c                     if npts2.lt.2, the routine will end with ier = 6.
c
c            points - double precision
c                     vector of dimension npts2, the first (npts2-2)
c                     elements of which are the user provided break
c                     points. if these points do not constitute an
c                     ascending sequence there will be an automatic
c                     sorting.
c
c            epsabs - double precision
c                     absolute accuracy requested
c            epsrel - double precision
c                     relative accuracy requested
c                     if  epsabs.le.0
c                     and epsrel.lt.max(50*rel.mach.acc.,0.5d-28),
c                     the routine will end with ier = 6.
c
c         on return
c            result - double precision
c                     approximation to the integral
c
c            abserr - double precision
c                     estimate of the modulus of the absolute error,
c                     which should equal or exceed abs(i-result)
c
c            neval  - integer
c                     number of integrand evaluations
c
c            ier    - integer
c                     ier = 0 normal and reliable termination of the
c                             routine. it is assumed that the requested
c                             accuracy has been achieved.
c                     ier.gt.0 abnormal termination of the routine.
c                             the estimates for integral and error are
c                             less reliable. it is assumed that the
c                             requested accuracy has not been achieved.
c            error messages
c                     ier = 1 maximum number of subdivisions allowed
c                             has been achieved. one can allow more
c                             subdivisions by increasing the value of
c                             limit (and taking the according dimension
c                             adjustments into account). however, if
c                             this yields no improvement it is advised
c                             to analyze the integrand in order to
c                             determine the integration difficulties. if
c                             the position of a local difficulty can be
c                             determined (i.e. singularity,
```

```
c                           discontinuity within the interval), it
c                           should be supplied to the routine as an
c                           element of the vector points. if necessary
c                           an appropriate special-purpose integrator
c                           must be used, which is designed for
c                           handling the type of difficulty involved.
c                       = 2 the occurrence of roundoff error is
c                           detected, which prevents the requested
c                           tolerance from being achieved.
c                           the error may be under-estimated.
c                       = 3 extremely bad integrand behaviour occurs
c                           at some points of the integration
c                           interval.
c                       = 4 the algorithm does not converge.
c                           roundoff error is detected in the
c                           extrapolation table.
c                           it is presumed that the requested
c                           tolerance cannot be achieved, and that
c                           the returned result is the best which
c                           can be obtained.
c                       = 5 the integral is probably divergent, or
c                           slowly convergent. it must be noted that
c                           divergence can occur with any other value
c                           of ier.gt.0.
c                       = 6 the input is invalid because
c                           npts2.lt.2 or
c                           break points are specified outside
c                           the integration range or
c                           (epsabs.le.0 and
c                            epsrel.lt.max(50*rel.mach.acc.,0.5d-28))
c                           result, abserr, neval, last are set to
c                           zero. exept when leniw or lenw or npts2 is
c                           invalid, iwork(1), iwork(limit+1),
c                           work(limit*2+1) and work(limit*3+1)
c                           are set to zero.
c                           work(1) is set to a and work(limit+1)
c                           to b (where limit = (leniw-npts2)/2).
c
c         dimensioning parameters
c            leniw - integer
c                    dimensioning parameter for iwork
c                    leniw determines limit = (leniw-npts2)/2,
c                    which is the maximum number of subintervals in the
c                    partition of the given integration interval (a,b),
c                    leniw.ge.(3*npts2-2).
c                    if leniw.lt.(3*npts2-2), the routine will end with
c                    ier = 6.
c
c            lenw  - integer
c                    dimensioning parameter for work
c                    lenw must be at least leniw*2-npts2.
c                    if lenw.lt.leniw*2-npts2, the routine will end
c                    with ier = 6.
c
c            last  - integer
c                    on return, last equals the number of subintervals
c                    produced in the subdivision process, which
c                    determines the number of significant elements
c                    actually in the work arrays.
c
c         work arrays
c            iwork - integer
c                    vector of dimension at least leniw. on return,
c                    the first k elements of which contain
c                    pointers to the error estimates over the
```

```
c                   subintervals, such that work(limit*3+iwork(1)),...,
c                   work(limit*3+iwork(k)) form a decreasing
c                   sequence, with k = last if last.le.(limit/2+2), and
c                   k = limit+1-last otherwise
c                   iwork(limit+1), ...,iwork(limit+last) contain the
c                    subdivision levels of the subintervals, i.e.
c                    if (aa,bb) is a subinterval of (p1,p2)
c                    where p1 as well as p2 is a user-provided
c                    break point or integration limit, then (aa,bb) has
c                    level l if abs(bb-aa) = abs(p2-p1)*2**(-l),
c                   iwork(limit*2+1), ..., iwork(limit*2+npts2) have
c                    no significance for the user,
c                   note that limit = (leniw-npts2)/2.
c
c            work  - double precision
c                   vector of dimension at least lenw
c                   on return
c                   work(1), ..., work(last) contain the left
c                    end points of the subintervals in the
c                    partition of (a,b),
c                   work(limit+1), ..., work(limit+last) contain
c                    the right end points,
c                   work(limit*2+1), ..., work(limit*2+last) contain
c                    the integral approximations over the subintervals,
c                   work(limit*3+1), ..., work(limit*3+last)
c                    contain the corresponding error estimates,
c                   work(limit*4+1), ..., work(limit*4+npts2)
c                    contain the integration limits and the
c                    break points sorted in an ascending sequence.
c                   note that limit = (leniw-npts2)/2.
c
c***references  (none)
c***routines called  dqagpe,xerror
c***end prologue  dqagp
c
```

dqagse.f

```
* ====================================================================
* nist guide to available math software.
* fullsource for module dqagse from package cmlib.
* retrieved from camsun on wed oct  8 08:26:30 1997.
* ====================================================================
      subroutine dqagse(f,a,b,epsabs,epsrel,limit,result,abserr,neval,
     1   ier,alist,blist,rlist,elist,iord,last)
c***begin prologue  dqagse
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a1a1
c***keywords  (end point) singularities,automatic integrator,
c             extrapolation,general-purpose,globally adaptive
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  the routine calculates an approximation result to a given
c            definite integral i = integral of f over (a,b),
c            hopefully satisfying following claim for accuracy
c            abs(i-result).le.max(epsabs,epsrel*abs(i)).
c***description
c
c        computation of a definite integral
c        standard fortran subroutine
c        real*8 version
c
```

```
c        parameters
c         on entry
c            f      - real*8
c                     function subprogram defining the integrand
c                     function f(x). the actual name for f needs to be
c                     declared e x t e r n a l in the driver program.
c
c            a      - real*8
c                     lower limit of integration
c
c            b      - real*8
c                     upper limit of integration
c
c            epsabs - real*8
c                     absolute accuracy requested
c            epsrel - real*8
c                     relative accuracy requested
c                     if  epsabs.le.0
c                     and epsrel.lt.max(50*rel.mach.acc.,0.5d-28),
c                     the routine will end with ier = 6.
c
c            limit  - integer
c                     gives an upperbound on the number of subintervals
c                     in the partition of (a,b)
c
c         on return
c            result - real*8
c                     approximation to the integral
c
c            abserr - real*8
c                     estimate of the modulus of the absolute error,
c                     which should equal or exceed abs(i-result)
c
c            neval  - integer
c                     number of integrand evaluations
c
c            ier    - integer
c                     ier = 0 normal and reliable termination of the
c                             routine. it is assumed that the requested
c                             accuracy has been achieved.
c                     ier.gt.0 abnormal termination of the routine
c                             the estimates for integral and error are
c                             less reliable. it is assumed that the
c                             requested accuracy has not been achieved.
c            error messages
c                         = 1 maximum number of subdivisions allowed
c                             has been achieved. one can allow more sub-
c                             divisions by increasing the value of limit
c                             (and taking the according dimension
c                             adjustments into account). however, if
c                             this yields no improvement it is advised
c                             to analyze the integrand in order to
c                             determine the integration difficulties. if
c                             the position of a local difficulty can be
c                             determined (e.g. singularity,
c                             discontinuity within the interval) one
c                             will probably gain from splitting up the
c                             interval at this point and calling the
c                             integrator on the subranges. if possible,
c                             an appropriate special-purpose integrator
c                             should be used, which is designed for
c                             handling the type of difficulty involved.
c                         = 2 the occurrence of roundoff error is detec-
c                             ted, which prevents the requested
c                             tolerance from being achieved.
```

```
c                            the error may be under-estimated.
c                        = 3 extremely bad integrand behaviour
c                            occurs at some points of the integration
c                            interval.
c                        = 4 the algorithm does not converge.
c                            roundoff error is detected in the
c                            extrapolation table.
c                            it is presumed that the requested
c                            tolerance cannot be achieved, and that the
c                            returned result is the best which can be
c                            obtained.
c                        = 5 the integral is probably divergent, or
c                            slowly convergent. it must be noted that
c                            divergence can occur with any other value
c                            of ier.
c                        = 6 the input is invalid, because
c                            epsabs.le.0 and
c                            epsrel.lt.max(50*rel.mach.acc.,0.5d-28).
c                            result, abserr, neval, last, rlist(1),
c                            iord(1) and elist(1) are set to zero.
c                            alist(1) and blist(1) are set to a and b
c                            respectively.
c
c            alist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the left end points
c                     of the subintervals in the partition of the
c                     given integration range (a,b)
c
c            blist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the right end points
c                     of the subintervals in the partition of the given
c                     integration range (a,b)
c
c            rlist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the integral
c                     approximations on the subintervals
c
c            elist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the moduli of the
c                     absolute error estimates on the subintervals
c
c            iord   - integer
c                     vector of dimension at least limit, the first k
c                     elements of which are pointers to the
c                     error estimates over the subintervals,
c                     such that elist(iord(1)), ..., elist(iord(k))
c                     form a decreasing sequence, with k = last
c                     if last.le.(limit/2+2), and k = limit+1-last
c                     otherwise
c
c            last   - integer
c                     number of subintervals actually produced in the
c                     subdivision process
c***references  (none)
c***routines called  d1mach,dqelg,dqk21,dqpsrt
c***end prologue  dqagse
c
```

dqagseb.f

```
* ====================================================================
```

```
* nist guide to available math software.
* fullsource for module dqagse from package cmlib.
* retrieved from camsun on wed oct  8 08:26:30 1997.
* ====================================================================
      subroutine dqagseb(f,a,b,epsabs,epsrel,limit,result,abserr,neval,
     1   ier,alist,blist,rlist,elist,iord,last)
c***begin prologue  dqagse
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a1a1
c***keywords  (end point) singularities,automatic integrator,
c             extrapolation,general-purpose,globally adaptive
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  the routine calculates an approximation result to a given
c            definite integral i = integral of f over (a,b),
c            hopefully satisfying following claim for accuracy
c            abs(i-result).le.max(epsabs,epsrel*abs(i)).
c***description
c
c        computation of a definite integral
c        standard fortran subroutine
c        real*8 version
c
c        parameters
c         on entry
c            f      - real*8
c                     function subprogram defining the integrand
c                     function f(x). the actual name for f needs to be
c                     declared e x t e r n a l in the driver program.
c
c            a      - real*8
c                     lower limit of integration
c
c            b      - real*8
c                     upper limit of integration
c
c            epsabs - real*8
c                     absolute accuracy requested
c            epsrel - real*8
c                     relative accuracy requested
c                     if  epsabs.le.0
c                     and epsrel.lt.max(50*rel.mach.acc.,0.5d-28),
c                     the routine will end with ier = 6.
c
c            limit  - integer
c                     gives an upperbound on the number of subintervals
c                     in the partition of (a,b)
c
c         on return
c            result - real*8
c                     approximation to the integral
c
c            abserr - real*8
c                     estimate of the modulus of the absolute error,
c                     which should equal or exceed abs(i-result)
c
c            neval  - integer
c                     number of integrand evaluations
c
c            ier    - integer
c                     ier = 0 normal and reliable termination of the
c                             routine. it is assumed that the requested
```

```
c                           accuracy has been achieved.
c                    ier.gt.0 abnormal termination of the routine
c                           the estimates for integral and error are
c                           less reliable. it is assumed that the
c                           requested accuracy has not been achieved.
c            error messages
c                       = 1 maximum number of subdivisions allowed
c                           has been achieved. one can allow more sub-
c                           divisions by increasing the value of limit
c                           (and taking the according dimension
c                           adjustments into account). however, if
c                           this yields no improvement it is advised
c                           to analyze the integrand in order to
c                           determine the integration difficulties. if
c                           the position of a local difficulty can be
c                           determined (e.g. singularity,
c                           discontinuity within the interval) one
c                           will probably gain from splitting up the
c                           interval at this point and calling the
c                           integrator on the subranges. if possible,
c                           an appropriate special-purpose integrator
c                           should be used, which is designed for
c                           handling the type of difficulty involved.
c                       = 2 the occurrence of roundoff error is detec-
c                           ted, which prevents the requested
c                           tolerance from being achieved.
c                           the error may be under-estimated.
c                       = 3 extremely bad integrand behaviour
c                           occurs at some points of the integration
c                           interval.
c                       = 4 the algorithm does not converge.
c                           roundoff error is detected in the
c                           extrapolation table.
c                           it is presumed that the requested
c                           tolerance cannot be achieved, and that the
c                           returned result is the best which can be
c                           obtained.
c                       = 5 the integral is probably divergent, or
c                           slowly convergent. it must be noted that
c                           divergence can occur with any other value
c                           of ier.
c                       = 6 the input is invalid, because
c                           epsabs.le.0 and
c                           epsrel.lt.max(50*rel.mach.acc.,0.5d-28).
c                           result, abserr, neval, last, rlist(1),
c                           iord(1) and elist(1) are set to zero.
c                           alist(1) and blist(1) are set to a and b
c                           respectively.
c
c            alist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the left end points
c                     of the subintervals in the partition of the
c                     given integration range (a,b)
c
c            blist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the right end points
c                     of the subintervals in the partition of the given
c                     integration range (a,b)
c
c            rlist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the integral
c                     approximations on the subintervals
```

```
c
c            elist  - real*8
c                     vector of dimension at least limit, the first
c                      last  elements of which are the moduli of the
c                     absolute error estimates on the subintervals
c
c            iord   - integer
c                     vector of dimension at least limit, the first k
c                     elements of which are pointers to the
c                     error estimates over the subintervals,
c                     such that elist(iord(1)), ..., elist(iord(k))
c                     form a decreasing sequence, with k = last
c                     if last.le.(limit/2+2), and k = limit+1-last
c                     otherwise
c
c            last   - integer
c                     number of subintervals actually produced in the
c                     subdivision process
c***references  (none)
c***routines called  d1mach,dqelg,dqk21b,dqpsrt
c***end prologue  dqagse
c
```

dqelg.f

```
      subroutine dqelg(n,epstab,result,abserr,res3la,nres)
c***begin prologue  dqelg
c***refer to  dqagie,dqagoe,dqagpe,dqagse
c***routines called  d1mach
c***revision date  830518   (yymmdd)
c***keywords  convergence acceleration,epsilon algorithm,extrapolation
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  the routine determines the limit of a given sequence of
c            approximations, by means of the epsilon algorithm of
c            p.wynn. an estimate of the absolute error is also given.
c            the condensed epsilon table is computed. only those
c            elements needed for the computation of the next diagonal
c            are preserved.
c***description
c
c           epsilon algorithm
c           standard fortran subroutine
c           real*8 version
c
c           parameters
c              n      - integer
c                       epstab(n) contains the new element in the
c                       first column of the epsilon table.
c
c              epstab - real*8
c                       vector of dimension 52 containing the elements
c                       of the two lower diagonals of the triangular
c                       epsilon table. the elements are numbered
c                       starting at the right-hand corner of the
c                       triangle.
c
c              result - real*8
c                       resulting approximation to the integral
c
c              abserr - real*8
c                       estimate of the absolute error computed from
c                       result and the 3 previous results
```

```
c
c              res3la - real*8
c                       vector of dimension 3 containing the last 3
c                       results
c
c              nres   - integer
c                       number of calls to the routine
c                       (should be zero at first call)
c***end prologue  dqelg
c
```

dqk21.f

```
      subroutine dqk21(f,a,b,result,abserr,resabs,resasc)
c***begin prologue  dqk21
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a1a2
c***keywords  21-point gauss-kronrod rules
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  to compute i = integral of f over (a,b), with error
c                           estimate
c                       j = integral of abs(f) over (a,b)
c***description
c
c           integration rules
c           standard fortran subroutine
c           real*8 version
c
c           parameters
c            on entry
c              f      - real*8
c                       function subprogram defining the integrand
c                       function f(x). the actual name for f needs to be
c                       declared e x t e r n a l in the driver program.
c
c              a      - real*8
c                       lower limit of integration
c
c              b      - real*8
c                       upper limit of integration
c
c            on return
c              result - real*8
c                       approximation to the integral i
c                       result is computed by applying the 21-point
c                       kronrod rule (resk) obtained by optimal addition
c                       of abscissae to the 10-point gauss rule (resg).
c
c              abserr - real*8
c                       estimate of the modulus of the absolute error,
c                       which should not exceed abs(i-result)
c
c              resabs - real*8
c                       approximation to the integral j
c
c              resasc - real*8
c                       approximation to the integral of abs(f-i/(b-a))
c                       over (a,b)
c***references  (none)
c***routines called  d1mach
c***end prologue  dqk21
```

```
c
```

dqk212.f

```
      subroutine dqk212(f,func2,a,b,result,abserr,resabs,resasc)
*****************************************************************
*****************************************************************
c CLONE of dqk21, with external func2 as additional input
c Instead of f(x), this routine thus integrates f(func2,x)
c
c  Author: torsten.bringmann@fys.uio.no, 2018-04-27
*****************************************************************
*****************************************************************
c***begin prologue  dqk21
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a1a2
c***keywords  21-point gauss-kronrod rules
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  to compute i = integral of f over (a,b), with error
c                           estimate
c                       j = integral of abs(f) over (a,b)
c***description
c
c           integration rules
c           standard fortran subroutine
c           real*8 version
c
c           parameters
c            on entry
c              f      - real*8
c                       function subprogram defining the integrand
c                       function f(x). the actual name for f needs to be
c                       declared e x t e r n a l in the driver program.
c
c              a      - real*8
c                       lower limit of integration
c
c              b      - real*8
c                       upper limit of integration
c
c            on return
c              result - real*8
c                       approximation to the integral i
c                       result is computed by applying the 21-point
c                       kronrod rule (resk) obtained by optimal addition
c                       of abscissae to the 10-point gauss rule (resg).
c
c              abserr - real*8
c                       estimate of the modulus of the absolute error,
c                       which should not exceed abs(i-result)
c
c              resabs - real*8
c                       approximation to the integral j
c
c              resasc - real*8
c                       approximation to the integral of abs(f-i/(b-a))
c                       over (a,b)
c***references  (none)
c***routines called  d1mach
c***end prologue  dqk21
c
```

dqk21b.f

```
      subroutine dqk21b(f,a,b,result,abserr,resabs,resasc)
c***begin prologue  dqk21b
c***date written   800101   (yymmdd)
c***revision date  830518   (yymmdd)
c***category no.  h2a1a2
c***keywords  21-point gauss-kronrod rules
c***author  piessens, robert, applied math. and progr. div. -
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  to compute i = integral of f over (a,b), with error
c                           estimate
c                       j = integral of abs(f) over (a,b)
c***description
c
c           integration rules
c           standard fortran subroutine
c           real*8 version
c
c           parameters
c            on entry
c              f      - real*8
c                       function subprogram defining the integrand
c                       function f(x). the actual name for f needs to be
c                       declared e x t e r n a l in the driver program.
c
c              a      - real*8
c                       lower limit of integration
c
c              b      - real*8
c                       upper limit of integration
c
c            on return
c              result - real*8
c                       approximation to the integral i
c                       result is computed by applying the 21-point
c                       kronrod rule (resk) obtained by optimal addition
c                       of abscissae to the 10-point gauss rule (resg).
c
c              abserr - real*8
c                       estimate of the modulus of the absolute error,
c                       which should not exceed abs(i-result)
c
c              resabs - real*8
c                       approximation to the integral j
c
c              resasc - real*8
c                       approximation to the integral of abs(f-i/(b-a))
c                       over (a,b)
c***references  (none)
c***routines called  d1mach
c***end prologue  dqk21b
c
```

dqpsrt.f

```
      subroutine dqpsrt(limit,last,maxerr,ermax,elist,iord,nrmax)
c***begin prologue  dqpsrt
c***refer to  dqage,dqagie,dqagpe,dqawse
c***routines called  (none)
c***revision date  810101   (yymmdd)
c***keywords  sequential sorting
c***author  piessens, robert, applied math. and progr. div. -
```

```
c             k. u. leuven
c           de doncker, elise, applied math. and progr. div. -
c             k. u. leuven
c***purpose  this routine maintains the descending ordering in the
c            list of the local error estimated resulting from the
c            interval subdivision process. at each call two error
c            estimates are inserted using the sequential search
c            method, top-down for the largest error estimate and
c            bottom-up for the smallest error estimate.
c***description
c
c           ordering routine
c           standard fortran subroutine
c           real*8 version
c
c           parameters (meaning at output)
c              limit  - integer
c                       maximum number of error estimates the list
c                       can contain
c
c              last   - integer
c                       number of error estimates currently in the list
c
c              maxerr - integer
c                       maxerr points to the nrmax-th largest error
c                       estimate currently in the list
c
c              ermax  - real*8
c                       nrmax-th largest error estimate
c                       ermax = elist(maxerr)
c
c              elist  - real*8
c                       vector of dimension last containing
c                       the error estimates
c
c              iord   - integer
c                       vector of dimension last, the first k elements
c                       of which contain pointers to the error
c                       estimates, such that
c                       elist(iord(1)),...,  elist(iord(k))
c                       form a decreasing sequence, with
c                       k = last if last.le.(limit/2+2), and
c                       k = limit+1-last otherwise
c
c              nrmax  - integer
c                       maxerr = iord(nrmax)
c***end prologue  dqpsrt
c
```

Chapter 12

cr_aux: Cosmic rays – general

12.1 Cosmic Rays – auxiliary routines

This folder contains auxiliary functions needed by the cosmic ray routines. For example, it contains different versions of simple integration routines and a set of routines to handle the correct setting and interpreting of labels for the tabulation of confinement times and Green's functions needed for the calculation of cosmic ray fluxes.

Physics-wise, we include here also the interstellar intensity of the dominant cosmic ray species, dI/dR, where R is the particle's rigidity, and the corresponding fluxes, $d\Phi/dT = 4\pi(dR/dT)(dI/dR)$. These are needed by the routines for cosmic-ray upscattering – see Section 17.2 – and returned by the functions dscrISRintensity and dscrISRflux.

We finally provide for convenience two rescaling factors that can be applied to all (charged and neutral) cosmic ray fluxes that result from the annihilation of DM particles in the halo, in situations where the DM candidate in question (as initialized with the standard sequence of dsgive_model_xxx and dsmodelsetup) only provides some fraction of the total DM density in the halo. For symmetric DM, this simply amounts to a rescaling by the DM density squared, as returned by the function dscrrescale_sym. If the DM density has both a symmetric and an asymmetric component, on the other hand, one should rescale by the factor returned by dscrrescale_asym.

12.2 Routine headers – fortran files

dscrISRflux.f

```
*************************************************************************
*** Function dscrISRflux provides the differential local interstellar       ***
*** cosmic ray flux, per kinetic energy, computed from the intensity        ***
*** provided by dscrISRintensity.                                           ***
***                                                                         ***
***  Input:                                                                 ***
***    Tkin  - kinetic energy of CR particle [GeV]                          ***
***    CRtype - 1 for protons                                               ***
***             2 for helium                                                ***
***             3 for oxygen                                                ***
***             4 for carbon                                                ***
***  Units of output: 1/ [cm**2 s GeV]                                      ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-06-22                                                         ***
```

```
************************************************************************
      real*8 function dscrISRflux(Tkin, CRtype)
```

dscrISRintensity.f

```
************************************************************************
*** Function dscrISRintensity provides the differential intensity of the    ***
*** local interstellar cosmic rays (LIS).                                   ***
***                                                                         ***
***  Input:                                                                 ***
***    r  - rigidity of CR partisle [GV]                                    ***
***    CRtype - 1 for protons                                               ***
***             2 for helium                                                ***
***             3 for oxygen                                                ***
***             4 for carbon                                                ***
***                                                                         ***
***                                                                         ***
***  Units of output: 1/ [m**2 s sr GV]                                     ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-06-22                                                         ***
*** mod 2022-01-16 (H. Kolesova): updated CR fluxes, added O&C              ***
************************************************************************
      real*8 function dscrISRintensity(r, CRtype)
```

dscrISRread.f

```
************************************************************************
*** subroutine dscrISRread reads in cosmic ray intensities provided by
***
*** Boschini et al, APJ 250, 27 (2020)
***
*** Please remember to cite the above reference on top of the DarkSUSY paper
*** if you use these cosmic ray fluxes.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 2022-01-16
************************************************************************

      subroutine dscrISRread()
```

dscrrescale_asym.f

```
************************************************************************
*** The fluxes provided by the indirect detection routines in cr_.../ are   ***
*** based on the assumption that the initialized DM model corresponds to    ***
*** *all* of the Galactic DM, and that there is no asymmetric component.    ***
***                                                                         ***
*** The auxiliary function dscrrescale_asym can be used to rescale such     ***
*** fluxes -- similar to the function dscrrescale_asym, but for the case of ***
*** asymmetric DM, to take into account that the initialized DM model may   ***
*** only accounts for a fraction of the actual DM density in the halo (as,  ***
*** e.g., determined by a relic density calculation with dsrdomega_adm).    ***
***                                                                         ***
***  Input:                                                                 ***
***    oh2_rd - relic density of symmetric DM component                     ***
***    rsym   - relic fraction of symmetric to total DM density             ***
***    eta_model - asymmetry parameter used to initialize the model         ***
***    opt    - whether to rescale only the asymmetry (opt=2) or also the   ***
***             total DM density (opt=1, default).                          ***
***                                                                         ***
*** NB: This function is not automatically used elsewhere in the code. You  ***
***     as a user thus have to explicitly add the returned factor           ***
```

```
***     (if desired).                                                          ***
***                                                                            ***
*** author: Torsten.Bringmann.fys.uio.no                                       ***
*** date:   2022-10-18                                                         ***
********************************************************************************
      real*8 function dscrrescale_asym(oh2_rd,rsym,eta_model,opt)
```

dscrrescale_sym.f

```
********************************************************************************
*** The fluxes provided by the indirect detection routines in cr_.../ are   ***
*** based on the assumption that the initialized DM model corresponds to    ***
*** *all* of the Galactic DM, and that there is no asymmetric component.    ***
***                                                                          ***
*** The auxiliary function dscrrescale_sym can be used to rescale such      ***
*** fluxes by a factor                                                      ***
***    xi**2 = (oh2_rd/oh2_obs)**2,                                        ***
*** to account for the fact that the initialized DM model is symmetric, but ***
*** only accounts for a fraction xi of the actual DM density in the halo    ***
*** (as, e.g., determined by a relic density calculation with dsrdomega).   ***
***                                                                          ***
***  Input:                                                                  ***
***    oh2_rd  - relic density of symmetric DM component                     ***
***                                                                          ***
*** NB: This function is not automatically used elsewhere in the code. You  ***
***     as a user thus have to explicitly add the returned factor           ***
***     (if desired).                                                        ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date:   2022-10-18                                                      ***
********************************************************************************
      real*8 function dscrrescale_sym(oh2_rd)
```

dsfun_int.f

```
      subroutine dsfun_int(f,a,b,eps,prec,res)
c-----------------------------------------------------------------------
c  integrate function f between a and b using dqagse
c  input:
c    integration function: f
c    integration limits: a and b
c    integration precision: eps (rechecked and reset internally)
c    relative precision needed in worst case: prec
c  output:
c    result of integration res
c-----------------------------------------------------------------------
```

dsfun_intpar.f

```
      subroutine dsfun_intpar(f1,f2,a,b,tollf2,tollab,nmax,prec,how,res)
c-----------------------------------------------------------------------
c  integrate function f1 between a and b breaking the integration
c  interval into subintervals (ap,bp) such that
c    max(f2(ap),f2(bp)) < tollf2 * min(f2(ap),f2(bp))
c  input:
c    integration function f1
c    weight function f2
c    integration limits a and b
c    tollerance on weight function f2, tollf2
c    tollerance on how small the interval (ap,bp) can be tollab
c    maximum number of intervals nmax
c    precision on the sub-integral prec
```

```
c    how = 1: add subintervals in linear scale,
c    how = 2: add subintervals in log scale
c  integration performed with the subroutine dsfun_int
c  output:
c    result of integration res
c-----------------------------------------------------------------------
```

dsfun_intvec.f

```
      subroutine dsfun_intvec(f1,f2,a,b,tollf2,tollab,nmax,prec,how,
     &  abvec,resvec,nk)
c-----------------------------------------------------------------------
c  integral of function f1 between a and b + res1:
c     int_a^b dx f1 + res1
c  breaking the integration interval into subintervals abvec such that
c    max(f2(abvec(i)),f2(abvec(i+1)))
c       < tollf2 * min(f2(abvec(i)),f2(abvec(i+1)))
c  i.e. in the same way as it is done in dsfun_intpar.
c  input:
c    integration function f1
c    weight function f2
c    integration limits a and b
c    tollerance on weight function f2, tollf2
c    tollerance on how small (abvec(i),abvec(i+1)) can be tollab
c    maximum number of intervals nmax
c    precision on the sub-integral prec
c    how = 1: add subintervals in linear scale,
c    how = 2: add subintervals in log scale
c  integration performed with the subroutine dsfun_int
c  output:
c    number of subintervals nk
c    vector abvec such that:
c      abvec(0) = a
c      abvec(nk) = b
c    vector resvec such that:
c      resvec(0) = 0.d0
c      resvec(i) = integral of f1 in the interval (abvec(i-1),abvec(i))
c-----------------------------------------------------------------------
```

dslabelset.f

```
      subroutine dsaddlabel2(labelinoutfun,labelin)
**********************************************************************
*** subroutine which adds to file 'labelfile' the new label input
*** 'labelin' and a row of associated npar parameters par
*** if labelain already exists and it is associated to the same set of
*** parameters, the subroutine just returns
*** if labelin already exists and it is associated to a different set
*** of parameters, an error message is printed and the program stops
***
*** inputs:
***     labelinoutfun - external function setting internal common blocks
***     labelin - the label to be added (character*10)
*** inputs from the common block dslabelcom:
***     labelfile - in the file in which label and parameters are stored
***     npar - the number of parameters to be stored (npar.le.100)
***     par - a real*8 vector with the parameters to be stored
**********************************************************************
```

dslinint.f

```
      subroutine dslinint(xvec,yvec,n,x,y)
c simple linear interpolation
```

dstabfun.f

```
      subroutine dstabfun(fun,rmin,rmax,tollf,tollr,how)
c-----------------------------------------------------------------------
c  set a tabulation for the function fun in the interval (rmin, rmax);
c  intermadiate points r_i are such that
c    max(fun(r_i),fun(r_i+1)) < tollf * min(fun(r_i),fun(r_i+1))
c  input:
c    function to tabulate fun
c    extrema for tabulation rmin,rmax
c    tollerance on tabulation tollf
c    tollerance on how small the interval (r_i,r_i+1) can be tollr
c    maximum numeber of intervals nmax (this must be .le.1000)
c    how = 1: add subintervals and tabulate fun in linear scale
c    how = 2: add subintervals and tabulate fun in log scale
c  output:
c    result of integration res
c-----------------------------------------------------------------------
```

dstabfun2.f

```
      subroutine dstabfun2(fun,rmin,rmax,tollf,tollr,how)
c-----------------------------------------------------------------------
c  set a tabulation for the function fun in the interval (rmin, rmax);
c  intermadiate points r_i are such that
c    max(fun(r_i),fun(r_i+1)) < tollf * min(fun(r_i),fun(r_i+1))
c  input:
c    function to tabulate fun
c    extrema for tabulation rmin,rmax
c    tollerance on tabulation tollf
c    tollerance on how small the interval (r_i,r_i+1) can be tollr
c    maximum numeber of intervals nmax (this must be .le.1000)
c    how = 1: add subintervals and tabulate fun in linear scale
c    how = 2: add subintervals and tabulate fun in log scale
c  output:
c    result of integration res
c-----------------------------------------------------------------------
```

dstabfunl.f

```
      subroutine dstabfunl(fun,rmin,rmax,fpmin,fpmax,tollf,tollr,how)
c-----------------------------------------------------------------------
c  set a tabulation for the function fun in the interval (rmin, rmax);
c  a sparse regular grid is set and then intermadiate points r_i are
c  added up to reaching the condition that:
c
c    abs(1-interpolation(r_i)/fun(r_i)) < tollf
c
c  input:
c    function to tabulate fun
c    extrema for tabulation rmin,rmax
c    fpmin & fpmax: first derivative of fun computed in rmin,rmax; to
c      use natural splines call the subroutine with values larger than
c      1.d30; for the linear interpolation case these are not used
c    tollerance on tabulation tollf
c    tollerance on how small the interval (r_i,r_i+1) can be tollr
c    maximum numeber of intervals nmax (this must be .le.1000)
c    how = 1: add subintervals and tabulate fun in linear scale with
c             linear interpolation
c    how = 2: add subintervals and tabulate fun in log scale with
c             linear interpolation
c    how = 101: the same as how=1 but with cubic spline interpolation
c    how = 102: the same as how=2 but with cubic spline interpolation
c  output:
```

```
c    tabulation stored in dstabfuncom common block
c-------------------------------------------------------------------------
```

Chapter 13

cr_axi: Cosmic rays – diffusion routines for axisymmetric distributions

13.1 Cosmic ray propagation in axially symmetric halos

There is a clean asymmetry between particles and antiparticles in the standard cosmic ray picture: The bulk of cosmic rays – protons, nuclei and electrons – are mainly "primary" species, i.e. particles accelerated in astrophysical sources and then copiously injected in the interstellar medium; "secondary" components, including antimatter, are instead produced in the interaction of primaries with the interstellar medium during the propagation. It follows that there is a pronounced deficit of antimatter compared to matter in the locally measured cosmic ray flux (about 1 antiproton in 10^4 protons). When considering instead a source term due to DM annihilations or decays, a significant particle-antiparticle asymmetry is in general not expected, and antiprotons, positrons and antideuterons turn out to be competitive indirect DM probes.

Charged particles propagate diffusively through the regular and turbulent components of Galactic magnetic fields. This makes it more involved for local measurements to track spectral and morphological imprints of DM sources than, e.g., for the gamma-ray and neutrino channels (though searches for spectral features in CR positron fluxes still lead to very competitive limits [32]). In fact the transport of cosmic rays in the Galaxy is still a debated subject: Most often one refers to the quasi-linear theory picture (with magnetic inhomogeneities as a perturbation compared to regular field lines) in which propagation can be described in terms of a (set of) equation(s) linear in the density of a given species, containing terms describing diffusion in real space, diffusion in momentum space (the so-called reacceleration), convective effects due to Galactic winds, energy or fragmentation losses and primary and secondary sources (see, e.g., Ref. [33] for a review). Dedicated codes have been developed to solve numerically this transport equation, including GALPROP [34], DRAGON [35] and PICARD [36].

Here we follow instead a semi-analytical approach, analogous to that developed for the USINE code [37]. In particular, we model the propagation of antiprotons and antideuterons by considering the steady state equation [21]

$$\frac{\partial N}{\partial t} = 0 = \nabla \cdot (D\,\nabla N) - \nabla \cdot (\vec{u}\,N) - \frac{N}{\tau_N} + Q\,. \tag{13.1}$$

We solve it for situations where *i)* the diffusion coefficient D can have an arbitrary dependence on the particle rigidity but can at most take two different values in the Galactic disc and in the diffusive halo, *ii)* the convective velocity $\vec{u}$ has a given fixed modulus and is oriented outwards

and perpendicular to the disc, *iii)* the loss term due to inelastic collisions has an interaction time τ_N which is energy dependent but spatially constant in the disc and going to infinity in the halo (corresponding to a constant target gas density in the disc and no gas in the halo), *iv)* the DM source Q is spatially axisymmetric and has a generic energy dendence. Under these approximations and assuming, as is usually done, that the propagation volume is a cylinder centred at the disc and that particles can freely escape at the boundaries of the diffusion region, Eq. (13.1) can be solved analytically by expanding N in a Fourier-Bessel series; the computation of the flux involves, at each energy, a sum over the series of zeros of a Bessel function of first kind and order zero, and a volume integral of the spatially dependent part in the axisymmetric source term Q (basically the DM density ρ_χ for decaying DM and its square for pair annihilating DM) times a weight function depending on the given zero in the series (see [21] for further details).

Since the path lengths for antiprotons and antideuterons are rather large, of the order of a few kpc, taking average values for parameters in the transport equation rather than the more realistic modelling that can be implemented in full numerical solutions, has no large impact in case of extended and rather smooth sources such as for DM. Eq. (13.1) neglects reacceleration effects, which may in general be relevant at low energies (rigidities below a few GV); however even this does not have a large impact in case of the species at hand, see, e.g., [38] for a comparison of results with numerical and semi-analytical solutions for cosmic-ray antiprotons. The power of our semi-analytic approach is that one can store values of the solution of the transport equation obtained by assuming a given mono-energetic source – provided by the functions dspbtdaxi and dsdbtdaxi for, respectively, antiprotons and antideuterons – and then apply these as weights to any particle source term $\mathcal{S}_n(E_f)$ as introduced above. For antiprotons and antideuterons, this latter step is done in the functions that compute the local galactic differential fluxes from DM annihilation and decay, dspbdphidtaxi and dsdbdphidtaxi, respectively. Note that while the outputs of dspbtdaxi and dsdbtdaxi are labelled "confinement time" in the code, since they do have a dimension of time and scale the dependence between source and flux, one cannot trade them for what is usually intended as confinement time for standard cosmic ray components, given that the morphology of the DM source is totally different from supernova remnant distributions usually implemented for describing ordinary primary components.

The structure we implemented gives a particularly clear advantage when the code is used to scan over many particle physics DM models, but only over a limited number of propagation parameters and DM density profiles. For such an application, it is useful to tabulate the 'confinement times' (returned by dspbtdaxi and dsdbtdaxi) over a predefined range of energy; this is done in the functions dspbtdaxitab and dsdbtdaxitab when calling the flux routines with an appropriate option (and only in case the DM halo profile currently active is within the halo profile database). Such tabulations can be saved and re-loaded for later use; here the proper table association is ensured by a propagation parameter label setting system in analogy to the one implemented for the halo profile database. Computing such a table on the first call is rather CPU consuming, especially for DM profiles that are singular towards the Galactic center, so in case only a few flux computations are needed it may be better to switch off the tabulation option; this is true also in case the flux is needed at a small number of energy values, since the latest 100 (non-equivalent) calls to dspbtdaxi and dsdbtdaxi are stored in memory (with the corresponding propagation parameters and halo model correctly associated).

The case for positrons is treated analogously, except that energy losses and spatial diffusion are the most important effects for propagating cosmic ray leptons. The transport equation we solve semi-analytically therefore has the form [20]

$$\frac{\partial N}{\partial t} = 0 = \nabla \cdot (D\,\nabla N) + \frac{\partial}{\partial p}\left(\frac{dp}{dt}N\right) + Q\,, \tag{13.2}$$

where the functional form of D and Q can be chosen as for antiprotons and antideuterons, and the energy loss rate dp/dt can have a generic momentum dependence but needs again to be spatially constant. Assuming the same topology for the propagation volume and free escape conditions at

the vertical boundaries (to compute the local positron flux the radial boundary turns out to be irrelevant) , the solution of the propagation equation is given in terms of a Green's function in energy (the function dsepgreenaxi in the code) to be convoluted over the source energy spectrum at emission for a given particle DM candidate. This last step is performed by the function dsepdndpaxi returning the local positron number density, while dsepdphidpaxi converts it to a flux and is the function which should be called from the main file. The method to implement this solution is a slight generalization of the one described [20] and generalizes the one introduced in [39] for a spherically symmetric configuration to an axisymmetric system.

The computation of the Green's function involves a volume integral over the spatially dependent part of the DM source function Q (again basically the DM density ρ_χ for decaying DM and its square for pair annihilating DM) and the implementation via the so-called method of image charges (again a sum over a series) of the free escape boundary condition. It can again be CPU expensive for singular halo profiles, but its tabulation is always needed since the Green function appears in a convolution. The main limiting factor with respect to full numerical solutions is that one is forced to assume an (spatially) average value for dp/dt. However this has not a severe impact on our results for the local DM-induced positron flux since, especially for energies above 10 GeV, the bulk of the DM contribution to the local flux stems from a rather close-by emission volume; one thus just has to make sure to normalize dp/dt to the mean value for *local* energy losses, as opposed to the mean value in the Galaxy, which are mainly due to the synchrotron and inverse Compton processes.

While the transport equations (13.1) and (13.2) are essentially the same as considered in previous releases of the code, their implementation in the present release is completely new and appears to be numerically more stable. In particular cases with very singular DM profiles still give numerically accurate results and converge faster (in case of antiprotons and antideuterons implementing a procedure which applies to point sources). Note however that the case of very singular DM profiles is also the one in which our models or *any* propagation model is subject to a significant uncertainty related to the underlying physics, since propagation in the Galactic center region is difficult to model and probably rather different from what can be tested in the local neighbourhood by measurements of primary and secondary cosmic rays. Finally, the implementation starting from DarkSUSY 6 is more flexible regarding parameter choices, such as for the rigidity scaling of the diffusion coefficient and energy scaling of energy losses, in a framework which is now fully consistent for antiprotons, antideuterons and positrons.

13.2 Routine headers – fortran files

dscraxi_init.f

```
      subroutine dscraxi_init
**********************************************************************
*** initialization routine for cosmic ray propagation routine
*** the subroutine contains settings of options or control parameters
*** that not necessarily needed to be changed when changing propagation
*** parameters, but however may require different optimizations.
*** a few propagation parameters are also set here, however are
*** parameters that the user would seldomly change
*** finally a switch on whether you want to treat propagation of
*** antiprotons and antideuterons in the approximation of delta
*** function for the gas disc or finite thickness is set here
***
*** the meaning of options/paramters specified before their setting in
*** the file
***
**********************************************************************
```

dscraxirzck.f

```
************************************************************************
***  subroutine checking whether you need to reload the label associated
***  to the position in the galaxy where craxi tabulations are computed
***
***  output: noout -> an integer which is increased for any new set of
***          parameters
***
************************************************************************
      subroutine dscraxirzck(R,z,noout)
```

dscraxirzsetlabel.f

```
************************************************************************
***  function setting the correspondence between a label and a set of
***  paramters defining to the position in the galaxy where craxi
***  tabulations are computed
***
************************************************************************
      real*8 function dscraxirzsetlabel(inout)
```

dscraxiset_default.f

```
      subroutine dscraxiset_default
************************************************************************
*** a sample default setting of propagation parameters. this subroutine
*** is called in the program inizialization. to change the propagation
*** parameters to another set call the routine dscraxiset_model.
*** users CAN NOT change hardcoded values of paramaters in this routine
*** changing parameters here could possibly give wrong results
***
************************************************************************
```

dscraxiset_model.f

```
      subroutine dscraxiset_model(gparin,kparin,eparin,labpbin,labepin)
************************************************************************
*** subroutine setting propagation parameters.
***
***   type : REPLACEABLE
***
*** it assumes that the diffusion coefficient dskdiff and the mean
*** positron energy loss dseppdotmmean are in the hardcoded versions
*** given in the cr_axi directory. if that are replaced you should
*** change accordingly this subroutine:
***
*** input:
*** - gparin -> vector of propagation parameters for antiprotons,
**    antideuterons and positrons (for the latter only diffhh
***      gparin(1) = diffhh -> 1/2 of vertical size of the diff. region
***                in kpc
***      gparin(2) = diffRh -> radial size of the diff. region in kpc
***      gparin(3) = diffcvel -> galactic wind velocity in km/s
*** - kparin -> vector of parameters for the diffusion coefficient as
***   given in the function dskdiff -- dskdiff can be replaced by some
***   other function and then you need to change this routine
***      kparin(1) = dble(nkdiff) -> index for functional form hardcoded
***                 in dskdiff
***      kparin(2) = k0halo, k0gasdisc -> normalization of the diffusion
***      coefficient in the halo and the disc in 10^27 cm^2 s^-1
***      kparin(3) = kdiffrig0 -> rigidity break in GV
***      kparin(4) = kdiffdelta ->  spectral index (above kdiffrig0 if
```

```
***              nkdiff=2)
***      kparin(5) = kdiffeta -> exponent of beta**kdiffeta prefactor,
***              in case kdiffeta is larger than 1.d-7
***      kparin(6) = kdiffdeltalow ->  spectral index below kdiffrig0
***              if nkdiff=2
*** - eparin -> vector of parameters for the positron energy loss
***   function dseppdotmmean (ivopt=1) or dseppdotmana (ivopt=1).
***   dseppdotmmean can be replaced by some other function and then you
***   need to change this routine, dseppdotmana cannot be replaced
***   since it is the only form for which the analytic matching pp<->v
***   works. ivopt=2 also works only in case nkdiff = 1 or 4
***      eparin(1) = dble(ivopt) -> option for energy losses and tabulation
***           of the positron green function in the variable v
***      ivopt=1 -> the matching between  pp and the variable v is done
***           assuming fully general momentum loss rate and spatial
***           diffusion coefficient, with a numerical integral + a
***           tabulation involved; you link to dskdiff and dseppdotmmean
***           which can be replaced by arbitry functions of, respectively
***           rigidity and energy
***      ivopt=2 -> the matching between  pp and the variable v is done
***           assuming the momentum loss rate scales with pp^2, and the
***           diffusion coefficient has a form which allows for an
***           analitical inversion between v and pp, see Eq.~(6) & (7) in
***           Baltz & Edsjo (1998)
***      NOTE: ivopt is a common block variable on a fixed range, user
***      cannot other ivopt values!
***      if ivopt = 1
***        eparin(2) = starlightmean -> mean optical + IR light density,
***           eV cm^-3
***        eparin(3) = bfieldmean -> mean magnetic field, \muG
***      if ivopt = 2
***        eparin(2) = blossmean -> some mean value pdot0 assuming the
***           momentum loss rate scales with pdot0 * pp^2
*** - labpbin = label for pb & db confinement time; if set to 'none'
***   a label is searched in the label file and automatically generated
***   if it does not exist -- default option as set in dscraxi_init, if
***   that option has been changed then output modified accordingly
*** - labepin = label for ep green functions; if set to 'none'
***   a label is searched in the label file and automatically generated
***   if it does not exist -- default option as set in dscraxi_init, if
***   that option has been changed then output modified accordingly
***
**********************************************************************
```

dsdbdphidtaxi.f

```
      real*8 function dsdbdphidtaxi(tdin,phiin,how,labhalo)
**********************************************************************
*** function which computes the local galactic differential flux of
*** antideuterons at the kinetic energy per nucleon tdin as a result of
*** dark matter pair annihilations or decay in the halo.
*** inputs:
***     tdin = antideuteron kinetic energy per nucleon td in GeV
***     phiin - solar modulation parameter in GV, assuming that solar
***             modulation can be treated with the force-field method
***             - if phiin.le.1.d-3, the interstellar flux is returned
***     how = 1 - calculate the flux only for requested tdin
***           2 - a table is tabulated on first  call, and then
***               interpolated
***           3 - as 2, but also write the table to disk at the
***               first call
***           4 - read table from disk on first call, and use the
***               subsequent calls. If the file does not exist, it
***               will be created (as in 3). (use as default)
***     labhalo = halo model access label, within the set of models
```

```
***              defined in the halo model repository
*** output: cm^-2 s^-1 GeV^-1 sr^-1
***
***   type : commonly used
***   desc : Local galactic differential antideuteron flux from dark matter
***
***
***  author: Piero Ullio
***  modified: Joakim Edsjo, to use new yields and crsource directly
**********************************************************************
```

dsdbsigmavdbar.f

```
      real*8 function dsdbsigmavdbar(en)
c total inelastic cross section dbar + h
c Yad.Fiz.14:134-136,1971
c check Review of Particle Properties of 1992, rev [9] in DFS
```

dsdbtdaxi.f

```
      real*8 function dsdbtdaxi(R,z,td,powerin,isin,prec,nprec,labhalo
     &   ,loadlab)
**********************************************************************
*** antideuteron "confinement time" at the position (R,z)
***    (NOTE: at the moment only z=0 is implemented)
*** for an axisymmetric primary source assumed to extend over the
*** whole diffusion region, as specified by the function dspbdmasaxi
***
*** input:
***     td - kinetic energy per nucleon (gev)
***
*** see the header of the function dspbtdaxi for details on other inputs
*** and outputs, since they are in perfect analogy to those here.
*** NOTE: the setup with the inner cylinder at the Galactic center
*** excluded from the spatial integration and treated with the point
*** source green function to speed up convergence is NOT changed here,
*** assuming that convergence issues, for a given source function and R
*** are analogous
***
*** output in 10^15 s
***
**********************************************************************
```

dsdbtdaxitab.f

```
      real*8 function dsdbtdaxitab(R,z,tpin,powerin,isin,how,labhalo
     &   ,loadlab)
**********************************************************************
*** tabulated version of the db "confinement time" dsdbtdaxi
***
*** input variables:
***   R = radial coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***   z = vertical coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***      NB: only z=0 works at the moment
***   tpin = antideuteron kinetic energy per nucleon in GeV
***   powerin = integer selecting annihilations (=2) or decays (=1)
***   isin = propagation model option : isin=1 -> model with delta
***      function approximation for the gas disc; isin=2 -> two-zone
***      model with finite thickness for the gas disk
***   how = 1 - calculate the time only for requested tpin
***         2 - table is tabulated on first call, and then
```

```
***             interpolated
***         3 - as 2, but also write the table to disk at the
***             first call
***         4 - read table from disk on first call, and use the
***             subsequent calls. If the file does not exist, it
***             will be created (as in 3). (use as default)
***     the tabulation is created with dsdbtdaxi with nprec and prec as
***     set by the corresponding pbprec and pbnprec in common blocks.
***     you need to load a new table whenever:
***       - the halo model is changed
***       - the propagation model is changed
***       - R, z or isin are changed
***     if pbtdaxiloadck=.true. there is an automatic check on these
***     and tables are eventually reloaded
***   labhalo = halo model access label, within the set of models
***             defined in the halo model repository
***   loadlab = logical variable: in case it is set to true the halo
***     labhalo is selected within this function, otherwise it is
***     assumed that it has been loaded before linking to this function
***     such as in the function dspbdphidtaxi
***
*** output: 10^15 s
**************************************************************************
```

dsepcraxick.f

```
**************************************************************************
***  subroutine checking whether you need to reload the label associated
***  to diffusion parameters involved in the ep axisymmetric green
***  function tabulation. If they are the integer dsepcraxino is
***  increased by one unit
***
**************************************************************************
      subroutine dsepcraxick
```

dsepcraxisetlabel.f

```
**************************************************************************
***  function setting the correspondence between a label and a set of
***  parameters involved in the ep axisymmetric green function
***  tabulation
***
**************************************************************************
      real*8 function dsepcraxisetlabel(inout)
```

dsepdmasaxi.f

```
      real*8 function dsepdmasaxi(R,z,power)
**************************************************************************
***  spatially dependent part of the positron source function.
***  an axisymmetric profile is required here.
***
***  input: galactic coordinates R,z in kpc
***  power = integer selecting annihilations (=2) or decays (=1)
***  output:  [GeV^2 cm^-6] for dark matter pair annihilations
***           [GeV cm^-3] for dark matter decays
**************************************************************************
```

dsepdndpaxi.f

```
**************************************************************************
*** electron/positron equilibrium number density per unit momentum, at
```

```
*** the momentum pp and at the location (R,z), due to wimp pair
*** annihilations in a static, axisymmetric dark matter halo and up to
*** a normalization factor, see below.
*** set of further assumptions:
***   - axisymmetric diffusion region with radial boundary condition
***     neglected (do not use this function with R too close to the
***     radial size of the diffusion region);
***   - spatial diffusion coefficient and the energy loss rate
***     independent of R and z;
***   - reacceleration and convection neglected.
*** inputs:
***     R - radial coordinate in kpc
***     z - vertical coordinate in kpc
***     pp - positron momentum in GeV
***     power - integer selecting annihilations (=2) or decays (=1)
***     iv = 1 - the matching between  pp and the variable v is done
***              assuming fully general momentum loss rate and spatial
***              diffusion coefficient, with a numerical integral + a
***              tabulation involved
***        = 2 - the matching between  pp and the variable v is done
***              assuming the momentum loss rate scales with pp^2,
***              and the diffusion coefficient has a form which
***              allows for an analitical inversion between v and pp,
***              see Eq.~(6) & (7) in Baltz & Edsjo (1998)
***     how = 2 - a table for green function is created on first  call,
***               and then interpolated
***           3 - as 2, but also write the table to disk at the
***               first call
***           4 - read table from disk on first call, and use the
***               subsequent calls. If the file does not exist, it
***               will be created (as in 3). (use as default)
***     labhalo = halo model access label, within the set of models
***             defined in the halo model repository
***     loadlab = logical variable: in case it is set to true the halo
***       labhalo is selected within this function, otherwise it is
***       assumed that it has been loaded before linking to this
***       function such as in the function dsepdphidpaxi
*** output: cm^-3 GeV^-1
***
*** author: Piero Ullio
*** modified: Torsten Bringmann, 12/06/2015
***           (made replaceable & linked to dscrsource_line)
**********************************************************************
      real*8 function dsepdndpaxi(R,z,pp,power,iv,how,labhalo,loadlab)
```

dsepdndpi.f

```
**********************************************************************
*** initial (i.e. prior propagation) electron/positron spectrum
*** according to the the default in ds, i.e. dscrsource.
***
***  type : REPLACEABLE
***
***  input: pp = momentum (GeV)
***         power = integer selecting annihilations (=2) or decays (=1)
***  output: GeV^-1  / [(cm^3 s) / (GeV / cm^3)^crdmpow]
***
*** author: Piero Ullio
*** modified: Torsten Bringmann, 12/06/2015
***           (made replaceable & linked to dscrsource)
**********************************************************************
      real*8 function dsepdndpi(pp,power)
```

dsepdphidpaxi.f

```
      real*8 function dsepdphidpaxi(ppin,phiin,how,labhalo)
**************************************************************************
*** function which computes the local galactic differential flux of
*** positrons at the momentum pp as a result of dark matter annihilation
*** or decay in the halo.
*** inputs:
***     ppin - positron momentum in GeV
***     phiin - solar modulation parameter in GV, assuming that solar
***             modulation can be treated with the force-field method
***             - if phiin.le.1.d-3, the interstellar flux is returned
***     how = 2 - a table is tabulated on first  call, and then
***               interpolated
***           3 - as 2, but also write the table to disk at the
***               first call
***           4 - read table from disk on first call, and use the
***               subsequent calls. If the file does not exist, it
***               will be created (as in 3). (use as default)
***     labhalo = halo model access label, within the set of models
***               defined in the halo model repository
*** output: cm^-2 s^-1 GeV^-1 sr^-1
***
***   type : commonly used
***   desc : Local galactic differential positron flux from dark matter
***
***
***  author: Piero Ullio
**************************************************************************
```

dsepgreenaxi.f

```
      real*8 function dsepgreenaxi(R,z,DeltaV,powerin,labhalo,loadlab)
**************************************************************************
*** green function to derive the electron/positron equilibrium number
*** density due to wimp pair annihilations in a static, axisymmetric
*** dark matter halo. see dsepdndpaxi for a complete list of underlying
*** assumptions.
*** inputs:
***     R - radial position of the observer in kpc
***     z - vertical position of the observer in kpc
***     DeltaV - increment in the variable vvar in kpc^2
***     4*DeltaVin = lambda**2, with lambda the diffusion length in kpc
***     labhalo = halo model access label, within the set of models
***               defined in the halo model repository
***     loadlab = logical variable: in case it is set to true the halo
***       labhalo is selected within this function, otherwise it is
***       assumed that it has been loaded before linking to this
***       function such as in the function dsepdphidpaxi, dsepdndpaxi or
***       dsepgreenaxitab
***
*** output: dimensionless
**************************************************************************
```

dsepgreenaxitab.f

```
      real*8 function dsepgreenaxitab(R,z,DeltaVin,powerin,how,
     &  labhalo,loadlab)
**************************************************************************
*** tabulated version of the green function dsepgreenaxi(R,z,DeltaV).
*** on first call the table is either computed or read from a file
*** provided by the user. this is handled through the input variable:
***     how = 2 - table is tabulated on first  call, and then
***               interpolated
```

```
***           3 - as 2, but also write the table to disk at the
***               first call
***           4 - read table from disk on first call, and use the
***               subsequent calls. If the file does not exist, it
***               will be created (as in 3). (use as default)
*** you need to load a new table whenever:
***   - the halo model is changed
***   - the vertical height of the diffusion zone or the inner radial
***     cutoff are changed
***   - R or z are changed
***     if epgraxiloadck=.true. there is an automatic check on these
***     and tables are eventually reloaded
***
*** other inputs:
***     R - radial position of the observer in kpc
***     z - vertical position of the observer in kpc
***     DeltaVin - increment in the variable vvar in kpc^2
***     4*DeltaVin = lambda**2, with lambda the diffusion length in kpc
***     powerin = integer selecting annihilations (=2) or decays (=1)
***     labhalo = halo model access label, within the set of models
***             defined in the halo model repository
***     loadlab = logical variable: in case it is set to true the halo
***       labhalo is selected within this function, otherwise it is
***       assumed that it has been loaded before linking to this
***       function such as in the function dsepdphidpaxi or dsepdndpaxi
***
*** output: dimensionless
**********************************************************************
```

dseppdotm.f

```
      real*8 function dseppdotm(pp,iv)
**********************************************************************
*** electron/positron momentum loss rate -pdot(pp) -- average value in
*** the milky way, or value applying close to the place where the
*** equilibrium number density for electron/positrons is computed.
*** input: pp - momentum (GeV)
***        iv = 1 - links to a function including a full set of
***                 energy loss effects
***        iv = 2 - links to a function with the simple scaling
***                 blossmean *pp^2
*** output: 10^-16 GeV s^-1
**********************************************************************
```

dsepvarch.f

```
      real*8 function vvar(pp,iv)
**********************************************************************
*** input: pp - momentum (GeV)
***        iv = 1 - links to vvarnum
***        iv = 2 - links to vvarana
*** output: v = int_0^{u(p)} d\tilde{u} D(\tilde{u}) (kpc^2)
**********************************************************************
```

dsffsolmodaxi.f

```
      subroutine dsffsolmodaxi(tpin,phiin,mass,A,modZ,tp,smrf)
**********************************************************************
***  subroutine which computes the shift in kinetic energy per nucleon
***  to account for solar modulation in the local flux of cosmic rays.
***  the 1-parameter force-field method is assumed.
***
***  inputs:
```

```
***     tpin = kinetic energy per nucleon at earth position [GeV]
***     phiin = solar modulation parameter in GV, assuming that solar
***             modulation can be treated with the force-field method
***             - if phiin.le.1.d-3, tp = tpin is returned
***     mass = mass of the cr particle
***     A = number of nucleons of the cr particle
***     modZ = absolute value of the charge of the cr particle
***
***  output:
***     tp  = kinetic energy per nucleon in the local interstellar (LIS)
***           medium [GeV]
***     smrf = flux rescaling factor: flux_earth = smrf * flux_interst.
***
***  NOTE: this routine assumes the rigid shift:
***          E_LIS = E_earth + modZ * phiin
***  and not, as sometimes done, a rigid shift in rigidity (p/modZ):
***          R_LIS = R_earth + phiin
***  this second option is commented out within the file.
***  author: Piero Ullio
**********************************************************************
```

dskdiff.f

```
**********************************************************************
*** Function dskdiff returns the spatial diffusion coefficient
***
***  type : REPLACEABLE
***
*** inputs:
***     rig - particle rigidity in GV
***     n = 1 - diffusion coefficient in the halo
***     n = 2 - diffusion coefficient in the gas disc
***     beta - particle velocity in c units (this value affects
***            dskdiff only in case betalabel=.true.)
***
*** additional inputs are through common blocks, in particular the
*** spectral is set by nkdiff:
***     nkdiff = 1 - K0 * (rig/kdiffrig0)**kdiffdelta
***     nkdiff = 2 - K0 * (rig/kdiffrig0)**kdiffdelta
***                    iff rig>kdiffrig0
***                  K0 * (rig/kdiffrig0)**kdiffdeltalow
***                    iff rig<kdiffrig0
***     nkdiff = 3 - K0 * (1+rig/kdiffrig0)**kdiffdelta
***     nkdiff = 4 - K0 * [1+(rig/kdiffrig0)**kdiffdelta]
*** K0 is eqal to k0halo or k0gasdisc depending on n. For
*** dabs(kdiffeta).gt.1.d-7 all the above are multiplied by
***   beta**kdiffeta
***
*** output: 10^27 cm^2 s^-1
***
***  author: Piero Ullio
***  modified: Torsten Bringmann, 11/06/2015 (adapted to replaceable
***                                           function concept)
**********************************************************************
      real*8 function dskdiff(rig,n,beta)
```

dspbbessel.f

```
      real*8 function dspbbesselzeroj0(s)
**********************************************************************
*** s-th zero of J0
**********************************************************************
```

dspbcraxick.f

```
************************************************************************
***  subroutine checking whether you need to reload the label associated
***  to diffusion parameters involved in the pb and db diffusion time
***  tabulations. If they are the integer dspbcraxino is increased by
***  one unit
***
************************************************************************
      subroutine dspbcraxick
```

dspbcraxisetlabel.f

```
************************************************************************
***  function setting the correspondence between a label and a set of
***  parameters involved in the pb and db diffusion time tabulations
***
***  NOTE: this version is connected to the specific choice of the
***  diffusion coefficient function dskdiff
***
***  type : REPLACEABLE
***
************************************************************************
      real*8 function dspbcraxisetlabel(inout)
```

dspbdmasaxi.f

```
      real*8 function dspbdmasaxi(R,z,power)
************************************************************************
*** spatially dependent part of the antiproton and antideuteron source
*** function. an axisymmetric profile is required here.
***
***  input: galactic coordinates R,z in kpc
***  power = integer selecting annihilations (=2) or decays (=1)
***  output:  [GeV^2 cm^-6] for dark matter pair annihilations
***           [GeV cm^-3] for dark matter decays
************************************************************************
```

dspbdphidtaxi.f

```
      real*8 function dspbdphidtaxi(tpin,phiin,how,labhalo)
************************************************************************
*** function which computes the local galactic differential flux of
*** antiprotons at kinetic energy tpin as a result of dark matter
*** annihilation or decay in the halo.
***
*** type : commonly used
*** desc : Local galactic differential antiproton flux from dark matter
***
*** inputs:
***     tpin = antiproton kinetic energy tp in GeV
***     phiin - solar modulation parameter in GV, assuming that solar
***             modulation can be treated with the force-field method
***             - if phiin.le.1.d-3, the interstellar flux is returned
***     how = 1 - calculate the flux only for requested tpin
***           2 - a table is tabulated on first  call, and then
***               interpolated
***           3 - as 2, but also write the table to disk at the
***               first call
***           4 - read table from disk on first call, and use the
***               subsequent calls. If the file does not exist, it
***               will be created (as in 3). (use as default)
***     labhalo = halo model access label, within the set of models
```

```
***              defined in the halo model repository
*** output: cm^-2 s^-1 GeV^-1 sr^-1
***
***  author: Piero Ullio
**************************************************************************
```

dspbpstab.f

```
      subroutine dspbpstab(tp,R,isin,tollfz,tollfr,tollfr2)
**************************************************************************
*** tabulation of pbpoint function needed for axisymmetric source
**************************************************************************
```

dspbsigmavpbar.f

```
      real*8 function dspbsigmavpbar(en)
c total inelastic cross section pbar + h
c tan and ng, j.phys.g 9 (1983) 227. formula 3.7
```

dspbtdaxi.f

```
      real*8 function dspbtdaxi(R,z,tp,powerin,isin,prec,nprec,labhalo
     &   ,loadlab)
**************************************************************************
*** antiproton "confinement time" at the position (R,z)
***    (NOTE: at the moment only z=0 is implemented)
*** for an axisymmetric primary source assumed to extend over the
*** whole diffusion region, as specified by the function dspbdmasaxi
***
*** output in 10^15 s
***
*** the antiproton flux from wimp pair annihilation is obtained by
*** multiplying the result of this function by the factor:
***   unit_fact * 1/(4 pi) * vp * norm * (sigma v)/(2*mwimp**2) * dN/dtp
*** where:
***   vp = vp(tp) = antiproton velocity for given antiproton kinetic
***      energy tp in unit of c
***   norm = normalization of dspbdmasaxi = (with standard definition)
***      (rho(R,z))**2 = halo density at (R,z) squared in GeV^2 cm^-6
***   sigma v = dssigmav0tot in cm^3 s^-1
***   mwimp = WIMP mass in GeV
***   dN/dtp = dN/dtp(tp) = number of antiproton within the interval
***      (tp,tp+dtp) of kinetic energy tp (the latter in GeV) in GeV^-1
***   unit_fact = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV^2 cm^-6 cm^3 s^-1 GeV^-3
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
*** for dark matter particle decays, the antiproton flux is obtained by
*** multiplying the result of this function by the factor:
***   unit_fact * 1/(4 pi) * vp * norm * (dec-rate)/mwimp * dN/dtp
*** having defined (as opposed to the previous case):
***   norm = normalization of dspbdmasaxi = (with standard definition)
***      rho(R,z) = halo density at (R,z) in GeV cm^-3
***   dec-rate = dsdecratewimp = dark matter particle decay rate in s^-1
***   mwimp = decaying particle mass in GeV
***
*** input:
***   R = radial coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***   z = vertical coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***      NB: only z=0 works at the moment
***   tp = antiproton kinetic energy tp in GeV
```

```
***   powerin = integer selecting annihilations (=2) or decays (=1)
***   isin = propagation model option : isin=1 -> model with delta
***      function approximation for the gas disc; isin=2 -> two-zone
***      model with finite thickness for the gas disk
***   prec = value setting the truncation of the sum in the following
***      way: the sum oscillates around a mean value; a series of
***      estimated mean values is computed with a method that is a
***      slight variant of the method proposed in the PhD Thesis of
***      Torsten Bringmann, pag. 72, Eq. 5.33.
***      Educated guess: set, e.g., prec=1.d-3
***   nprec = number of zeros within which the convergence in the sum
***      is required; if it does not happen, the code stops. This is a
***      flag preventining the code to spend too much time within this
***      routine for propagation or halo models for which the form in
***      which the analytic solution is not converging. nprec cannot be
***      lower than 20. educated guess: set, e.g., nprec=80 for prec=1.d-3
***   labhalo = halo model access label, within the set of models
***             defined in the halo model repository
***   loadlab = logical variable: in case it is set to true the halo
***     labhalo is selected within this function, otherwise it is
***     assumed that it has been loaded before linking to this function
***     such as in the function dspbdphidtaxi or dspbtdaxitab
***
*** other inputs from common blocks:
***   parameters for the propagation model
***   setup for dspbdmasaxi
***
**********************************************************************
```

dspbtdaxitab.f

```
      real*8 function dspbtdaxitab(R,z,tpin,powerin,isin,how,labhalo
     &   ,loadlab)
**********************************************************************
*** tabulated version of the pb "confinement time" dspbtdaxi
***
*** input variables:
***   R = radial coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***   z = vertical coordinate for the position at which the flux is
***      measured in kpc (cylindrical coordinate system)
***      NB: only z=0 works at the moment
***   tpin = antiproton kinetic energy tp in GeV
***   powerin = integer selecting annihilations (=2) or decays (=1)
***   isin = propagation model option : isin=1 -> model with delta
***      function approximation for the gas disc; isin=2 -> two-zone
***      model with finite thickness for the gas disk
***   how = 1 - calculate the time only for requested tpin
***         2 - table is tabulated on first call, and then
***             interpolated
***         3 - as 2, but also write the table to disk at the
***             first call
***         4 - read table from disk on first call, and use the
***             subsequent calls. If the file does not exist, it
***             will be created (as in 3). (use as default)
***     the tabulation is created with dspbtdaxi with nprec and prec as
***     set by the corresponding pbprec and pbnprec in common blocks.
***     you need to load a new table whenever:
***       - the halo model is changed
***       - the propagation model is changed
***       - R, z or isin are changed
***     if pbtdaxiloadck=.true. there is an automatic check on these
***     and tables are eventually reloaded
***   labhalo = halo model access label, within the set of models
***             defined in the halo model repository
```

```
***   loadlab = logical variable: in case it is set to true the halo
***     labhalo is selected within this function, otherwise it is
***     assumed that it has been loaded before linking to this function
***     such as in the function dspbdphidtaxi
***
*** output: 10^15 s
**************************************************************************
```

Chapter 14

cr_gamma: Cosmic rays – Gamma fluxes

14.1 Gamma rays from the halo – theory

Among the yields of decay or pair annihilations of halo dark matter particles, the role played by gamma-rays could be a major one. Unlike the cases involving charged particles, for gamma-rays it is straightforward to relate the distribution of sources and the expected flux at the earth. Most flux estimated can be obtained just by summing over the contributions along lines of sight (or better, geodesics): gamma-rays have a low enough cross section on gas and dust and therefore the Galaxy is essentially transparent to them (except perhaps in the innermost part, very close to the region where a massive black hole is inferred); absorption by starlight and infrared background becomes effecient only for very far away sources (redshift larger than about 1).

It follows that in case the gamma-ray signal is detectable, this might be the only chance for mapping the fine structure of a dark halo, with a much better resolution for inhomogeneities (clumps) with respect to what is achievable through dynamical measurements or lensing effects. This is especially true for annihilating dark matter. Turning the latter argument around, if the fine structure of the Galactic halo is clumpy, or if a large density enhancement is present towards the Galactic center, as seen in N-body simulations of dark matter halos, this dark matter detection technique may be more promising than indicated by the estimates in which smooth non-singular halo scenarios are considered (recall that the fluxes per unit volume are proportional to the square of the dark matter density locally in space).

Several targets have been discussed as sources of gamma-rays from the annihilation of dark matter particles. An obvious source is the dark halo of our own galaxy [40, 41, 42, 43] and in particular the Galactic center, as the dark matter density profile is expected, in most models, to be picked towards it, possibly with huge enhancements close to te central black hole. The Galactic center is an ideal target for both ground- and space-based gamma-ray telescopes. As satellite experiments have much wider field of views and will provide a full sky coverage, they can in principle test the hypothesis of gamma-rays emitted in clumps of dark matter which may be present in the halo [44, 45, 46, 18, 47, 48]. Another possibility which has been considered is the case of gamma-ray fluxes from external nearby galaxies [49]. Furthermore, it has been proposed to search for an extragalactic flux originated by all cosmological annihilations of dark matter particles [50, 51, 52]. DarkSUSY is suitable to compute the gamma-ray flux from all these (and possibly other) sources.

14.2 Continuous gamma yields and line signals

The bulk of the gamma-ray yield from DM annihilation or decay arises from quark jets that when they hadronize/decay give rise to neutral pions which decay to gamma rays. At loop level, it could also be possible to produce monochromatic gamma rays, which could be a smoking gun for dark matter searches. The advantage with the gamma-ray lines is the distinctive spectral signature, which has no plausible astrophysical counterpart.

Compared to the monochromatic flux, the gamma-ray flux produced in π^0 decays is much larger but has less distinctive features. The photon spectrum in the process of a pion decaying into 2γ is, independent of the pion energy, peaked at half of the π^0 mass, about 70 MeV, and symmetric with respect to this peak if plotted in logarithmic variables. Of course, this is true both for pions produced by DM and, e.g., for those generated by cosmic ray protons interacting with the interstellar medium.

When considered together with to the cosmic-ray induced Galactic gamma-ray background, the DM induced signal looks like a component analogous to the secondary flux due to nucleon nucleon interactions: it is drowned into the Bremsstrahlung component at low energy, while it may be the dominant contribution at energies above 1 GeV or so. In fact, if the exotic component is indeed significant an option to disentangle it would be to search for a break in the energy spectrum at about the WIMP mass, where the line feature might be present as well: while the maximal energy for a photon emitted in WIMP pair annihilations is equal to the WIMP mass, the component from cosmic ray protons extends to much higher energies, essentially with the same spectral index as for the proton spectrum (the role played by the third main background component, inverse Compton emission, has still to settled at the time being and may worsen the problem of discrimination against background).

Besides this (weak) spectral feature, another way to disentangle the dark matter signal may be to exploit a directional signature: data with a wide angular coverage should be analyzed to search for a gamma-ray flux component following the shape and density profile of the dark halo, including eventual contributions from clumps.

14.3 Fluxes

Given a density distribution of DM particles, we can define a source function that tells us how many particles that are produced per volume element per time unit (and per energy interval for differential yields) from either annihilation or decay. The DM signal ultimately only depends on the local injection rate of some stable (cosmic ray) particle f, per volume and energy,

$$\frac{d\mathcal{Q}(E_f, \mathbf{x})}{dE_f} = \sum_n \rho_\chi^n(\mathbf{x}) \left\langle \mathcal{S}_n(E_f) \right\rangle \ . \tag{14.1}$$

Here, ρ_χ is the DM density (of the respective component, in case of multi-component DM) and the ensemble average $\langle ... \rangle$ is taken over the DM velocities; in principle, it depends on the spatial location $\mathbf{x}$, but in many applications of interest this can be neglected.

The particle source terms $\mathcal{S}_n(E_f)$ encode the full information about the DM particle physics model. For a typical WIMP DM candidate, e.g., only the annihilation part ($n = 2$) contributes,

$$\mathcal{S}_2(E_f) = \frac{1}{N_\chi m_\chi^2} \sum_i \sigma_i v \frac{dN_i}{dE_f} \,, \tag{14.2}$$

where σ_i is the annihilation cross section of two DM particles into final state i and dN_i/dE_f is the resulting number of stable particles of type f per such annihilation and unit energy. N_χ is a symmetry factor that depends on the nature of DM; if DM is (not) its own antiparticle we have $N_\chi = 2$ ($N_\chi = 4$). More generally, taking into account that there may be an asymmetry between

DM and anti-DM, $n_+ - n_- > 0$ and that hence only the symmetric fraction $r = 2n_-/(n_+ + n_-)$ can annihilate, this factor is given by

$$N_\chi = 2\frac{n_+ n_-}{(n_+ + n_-)^2} = \frac{4}{r(2-r)}\,. \tag{14.3}$$

The right-hand side of Eq. (14.2) must be further integrated over $f(v)$, the velocity distribution of the *relative* velocity of the two dark matter particles, but in practice it is typically sufficient to evaluate it for $v = 0$.

For decaying DM, on the other hand, we have

$$\mathcal{S}_1(E_f) = \frac{1}{m_\chi}\sum_i \Gamma_i \frac{dN_i}{dE_f}\,, \tag{14.4}$$

where Γ_i denotes the partial decay widths. Let us stress, however, that Eq. (14.1) is much more general in that it encapsulates also DM that is *both* annihilating and decaying, multi-component SM, as well as DM models with an internal Z^n symmetry [53, 54, 55, 56].

For a telescope pointing in the direction ψ, the expected DM-induced differential flux in gamma rays or neutrinos – i.e. the expected number of particles per unit area, time and energy – from a sky-region $\Delta\psi$ is thus given by a line-of-sight integral

$$\frac{d\Phi}{dE} = \frac{1}{4\pi}\int_{\Delta\psi} d\Omega \int_{\text{l.o.s.}} d\ell\, \frac{d\mathcal{Q}}{dE}\,. \tag{14.5}$$

For an effectively point-like source at distance d, this line-of-sight integral simplifies to

$$\int_{\Delta\psi} d\Omega \int_{\text{l.o.s.}} d\ell\, \frac{d\mathcal{Q}}{dE} \longrightarrow \frac{1}{d^2}\int dV \frac{d\mathcal{Q}}{dE}\,, \tag{14.6}$$

where the integral is over the extention of the source (much smaller than d).

For decaying DM, the above line-of-sight integral always factorizes into the particle source term S_1 given in Eq. (14.4) and a term that only depends on the DM distribution,

$$\frac{d\Phi^{\text{dec}}}{dE d\Omega} = \frac{1}{4\pi} J^{\text{dec}} S_1\,, \qquad J^{\text{dec}} \equiv \int_{\text{l.o.s.}} d\ell\, \rho \tag{14.7}$$

For annihilating DM, the corresponding factorization strictly speaking *only* holds *if* the annihilation rate is independent of velocity:

$$\frac{d\Phi^{\text{ann}}}{dE d\Omega} = \frac{1}{4\pi} J^{\text{ann}} S_2\,, \qquad J^{\text{ann}} \equiv \int_{\text{l.o.s.}} d\ell\, \rho^2\,. \tag{14.8}$$

While notable exceptions exist (in particular for resonances [57], p-wave annihilation [58] and Sommerfeld-enhanced annihilation [59]), this is a commonly encountered situation and hence of general interest.

14.4 Gamma rays from the halo – routines

DarkSUSY provides the functions dscrgaflux_dec and dscrgaflux_v0ann that take J^{dec} (or J^{ann}) as input and return the fluxes given in Eqs. (14.8) and (14.7). Here, the subscript _v0ann refers to the fact that, for the purpose of those routines, S_2 is evaluated in the limit of vanishing relative velocity of the annihilating DM pair. While this is the only situation of practical interest in many DM models, future DarkSUSY versions will offer support for a full velocity dependence of this quantity. In analogy with dscrsource_line, the core library furthermore provides routines dscrgaflux_line_dec

and dscrgaflux_line_v0ann (and correspondingly for neutrinos) that return strength, width and location of *monochromatic* ('line') signals. The latter is a convenient change with respect to previous versions of the code, as one cannot generally know how many lines there are (this depends on the particle physics module).

If a halo label is available, one can also more conveniently call dsgafluxsph instead. This function directly returns the gamma-ray flux, for that halo, automatically calculating the required line-of-sight integrals and adding decaying and annihilating DM components depending on which particle model is initialized.

14.5 Routine headers – fortran files

dscrgaflux_dec.f

```
      real*8 function dscrgaflux_dec(egev,diff,jpsi_dec,xi,istat)
**********************************************************************
***   function dscrgaflux_dec gives the flux of gamma-rays from WIMP
***   annihilation in the halo, in the limit of zero relative velocity.
***
***   input: egev     - gamma-ray energy [in GeV]
***          diff     - dictates whether differential source term at egev (diff=1)
***                     or integrated source term above egev (diff=0) is returned
***          jpsi_dec - value of line-of sight integration over a solid angle
***                     (\int d\Omega\int ds rho [kpc sr GeV cm^-3],
***                     obtained e.g. with a call to dsomlosisph)
***          xi       - factor by which the DM density should be rescaled
***                     (obtained e.g. as ratio of calculated to measured relic density)
***
***
***   unit of return value: cm^-2 s^-1/ GeV^diff
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
**********************************************************************
```

dscrgaflux_line_dec.f

```
      real*8 function dscrgaflux_line_dec(n,egev,widthline,jpsi_dec,xi,istat)
**********************************************************************
***   function dscrgaflux_line_dec gives the flux of *monoenergetic*
***   gamma-rays from WIMP annihilation in the halo, in the limit of
***   zero relative velocity.
***
***   input:   n - returns the nth monochromatic gamma-ray signal
***                implemented for the particle physics mode that is linked to
***                [if called with n=0, the first line will be returned,
***                 and n be set to the total number of existing lines]
***
***   output: egev      - gamma-ray energy [in GeV]
***           widthline - signal width
***           jpsi_dec  - value of line-of sight integration over a solid angle
***                       (\int d\Omega\int ds rho [kpc sr GeV cm^-3],
***                       obtained e.g. with a call to dsomlosisph)
***           xi        - factor by which the DM density should be rescaled
***                       (obtained e.g. as ratio of calculated to measured
***                       relic density)
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***
```

```
***   unit of return value: cm^-2 s^-1
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
**********************************************************************
```

dscrgaflux_line_v0ann.f

```
      real*8 function dscrgaflux_line_v0ann(n,egev,widthline,jpsi,xi,istat)
**********************************************************************
***   function dscrgaflux_line_v0ann gives the flux of *monoenergetic*
***   gamma-rays from WIMP annihilation in the halo, in the limit of
***   zero relative velocity.
***
***   type : commonly used
***   desc : Flux of monoenergetic gamma-rays from annihilation in the halo,
***   desc : in the limit of zero relative velocity
***
***   input:   n - returns the nth monochromatic gamma-ray signal
***                implemented for the particle physics mode that is linked to
***                [if called with n=0, the first line will be returned,
***                 and n be set to the total number of existing lines]
***
***   output: egev      - gamma-ray energy [in GeV]
***           widthline - signal width
***           jpsi      - value of line-of sight integration over a solid angle
***                       (\int d\Omega\int ds rho^2 [kpc sr GeV^2 cm^-6],
***                        obtained e.g. with a call to dsomlosisph)
***           xi        - factor by which the DM density should be rescaled
***                       (obtained e.g. as ratio of calculated to measured
***                        relic density)
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***
***   unit of return value: cm^-2 s^-1
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
**********************************************************************
```

dscrgaflux_v0ann.f

```
      real*8 function dscrgaflux_v0ann(egev,diff,jpsi,xi,istat)
**********************************************************************
***   function dscrgaflux_v0ann gives the flux of gamma-rays from WIMP
***   annihilation in the halo, in the limit of zero relative velocity.
***
***   type : commonly used
***   desc : Flux of gamma-rays from annihilation in the halo, in the
***   desc : limit of zero relative velocity
***
***   input: egev   - gamma-ray energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          jpsi   - value of line-of sight integration over a solid angle
***                    (\int d\Omega\int ds rho^2 [kpc sr GeV^2 cm^-6],
***                     obtained e.g. with a call to dsomlosisph)
***          xi     - factor by which the DM density should be rescaled
***                   (obtained e.g. as ratio of calculated to measured relic density)
***
***
```

```
***   unit of return value: cm^-2 s^-1/ GeV^diff
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
**********************************************************************
```

dsgafluxsph.f

```
      real*8 function dsgafluxsph(egev,diff,xi,labhalo,psi,theta,istat)
**********************************************************************
***   function dsgafluxsph gives the flux of gamma-rays from DM decays
***   and/or annihilations in the halo (in the limit of zero relative
***   velocity) for a spherical dark matter profile
***
***   inputs:
***     egev  - gamma-ray energy [in GeV]
***     diff  - dictates whether differential source term at egev
***             (diff=1) or integrated source term above egev (diff=0)
***             is returned
***     xi    - factor by which the DM density should be rescaled
***             (obtained e.g. as ratio of calculated to measured relic
***              density)
***     labhalo = halo model access label, to load the halo model via
***         dsdmdselect_halomodel
***     psi = the angular offset between the pointing direction
***         and the direction of the center of the distributionin
***         (in rad)
***     theta = aperture of the acceptance cone (in rad; it defines
***         the solid angle within which line-of-sight-integrals are
***         performed assuming a step-function response from the
***         instrument; more options are available in dsomlosisph, use
***         that to compute line-of-sight-integrals in your main file
***         and compute fluxes using dscrgaflux_v0ann or
***         dscrgaflux_dec)
***
***   type : commonly used
***   desc : gamma-rays from decay/annihilation from halo specified by halo label
***
***
***   unit of return value: cm^-2 s^-1/ GeV^diff
***   author: Piero Ullio (ullio@sissa.it)
***   date: 2017
**********************************************************************
```

Chapter 15

cr_nu:
Cosmic rays – Neutrino fluxes

15.1 Neutrino fluxes from the halo – theory

Usually, the flux of neutrinos from annihilation of WIMPs in the Milky Way halo is too small to be detectable, but for some clumpy or cuspy models, it might be detectable. The calculation of the neutrino-flux follows closely the calculation of the continuous gamma ray flux, with the main addition that neutrino interactions close to the detector are also included. Hence, both the neutrino flux and the neutrino-induced muon flux can be obtained. The neutrino to muon conversion rate in the Earth can also be obtained.

15.2 Routine headers – fortran files

dscrmu_PMNS.f

```
************************************************************************
***   function dscrmu_PMNS returns the elements of the PMNS matrix for      ***
***   neutrino mixing.                                                      ***
***                                                                         ***
***   input: il   - lepton flavour index (il=1..3)                          ***
***          im   - mass eigenstate index (im=1..3)                         ***
***                                                                         ***
***   author: Torsten.Bringmann@fys.uio.no                                  ***
***   date: 2022-04-28                                                      ***
************************************************************************
      complex*16 function dscrmu_PMNS(il,im)
```

dscrmuflux_dec.f

```
      real*8 function dscrmuflux_dec(egev,diff,jpsi_dec,xi,istat)
*********************************************************************
***   function dscrmuflux_dec gives the flux of neutrino-induced mouns
***   from WIMP annihilation in the halo, in the limit of zero relative
***   velocity. The function returns the flux as measured by a detector
***   placed in ice.
***
***   input: egev     - muon threshold energy [in GeV]
***          diff     - dictates whether differential source term at egev (diff=1)
***                     or integrated source term above egev (diff=0) is returned
***          jpsi_dec - value of line-of sight integration over a solid angle
```

```
***                    (\int d\Omega\int ds rho [kpc sr GeV cm^-3],
***                    obtained e.g. with a call to dsomlosisph)
***          xi       - factor by which the DM density should be rescaled
***                    (obtained e.g. as ratio of calculated to measured relic density)
***
***
***   unit of return value: km^-2 yr^-1 / GeV^diff
***
***   WARNING: Presently, this function does not take into account oscillatons
***            (unlike the dscrnuflux... routines) !
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
**********************************************************************
```

dscrmuflux_v0ann.f

```
**************************************************************************
***   function dscrmuflux_v0ann gives the flux of neutrino-induced mouns    ***
***   from WIMP annihilation in the halo, in the limit of zero relative     ***
***   velocity. The function returns the flux as measured by a detector     ***
***   placed in ice.                                                        ***
***                                                                         ***
***   type : commonly used                                                  ***
***   desc : Neutrino-induced muon flux from WIMP annihilation in the halo  ***
***                                                                         ***
***   input: egev   - muon threshold energy [in GeV]                        ***
***          diff   - return differential source term at egev(diff=1) or    ***
***                   integrated source term above egev (diff=0)            ***
***          jpsi   - value of line-of sight integration over a solid angle ***
***                   (\int d\Omega\int ds rho^2 [kpc sr GeV^2 cm^-6],      ***
***                    obtained e.g. with a call to dsomlosisph)            ***
***          xi     - factor by which the DM density should be rescaled     ***
***                   (obtained e.g. as ratio of calculated to measured     ***
***                    relic density)                                       ***
***                                                                         ***
***                                                                         ***
***   unit of return value: km^-2 yr^-1 / GeV^diff                          ***
***                                                                         ***
***   WARNING: Presently, this function does not take into account oscillatons
***            (unlike the dscrnuflux... routines) !
***                                                                         ***
***   author: Torsten.Bringmann@fys.uio.no                                  ***
***   date: 2016-02-09                                                      ***
**************************************************************************
      real*8 function dscrmuflux_v0ann(egev,diff, jpsi,xi,istat)
```

dscrnuflux_dec.f

```
      real*8 function dscrnuflux_dec(egev,diff,jpsi_dec,xi,istat)
**********************************************************************
***   function dscrnuflux_dec gives the flux of muon neutrinos from WIMP
***   annihilation in the halo, in the limit of zero relative velocity.
***
***   input: egev     - neutrino energy [in GeV]
***          diff     - dictates whether differential flux at egev (diff=1)
***                     or integrated flux above egev (diff=0) is returned
***          jpsi_dec - value of line-of sight integration over a solid angle
***                     (\int d\Omega\int ds rho [kpc sr GeV cm^-3],
***                     obtained e.g. with a call to dsomlosisph)
***           xi      - factor by which the DM density should be rescaled
***                     (obtained e.g. as ratio of calculated to measured relic density)
***
```

```
***
***   unit of return value: cm^-2 s^-1 / GeV^diff
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
***   mod: 2019-11-15 (added oscillations)
**********************************************************************
```

dscrnuflux_line_dec.f

```
      real*8 function dscrnuflux_line_dec(n,egev,widthline,jpsi_dec,xi,istat)
**********************************************************************
***   function dscrnuflux_line_dec gives the flux of *monoenergetic*
***   muon neutrinos from WIMP annihilation in the halo, in the limit of
***   zero relative velocity.
***
***   input:   n - returns the nth monochromatic gamma-ray signal
***                implemented for the particle physics mode that is linked to
***
***   output: egev      - neutrino energy [in GeV]
***           widthline - signal width
***           jpsi_dec  - value of line-of sight integration over a solid angle
***                       (\int d\Omega\int ds rho [kpc sr GeV cm^-3],
***                        obtained e.g. with a call to dsomlosisph)
***           xi        - factor by which the DM density should be rescaled
***                       (obtained e.g. as ratio of calculated to measured
***                        relic density)
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***
***   unit of return value: cm^-2 s^-1
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
***   mod: 2019-11-15 (added oscillations)
**********************************************************************
```

dscrnuflux_line_v0ann.f

```
      real*8 function dscrnuflux_line_v0ann(n,egev,widthline,jpsi,xi,istat)
**********************************************************************
***   function dscrnuflux_line_v0ann gives the flux of *monoenergetic*
***   muon neutrinos from WIMP annihilation in the halo, in the limit of
***   zero relative velocity.
***
***   input:   n - returns the nth monochromatic gamma-ray signal
***                implemented for the particle physics mode that is linked to
***
***   output: egev      - neutrino energy [in GeV]
***           widthline - signal width
***           jpsi      - value of line-of sight integration over a solid angle
***                       (\int d\Omega\int ds rho^2 [kpc sr GeV^2 cm^-6],
***                        obtained e.g. with a call to dsomlosisph)
***           xi        - factor by which the DM density should be rescaled
***                       (obtained e.g. as ratio of calculated to measured
***                        relic density)
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***
```

```
***   unit of return value: cm^-2 s^-1
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
***   mod: 2019-11-15 (added oscillations)
**********************************************************************
```

dscrnuflux_v0ann.f

```
      real*8 function dscrnuflux_v0ann(egev,diff,jpsi,xi,istat)
**********************************************************************
***   function dscrnuflux_v0ann gives the flux of muon neutrinos from WIMP
***   annihilation in the halo, in the limit of zero relative velocity.
***
***   input: egev   - neutrino energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          jpsi   - value of line-of sight integration over a solid angle
***                     (\int d\Omega\int ds rho^2 [kpc sr GeV^2 cm^-6],
***                      obtained e.g. with a call to dsomlosisph)
***           xi    - factor by which the DM density should be rescaled
***                   (obtained e.g. as ratio of calculated to measured relic density)
***
***
***   unit of return value: cm^-2 s^-1 / GeV^diff
***
***   author: Torsten.Bringmann@fys.uio.no
***   date: 2016-02-09
***   mod: 2019-11-15 (added oscillations)
**********************************************************************
```

dscroscillation_simp.f

```
      real*8 function dscroscillation_simp(nui,phinusource)
**************************************************************************
***   function dscroscillation_simp returns the average neutrino flux of a  ***
***   given flavour at the surface of the earth, given the neutrino         ***
***   fluxes of all flavours at sorce, assuming a distant location in the   ***
***   galactic galo.                                                        ***
***                                                                         ***
***   input: nui         - return neutrino type (1- nue, 2- numu, 3-tau)    ***
***          phinusource - neutrino PLUS antineutrino fluxes at source      ***
***                        (all three flavours, in the same order)          ***
***                                                                         ***
***   unit of return value: same as input source fluxes                     ***
***                                                                         ***
***   author: Torsten.Bringmann@fys.uio.no                                  ***
***   date: 2019-11-15                                                      ***
***    mod: 2022-04-28 calculation of conversion probabilities directly     ***
***                    from neutrino mixing parameters                      ***
**************************************************************************
```

Chapter 16

cr_ps:
Cosmic rays – point sources

16.1 Cosmic Ray propagation for point sources

In this directory, we collect the relevant cosmic ray routines for DM point sources like ordinary DM clumps or mini-spikes around intermediate-mass black holes.

16.2 Routine headers – fortran files

dsdbtdps.f

```
      real*8 function dsdbtdps(x,y,z,x0,y0,z0,td,isin)
**********************************************************************
*** antideuteron "confinement time per annihilation volume" for a static
*** point source within which WIMP pair annihilations take place.
***
*** inputs:
***   x,y,z - position of the observer (kpc)
***   x0,y0,z0 - position of the source (kpc)
***   td - kinetic energy per nucleon (gev)
***   isin = propagation model option : isin=1 -> model with delta
***     function approximation for the gas disc; isin=2 -> two-zone
***     model with finite thickness for the gas disk; in both cases
***     the radial boundary conditions are neglected.
***
*** output: in kpc^-3 10^15 s
***
*** the antideuteron flux from wimp pair annihilation is obtained by
*** multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vd * rho2vol * (sigma v)/(2*mwimp**2) * dn/dtd
*** where:
***   vd = vd(td) = antideuteron velocity for given antideuteron kinetic
***      energy td in unit of c
***   rho2vol = int dr^3_s \rho_s^2(r_s) = integral over the source
***      volume of the wimp density squared in kpc^3 GeV^2 cm^-6
***   sigma v = dssigmav0tot in cm^3 s^-1
***   mwimp = WIMP mass in GeV
***   dn/dtd = dn/dtd(td) = antideuteron spectrum in GeV^-1
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV^2 cm^-6 cm^3 s^-1 GeV^-3
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
```

```
*** for dark matter particle decays, the antideuteron flux is obtained
*** by multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vd * rhovol * dec-rate/mwimp * dN/dtd
*** having defined (as opposed to the previous case):
***   rho2vol = int dr^3_s \rho_s(r_s) = integral over the source
***      volume of the source density in kpc^3 GeV cm^-3
***   dec-rate = dsdecratewimp  = dark matter particle decay rate in s^-1
***   mwimp = decaying particle mass in GeV
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV cm^-3 s^-1 GeV^-2
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
**********************************************************************
```

dsdbtdps_td.f

```
      real*8 function dsdbtdps_td(t,td)
**********************************************************************
*** antideuteron "confinement time per annihilation volume" for a
*** non-static point source within which WIMP pair annihilations take
*** place.
***
*** inputs:
***   it assumes that the observer is located at x,y,z=0
***     t - time of observation (10^15 s), having assumed t=0 for the
***         time at which the source is the closest to the observer
***     td - antideuteron kinetic energy per nucleon (gev)
***
*** output: in kpc^-3 10^15 s
***
*** the antideuteron flux from wimp pair annihilation is obtained by
*** multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vd * rho2vol * (sigma v)/(2*mwimp**2) * dn/dtd
*** where:
***   vd = vd(td) = antideuteron velocity for given antideuteron kinetic
***      energy td in unit of c
***   rho2vol = int dr^3_s \rho_s^2(r_s) = integral over the source
***      volume of the wimp density squared in kpc^3 GeV^2 cm^-6
***   sigma v = dssigmav0tot in cm^3 s^-1
***   mwimp = WIMP mass in GeV
***   dn/dtd = dn/dtd(td) = antideuteron spectrum in GeV^-1
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV^2 cm^-6 cm^3 s^-1 GeV^-3
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
*** for dark matter particle decays, the antideuteron flux is obtained
*** by multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vd * rhovol * dec-rate/mwimp * dn/dtd
*** having defined (as opposed to the previous case):
***   rho2vol = int dr^3_s \rho_s(r_s) = integral over the source
***      volume of the source density in kpc^3 GeV cm^-3
***   dec-rate = dsdecratewimp = dark matter particle decay rate in s^-1
***   mwimp = decaying particle mass in GeV
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV cm^-3 s^-1 GeV^-2
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
**********************************************************************
```

dsorbitps.f

```
**********************************************************************
*** subroutine dsorbitps sets the orbit of a point source.
***
```

```
***  type : REPLACEABLE
***
*** inputs:
***     t0 - time at which the position along the orbit is computed
***          (10^15 s)
***     it assumes that the observer is located at x,y,z=0
*** inputs through common blocks:
***     horbit - which type of orbit you are considering; currently
***     the available option is:
***     horbit=1 - motion on a straight line in the reference frame
***       in which vy=0 and y0 is constant = y0ini (kpc);
***       the trajectory is set by 3 extra parameters, which we define
***       to be the velocity in the x and z direction, vxin and
***       vzin (km s^-1) and the distance of the trajectory to the
***       observer, dminin (kpc) (of course,  dminin >= y0in);
***       time is normalized assuming t0=0 when the source is at
***       the distance dminin
***     horbit=2 - motion on a circular orbit in the galactic plane
***       with circular velocity vcirc and radius of the orbit radorb
***       assuming that the observer is, in galactic coordinates, at
***       x=rosb, y=z=0 (and that t0=0 corresponds to the source at the
***       position x0=radorb, y0=z0=0). the initial time ti is set
***       equal the time when the source is located at x0 = - radorb,
***       eventually a number of full orbits "norbit" before the
***       current orbit. for the output a coordinate change is made and
***       the observer is assumed to be at x,y,z=0
***
*** outputs:
***     x0,y0,z0 - position of the source (kpc) at t0
***     ti - time at which the source enters the diffusion region
***     tf - time at which the source gets out of the diffusion region
***          (10^15 s)
***
***  author: Piero Ullio
***  modified: Torsten Bringmann, 11/06/2015 (adapted to replaceable
***                                           function concept)
**************************************************************************
      subroutine dsorbitps(t0,x0,y0,z0,ti,tf)
```

dspbtdps.f

```
      real*8 function dspbtdps(x,y,z,x0,y0,z0,tp,isin)
**************************************************************************
*** antiproton "confinement time per annihilation volume" for a static
*** point source within which WIMP pair annihilations take place.
***
*** inputs:
***   x,y,z - position of the observer (kpc)
***   x0,y0,z0 - position of the source (kpc)
***   tp - antiproton kinetic energy (gev)
***   isin = propagation model option : isin=1 -> model with delta
***     function approximation for the gas disc; isin=2 -> two-zone
***     model with finite thickness for the gas disk; in both cases
***     the radial boundary conditions are neglected.
***
*** output: in kpc^-3 10^15 s
***
*** the antiproton flux from wimp pair annihilation is obtained by
*** multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vp * rho2vol * (sigma v)/(2*mwimp**2) * dn/dtp
*** where:
***   vp = vp(tp) = antiproton velocity for given antiproton kinetic
***      energy tp in unit of c
***   rho2vol = int dr^3_s \rho_s^2(r_s) = integral over the source
***      volume of the wimp density squared in kpc^3 GeV^2 cm^-6
```

```
***   sigma v = dssigmav0tot in cm^3 s^-1
***   mwimp = WIMP mass in GeV
***   dn/dtp = dn/dtp(tp) = antiproton spectrum in GeV^-1
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV^2 cm^-6 cm^3 s^-1 GeV^-3
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
*** for dark matter particle decays, the antiproton flux is obtained
*** by multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vd * rhovol * dec-rate/mwimp * dN/dtp
*** having defined (as opposed to the previous case):
***   rho2vol = int dr^3_s \rho_s(r_s) = integral over the source
***      volume of the source density in kpc^3 GeV cm^-3
***   dec-rate = dsdecratewimp = dark matter particle decay rate in s^-1
***   mwimp = decaying particle mass in GeV
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV cm^-3 s^-1 GeV^-2
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
**********************************************************************
```

dspbtdps_td.f

```
      real*8 function dspbtdps_td(t,tp)
**********************************************************************
*** antiproton "confinement time per annihilation volume" for a
*** non-static point source within which WIMP pair annihilations take
*** place.
***
*** inputs:
***   it assumes that the observer is located at x,y,z=0
***     t - time of observation (10^15 s), having assumed t=0 for the
***         time at which the source is the closest to the observer
***     tp - antiproton kinetic energy (gev)
***
*** output: in kpc^-3 10^15 s
***
*** the antiproton flux from wimp pair annihilation is obtained by
*** multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vp * rho2vol * (sigma v)/(2*mwimp**2) * dn/dtp
*** where:
***   vp = vp(tp) = antiproton velocity for given antiproton kinetic
***      energy tp in unit of c
***   rho2vol = int dr^3_s \rho_s^2(r_s) = integral over the source
***      volume of the wimp density squared in kpc^3 GeV^2 cm^-6
***   sigma v = dssigmav0tot in cm^3 s^-1
***   mwimp = WIMP mass in GeV
***   dn/dtp = dn/dtp(tp) = antiproton spectrum in GeV^-1
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV^2 cm^-6 cm^3 s^-1 GeV^-3
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
*** for dark matter particle decays, the antiproton flux is obtained
*** by multiplying the result of this function by the factor:
***   units * 1/(4 pi) * vp * rhovol * dec-rate/mwimp * dn/dtp
*** having defined (as opposed to the previous case):
***   rho2vol = int dr^3_s \rho_s(r_s) = integral over the source
***      volume of the source density in kpc^3 GeV cm^-3
***   dec-rate = dsdecratewimp = dark matter particle decay rate in s^-1
***   mwimp = decaying particle mass in GeV
***   units = factor taking into account units =
***    1.d15 s sr^-1 2.99792d10 cm s^-1 GeV cm^-3 s^-1 GeV^-2
***      = 2.99792d25 cm^-2 s^-1 GeV^-1 sr^-1
***
```

****

Chapter 17

dd: Direct detection

17.1 Direct detection – theory

Specific choices of nuclear structure functions can be selected by calling dsdd_set('sf', label), where the character variable label indicates the set of structure functions. For the default option ('best'), e.g., the code automatically picks the best currently available structure function (depending on the nucleus). This mean, in order Fourier-Bessel, Sum-of-Gaussians, Fermi, Lewin-Smith. The function returning the value of $\tilde{\sigma}_{\chi T}$ is to be provided by an interface function dsddsigma(v,Er,A,Z,sigij, ierr) residing in the particle physics module, where on input v=v, Er=E_R, A=A, Z=Z and on output the 27×27 array sigij contains the (partial) equivalent cross sections $\tilde{\sigma}_{ij}$ in cm^2 and the integer ierr contains a possible error code. The order of the entries in sigij corresponds to that of the independent nonrelativistic operators $\mathcal{O}_i$; for the first 11 entries, we use the same operators and convention as in Ref. [60], while for the last 16 entries we add the additional operators discussed in Ref. [61]. In particular, sigij(1,1) is the usual spin-independent cross section and sigij(4,4) is the usual spin-dependent cross section. In addition, the direct detection module in DarkSUSY provides utility functions that can be used in the computation of the cross section. For example, the subroutine dsddgg2sigma(v, er,A,Z,gg,sigij,ierr) computes the (partial) equivalent scattering cross sections $\tilde{\sigma}_{ij}$ for nucleus (A, Z) at relative velocity v and recoil energy E_R starting from values of the G_i^N constants in gg, with nuclear structure functions set by the previous call to dsdd_set. The actual nuclear recoil event rate as given in

$$\frac{dR}{dE_R} = \sum_T c_T \frac{\rho_\chi^0}{m_T m_\chi} \int_{v>v_{\rm min}} \frac{d\sigma_{\chi T}}{dE_R} \frac{f(\mathbf{v},t)}{v} d^3v \,, \tag{17.1}$$

finally, is computed by the function dsdddrde. The latter two functions are independent of the specific particle physics implementation and hence are contained in the core library. In the above expression, the sum runs over nuclear species in the detector, c_T being the detector mass fraction in nuclear species i. m_T is the nuclear (target) mass, and $\mu_{\chi T} = m_\chi m_T/(m_\chi + m_T)$ is the reduced DM–nucleus mass. Furthermore, ρ_χ^0 is the local DM density, $\mathbf{v}$ the DM velocity relative to the detector, $v = |\mathbf{v}|$, and $f(\mathbf{v}, t)$ is the (3D) DM velocity distribution. In order to impart a recoil energy E_R to the nucleus, the DM particle needs a minimal speed of $v_{\rm min} = \sqrt{M_T E_R/2\mu_{\chi T}^2}$.

17.2 Cosmic-ray upscattered dark matter

.

17.3 Routine headers – fortran files

dsdd_init.f

```
      subroutine dsdd_init
c...  initialize the dd routines
```

dsdd_set.f

```
      recursive subroutine dsdd_set(key,value)
c...
c...  desc : Set parameters for scattering cross sections
c...  c - character string specifying choice to be made
c...author: paolo gondolo 2000-07-07
c...modified by Gintaras Duda 2007-06-27 for new FF options
c...modified by Paolo Gondolo 2008-02-18
c...modified by Torsten Bringmann 2014-12-10: removed model-specific part
c...modified by Paolo Gondolo 2016-02-06: completely removed mssm part
c...modified by Paolo Gondolo 2016-11-19: added dsddsf option with values sisd, eft (make better later)
c...modified by Paolo Gondolo 2016-11-20: cleaned up the options
```

dsdddrde.f

```
      subroutine dsdddrde(rho0,t,e,n,a,z,stoich,rsi,rsd)!,modulation)
c-----------------------------------------------------------------------
c
c  type : commonly used
c  desc : Differential WIMP-nucleus recoil rates
c
c  differential recoil rate
cc
c  input:
c    e : real*8              : nuclear recoil energy in keV
c    n : integer             : number of nuclear species
c    a : n-dim integer array : mass numbers
c    z : n-dim integer array : atomic numbers
c    stoich : n-dim integer array : stoichiometric coefficients
c    rho0 : local density of *active* DM particle candidate[GeV/cm^3]
c    t : real*8 : time in days from 12:00UT Dec 31, 1999
c    modulation : integer : 0=no modulation 1=annual modulation
c      with no modulation (ie without earth velocity), t is irrelevant
c      [NOT CURRENTLY USED]
c  output:
c    rsi : real*8 : spin-independent differential rate
c    rsd : real*8 : spin-dependent differential rate
c  units: counts/kg-day-keV
c
c  NB: This assumes that neutralinos make up 100% of the local DM density
c      If not, rsi and rsd need to be multiplied (by hand) by the fraction
c      of local DM that consists of neutralinos!
c
c  author: paolo gondolo (paolo@physics.utah.edu) 2004
c
c  2018-01-19: removed rescaling factor for local neutralino density (rhox)
c              from common block [TB]
c  2020-04-27: made rho0 input parameter [TB]
c=======================================================================
```

dsddeta.f

```
      subroutine dsddeta(vmin,t,eta)
c-----------------------------------------------------------------------
```

```
c  eta function entering the differential rate: eta = \int {f(v)/v} d^3 v
c
c  Truncated Maxwellian.
c
c  input:
c    vmin : minimum velocity to deposit energy e, in km/s
c           vmin=sqrt(M*E/2/mu^2)
c    t : time, in fraction of the year
c  output:
c    eta : in (km/s)^{-1}
c  authors: paolo gondolo (paolo@physics.utah.edu) 2004
c           piero ullio (ullio@sissa.it) 2004
c
c  2010/06/17 [C Savage]: corrected error in iso formula;
c                         added z < y case
c=======================================================================
```

dsddfffb.f

```
      subroutine dsddfffb(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F(q^2) for a charge density expressed as a
c     Fourier-Bessel series.
c
c     Parameters and equations found in:
c     H.DeVries, C.W.DeJager and C.DeVries, At. Data and Nucl.
c         Data Tables 36 (1987) 495.
c     Duda, Kemper, and Gondolo JCAP 0704:012,2007.
c
c     Authors: Gintaras Duda gkduda@creighton.edu (2007-06-27)
c              (w/ students Ann Kemper and George Reifenberger)
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

dsddfffermi.f

```
      subroutine dsddfffermi(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F^2(q)
c
c     F(q) is Helm type form factor computed using
c     R1 from Lewin and Smith with s = 0.9 fm and with a and c
c     directly from the Two-Parameter Fermi distribution in table VIII
c     in Fricke et al. (instead of using the least squares fit from
c     Table IIIa in Lewin and Smith).
c
c     Parameters and Equations found in:
c     Lewin J D and Smith P F 1996 Astropart. Phys. 6 87-112.
c     Fricke G et al. 1995 Atomic Data and Nuclear Data Tables 60 177-285.
c     Duda, Kemper, and Gondolo JCAP 0704:012,2007.
c
c
c     Authors: Gintaras Duda gkduda@creighton.edu (2007-06-27)
c              (w/ students George Reifenberger and Katherine Garrett)
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
```

```
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------
```

dsddffgauss.f

```
      subroutine dsddffgauss(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F^2(q)
c
c     F(q) is the gaussian form factor
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------
```

dsddffgould.f

```
      subroutine dsddffgould(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F^2(q)
c
c     F(q) is the old obsolete (incorrect) form factor from Gould
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------
```

dsddffh41.f

```
      subroutine dsddffh41(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F^2(q)
c
c     F(q) is Helm type form factor computed using
c      as in DarkSUSY 4.1 (with corrected expansion of f, PG 20080216)
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------
```

dsddffism.f

```
      subroutine dsddffism(q,a,z,l2jjpp,l2jjnn,l2jjpn,ierr)
c_____________________________________________________________________
c  Spin-dependent structure functions for direct detection.
c  Full interacting shell model (ISM) calculations
c  input:
c    q : real*8  : momentum transfer in GeV ( q=sqrt(2ME) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    l2jjnn, l2jjpp, l2jjpn : the factors \lambda^2 J (J+1) made functions
c       of the momentum transfer q, for pp, nn, and pn; they are
c       related to Spp, Snn, and Spn through
c           S = \lambda^2 J (J+1) (2J+1)/\pi
c       In turn, Spp, Snn, and Spn are related to the spin-dependent
c       structure functions S_{00}(q), S_{01}(q), S_{11}(q) defined in
c       Engel (PLB264,114,1991) via
c           Spp = S00+S11+S01
c           Snn = S00+S11-S01
c           Spn = 2*(S00-S11)
c  author: paolo gondolo (paolo@physics.utah.edu) 2008
c  modified: Joakim Edsjö, edsjo@fysik.su.se. Corrected one numerical
c    error and added failsafe for large y where polynomial fits
c    are bad
c=====================================================================
```

dsddffls.f

```
      subroutine dsddffls(q,a,z,ff,ierr)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Calculates F^2(q)
c
c     F(q) is Helm type form factor computed using
c     R1 from Lewin and Smith with s = 0.9 fm and with a and c
c     using the least squares fit from Table IIIa in Lewin and Smith.
c
c     Parameters and Equations found in:
c     Lewin J D and Smith P F 1996 Astropart. Phys. 6 87-112.
c     Fricke G et al. 1995 Atomic Data and Nuclear Data Tables 60 177-285.
c     Duda, Kemper, and Gondolo JCAP 0704:012,2007.
c
c
c     Authors: Gintaras Duda gkduda@creighton.edu (2007-06-27)
c              (w/ students George Reifenberger and Katherine Garrett)
c
c     Input: a: Mass number of Element
c            z: Atomic Number of Element
c            q: Momentum transfer in GeV
c
c     Output: ff: Form Factor squared (normalized to F^2(0) = 1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------
```

dsddffoddg.f

```
      subroutine dsddffoddg(q,a,z,l2jjpp,l2jjnn,l2jjpn,ierr)
c_____________________________________________________________________
c  Spin-dependent structure functions for direct detection.
c  Odd group model calculations.
c  With or without gaussian form factor.
c  input:
```

```
c    q : real*8  : momentum transfer in GeV ( q=sqrt(M*E/2/mu^2) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    l2jjnn, l2jjpp, l2jjpn : the factors \lambda^2 J (J+1) made functions
c       of the momentum transfer q, for pp, nn, and pn; they are
c       related to Spp, Snn, and Spn through
c           S = \lambda^2 J (J+1) (2J+1)/\pi
c       In turn, Spp, Snn, and Spn are related to the spin-dependent
c       structure functions S_{00}(q), S_{01}(q), S_{11}(q) defined in
c       Engel (PLB264,114,1991) via
c           Spp = S00+S11+S01
c           Snn = S00+S11-S01
c           Spn = 2*(S00-S11)
c  author: paolo gondolo (paolo@physics.utah.edu) 2008
c=======================================================================
```

dsddffsd.f

```
      subroutine dsddffsd(q,a,z,l2jjpp,l2jjnn,l2jjpn,ierr)
c-----------------------------------------------------------------------
c  Spin-dependent structure functions for direct detection.
c  input:
c    q : real*8  : momentum transfer in GeV ( q=sqrt(M*E/2/mu^2) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    l2jjnn, l2jjpp, l2jjpn : the factors \lambda^2 J (J+1) made functions
c       of the momentum transfer q, for pp, nn, and pn; they are
c       related to Spp, Snn, and Spn through
c           S = \lambda^2 J (J+1) (2J+1)/\pi
c       In turn, Spp, Snn, and Spn are related to the spin-dependent
c       structure functions S_{00}(q), S_{01}(q), S_{11}(q) defined in
c       Engel (PLB264,114,1991) via
c           Spp = S00+S11+S01
c           Snn = S00+S11-S01
c           Spn = 2*(S00-S11)
c  author: paolo gondolo (paolo@physics.utah.edu) 2004
c  modified: pg 040605 switched s01 and s11 in Na-23
c  modified: pg 080217 changed to \lambda^2 J (J+1)
c=======================================================================
```

dsddffsi.f

```
      subroutine dsddffsi(q,a,z,ff,ierr)
c-----------------------------------------------------------------------
c  Spin-independent form factor for direct detection.
c  input:
c    q : real*8  : momentum transfer in GeV ( q=sqrt(2*M*E) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    ff : |F(q)|^2, the square of the form factor
c  author: paolo gondolo (paolo@physics.utah.edu) 2004-2008
c=======================================================================
```

dsddffsimsd.f

```
      subroutine dsddffsimsd(q,a,z,l2jjpp,l2jjnn,l2jjpn,ierr)
c-----------------------------------------------------------------------
c  Spin-dependent structure functions for direct detection.
c  Simplified estimates where no more accurate model descriptions exist,
c  like for isotopes with two odd neucleons. This routine is based on the
```

```
c  the routines dsddoddgff, which radii from Ellis and Flores where
c  applicable for similiar isotopes, but spin part estimated rather crudely.
c  With or without gaussian form factor.
c  input:
c    q : real*8  : momentum transfer in GeV ( q=sqrt(M*E/2/mu^2) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    l2jjnn, l2jjpp, l2jjpn : the factors \lambda^2 J (J+1) made functions
c       of the momentum transfer q, for pp, nn, and pn; they are
c       related to Spp, Snn, and Spn through
c           S = \lambda^2 J (J+1) (2J+1)/\pi
c       In turn, Spp, Snn, and Spn are related to the spin-dependent
c       structure functions S_{00}(q), S_{01}(q), S_{11}(q) defined in
c       Engel (PLB264,114,1991) via
c           Spp = S00+S11+S01
c           Snn = S00+S11-S01
c           Spn = 2*(S00-S11)
c  author of original routine: paolo gondolo (paolo@physics.utah.edu) 2008
c  author of this modified version: Joakim Edsjo (edsjo@fysik.su.se) 2010
c=======================================================================
```

dsddffsog.f

```
      subroutine  dsddffsog(q,a,z,ff,ierr)
No header found.
```

dsddffspsm.f

```
      subroutine dsddffspsm(q,a,z,l2jjpp,l2jjnn,l2jjpn,ierr)
c-----------------------------------------------------------------------
c  Spin-dependent structure functions for direct detection.
c  Single particle shell model calculations.
c  With or without gaussian form factor.
c  input:
c    q : real*8  : momentum transfer in GeV ( q=sqrt(M*E/2/mu^2) )
c    a : integer : mass number
c    z : integer : atomic number
c  output:
c    l2jjnn, l2jjpp, l2jjpn : the factors \lambda^2 J (J+1) made functions
c       of the momentum transfer q, for pp, nn, and pn; they are
c       related to Spp, Snn, and Spn through
c           S = \lambda^2 J (J+1) (2J+1)/\pi
c       In turn, Spp, Snn, and Spn are related to the spin-dependent
c       structure functions S_{00}(q), S_{01}(q), S_{11}(q) defined in
c       Engel (PLB264,114,1991) via
c           Spp = S00+S11+S01
c           Snn = S00+S11-S01
c           Spn = 2*(S00-S11)
c  author: paolo gondolo (paolo@physics.utah.edu) 2008
c  modified: tb 2016 (bug fixes real to integer conversion)
c=======================================================================
```

dsddg2sigma.f

```
      subroutine dsddg2sigma(mwimp,swimp,v,er,a,z,gg,sigij,ierr)
c-----------------------------------------------------------------------
c  Calculate the unpolarized *equivalent* scattering cross section
c      sigma = E_{max} d\sigma/dE_R
c  The calculation depends on the definition of the couplings G_a, i.e.,
c    the dd scheme,
c      sigma = \sum_{ij} sig_{ij}
c    where sig_{ij} = \sum_{NN'} G_i^{N*} G_j^{N'} P_{ij}^{NN'}.
```

```
c  input:
c    mwimp : real*8: WIMP mass in GeV
c    swimp : real*8: WIMP spin in hbar
c    v   : real*8  : WIMP-nucleus relative velocity in km/s
c    er  : real*8  : nucleus recoil energy in keV
c    a,z : integer : nucleus mass number and atomic number, respectively
c    gg  : real*8 array gg(27,2)  : list of couplings G_i^N
c  output:
c    sigij : real*8  : sig_{ij} in cm^2
c    ierr  : integer : error flag (0=no error)
c              1-10  : general error
c              11-20 : error in SI part
c              21-30 : error in SD part
c              31-40 : error in SI and SD part
c     Note: if there is an error in any part, thos cross sections are
c     put to zero.
c
c
c  20161113 Paolo Gondolo   first version
c  20161119 Paolo Gondolo   second version
c=======================================================================
```

dsddhelp.f

```
      subroutine dsddhelp
c...
c...  type : commonly used
c...  desc : help with options for scattering cross sections
c...author: paolo gondolo 2000-07-07
c...modified by Gintaras Duda 2007-06-27 for new FF options
c...modified by Paolo Gondolo 2008-02-18
c...modified by Torsten Bringmann 2014-12-10: removed model-specific part
c...modified by Paolo Gondolo 2016-02-06: completely removed mssm part
c...modified by Paolo Gondolo 2016-11-20: split from dsdd_set
```

dsddlim.f

```
      function dsddlim(mx,iexp)
c
c     limits on scattering cross section on nucleons from
c     direct dark matter searches
c
c     input: mx - wimp mass
c            iexp - experiment
c                   1 = future cawo_4 cresst
c                   2 = future genius
c                   3 = dama 1997, plb389, 757
c     output: dsddlim - upper limit or sensitivity limit on wimp-nucleon
c                       spin-independent cross section in pb
c                       (returns 10^99 if no limit)
c
c     author: paolo gondolo 1999
c
```

dsddlimits.f

```
***********************************************************************
*** function dsddlimits gives the limits on f*sigma as a function
*** of WIMP mass. f is the halo fraction of dm (f=1 for 0.3
*** gev/cm^3) and sigma is the cross section in pb.
*** input: mx (wimp mass in gev)
***        type (1=spin-independent, 2=spin-dependent)
*** output: limit on f*sigma (pb)
```

```
*** based upon r. bernabei et al, plb 389 (1996) 757.
*** author: j. edsjo
*** date: 98-03-19
**********************************************************************

      real*8 function dsddlimits(mx,type)
```

dsddpp.f

```
      subroutine dsddpp(swimp,q,a,z,i,j,pp,ierr)
c----------------------------------------------------------------------
c  Structure functions for direct detection.
c  input:
c    swimp   : real*8: WIMP spin in hbar
c    [removed] vperpsq : real*8  : vperp^2 in c^2
c    q       : real*8  : momentum transfer in GeV ( q=sqrt(2*M*E) )
c    a       : integer : mass number
c    z       : integer : atomic number
c    i       : integer : index of P_{ij}^{NN'}
c    j       : integer : index of P_{ij}^{NN'}
c  output:
c    pp : complex*16 : array pp(2,2) containing P_{ij}^{NN'}
c    ierr : integer : error flag (0=no error)
c
c  20161119 Paolo Gondolo   second version
c======================================================================
```

dsddvearth.f

```
      subroutine dsddvearth(t,vearth)
c
c speed of the earth relative to the galaxy in km/s at
c time t in days from 12:00 UT Dec 31, 1999
c
c formulas from Green, astro-ph/0304446, originally by Lewin and Smith
c
```

dsnucldindx.f

```
      function dsnucldindx(a,z)
No header found.
```

dsz_of.f

```
      subroutine dsz_of(elem,z)
c...return the charge Z of the chemical element with symbol elem
c...input:
c...  elem - character string specifying the chemical symbol
c...author: paolo gondolo 2008-02-17
```

Chapter 18

dd_crdm: Direct detection – cosmic ray upscattered DM

18.1 Upscattering of dark matter by cosmic rays

DM particles move at non-relativistic velocities in the Galaxy, implying that the kinetic energy of DM particles with sub-GeV masses is not sufficient to trigger nuclear recoil energies in the keV range, as needed by classical direct detection facilities. However, as long as the elastic scattering cross section between DM and nuclei does not vanish, there must also be an irreducible local flux component of (semi-)relativistic DM particles, resulting from high-energy cosmic-ray (CR) particles scattering on DM particles initially (almost) at rest [23]. As a result, even very light DM is accessible to direct detection (and neutrino) experiments.

Here we briefly review how CRDM is produced, attenuated in the overburden and produces recoil rates in direct detection or neutrino experiments. The implementation of the corresponding routines in DarkSUSY is based on Refs. [23, 24, 62] and explained in Sec. 18.2.

Production

We consider CR nuclei N, with a flux of $d\Phi_N/dT_N$, that elastically scatter on non-relativistic DM particles χ in the Galaxy. For DM with a mass m_χ and density profile $\rho_\chi(\mathbf{r})$, at Galactic position $\mathbf{r}$, this induces a relativistic CRDM flux incident on Earth (at the top of the atmosphere) of

$$\frac{d\Phi_\chi}{dT_\chi} = \int \frac{d\Omega}{4\pi} \int_{\text{l.o.s.}} d\ell \, \frac{\rho_\chi}{m_\chi} \sum_N \int_{T_N^{\min}}^{\infty} dT_N \, \frac{d\sigma_{\chi N}}{dT_\chi} \frac{d\Phi_N}{dT_N} \tag{18.1}$$

$$\equiv D_{\text{eff}} \frac{\rho_\chi^{\text{local}}}{m_\chi} \sum_N \int_{T_N^{\min}}^{\infty} dT_N \, \frac{d\sigma_{\chi N}}{dT_\chi} \frac{d\Phi_N^{\text{LIS}}}{dT_N} \, . \tag{18.2}$$

Here $d\sigma_{\chi N}/dT_\chi$ is the differential elastic scattering cross section for CR nuclei scattering on DM at rest, and T_χ is the resulting kinetic recoil energy. The minimal CR energy $T_N^{\min}$ for this to happen is given by

$$T_N^{\min} = \begin{cases} \left(\frac{T_\chi}{2} - m_N\right)\left[1 - \sqrt{1 + \frac{2T_\chi}{m_\chi}\frac{(m_N+m_\chi)^2}{(2m_N - T_\chi)^2}}\right] & \text{for } T_\chi < 2m_N \\ \sqrt{\frac{m_N}{m_\chi}}\,(m_N + m_\chi) & \text{for } T_\chi = 2m_N \\ \left(\frac{T_\chi}{2} - m_N\right)\left[1 + \sqrt{1 + \frac{2T_\chi}{m_\chi}\frac{(m_N+m_\chi)^2}{(2m_N - T_\chi)^2}}\right] & \text{for } T_\chi > 2m_N \end{cases} . \tag{18.3}$$

Finally, introducing the effective distance $D_{\rm eff}$ as above allows us to express the CRDM flux in the solar system in terms of the relatively well measured *local* interstellar CR flux, $d\Phi_N{}^{\rm LIS}/dT_N$, and the *local* DM density, $\rho_\chi^{\rm local}$.

Attenuation
Once entering the atmosphere and/or soil, the CRDM flux given by Eq. (18.2) will be attenuated due to scattering on the corresponding nuclei. This can be modelled by the energy loss equation

$$\frac{dT_\chi^z}{dz} = -\sum_N n_N \int_0^{\omega_\chi^{\rm max}} d\omega_\chi \, \frac{d\sigma_{\chi N}}{d\omega_\chi} \omega_\chi \, . \tag{18.4}$$

This equation relates the average kinetic energy at depth z, T_χ^z, to an initial energy T_χ at the top of the atmosphere ('$z = 0$'). Note that the sum here runs over the nuclei N in the overburden, i.e. no longer over the CR species, and ω_χ is the *energy loss* of a CRDM particle in a single collision. For elastic scattering, ω_χ equals by definition the nuclear recoil energy T_N and the maximal energy loss of a DM particle with initial kinetic energy T_χ^z is given by

$$\omega_\chi^{\rm max} = T_N^{\rm max} = \frac{2m_N}{s}\left[\left(T_\chi^z\right)^2 + 2m_\chi T_\chi^z\right], \tag{18.5}$$

where

$$s = (m_N + m_\chi)^2 + 2m_N T_\chi^z \tag{18.6}$$

is the (squared) CMS energy of the process. For inelastic scattering, on the other hand, the energy loss can be as high as $\omega_\chi^{\rm max} = T_\chi^z$ (see Ref. [62] for a detailed discussion).

Recoil rate in detector
The final recoil rate of CRDM particles, from their elastic scattering with nuclei in underground detectors like XENON, is given by

$$\frac{d\Gamma_N}{dT_N} = \int_{T_\chi^{\rm min}}^\infty dT_\chi \, \frac{d\sigma_{\chi N}}{dT_N} \frac{d\Phi_\chi}{dT_\chi} \, . \tag{18.7}$$

In this expression, the integration is performed over the energy of the DM particles *before* entering the atmosphere, while the elastic scattering cross section $d\sigma_{\chi N}/dT_N$ must be evaluated at the *actual* DM energy, T_χ^z, at the detector location; similarly, $T_\chi^{\rm min} = T_\chi(T_\chi^{z,{\rm min}})$ represents the minimal *initial* CRDM energy that is needed to induce a nuclear recoil of energy T_N *at depth* z. This requires a numerical solution of Eq. (18.4) to obtain $T_\chi^z(T_\chi)$, as well as its inverse.

In general, $d\sigma_{\chi N}/dT_N$ is a function of both s and the (spatial) momentum transfer,

$$Q^2 = 2m_N T_N \, . \tag{18.8}$$

If the (dominant) dependence on Q^2 factorizes – as, for example, in the case of standard form factors – then the rate in Eq. (18.7) features the *same* Q^2-dependence as that for the standard population of non-relativistic halo DM, allowing to directly re-interpret published limits on the latter into limits on the former [23].

18.2 CRDM– routines

The CRDM routines are initialized with a call to dsddcrdm_init; this is typically done directly from dsinit and sets default values for, e.g., $\rho_\chi^{\rm local}$ and $D_{\rm eff}$, as well as common block variables that steer accuracy settings and which channels (including inelastic scattering) to include. For typical applications, the functions of greatest interest are dsddDMCRflux and dsddDMCRcountrate, with detailed usage illustrated in the example programs examples/aux/DDCR_flux.f and examples/aux/DDCR_limits.f, respectively.

Central functions
The function dsddDMCRflux returns the CRDM flux before attenuation, as given by Eq. (18.2). This function takes the product $\rho_\chi^{\rm local} \cdot D_{\rm eff}$ directly as input, thus overriding the default values set in the initialization process described above. The sum over CR nuclei N is performed over the dominant CR species contributing to the CRDM flux, as returned by dscrISRflux (see section 12.1).

The other commonly used function, dsddDMCRcountrate, only takes the name of a specific experiment as input (for example Xenon1T [63], but also neutrino experiments such as Borexino [64]) and returns the recoil rate in Eq. (18.7) in dimensionless units – namely in units of the recoil rate that would correspond to the reported 95 *C.L.* limit of that experiment. Internally, this is essentially a driver routine for the more general function dsddDMCRdgammadt that returns the full recoil rate for a specific element.

In principle, the only interface function that a particle module needs to provide in order to access the CRDM routines is dsddsigma, returning the nuclear scattering cross section in the non-relativistic limit.* An important *replaceable* function, however, is dsddsigmarel which by default (i.e. the instance one can find in src/dd_crdm) simply returns unity – but which in general is assumed to return the ratio of the full (relativistic) elastic scattering cross section to the cross section in the non-relativistic limit. The example program examples/aux/DDCR_limits.f (along with makefile target DDCR_limits_q2) illustrates how to replace this function in practice, e.g. for interactions mediated by light scalar or vector particles.

Further functions of potential interest
Per default, the CRDM routines fully implement the effect of inelastic scattering – as modelled in Ref. [62] – on the attenuation of the CRDM flux in the overburden of an experiment. For that purpose, the subroutine dsddTDMattenuation handles the conversion between T_χ and T_χ^z; for code-speed-up, the solution to Eq. (18.4) is tabulated only once and then interpolated (with the full differential cross section $d\sigma_{\chi N}/d\omega_\chi$ provided by the function dsddDMCRsigsoil).
For neutrino telescopes based on liquid scintillators, furthermore, one needs to relate the proton recoil energy in the scintillator to an apparent electron T_e equivalent before one can compare Eq. (18.7) to reported limits (see Refs. [65, 23] for details). This conversion is handled by the function dsddDMCRquenching.

18.3 Routine headers – fortran files

dsddcrdm_init.f

```
      subroutine dsddcrdm_init
c...  initialize the crdm routines
```

dsddDMCRcountrate.f

```
************************************************************************
*** Function dsddDMCRcountrate compares the total event rate from nuclear   ***
*** recoils caused by high-energy DM particles (from their interaction with ***
*** cosmic rays, see Bringmann & Pospelov (2018)) to limits reported by     ***
*** various experiments.                                                    ***
***                                                                         ***
*** type : commonly used                                                    ***
*** desc : experimental sensitivity to cosmic-ray uppscattered dark matter  ***
***                                                                         ***
***  Output : *ratio* of expected count rates to reported sensitivity       ***
***                                                                         ***
*** In a sense, this is a collection of examples for how to call            ***
*** dsddDMCRdgammadt in order to get the differential countrate for a       ***
```

*This function also exists as an auxiliary function in src_models/common/aux, implying that a particle module in practice only needs to provide dsddgpgn for the nucleon couplings.

```
*** number of possible configurations. See each of the implementd          ***
*** options further down for further details.                               ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2019-02-02                                                        ***
*** mod  2019-03-15 merged with previous 'dsddDMCRcountrate_full', added    ***
***                 dsddDMCRsigtarget with options set here                ***
******************************************************************************
      real*8 function dsddDMCRcountrate(option)
```

dsddDMCRdgammadt.f

```
******************************************************************************
*** Function dsddDMCRdgammadt returns the differential rate of recoils      ***
*** for DM hitting a nucleus at rest. Here, the DM flux is assumed to be    ***
*** the one that results from CR interactions with DM particles in the      ***
*** halo, see Bringmann & Pospelov (2018).                                  ***
***                                                                        ***
***  Input:                                                                ***
***    Trnuc    - recoil energy of nucleus [GeV]                           ***
***    mN       - mass of target nucleus [GeV]                             ***
***    depth    - penetration depth (detector location) [cm]               ***
***    rhoDeff  - local DM density [GeV/cm**3] multiplied by the effective  ***
***               distance [kpc] out to which the DM flux is assumed to    ***
***               originate from:                                          ***
***               Deff = (\int dV \rho_DM \rho_CR /(4pi d^2)                ***
***                      / (\rho_DM \rho_CR)_local                         ***
***                                                                        ***
***  Output: d\Gamma/dTrnuc, units: [1/(s GeV)/nucleus]                    ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-06-24                                                        ***
*** mod  2019-03-13 removed scattering cross section as argument           ***
***                 (DMCR routines now use dsddDMCRsigtarget and           ***
***                  dsddsigmanucleon)                                     ***
*** mod  2019-10-04 changed argument zlfree -> depth                       ***
******************************************************************************
      real*8 function dsddDMCRdgammadt(Trnuc,mN,depth,rhoDeff)
```

dsddDMCRflux.f

```
******************************************************************************
*** Function dsddDMCRflux provides the local differential flux of DM       ***
*** particles that results from cosmic rays impinging on DM in the         ***
*** diffusive halo, see Bringmann & Pospelov (2018).                       ***
***                                                                        ***
***  Input:                                                                ***
***    Tkin     - kinetic energy of DM particle [GeV]                      ***
***    rhoDeff - local DM density [GeV/cm**3] multiplied by the effective  ***
***              distance [kpc] out to which the DM flux is assumed to     ***
***              originate from:                                           ***
***                Deff = (\int dV \rho_DM \rho_CR /(4pi d^2)               ***
***                       / (\rho_DM \rho_CR)_local                        ***
***                                                                        ***
***  Output: d\Phi/dT, units: 1/ [cm**2 s GeV]                             ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-06-22                                                        ***
***  mod 2019-03-10 (diff. scattering x-section now in dsddDMCRsigCR)      ***
***  mod tb, hk 2022-01-14 (added further elements)                        ***
******************************************************************************
      real*8 function dsddDMCRflux(Tkin, rhoDeff)
```

dsddDMCRquenching.f

```
************************************************************************
*** Subroutine dsddDMCRquenching returns the nuclear recoil rate Tr that    ***
*** corresponds to a given quenched energy Tquench, as wee as the           ***
*** derivative (dTr/dTquench) at the same quenched energy Tquench.          ***
*** A call to dsddDMCRquenching_set determines which quenching prescription ***
*** is used. See there for available options.                                ***
***                                                                         ***
***  Input:                                                                 ***
***    Tquench - quenched energy as observable at the detector              ***
***                                                                         ***
***  Output:                                                                ***
***    Tr - nuclear recoil energy that corresponds to Tquench               ***
***    dTrdTq - (dTr/dTquench) evalauted at Tquench                         ***
***                                                                         ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-06-27                                                         ***
************************************************************************
      subroutine dsddDMCRquenching(Tquench, Tr, dTrdTq)
```

dsddDMCRquenching_set.f

```
************************************************************************
*** Subroutine dsddDMCRquenching_set is used to steer which quenching       ***
*** factors should be used for the dsddDMCR routines. The default is no     ***
*** no quenching, i.e. the energy range that dsddDMCRcountrate receives as  ***
*** input refers directly to the nuclear recoil energy.                     ***
*** For other implemented options, see below. Note that most just rely on   ***
*** tabulated queching factors, which makes it straightforward to add new   ***
*** ones (by replacing this function and dsddDMCRquenching).                ***
***                                                                         ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-07-04                                                         ***
************************************************************************
      subroutine dsddDMCRquenching_set(option)
```

dsddDMCRsigCR.f

```
************************************************************************
*** Function dsddDMCRsigCR provides the differential scattering cross       ***
*** section of cosmic rays on DM, d\sigma_CR / dTkin.                       ***
***                                                                         ***
***  Input:                                                                 ***
***    Tkin    - kinetic energy of DM particle [GeV]                        ***
***              (recoil energy after scattering)                           ***
***    Tcr     - initial kinetic energy of CR particle [GeV]                ***
***    CRtype  - CR type (1=proton, 2=helium)                               ***
***                                                                         ***
***  Output:  d\sigma_CR / dTkin, units: cm**2/ GeV                         ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2020-10-30                                                         ***
*** mod 2021-12 (H.Kolesova): added more elements + form factor suppresion  ***
************************************************************************
      real*8 function dsddDMCRsigCR(Tkin,Tcr,CRtype)
```

dsddDMCRsigsoil.f

```
************************************************************************
*** Function dsddDMCRsigsoil provides the differential scattering cross     ***
```

```
*** section of a (potentially relativistic) DM on a target nucleus, taking  ***
*** into account both elastic and inelastic energy losses.                  ***
***                                                                         ***
***  Input:                                                                 ***
***    Tdm     - initial kinetic energy of DM particle [GeV]                ***
***    omega   - energy loss of DM particle [GeV]                           ***
***              (=recoil energy of nucleus only for elastic scattering)    ***
***    iel     - index for the nucleus that the DM particles are scattering ***
***              on (see list in dsnuclides.h)                              ***
***                                                                         ***
***  Output:  d\sigma_N / domega, units: cm**2/ GeV                         ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2022-05-05                                                         ***
*******************************************************************************
      real*8 function dsddDMCRsigsoil(Tdm,omega,iel)
```

dsddDMCRsigtarget.f

```
*******************************************************************************
*** Function dsddDMCRsigtarget provides the differential scattering cross   ***
*** section of a (potentially relativistic) DM on a target (detector)       ***
*** nucleus, d\sigma_N / dTr, where Tr is the nuclear recoil energy.        ***
***                                                                         ***
***  Input:                                                                 ***
***    Tr    - kinetic energy of nucleus [GeV]                             ***
***              (recoil energy after scattering)                           ***
***    Tdm   - initial kinetic energy of DM particle [GeV]                 ***
***                                                                         ***
***  'Hidden' input (common block flags):                                   ***
***    targetoption - determines which of the implemented concrete          ***
***                   target options should be used. Typically set in       ***
***                   dsddDMCRcountrate                                     ***
***                                                                         ***
***  Output:  d\sigma_N / dTr, units: cm**2/ GeV                            ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2019-03-10                                                         ***
*** mod  2019-10-30 (added light mediator examples)                         ***
*** mod  2022-05-05 removed soil scattering, added call to general sigma    ***
*******************************************************************************
      real*8 function dsddDMCRsigtarget(Tr,Tdm)
```

dsddlfreesimp.f

```
*******************************************************************************
*** Function dsddlfreesimp returns a simple estimate for the mean free      ***
*** path of strongly interacting DM through the Earth crust. We include     ***
*** contributions from the 11 most abundant elements, with densities as     ***
*** returned from dsddsoilcomp.                                             ***
***                                                                         ***
***  Input:                                                                 ***
***    sigsi - spin-independent cross section per nucleon [cm**2]           ***
***    depth - detector location below surface [cm]                         ***
***    how   - assume equal couplings of DM to protons and neutrons (how=1) ***
***            or assume DM only couples to protons (how=2)                 ***
***                                                                         ***
***  Output: *average* mean free path in cm, between surface and depth      ***
***          specified as input.                                            ***
***                                                                         ***
***  NB: If this routine is called directly, you first need to set          ***
***          'targetoption' (typically set in dsddDMCRcountrate) for the    ***
***          correct experimental location                                  ***
***                                                                         ***
```

```
*** author: Torsten.Bringmann.fys.uio.no                                  ***
*** date 2019-02-06                                                       ***
*** mod tb, 12/01/2021: added target location dependence                  ***
*****************************************************************************
      real*8 function dsddlfreesimp(sigsi, depth, how)
```

dsddnu_inel.f

```
*****************************************************************************
*** Function dsddnu_inel returns the inelastic scattering cross section of  ***
*** neutrinos on selected nuclei, as computed with GiBUU, normalized to the ***
*** low-energy elastic cross section on nucleons.                           ***
*** The return value is thus the factor I_{i,\nu} introduced in Alvey,      ***
*** Bringmann & Kolesova (2022); see there for further details.             ***
***                                                                         ***
***  Input:                                                                 ***
***    Tdm   - initial kinetic energy of neutrino [GeV]                     ***
***    omega - neutrino energy loss [GeV**2]                                ***
***    a     - atomic mass number of nucleus N                              ***
***    z     - element number of nucleus                                    ***
***    p     - subprocess:                                                  ***
***            0: all processes (total cross section)                       ***
***            1: only quasi-elastic scattering                             ***
***            2: only delta resonance                                      ***
***            3: only nuclear resonances (excluding delta)                 ***
***            4: only deep ineleastic scattering                           ***
***    ierr  - error code (0 on return if no error)                         ***
***                                                                         ***
***  Output:  ~ dsigma/domega / [G_F**2*(mp+mn)/(4*pi)], dimensionless      ***
***             (more precisely, the normalization is given by Eq. 4.4      ***
***              in Alvey+ '22)                                             ***
***                                                                         ***
***  NB: This funtion interpolates results obtained with GiBUU, tabulated   ***
***      up to energies of Tdm = 10 GeV. It is used in dsddDMCRsigsoil,     ***
***      where an extrapolation is adopted for large energies (as steered   ***
***      via a common block variable CRDM_inel_Eref set in dsdd_init).      ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2022-02-25                                                        ***
*****************************************************************************
      real*8 function dsddnu_inel(Tdm,omega,a,z,p,ierr)
```

dsddnu_inel_read.f

```
*****************************************************************************
*** subroutine dsddnu_inel_read reads in inelastic scattering cross         ***
*** sections of neutrinos on selected nuclei, as provided in                ***
*** J. Alvey, T.Bringmann & H. Kolesova (2020).                             ***
***                                                                         ***
*** Called by dsddnu_inel.                                                 ***
***                                                                         ***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no                 ***
*** Date: 2022-05-04                                                       ***
*****************************************************************************
      subroutine dsddnu_inel_read()
```

dsddreadQF.f

```
*****************************************************************************
*** Read in quenching factors and store in common block tables.             ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-07-04                                                        ***
```

```
************************************************************************
      subroutine dsddreadQF(filename1,filename2)
```

dsddsigmarel.f

```
************************************************************************
*** Function dsddsigmarel returns the full differential cross section for   ***
*** elastic scattering of DM on a nucleus, or vice versa, expressed in      ***
*** units of the same quantity in the highly non-relativistic limit.        ***
*** Everything is expressed in terms of Lorentz-invariant quantities, so    ***
*** no assumption is made about the frame in which the cross section is     ***
*** is evaluated.                                                           ***
***                                                                         ***
***  type : replaceable                                                     ***
***  desc : relativistic DM - nucleus scattering cross section              ***
***                                                                         ***
***  Input:                                                                 ***
***    Q2    - momentum transfer squared [GeV**2]                          ***
***    s     - CMS energy squared [GeV**2]                                 ***
***    mN    - mass of nucleus [GeV]                                       ***
***    nuc2S - twice the nuclear spin                                      ***
***            (i.e. 0 for scalar nuclei, 1 for spin 1/2 nuclei etc.)      ***
***                                                                         ***
***  Output:  \sigma_N(Q2) / \sigma_N(Q2)_NR, dimensionless                 ***
***           = (mN+mDM)**2/s * |M|^2 / |M|^2_NR                            ***
***           [The output thus does not contain the nuclear form factor]    ***
***                                                                         ***
***  NB: This is the default function that corresponds to the asumption of  ***
***      a *constant* cross section, where the return value by definition   ***
***      is 1d0. A particle module or main program can replace this         ***
***      function with any expression for the relativistic amplitude. For   ***
***      convenience, a few examples for contact interactions and light     ***
***      mediators are provided below (just comment in the corresponding    ***
***      block).                                                            ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date  : 2022-07-29                                                      ***
***    mod: added nuclear spin as argument (H. Kolesova)                    ***
************************************************************************
      real*8 function dsddsigmarel(Q2,s,mN,nuc2S)
```

dsddsoilcomp.f

```
************************************************************************
*** Function dsddsoilcomp gives the average number density of nucleons of   ***
*** mass number A per cm^3, in the overburden of a given experiment.        ***
***                                                                         ***
*** input:                                                                  ***
***    exploc - string indicating the experimental location                 ***
***          (For unknown exp, or exp='default', the average distribution   ***
***           in the Earth's crust from dssem_earthdenscomp is returned)    ***
***    Z   - atomic number. Implemented values:                             ***
***              Z    |  Element                                           ***
***             ------+----------                                          ***
***                6   |   C                                               ***
***                7   |   N                                               ***
***                8   |   O                                               ***
***               11   |   Na                                              ***
***               12   |   Mg                                              ***
***               13   |   Al                                              ***
***               14   |   Si                                              ***
***               15   |   P                                               ***
***               16   |   S                                               ***
```

```
***              18   |   Ar                                                ***
***              19   |   K                                                 ***
***              20   |   Ca                                                ***
***              24   |   Cr                                                ***
***              26   |   Fe                                                ***
***              28   |   Ni                                                ***
***                                                                          ***
*** Return: density in 1/cm^3                                               ***
***                                                                          ***
*** author: Torsten Bringmann, Helena Kolesova                              ***
*** date: 12/01/2022                                                        ***
*** updated: 17/06/2022 added atmospheric elements                          ***
***          04/11/2022 changed input A->Z                                  ***
******************************************************************************

      real*8 function dsddsoilcomp(exploc,z)
```

dsddTDMattenuation.f

```
******************************************************************************
*** Subroutine dsddTDMattenuation relates average DM kinetic energies       ***
*** before and after propagating through a dense medium                     ***
***                                                                          ***
***  Input:                                                                 ***
***    Tin    - initial kinetic energy  [GeV]                               ***
***    Tz     - average kinetic energy after scattering in medium [GeV]     ***
***    depth  - penetration depth (detector location) [cm]                  ***
***    how    - convert from Tin to Tz (how=1)                              ***
***             or from Tz to Tin (how=2)                                    ***
***                                                                          ***
***  Output:                                                                ***
***    Tin  - initial kinetic energy  [GeV]                                  ***
***    Tz    - average kinetic energy after scattering in medium [GeV]      ***
***                                                                          ***
***  NB: i) For how=1, the input value of Tz is overwritten on output       ***
***         For how=2, the input value of Tin is overwritten on output      ***
***      ii) If this routine is called directly, you first need to set      ***
***          'targetoption' (typically set in dsddDMCRcountrate) for the    ***
***          correct experimental location                                   ***
***                                                                          ***
*** WARNING: While this now should work for 'arbitrary' Q-dependence, it    ***
*** has only been tested for single mediators that are neither too light    ***
*** (<<MeV) nor too heavy (>>10 GeV). Since the numerical integration is    ***
*** potentially unstable, the behaviour of this routine must be carefully   ***
*** tested outside this regime and/or for more complicated energy           ***
*** dependendences.                                                         ***
***                                                                          ***
*** The default behaviour (full Q2-dependence, including inelastic          ***
*** scattering) can be changed by setting the common block flag             ***
*** 'attenuation_how' to 1 in dsdd_init. In that case, dsddTDMattenuation   ***
*** will NOT take into account a possible Q-dependence of the scattering    ***
*** cross section. Effectively assuming the cross section at zero momentum  ***
*** transfer, the code runs much faster in this case -- but typically       ***
*** over-estimates the stopping power for e.g. light mediators.             ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-07-07                                                         ***
*** mod tb, 31/10/2019: added full Q2 dependence; argument zlfree -> depth  ***
*** mod tb, 12/01/2021: added target location dependence                    ***
******************************************************************************
      subroutine dsddTDMattenuation(Tin, Tz, depth, how)
```

dsddTrmax.f

```
*************************************************************************
*** Function dsddTrmax returns the maximal (kinetic) recoil energy Trmax of ***
*** a target particle at rest, when hit by an incoming particle of kinetic  ***
*** energy Tkin. The maximal recoil energy is given in the lab frame (where ***
*** the target particle is initially at rest), with no assumption made      ***
*** concerning whether the incident particle is relativistic or not.        ***
*** For isotropic scattering (in the CMS), the actual recoil energy is      ***
*** equally distributed between 0 and the value returned by this function.  ***
***                                                                         ***
***  Input:                                                                 ***
***    Tkin    - kinetic energy of incident particle [GeV]                  ***
***    min     - mass of incident particle [GeV]                            ***
***    mtarget - mass of target particle [GeV]                              ***
***                                                                         ***
***  Units of output: [GeV]                                                 ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2018-06-22                                                         ***
*************************************************************************
      real*8 function dsddTrmax(Tkin, min, mtarget)
```

Chapter 19

dmd_astro: Astrophysical source functions

In the folder src/dmd_astro we provide a set of wrapper routines that all call the halo driver function dsdmsdriver in a specific way. The rest of the code communicates with dsdmsdriver, and hence the halo model(s), only via these functions and never by calling dsdmsdriver directly. The routines in src/dmd_astro therefore provide examples of functions that *cannot be replaced* individually in a consistent way, but only as a whole set (along with dsdmsdriver, in case the user wants to change the structure of the driver itself).

19.1 Routine headers – fortran files

dsdmasaxi.f

```
************************************************************************
***  axisymmetric version of the astrophysical source function for dark
***  matter pair annihilation or decays, setting the spatial part of the
***  local injection yield. this function just links to the
***  corresponding dark matter density profile to the appropriate power
***
***  NOTE: for a smooth dark matter distribution dsdmddriver must set:
***    dsdmasaxi -> dsdmdaxi**2  for dark matter pair annihilations
***    dsdmasaxi -> dsdmdaxi     for dark matter decays
***  while including substructures may result in more involved forms
***
***  input: R,z - cylindrical galactic coordinates [kpc]
***         power - selecting annihilations or decays
***
***  output:  [GeV^2 cm^-6] for pair annihilations (power=2)
***           [GeV cm^-3] for decays (power=1)
************************************************************************
      real*8 function dsdmasaxi(R,z,power)
```

dsdmassph.f

```
************************************************************************
***  spherically symmetric version of the astrophysical source function
***  for dark matter pair annihilation or decays, setting the spatial
***  part of the local injection yield. this function just links to the
***  replaceable dmd driver dsdmddriver
***
***  NOTE: for a smooth dark matter distribution dsdmddriver must set:
***    dsdmassph -> dsdmdsph**2  for dark matter pair annihilations
```

```
***    dsdmassph -> dsdmdsph     for dark matter decays
***  while including substructures may result in more involved forms
***
***  input: r - spherical galactocentric radius [kpc]
***         power - selecting annihilations or decays
***
***  output:  [GeV^2 cm^-6] for pair annihilations (power=2)
***           [GeV cm^-3] for decays (power=1)
**********************************************************************
      real*8 function dsdmassph(r,power)
```

dsdmastri.f

```
**********************************************************************
***  cartesian coordinates version of the astrophysical source function
***  for dark matter pair annihilation or decays, setting the spatial
***  part of the local injection yield. this function just links to the
***  replaceable dmd driver dsdmddriver
***
***  NOTE: for a smooth dark matter distribution dsdmddriver must set:
***    dsdmastriax -> dsdmdtriax**2  for dark matter pair annihilations
***    dsdmastriax -> dsdmdtriax     for dark matter decays
***  while including substructures may result in more involved forms
***
***  input: x,y,z - carthesian coordinates [kpc]
***         power - selecting annihilations or decays
***
***  output:  [GeV^2 cm^-6] for pair annihilations (power=2)
***           [GeV cm^-3] for decays (power=1)
**********************************************************************
      real*8 function dsdmastri(x,y,z,power)
```

dsdmdaxi.f

```
**********************************************************************
***  axisymmetric version of the dark matter density profile
***  this function just links to the replaceable dmd driver dsdmddriver
***
***  input: R,z - cylindrical galactic coordinates [kpc]
***
***  output:  [GeV cm^-3]
**********************************************************************
      real*8 function dsdmdaxi(R,z)
```

dsdmdrho0mw.f

```
      real*8 function dsdmdrho0mw(labhalo)
**********************************************************************
***  Local halo density for the halo model with access label 'labhalo',
***  within the set of models defined in the halo model repository.
***  In case you are linking to a model which has not been defined as
***  suitable for the Milky Way, the program stops.
**********************************************************************
```

dsdmdsph.f

```
**********************************************************************
***  spherically symmetric version of the dark matter density profile
***  this function just links to the replaceable dmd driver dsdmddriver
***
***  input: r - spherical galactocentric radius [kpc]
***
***  output:  [GeV cm^-3]
```

```
************************************************************************
      real*8 function dsdmdsph(r)
```

dsdmdtri.f

```
************************************************************************
***  dark matter density profile in cartesian coordinates
***  this function just links to the replaceable dmd driver dsdmddriver
***
***  input: x,y,z - carthesian coordinates [kpc]
***
***  output:  [GeV cm^-3]
************************************************************************
      real*8 function dsdmdtri(x,y,z)
```

Chapter 20

dmd_aux: Auxiliary functions for dark matter distribution routines

20.1 Routine headers – fortran files

dsdfactor.f

```
      real*8 function dsdfactor(labhalo,psi,theta)
**********************************************************************
***   the function dsdfactor gives the so-called Dfactor, namely the
***   line of sight integral of the square of a spherical dark matter
***   density profile, to be implemented in gamma-ray and neutrino
***   fluxes from particle decays in DM halos; it links to dsomlosisph
***   and saves values under subsequent calls.
***
***   inputs:
***     labhalo = halo model access label, to load the halo model via
***         dsdmdselect_halomodel
***     psi = the angular offset between the pointing direction
***         and the direction of the center of the distributionin
***         (in rad)
***     theta = aperture of the acceptance cone (defines the
***         solid angle within which line-of-sight-integrals are
***         performed assuming a step-function response from the
***         instrument; more options are available in dsomlosisph, use
***         that to compute line-of-sight-integrals in your main file)
***
***   type : commonly used
***   desc : D-factor (l.o.s. integral for decaying DM)
***
***
***   unit of return value: kpc sr * GeV cm^-3
**********************************************************************
```

dsjfactor.f

```
      real*8 function dsjfactor(labhalo,psi,theta)
**********************************************************************
***   the function dsjfactor gives the so-called Jfactor, namely the
***   line of sight integral of the square of a spherical dark matter
***   density profile, to be implemented in gamma-ray and neutrino
***   fluxes from pair annihilations in DM halos (in the limit of zero
```

```
***   relative velocity); it links to dsomlosisph and saves values
***   under subsequent calls.
***
***   inputs:
***     labhalo = halo model access label, to load the halo model via
***         dsdmdselect_halomodel; NOTE: in case it is set to 'none'
***         no model is loaded and jfactor and dfactor are NOT
***         computed here, but given as an input
***     psi = the angular offset between the pointing direction
***         and the direction of the center of the distributionin
***         (in rad)
***     theta = aperture of the acceptance cone (defines the
***         solid angle within which line-of-sight-integrals are
***         performed assuming a step-function response from the
***         instrument; more options are available in dsomlosisph, use
***         that to compute line-of-sight-integrals in your main file)
***
***   type : commonly used
***   desc : J-factor (l.o.s. integral for annihilating DM)
***
***
***   unit of return value: kpc sr * GeV^2 cm^-6
**********************************************************************
```

dslosidriver.f

```
      subroutine dslosidriver(iwin,nrein,rein,reout)
c-----------------------------------------------------------------------
c this is a sample driver to initialize, tabulate or link functions
c needed for line of sign intergration of different dark matter
c emissivities, such as synchrotron, inverse Compton or Bremstrahlung.
c
c this version is actually empty, since non of the above is included in
c the present release of the code.
c inputs:
c   - iwin: integer setting the action of the driver; ilosigasph is the
c        only one currently available
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dslosifunsph.f

```
      real*8 function dslosifunsph(radius,ilosi)
**********************************************************************
*** function to be integrated over the line of sight and, eventually,
*** over solid angle in case of spherically symmetric emissivities,
*** including (eventually) an absorption factor
*** inputs:
***   radius = radial coordinate in kpc
***   ilosi = index for link to dslosidriver
**********************************************************************
```

dslosisph.f

```
      real*8 function dslosisph(theta0in,outradius,ilosi,power,labhalo,
     &  loadlab)
**********************************************************************
*** integral over the line of sight of the product:
***   dslosifunsph * (eventually) exp(-absorption factor)
```

```
*** for a given spherically symmetric configuration, i.e.:
***
***  = int_{l.o.s.} dl dslosifunsph * (eventually) dexp(-dstaufunsph)
***
*** inputs:
***   thetain(1) (in rad) the angular offset from the line of sight in
***     the direction of the center of the distribution
***   outradius (in kpc) is the maximal radius within which the l.o.s.
***     integral has to be performed
***   ilosi = option selecting which dslosifunsph to integrate; in
***     case it exceeds ilosiabssft (set in dslosidrvrcom.h) steps
***     to include absorption are performed
***   power = 1 or 2 selecting annihilation or decay
***   labhalo = halo model access label, within the set of models defined
***     in the halo model repository
***   loadlab = logical variable: in case it is set to true the halo
***     labhalo is selected within this function, otherwise it is
***     assumed that it has been loaded before linking to this function
***     such as in the function dsgadphidEsph
***
*** output:
***   result in kpc * [dslosifunsph]
**********************************************************************
```

dsomlosisph.f

```
      real*8 function dsomlosisph(thetain,nthetain,npsfin,outradius,
     &  ilosi,power,labhalo,loadlab)
**********************************************************************
*** integral over the solid angle of the line-of-sight-integral of the
*** product:
***   dslosifunsph [which includes eventually exp(-absorption factor)]
***        * (eventually) gaussian (elliptical) instrument response
***          function
*** for a given spherically symmetric configuration, i.e.:
***
***  = int_{PSF} dOmega
***    int_{l.o.s.} dl dslosifunsph * dsomlosiweight
***
*** inputs:
***   thetain(1) (in rad) the angular offset from the line of sight in
***     the direction of the center of the distribution (except in case
***     npsfin = 6, see below)
***
***   nthetain = number of entries and dimension of the vector thetain
***
***   npsfin = 1 -> integral over solid angle assuming step-function
***          response from the instrument within a cone of aperture
***          thetain(2) (in rad)
***   npsfin = 2 -> integral over solid angle assuming gaussion response
***          from the instrument with angular resolution thetain(2)
***          (in rad)
***   npsfin = 3 -> integral over solid angle assuming gaussion
***          elliptical response from the instrument with angular
***          resolution thetain(2) (in rad) in the direction of the
***          center of the system, angular resolution thetain(3) (in
***          rad) perpendicular to it
***   npsfin = 4 -> the same as npsfin = 2 except for thetain(2) (in
***          rad) being the full width at half maximum
***   npsfin = 5 -> the same as npsfin = 3 except for thetain(2,3)
***          (in rad) being the full widths at half maximum
***
***   npsfin = 6 -> integral over solid angle, assuming step-function
***          response from the instrument within an angular window in
***          galactic coordinates:
```

```
***                  b_min = thetain(1) in rad
***                  b_max = thetain(2) in rad
***                  l_min = thetain(3) in rad
***                  l_max = thetain(4) in rad
***          for the moment you must restrict to:
***             0 < b < pi/2
***             0 < l < pi
***
***   outradius (in kpc) is the maximal radius within which the l.o.s.
***     integral has to be performed
***
***   ilosi = option selecting which dslosifunsph to integrate; in
***     case it exceeds ilosiabssft (set in dslosidrvrcom.h) steps
***     to include absorption are performed
***
***   power = 1 or 2 selecting annihilation or decay
***
***   labhalo = halo model access label, within the set of models
***     defined in the halo model repository
***   loadlab = logical variable: in case it is set to true the halo
***     labhalo is selected within this function, otherwise it is
***     assumed that it has been loaded before linking to this function
***     such as in the function dsgadphidEsph
***
*** output:
***   result in kpc sr * [dslosifunsph]
**********************************************************************
```

dstaufunsph.f

```
      real*8 function dstaufunsph(xx,ilosi)
**********************************************************************
*** function needed for absorption along the l.o.s.i. as implemented in
*** the function dslosisph and dsomlosisph
***
*** this function is linked after a call to dstausetsph; if in that call
*** locilosi has been set equal to ilosiabscomp then this function calls
*** dslosidriver and get the value of dstaufunsph, otherwise use the
*** tabulated version loaded in dstausetsph
***
*** xx is a line of sight variable, as measured from the closest point
*** to the center of the spherical system
*** ilosi = index for (eventually) linking the driver dslosidriver
***
*** dstaufunsph is dimensionless
**********************************************************************
```

dstausetsph.f

```
      subroutine dstausetsph(xxmin,xxmax,nstep,ilosi)
**********************************************************************
*** subroutine setting the tabulations of dstaufunsph. it needs to be
*** called with the correct sequence for the variable nstep, within the
*** functions dslosisph and dsomlosisph
**********************************************************************
```

dstausph.f

```
      real*8 function dstausph(theta0in,outerradius,ilosi)
**********************************************************************
*** function giving the total optical depth as needed for spherical
*** signals including absorption
***
```

```
*** a call to dslosidriver selects whether an explicit function is set
*** therein or dstausph is computed here via an integral over the line
*** of sight of an appropriate entry in the function dslosifunsph for a
*** given spherically symmetric configuration, i.e.:
***
***  = int_l.o.s. dl dslosifunsph
***
*** as it applies in case of self-absorption
***
*** inputs:
***   theta0 (in rad) the angular offset from the line of sight in the
***     direction of the center of the distribution
***   outerradius (in kpc) is the maximal radius within which the l.o.s.
***     integral has to be performed
***   ilosi = option selecting which dslosifunsph to integrate; you
***     should link with ilosi in the interval:
***       (ilosiabssft+1,ilosiabssft+ilosinsph)
***     as set in dslosidrvrcom.h
***
*** inputs from common blocks:
***   objdistance (in kpc) is the distance between the observer and the
***     center of the spherical system
***   radinnertr (in kpc) is the innner truncation radius below which
***     the integrated function is assumed to be constant
***
*** output: dimensionless (setting a proper link in dslosifunsph)
**************************************************************************
```

Chapter 21

dmd_mod: Dark matter distributions

21.1 Dark matter distributions – theory

All the dark matter detection rates depend in one way or another on the properties of the Milky Way dark matter halo. We will here outline the halo model that by default is included with DarkSUSY.

Observationally, the distribution of DM on scales relevant for DM searches is only poorly constrained. The situation is somewhat improved when instead referring to the results of large N-body simulations of gravitational clustering, which consistently find that DM halos *on average* are well described by Einasto [8] or Navarro-Frenk-White profiles [9], with the halo mass being essentially the only free parameter (after taking into account that the halo concentration strongly correlates with the halo mass [66]). On the other hand, there is a considerable halo-to-halo scatter associated to these findings, so that it remains challenging to make concrete predictions for individual objects – in particular if they are located in cosmologically somewhat 'special' environments like in the case of the Milky Way and its embedding in the Local Group. Even worse, baryonic physics can have a large impact on the DM profiles, especially on their inner parts most relevant for indirect detection, and even though hydrodynamic simulations taking into account such effect have made tremendous progress in recent years [67, 68, 69, 70], there is still a significant uncertainty related to the modelling of the underlying processes. In light of this situation, there is a considerable degree of freedom concerning halo models and the DM density profiles, and a code computing observables related to DM should be able to fully explore this freedom.

21.1.1 Rescaling of the WIMP density

While it is natural to assume that the DM particles described by a given particle module implemented in DarkSUSY make up most of the DM in our galaxy, they may also just constitute a sub-dominant part of a multi-component realization of DM. What is more, there might be both a thermal contribution to the cosmological DM abundance – as computed by the relic density routines in DarkSUSY– and a non-thermal contribution, e.g. via out-of-equilibrium production or via the decay of heavier particles. In this context, it is important to remember that DM detection rates only depend on the *local* DM density *of that particular DM candidate*. For the case of direct detection, e.g., the rate scales linearly with the local density *at earth*, while for indirect detection it scales linearly or quadratically with the local DM density *at the point of of decay or or annihilation*, respectively.

In previous versions of the code, the ratio of thermal relic abundance returned by dsrdomega and observed cosmological DM abundance was used to internally rescale the results from rate

calculations. For the reasons given above, this is not fully satisfactory and in any case obscures the origin of this rescaling. Starting from DarkSUSY 6, this is therefore no longer the case. All rate routines now assume that the local DM density equals the local density in the particular DM candidate realized in the particle module – unless they explicitly take the local DM density as an input parameter (which in that case refers to the local density in the particles described by the respective DarkSUSY module). An example for such an exception are the neutrino telescope routines, because the combined effect of DM capture and annihilation makes the dependence on the local DM density more complicated. In all other cases, e.g. if DM rate routines just take a halo label as input, the user has to make sure to rescale the rates, as described above, *by hand* to reflect possible sub-dominant DM populations.

21.2 Implementation in DarkSUSY

The implementation of dark matter halo models and related quantities in the library ds_core follows a new and highly flexible scheme, compared to earlier versions of the code, avoiding pre-defined hardcoded functions. For convenience a few pre-defined options are provided, however these can be either complemented by other profiles eventually needed, or the entire sample configuration can be simply replaced linking to a user-defined setup - in both cases without editing routines provided in this release of the code. A further improvement compared to previous versions of DarkSUSY is that different dark matter density profiles, possibly referring to different dark matter detection targets, can be defined at the same time: E.g., one can easily switch back and forth from a computation of the local positron flux induced by dark matter annihilations or decays in the Milky Way halo to the computation of the gamma-ray flux from an external halo or a Milky Way satellite within the same particle physics scenario. Finally the present implementation simplifies the task of keeping track of consistent definitions for related quantities, such as, e.g., a proper connection between the dark matter profile and the source function for a given dark matter yield (see Chapter 6), or calling within an axisymmetric coordinate system a spherically symmetric function (and preventing the opposite).

At the basis of the implementation, there is the subroutine dsdmsdriver routine which acts as an interface to quantities related to the dark matter density profiles. This routine must contain a complete set of instructions on how to retrieve the different observables: E.g. it checks the scaling of the various DM source functions (Chapter 6 with the DM density ρ_χ – namely ρ for decaying dark matter and ρ^2 for dark matter pair annihilations (in case the effect of substructures is neglected) – and passes this information to the routines for propagations of charged cosmic rays in the Galaxy (Chapter 13), assuming that such source function is axisymmetric; line of sight integration routines (needed, e.g. for the computation of gamma-ray fluxes, see Chapter 14) call this same subroutine, but may assume instead that the corresponding source function is spherically symmetric. The routine dsdmsdriver must contain the specification on whether the dark matter density profile is spherically symmetric or axisymmetric, and in the first case provide a consistent numerical output to both calls, in the latter return an error to the second call (since a spherically symmetric profile was expected). It may also be useful to use the dsdmsdriver routine for initialization calls, for instance to set parameters for a given parametric density profile, and test calls, for instance to print which dark matter profile is currently active within a set of available profiles. The input/output structure of the routine is rather general, with the first entry being however fixed to an integer flag specifying the action of the routine; currently available values and relative action are implicitly defined (through integer variables such as 'idmddensph' referring to the spherical dark matter density profile) in an include file and are global parameter. Such set of definitions should not be changed, but can be enlarged in case further profile-related quantities would be needed.

While a specific dsdmsdriver routine should match the user needs in the problem at hand, the present release provides a sample version, illustrating the flexibility of the setup. In particular the version included in the library ds_core assumes that the dark matter density profile is spherically

symmetric and does not include dark matter substructure; it allows to choose as dark matter profile one among three parametric profiles, namely the Einasto [8], the Navarro-Frenk-White [9] and the Burkert [71] profiles, or a profile interpolated from a table of values of the dark matter density at a given radius. Besides providing parameters as needed in case of parametric profiles, to complete the initialization of a profile one should also specify: *i)* an inner truncation radius, namely some r_{ic} fixing $\rho(r) = \rho(r_{ic})$ for any $r < r_{ic}$ (the choice has an impact on predictions for dark matter rates only for very singular dark matter profiles; choosing a value which is not too small allows for a faster numerical convergence of some rate computations); *ii)* an outer truncation radius, namely some r_{oc} beyond which the profile is assumed to be zero; *iii)* the distance from the observer of the center of the profile, corresponding to the Sun galactocentric distance only in case the profile refers to the Milky Way; *iv)* whether it is a profile that refers to the Milky Way and hence for which rates that are Milky Way specific, such as the local contribution to antimatter fluxes, can be computed; *v)* whether it is a profile to be saved in a halo profile database for later use.

Regarding this last point, the code implements a procedure of associating the set of entries fully specifying a halo profile (namely the choice of the parametric profile, the corresponding parameters and and the entries *i)–v)* above) to a given *input label*, and this can be reloaded at any time when needed; in particular all indirect detection flux routines have the label among their input parameters, so that in case of several dark matter detection targets or several profiles for the same targets it becomes unambiguous which profile is being considered. On the other hand it may be the case that the user needs to loop over many different profiles without keeping track of all of them for later reuse (e.g., in a scan over parameter space in the estimate of line of sight integrals towards a dwarf satellite); in such case the profile can be defined as "temporary", with only the latest set temporary profile available at any given time. For halos that are stored in the halo profile database, one can save and/or read from disc tabulated quantities, such as, e.g., the Green function needed for the computation of the local positron flux, for temporary profiles tables can (or in some cases need) to be computed running the code but are overwritten any time the temporary profile is changed. While in the previous releases of DarkSUSY, halo parameters were typically set via common blocks to be included in the main file, the default dsdmsdriver.f implements a procedure in which, when initializing a given halo profile, parameters are given as an input in association with a corresponding parameter label, and profile settings are specified as character strings appearing within the profile label.

Along with the default dsdmsdriver.f routine, which is unfortunately rather involved since it allows for several different options, in the present release we provide example main files which illustrate a few of the possible user needs. Those are described in the 'quick start' part of this manual, see Section 2.2.1, and cover examples of how to use pre-defined halo profiles, read in tabulated profiles as well as how to create completely new ones.

21.2.1 Dark matter distributions – routines

The main routines in this directory are

dsdmsdriver Main driver function for setting the halo profile and retrieving observables. All calls to the halo models go through this routine.

dsdmssethm Wrapper routine for dsdmsdriver to set a halo model.

dsdmsselecthm Wrapper routine for dsdmsdriver to select an already defined halo model.

21.3 Routine headers – fortran files

dsdmd_init.f

```
subroutine dsdmd_init
```

```
c-----------------------------------------------------------------------
c subroutine adding to the database of currently available halo models
c a set of default profiles which have been defined as 'defaults' in
c dsdmddriver
c-----------------------------------------------------------------------
```

dsdmddriver.f

```
      subroutine dsdmddriver(iwin,nrein,rein,nchin,chin,labin,reout)
c-----------------------------------------------------------------------
c this is a sample driver to initialize, print or link functions
c corresponding to the halo model.
c
c inputs:
c   - iwin: integer setting the action of the driver; it can be equal
c     to:
c       idmdinit -> intialization of a set of default halo model: you
c         can access via the label:
c          - 'mwnfwdef': the default Milky Way NFW profile
c          - 'mwburdef': the default Milky Way Burkert profile
c          - 'mweindef': the default Milky Way Einasto profile
c       idmdcreate -> a profile is created (started here and completed
c         in the sub-drivers called in this subroutine); unless another
c         model is loaded via idmdload below, this is assumed as current
c         halo model
c       idmdload -> the profile corresponding to 'labin' is searched
c         for in the halo model database and loaded as current model
c       idmdparprint -> the current halo profile is printed
c         depending on the input labin, see below
c       idmddensph -> the driver links to the spherical density profile
c       idmdsousph -> the driver links to the spherical dark matter
c         source profile, scaling with rho (dm decay) or rho**2 (dm pair
c         annihilation) [GeV cm^-3] or [GeV^2 cm^-6]; no substructure
c         component in this sample implementation.
c       idmdmassph -> the driver links to the spherical mass profile
c     since in this implemetation all halo models are spherically
c       symmetric calling with iwin idmddenaxi (axisymmetric coordinates)
c       or idmddentri (cartesian coordinates) is just turned to the case
c       iwin=idmddensph after proper coordinate transformation; the same
c       applies to idmdsouaxi and idmdsoutri
c   - labin: external label of the current halo model; the character
c       string MUST contain substrings for the driver initialization:
c       - one of the following is needed to specify which profile to
c         link:
c          a) 'nfw' a spherical NFW profile is assumed
c          b) 'bur' a spherical Burkert profile is assumed
c          c) 'ein' a spherical Einasto profile is assumed
c          d) 'num' a spherical profile is given as a table interpolation
c       - if it contains 'hdb' the halo model is stored as one entry in
c         halo database; if it contains 'def' it is a default DS model,
c         also stored in the database; if it does not contain any of the
c         it is assumed to be a temporary model, one for which you cannot
c         store tabulations. Only one temporary model is available at any
c         given time. A 'num' models needs to be temporary.
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c   - chin(nchin) character*10 vector for input parameter labels
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dsdmddriver_bur.f

```
      subroutine dsdmddriver_bur(iwin,nrein,rein,nchin,chin,labin,reout)
c-----------------------------------------------------------------------
c this is the sample driver to initialize, print or link corresponding
c functions in case of a dark matter density profile described by the
c spherical Burkert profile, i.e.:
c
c     rho(r) = rhosbur/(1+(r/rsbur))/(1+(r/rsbur)^2)
c
c inputs:
c   - iwin: integer setting the action of the driver; it can be equal
c     to:
c       idmdcreate -> profile creation is completed here (first settings
c         are in the dsdmddriver subroutine)
c       idmdparprint -> the current halo profile is printed
c         depending on the input labin, see below
c       idmddensph -> the driver links to the spherical density profile
c   - labin: external label of the current halo model: you got here in
c       case of labin containing the string 'bur'; if it contains also
c       the string 'mw' the model is intepreted as referring to the
c       Milky Way, you can define it assigning the local halo density
c       rather than the reference density 'rhosbur' and an internal
c       label for tabulation files is automatically created
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c   - chin(nchin) character*10 vector for input parameter labels
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dsdmddriver_choice.f

```
      subroutine dsdmddriver_choice(iwin,nrein,rein,nchin,chin,labin,
     &  reout)
c-----------------------------------------------------------------------
c subroutine looping over available halo model drivers; this is the
c default version; if you need to add other profiles you need to replace
c this subroutine
c-----------------------------------------------------------------------
```

dsdmddriver_ein.f

```
      subroutine dsdmddriver_ein(iwin,nrein,rein,nchin,chin,labin,reout)
c-----------------------------------------------------------------------
c this is the sample driver to initialize, print or link corresponding
c functions in case of a dark matter density profile described by the
c spherical Einasto profile, i.e.:
c
c     rho(r) = rhosein*exp(-2/alphaein*((r/rsein)^alphaein-1))
c
c inputs:
c   - iwin: integer setting the action of the driver; it can be equal
c     to:
c       idmdcreate -> profile creation is completed here (first settings
c         are in the dsdmddriver subroutine)
c       idmdparprint -> the current halo profile is printed
c         depending on the input labin, see below
c       idmddensph -> the driver links to the spherical density profile
c       idmdmassph -> the driver links to the spherical mass profile
c   - labin: external label of the current halo model: you got here in
c       case of labin containing the string 'ein'; if it contains also
```

```
c       the string 'mw' the model is intepreted as referring to the
c       Milky Way, you can define it assigning the local halo density
c       rather than the reference density 'rhosein' and an internal
c       label for tabulation files is automatically created
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c   - chin(nchin) character*10 vector for input parameter labels
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dsdmddriver_nfw.f

```
      subroutine dsdmddriver_nfw(iwin,nrein,rein,nchin,chin,labin,reout)
c-----------------------------------------------------------------------
c this is the sample driver to initialize, print or link corresponding
c functions in case of a dark matter density profile described by the
c spherical Navarro-Frenk-White profile, i.e.:
c
c     rho(r) = rhosnfw/((r/rsnfw)*(1+r/rsnfw)^2)
c
c inputs:
c   - iwin: integer setting the action of the driver; it can be equal
c     to:
c       idmdcreate -> profile creation is completed here (first settings
c         are in the dsdmddriver subroutine)
c       idmdparprint -> the current halo profile is printed
c         depending on the input labin, see below
c       idmddensph -> the driver links to the spherical density profile
c       idmdmassph -> the driver links to the spherical mass profile
c   - labin: external label of the current halo model: you got here in
c       case of labin containing the string 'nfw'; if it contains also
c       the string 'mw' the model is intepreted as referring to the
c       Milky Way, you can define it assigning the local halo density
c       rather than the reference density 'rhosnfw' and an internal
c       label for tabulation files is automatically created
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c   - chin(nchin) character*10 vector for input parameter labels
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dsdmddriver_num.f

```
      subroutine dsdmddriver_num(iwin,nrein,rein,nchin,chin,labin,reout)
c-----------------------------------------------------------------------
c this is the sample driver to initialize, print or link corresponding
c functions in case of a dark matter density profile defined as the
c interpolation of a table radius versus density profile
c
c inputs:
c   - iwin: integer setting the action of the driver; it can be equal
c     to:
c       idmdcreate -> profile creation is completed here (first settings
c         are in the dsdmddriver subroutine)
c       idmdparprint -> the current halo profile is printed
c         depending on the input labin, see below
c       idmddensph -> the driver links to the spherical density profile
c   - labin: external label of the current halo model: you got here in
c       case of labin containing the string 'num'; if it contains also
```

```
c       the string 'mw' the model is intepreted as referring to the
c       Milky Way
c   - rein(nrein) real*8 vector for different inputs according to
c      different iwin values
c   - chin(nchin) character*10 vector for input parameter labels
c output:
c   - reout: real*8 output for the different function linking as
c     specified by the input value iwin
c-----------------------------------------------------------------------
```

dsdmdgetlabel.f

```
      subroutine dsdmdgetlabel(labelinoutfun,labelout)
************************************************************************
*** subroutine to search in file 'labelfile' for the label corresponding
*** to the currently implemented set of npar parameters par
*** if the set of npar parameters par does not exist in labelfile, a new
*** label is added
*** automatically created out of the suffix labelsuffix and a sequential
*** number, and label/parameters are stored in labelfile; if addlabel is
*** set to false, an error message is printed and the program stops
***
*** input:
***     labelinoutfun - external function setting internal common blocks
*** inputs from the common block dslabelcom:
***     labelfile - in the file in which label and parameters are stored
***     npar - the number of parameters to be stored (npar.le.100)
***     par - a real*8 vector with the parameters to be stored
***     addlabel - logical variable, if set to true new label/parameters
***       can be stored in labelfile
***     labelsuffix - character*4 suffix to create automatically new
***       labels to store new sets of parameters
*** output:
***     labelout - the label found or autmatically generated
************************************************************************
```

dsdmdprint_halomodel.f

```
      subroutine dsdmdprint_halomodel(labin)
c-----------------------------------------------------------------------
c subroutine printing name and parameters for the halo with label equal
c to 'labin' within the database of currently set halo models
c if labin='printall' all models in the database are printed
c if labin refers to a temporary halo model the last temporary setting
c is returned.
c-----------------------------------------------------------------------
```

dsdmdprof_bur.f

```
************************************************************************
***  dark matter density profiles written in the form:
***
***    rho(r) = rhos * dsdmprof(r/rs)
***
***  namely dsdmdprof is the function g(x), e.g., in Eq. (10) of
***  astro-ph/0207125.
***
***  NOTE: version valid for a Burkert profile only
***
***  input:
***    x [1]
************************************************************************
      real*8 function dsdmdprof_bur(x)
```

dsdmdprof_ein.f

```
**************************************************************************
***  dark matter density profiles written in the form:
***
***    rho(r) = rhos * dsdmdprof(r/rs,alpha)
***
***  namely dsdmdprof is the function g(x), e.g., in Eq. (10) of
***  astro-ph/0207125.
***
***  NOTE: version valid for a Einasto profile only
***
***  input:
***    x [1]
***    alpha [1]: Einasto index
**************************************************************************
      real*8 function dsdmdprof_ein(x,alpha)
```

dsdmdprof_nfw.f

```
**************************************************************************
***  dark matter density profiles written in the form:
***
***    rho(r) = rhos * dsdmdprof(r/rs)
***
***  namely dsdmdprof is the function g(x), e.g., in Eq. (10) of
***  astro-ph/0207125.
***
***  NOTE: version valid for a NFW profile only
***
***  input:
***    x [1]
**************************************************************************
      real*8 function dsdmdprof_nfw(x)
```

dsdmdselect_halomodel.f

```
      subroutine dsdmdselect_halomodel(labin)
c-------------------------------------------------------------------------
c subroutine selecting from the database of currently set halo models
c the one corresponding to the label 'labin'.
c if labin refers to a temporary halo model the last temporary setting
c is returned.
c this subroutine also make sure that the following quantities, needed
c for detection rates are set by dsdmddriver consistently with 'labin':
c   - dmdmw: logical variable saying whether the model should be used
c     for Milky Way rates or not
c   - dmdlabel: internal dmd label for tabulation files
c   - dmdobjdist: distance from the center of dark matter object (the
c     Sun galactocentric distance in case of the Milky Way)
c   - dmdradintr: inner density truncation radius
c   - dmdradouttr: outer radius at which the density is set to zero
c   - dmdrho0: if dmdmw=.true. this is the local halo density
***
***   type : commonly used
***   desc : select between halo models in the halo repository
***
c-------------------------------------------------------------------------
```

dsdmdset_halomodel.f

```
      subroutine dsdmdset_halomodel(labin,nin,chin,rein)
```

```
c-----------------------------------------------------------------------
c subroutine adding to the repository of currently available halo models
c the one corresponding to the label 'labin'.
c at this stage you need to know the rules that have been defined in
c dsdmddriver on how to link 'labin' to the corresponding object and
c profile as well as what is the list of input parameters and relative
c tags dsdmddriver expects. if such rules are not respected an error
c message is printed and the program stops.
***
***   type : commonly used
***   desc : add an existing halo model to halo repository
***
c-----------------------------------------------------------------------
```

dsdmdsetind.f

```
      subroutine dsdmdsetind(indexout,rein1)
c-----------------------------------------------------------------------
c subroutine assigning the index 'indexout' to store in the vector
c dmdparvec the real*8 input value rein1.
c-----------------------------------------------------------------------
```

dsdmdsetlabel.f

```
      real*8 function dsdmdsetlabel(inout)
c-----------------------------------------------------------------------
c auxiliary function to read/write/store halo profile within the file
c dmdlabel.dat
c NOTE: since dmdlabel.dat may contain automatically generated labels
c different users may have the same halo profile stored under a
c different label, hence also linking to different tabulation files;
c to avoid this, make sure that the file dmdlabel.dat is the same and
c replace names for the corresponding linked files
c-----------------------------------------------------------------------
```

dsdvmatch.f

```
      subroutine dsdvmatch(itypein,nrein,rein,itypeout,nreout,reout)
c-----------------------------------------------------------------------
c subroutine to match the input rein(nrein) dependent variables of type
c itypein onto the output reout(nreout) dependent variables of type
c itypeout.
c the possible matches are:
c   itypeout = itypein -> you just copy rein(nrein) into reout(nreout)
c   itypeout spherical from itypein axisymmetric or triaxial
c   itypeout axisymmetric from itypein triaxial
c in all other cases a error message is returned and the program stops
c-----------------------------------------------------------------------
```

dslabcheck.f

```
      subroutine dslabcheck(nlabin,labin,nlabsuff,labsuff,match)
c-----------------------------------------------------------------------
c subroutine to search fot the character string 'labsuff' made of
c nlabsuff characters within the label 'labin' made of nlabin characters
c if the string is found match=.true. is returned, otherwise
c match=.false. is returned
c-----------------------------------------------------------------------
```

Chapter 22

dmd_vel: Dark matter velocity distributions

22.1 Dark matter phase-space distributions – theory

The DM velocity and density profiles cannot be chosen independently, in principle, but have to satisfy consistency relations. For a spherically symmetric and isotropic system, e.g., the two profiles are related by the Eddington equation [72, 73]. A fully self-consistent implementation of phase-space distributions will be available with a later DarkSUSY version, at which point the directory dmd_vel will become obsolete.

Until then, the user can freely choose a DM velocity distribution among those provided in src/dmd_vel – but should keep in mind this consistency requirement when comparing direct detection rates (which require the local DM velocity profile) to, e.g., the gamma-ray flux from the galactic center (which requires choosing a density profile). Concretely, it is the function dshmuDF that returns the 3D distribution function $f(\mathbf{v})$ needed by the direct detection routines. It allows to switch between various pre-implemented functional forms, including tabulated velocity profiles, but can of course also be replaced by an arbitrary function supplied by the user (c.f. Section 3.6).

One of these options is the often adopted truncated gaussian, which in the detector frame moving at speed v_O relative to the galactic halo reads

$$f(v) = \frac{1}{\mathcal{N}_{\rm cut}} \frac{v^2}{u v_O \sigma} \left\{ \exp\left[-\frac{(u-v_O)^2}{2\sigma^2} \right] - \exp\left[-\frac{\min(u+v_O, v_{\rm cut})^2}{2\sigma^2} \right] \right\} \tag{22.1}$$

for $v_{\rm esc} < v < \sqrt{v_{\rm esc}^2 + (v_O + v_{\rm cut})^2}$ and zero otherwise, with $u = \sqrt{v^2 + v_{\rm esc}^2}$ and

$$\mathcal{N}_{\rm cut} = \frac{v_{\rm cut}}{\sigma} \exp\left(-\frac{v_{\rm cut}^2}{2\sigma^2} \right) - \sqrt{\frac{\pi}{2}} {\rm erf}\left(\frac{v_{\rm cut}}{\sqrt{2}\sigma} \right). \tag{22.2}$$

As default, we have taken the halo line-of-sight (one-dimensional) velocity dispersion σ =120 km/s,* the galactic escape speed $v_{\rm cut}$ = 600 km/s, the relative Earth-halo speed v_O = 264 km/s (a yearly average) and the Earth escape speed $v_{\rm esc}$ = 11.9 km/s. These parameters can be changed by the user.

22.2 Routine headers – fortran files

dshm_set.f

```
      subroutine dshm_set(c)
```

*Other authors write $\exp(-3v^2/2\overline{v}^2)$, in which case $\overline{v} = \sqrt{3}\sigma$.

```
*****************************************************************
*** subroutine dshm_set:                                         ***
*** initialize the density profile and or the small clump        ***
*** probability distribution                                     ***
*** type of halo:                                                ***
***   hclumpy=1 smooth, hclumpy=2 clumpy                        ***
***                                                             ***
*** a few sample cases are given; specified values of the        ***
*** local halo density 'rho0' and of the length scale            ***
*** parameter 'a' should be considered just indicative           ***
***                                                             ***
*** author: piero ullio (piero@tapir.caltech.edu)                ***
*** date: 00-07-13                                               ***
*** small modif: paolo gondolo 00-07-19                          ***
*** mod: 03-11-19 je, 04-01-13 pu, 09-05-07 ps, 09-08-08 ps      ***
*****************************************************************
```

dshmdfisotr.f

```
*****************************************************************
***                                                             ***
*** halo local velocity distribution function DF(\vec{v})       ***
*** for the case of an isotropic distribution, i.e. for         ***
***       DF(\vec{v}) = DF(|\vec{v}|) = DF(v)                   ***
***                                                             ***
*** dshmDFisotr is normalized such that                         ***
*** int d^3v DF(v) = 4 pi int_0^\infty dv v^2 DF(v) = 1         ***
***                                                             ***
*** Input:  |\vec{v}| = Speed in km/s                           ***
*** Output: DF(|\vec{v}|)  in (km/s)^(-3)                       ***
***                                                             ***
*** Calls other routines depending of choice of velocity        ***
*** distribution function (as set by isodf in the common        ***
*** blocks in dshmcom.h)                                        ***
***                                                             ***
*** Author: Piero Ullio                                         ***
*** Date: 2004-01-30                                            ***
*****************************************************************

      real*8 function dshmDFisotr(v)
```

dshmdfisotrnum.f

```
*****************************************************************
***                                                             ***
*** halo local velocity distribution function DF(\vec{v})       ***
*** for the case of an isotropic distribution, i.e. for         ***
***       DF(\vec{v}) = DF(|\vec{v}|) = DF(v)                   ***
*** as loaded from table in file provided by user               ***
***                                                             ***
*** on first call the function loads from file a table of       ***
*** values and then interpolates between them.                  ***
***                                                             ***
*** the file name is set by the isodfnumfile variable           ***
*** it is assumed that this file has no header and v DF(v)      ***
*** are given with the format 1000 below                        ***
***                                                             ***
*** to reload a (different) file the int. flag dfisonumset      ***
*** into the DFisosetcom common block has to be manually        ***
*** reset to 0                                                  ***
***                                                             ***
*** v in km s**-1                                               ***
*** DF(v) in km**-3 s**3                                        ***
***                                                             ***
```

```
*** Author: Piero Ullio                                          ***
*** Date: 2004-01-30                                             ***
*******************************************************************

      real*8 function dshmDFisotrnum(v)
```

dshmudf.f

```
*******************************************************************
*** Dark matter halo velocity profile.                          ***
*** This routine gives back u*DF(u) in units of (km/s)^(-2)     ***
***                                                             ***
***    u is the modulus of \vec{u} = \vec{v} - \vec{v}_{MY}     ***
***    with \vec{v} the 3-d velocity of a WIMP in the           ***
***    galactic frame, and \vec{v}_{MY} the projection on       ***
***    the frame you are considering                            ***
***                                                             ***
***    DF(u) = int dOmega DF(\vec{v}), where                    ***
***    DF(\vec{v}) is the halo local velocity distribution      ***
***    function in the galactic frame                           ***
***                                                             ***
*** Note: u*DF(u) is the same as f(u)/u, where f(u) is the      ***
*** one-dimensional distribution function as defined in e.g.    ***
*** Gould, ApJ 321 (1987) 571.                                  ***
***                                                             ***
*** Note: it is also the same as the one-dimensional            ***
*** distribution function g(u) as defined in, e.g.,             ***
*** Ullio & Kamionkowski, JHEP ....                             ***
***                                                             ***
*** dshmuDF is normalized such that                             ***
*** int_0^\infty u*dshmuDF du = int_0^\infty u^2 DF(u) du =     ***
*** int_0^\infty f(u) du = 1                                    ***
***                                                             ***
*** Input:  u = Speed in km/s                                   ***
*** Output: u*DF(u) in (km/s)^(-2)                              ***
***                                                             ***
*** Calls other routines depending of choice of velocity        ***
*** distribution function (as set by veldf in the common        ***
*** blocks in dshmcom.h)                                        ***
*** Author: Joakim Edsjo                                        ***
*** Date: 2004-01-29                                            ***
*******************************************************************

      real*8 function dshmuDF(u)
```

dshmudfearth.f

```
*******************************************************************
*** Dark matter halo velocity profile as seen from the          ***
*** Earth. Compared to dshmuDF, this routine also includes      ***
*** the possibility to use distribution functions where         ***
*** solar system diffusion is included.                         ***
***                                                             ***
*** This routine gives back u*DF(u) in units of (km/s)^(-2)     ***
*** DF(u) = int dOmega DF(abs(v)) where DF(abs(v-vector)) is    ***
*** the three-dimensional distribution function in the halo     ***
*** and v = v_us + u with u being the velocity relative us      ***
*** (Earth/Sun).                                                ***
*** Note: u*DF(u) is the same as f(u)/u, where f(u) is the      ***
*** one-dimensional distribution function as defined in e.g.    ***
*** Gould, ApJ 321 (1987) 571.                                  ***
***                                                             ***
*** dshmuDF is normalized such that                             ***
*** int_0^\infty u*dshmuDF du = int_0^\infty u^2 DF(u) du =     ***
```

```
*** int_0^\infty f(u) du = 1                                    ***
***                                                             ***
*** Input:  u = Speed in km/s                                   ***
*** Output: u*DF(u) in (km/s)^(-2)                              ***
***                                                             ***
*** Calls other routines depending of choice of velocity        ***
*** distribution function (as set by veldfearth in the   ***
*** common blocks in dshmcom.h)                                 ***
*** Author: Joakim Edsjo                                        ***
*** Date: 2004-01-29                                            ***
*****************************************************************

      real*8 function dshmuDFearth(u)
```

dshmudfearthtab.f

```
*****************************************************************************
*** dshmudfearthtab returns the halo velocity distribution
*** (same as dshmudfearth.f), but reads it from a file.
*** The file should have two header lines (with arbitrary content)
*** and then lines with two columns each  with u and u*DF(u).
*** u should be in units of km/s and u*DF(u) (or f(u)/u) in units
*** of (km/s)^(-2).
***
*** The file loaded is given by the option type.
*** Some possible types are velocity distributions as obtained from
*** numerical simulations of WIMP propagation in the solar system
*** including solar capture.
***
*** For the simulations made by Johan Lundberg, see astro-ph/0401113,
*** available options are
***   type = 1, reads file <ds-root>/dat/vdfearth-sdbest.dat :
***      best estimate of distribution at Earth from numerical sims
***   type = 2, reads file <ds-root>/dat/vdfearth-sdconserv.dat :
***      conservative estimate, only including free orbits and
***      jupiter-crossing orbits
***   type = 3, reads file <ds-root>/dat/vdfearth-sdultraconserv.dat :
***      ultraconservative estimate, only including free orbits
***   type = 4, reads file <ds-root>/dat/vdfearth-sdgauss.dat :
***      as if Earth was in free space, i.e. gaussian approx.
***   Note: tot.txt is the best estimate of the distribution at Earth
***   and should be used as a default
***
*** There are also other options, like
***   type = 5, read a user-supplied file with file name given
***     by udfearthfile in dshmcom.h. If you change the file or
***     for any other reason want to reload it here, you have to
***     set the flag udfearthload to true, in which case it will
***     be loaded here on next call.
***
*** Input: velocity relative to earth [ km s^-1 ]
*** Output: f(u) / u [ (km/s)^(-2) ]
*** Date: January 30, 2004
*****************************************************************************

      real*8 function dshmuDFearthtab(u,type)
```

dshmudfgauss.f

```
*****************************************************************************
*** The halo velocity profile in the Maxwell-Boltzmann (Gaussian)
*** approximation.
*** input: velocity relative to earth [ km s^-1 ]
*** output: f(u) / u [ (km/ s)^(-2) ]
```

```
*** date: april 6, 1999
*** Modified: 2004-01-29
**********************************************************************

      real*8 function dshmuDFgauss(u)
```

dshmudfgaussdd.f

```
**********************************************************************
*** The halo velocity profile in the Maxwell-Boltzmann (Gaussian)
*** approximation with the addition of a dark disk.
*** input: velocity relative to earth [ km s^-1 ]
*** output: f(u) / u [ (km/ s)^(-2) ]
*** date: april 6, 1999
*** Modified: 2004-01-29, 2009-04-30
*** Modified for best case dark disc by Lars Rosenstrom
**********************************************************************

      real*8 function dshmuDFgaussdd(u)
```

dshmudfiso.f

```
*****************************************************************
***                                                           ***
*** function which gives u*DF(u) where:                       ***
***                                                           ***
***    u is the modulus of \vec{u} = \vec{v} - \vec{v}_{ob}   ***
***    with \vec{v} the 3-d velocity of a WIMP in the         ***
***    galactic frame, and \vec{v}_{ob} the projection on     ***
***    the frame you are considering                          ***
***                                                           ***
***    DF(u) = int dOmega DF(\vec{v}), where                  ***
***    DF(\vec{v}) is the halo local velocity distribution    ***
***    function in the galactic frame                         ***
***                                                           ***
*** the function implemented here is valid for:              ***
***    a) an isothermal sphere profile                        ***
***    b) an isotropic profile, i.e.                          ***
***       DF(\vec{v}) = DF(|\vec{v}|)                         ***
*** condition b) implies that the integral is performed by    ***
*** setting |\vec{v}|^2 = u^2 + |\vec{v}_ob|^2                ***
***                     + 2*cos(alpha)*|\vec{v}_ob|*u         ***
*** and then integrating in d(cos(alpha))                     ***
***                                                           ***
*** u in km s**-1                                            ***
*** u*DF(u) in km**-2 s**2                                   ***
***                                                           ***
*** Author: Piero Ullio                                      ***
*** Date: 2004-01-29                                         ***
*****************************************************************

      real*8 function dshmuDFiso(u)
```

dshmudfnum.f

```
*****************************************************************
***                                                           ***
*** function which gives u*DF(u) where:                       ***
***                                                           ***
***    u is the modulus of \vec{u} = \vec{v} - \vec{v}_{ob}   ***
***    with \vec{v} the 3-d velocity of a WIMP in the         ***
***    galactic frame, and \vec{v}_{ob} the projection on     ***
***    the frame you are considering                          ***
***                                                           ***
```

```
***    DF(u) = int dOmega DF(\vec{v}), where                 ***
***    DF(\vec{v}) is the halo local velocity distribution   ***
***    function in the galactic frame                        ***
***                                                          ***
*** on first call the function loads from file a table of    ***
*** values and then interpolates between them.               ***
***                                                          ***
*** the file name is set by the udfnumfile variable          ***
*** it is assumed that this file has no header and u uDF(u)  ***
*** are given with the format 1000 below                     ***
***                                                          ***
*** to reload a (different) file the integer flag uDFnumset  ***
*** into the uDFnumsetcom common block has to be manually    ***
*** reset to 0                                               ***
***                                                          ***
*** u in km s**-1                                            ***
*** u*DF(u) in km**-2 s**2                                   ***
***                                                          ***
*** Author: Piero Ullio                                      ***
*** Date: 2004-01-30                                         ***
****************************************************************

      real*8 function dshmuDFnum(u)
```

dshmudfnumc.f

```
****************************************************************
***                                                          ***
*** function which gives u*DF(u) where:                      ***
***                                                          ***
***    u is the modulus of \vec{u} = \vec{v} - \vec{v}_{ob}  ***
***    with \vec{v} the 3-d velocity of a WIMP in the        ***
***    galactic frame, and \vec{v}_{ob} the projection on    ***
***    the frame you are considering                         ***
***                                                          ***
***    DF(u) = int dOmega DF(\vec{v}), where                 ***
***    DF(\vec{v}) is the halo local velocity distribution   ***
***    function in the galactic frame                        ***
***                                                          ***
*** on first call the function tabulates uDF(u) and saves    ***
*** the tabulated values in the file whose name is set by    ***
*** the udfnumfile variable in dshmcom.h                     ***
*** interpolation between tabulated values are then used     ***
*** the tabulation has at least 200 points, and more points  ***
*** are added if there are jumps in u*DF which are more than ***
*** 10%; this can be adjusted by changing the reratio        ***
*** variable which is hard coded in the file                 ***
***                                                          ***
*** the implementation is valid only for an isotropic        ***
*** profile, i.e. for                                        ***
***       DF(\vec{v}) = DF(|\vec{v}|)                        ***
*** with the integral performed by setting                   ***
***       |\vec{v}|^2 = u^2 + |\vec{v}_ob|^2                 ***
***                         + 2*cos(alpha)*|\vec{v}_ob|*u    ***
*** and then integrating in d(cos(alpha))                    ***
***                                                          ***
*** u in km s**-1                                            ***
*** u*DF(u) in km**-2 s**2                                   ***
***                                                          ***
*** Author: Piero Ullio                                      ***
*** Date: 2004-01-30                                         ***
****************************************************************


      real*8 function dshmuDFnumc(u)
```

dshmudftab.f

```
************************************************************************
*** dshmudftab returns the halo velocity distribution
*** (same as dshmudf.f), but reads it from a file.
*** The file should have two header lines (with arbitrary content)
*** and then lines with two columns each  with u and u*DF(u).
*** u should be in units of km/s and u*DF(u) (or f(u)/u) in units
*** of (km/s)^(-2).
***
*** The file loaded is given by the option type.
*** Some possible types are velocity distributions as obtained from
*** numerical simulations of WIMP propagation in the solar system
*** including solar capture.
***
*** Available options
***   type = 1, read a user-supplied file with file name given
***     by udffile in dshmcom.h. If you change the file or
***     for any other reason want to reload it here, you have to
***     set the flag udfload to true, in which case it will
***     be loaded here on next call.
***
*** Input: velocity relative to earth [ km s^-1 ]
*** Output: f(u) / u [ (km/s)^(-2) ]
*** Date: January 30, 2004
************************************************************************

      real*8 function dshmudftab(u,type)
```

dshmvelearth.f

```
      subroutine dshmvelearth(tdays)
No header found.
```

Chapter 23

fi: Freeze-In

23.1 Freeze-in – theory

The freeze-in mechanism to produce DM in the early Universe describes scenarios where the interaction with the primordial heat bath is so weak that equilibrium was never obtained [74, 75, 76]. This is in contrast to the situation of typical WIMPs that acquire their relic density through the freeze-*out* mechanism, as described in Section 27.1. The implementation of freeze-in routines in DarkSUSY follows a formulation [16] of the freeze-in process that maximizes the analogy with production by freeze-out.

Concretely, the abundance of feably interacting particles (FIMPs) increases as

$$\frac{dY_\chi}{dx} = \frac{\left(n_\chi^{\rm MB}\right)^2}{xs\tilde{H}}\langle\sigma v\rangle \tag{23.1}$$

as long as it stays far below the equilibrium abundance (which we *define* here as the freeze-in regime). Here, the DM abundance is $Y_\chi \equiv n_\chi/s$, with n_χ the DM density and s the total entropy (assumed to be conserved). Further, $x \equiv m_\chi/T$ and $\tilde{H} \equiv H/\left[1 + (1/3)d(\log g_{\rm eff}^s)/d(\log T)\right]$, where $g_{\rm eff}^s$ denote the effective entropy degrees of freedom.

Following slightly different conventions, this is essentially Eq. (27.54) for the freeze-out case, with three important differences:

- By assumption, there is no 'backreaction' term that would describe the annihilation of FIMPs into heat bath particles.
- The factor $n_\chi^{\rm MB}$ denotes the number density corresponding to a *would-be* Maxwell-Boltzmann distribution of DM particles. Unlike in the case of WIMPs – cf. the assumption $f_\chi(E,T) = A(T)f_\chi^{\rm eq}(E/T)$ entering in the derivation of Eq. (27.54) – the above formulation does not assume that the *actual* DM distribution is related to a thermal one.
- In analogy with the freeze-out case, Eq. (23.1) is formulated in terms the DM *annihilation* cross section σ. Different from WIMPs, however, DM is relativistic during a significant part of the production period, such that in particular effects due to quantum statistics can not be neglected, viz. Bose enhancement/Fermi suppression due to large occupancies of the SM particles in the final states.

The thermal average appearing in Eq. (23.1) is most conveniently expressed in terms of a model-

independent thermal kernel and a model-dependent invariant rate $W_{\rm eff}$:

$$\langle\sigma v\rangle = \int_1^\infty d\tilde{s}\,\frac{x\sqrt{\tilde{s}-1}\,K_1\Big(2\sqrt{\tilde{s}}x\Big)}{2m_\chi^2 {K_2}^2(x)} W_{\rm eff}(s,T)\,. \tag{23.2}$$

The main conceptual difference to the corresponding expression for WIMPs is that the invariant rate is itself T-dependent, as a direct consequence of the third point in the bullet list above. Concretely, the invariant rate is given by

$$W_{\rm eff}(s,T) \equiv 16 m_\chi^2 \frac{x\tilde{s}\sqrt{\tilde{s}-1}}{K_1\Big(2\sqrt{\tilde{s}}x\Big)} \int_1^\infty d\gamma\,\sqrt{\gamma^2-1} e^{-2\sqrt{\tilde{s}}x\gamma} \sum_{\psi_1\psi_2} \sigma_{\chi\chi\to\psi_1\psi_2}(s,\gamma)\,, \tag{23.3}$$

where the integration is over Lorentz boosts γ from the CMS to the cosmic rest frame, and the in-medium cross section can be written as

$$\sigma_{\chi\chi\to\psi_1\psi_2}(s,\gamma) = \frac{N_\psi^{-1}}{8\pi s}\frac{|\mathbf{k}_{\rm CM}|}{\sqrt{s-4m_\chi^2}} \int_{-1}^1 \frac{d\cos\theta}{2}\,\big|\overline{\mathcal{M}}\big|^2_{\chi\chi\to\psi_1\psi_2}(s,\cos\theta)\,G_{\psi_1\psi_2}(\gamma,s,\cos\theta)\,, \tag{23.4}$$

where $N_\psi = 2$ for identical SM particles ($\psi_1 = \psi_2$) and $N_\psi = 1$ otherwise. The effect of quantum statistics in the final state, leading to an enhancement or decrease of the corresponding cross section in vacuum, is encoded in the quantity

$$\begin{aligned} G_{\psi_1\psi_2}(\gamma,s,\cos\theta) \;&=\; 1+\varepsilon_\psi^2 e^{-2\sqrt{\tilde{s}}x\gamma} \\ &\quad -\varepsilon_\psi\left\{e^{-\frac{1}{T}\left(E_{\psi_1}\gamma+|\mathbf{k}_{\rm CM}|\cos\theta\sqrt{\gamma^2-1}\right)} + e^{-\frac{1}{T}\left(E_{\psi_2}\gamma-|\mathbf{k}_{\rm CM}|\cos\theta\sqrt{\gamma^2-1}\right)}\right\}, \end{aligned} \tag{23.5}$$

with $E_{\psi_i} = \sqrt{\mathbf{k}_{\rm CM}^2 + m_{\psi_i}^2}$ and $\epsilon_\psi = +1$ ($\epsilon_\psi = -1$) for fermions (bosons).

It is worth noting that for $G_{\psi_1\psi_2} = 1$ – which formally corresponds to setting '$\epsilon_\psi = 0$' in the last expression – one recovers the standard expression for the cross section in the CMS frame which, in particular, is independent of γ. The integral in Eq. (23.4) can then be solved analytically [16] resulting as expected in the T-independent expression (27.18) for $W_{\rm eff}$ in the non-relativistic case.

The other relevant observation is that Eq. (23.2) allows to include other T-dependent effects in the model-dependent part as well. Notable examples include thermal masses as well as phase transitions, which generally affect both interaction rates and the spectrum of relevant SM states. While this can in principle also affect relic density calculations for WIMPs – if freeze-out happens, e.g., very close to a phase transition – it is generically much more important for freeze-in because of the large range of temperatures during which freeze-in production is efficient.

23.2 Freeze-in – routines

The routines provided in `src/fi` are accessible after an initial call to dsfi_init (as done directly from dsinit), which sets some global performance flags related to the handling of SM phase-transitions and finite-temperature effects (see documentation in the `.f` file for further details). Typically, the function of greatest interest will be dsfi2to2oh2, and there should be no need to call any of the other routines directly.

dsfi2to2oh2 numerically solves the Boltzmann equation (23.1) and returns the resulting DM relic density in terms of Ωh^2 as a function of the reheating temperature (defined as the starting point of the integration). As explained in more detail in the function header, and in analogy to the corresponding routine dsrdomega for freeze-out calculations, dsfi2to2oh2 further takes two performance flags as input – 'option' steering particle-module specific settings, and 'fast' allowing

to adjust global settings (such as whether to take into account the effect of quantum statistics) to be different from the default ones.

Technically, the integration of Eq. (23.1) – with the r.h.s. provided by dsfi2to2rhs – is performed in dsfi2to2ab, taking special care of not missing the QCD and EW phase transitions. Another function worth mentioning is dsfithav which returns the thermally averaged cross section as given in Eq. (23.2). Internally, this function rescales the invariant rate to a normalization comparable to WIMPs, before using the same thermal average routine dsrdthav as in that case (which requires a call to the subroutine dsrd_set in order to notify the routines in /rd that the invariant rate depends on p and T rather than only on p).

There are two interface functions that a particle physics module must provide for the freeze-in routines in src/ to work: *i)* subroutine dsrdparticles provides kinematic information about (potentially T-dependent) thresholds and resonances. The same interface function is also required by the freeze-out routines in src/rd. *ii)* a function dsanwx_finiteT returning the temperature-dependent rate entering in Eq. (23.2).

23.3 Routine headers – fortran files

dsfi2to2ab.f

```
*************************************************************************
*** Auxiliary function dsfi2to2ab integrates the Boltzmann equation for     ***
*** freeze-in from 2->2 processes, x dY/dx=RHS (with RHS returned by        ***
*** dsfi2to2rhs). Called by dsfi2to2oh2. Note the linear interpolations     ***
*** very close to EW and QCD phase transitions; see settings in dsfi_init   ***
*** for further details.                                                    ***
***                                                                        ***
*** Input:                                                                 ***
***    TR - reheating temperature [GeV]                                    ***
***                                                                        ***
*** author: torsten.bringmann@fys.uio.no, 2021-09-22                       ***
***         (based on previous version by Kristian Gjestad Vangsnes)       ***
*************************************************************************
      real*8 function dsfi2to2ab(TR)
```

dsfi2to2oh2.f

```
*************************************************************************
*** Function dsfi2to2oh2 returns the relic density, in terms of omega h^2,  ***
*** from thermal freeze-in due to 2->2 processes.                           ***
***                                                                        ***
***  type : commonly used                                                  ***
***  desc : Calculate the relic density $\Omega h^2$ from freeze-in        ***
***         production (thorugh 2->2 processes)                            ***
***                                                                        ***
*** Input:                                                                 ***
***    TR        - reheating temperature [GeV]                             ***
***    option    - integer parameter passed to dsrdparticles provided by   ***
***                particle module; used for model-specific settings       ***
***                related to kinematic information. [default: option=0]   ***
***    fast =  0 - standard accurate calculation [default]                 ***
***         =  1 - ignore any temperature dependence in effective invariant ***
***                rate, including the impact of quantum statistics        ***
***                (much faster than 0, but up to 20% error in result)     ***
***         =  2 - as 0, but bypasses detailed resolution of EW and QCD PTs ***
***                when integrating the Boltzmann equation                 ***
***                (not recomended, unless you know that the invariant rate ***
***                 of the initialized particle model has no strong        ***
***                 T-dependence in this regime)                           ***
***                                                                        ***
*** NB: Some particle module may provide further options corresponding to   ***
```

```
***     the 'fast' flag above. In this case, you can access non-default     ***
***     options by calling the respective subroutine dsfiset_[module name]  ***
***     prior to calling dsfi2to2oh2. (See e.g. the silveira_zee module)    ***
***                                                                         ***
*** author: torsten.bringmann@fys.uio.no, 2021-11-14                        ***
***         (based on previous version by Kristian Gjestad Vangsnes)        ***
*******************************************************************************
      real*8 function dsfi2to2oh2(option,fast,TR)
```

dsfi2to2rhs.f

```
*******************************************************************************
***  Function dsfi2to2rhs gives the right hand side of the Boltzmann        ***
***  equation for freeze-in: x dY/dx=RHS.                                   ***
***                                                                         ***
***  input:                                                                 ***
***     ln x - ln(m/T)                                                      ***
***                                                                         ***
***  author: Kristian Gjestad Vangsnes (kristgva@uio.no)   2020-09-07       ***
***          Torsten Bringmann                                              ***
*******************************************************************************
      real*8 function dsfi2to2rhs(lnx)
```

dsfi_init.f

```
*******************************************************************************
*** dsfi_init initializes freeze-in routines and sets defaults              ***
***                                                                         ***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no                 ***
*** Date: 2021-09-10                                                        ***
*******************************************************************************
      subroutine dsfi_init
```

dsfianwx.f

```
*******************************************************************************
*** Function dsfianwx_finiteT provides a *re-scaled* version of the         ***
*** invariant rate for DM self-annihilation, in order to ensure that        ***
*** thermal averages are numerically as stable as for WIMP-scale rates.     ***
***                                                                         ***
*** author: Kristian Vangsnes, 2020                                         ***
*** mod 2021-11-15 (tb): added fi_how flag                                  ***
*******************************************************************************
      real*8 function dsfianwx(p,x)
```

dsfiprep.f

```
*******************************************************************************
*** Auxiliary routine needed to set up RD routines, and in particular       ***
*** thermal averages, for feably interacting DM particles. Typically called ***
*** by dsfi2to2oh.                                                          ***
***                                                                         ***
***  Input:                                                                 ***
***     option,fast - see dsfi2to2oh2                                       ***
***                                                                         ***
***  Output                                                                 ***
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)      ***
***                                                                         ***
*** author: torsten.bringmann@fys.uio.no, 2021-10-20                        ***
*******************************************************************************
      subroutine dsfiprep(option,fast,selfconj)
```

dsfithav.f

```
************************************************************************
*** Function dsfithav provides the thermal average <sigma v> of the  ***
*** effective annihilation cross section. Similar to dsrdthav, but   ***
*** adapted to the small rates associated to freeze-in production. Also ***
*** including finite temperature effects in the cross section, as well as ***
*** quantum statistics effects from the final state (and/or virtual heat ***
*** bath) particles.                                                  ***
***                                                                   ***
***  Input:                                                           ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma    ***
***          temperature [both in GeV] at which sigma v is evaluated  ***
***                                                                   ***
***  Output:                                                          ***
***     <sigma v> in units of GeV^-2                                  ***
***                                                                   ***
***  Note: Unlike dsrdthav, this function does not explicitly take the ***
***  invariant annihilation rate as (external) input. Instead, the    ***
***  interface function dsanwx_finite (in src_models/) is always used for ***
***  this puporse (via the auxiliary function dsfianwx).              ***
***                                                                   ***
***  NB: This function assumes that dsfiprep has been called previously ***
***      (typically through a call to dsfi2to2oh2)                    ***
***                                                                   ***
***  Author: Torsten Bringmann                                        ***
***  Date: 2021-09-21                                                 ***
************************************************************************
      real*8 function dsfithav(x)
```

Chapter 24

include: src/include

24.1 Routine headers – fortran files

dscontribstat.F

```
#define HEALPIX_INSTALLATION succeeded

#define HIGGSBOUNDS_INSTALLATION succeeded

#define HIGGSSIGNALS_INSTALLATION succeeded

#define HEALPIX_INSTALLATION succeeded

#define HIGGSBOUNDS_INSTALLATION succeeded

#define HIGGSSIGNALS_INSTALLATION succeeded

#define HEALPIX_INSTALLATION succeeded

#define HIGGSBOUNDS_INSTALLATION succeeded

#define HIGGSSIGNALS_INSTALLATION succeeded
```

Chapter 25

ini: Initialization routines

25.1 Initialisation routines

This directory contains general initialisation routines that need to be called to prepare calculations with DarkSUSY, but which are independent of the particle physics. Most importantly, it contains the subroutine dsinit, which should be called at the beginning of any program using DarkSUSY. In particular, this routine calls various more specific initialisation routines that are relevant only for specific applications (such as the relic density calculation) and therefore reside in the respective directory in src/. It also calls dsinit_module, which initializes the specific particle module that the user has chosen to link to when compiling the main program. Another subroutine called by dsinit is ini/dsreadnuclides. It reads in names and properties of nuclei and stores them in common blocks, to which both direct detection and solar/earth capture routines need access.

Lastly, the directory /ini also contains two files dsdirname.c.in and dsvername.c.in. These are needed to provide a system-wide reference to the installation directory and version of DarkSUSY, respectively. These refer to include files that are set at configure time, dsdir.h and dsver.h. The user should never have to modify these manually as they are determined at the configure stage.

25.2 Routine headers – fortran files

dsinit.f

```
************************************************************************
*** Subroutine dsinit intializes DarkSUSY. Every DarkSUSY main program      ***
*** should call this routine first thing to get DarkSUSY set up and ready   ***
*** to run                                                                  ***
***                                                                        ***
***   type : commonly used                                                 ***
***   desc : Initialize DarkSUSY (should always be called)                 ***
***                                                                        ***
*** Author: Joakim Edsjo, edsjo@fysik.su.se                                ***
*** Date: June 11, 2013                                                    ***
*** (mod Torsten Bringmann, 2015, 2021)                                    ***
************************************************************************
      subroutine dsinit
```

dsinit_module.f

```
************************************************************************
```

```
***  This is a dummy version of dsinit_module.                              ***
***  This file exists so that you can link to DarkSUSY without linking to   ***
***  a particle physics module in case you actually do not need anything    ***
***  from the particle physics module. In the default linking where         ***
***  the particle physics modules are linked before the core, this dummy    ***
***  routine will be replaced with the proper one from the particle physics ***
***  module.                                                                ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  desc : Intialzation of module                                          ***
***                                                                         ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)               ***
***  Date: 04/2014                                                          ***
*******************************************************************************
      subroutine dsinit_module
```

dsreadnuclides.f

```
      subroutine dsreadnuclides
No header found.
```

Chapter 26

kd: Kinetic decoupling

26.1 Kinetic decoupling and microhalos (kd) – theory

Even after *chemical decoupling*, which sets the DM relic density (see Section 27.1.1), DM frequently scatters with the very abundant standard model particles and thereby stays in local thermal equilibrium with the heat bath until the temperature has dropped by another factor of between 10 and a few 1000 (for typical WIMPs like, e.g. neutralino DM in the MSSM); after *kinetic decoupling*, even these scattering events cease and DM no longer interacts with standard model particles. Inhomogeneities in the DM density can only develop after this has happenened, and the DM particles have sufficiently cooled down so that free streaming becomes negligible. The scale of kinetic decoupling can therefore directly be translated into a cutoff in the power spectrum of (dark) matter fluctuations and thus the size of the smallest (at least when not taking into account primordial black holes) gravitationally bound objects in the universe (for a review, see Ref. [17]).

26.1.1 Kinetic decoupling

The process of kinetic decoupling is governed by the full Boltzmann equation for the WIMP distribution function $f(\mathbf{x}, \mathbf{p})$, which in a flat Friedmann-Robertson Walker spacetime reads

$$E(\partial_t - H\,\mathbf{p}\cdot\nabla_{\mathbf{p}})\,f = C[f]\,, \tag{26.1}$$

where $C[f]$ is known as the collision term. The Boltzmann equation quoted in Eq. (27.1) for the description of the (chemical) DM freeze-out process is actually just the first moment of this expression, i.e. obtained by integrating it over $\int d^3p$ (after dividing it by E). As was realized in [77, 17], kinetic decoupling can be described to a very high precision by considering, instead, the *second* moment of Eq. (26.1). For this purpose, one introduces the WIMP "temperature" T_χ,

$$\int \frac{d^3p}{(2\pi)^3}\frac{\mathbf{p}^2}{E} f(\mathbf{p}) \equiv 3\,T_\chi n_\chi\,, \tag{26.2}$$

as a parameter that characterizes the deviation from thermal equilibrium (for which $T_\chi = T$ holds). After kinetic decoupling, the DM 'temperature' will simply decrease due to the expansion of the universe and, for non-relativistic DM, scale like $T_\chi \propto a^{-2}$. It is thus natural to define the moment of decoupling as the transition between these two regimes [17]. Allowing for the scattering partners to have a different temperature ($T_{\tilde\gamma}$) than that of the photons (T), this implies

$$T_\chi(T) = \begin{cases} T_{\tilde\gamma}(T) & \text{for } T \gtrsim T_{\rm kd} \\ T_{\tilde\gamma}(T_{\rm kd})\left(\frac{a(T_{\rm kd})}{a(T)}\right)^2 & \text{for } T \lesssim T_{\rm kd} \end{cases} \tag{26.3}$$

For practical purposes, one may now further introduce

$$x \equiv m_\chi/T\,, \tag{26.4}$$

$$y \equiv m_\chi T_\chi s^{-2/3}\,. \tag{26.5}$$

Multiplying Eq.(26.1) by $\mathbf{p}^2/E$, integrating it over $\mathbf{p}$ and keeping only the leading order terms* in $\mathbf{p}^2/m_\chi^2$ then results in [17, 79]

$$\frac{d\log y}{d\log x} = \left(1 - \frac{1}{3}\frac{d\log g_{*S}}{d\log x}\right)\frac{\gamma(T_{\tilde{\gamma}})}{H(T)}\left(\frac{y_{\rm eq}}{y} - 1\right)\,. \tag{26.6}$$

Here, g_{*S} is the number of effective entropy degrees of freedom, $y_{\rm eq}$ is the value of y in thermal equilibrium and γ denotes the momentum transfer rate,

$$\gamma(T_{\tilde{\gamma}}) = \frac{1}{48\pi^3 g_\chi T_{\tilde{\gamma}} m_\chi^3}\sum_i \int d\omega\, k^4\left(1 \mp g_i^\pm\right) g_i^\pm(\omega)\, |\mathcal{M}|^2_{\substack{t=0\\ s=m_\chi^2+2m_\chi\omega+m_{\tilde{\gamma}}^2}}\,, \tag{26.7}$$

where the sum runs over all DM scattering partners (counting, e.g., electrons and positrons separately), $k \equiv |\mathbf{k}|$ is their momentum and ω their energy. The g_i denote the SM distribution functions, which are assumed to be thermal (note, however, that no assumption has been made about the form of the WIMP distribution function f). The scattering amplitude squared in this expression, $|\mathcal{M}|^2$, is understood to be *summed* over all internal (spin or color) degrees of freedom, including initial ones. If it is not Taylor expandable around $t = 0$, one has to make the replacement [80, 81]

$$|\mathcal{M}|^2_{\substack{t=0\\ s=m_\chi^2+2m_\chi\omega+m_{\tilde{\gamma}}^2}} \longrightarrow \left\langle|\mathcal{M}|^2\right\rangle_t \equiv \frac{1}{8k^4}\int_{-4k_{CM}^2}^0 dt(-t)\,|\mathcal{M}|^2 \tag{26.8}$$

in the above expression. We may easily check that the asymptotic behaviour for T_χ described by Eq. (26.6) is consistent with the expectation outlined above: At large T, we have $\gamma \gg H$, thus enforcing $T_\chi = T$; when T becomes small, the WIMPs completely decouple from the thermal bath and y stays constant, i.e. $T_\chi \propto s^{2/3} \propto a^{-2}$. A practical way to determine the *kinetic decoupling* temperature $T_{\rm kd}$ as defined in Eq. (26.3) is by solving the above differential equation until y stays constant (indicated by the '$x \to \infty$'):

$$x_{\rm kd} = \frac{m_\chi}{T_{\rm kd}} \equiv g_{\rm eff}(T_{\rm kd})\;\; y|_{x\to\infty}\;. \tag{26.9}$$

26.1.2 The smallest protohalos

Before kinetic decoupling, WIMPs are tightly coupled to the heat bath, so any small-scale perturbations in the DM fluid would immediately be washed out. For temperatures $T \lesssim T_{\rm kd}$, however, this is no longer the case and perturbations in the DM density start to devolop under the influence of gravity; however, the remaining viscous coupling between the two fluids and, subsequently, the free-streaming of the WIMPs generate an exponential cutoff in the power spectrum [82], with a characteristic comoving damping scale $k_{\rm fs}$. The WIMP mass contained in a sphere of the corresponding size is thus given by [17]

$$M_{\rm fs} \approx \frac{4\pi}{3}\rho_\chi\left(\frac{\pi}{k_{\rm fs}}\right)^3 = 4.0\times10^{-6}\left(\frac{1 + \ln\left(g_{\rm eff}^{1/4}T_{\rm kd}/30\text{ MeV}\right)/18.6}{(m_\chi/100\text{ GeV})^{1/2}\, g_{\rm eff}^{1/4}\,(T_{\rm kd}/30\text{ MeV})^{1/2}}\right)^3 M_\odot\,. \tag{26.10}$$

*For very early decoupling, next-order correction terms may become relevant [78].

Acoustic oscillations also have to be taken into account as a damping mechanism and lead to a similar exponential cutoff in the power spectrum [83, 84]. In this case, the characteristic damping mass is given by the total amount of DM inside the horizon at the time of kinetic decoupling:

$$M_{\rm ao} \approx \frac{4\pi}{3}\,\frac{\rho_\chi}{H^3}\Big|_{T=T_{\rm kd}} = 3.4\times10^{-6}\left(\frac{T_{\rm kd}\,g_{\rm eff}^{1/4}}{50\,{\rm MeV}}\right)^{-3} M_\odot\,. \tag{26.11}$$

Note that $T_{\rm kd}$ in the above expressions only holds when using the definition given in Eq. (26.3); for an alternative definition, the expected magnitude of the cutoff mass has to be correspondingly re-scaled.

In general, the actual cutoff in the power spectrum is given by $M_{\rm cut} = \max\left[M_{\rm fs}, M_{\rm ao}\right]$; which of the two physically independent damping mechanisms is more efficient depends on the particle nature of the WIMP. The expected mass for the smallest gravitationally bound objects in the universe is then also simply given by $M_{\rm cut}$. Numerically, the formation of protohalos with masses down to the cutoff scale has been confirmed and their evolution could be followed until a redshift of $z \sim 26$ [85]; the further survival probabilities of these objects, however, as well as the resulting consequences for the indirect detection of DM, are subject to a presently still ongoing discussion.

26.2 Kinetic decoupling – routines

Before using any of the routines provided in `src/kd` for the first time, one has to call dskd_set in order to make some necessary initializations; in particular, a call to this routine ensures that the relevant tables for the relativistic degrees of freedom in the early universe are correctly read in. Typically, the subroutines of greatest interest will be dskdtkd and dskdmcut, and there should be no need to call any of the other routines directly.

dskdtkd numerically solves the Boltzmann equation (26.6) and determines $T_{\rm kd}$ as given in Eq. (26.9). Here, special care is taken to accurately handle potential resonances in the scattering amplitude; to this end, dskdboltz_init identifies the location of all relevant resonances and passes this information to dskdgammarate where the integral of Eq. (26.7) is performed.

Finally, dskdmcut returns the mass cutoff in the power spectrum, with an input parameter determining whether it is $M_{\rm fs}$ or $M_{\rm ao}$; the default call results in $M_{\rm cut} = \max\left[M_{\rm fs}, M_{\rm ao}\right]$, i.e. the mass of the smallest protohalos.

There are three interface functions that a particle physics module must provide for the kinetic decoupling routines in src/ to work: dskdm2 returns the full scattering matrix element squared, evaluated at $t = 0$ or averaged over t, while dskdm2simp returns only the leading contribution for small ω (expressed as a simple power-law in ω, in which case there exists an analytic solution for $T_{\rm kd}$ [77]). Lastly, the particle physics module must provide a routine dskdparticles which, in analogy to the routine dsrdparticles for the case of chemical decoupling discussed above, sets the location of resonances in $|\mathcal{M}|^2$.

26.3 Routine headers – fortran files

dskd_set.f

```
************************************************************************
*** The routine dskd_set has to be called once before any microhalo (mh)
*** (mh) calculations; it makes all necessary initializations and sets
*** some default settings.
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2013-06-12 (removed model-dependence)
***          2017-04-28 (added two options for quark scattering)
***          2017-05-11 (changed d.o.f. treatment - unified with RD)
```

```
**************************************************************************

      subroutine dskd_set(key)
```

dskdboltz.f

```
**************************************************************************
*** dskdboltz implements the Boltzmann equation for the evolution of
*** the WIMP temperature as dy/dx = rhs, expressed in the variables
*** (see arXiv:1603.04884)
***
***    y = m0 * T_DM * s**(-2/3)
***    x = m0 / T
***
*** NB: src_models/xxx/mh/dskdparticles MUST be called before using this!
***    (typically done via a call to dskdtkd)
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2013-06-11 (removed model-dependence)
***          2017-05-11 (changed definition of y)
***          2018-05-28 (allow for Tdark != Tphoton)
**************************************************************************

      subroutine dskdboltz(x,y,rhs)
```

dskdcint.f

```
**************************************************************************
*** dskdcfull returns the integrand for the integration over the SM
*** (or other scattering partner) momenta that appears in the collision
*** term. Called by dskdgammarate. itegrand is multiplied by 1d20 to work
*** better with the integration routine
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2014-05-09 (removed model-dependence)
***          2017-04-28 (added two options for quark scattering)
***          2018-05-14 (added option for non-SM scattering)
**************************************************************************

      real*8 function dskdcint(kint)
```

dskdcint2.f

```
**************************************************************************
*** auxiliary function for the quick determination of Tkd
*** (integrand for the thermal average). Called by dskdTkd.
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2014-05-09 (removed model-dependence)
**************************************************************************

      real*8 function dskdcint2(y)
```

dskdgammarate.f

```
**************************************************************************
*** dskdgammarate returns the momentum transfer rate [in GeV], as defined in
*** (A6) of arXiv:1603.04884. Input is the *photon* temperature,
***
***    x = mDM / Tphoton
***
*** NB: src_models/xxx/mh/dskdparticles MUST be called before using this!
***    (both typically done via a call to dskdtkd)
```

```
***
*** author: torsten.bringmann@fys.uio.no, 2016-12-21
*** updates: 2018-05-28 (allow for Tdark != Tphoton)
**********************************************************************

      real*8 function dskdgammarate(x)
```

dskdgeff.f

```
**********************************************************************
*** returns effective degrees of freedom during kinetic decoupling
***
***   input:   T    -- Temperature in GeV
***   output:  geff -- rel. energy d.o.f [\rho = geff*(pi^2/30)*T^4 ]
***
*** author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2017-05-12
***         (modified version of Paolo Gondolos dsrddof)
**********************************************************************

      subroutine dskdgeff(T,geff)
```

dskdinty.f

```
**********************************************************************
*** dskdinty integrates the ODE y'(x) = f(x,y) from xi to xf,
*** using an adaptive stepsize control for Runge-Kutta
*** (adapted from code provided by NUMERICAL RECIPES).
*** further input: eps is the accuracy of the result,
*** f is provided via an external subroutine derivs(x,y,f)
***
*** author: torsten bringmann (troms@physto.se), 2008-12-03
**********************************************************************

      subroutine dskdinty(ystart,xi,xf,eps,hi,hmin,derivs,ier)
```

dskdmcut.f

```
**********************************************************************
*** The function dskdmcut returns the typical mass scale of the smallest
*** gravitationally bound objects [in units of M_sun]
***
***  type : commonly used
***  desc : Cutoff mass in linear power spectrum
***
***  input: m0  - DM mass [in GeV]
***         tkd - kinetic decoupling temperature [in MeV]
***         how = 1,2,3
***               1 - maximum of 2,3 [default]
***               2 - cutoff scale associated to acoustic oscillations
***                   (horizon mass at T_kd / fit to ETHOS simulations)
***               3 - cutoff scale associated to free-streaming
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2013-06-12 (removed model-dependence)
***          2020-05-27 (added ETHOS result for late KD)
**********************************************************************

      real*8 function dskdmcut(m0,tkd,how)
```

dskdtkd.f

```
**********************************************************************
```

```
*** The function dskdtkd returns the kinetic coupling temperature [in MeV]
***
***  input: m0  - DM mass (in GeV)
***         how - integer flag
***
***  possible values for 'how':
***  1 - full calculation
***      [recommended: following Bringmann, New J. Phys. 11, 105027 (2009);
***       updated in Ref.A of 1603.04884]
***  2 - fast calculation
***      [neglecting SM masses and resonances in the scattering amplitude]
***  3 - "old" order-of-magnitude estimate
***      [use only for comparison!]
***
***   type : commonly used
***   desc : Kinetic decoupling temperature
***
***
*** author: torsten bringmann (troms@physto.se), 2010-01-10
*** updates: 2013-06-12 (removed model-dependence)
***          2015-06-06 (added non-standard heat bath temperature)
**********************************************************************

      real*8 function dskdtkd(how)
```

Chapter 27

rd:
Relic density

27.1 Relic density – theoretical background

27.1.1 The Boltzmann equation and thermal averaging

Griest and Seckel [86] have worked out the Boltzmann equation when coannihilations are included. We start by reviewing their expressions and then continue by rewriting them into a more convenient form that resembles the familiar case without coannihilations. This allows us to use similar expressions for calculating thermal averages and solving the Boltzmann equation whether coannihilations are included or not. The implementation in DarkSUSY is based upon the work done in [13].

27.1.2 Review of the Boltzmann equation with coannihilations

Consider annihilation of N DM particles χ_i ($i = 1, \ldots, N$) with masses m_i and internal degrees of freedom (statistical weights) g_i. Also assume that $m_1 \leq m_2 \leq \cdots \leq m_{N-1} \leq m_N$ and that R-parity is conserved. Note that for the mass of the lightest stable of the particles we will use the notation m_χ and m_1 interchangeably.

The evolution of the number density n_i of particle i is

$$\begin{aligned}\frac{dn_i}{dt} &= -3Hn_i - \sum_{j=1}^{N}\langle\sigma_{ij}v_{ij}\rangle\left(n_i n_j - n_i^{\rm eq}n_j^{\rm eq}\right)\\ &\quad - \sum_{j\neq i}\left[\langle\sigma'_{Xij}v_{ij}\rangle\left(n_i n_X - n_i^{\rm eq}n_X^{\rm eq}\right) - \langle\sigma'_{Xji}v_{ij}\rangle\left(n_j n_X - n_j^{\rm eq}n_X^{\rm eq}\right)\right]\\ &\quad - \sum_{j\neq i}\left[\Gamma_{ij}\left(n_i - n_i^{\rm eq}\right) - \Gamma_{ji}\left(n_j - n_j^{\rm eq}\right)\right]. \end{aligned} \tag{27.1}$$

The first term on the right-hand side is the dilution due to the expansion of the Universe. H is the Hubble parameter. The second term describes $\chi_i\chi_j$ annihilations, whose total annihilation cross section is

$$\sigma_{ij} = \sum_X \sigma(\chi_i\chi_j \to X). \tag{27.2}$$

The third term describes $\chi_i \to \chi_j$ conversions by scattering off the cosmic thermal background,

$$\sigma'_{Xij} = \sum_Y \sigma(\chi_i X \to \chi_j Y) \tag{27.3}$$

being the inclusive scattering cross section. The last term accounts for χ_i decays, with inclusive decay rates

$$\Gamma_{ij} \quad = \quad \sum_X \Gamma(\chi_i \to \chi_j X). \tag{27.4}$$

In the previous expressions, X and Y are (sets of) standard model particles involved in the interactions, v_{ij} is the 'relative velocity' defined by

$$v_{ij} = \frac{\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2}}{E_i E_j} \tag{27.5}$$

with p_i and E_i being the four-momentum and energy of particle i, and finally $n_i^{\rm eq}$ is the equilibrium number density of particle χ_i,

$$n_i^{\rm eq} = \frac{g_i}{(2\pi)^3} \int d^3\mathbf{p}_i f_i \tag{27.6}$$

where $\mathbf{p}_i$ is the three-momentum of particle i, and f_i is its equilibrium distribution function. In the Maxwell-Boltzmann approximation it is given by

$$f_i = e^{-E_i/T}. \tag{27.7}$$

The thermal average $\langle \sigma_{ij} v_{ij} \rangle$ is defined with equilibrium distributions and is given by

$$\langle \sigma_{ij} v_{ij} \rangle = \frac{\int d^3\mathbf{p}_i d^3\mathbf{p}_j f_i f_j \sigma_{ij} v_{ij}}{\int d^3\mathbf{p}_i d^3\mathbf{p}_j f_i f_j} \tag{27.8}$$

Normally, the decay rate of particles χ_i other than the lightest which is stable is much faster than the age of the universe. Since we have assumed R-parity conservation, all of these particles decay into the lightest one. So its final abundance is simply described by the sum

$$n = \sum_{i=1}^{N} n_i. \tag{27.9}$$

For n we get the following evolution equation

$$\frac{dn}{dt} = -3Hn - \sum_{i,j=1}^{N} \langle \sigma_{ij} v_{ij} \rangle \left(n_i n_j - n_i^{\rm eq} n_j^{\rm eq} \right) \tag{27.10}$$

where the terms on the second and third lines in Eq. (27.1) cancel in the sum.

The scattering rate of DM particles off particles in the thermal background is much faster than their annihilation rate, because the scattering cross sections σ'_{Xij} are of the same order of magnitude as the annihilation cross sections σ_{ij} but the background particle density n_X is much larger than each of the DM particle densities n_i when the former are relativistic and the latter are non-relativistic, and so suppressed by a Boltzmann factor. In this case, the χ_i distributions remain in thermal equilibrium, and in particular their ratios are equal to the equilibrium values,

$$\frac{n_i}{n} \simeq \frac{n_i^{\rm eq}}{n^{\rm eq}}. \tag{27.11}$$

We then get

$$\frac{dn}{dt} = -3Hn - \langle \sigma_{\rm eff} v \rangle \left(n^2 - n_{\rm eq}^2 \right) \tag{27.12}$$

where

$$\langle \sigma_{\rm eff} v \rangle = \sum_{ij} \langle \sigma_{ij} v_{ij} \rangle \frac{n_i^{\rm eq}}{n^{\rm eq}} \frac{n_j^{\rm eq}}{n^{\rm eq}}. \tag{27.13}$$

27.1.3 Thermal averaging

So far the reviewing. Now let's continue by reformulating the thermal averages into more convenient expressions.

We rewrite Eq. (27.13) as

$$\langle \sigma_{\rm eff} v \rangle = \frac{\sum_{ij} \langle \sigma_{ij} v_{ij} \rangle n_i^{\rm eq} n_j^{\rm eq}}{n_{\rm eq}^2} = \frac{A}{n_{\rm eq}^2} \,. \tag{27.14}$$

For the denominator we obtain, using Boltzmann statistics for f_i,

$$n^{\rm eq} = \sum_i n_i^{\rm eq} = \sum_i \frac{g_i}{(2\pi)^3} \int d^3 p_i e^{-E_i/T} = \frac{T}{2\pi^2} \sum_i g_i m_i^2 K_2 \left(\frac{m_i}{T} \right) \tag{27.15}$$

where K_2 is the modified Bessel function of the second kind of order 2.

The numerator is the total annihilation rate per unit volume at temperature T,

$$A = \sum_{ij} \langle \sigma_{ij} v_{ij} \rangle n_i^{\rm eq} n_j^{\rm eq} = \sum_{ij} \frac{g_i g_j}{(2\pi)^6} \int d^3 \mathbf{p}_i d^3 \mathbf{p}_j f_i f_j \sigma_{ij} v_{ij} \tag{27.16}$$

It is convenient to cast it in a covariant form,

$$A = \sum_{ij} \int W_{ij} \frac{g_i f_i d^3 \mathbf{p}_i}{(2\pi)^3 2E_i} \frac{g_j f_j d^3 \mathbf{p}_j}{(2\pi)^3 2E_j}. \tag{27.17}$$

W_{ij} is the (unpolarized) annihilation rate per unit volume corresponding to the covariant normalization of $2E$ colliding particles per unit volume. W_{ij} is a dimensionless Lorentz invariant, related to the (unpolarized) cross section through*

$$W_{ij} = 4 p_{ij} \sqrt{s} \sigma_{ij} = 4 \sigma_{ij} \sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4 E_i E_j \sigma_{ij} v_{ij}. \tag{27.18}$$

Here

$$p_{ij} = \frac{\left[s - (m_i + m_j)^2 \right]^{1/2} \left[s - (m_i - m_j)^2 \right]^{1/2}}{2\sqrt{s}} \tag{27.19}$$

is the momentum of particle χ_i (or χ_j) in the center-of-mass frame of the pair $\chi_i \chi_j$.

Averaging over initial and summing over final internal states, the contribution to W_{ij} of a general n-body final state is

$$W_{ij}^{n-\rm body} = \frac{1}{g_i g_j S_f} \sum_{\text{internal d.o.f.}} \int |\mathcal{M}|^2 (2\pi)^4 \delta^4 (p_i + p_j - \textstyle\sum_f p_f) \displaystyle\prod_f \frac{d^3 \mathbf{p}_f}{(2\pi)^3 2E_f}, \tag{27.20}$$

where S_f is a symmetry factor accounting for identical final state particles (if there are K sets of N_k identical particles, $k = 1, \ldots, K$, then $S_f = \prod_{k=1}^K N_k!$). In particular, the contribution of a two-body final state can be written as

$$W_{ij \to kl}^{2-\rm body} = \frac{p_{kl}}{16\pi^2 g_i g_j S_{kl} \sqrt{s}} \sum_{\text{internal d.o.f.}} \int |\mathcal{M}(ij \to kl)|^2 \, d\Omega, \tag{27.21}$$

where p_{kl} is the final center-of-mass momentum, S_{kl} is a symmetry factor equal to 2 for identical final particles and to 1 otherwise, and the integration is over the outgoing directions of one of the final particles. As usual, an average over initial internal degrees of freedom is performed.

*The quantity w_{ij} in Ref. [87] is $W_{ij}/4$.

We now reduce the integral in the covariant expression for A, Eq. (27.17), from 6 dimensions to 1. Using Boltzmann statistics for f_i (a good approximation for $T \lesssim m$)

$$A = \sum_{ij} \int g_i g_j W_{ij} e^{-E_i/T} e^{-E_j/T} \frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i} \frac{d^3\mathbf{p}_j}{(2\pi)^3 2E_j}, \tag{27.22}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the three-momenta and E_i and E_j are the energies of the colliding particles. Following the procedure in Ref. [12] we then rewrite the momentum volume element as

$$d^3\mathbf{p}_i d^3\mathbf{p}_j = 4\pi |\mathbf{p}_i| E_i dE_i \, 4\pi |\mathbf{p}_j| E_j dE_j \, \frac{1}{2} d\cos\theta \tag{27.23}$$

where θ is the angle between $\mathbf{p}_i$ and $\mathbf{p}_j$. Then we change integration variables from E_i, E_j, θ to E_+, E_- and s, given by

$$\begin{cases} E_+ &= E_i + E_j \\ E_- &= E_i - E_j \\ s &= m_i^2 + m_j^2 + 2E_iE_j - 2|\mathbf{p}_i||\mathbf{p}_j|\cos\theta, \end{cases} \tag{27.24}$$

whence the volume element becomes

$$\frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i} \frac{d^3\mathbf{p}_j}{(2\pi)^3 2E_j} = \frac{1}{(2\pi)^4} \frac{dE_+ dE_- ds}{8}, \tag{27.25}$$

and the integration region $\{E_i \geq m_i, E_j \geq m_j, |\cos\theta| \leq 1\}$ transforms into

$$s \geq (m_i + m_j)^2, \tag{27.26}$$

$$E_+ \geq \sqrt{s}, \tag{27.27}$$

$$\left| E_- - E_+ \frac{m_j^2 - m_i^2}{s} \right| \leq 2p_{ij} \sqrt{\frac{E_+^2 - s}{s}}. \tag{27.28}$$

Notice now that the product of the equilibrium distribution functions depends only on E_+ and not E_- due to the Maxwell-Boltzmann approximation, and that the invariant rate W_{ij} depends only on s due to the neglect of final state statistical factors. Hence we can immediately integrate over E_-,

$$\int dE_- = 4p_{ij} \sqrt{\frac{E_+^2 - s}{s}}. \tag{27.29}$$

The volume element is now

$$\frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i} \frac{d^3\mathbf{p}_j}{(2\pi)^3 2E_j} = \frac{1}{(2\pi)^4} \frac{p_{ij}}{2} \sqrt{\frac{E_+^2 - s}{s}} dE_+ ds \tag{27.30}$$

We now perform the E_+ integration. We obtain

$$A = \frac{T}{32\pi^4} \sum_{ij} \int_{(m_i+m_j)^2}^{\infty} ds g_i g_j p_{ij} W_{ij} K_1\left(\frac{\sqrt{s}}{T}\right) \tag{27.31}$$

where K_1 is the modified Bessel function of the second kind of order 1.

We can take the sum inside the integral and define an effective annihilation rate W_{eff} through

$$\sum_{ij} g_i g_j p_{ij} W_{ij} = g_1^2 p_{\text{eff}} W_{\text{eff}} \tag{27.32}$$

with

$$p_{\text{eff}} = p_{11} = \frac{1}{2}\sqrt{s - 4m_1^2}. \tag{27.33}$$

In other words

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}. \tag{27.34}$$

Because $W_{ij}(s) = 0$ for $s \leq (m_i + m_j)^2$, the radicand is never negative.

In terms of cross sections, this is equivalent to the definition

$$\sigma_{\text{eff}} = \sum_{ij} \frac{p_{ij}^2}{p_{11}^2} \frac{g_i g_j}{g_1^2} \sigma_{ij}. \tag{27.35}$$

Eq. (27.31) then reads

$$A = \frac{g_1^2 T}{32\pi^4} \int_{4m_1^2}^{\infty} ds p_{\text{eff}} W_{\text{eff}} K_1 \left(\frac{\sqrt{s}}{T} \right) \tag{27.36}$$

This can be written in a form more suitable for numerical integration by using p_{eff} instead of s as integration variable. From Eq. (27.33), we have $ds = 8 p_{\text{eff}} dp_{\text{eff}}$, and

$$A = \frac{g_1^2 T}{4\pi^4} \int_0^{\infty} dp_{\text{eff}} p_{\text{eff}}^2 W_{\text{eff}} K_1 \left(\frac{\sqrt{s}}{T} \right) \tag{27.37}$$

with

$$s = 4p_{\text{eff}}^2 + 4m_1^2 \tag{27.38}$$

So we have succeeded in rewriting A as a 1-dimensional integral.

From Eqs. (27.37) and (27.15), the thermal average of the effective cross section results

$$\langle \sigma_{\text{eff}} v \rangle = \frac{\int_0^{\infty} dp_{\text{eff}} p_{\text{eff}}^2 W_{\text{eff}} K_1 \left(\frac{\sqrt{s}}{T} \right)}{m_1^4 T \left[\sum_i \frac{g_i}{g_1} \frac{m_i^2}{m_1^2} K_2 \left(\frac{m_i}{T} \right) \right]^2}. \tag{27.39}$$

This expression is very similar to the case without coannihilations, the differences being the denominator and the replacement of the annihilation rate with the effective annihilation rate. In the absence of coannihilations, this expression correctly reduces to the formula in Gondolo and Gelmini [12].

The definition of an effective annihilation rate independent of temperature is a remarkable calculational advantage. As in the case without coannihilations, the effective annihilation rate can in fact be tabulated in advance (see Section 27.5 for details), before taking the thermal average and solving the Boltzmann equation.

In the effective annihilation rate, coannihilations appear as thresholds at $\sqrt{s}$ equal to the sum of the masses of the coannihilating particles. We show an example for neutralino DM (in the mssm module) in Fig. 27.1 where it is clearly seen that the coannihilation thresholds appear in the effective invariant rate just as final state thresholds do. For the same example, Fig. 27.2 shows the differential annihilation rate per unit volume dA/dp_{eff}, the integrand in Eq. (27.37), as a function of p_{eff}. We have chosen a temperature $T = m_\chi/20$, a typical freeze-out temperature. The Boltzmann suppression contained in the exponential decay of K_1 at high p_{eff} is clearly visible. At higher temperatures the peak shifts to the right and at lower temperatures to the left. For the particular model shown in Figs. 27.1–27.2, the relic density results $\Omega_\chi h^2 = 0.030$ when coannihilations are included and $\Omega_\chi h^2 = 0.18$ when they are not. Coannihilations have lowered $\Omega_\chi h^2$ by a factor of 6.

27.1.4 Reformulation of the Boltzmann equation

We now follow Gondolo and Gelmini [12] to put Eq. (27.12) in a more convenient form by considering the ratio of the number density to the entropy density,

$$Y = \frac{n}{s}. \tag{27.40}$$

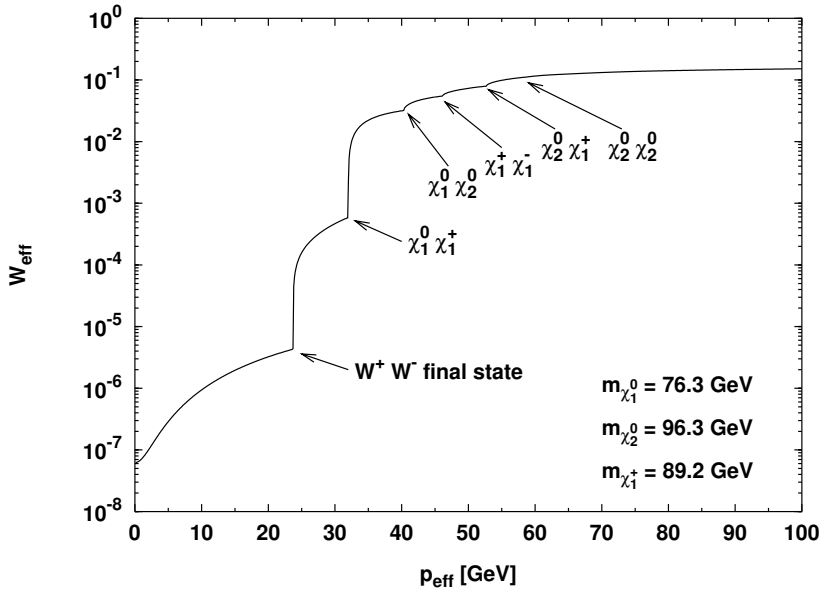

Figure 27.1: The effective invariant annihiliation rate $W_{\rm eff}$ as a function of $p_{\rm eff}$ for an example mssm model. The final state threshold for annihilation into W^+W^- and the coannihilation thresholds, as given by Eq. (27.34), are indicated. The $\chi_2^0\chi_2^0$ coannihilation threshold is too small to be seen.

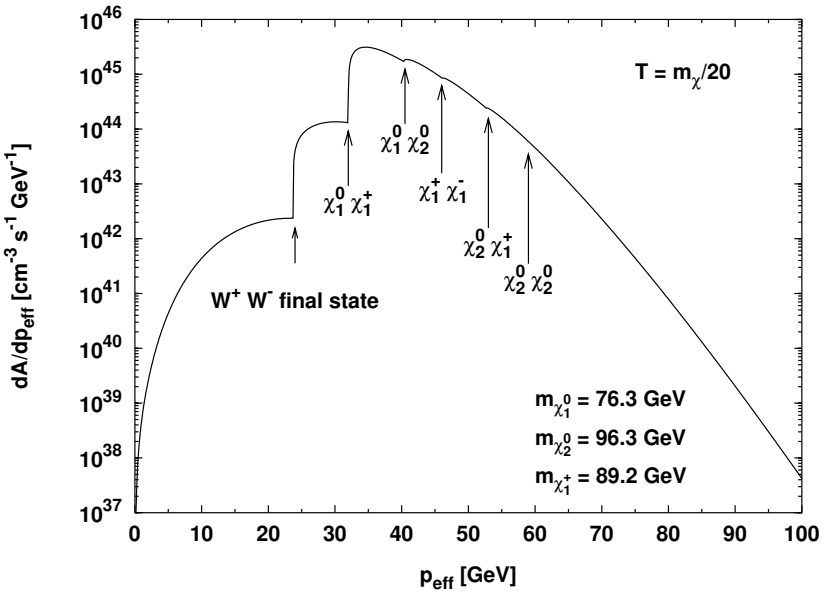

Figure 27.2: Total differential annihilation rate per unit volume $dA/dp_{\rm eff}$ for the same model as in Fig. 27.1, evaluated at a temperature $T = m_\chi/20$, typical of freeze-out. Notice the Boltzmann suppression at high $p_{\rm eff}$.

Consider

$$\frac{dY}{dt} = \frac{d}{dt}\left(\frac{n}{s}\right) = \frac{\dot{n}}{s} - \frac{n}{s^2}\dot{s} \tag{27.41}$$

where dot means time derivative. In absence of entropy production, $S = R^3 s$ is constant (R is the scale factor). Differentiating with respect to time we see that

$$\dot{s} = -3\frac{\dot{R}}{R}s = -3Hs \tag{27.42}$$

which yields

$$\dot{Y} = \frac{\dot{n}}{s} + 3H\frac{n}{s}. \tag{27.43}$$

Hence we can rewrite Eq. (27.12) as

$$\dot{Y} = -s\langle\sigma_{\rm eff} v\rangle \left(Y^2 - Y_{\rm eq}^2\right). \tag{27.44}$$

The right-hand side depends only on temperature, and it is therefore convenient to use temperature T instead of time t as independent variable. Defining $x = m_1/T$ we have

$$\frac{dY}{dx} = -\frac{m_1}{x^2}\frac{1}{3H}\frac{ds}{dT}\langle\sigma_{\rm eff} v\rangle \left(Y^2 - Y_{\rm eq}^2\right). \tag{27.45}$$

where we have used

$$\frac{1}{\dot{T}} = \frac{1}{\dot{s}}\frac{ds}{dT} = -\frac{1}{3Hs}\frac{ds}{dT} \tag{27.46}$$

which follows from Eq. (27.42). With the Friedmann equation in a radiation dominated universe

$$H^2 = \frac{8\pi G\rho}{3}, \tag{27.47}$$

where G is the gravitational constant, and the usual parameterization of the energy and entropy densities in terms of the effective degrees of freedom $g_{\rm eff}$ and $h_{\rm eff}$,

$$\rho = g_{\rm eff}(T)\frac{\pi^2}{30}T^4, \quad s = h_{\rm eff}(T)\frac{2\pi^2}{45}T^3, \tag{27.48}$$

we can cast Eq. (27.45) into the form [12]

$$\frac{dY}{dx} = -\sqrt{\frac{\pi}{45G}}\frac{g_*^{1/2} m_1}{x^2}\langle\sigma_{\rm eff} v\rangle \left(Y^2 - Y_{\rm eq}^2\right) \tag{27.49}$$

where $Y_{\rm eq}$ can be written as

$$Y_{\rm eq} = \frac{n_{\rm eq}}{s} = \frac{45x^2}{4\pi^4 h_{\rm eff}(T)}\sum_i g_i \left(\frac{m_i}{m_1}\right)^2 K_2\left(x\frac{m_i}{m_1}\right), \tag{27.50}$$

using Eqs. (27.15), (27.40) and (27.48).

The parameter $g_*^{1/2}$ is defined as

$$g_*^{1/2} = \frac{h_{\rm eff}}{\sqrt{g_{\rm eff}}}\left(1 + \frac{T}{3h_{\rm eff}}\frac{dh_{\rm eff}}{dT}\right) \tag{27.51}$$

For $g_{\rm eff}$, $h_{\rm eff}$ and $g_*^{1/2}$, the user can choose different implementations by a call to the routine dsrd_set (see that routine for further details). The default for those effective relativistic degrees of freedom is to use the results from Drees et al [88].

To obtain the relic density we integrate Eq. (27.54) from $x = 0$ to $x_0 = m_\chi/T_0$ where T_0 is the photon temperature of the Universe today. The relic density today in units of the critical density is then given by

$$\Omega_\chi = \rho_\chi^0/\rho_{\rm crit} = m_\chi s_0 Y_0/\rho_{\rm crit} \tag{27.52}$$

where $\rho_{\rm crit} = 3H^2/8\pi G$ is the critical density, s_0 is the entropy density today and Y_0 is the result of the integration of Eq. (27.54). With a background radiation temperature of $T_0 = 2.726$ K we finally obtain

$$\Omega_\chi h^2 = 2.755 \times 10^8 \frac{m_\chi}{\rm GeV} Y_0. \tag{27.53}$$

It is worth stressing that the entire formalism described above is readily adapted to the case of freeze-out happening in a secluded *dark sector*, with only relatively modest modifications [89]. This option has been fully implemented in DarkSUSY since version 6.3 [90].

27.2 Relic density – numerical integration of the density equation

Let us write the evolution equation for the density,

$$\frac{dY}{dx} = -\sqrt{\frac{\pi}{45G}} \frac{g_*^{1/2} m_1}{x^2} \langle \sigma_{\rm eff} v \rangle \left(Y^2 - Y_{\rm eq}^2 \right) \tag{27.54}$$

as

$$\frac{dY}{dx} = \lambda (Y^2 - q^2), \tag{27.55}$$

where λ contains the annihilation rate and q represents the thermal-equilibrium density.

This equation is stiff and an explicit method, like Euler or Runge-Kutta, fails to converge. To obtain a numerical solution, we use an adaptive implicit trapezoidal method which we explain in the following. Basically we discretize the equation first with a trapezoidal then with an Euler method, and adapt the step size according to the difference in the updated function values.

For simplicity we denote the right hand wide of eq. (27.55) as $f(x)$. We further write $f_i = f(x_i)$ and similarly for the other functions $\lambda(x)$ and $q(x)$. Given $Y_i = Y(x_i)$ we find $Y_{i+1} = Y(x_{i+1})$ with $x_{i+1} = x_i + h$ as follows.

First we discretize the evolution equation as

$$Y_{i+1} - Y_i = h \frac{f_i + f_{i+1}}{2}. \tag{27.56}$$

We insert

$$f_i = \lambda_i \left(Y_i^2 - q_i^2 \right), \tag{27.57}$$

$$f_{i+1} = \lambda_{i+1} \left(Y_{i+1}^2 - q_{i+1}^2 \right), \tag{27.58}$$

and solve the resulting quadratic equation for Y_{i+1} to obtain

$$Y_{i+1} = \frac{c}{1 + \sqrt{1 + uc}}, \tag{27.59}$$

where

$$c = 2Y_i + u \left[(q_{i+1}^2 + \rho q_i^2) - \rho Y_i^2 \right], \tag{27.60}$$

$$u = h\lambda_{i+1}, \tag{27.61}$$

$$\rho = \lambda_i/\lambda_{i+1}. \tag{27.62}$$

In the expression for c we have explicitly indicated the order of evaluation which we found avoids round-off errors. If in eq. (27.59) $1 + uc$ is negative, we simply reduce the step h to $h/2$ and try again.

Secondly we discretize the evolution equation as

$$Y_{i+1} - Y_i = h f_{i+1}. \tag{27.63}$$

We insert the expression for f_{i+1} and solve the quadratic equation for Y_{i+1} to obtain

$$Y'_{i+1} = \frac{1}{2} \frac{c'}{1 + \sqrt{1 + uc'}}, \tag{27.64}$$

where

$$c' = 4 \left(Y_i + u q_{i+1}^2 \right). \tag{27.65}$$

Again if in eq. (27.64) $1 + uc' < 0$, we reduce the step h to $h/2$ and try again.

We then adapt the step size according to the relative difference of Y_{i+1} and Y'_{i+1},

$$d = \left| \frac{Y_{i+1} - Y'_{i+1}}{Y_{i+1}} \right|. \tag{27.66}$$

If the difference is larger than a prefixed ϵ, set at 0.01, we reduce the step size h to $hs/\sqrt{\epsilon}$ but never to less than $h/10$. s is a safety factor set to 0.9. If $d < \epsilon$, we increase the step size by a factor $s/\sqrt{\epsilon}$ but never by more than a factor of 5. We do not allow the step size to become smaller than $h_{\min} = 10^{-9}$. Error code 5 is reported if this happens. Error code 4 occurs when x_{i+1} is numerically equal to x_i because of round-off. Error code 6 occurs when the number of steps exceeds a maximum of 100000. Finally the initial step size is taken to be 0.01.

27.3 Relic density – asymmetric dark matter

The discussion above is readily generalized to the case where DM is not self-conjugate, and where there may be an asymmetry between the number density of DM particles, n_+, and the number density of anti-DM particles, n_-. Neglecting oscillations and CP violations, the Boltzmann equations for these number densities are then given by

$$\dot{n}_\pm 3Hn_\pm = -\langle \sigma v \rangle \left(n_+ n_- - n_{\rm eq}^2 \right). \tag{27.67}$$

or, equivalently,

$$\frac{dY_\pm}{dx} = -\lambda \left(Y_+ Y_- - Y_{\rm eq}^2 \right). \tag{27.68}$$

We can then introduce an asymmetry parameter

$$\eta \equiv Y_+ - Y_- > 0\,, \tag{27.69}$$

which we assume to be fixed at a primordial stage, and to stay constant during the subsequent cosmological evolution. In particular, this requires *i)* the absence of DM-number violating interactions (such as through Majorana mass terms allowing for oscillations) and *ii)* the absence of entropy production, as in the standard formulation of freeze-out described above. The Boltzmann equation for $Y \equiv Y_-$ thus becomes

$$\frac{dY}{dx} = -\lambda \left(Y^2 + Y\eta - Y_{\rm eq}^2 \right). \tag{27.70}$$

Clearly, for $\eta \to 0$, we recover the Boltzmann equation in the symmetric case.

The *total* DM relic abundance in such a scenario is obtained as

$$\Omega_{\rm DM} h^2 = c \times \left(Y_-^\infty + Y_+^\infty \right) = c \times \left(2Y^\infty + \eta \right), \tag{27.71}$$

with

$$c = 2.755 \times 10^{10} \left(\frac{m_{\rm DM}}{100\,{\rm GeV}}\right) , \tag{27.72}$$

where Y^∞ is the abundance of the anti-DM component obtained when integrating Eq. (27.70) until today, i.e. formally in the limit of $x \to \infty$. The symmetric component – of DM (and anti-DM) – thus makes up a fraction

$$r \equiv \frac{2Y^\infty}{2Y^\infty + \eta} = \frac{1}{1 + \eta/(2Y^\infty)} \tag{27.73}$$

$$= 1 - \eta \frac{c}{\Omega_{\rm DM} h^2} \tag{27.74}$$

of the total DM density today. It is also worth noting that for $m_{\rm DM} > 4.07\,{\rm GeV}/(10^{10}\eta)$ already the asymmetric component would contribute more than the observed DM density, $\Omega_{\rm obs}h^2 = 0.112$.

The numerical implementation of Eq. (27.70) follows the same idea as in the symmetric ($\eta \to 0$), i.e. by an implicit trapezoidal method with adaptive stepsize h. Concretely, when going from x_i to $x_{i+1} = x_i + h$, we estimate the abundance $Y_{i+1} \equiv Y(x_{i+1})$ to be

$$Y_{i+1} = Y_i + \frac{h}{2}\left(Y_i' + Y_{i+1}'\right) \tag{27.75}$$

$$= \frac{\tilde{C}_i}{2} - \frac{1}{2}\tilde{\lambda}_{i+1} Y_{i+1}^2 - \frac{1}{2}\tilde{\eta}_{i+1} Y_{i+1} , \tag{27.76}$$

where $\tilde{\lambda}_i \equiv h\lambda(x_i)$, $\tilde{\eta}_i \equiv \eta\tilde{\lambda}_i$ and

$$\tilde{C}_i \equiv 2Y_i(1 - \frac{\tilde{\eta}_i}{2}) - \tilde{\lambda}_i \left(Y_i^2 - Y_{{\rm eq},i}^2\right) + \tilde{\lambda}_{i+1} Y_{{\rm eq},i+1}^2 . \tag{27.77}$$

This constitutes a quadratic equation for Y_{i+1}, solved by

$$Y_{i+1} = \frac{\tilde{C}_i / \left[1 + \tilde{\eta}_{i+1}/2\right]}{1 + \sqrt{1 + \tilde{\lambda}_{i+1}\tilde{C}_i / \left[1 + \tilde{\eta}_{i+1}/2\right]^2}} . \tag{27.78}$$

In order to estimate the local relative error we also calculate the abundance at x_{i+1} with a modified Euler method. This alternative estimate we denote by a small y_{i+1}:

$$y_{i+1} \equiv Y_i + hY_{i+1}' \tag{27.79}$$

$$= Y_i + \tilde{\lambda}_{i+1} Y_{{\rm eq},i+1}^2 - \tilde{\lambda}_{i+1} y_{i+1}^2 - \tilde{\eta}_{i+1} y_{i+1} , \tag{27.80}$$

which we solve for

$$y_{i+1} = \frac{\tilde{c}_i / \left(2\left[1 + \tilde{\eta}_{i+1}\right]\right)}{1 + \sqrt{1 + \tilde{\lambda}_{i+1}\tilde{c}_i / \left[1 + \tilde{\eta}_{i+1}\right]^2}} , \tag{27.81}$$

with

$$\tilde{c}_i \equiv 4Y_i + 4\tilde{\lambda}_{i+1} Y_{{\rm eq},i+1}^2 . \tag{27.82}$$

When solving the Boltzmann equation (27.70), as implemented in Eq. (27.78), finally, we adaptively decrease the stepsize h if

$$\left(Y_{i+1} - y_{i+}\right) / Y_{i+1} \tag{27.83}$$

exceeds a given tolerance (stored in the common block variable compeps). The initial stepsize for h, finally, is stored in the common block variable hstep.

27.4 Relic density – coupled Boltzmann equations

27.4.1 Boltzmann equation at phase-space level

Everything discussed so far rests on the assumption that the DM particles are in full *kinetic* equilibrium during the freeze-out process. If that is not the case, one must in principle directly solve the full Boltzmann equation at the phase-space level. Not taking into account coannihilations, it reads [91, 92]

$$(\partial_t - Hp\partial_p)\, f_\chi = \frac{1}{E}\left(C_{\rm ann}[f_\chi] + C_{\rm el}[f_\chi]\right) , \tag{27.84}$$

where

$$\begin{aligned} C_{\rm ann} \;=\; & \frac{1}{2g_\chi}\int \frac{d^3\tilde{p}}{(2\pi)^3 2\tilde{E}} \int \frac{d^3k}{(2\pi)^3 2\omega} \int \frac{d^3\tilde{k}}{(2\pi)^3 2\tilde{\omega}} \\ & \times (2\pi)^4 \delta^{(4)}(\tilde{p}+p-\tilde{k}-k) \\ & \times \left[|\mathcal{M}|^2_{\bar{\chi}\chi\leftarrow\bar{f}f}\, g(\omega)g(\tilde{\omega}) - |\mathcal{M}|^2_{\bar{\chi}\chi\rightarrow\bar{f}f}\, f_\chi(E) f_\chi(\tilde{E}) \right] \end{aligned} \tag{27.85}$$

describes the effect of two-body annihilations, and the collision term for elastic scattering processes is given by

$$\begin{aligned} C_{\rm el} \;=\; & \frac{1}{2g_\chi}\int \frac{d^3k}{(2\pi)^3 2\omega} \int \frac{d^3\tilde{k}}{(2\pi)^3 2\tilde{\omega}} \int \frac{d^3\tilde{p}}{(2\pi)^3 2\tilde{E}} \\ & \times (2\pi)^4 \delta^{(4)}(\tilde{p}+\tilde{k}-p-k) |\mathcal{M}|^2_{\chi f\leftrightarrow\chi f} \\ & \times \left[\left(1 \mp g^\pm(\omega)\right)\, g^\pm(\tilde{\omega}) f_\chi(\tilde{E}) - (\omega\leftrightarrow\tilde{\omega}, E\leftrightarrow\tilde{E}) \right] . \end{aligned} \tag{27.86}$$

Here, $f_\chi(t,p)$ is the DM phase-space density, and both collision terms and the squared amplitudes $|\mathcal{M}|^2$ for the respective process are implicitly summed over all heat bath particles f, and final *and* initial state internal degrees of freedom, respectively. For the heat bath particles, the phase-space distribution is given by $g^\pm(\omega) = 1/\left[\exp(\omega/T) \pm 1\right]$.

DRAKE [15] is at the time of this writing (pre-release of DarkSUSY 6.4) the only numerical tool that directly solves Eq. (27.84), which comes with its own challenges. In DarkSUSY we follow a different approach, namely considering (momentum) *moments* of Eq. (27.84). In particular, if f_χ follows a Maxwell-Boltzmann distribution, integration over $\int d^3p$ directly results in the Boltzmann equation for the number density discussed previously, Eq. (27.12). Otherwise, one needs to at least consider the second moment of the DM distribution as well, leading to a set of Boltzmann equations that couple the evolution of the DM number density and velocity dispersion, respectively. This method was first introduced in Ref. [93], and later refined in Ref. [78].[†]

27.4.2 Boltzmann equations for number density and velocity dispersion

Concretely, we obtain two coupled equations by *(i)* directly integrating Eq. (27.84) and *(ii)* first multiplying by p^2 before doing so. One can then introduce the (non-thermal) DM velocity dispersion as a 'DM temperature parameter',

$$T_\chi \equiv \frac{g_\chi}{3n_\chi}\int \frac{d^3p}{(2\pi)^3}\frac{p^2}{E} f_\chi \,, \tag{27.87}$$

[†]For the case of an interacting dark sector that is in local thermal equilibrium with itself (but with a temperature T_χ different from that of the SM heat bath), this results in an *exact* description of the DM density evolution. However, this method still provides a very reasonable approximation to the full result (at the phase-space density level) even in many other case, like e.g. in the case of Scalar Singlet DM near the Higgs resonance [78].

as well as

$$y \equiv m_\chi T_\chi s^{-2/3} \,. \tag{27.88}$$

The two coupled Boltzmann equations then take the form

$$\frac{x}{Y}\frac{dY}{dx} = \frac{sY}{\tilde{H}}\left[\frac{Y_{\rm eq}^2}{Y^2}\langle\sigma v\rangle_T - \langle\sigma v\rangle_{T_\chi}\right], \tag{27.89}$$

$$\frac{x}{y}\frac{dy}{dx} = \frac{\gamma w}{\tilde{H}}\left[\frac{y_{\rm eq}}{y}-1\right] + \frac{sY}{\tilde{H}}\left[\langle\sigma v\rangle_{T_\chi} - \langle\sigma v\rangle_{2,T_\chi}\right] + \frac{sY}{\tilde{H}}\frac{Y_{\rm eq}^2}{Y^2}\left[\frac{y_{\rm eq}}{y}\langle\sigma v\rangle_{2,T} - \langle\sigma v\rangle_T\right] + 2(1-w)\frac{H}{\tilde{H}} \,. \tag{27.90}$$

Here, a subscript T or T_χ indicates the temperature at which to take thermal averages. Introducing $s \equiv E_{\rm CM}^2/(4m_\chi^2)$, these can be expressed as

$$\langle\sigma v\rangle_T = \int_1^\infty d\tilde{s}\,\sigma_{\bar\chi\chi\to\bar f f} v_{\rm lab} \frac{2x\sqrt{\tilde{s}-1}(2\tilde{s}-1)K_1\Big(2\sqrt{\tilde{s}}x\Big)}{{K_2}^2(x)} \,, \tag{27.91}$$

which is the same thermal average as for standard freeze-out, and

$$\langle\sigma v\rangle_{2,T} = \frac{g_\chi^2}{T n_{\chi,{\rm eq}}^2}\int\frac{d^3p\,d^3\tilde{p}}{(2\pi)^6}\frac{p^2}{3E}\sigma v_{\bar\chi\chi\to\bar f f} f_{\chi,{\rm eq}}(\mathbf{p}) f_{\chi,{\rm eq}}(\tilde{\mathbf{p}}) \tag{27.92}$$

$$= \int_1^\infty d\tilde{s}\,\sigma_{\bar\chi\chi\to\bar f f} v_{\rm lab} \frac{2x\sqrt{\tilde{s}-1}(2\tilde{s}-1)K_1\Big(2\sqrt{\tilde{s}}x\Big)}{{K_2}^2(x)} \times F_2(\tilde{s},x) \tag{27.93}$$

with

$$F_2(\tilde{s},x) \equiv \frac{2\tilde{s}x^2 e^{-2\sqrt{\tilde{s}}x}}{3\sqrt{\tilde{s}-1}K_1\Big(2\sqrt{\tilde{s}}x\Big)} \tag{27.94}$$

$$\times \int_1^\infty d\epsilon_+ e^{-2\sqrt{\tilde{s}}x(\epsilon_+-1)}\left[\epsilon_+\sqrt{(\tilde{s}-1)(\epsilon_+^2-1)} + \frac{1}{2\sqrt{\tilde{s}}}\log\left(\frac{\sqrt{\tilde{s}}\epsilon_+ - \sqrt{(\tilde{s}-1)(\epsilon_+^2-1)}}{\sqrt{\tilde{s}}\epsilon_+ + \sqrt{(\tilde{s}-1)(\epsilon_+^2-1)}}\right)\right] ,$$

which is a p^2-weighted version of it.

In Eqs. (27.89,27.90), we also introduced $\tilde{H} \equiv H/\left[1+\tilde{g}(x)\right]$ with $\tilde{g} \equiv \frac{1}{3}\frac{T}{g_{\rm eff}^s}\frac{dg_{\rm eff}^s}{dT}$ and

$$w \equiv 1 - \frac{\langle p^4/E^3\rangle}{6T_\chi} = 1 - \frac{g_\chi}{6T_\chi n_{\chi,{\rm eq}}}\int\frac{d^3p}{(2\pi^3}\frac{p^4}{E^3}f_{\chi,{\rm eq}} \,. \tag{27.95}$$

Finally, γ is the momentum transfer rate – see section 26.1 and in particular Eq. (26.7).

Numerically integrating the coupled system of Eqs. (27.89,27.90) one can now obtain the DM abundance today, Y_0, and from this as usual the density $\Omega_\chi h^2$ from Eq. (27.53). The example program examples/aux/oh2_cBE_ScalarSinglet.f demonstrates how to do this with DarkSUSY for the case of ScalarSinglet DM (as opposed to examples/aux/oh2_ScalarSinglet.f, which demonstrates the same when following the standard approach based on Eq. (27.44)). More details about the numerical implementation of the underlying routines are given further down.

27.5 Relic density – tabulation of $W_{\rm eff}$

One of the most CPU-intensive tasks when calculating the relic density is the evaluation of the annihilation cross section, i.e. the effective annihilation rate $W_{\rm eff}$ in Eq. (27.34). $W_{\rm eff}$ enters in the

calculation of the thermally averaged cross section, Eq. (27.39), which then in turn enters when solving the Boltzmann equation. As we see in Eqs. (27.34) and (27.39), $W_{\rm eff}$ does not in itself depend on the temperature, just on the effective momentum $p_{\rm eff}$ as defined in Eq. (27.33). The temperature dependence enters in Eq. (27.39).

We can use the effective annihilation rate as provided by the particle module to calculate the thermal average in each step of the Boltzmann equation solving, but this will be very inefficient (unless the effective annihilation rate is a very simple expression, which it typically is not). Instead we tabulate $W_{\rm eff}$ before solving the Boltzmann equation and interpolate in this table during the Boltzmann equation solving. A crucial step in this tabulation is to use as few points as possible, while still having enough points to make the interpolated table accurate enough. For this purpose, DarkSUSY has two main ways of tabulating $W_{\rm eff}$. We will first go through the old standard way of tabulating, and then describe a new improved method. The new method is as of DarkSUSY 6.4 the default.

The different methods are chosen with the flag fast when calling dsrdomega.

27.5.1 Method A: pre-tabulation of $W_{\rm eff}$

In Fig. 27.1 we see an example of an effective annihilation rate. However, we do not need all parts of this function to be well tabulated, as can be seen in Fig. 27.2, where the Boltzmann suppression is included. The function in Fig. 27.2 is essentially the function that is integrated to get the thermally averaged cross section $\langle\sigma_{\rm eff}v\rangle$. As is clear from these figures is that it is most important to have a good tabulation for lower $p_{\rm eff}$ than for really high ones. With this in mind, the pre-tabulation method tabulates $W_{\rm eff}$ in the following way:

1. A set of points are added between $p_{\rm eff}=0$ and typically $p_{\rm eff}=2m_{\rm WIMP}$.
2. A set of points is added (sparser) from $p_{\rm eff}=2m_{\rm WIMP}$ up to $p_{\rm max}$, where $p_{\rm max}$ is typically of the order of $10m_{\rm WIMP}$.
3. A set of points are added at each resonance
4. A set of points are added at each threshold, both final state thresholds and coannihilation thresholds.
5. Optionally, a test is performed on a normalized version of the $W_{\rm eff}-p_{\rm eff}$ plane of tabulated points to see if the angle between lines connection the adjacent tabulated points is large. If it is, new points are added.
6. A spline interpolation is set up. If the difference between the spline interpolation and a linear interpolation is too large (checked midway in between tabulated points), a point is added.

Finally, a spline interpolation is used to interpolate in the table. When the actual thermal average is calculated, the integration is split at thresholds and around resonances to make sure that these are well integrated.

fast=0,1,2,3,9 all use method A (without the angle test 5 above), with decreasing accuracy (and increasingly faster calculation).

Up until DarkSUSY 6.3, the default was fast=1.

27.5.2 Method B: Breit-Wigner fit to resonances and tabulation on the fly

In this method, a completely different approach is used to tabulate $W_{\rm eff}$. First, $W_{\rm eff}$ is calculated on and near resonances to see if they can be well-fit with a Breit-Wigner form on a linear curve. We thus check if the following function is a good fit to $W_{\rm eff}$ on and near the resonance:

$$W_{\rm BW}=a+f_{\rm BW}(p_{\rm eff}) \tag{27.96}$$

$$f_{\rm BW} = \frac{k}{(E^2 - m_{\rm res}^2)^2 + m_{\rm res}^2 \Gamma_{\rm res}^2} p_{\rm eff}^{\alpha} \quad ; \quad E = 2\sqrt{m_{\rm WIMP}^2 + p_{\rm eff}^2} \tag{27.97}$$

In this expression, a, k and α are constants to be fit, whereas $m_{\rm res}$ and $\Gamma_{\rm res}$ are given from the known properties of the resonance, i.e. they are not fit.

If the fit is good, the analytical form above is used instead of the actual $W_{\rm eff}$ expression on and around the resonance (how far away it is used depends on how far away the fit is good). Note that to get good fits we include both a constant term, a linear term and a correction factor $p_{\rm eff}^{\alpha}$. On this form, the Breit-Wigner fit is usually quite good many $\Gamma_{\rm res}$ away from the peak.

Technically, the fit is done by calculating $W_{\rm eff}$ on peak and ngastart times the resonance width away from the peak. If the difference in $W_{\rm eff}$ on and off peak is larger than wresratio, the resonance is deemed important and we fit a, k and α. We then check that the fit is better than wdev. If it is, we go 50% further away from the peak and check if the fit is still better than wdev compared to the full $W_{\rm eff}$ calculation. If it is, we continue like this until we have found out how far away from the peak the fit is good.

Secondly, a few points are optionally added near thresholds. Finally, the Boltzmann solver and thermal average integration call a wrapper routine to the full $W_{\rm eff}$ expression that

1. Check if $W_{\rm eff}$ is requested for a $p_{\rm eff}$ where we have a good Breit-Wigner fit. If we do, that fit is used to calculate $W_{\rm eff}$.
2. Check if we already have calculated $W_{\rm eff}$ close to the requested $p_{\rm eff}$. If we have, we use either a linear or quadratic interpolation to interpolate between previously calculated $W_{\rm eff}$ points. If we do not have a calculation for a nearby $p_{\rm eff}$, $W_{\rm eff}$ will be evaluated and added to our table of calculated values. The criterion for when to add new points is different if we are close to resonances or thresholds.

Hence, this way, we add points on the fly where the thermal average integration routine is actually asking for $W_{\rm eff}$ values. Compare this to method A where we try to foresee where values are needed. Also with this method, the integration of the thermal average is split around resonances and at thresholds to make sure these are well-handled.

fast=20,21 use this method with decreasing accuracy (and increased speed). As of DarkSUSY 6.4, the default is to use fast=20. Compared to the previous default fast=1, the new method is typically both faster and more accurate, especially for complex $W_{\rm eff}$.

27.5.3 Method C: full expression of $W_{\rm eff}$

This is not really a tabulation method, but we include it here for completeness. There is also an option to skip tabulation completely and instead use the full $W_{\rm eff}$ expression when solving the Boltzmann equation and calculating the thermally averaged annihilation cross section. This option is included mostly for checks and debugging, as it is typically extremely slow. It can however be used when $W_{\rm eff}$ is fast to calculate and tabulations are not needed to speed things up.

27.5.4 Method D: quick and dirty

This method uses an old way of calculating the relic density by approximating the annihilation cross section as

$$\sigma v = a + b v^2$$

where a and b are constants and v is the relative velocity. The method finds a and b from evaluating $W_{\rm eff}$ at $p_{\rm eff} = 0$ and $p_{\rm eff} = 0.25 m_{\rm WIMP}$ and extracts a and b from these values. It then calculates the relic density by an approximate analytic solution to the Boltzmann equation, following Ref. [92]. The method should not really be used, and is included only for comparison. The errors from this way of calculating the relic density can be large, several orders of magnitude if there are important thresholds, coannihilations or resonances.

27.5.5 Comparison of methods

With a pre-release of DarkSUSY 6.4, we calculated the relic density for the long set of 19 MSSM models in dstest_mssm. As reference results, we used calculations with fast=99. Compared to this reference, the errors for the different methods are shown in Table 27.1. As can be seen, fast=20 is both faster and more accurate than both fast=0 and fast=1.

Method	fast	Average error	Max error	Approximate CPU time
C	99	–	–	$> 10^6$ s
A	0	0.024%	−0.150%	135 s
A	1	0.21%	1.47%	131 s
B	20	0.010%	0.025%	61 s
B	21	0.037%	0.135%	58 s

Table 27.1: Table of comparison of relic density calculations with different fast settings. These were run on a MacBook Pro with an M1 Pro processor, using a pre-release of DarkSUSY 6.4. The CPU-times are approximate.

27.5.6 Time constraints

When scanning large parameter spaces for complex particle physics models, some models can be significantly slower to calculate the relic density for than others. This happens when e.g. there are many coannihilating particles and lots of resonances and thresholds. In these cases, it is sometimes preferable to put a cap on how long a certain calculation can take. For both method A and B above, one can set an approximate maximum number of CPU seconds one is willing to spend on the relic density calculation. This is done by setting the global variable rdt_max to the maximum number of seconds for the calculation. This is then used by the code as an approximate upper limit, in practice it spends a little bit longer to wrap things up. For method A, if the upper limit is reached during tabulation of $W_{\rm eff}$, the tabulation is terminated the the relic density routine returns $\Omega h^2 = 0$. For method B, if the upper limit is reached, it will just stop adding more points to the $W_{\rm eff}$ table, but it will proceed with interpolating the table it has. Hence, in this case, one would still get a result, but maybe not as one expects. A warning flag is issued in this case to indicate that a time cut-off has been applied. The default is to not use any time constraint at all.

27.6 Relic density – routines

In src/rd, the general relic density routines are found. These routines can be used for any dark matter candidate and e.g. the interface to neutralino dark matter is in src_models/mssm/rd. The initialization of these routines happens via a call to dsrd_init (this is typically done automatically, from dsinit); dsrd_set can be used to steer general global behaviour, in particular which parameterization of the relativistic plasma (standard model) degrees of freedom that should be used.

The main routine for relic density calculations is dsrdomega. It calculates and returns the relic density as well as the approximate temperature of freeze-out. It taks a few arguments: i) one option argument that typically (depending on the particle physics module) determines how co-annihilations are included, ii) one fast argument that determines which tabulation method that should be used and how careful the Boltzmann solver should be in calculating the relic density. See Section 27.5 and the header of dsrdomega for more details.

In full analogy to dsrdomega for the symmetric case, dsrdomega_aDM calculates the relic density in the presence of a primordial asymmetry. Compared to dsrdomega, this function takes one additional input parameter η. It also has one additional output parameter, r, that describes the

late-time fraction of the symmetric DM component as introduced in Eq. (27.73). Note that η is a free input parameter here that can in principle be chosen differently from the asymmetry parameter $\eta_{\rm model}$ used to set up a model with DM particles that is not self-conjugate (see, e.g., the generic WIMP example in chapter 38) – though this is discouraged. The value of r returned by dsrdomega_aDM is always consistent with the input value of η, i.e. the result of a relic density calculation as given by Eq. (27.73).
(Indirect detection yields, on the other hand, will per default use the expression in Eq. (27.74) to define r, under the assumption that $\Omega_{\rm DM}$ corresponds to all of the observed DM. This behaviour can be changed by using the rescaling function dscrrescale_asym described in section 12.1.)

27.6.1 Global parameters

All internal settings of the relic density routines are set in common blocks in dsrdcom.h. The most important parameters that can be changed are given below. If changing these, be aware though that many of these are set in dsrdomega when it is called with a given fast setting (see dsrdomega for details).

Most important general parameters in **dsrdcom.h**

Purpose:	Provide a set of parameters, with which the internal behaviour of the relic density routines can be changed.
Parameters	
tharsi i	Size of the coannihilation, resonance and threshold arrays (default=50). Increase this size if you have more than 50 coannihilating particles, more than 50 resonances or more than 50 thresholds.
rdluerr i	Logical unit number where error messages are printed.
rdtag c*12	Idtag that is printed in case of errors.
hstep r8	Initial step size in h for Boltzmann solver
hmin r8	Minimum allowed step size in h in Boltzmann solver
compeps r8	Relative accuracy of Boltzmann solver
xinit r8	Initial value of x for Boltzmann solver
xfinal r8	Final value of x for Boltzmann solver
umax r8	Maximum value of u for $\langle\sigma v\rangle$ integration.
theps r8	Relative accuracy in thermal average integration in dsrdthav.
thepshi r8	Relative accuracy in thermal average integration for high x in dsrdthav.
ptopr r8	Cut-off upper limit in $p_{\rm eff}$ for thermal average integration. Unit: $m_{\rm WIMP}$
pmaxt r8	Maximum momentum, only used for high temperature region (freeze-in).
rdt_max r8	Maximum number of seconds to spend on relic density calculation. If the time limit is exceeded, dsrdomega returns with an error flag and the result 0 for method A, fast=0,1,2,3,9 and an approximate result for method B, fast=20,21. The time is the total CPU time (i.e. summed up over all cores/threads) and the limit is approximate as it is only checked before a new point is added to the $W_{\rm eff}$ tabulation.
rd_excl_th_ int b	exclude (or not) integration over threshold (just a tiny region, approximated with linear $W_{\rm eff}$

Most important parameters for method A in **dsrdcom.h**

Purpose:	Provide a set of parameters, for method A, fast=0,1,2,3,9.
Parameters	
waccd r8	Allowed difference between linear and spline interpolation. If difference is larger than this, a new point is inserted.
dpminr r8	Minimum allowed difference in $p_{\rm eff}$ between to points in $W_{\rm eff}-p_{\rm eff}$-table. Points are not inserted closer to each other than this. Unit: $m_{\rm WIMP}$.
dpthr r8	Scale for point insertion around thresholds. Unit: $m_{\rm WIMP}$.

wdiffr	r8	Relative difference in $W_{\rm eff}$ on and close to a resonance to deem it important.
wdifft	r8	Relative difference in $W_{\rm eff}$ below and above threshold to deem it important.
pdivr	r8	Division between low and high $p_{\rm eff}$ region. Unit: $m_{\rm WIMP}$.
dpres	r8	Scale for point insertion around resonances. Unit: $\Gamma_{\rm res}$.
cosmin	r8	maximum $\cos\theta$-difference between adjacent lines in normalized $W_{\rm eff}-p_{\rm eff}$-table. If difference is larger than this, a point is added, if cthtest=1.
cthtest	i	switch to determine if $\cos\theta$-check should be performed (1) or not (0).
spltest	i	Switch to determine if spline test should be performed (1) or not (0).
nlow	i	Number of points to add in low $p_{\rm eff}$ region
nhigh	i	Number of points to add in high $p_{\rm eff}$ region
npres	i	Number of points to add on each side of a resonance (if important).
nthup	i	Number of points to add just above a threshold.

Most important parameters for method B in **dsrdcom.h**

Purpose: Provide a set of parameters, for method B, fast=20,21.

Parameters

dpminr	r8	Minimum allowed difference in $p_{\rm eff}$ between to points in $W_{\rm eff}-p_{\rm eff}$-table. Points are not inserted closer to each other than this. Unit: $m_{\rm WIMP}$.
dpthr	r8	Scale for point insertion around thresholds. Unit: $m_{\rm WIMP}$.
rdquaderr	r8	Parameter that determines if quadratic interpolation is expected to be good enough so that we do not need to add a new point.
rdlinerr	r8	Parameter that determines if linear interpolation is expected to be good enough so that we do not need to add a new point.
wresratio	r8	Ratio of on/off peak to treat resonance as important (off-peak means ngastart number of widths from peak).
wdev	r8	Maximum deviation from Breit-Wigner fit to calculation to use fit
ngammamax	r8	Maximum number of widths of resonance to include in fit.
ngastart	r8	Number of widths from resonance to start checking Breit-Wigner fit.
almin	r8	Minimum α allowed in Breit-Wigner fit.
almax	r8	Maximum α allowed in Breit-Wigner fit.
ngafac	r8	Factor to increase nga (number of widhts) with in each step when checking a resonance.

27.6.2 Brief description of internal routines for standard Boltzmann approach

Below, the remaining routines related to relic density calculation in the standard approach (based on the Boltzmann equation for the number density) are briefly mentioned. For more details, we refer to the routines themselves.

Routine	Purpose
dsrdomega	Main routine to calculate the relic density.
dsrdens	Routine to solve the Boltzmann equation and return the relic density. Called by dsrdomega.
dsrdaddpt	To add one point in the $W_{\rm eff}$-$p_{\rm eff}$ table, method A.
dsrdbreit_wigner	Returns the Breit-Wigner function for given input parameters.
dsrdbw_get	Get $W_{\rm eff}$ from Breit-Wigner fit if we are on a well-fit resonance, method B.
dsrdbw_setup	Setup Breit-Wigner fits, method B.
dsrddeltaneff	$\Delta N_{\rm eff}$ (additional number of effective neutrino degrees of freedom)
dsrddof	Returns the SM degrees of freedom as a function of the temperature in the early Universe.

dsrddofDS Returns potential dark sector degrees of freedom as a function of the temperature in the early Universe.

dsrddpmin To return the allowed minimal distance in $p_{\rm eff}$ between two points in the $W_{\rm eff}$-$p_{\rm eff}$ plane. The returned value depends on if there is a resonance present or not at the given $p_{\rm eff}$.

dsrdeqn To solve the relic density equation by means of an implicit trapezoidal method with adaptive stepsize and termination.

dsrdequil returns entropy and equilibrium abundance of DM particles

dsrdfunc To return the invariant annihilation rate times the thermal distribution.

dsrdfuncs To provide dsrdfunc in a form suitable for numerical integration.

dsrdHubble Hubble rate (changing this allows to study the effect of modified expansion rates *during freeze-out* on the relic density)

dsrdlny To return $\ln W_{\rm eff}$ for a given $p_{\rm eff}$.

dsrdnormlz To return a unit vector in a given direction.

dsrdqad To calculate the relic density with a quick-and-dirty method. It uses the approximative expressions in Kolb & Turner with the cross section expanded in v^2.

dsrdqrkck To numerically integrate a function with a Runge-Kutta method

dsrdquad Auxiliary function to perform quadratic interpolation

dsrdreaddof Reads relevant tables for relativistic d.o.f. (as controlled by a previous call to ds_set).

dsrdrhs To calculate terms on the right-hand side in the Boltzmann equation.

dsrdsetdefaults (re-)sets default values of some common block variables in dsrdcom.h that might get changed by other routines in /rd or /fi

dsrdsingledof Effective energy d.o.f. for a semi-relativistic particle

dsrdspline To set up the table $W_{\rm eff}$-$p_{\rm eff}$ for spline interpolation, method A.

dsrdstart To sort and store information about coannihilations, resonances and thresholds in common blocks.

dsrdstate saves or resets the common blocks in dsrdcom.h to previously stored values

dsrdtab To set up the table $W_{\rm eff}$-$p_{\rm eff}$, method A.

dsrdthav To calculate the thermally averaged annihilation cross section $\langle\sigma v\rangle$ at a given temperature.

dsrdthclose returns the location of the threshold closest to a given momentum p ...

dsrdthlim To determine the end-points for the thermal average integration.

dsrdthtest To check if a given entry in the $W_{\rm eff}$-$p_{\rm eff}$ table is at a threshold.

dsrdwfunc To write out dsrdfunc for a given $x = m_\chi/T$.

dsrdwintp To return the invariant rate $W_{\rm eff}$ for any given $p_{\rm eff}$ by performing a spline interpolation in the $W_{\rm eff}$-$p_{\rm eff}$ table, method A.

dsrdwintpch To check the spline interpolation in the $W_{\rm eff}$-$p_{\rm eff}$ table and compare with a linear interpolation.

dsrdwintprint Print out a the table of invariant rate with the points that are tabulated.

dsrdwintrp To write out a table of the invariant rate $W_{\rm eff}$ and some internal integration variables and expressions.

dsrdwprint Print out a the table of invariant rate within a given interval

dsrdwres To write out the table $W_{\rm eff}$-$p_{\rm eff}$.

dsrdtabth Add a few points at thresholds, method B.

dsrdwx	To return the invariant rate $W_{\rm eff}$ for any given $p_{\rm eff}$ by either using Breit-Wigner fits, making an interpolation, or adding points on the fly to the $W_{\rm eff} - p_{\rm eff}$-table. Method B.
dsrdxi	temperature ratio $\xi = T_\chi/T$

27.6.3 Implementation of coupled Boltzmann equations

In analogy to dsrdomega for the standard approach of tracking only the DM number density, DarkSUSY also provides a function dsrdomega_cBE that calculates the relic density based on the coupled system of Boltzmann equations (27.89,27.90). This function takes the same input parameters as dsrdomega, but has additional output parameters containing the value of x up to which the Boltzmann equations were adaptively integrated (because Y would remain constant afterwards) as well as the value of y at this maximal x (from which the time of kinetic decoupling can be inferred, cf. section 26.1).

The actual integration of the Boltzmann equations is performed by dsrdens_cBE in two steps, or phases, using the SFODE algorithm from the NSWC mathematical library (optimized for n-dimensional, stiff ODEs). During the first phase, where the momentum exchange rate is much higher than the Hubble rate (i.e. long before kinetic decoupling), the problem is simplified to the standard 1D case; the right-hand side of the Boltzmann equations – as returned by dsrdrhs_cBE – is thus only given by the first of the two equations, Eq. (27.89), with $T_\chi = T$. Only in a second phase, once γ is no longer much smaller than H, dsrdrhs_cBE returns the full r.h.s. of Eqs. (27.89,27.90).

Below, the remaining routines exclusively related to relic density calculations based on coupled Boltzmann equations are briefly mentioned. For more details, we refer to the routines themselves.

Routine	Purpose
dsrdp4E3av	Computes $\langle p^4/E^3 \rangle$.
dsrdthav_2	To calculate the thermally averaged annihilation cross section $\langle \sigma v \rangle_2$ at a given temperature.
dsrdthav_select	Returns all thermal averages appearing in coupled Boltzmann equations; based on the interpolation of pre-tabulated values.
dsrdthav_tab	Tabulates the various thermal averages appearing in coupled Boltzmann equations

27.7 Routine headers – fortran files

dgadap2_thav.f

```
c######################################################################
c
c   one- and two-dimensional adaptive gaussian integration routines.
c
c   This is a special version of the dgadap2 routine that is used
c   for the option fast=20 in the relic density routines. In this case
c   the integrand (which contains dsrdwx, the invariant rate) can change
c   if new tabulation points are added on the fly. If a point is added,
c   the integration starts over to avoid integrating an inconsistent
c   integrand that changes during integration. (J. Edsjo)
c
c **********************************************************************

      subroutine dgadap2_thav(a0,b0,f,func2,eps0,sum)
**********************************************************************
**********************************************************************
```

```
c CLONE of dgadap, with external func2 as additional input
c Instead of f(x), this routine thus integrates f(func2,x)
c
c  Author: torsten.bringmann@fys.uio.no, 2018-04-26
*****************************************************************
*****************************************************************
c
c   purpose           - integrate a function f(x)
c   method            - adaptive gaussian
c   usage             - call gadap(a0,b0,f,eps,sum)
c   parameters  a0    - lower limit (input,real)
c               b0    - upper limit (input,real)
c               f     - function f(x) to be integrated. must be
c                       supplied by the user. (input,real function)
c               eps0  - desired relative accuracy. if sum is small eps
c                       will be absolute accuracy instead. (input,real)
c               sum   - calculated value for the integral (output,real)
c   precision         - single (see below)
c   req'd prog's      - f
c   author            - t. johansson, lund univ. computer center, 1973
c   reference(s)      - the australian computer journal,3 p.126 aug. -71
c
c  made real*8 by j. edsjo 97-01-17
```

dsrd_init.f

```
***********************************************************************
*** dsrd_init initializes relic density routines and sets defaults
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: June 19, 2019
***********************************************************************

      subroutine dsrd_init
```

dsrd_set.f

```
      subroutine dsrd_set(key,value)
c...set parameters for relic density routines
c...  key - character string
c...  value - character string
c...parameters implemented:
c...  key='dof'
c...    value='help' - show a list of possible values
c...    value='default' - default
c...    value='1' - Gelmini-Gondolo 150MeV
c...    value='2' - Hindmarsch-Philipsen HP-A
c...    value='3' - Hindmarsch-Philipsen HP-B (default)
c...    value='4' - Hindmarsch-Philipsen HP-B2
c...    value='5' - Hindmarsch-Philipsen HP-B3
c...    value='6' - Hindmarsch-Philipsen HP-C
c...    value='7' - Drees-Hajkarim-Schmitz, 1503.03513
c...    value='8' - Laine-Meyer, 1503.04935
c...  key='help'
c...    any value - show a list of possible keys
c...author: paolo gondolo 2007-12-29
c...modified: 2017-05-12 added Drees et al [torsten bringmann]
c...modified: 2018-01-18 directly read in dof tables because they
c...                     are also needed elsewhere [tb]
c...modified: 2021-08-08 option to add temperature dependence to Weff [tb]
c...                     made Laine tables new default
```

dsrdaddpt.f

```
      subroutine dsrdaddpt(wrate,pres,deltap,deltapminr)
c-----------------------------------------------------------------------
c  add a point in rdrate table
c  input:
c    wrate - invariant annihilation rate (real, external)
c    pres - momentum of the point to add
c    deltap - scaling factor used in dsrdtab
c    deltapminr - minimum delta p between existing point and new point
c                 (in units of WIMP mass) for it to be added
c    pmax - maximum p used in dsrdtab (from common block)
c  common:
c    'dsrdcom.h' - included common blocks
c  used by dsrdtab
c  author: joakim edsjo (edsjo@fysik.su.se)
c  modified: 01-01-31 paolo gondolo (paolo@mamma-mia.phys.cwru.edu)
c  modified: 15-12-08 Joakim Edsjo, to allow for wrate=0
c=======================================================================
```

dsrdbreit_wigner.f

```
      real*8 function dsrdbreit_wigner(mres,gres,alpha,mwimp,pwimp)
c-----------------------------------------------------------------------
c
c  input:
c    - mres - mass of resonance (GeV)
c    - gres - width of resonance (GeV)
c    - alpha, exponent of correction factor, pwimp**alpha
c    - mwimp - mass of WIMP (GeV)
c    - pwimp - momentum of WIMP (GeV)
c  output:
c    the Breit-Wigner distribution, unnormalized for a resonance of mass
c    mres and width gres, calculated at centre of mass
c    energy 2*sqrt(mwimp**2+pwimp**2)
c  author: joakim edsjo (edsjo@fysik.su.se) 2018-10-25
c=======================================================================
```

dsrdbw_get.f

```
      real*8 function dsrdbw_get(p,istat)
**********************************************************************
*** This routine returns the Breit-wigner fit to the invariant
*** annihilation rate. It only returns a value if
***   a) the resonance has been found and fit within the required
***      accuracy
***   b) the p value is within the range where the fit is good
***
*** The setup of the resonances is done with the routine dsrdbw_setup.
*** that has to be called before this routine is called.
***
*** Upon return, istat=0: within p range of well-fit resonance
***              istat=1: outside or p range, returns 0
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2018-10-26
**********************************************************************
```

dsrdbw_setup.f

```
      subroutine dsrdbw_setup(wrate,xmin)
```

```
**********************************************************************
*** This routine will go through all supplied resonances and see if
***   a) they are important enough to actually include in the calculation
***   b) if they can be fit with a Breit-Wigner form which can be used
***      instead of calling the invariant rate routine. If this is the
***      case, a fit with a momentum-dependent correction factor will be
***      applied and used and it will be determined how far away from the
***      resonance that this is useful
***
*** The routine takes input from and write outpout to common blocks
*** in dsrdcom.h. The routine dsrdwx will use this info when calculating
*** the effective invariant annihilation rate. Currently this is used
*** only for the fast=20 and fast=21 options.
***
*** Some options this routine uses are set in dsrdcom.h. Defaults are
*** set in dsrdsetdefaults. Variables from there include
***    wresratio - ratio of on/off peak to include resonance
***    wdev - maximum deviation fit/calculation to include fit
***    ngammamax - maximum number of widhts to use fit for
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2018-10-26
**********************************************************************
```

dsrddeltaneff.f

```
c-----------------------------------------------------------------------
c  Function dsrddeltaneff returns the additional number of effective
c  relativistic degrees of freedom, in units of the contribution by
c  one neutrino species.
c
c  input:
c    T - photon temperature
c
c  NB: This functions requires the functiom dsrddofDS that returns the
c      relativistic degrees of freedom in the dark sector (typically
c      provided as an interface function by the particle module, see
c      e.g. src_models/vdSIDM/rd/)
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2018-05-31
c=======================================================================
      real*8 function dsrddeltaneff(T)
```

dsrddof.f

```
      subroutine dsrddof(t,sqrtgstar,heff)
c-----------------------------------------------------------------------
c     return effective degrees of freedom at temperature t (in GeV)
c
c         sqrtgstar - g_*^0.5 as defined in (2.24) of Gelmini&Gondolo 1991
c         heff - entropy d.o.f: s=heff*(2*pi^2/45)*T^3
c
c     common:
c       'dsrdcom.h' - included common blocks
c  author: paolo gondolo (paolo@physics.utah.edu) 2005
c  modified: paolo gondolo (paolo@physics.utah.edu) 2008
c  modified: torsten bringmann 2017 -- extended header+moved geff to separate routine in /kd
c=======================================================================
```

dsrddofDS.f

```
c-----------------------------------------------------------------------
c  Function dsrddofDS returns the effective number of energy degrees of
```

```
c  freedom in the dark sector.
c
c  desc : Relativistic BSM degrees of freedom
c
c  input:
c    Td - dark sector temperature [GeV]
c
c   NB: this particular version resides in src/, and simply returns
c       a constant 0.0d0. A particle module can replace this function
c       to implement non-zero d.o.f. contribution from the dark sector
c       particles for all routines in src/rd/ (currently only dsrdrhs.f).
c       An example of this is implemented in the vdSIDM module,
c       which is straightforward to extend to other field contents
c       in the dark sector.
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2020-10-28
c=======================================================================
      real*8 function dsrddofDS(Td)
```

dsrddpmin.f

```
      real*8 function dsrddpmin(p,dpmin)
c-----------------------------------------------------------------------
c  routine to determine if there is a narrow resonance present which
c  jusifies changing dpmin to some fraction of lambda
c  author: joakim edsjo, edsjo@fysik.su.se
c  date: april 30, 1998
c  modified: april 30, 1998.
c=======================================================================
```

dsrdens.f

```
      subroutine dsrdens(wrate,oh2,xf,fast,ierr,iwar)
c-----------------------------------------------------------------------
c  present density in units of the critical density times the
c    hubble constant squared. Called from dsrdomega.
c  input:
c    wrate - invariant annihilation rate (real, external)
c    fast - see meaning of options in dsrdomega header
c  output:
c    oh2   - relic density parameter times h**2 (real*8)
c    xf    - m_WIMP / T_f where T_f is the photon temperature at freeze-out
c    ierr  - error code (integer)
c      bit 0 (1) = array capacity exceeded. increase nrmax in dsrdcom.h
c          1 (2) = a zero vector is given to dsrdnormlz.
c          2 (4) = step size underflow in dsrdeqn
c          3 (8) = stepsize smaller than minimum hmin in dsrdeqn
c          4 (16) = too many steps in dsrdeqn
c          5 (32) = step size underflow in dsrdqrkck
c          6 (64) = step size smaller than miminum in dsrdqrkck
c          7 (128) = too many steps in dsrdqrkck
c          8 (256) = gpindp integration failed in dsrdthav
c          9 (512) = threshold array too small. increase tharsi in dsrdcom.h
c         10 (1024) = calculation took longer than max time in rdt_max,
c                     calculation aborted
c    iwar  - warning code (integer)
c      bit 0 (1) = a difference of >5waccd in the ratio of w_spline
c                  and w_linear is obtained due to delta_p<dpmin.
c          1 (2) = a difference of >10waccd in the ratio of w_spline
c                  and w_linear is obtained due to delta_p<dpmin.
c          2 (4) = a difference of >15waccd in the ratio of w_spline
c                  and w_linear is obtained due to delta_p<dpmin.
c          3 (8) = wimp too heavy, d.o.f. table needs to be
c                  extended to higher temperatures. now the solution
```

```
c                  is started at a higher x than xinit (=2).
c          4 (16) = spline interpolated value too high (overflow) during
c                   check of interpolation accuracty (dsrdwintpch)
c          5 (32) = calculation took longer than max time in rdt_max,
c                   continued, but with lower accuracy (no new
c                   tabulated points)
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdtab, dsrdeqn.
c  authors: Paolo Gondolo (gondolo@lpthe.jussieu.fr) 1994-1996 and
c           Joakim Edsjo (edsjo@fysik.su.se) 30-april-98
c  modified:
c     2013-10-04 paolo gondolo: totally decouple from mssm
c     2018-04-26 torsten bringmann: version without tabulation
c=======================================================================
```

dsrdens_cBE.f

```
      subroutine dsrdens_cBE(wrate,fast,oh2,xf,xend,smallyend,ierr,iwar)
************************************************************************
***  present density in units of the critical density times the
***  hubble constant squared. Called from dsrdomega_cBE.
***
***  input:
***    wrate      = invariant annihilation rate
***                 (currently only dsrdwx is supported)
***    fast       = performance flag. See header of dsrdomega_cBE
***
***  output:
***    oh2        = \Omega h**2 for DM particle
***    xf         = m_WIMP / T_f where T_f is the freeze-out temperature
***    xend       = value of x up to which Boltmann equation is integrated
***    smallyend  = value of temperature parameter, y = m_chi T_chi s^{2/3}, at xend
***    ierr       = error code (see dsrdens)
***    iwar       = warning code (see dsrdens)
***
*** Comment: this only implements what corresponds to option 20 in
***          dsrdomega, i.e. Weff is tabulated along with <sv> caculation
***          (rather than being pre-tabulated)
***
*** author: Torsten Bringmann
*** date: 2020-12-12
************************************************************************
```

dsrdeqn.f

```
      subroutine dsrdeqn(wrate,x0,x1,y1,xf,nfcn)
c-----------------------------------------------------------------------
c  solve the relic density evolution equation by means of an implicit
c    trapezoidal method with adaptive stepsize and termination.
c  input:
c    wrate - invariant annihilation rate (real, external)
c    x0 - initial mass/temperature (real)
c    x1 - final mass/temperature (real)
c    y1 - final number/entropy densities (real)
c    nfcn - number of calls to wrate (integer)
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdrhs.
c  called by dsrdens.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1996
c  modified: joakim edsjo (edsjo@fysik.su.se) 961212
c            Paolo Gondolo, factor added 2003
```

```
c            Torsten Bringmann, added asymmetric component 07/2022
c======================================================================
```

dsrdequil.f

```
**********************************************************************
***  Subroutine dsrdequil returns entropy and equilibrium abundance
***  of DM particles.
***
***  input:
***    T - SM heat bath temperature [GeV]
***
***  output:
***    s   - entropy in SM heat bath [GeV^3]
***    pgt - 1+1/3 d(log h)/ d(log T), where h are the entropy d.o.f.
***    Yeq - Equilibrium abundance of DM particle [= neq/s]
***          based on Maxwell-Boltzmann distribution
***          NB: for non-selfconjugate DM, this is *half* of the
***              total DM abundance
***
***   Note: This routine relies on dsrdparticles and dsrdstart being
***         called previously (once per model, e.g. by a call to dsrdomega)
***
***  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2020-12-10
**********************************************************************

      subroutine dsrdequil(T,s,pgt,Yeq)
```

dsrdfunc.f

```
      function dsrdfunc(u,x,wrate)
c----------------------------------------------------------------------
c  invariant annihilation rate times thermal distribution.
c  when integrated over u, the effective thermal average times
c  m_chi^2 is obtained.
c  input:
c    u - integration variable (real)
c    x - mass/temperature (real)
c    wrate - invariant annihilation rate (real, external)
c  common:
c    'dsrdcom.h' - included common blocks
c  called by dsrdrhs, wirate, dsrdwintrp.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1996
c  modified: joakim edsjo (edsjo@fysik.su.se) 98-04-29
c  modified: torsten bringmann 2018-04-27 (caught NaN)
c  modified: torsten bringmann 2021-10 (allowed for T-dependent Weff
c                                       by adding flag Weff_finiteT)
c======================================================================
```

dsrdfuncs.f

```
      function dsrdfuncs(wrate,u)
c----------------------------------------------------------------------
c  10^15 * dsrdfunc.
c  input:
c    u - integration variable
c  uses dsrdfunc
c  used for gaussian integration with gadap.f
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 97-01-17
c  modified 2018-04-26 torsten bringmann: external wrate as input
c                                         (instead of hardcoded dsrdwintp)
c======================================================================
```

dsrdHubble.f

```
**********************************************************************
***  Function dsrdHubble returns the Hubble rate H(T) in GeV
***
***  input:
***    T - SM heat bath temperature [GeV]
***
***
***  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2020-12-10
**********************************************************************

      real*8 function dsrdHubble(T)
```

dsrdlny.f

```
      function dsrdlny(p,wrate)
c-----------------------------------------------------------------------
c  logarithm of the invariant rate.
c  input:
c    p - initial cm momentum (real)
c    wrate - invariant annihilation rate (real, external)
c  called by dsrdtab.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dsrdnormlz.f

```
      subroutine dsrdnormlz(x,y,nx,ny)
c-----------------------------------------------------------------------
c  find the unit vector (nx,ny) in the same direction as (x,y).
c  input:
c    x,y - coordinates of the vector (real)
c  output:
c    nx,ny - coordinates of the versor (real)
c  common:
c    'dsrdcom.h' - included common blocks
c  called by dsrdtab.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dsrdomega.f

```
      real*8 function dsrdomega(option,fast,xf,ierr,iwar,nfc)

**********************************************************************
*** function dsrdomega calculates omega h^2 from thermal freeze-out of
*** a dark matter particle.
***
***  type : commonly used
***  desc : Calculate the relic density $\Omega h^2$ of a dark matter particle
***
*** input:
***   option - integer parameter passed to dsrdparticles provided by particle
***            module (typically describes which coannihilations to include)
***   fast =   [default = 20]
***            0 - tabulated calculation (accuracy better than 1%)
***            1 - faster calculation: sets parameters for when to add
***                extra points less tough to avoid excessively
***                adding extra points, expected accuracy: 1% or better
***            2 - faster calculation: compared to fast=1, this option
***                adds less points in Weff tabulation in general and
***                is more elaborate in deciding when to include points
```

```
***                close to thresholds and resonances
***                expected accuracy: around 1%
***             3 - even more aggressive on trying minimize the number
***                of tabulated points
***                expected accuracy: 5-10%
***             9 - superfast. This method still makes sure to include
***                resonances and thresholds, but does not attempt to sample
***                them very well. Should give an order of magnitude estimate
***                expected accuracy: order of magnitude
***            10 - quick and dirty method, i.e. expand the annihilation
***                cross section in x (not recommended)
***                expected accuracy: can be orders of magnitude wrong
***                for models with strong resonances or thresholds
***            20 - new method to tabulate Weff along with
***                <sv> calculation (rather than independently)
***                Also uses Breit-Wigner fit to resonances if the fit
***                is good enough.
***            21 - as 20, but with more aggressive parameters to give
***                faster calculation
***            99 - uses same settings as fast=0, but instead of
***                tabulating dsanwx, it uses it directly. This is (unless
***                the annihilation cross section is fast to calculate)
***                very inefficient and slow and is included as an option
***                mostly for testing.
***
*** implicit input:
***    dsanwx -    external routine provided by particle physics module
***                to return the invariant annihilation rate W_eff
***                as a function of momentum, p_eff
***    dsrdparticles - routine provided by particle physics module
***                to return a list of coannihilating particles,
***                their properties, resonances and thresholds
***
*** output:
***   dsrdomega - omega h^2 for the TOTAL dark matter density of a given DM
***               candidate (i.e. the contributions from both the DM particle
***               and its antiparticle in case DM is not self-conjugate)
***   xf        = m_WIMP / T_f where T_f is the photon temperature at freeze-out
***   ierr      = error from dsrdens or dsrdqad, see dsrdens for details
***   iwar      = warning from dsrdens of dsrdqad, see dsrdens for details
***   nfc       - number of function calls to the effective annihilation
***               cross section
***
*** authors: joakim edsjo, paolo gondolo
*** date: 98-03-03
*** modified: 98-03-03
***           99-07-30 pg
***           02-02-27 joakim edsjo: including sfermion coanns
***           06-02-22 paolo gondolo: streamlined inclusion of coanns
***           13-10-04 paolo gondolo: totally decouple from mssm
***           18-02-28 torsten bringmann: check whether DM is self-conjugate
***           21-10-30 torsten bringmann: made sure that T-independent Weff
***              is used
***           22-12-05 joakim edsjo: new default fast=20
**********************************************************************
```

dsrdomega_aDM.f

```
**********************************************************************
*** function dsrdomega_aDM calculates omega h^2 from thermal freeze-out of  ***
*** a dark matter particle, for a given asymmetry eta = (n+ - n-)/s > 0,    ***
*** where s is the entropy density.                                         ***
***                                                                        ***
***  type : commonly used                                                  ***
```

```
***  desc : Calculate the relic density $\Omega h^2$ of an asymmetric       ***
***         dark matter particle                                            ***
***                                                                         ***
*** input:                                                                  ***
***   eta          - DM asymmetry = (n+ - n-)/s > 0                         ***
***                  (Note the conventional assumption about the sign!)     ***
***   option, fast - see dsrdomega                                          ***
***                                                                         ***
*** output:                                                                 ***
***   dsrdomega_aDM     - omega h^2 for the TOTAL dark matter density of a  ***
***                       given DM candidate (i.e. the contributions from   ***
***                       both the DM particle and its antiparticle)        ***
***   rsym              - fraction of DM in symmetric component             ***
***                       = 2*n-/(n+ + n-)                                  ***
***   xf,ierr,iwar, nfc - see dsrdomega                                     ***
***                                                                         ***
***  NB: The implementation of this routines assumes that there are no      ***
***      DM-number violating interactions in the theory (in particular      ***
***      Majorana mass terms that would allow for oscillations).            ***
***      Also note that eta here is treated as a free parameter, which does ***
***      not necessarily have to match the one used in dsgive_model_xxx     ***
***                                                                         ***
*** author: torsten.bringmann@fys.uio.no                                    ***
*** date: 2022-07-25                                                        ***
*****************************************************************************
      real*8 function dsrdomega_aDM(eta,rsym,option,fast,xf,ierr,iwar,nfc)
```

dsrdomega_cBE.f

```
      real*8 function dsrdomega_cBE(option,fast,xmax,ymax,ierr,iwar)

**********************************************************************
*** function dsrdomega_cBE calculates omega h^2 from thermal freeze-out of
*** a dark matter particle, using the coupled system of Boltzmann
*** equations described in 1706.07433.
***
***  type : commonly used
***  desc : Calculate the relic density $\Omega h^2$ of a dark matter
***  desc : particle, using coupled Boltzmann equations
***
*** input:
***   option - integer parameter passed to dsrdparticles provided by particle
***            module (typically describes which coannihilations to include)
***
*** output:
***   dsrdomega - omega h^2 for the TOTAL dark matter density of a given DM
***               candidate (i.e. the contributions from both the DM particle
***               and its antiparticle in case DM is not self-conjugate)
***   fast =  0 - [default] integrate Boltzmann equations based on pre-tabulated
***               thermal averages close to chemical and kinetic decoupling
***           1 - no pre-tabulation of thermal averages
***               (significantly slower, mostly for cross checks)
***   xf        = m_chi / T_f where T_f is the freeze-out temperature
***   xmax      = value of x up to which Boltmann equation was integrated
***   ymax      = value of temperature parameter, y = m_chi T_chi s^{2/3}, at xmax
***   ierr      = error from dsrdens or dsrdqad
***   iwar      = warning from dsrdens of dsrdqad
***
*** Comment: this only implements what corresponds to option 20 in
***          dsrdomega, i.e. Weff is tabulated along with <sv> calculation
***          (rather than being pre-tabulated)
***
*** author: torsten bringmann (building on dsrdomega)
```

```
*** date: 2020-12-12
**********************************************************************
```

dsrdp4E3av.f

```
      real*8 function dsrdp4E3av(x)
**********************************************************************
*** Function dsrdp3E4av computes the thermal average of <p^4/E^3>, as
*** defined in Eq. (33) in 1706.07433
***
***  input:
***    x   - mass/ DM temperature (real)
***          NB: this does not have to be the photon temperature
***
***  outut: <p^4/E^3>/mDM (dimensionless)
***
*** author: Torsten Bringmann
*** date: 2020-12-16
c======================================================================
```

dsrdqad.f

```
      subroutine dsrdqad(wrate,mgev,oh2,ierr)
c----------------------------------------------------------------------
c  present density in units of the critical density times the
c    hubble constant squared. quick and dirty method
c  input:
c    wrate - invariant annihilation rate (real, external)
c    mgev - relic and coannihilating mass in gev
c  output:
c    oh2 - relic density parameter times h**2 (real)
c    ierr - error code (integer)
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdtab, dsrdeqn.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1996
c  modified: joakim edsjo (edsjo@fysik.su.se) 97-05-12
c======================================================================
```

dsrdqrkck.f

```
      subroutine dsrdqrkck(f,p,wrate,x1,x2,s)
c----------------------------------------------------------------------
c  numerical integration with runge-kutta method.
c  input:
c    f - integrand (real,external)
c    p - parameter mass/temperature (real)
c    wrate - invariant rate (real,external)
c    x1 - lower limit (real)
c    x2 - upper limit (real)
c  output:
c    s - integral (real)
c  common:
c    'dsrdcom.h' - included common blocks
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c======================================================================
```

dsrdquad.f

```
      real*8 function dsrdquad(x0,y0,x1,y1,x2,y2,x,reslin,err)
**********************************************************************
```

```
*** Function dsrdquad performs a quadratic interpolation between the
*** points (x0,y0), (x1,y1) and (x2,y2). It will return value from
*** the interpolation at point x. x is assumed to be x0 <= x <= x2.
*** The points are assumed to be ordered in x, x0<x1<x2.
*** Upon return, err is the relative error compared to a linear
*** interpolation. The setup is done with Newton divided differences.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2018-10-27
**********************************************************************
```

dsrdreaddof.f

```
      subroutine dsrdreaddof(filename)
c----------------------------------------------------------------------
c     read table of effective degrees of freedom in the early universe
c     See header of dsrddof for a definition of the quantities provided
c     input:
c       filename - character*(*) - name of file containing table of dof
c     common:
c       'dsrdcom.h' - included common blocks
c  author: paolo gondolo (paolo@physics.utah.edu) 2005
c  mod: 2017-05-12 torsten bringmann (re-added fe = geff)
c  mod: 2018-02-02 torsten bringmann (initialized klo,khi to allow model-independent
c                                     use of dsrddof)
c======================================================================
```

dsrdrhs.f

```
      subroutine dsrdrhs(x,wrate,lambda,yeq,nfcn)
c----------------------------------------------------------------------
c  adimensional annihilation rate lambda in the boltzmann equation
c    y' = -lambda (y**2-yeq**2) and equilibrium dm density in units
c    of the entropy density.
c  input:
c    x - mass/ PHOTON temperature (real)
c    wrate - invariant annihilation rate (real)
c  output:
c    lambda - adimensional parameter in the evolution equation (real)
c    yeq - equilibrium number/entropy densities (real)
c    nfcn - number of calls to wrate (integer)
c  common:
c    'dsrdcom.h' - included common blocks
c  uses qrkck or dgadap.
c  called by dsrdeqn.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1996
c  modified: joakim edsjo (edsjo@fysik.su.se) 98-04-28
c  modified 98-04-28: y_eq corrected by a factor of 2 (je)
c  modified 18-05-28 (tb): allowed for Tdark != Tphoton (see dsrdxi for description)
c  modified  20-10-28 (tb): fully included dark sector contributions,
c                           now consistent with arXiv:2007.03696
c  modified  21-05-03 (tb): linking to general Hubble rate (rather than hard-coded one)
c======================================================================
```

dsrdrhs_cBE.f

```
      subroutine dsrdrhs_cBE(x,YIN,DY,RD,ID)
**********************************************************************
*** right-hand side of the 2D Boltzmann equation
***   (Y, y)' = f(x,y,Y)
*** as provided in Eqs. (27,28) in 1706.07433
***  input:
```

```
***    x   - mass/ PHOTON temperature (real)
***    YIN - (Y,y) NB: Y=n/s is here re-scaled by a factor of 1d10!
***    RD  - real array containg [currently not used]
***    ID - integer array for flags [currently not used]
***
***  output:
***    DY  - f = (f_1(x,y,Y),f_2(x,y,Y))
***
***  The format of this function follows the requirement of the ODE
***  solver SFODE supplied as part of the nswc library.
***
*** author: Torsten Bringmann
*** date: 2020-12-12
c=======================================================================
```

dsrdsetdefaults.f

```
**********************************************************************
*** subroutine dsrdsetdefaults sets default values of some common
*** block variables in dsrdcom.h. The variables that are set here are
*** those that various other routines change, e.g. those that are
*** changed in dsrdomega dpending on which fast option that is used.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: December 7, 2022
**********************************************************************

      subroutine  dsrdsetdefaults
```

dsrdsingledof.f

```
c----------------------------------------------------------------------
c  Function dsrdsingledof returns the energy density of a thermally
c  distributed single particle degree of freedom, normalized to the
c  energy density of a single bosonic degree of freedom in the highly
c  relativistic limit. At high tempertures, the return value is thus
c  1.0 (0.875) for bosons (fermions).
c
c  input:
c    x    - particle mass divided by temperature
c    stat - bosonic (stat=1) or fermionic (stat=2) d.o.f.
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2018-08-30
c======================================================================
      real*8 function dsrdsingledof(x,stat)
```

dsrdspline.f

```
      subroutine dsrdspline
c----------------------------------------------------------------------
c  set up 2nd derivatives for cubic dsrdspline interpolation.
c  common:
c    'dsrdcom.h' - included common blocks
c  called by dsrdtab.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c  modified by joakim edsjo, edsjo@fysik.su.se, to split spline
c  at thresholds.
c  modified: april 30, 1998.
c  modified: april 24, 2011 by pat scott (patscott@physics.mcgill.ca)
c   to allow code to run with array bounds-checking enabled
c======================================================================
```

dsrdstart.f

```
**********************************************************************
*** dsrdstart stores coannihilation, resonance and threshold information
*** in common blocks and sorts them
*** You do not need to add coannihilation thresholds, they are added
*** automatically from the supplied coannihilating particles.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: 03-march-98
*** modifed: 08-may-98
***   08-may-98: mdof now sorted correctly.
***   27-feb-02: allocation of threshold array corrected.
***              increased number of possible coannihilations
***   2013-10-04 paolo gondolo: completely decoupled from mssm
**********************************************************************


      subroutine dsrdstart(ncoann,mcoann,dof,nrs,rm,rw,nt,tm)
```

dsrdstate.f

```
**********************************************************************
*** dsrdstate saves or resets the common blocks in dsrdcom.h
*** It is used to make dsrdomega flexible by allowing for a
*** fast flag to conveniently change some parameters, without
*** permanently overwriting common block variables.
*** It is called with option 'save' at the beginning of dsrdomega
*** and called with option 'reset' at the end to restore.
***
*** Input: c='save'  saves the values of the common block varaiables
***                  in dsrdcom
***        c='reset' resets the common block variables in dsrdcom
***                  from saved values
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: June 19, 2019
**********************************************************************

      subroutine  dsrdstate(c)
```

dsrdtab.f

```
      subroutine dsrdtab(wrate,xmin,fast)
c---------------------------------------------------------------------
c  tabulate the invariant annihilation rate as a function of p.
c  input:
c    wrate - invariant annihilation rate (real*8, external)
c    xmin - minimum mass/temperature needed (real*8)
c    fast - options for fast tabulation (see dsrdomega header)
c  output (hidden):
c    rderr - error flag in common block
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdnormlz, dsrdlny, dsrdspline.
c  authors: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994 and
c           joakim edsjo (edsjo@fysik.su.se) 06-march-98
c=====================================================================
```

dsrdtabth.f

```
      subroutine dsrdtabth(wrate,xmin,fast)
c---------------------------------------------------------------------
c  tabulate only a few points around thresholds.
c  only used for fast=20, when dsrdwx is used.
c  input:
```

```
c    wrate - invariant annihilation rate (real*8, external)
c    xmin - minimum mass/temperature needed (real*8)
c    fast - options for fast tabulation (see dsrdomega header)
c  output (hidden):
c    rderr - error flag in common block
c  common:
c    'dsrdcom.h' - included common blocks
c  authors: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994 and
c           joakim edsjo (edsjo@fysik.su.se) 06-march-98
c  modified: Joakim Edsjö, edsjo@fysik.su.se, 2022-04-29
c=======================================================================
```

dsrdthav.f

```
      real*8 function dsrdthav(x,wrate)
c-----------------------------------------------------------------------
c  the thermal average of the effective annihilation cross section.
c  input:
c    x - mass / DM temperature (real)
c        NB: this temperature does not have to be the photon temperature!
c    wrate - invariant annihilation rate (real)
c    Input in common dsrdcom.h:
c       theps - relative accruacy of dgadap integration
c       thepshi - relative accuracy of dgadap integarion for high x
c  output:
c    dsrdthav - thermal averged cross section [GeV**-2]
c  common:
c    'dsrdcom.h' - included common blocks
c  uses qrkck or dgadap.
c  called by dsrdrhs
c  author: joakim edsjo (edsjo@fysik.su.se) 98-05-01
c  modified 2018-04-26 (torsten bringmann) : added external wrate as argument
c                                            to dsrdfuncs (and gpindp2 & dgadap2),
c                                            and updated integration to dqagp
c  modified 2021-02-07 (torsten bringmann) : caught very large x values
c  modified 2021-10-30 (torsten bringmann) : added check whether Weff is
c                                            T-dependent (Weff_finiteT)
c=======================================================================
```

dsrdthav_2.f

```
      real*8 function dsrdthav_2(x)
**********************************************************************
***  Returns the second moment of the thermally averaged effective
***  annihilation rate, as defined in (29,30) in 1706.07433.
***  input:
***    x - mass / DM temperature (real)
***        NB: this temperature does not have to be the photon temperature!
***  output:
***    dsrdthav_2 - <\sigma v>_2 [GeV**-2]
***
***  Uses dsrdthav for x<1d4; for larger values, it implements instead
***  the full expression of the highly nonrelativistic limit given in 1202.5456.
***
***  Note: Unlike dsrdthav, this function does not explicitly take the
***  invariant annihilation rate as (external) input. Instead, the function
***  dsrdwx is always used for this puporse (and hence needs to be set up
***  properly first, typically by a call to dsrdomega_cBE).

***  author: torsten bringmann
***  date: 2020-12-15
c=======================================================================
```

dsrdthav_select.f

```
      real*8 function dsrdthav_select(x,how)
**********************************************************************
***  returns thermal averages appearing on the r.h.s. of the coupled
***  system of Boltzmann equations
***
***  input:
***    x   - mass / DM temperature (real)
***          NB: this temperature does not have to be the photon temperature!
***    how - what to compute (see below)
***
***  output:
***    for how=1 - <\sigma v> [GeV**-2]
***    for how=2 - <\sigma v>_2 [GeV**-2]
***    for how=3 - w =  1d0-<p^4/E^3>/Tchi/6d0 [dimensionless]
***    for how=4 - gamma(T) [GeV]
***
***  Returns (interpolations of) tabulated values for x values between
***  xtabmin and xtabmax; these need to be set before calling this
***  routine, followed by a call to dsrdthav_tab for the actual calculation.
***  (This initialization is typically performed in dsrdens_cBE).
***  Outside this x range, the thermal averages are explicitly calculated.
***
***  Note: Unlike dsrdthav, this function does not explicitly take the
***  invariant annihilation rate as (external) input. Instead, the function
***  dsrdwx is always used for this puporse (and hence needs to be set up
***  properly first, typically by a call to dsrdomega_cBE).
***
***  author: torsten bringmann
***  date: 2021-02-23
c======================================================================
```

dsrdthav_tab.f

```
      subroutine dsrdthav_tab(how)
**********************************************************************
***  tabulates the following thermal averages:
***
***    how = 1 - <sv>
***    how = 2 - <sv>, <sv>_2, w
***
***  Tabulation starts at x=xmintab, then increases x such that there
***  are Nxtab bins per decade, as long as x<=xmaxtab. On return,
***  xmaxtab is set to highest value of x that is actually tabulated.
***
***  NB: So far, this function is hardcoded to use dsrdwx for the
***      invariant rate
***
***  author: torsten bringmann
***  date: 2021-02-23
c======================================================================
```

dsrdthclose.f

```
      real*8 function dsrdthclose(p)
c----------------------------------------------------------------------
c  returns
c  author: joakim edsjo, edsjo@fysik.su.se
c  date: april 30, 1998
c  modified: april 30, 1998.
c======================================================================
```

dsrdthlim.f

```
      subroutine dsrdthlim
c-----------------------------------------------------------------------
c  determine which limits in p_eff (or rather u) to use when
c  integrating for the thermal average
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 98-04-30
c=======================================================================
```

dsrdthtest.f

```
      logical function dsrdthtest(i)
c-----------------------------------------------------------------------
c  routine to check if the momentum p with index i is below a
c     threshold and p with index i+1 is above it. if that is the case,
c     .true. is returned, otherwise .false.
c  author: joakim edsjo, edsjo@fysik.su.se
c  date: april 30, 1998
c  modified: april 30, 1998.
c=======================================================================
```

dsrdwfunc.f

```
**************************************************************************
                      subroutine dsrdwfunc(x,wrate)
c-----------------------------------------------------------------------
c  write out dsrdfunc for the given x = mass/temperature
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdfunc
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 97-01-20
c=======================================================================
```

dsrdwintp.f

```
      function dsrdwintp(p)
c-----------------------------------------------------------------------
c  interpolation of tabulated invariant rate.
c  input:
c    p - initial cm momentum (real)
c  common:
c    'dsrdcom.h' - included common blocks
c  called by dsrdfunc.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dsrdwintpch.f

```
      subroutine dsrdwintpch(p,wspline,wlin)
c-----------------------------------------------------------------------
c  check of interpolation of tabulated invariant rate.
c  input:
c    p - initial cm momentum (real)
c  common:
c    'dsrdcom.h' - included common blocks
c  called by dsrdtab.
c  author: joakim edsjo 96-04-10
c  based on wintp.f by p. gondolo but wlin is also calculated
c=======================================================================
```

dsrdwintprint.f

```
      subroutine dsrdwintprint(unit)
No header found.
```

dsrdwintrp.f

```
**************************************************************************
                      subroutine dsrdwintrp(wrate,unit)
c-------------------------------------------------------------------------
c  write out a table of
c    initial cm momentum p
c    invariant annihilation rate w
c    integration variable u
c    integrand f
c    interpolated integrand g
c    interpolation relative error f/g-1
c  input:
c    unit - logical unit to write on (integer)
c    wrate - invariant annihilation rate (real, external)
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdtab,dsrdfunc
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c  modified:
c    2013-10-04 paolo gondolo: totally decoupled from mssm
c=========================================================================
```

dsrdwprint.f

```
      subroutine dsrdwprint(unit,np,wrate,p_min,p_max)
No header found.
```

dsrdwres.f

```
**************************************************************************
                      subroutine dsrdwres
c-------------------------------------------------------------------------
c  write out dsrdtab and check the interpolation routine
c  common:
c    'dsrdcom.h' - included common blocks
c  uses dsrdtab,dsrdwintp.f
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 96-03-26
c=========================================================================
```

dsrdwx.f

```
      real*8 function dsrdwx(p)
c-------------------------------------------------------------------------
c  Wrapper routine for dsanwx. This routine will call dsanwx to get the
c  invariant annihilation rate, but instead of calling it directly it will
c     a) call dsanwx when needed and store results
c     b) use previous results (with interpolation) if close enough in
c        momentum
c     c) consider resonances and thresholds when it determines if
c        a call to dsanwx is needed instead of tabulation
c     d) use a Breit-Wigner for for resonances iff it gives a very
c        good fit to the resonance (this is setup with dsrdbw_setup)
c  input:
c    p - momentum (GeV) in center of momentum frame
c      if p<0, invariant rate will be calculated for |p| and a call
```

```
c      to dsanwx is forced, i.e. it will not interpolate tabulated results
c      if rdwxtab (in commmon) is false, tabulation never happens (useful
c         for printing results without causing new tabulations)
c  common:
c    'dsrdcom.h' - included common blocks
c  output:
c    dsrdwx - invariant annihilation rate (i.e. the same as dsanwx)
c  Author: Joakim Edsjo (edsjo@fysik.su.se), 2018
c=====================================================================
```

dsrdxi.f

```
c----------------------------------------------------------------------
c  Function dsrdxi returns the temperature ratio for the temperature of
c  the heat bath that keeps DM in thermal equilibirum, Tdark, and the
c  photon temperature, Tphoton.
c
c  input:
c    x - mass/Tphoton
c
c  output:
c    xi = Tdark/Tphoton
c
c  NB: this particular version resides in src/, and simply returns
c      a constant 1.0d0. A particle module can replace this function
c      to implement a constant ratio different from 1, or a more
c      complicated behaviour. In this case, all routines in src/rd/ and
c      arc/kd/ will assume that the temperature of the heat bath
c      in contact with DM does *not* coincide with the photon temperature.
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2018-05-28
c=====================================================================
      real*8 function dsrdxi(x)
```

Chapter 28

se_aux: Auxiliary routines for WIMP annihilation in the Sun/Earth

28.1 Sun and Earth models – auxiliary routines

This directory contains parameterizations and semi-analytical calculations of auxiliary fluxes, like Earth atmospheric fluxes. They are not up to date with the latest calculations and are provided as is for convenience.

28.2 Routine headers – fortran files

dssea_atmmu.f

```
      real*8 function dssea_atmmu(e_mu,c_th,flt)
c ********************************************************************
c    gives muon flux from atmospheric neutrinos. uses dssea_honda.f and
c    dssea_gauss1.f.
c    based on the approximation in gaisser and stanev prd30 (1984) 985.
c    variables:
c     e_mu  muon energy in gev
c     c_th  cosine of zenith angle
c     fltype - 1 flux of muons in units of cm^-2 s^-1 sr^-1 gev^-1
c              2 cont. event rates in units of cm^-3 s^-1 sr^-1 gev^-1
c
c    output is dn/de_mu in muons per cm**2(3) per sec per sr per gev
c    l. bergstrom 1996-09-02
c    modified by j. edsjo (edsjo@fysik.su.se)
c    date: jun-03-98
c
c ********************************************************************
```

dssea_ff.f

```
      real*8 function dssea_ff(x)
No header found.
```

dssea_ff2.f

```
      real*8 function dssea_ff2(x)
No header found.
```

dssea_ff3.f

```
      real*8 function dssea_ff3(x)
No header found.
```

dssea_fff2.f

```
      real*8 function dssea_fff2(x)
No header found.
```

dssea_fff3.f

```
      real*8 function dssea_fff3(x)
No header found.
```

dssea_gauss1.f

```
      subroutine dssea_gauss1(f,a,b,result,eps,lambda)
```

dssea_honda.f

```
c ************************************************************************
       real*8 function dssea_honda(nu_type,e_nu,c_th)
c ************************************************************************
c
c      gives atmospheric neutrino flux according to m. dssea_honda et al.,
c      phys. rev. d52 (1995) 4985, by interpolating their tables iv
c      and v in log(e_nu) and cos_theta.
c
c      variables:
c                  nu_type - type of neutrino:
c
c                  nu_type=1 muon neutrino
c                  nu_type=2 muon antineutrino
c                  nu_type=3 electron antineutrino (not yet implemented)
c                  nu_type=4 electron antineutrino (not yet implemented)
c
c
c                  e_nu - neutrino energy in gev
c                  c_th - cosine of zenith angle
c      output: dssea_honda returns neutrino differential flux dn_nu/de in units
c              of cm^(-2)sec^(-1)sr^(-1)gev^(-1).
c
c      allowed energy range: 1 gev to 3.9 tev (above 3.1 tev extrapolation
c      is made)
c
c      lars bergstrom 1996-09-02
c
c
c ************************************************************************
```

dssea_ismbkg.f

```
**********************************************************************
*** dssea_ismbkg calculats the differential background of muons cosmic
*** ray interactions with the interstellar medium.
*** the muon neutrino fluxes are from
*** g. ingelman and m. thunman, hep-ph/9604286.
***   input:
***     emu - muon energy in gev
***     fltype = 1 - flux of muons
***              2 - contained event rate
***     rdelta = column density in units of nucleons / cm^2 kpc/cm
***   output:
***     muon flux in units of gev^-1 km^-2(3) yr^-1 sr^-1
*** partly based on routines by l. bergstrom.
*** author: j. edsjo (edsjo@fysik.su.se)
*** date: 1998-09-20
**********************************************************************

      real*8 function dssea_ismbkg(emu,flt,rdelta)
```

dssea_ismrd.f

```
**********************************************************************
*** function dssea_ismrd gives the column density of interstellar matter
*** along the line of sight. the model is from ingelman & thunman with
*** rho=rho_0 exp(z/z_0) with rho_0=1.0 nucleon/cm^3 and z_0=0.26 kpc
*** input: b   = galactic latitude (degrees)
***        psi = angle (in the plane) from the galactic centre (degrees)
*** output: column density in units of nucleons / cm^2 kpc/cm.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: 1998-09-20
**********************************************************************

      real*8 function dssea_ismrd(b,psi)
```

dssea_lnff.f

```
      real*8 function dssea_lnff(x)
No header found.
```

dssea_nuism.f

```
      real*8 function dssea_nuism(enu)
c...  with enu in gev, the flux is returned in units o
c...  gev^-1 cm^-2 s^-1 sr^-1
```

dssea_nusun.f

```
      real*8 function dssea_nusun(enu)
c... with enu in gev, the flux is returned in units of gev^-1 cm^-2 s^-1
```

dssea_sunbkg.f

```
**********************************************************************
*** dssea_sunbkg calculats the differential background of muons cosmic
*** ray interactions in the sun's corona. the muon neutrino fluxes are from
*** g. ingelman and m. thunman, prd 54 (1996) 4385.
***   input:
***     emu - muon energy in gev
```

```
***     fltype = 1 - flux of muons
***              2 - contained event rate
***   output:
***     muon flux in units of gev^-1 km^-2(3) yr^-1
*** partly based on routines by l. bergstrom.
*** author: j. edsjo (edsjo@fysik.su.se)
*** date: 1998-06-03
**********************************************************************

      real*8 function dssea_sunbkg(emu,flt)
```

Chapter 29

se_mod: Sun and Earth models

29.1 Sun and Earth models – theory

We need to have accurate models for the Sun and the Earth, both in terms of mass distributions, but also in terms chemical compositions. We need these to be able to calculate the capture rate of WIMPs in the Sun and the Earth accurately. These very same models are also used by external codes like **WimpSim** which simulates annihilations in the Earth and Sun (with the help of event generators) and subsequent neutrino interactions and oscillations on the way out of the Sun/Earth. Those simulation results are included as tables in `se_yield`.

In DarkSUSY we include several solar models, some of which are very detailed regarding chemical composition. Note that these details enable us to perform very detailed capture rate calculations on all relevant elements in the Sun. The solar models included are

- The BS05(OP) model with heavier elements from Grevesse and Sauval.
- The BS05(AGS,OP) model with updated heavier element fractions.
- The AGSS09 model (default), heavier element estimates from meteorites.
- The AGSS09 model with heavier elements from photospheric measurements.
- The AGS05 model.
- The GS98 model.

See **dssem_sunset.f** for details and references. The reason we have several different options for the heavier elements is that the capture rate in the Sun quite depends on these abundances, even if they are small. The reason being that the capture rate approximately goes as A^4 where A is the atomic number.

For the Earth, the model used is a standard reference model.

29.2 Sun and Earth models – routines

The main routines that could be of interest to most users are

Routine	Purpose
dssem_sundens	Density in the Sun.
dssem_sundenscomp	Composition in the Sun.

dssem_sunset	Routine to set which solar model to use.
dssem_sunmass	Mass of Sun within a given radius.
dssem_sunmfrac	Mass fraction of a given element at a given radius in the Sun.
dssem_sunput	Potential in Sun at given radius.
dssem_earthdens	Density in the Earth.
dssem_earthmass	Mass of Earth within a given radius.
dssem_earthpot	Potential in Earth at given radius.

There are more routines available, see src/se_mod for details.

29.3 Routine headers – fortran files

dssem_earthdens.f

```
c      program test
c      implicit none
c      integer i
c      real*8 radius,dssem_earthdens,dssem_earthmassint,dssem_earthmass,
c     &  dssem_earthpotint,dssem_earthpot,tmp,dssem_earthvesc,
c     &  dssem_earthdenscomp
c      real*8 depth(42)
c      data depth/0.0d0,3.0d0,15.0d0,24.0d0,80.0d0,
c     &  219.99d0,220.0d0,399.99d0,400.0d0,500.0d0,
c     &  600.0d0,669.99d0,670.0d0,770.0d0,1000.0d0,
c     &  1250.0d0,1500.0d0,1750.0d0,2000.0d0,2250.0d0,
c     &  2500.0d0,2750.0d0,2899.99d0,2900.0d0,3000.0d0,
c     &  3250.0d0,3500.0d0,3750.0d0,4000.0d0,4250.0d0,
c     &  4500.0d0,4750.0d0,5000.0d0,5149.99d0,5150.0d0,
c     &  5250.0d0,5500.0d0,5750.0d0,6000.0d0,6250.0d0,
c     &  6371.0d0,6379.0d0/   ! in km
c
cc      do i=42,1,-1
cc        radius=max((6378.140-depth(i))*1.0d3,0.0d0)
cc        write(*,*) radius,dssem_earthpotint(radius)
cc      enddo
c
c      do i=0,1100
c        radius=dble(i)/dble(1000.0d0)*6378.14d0*1.0d3
c        write(*,'(10(x,e12.6))') radius,dssem_earthdens(radius),
c     &    dssem_earthmassint(radius)/1.0d24,
c     &    dssem_earthmass(radius)/1.0d24,
c     &    dssem_earthpotint(radius),dssem_earthpot(radius),
c     &    dssem_earthvesc(radius)
c
c        write(*,'(10(x,e12.6))') radius,
c     &    dssem_earthdenscomp(radius,16),
c     &    dssem_earthdenscomp(radius,24),
c     &    dssem_earthdenscomp(radius,28),
c     &    dssem_earthdenscomp(radius,56)
c      enddo
c
c      end



**********************************************************************
*** dssem_earthdens gives the density in the earth as a function of radius
*** the radius should be given in m and the density is returned in
*** g/cm^3
*** author: joakim edsjo
*** date: march 19, 1999
```

```
**************************************************************************

      real*8 function dssem_earthdens(r)
```

dssem_earthdenscomp.f

```
**************************************************************************
*** dssem_earthdenscomp gives the number density of nucleons of mass
*** number a per cm^3
*** input: radius - in meters
***        mass number - 16 for o
***                      24 for mg
***                      28 for si
***                      56 for fe JE UPDATE!
*** the radius should be given in m and the density is returned in
*** 1/cm^3
*** author: joakim edsjo
*** date: april 6, 1999
*** Updated with new values by J. Edsjo, 2003-11-21
*** mod Helena Kolesova (2021): added C, K, N, Ar
**************************************************************************

      real*8 function dssem_earthdenscomp(r,a)
```

dssem_earthmass.f

```
**************************************************************************
*** dssem_earthmass gives the mass of the earth in units of kg within
*** a sphere with of radius r meters.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: april 1, 1999
**************************************************************************

      real*8 function dssem_earthmass(r)
```

dssem_earthmassint.f

```
**************************************************************************
*** dssem_earthmassint gives the mass of the earth in units of kg within
*** a sphere with of radius r meters.
*** in this routine, the actual integration is performed. for speed,
*** use dssem_earthmass instead which uses a tabulation of this result.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: april 1, 1999
**************************************************************************

      real*8 function dssem_earthmassint(r)
```

dssem_earthne.f

```
**************************************************************************
*** dssem_earthne gives the number density of electrons as a function
*** of the Earth's radius.
*** Input: Earth radius [m]
*** Output: n_e [cm^-3]
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2006-04-21
**************************************************************************
```

```
      real*8 function dssem_earthne(r)
```

dssem_earthpot.f

```
**********************************************************************
*** dssem_earthpot  gives the gravitational potential inside and outside
*** of the earth as a function of the radius r (in meters).
*** input: radius in meters
*** output units: s^-1
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: april 1, 1999
**********************************************************************

      real*8 function dssem_earthpot(r)
```

dssem_earthpotint.f

```
**********************************************************************
*** dssem_earthpotint gives the gravitational potential inside and outside
*** of the earth as a function of the radius r (in meters).
*** in this routine, the actual integration is performed. for speed,
*** use dssem_earthpot instead which uses a tabulation of this result.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: april 1, 1999
**********************************************************************

      real*8 function dssem_earthpotint(r)
```

dssem_earthvesc.f

```
**********************************************************************
*** dssem_earthvesc gives the escape velocity in km/s as a function of
*** the radius r (in meters) from the earth's core.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** input: radius in m
*** output escape velocity in km/s
*** date: april 1, 1999
**********************************************************************

      real*8 function dssem_earthvesc(r)
```

dssem_edfunc.f

```
************************
      real*8 function dssem_edfunc(r)
```

dssem_epfunc.f

```
************************
      real*8 function dssem_epfunc(r)
```

dssem_spfunc.f

```
************************
      real*8 function dssem_spfunc(r)
```

dssem_suncdens.f

```
************************************************************************
*** This routine uses a derived column density from the given solar model
*** The data in sdcens() is calculated by dssem_sunread.f.
************************************************************************

************************************************************************
*** dssem_suncdens gives the column density in the Sun from the
*** centre out the tha given radius r (in meters).
*** The radius should be given in m and the column density is returned in
*** g/cm^2
*** if type = 'N', the total column density (up to that r) is calculated
***         = 'p', the column density on protons is calculated
***         = 'n', the column density on neutrons is calculated
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2005-11-25
************************************************************************

      real*8 function dssem_suncdens(r,type)
```

dssem_suncdensint.f

```
************************************************************************
*** dssem_suncdensint gives the column density in the Sun from the
*** centre out the tha given radius r (in meters)
*** if type = 'N', the total column density (up to that r) is calculated
***         = 'p', the column density on protons is calculated
***         = 'n', the column density on neutrons is calculated
*** in this routine, the actual integration is performed. for speed,
*** use dssem_suncdens instead which uses a tabulation of this result.
*** Author: joakim edsjo (edsjo@fysik.su.se)
*** Date: November 24, 2005
************************************************************************

      real*8 function dssem_suncdensint(r,type)
```

dssem_suncdfunc.f

```
************************************************************************
*** dssem_suncdfunc returns the density of protons, neutrons or the total
*** density depending on the common block variable cdt. If
***   cdt='N': the total density is returned
***   cdt='p': the density in protons is returned
***   cdt='n': the density in neutrons is returned
*** the radius should be given in m and the density is returned in
*** g/cm^3.
*** This routine is used by dssem_suncdensint to calculate the column
*** density in the Sun.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2005-11-24
************************************************************************

      real*8 function dssem_suncdfunc(r)
```

dssem_sundens.f

```
************************************************************************
*** dssem_sundens gives the density in the Sun as a function of radius
*** the radius should be given in m and the density is returned in
*** g/cm^3
```

```
*** Density and element mass fractions up to O16 are from the standard
*** solar model BP2000 of Bahcall, Pinsonneault and Basu,
*** ApJ 555 (2001) 990.
*** The mass fractions for heavier elements are from N. Grevesse and
*** A.J. Sauval, Space Science Reviews 85 (1998) 161 normalized such that
*** their total mass fractions matches that of the heavier elements in
*** the BP2000 model.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2003-11-25
**************************************************************************

      real*8 function dssem_sundens(r)
```

dssem_sundenscomp.f

```
**************************************************************************
*** dssem_sundenscomp gives the number density of nucleons of atomic
*** number Z and isotope number i (as given in the definitions/files
*** in dssem_sunread) per cm^3
*** input: radius - in meters
***        z: atomic number of element
***        iso: isotope number (usually 1 for the most common and higher
***             for more rare isotopes)
*** Author: joakim edsjo
*** Date: 2003-11-26
*** Modified: 2006-03-21 (atomic mass unit fix (was off by 6%)) JE
***           2009-11-10 new more general routine with 'all' elements
***             and different isotopes
**************************************************************************

      real*8 function dssem_sundenscomp(r,z,iso)
```

dssem_sunmass.f

```
**************************************************************************
*** dssem_sunmass gives the mass of the Sun as a function of radius
*** the radius should be given in m and the mass is given in kg
*** up to the specified radius.
*** Density and element mass fractions up to O16 are from the standard
*** solar model BP2000 of Bahcall, Pinsonneault and Basu,
*** ApJ 555 (2001) 990.
*** The mass fractions for heavier elements are from N. Grevesse and
*** A.J. Sauval, Space Science Reviews 85 (1998) 161 normalized such that
*** their total mass fractions matches that of the heavier elements in
*** the BP2000 model.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2003-11-25
**************************************************************************

      real*8 function dssem_sunmass(r)
```

dssem_sunmfrac.f

```
**************************************************************************
*** dssem_sunmfrac gives the mass fraction of an element with atomic
*** number z and with isotope number iso (see dssem_sunread.f
*** for definition of iso) as a function of the solar radius r.
*** the radius should be given in m and returned is the mass fraction.
***
*** Element mass fractions are from the solar models read in in
*** dssem_sunread. Different choices are available in dssem_sunset.
```

```
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2003-11-26
**********************************************************************

      real*8 function dssem_sunmfrac(r,z,iso)
```

dssem_sunne.f

```
**********************************************************************
*** dssem_sunne gives the number density of electrons as a function
*** of the Sun's radius.
*** Input: solar radius [m]
*** Output: n_e [cm^-3]
*** See dssem_sunread for information about which solar model is used.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2006-03-27
**********************************************************************

      real*8 function dssem_sunne(r)
```

dssem_sunne2x.f

```
**********************************************************************
*** dssem_sunne2x takes an input number density of electrons and
*** converts this to a fractional solar radius, x.
*** Input: n_e [cm^-3]
*** Output: x = r/r_sun [0,1]
*** See dssem_sunread for information about which solar model is used.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2006-03-27
**********************************************************************

      real*8 function dssem_sunne2x(ne)
```

dssem_sunpot.f

```
**********************************************************************
*** This routine uses a derived potential from the BP2000 model
*** The data in sdphi() is calculated by dssem_sunread.f.
**********************************************************************

**********************************************************************
*** dssem_sunpot gives the potential in the Sun as a function of radius
*** the radius should be given in m and the potential is returned in
*** m^2 s^-2
*** Density and element mass fractions up to O16 are from the standard
*** solar model BP2000 of Bahcall, Pinsonneault and Basu,
*** ApJ 555 (2001) 990.
*** The mass fractions for heavier elements are from N. Grevesse and
*** A.J. Sauval, Space Science Reviews 85 (1998) 161 normalized such that
*** their total mass fractions matches that of the heavier elements in
*** the BP2000 model.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2003-11-26
**********************************************************************

      real*8 function dssem_sunpot(r)
```

dssem_sunpotint.f

```
************************************************************************
*** dssem_sunpotint gives the gravitational potential inside and outside
*** of the sun as a function of the radius r (in meters).
*** in this routine, the actual integration is performed. for speed,
*** use dssem_sunpot instead which uses a tabulation of this result.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: april 1, 1999
************************************************************************

      real*8 function dssem_sunpotint(r)
```

dssem_sunread.f

```
      subroutine dssem_sunread

************************************************************************
*** Reads in data about the solar model used and stores it in a
*** common block (as described in dssem_sun.h).
*** Author: Joakim Edsjo
*** Date: 2003-11-25
*** Modified: 2004-01-28 (calculates potential instead of reading file)
***           2009-11-10 Added possibility to read solar models from
***             Serenelli et al.
************************************************************************
```

dssem_sunset.f

```
      subroutine dssem_sunset(c)
c...determine which solar model to use
c...  c - character string specifying choice to be made
c...author: joakim edsjo, 2009-11-10
```

dssem_sunvesc.f

```
************************************************************************
*** dssem_sunvesc gives the escape velocity in km/s as a function of
*** the radius r (in meters) from the sun's core.
*** author: joakim edsjo (edsjo@fysik.su.se)
*** input: radius in m
*** output escape velocity in km/s
*** date: 2003-11-26
************************************************************************

      real*8 function dssem_sunvesc(r)
```

dssem_sunx2z.f

```
************************************************************************
*** The Sun routines uses different variables to describe position
*** in the Sun:
***    r:  radius (in meters)   [0,r_sun]
***    x:  radius in units of r_sun  [0,1]
***    z:  fraction of total column density traversed  [0,1]
***        the column density is either of p or n or the total
***        and the totals are stored in cd_sun
***
*** This routine converts from x to z (in p, n or total)
```

```
***
*** Inputs
***       x = radius in units of r_sun [0,1]
***    type = 'N', the total column density (up to that r) is calculated
***         = 'p', the column density on protons is calculated
***         = 'n', the column density on neutrons is calculated
***
*** Outputs
***       z = fraction of total column density (for chosen type) that
***           has been traversed from the centre of the Sun out to x.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2005-11-25
**********************************************************************

      real*8 function dssem_sunx2z(x,type)
```

dssem_sunz2x.f

```
**********************************************************************
*** The Sun routines uses different variables to describe position
*** in the Sun:
***    r:  radius (in meters)   [0,r_sun]
***    x:  radius in units of r_sun  [0,1]
***    z:  fraction of total column density traversed  [0,1]
***        the column density is either of p or n or the total
***        and the totals are stored in cd_sun
***
*** This routine converts from z (in p, n or total) to x
***
*** Inputs
***       z =  = fraction of total column density (for chosen type) that
***              has been traversed from the centre of the Sun
***    type = 'N', the total column density (up to that r) is calculated
***         = 'p', the column density on protons is calculated
***         = 'n', the column density on neutrons is calculated
***
*** Outputs
***       x = radius in units of r_sun [0,1] that corresponds to the
***           supplied z value
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2005-11-25
**********************************************************************

      real*8 function dssem_sunz2x(z,type)
```

Chapter 30

se_nu: Capture and annihilation in the Sun/Earth

30.1 Neutrinos from the Sun and Earth – theory

Note. This chapter is not up to date with the latest additions to the code, but provided as is as a reference right now.

There are several indirect methods for detection of WIMPs. One of the most promising [94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 19] is to make use of the fact that scattering of halo neutralinos by the Sun and the planets, in particular the Earth, during the several billion years that the Solar system has existed, will have trapped these neutralinos within these astrophysical bodies. Being trapped within the Solar or terrestrial material, they will sink towards the center, where a considerable enrichment and corresponding increase of annihilation rate will occur.

Searches for neutralino annihilation into neutrinos will be subject to extensive experimental investigations in view of the new neutrino telescopes (AMANDA, IceCube, Baikal, NESTOR, ANTARES) planned or under construction [109]. A high-energy neutrino signal in the direction of the centre of the Sun or Earth is an excellent experimental signature which may stand up against the background of neutrinos generated by cosmic-ray interactions in the Earth's atmosphere.

There are several different approximations one could do, or processes to include when calculating the capture rates in the Earth/Sun and many of these are coded into DarkSUSY. The default in DarkSUSY is always to use the best calculations available, but more approximate (older) routines are also available, as well as more speculative signals, like the Damour-Krauss signal (not included by default). If you want to use something else than the defaults, or want to call more internal rotuines (more internal than dsntrates or dsntdiffrates), you should read the following sections carefully.

30.1.1 Neutrino yield from annihilations

The differential neutrino flux from neutralino annihilation is

$$\frac{dN_\nu}{dE_\nu} = \frac{\Gamma_A}{4\pi D^2} \sum_f B_\chi^f \frac{dN_\nu^f}{dE_\nu} \tag{30.1}$$

where Γ_A is the annihilation rate, D is the distance of the detector from the source (the central region of the Earth or the Sun), f is the neutralino pair annihilation final states, and B_χ^f are the

branching ratios into the final state f. dN_ν^f/dE_ν are the energy distributions of neutrinos generated by the final state f and are obtained from the PYTHIA simulations described in section **??**.

In comparison with calculations using the results of [110] (e.g. [111, 104, 112, 107, 106, 113, 114]), this Monte Carlo treatment of the neutrino propagation through the Sun does not need the simplifying assumptions previously made, namely neutral currents are no more assumed to be much weaker than charged currents and energy loss is no more considered continuous.

The neutrino-induced muon flux may be detected in a neutrino telescope by measuring the muons that come from the direction of the centre of the Sun or Earth. For a shallow detector, this usually has to be done in the case of the Sun by looking (as always the case for the Earth) at upward-going muons, since there is a huge background of downward-going muons created by cosmic-ray interactions in the atmosphere. There is always in addition a more isotropic background coming from muon neutrinos created on the other side of the Earth in such cosmic-ray events (and also from cosmic-ray interactions in the outer regions of the Sun). The flux of muons at the detector is given by

$$\frac{dN_\mu}{dE_\mu} = N_A \int_{E_\mu^{\rm th}}^{\infty} dE_\nu \int_0^{\infty} d\lambda \int_{E_\mu}^{E_\nu} dE'_\mu \; P(E_\mu, E'_\mu; \lambda) \; \frac{d\sigma_\nu(E_\nu, E'_\mu)}{dE'_\mu} \; \frac{dN_\nu}{dE_\nu} \,, \tag{30.2}$$

where λ is the muon range in the medium (ice or water for the large detectors in the ocean or at the South Pole, or rock which surrounds the smaller underground detectors), $d\sigma_\nu(E_\nu, E'_\mu)/dE'_\mu$ is the weak interaction cross section for production of a muon of energy E'_μ from a parent neutrino of energy E_ν, and $P(E_\mu, E'_\mu; \lambda)$ is the probability for a muon of initial energy E'_μ to have a final energy E_μ after passing a path–length λ inside the detector medium. $E_\mu^{\rm th}$ is the detector threshold energy, which for "small" neutrino telescopes like Baksan, MACRO and Super-Kamiokande is around 1 GeV. Large area neutrino telescopes in the ocean or in Antarctic ice typically have thresholds of the order of tens of GeV, which makes them sensitive mainly to heavy neutralinos (above 100 GeV) [19].

The integrand in Eq. (30.2) is weighted towards high neutrino energies, both because the cross section σ_ν rises approximately linearly with energy and because the average muon energy, and therefore the range λ, also grow approximately linearly with E_ν. Therefore, final states which give a hard neutrino spectrum (such as heavy quarks, τ leptons and W or Z bosons) are usually more important than the soft spectrum arising from light quarks and gluons.

30.1.2 Evolution of the number density in the Earth/Sun

Neutralinos are steadily being trapped in the Sun or Earth by scattering, whereas annihilations take them away. Let $N(t)$ be the total number of neutralinos trapped, at time t, in the core of, for example, the Earth. The annihilation rate of neutralino pairs can be written as

$$\Gamma_a(t) = \frac{1}{2} \; C_a \, N^2(t) \,. \tag{30.3}$$

The evolution of $N(t)$ is the result of the competition between capture and annihilation:

$$\frac{dN}{dt} = C_c(t) - C_a \, N^2 \tag{30.4}$$

The constant C_c is the capture rate, and C_a entering equations (30.3) and (30.4) is linked to the annihilation cross-section σ_a, and to some effective volumes V_j, $j = 1, 2$, taking into account the quasi-thermal distribution of neutralinos in the Earth core:

$$C_a = \langle \sigma_a \, v \rangle \, \frac{V_2}{V_1^2} \,, \tag{30.5}$$

$$V_j \simeq 2.3 \times 10^{25} \left(\frac{j\, m_X}{10\,\mathrm{GeV}} \right)^{-3/2} \mathrm{cm}^3\,. \tag{30.6}$$

This has the solution for the annihilation rate implemented in DarkSUSY

$$\Gamma_A = \frac{C_c}{2} \tanh^2 \left(\frac{t}{\tau} \right), \tag{30.7}$$

where the equilibration time scale $\tau = 1/\sqrt{C_c C_a}$. In most cases for the Sun, and in the cases of observable fluxes for the Earth, τ is much smaller than a few billion years, and therefore equilibrium is often a good approximation ($\dot{N}(t) = 0$). This means that it is the capture rate which is the important quantity that determines the neutrino flux. However, in the program we keep the exact formula (30.7), with some modifications discussed in Sec. **??**).

30.1.3 Approximate capture rate expressions

The capture rate induced by scalar (spin-independent) interactions between the neutralinos and the nuclei in the interior of the Earth or Sun is the most difficult one to compute, since it depends sensitively on the Higgs mass, form factors, and other poorly known quantities. However, this spin-independent capture rate calculation is the same as for direct detection treated in Chapter 17. Therefore, there is a strong correlation between the neutrino flux expected from the Earth (which is mainly composed of spin-less nuclei) and the signal predicted in direct detection experiments [19, 115]. It seems that even the large (kilometer-scale) neutrino telescopes planned, when searching for neutralino annihilation in the Earth, will not be competitive with the next generation of direct detection experiments when it comes to detecting neutralino dark matter. However, the situation concerning the Sun is more favourable. Due to the low counting rates for the spin-dependent interactions in terrestrial detectors, high-energy neutrinos from the Sun constitute a competitive and complementary neutralino dark matter search. Of course, even if a neutralino is found through direct detection, it will be extremely important to confirm its identity and investigate its properties through indirect detection. In particular, the mass can be determined with reasonable accuracy by looking at the angular distribution of the detected muons [116, 117].

For the Sun, dominated by hydrogen, the axial (spin-dependent) cross section is important and relatively easy to compute. A reasonably good approximation is given by [118]

$$\frac{C_\odot^{\mathrm{sd}}}{(1.3 \cdot 10^{23}\,\mathrm{s}^{-1})\,(270\ \mathrm{km\,s}^{-1}/\bar{v})} = \left(\frac{\rho_\chi}{0.3\ \mathrm{GeV\,cm}^{-3}} \right) \left(\frac{100\,\mathrm{GeV}}{m_\chi} \right) \left(\frac{\sigma_{p\chi}^{\mathrm{sd}}}{10^{-40}\ \mathrm{cm}^2} \right) \tag{30.8}$$

where $\sigma_{p\chi}^{\mathrm{sd}}$ is the cross section for neutralino-proton elastic scattering via the axial-vector interaction, $\bar{v}$ is the dark-matter velocity dispersion, and ρ_χ is the local dark matter mass. The capture rate in the Earth is dominated by scalar interactions, where there may be kinematic and other enhancements, in particular if the mass of the neutralino almost matches one of the heavy elements in the Earth. For this case, a more detailed analysis is called for, which is available in [119] with convenient approximations in [118]. In fact, also for the Sun the spin-independent contribution can be important, in particular iron may contribute non-negligibly. For the Sun, the approximation in [118] is also available,

$$\begin{aligned} \frac{C_\odot^{\mathrm{si}}}{(4.8 \cdot 10^{22}\,\mathrm{s}^{-1})\,(270\ \mathrm{km\,s}^{-1}/\bar{v})} &= \left(\frac{\rho_\chi}{0.3\ \mathrm{GeV\,cm}^{-3}} \right) \left(\frac{100\,\mathrm{GeV}}{m_\chi} \right) \times \\ &\sum_A \left(\frac{\sigma_A^{\mathrm{si}}}{10^{-40}\ \mathrm{cm}^2} \right) F_A(m_\chi) f_A \phi_A S\left(m_\chi/m_A\right)/m_A, \end{aligned} \tag{30.9}$$

where f_A is the mass fraction of element A and ϕ_A is the typical gravitational potential (relative to the surface) for that element. I.e. an element that is concentrated in the core will have a higher ϕ_A

		Average parameters	
Element	Mass number (A)	f_i	ϕ_i
Hydrogen, H	1	0.670	3.15
Helium-4, ^{4}He	4	0.311	3.40
Carbon, C	12	0.00237	2.85
Nitrogen, N	14	0.00188	3.83
Oxygen, O	16	0.00878	3.25
Neon, Ne	20	0.00193	3.22
Magnesium, Mg	24	0.000733	3.22
Silicon, Si	28	0.000798	3.22
Sulphur, S	32	0.000550	3.22
Iron, Fe	56	0.00142	3.22

Table 30.1: The composition of the Sun with average parameters to be used in the approximative relations given in [118]. These values are updated with the solar model of [120] and differs slightly from the values used in [118].

	Mass	Mass fraction		Average parameters	
Element	number (A)	Core	Mantle	f_i	ϕ_i
Oxygen, O	16	0.0	0.440	0.298	1.20
Silicon, Si	28	0.06	0.210	0.162	1.24
Magnesium, Mg	24	0.0	0.228	0.154	1.20
Iron, Fe	56	0.855	0.0626	0.319	1.546
Calcium, Ca	40	0.0	0.0253	0.0171	1.20
Phosphor, P	30	0.002	0.00009	0.00071	1.56
Sodium, Na	23	0.0	0.0027	0.00183	1.20
Sulphur, S	32	0.019	0.00025	0.0063	1.59
Nickel, Ni	59	0.052	0.00196	0.0181	1.57
Aluminum, Al	27	0.0	0.0235	0.0159	1.20
Chromium, Cr	52	0.009	0.0026	0.0047	1.44

Table 30.2: The composition of the Earth's core and mantle. The core mass fractions are from [121][Table 4] and the mantle mass fractions are from [121][Table 2]. The average mass fractions and potentials in the last two columns are weighted averages assuming a core mass of $1.93 \cdot 10^{24}$ kg and a mantle mass of $4.04 \cdot 10^{24}$ kg with average potentials (relative to the surface) of 1.6 in the core and 1.2 in the mantle [119].

than an element at the surface. A is the atomic number of the element and M_A is its mass. The factor S is a kinematical suppression factor [118, 103]. In the next subsection we will go through the compositions of the Earth/Sun that we use.

The approximate capture rate expressions above are coded into the routines dsntcapsun and dsntcapearth. More accurate expressions will follow in the coming subsections.

30.1.4 Earth and Sun composition

When the capture rates are calculated, we need to know the composition and density of the Earth/Sun as a function of depth.

In [118] they used average mass fractions and potentials for the location of the various elements in the Sun. We have updated these to the BP2000 [120] values instead, as given in Table 30.1

For the Earth, we have also implemented more accurate density profiles and more up-to date chemical distributions within the Earth. We use the estimates for the Earth composition given in [121][Table 2 for the mantle and Table 4 for the core]. In Table 30.2 we list these values together with

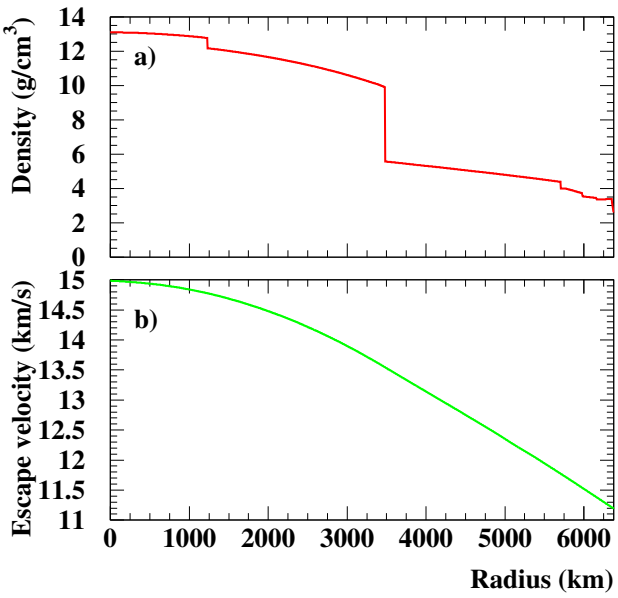

Figure 30.1: In a) the density profile and in b) the escape velocity in the Earth is shown.

the average parameters f_i and ϕ_i that should be used in the expressions for the approximate capture rates in the previous section. Note that using these average parameters instead of integrating over the full radius is equivalent to putting all the elements of the give type at the gravitational potential ϕ_i.

We also need the density profile of the Earth, and for this we use the values in [122]. Using this density profile, we can calculate the gravitational potential, $\phi(r)$ inside the Earth and from this one the escape velocity v inside the Earth,

$$v = 11.2\sqrt{\frac{\phi(r)}{\phi(R_\oplus)}}\ \text{km/s}. \tag{30.10}$$

In Fig. 30.1 we show the density profile and escape velocity inside the Earth.

30.1.5 More accurate capture rate expressions

Another complicating factor when calculating the capture rates is the integration over the velocity distribution. In [119], parts of the integrations are performed analytically for a Gaussian veolocity distributions. These expressions are also coded in DarkSUSYfor the Earth and give a more accurate calculation of the capture rate in the Earth than the approximations given above. The routine dsntcapearth2 performs these calculations for the Earth.

30.1.6 Accurate capture rates in the Earth for general velocity distributions

If one wants even more accurate and general expressions for the capture rates in the Sun/Earth, we have also implemented the full expressions in [119], but without assuming that the velocity distribution is a Gaussian (or Maxwell-Boltzmann). These routines are now the default in DarkSUSY.

We will here outline how these expressions look like for the Earth and how they can be used both for a Maxwell-Boltzmann distribution and for a general velocity distribution. The expressions will of course look analogously for the Sun. We start with the general case and study the special case of a Maxwell-Boltzmann distribution in the next section.

We will divide the Earth into shells and calculate the capture from element i in each shell individually. At the end we will integrate over all the shells and sum over all the elements in the Earth. The capture rate from element i per unit shell volume is given by [119][Eq. (2.8)]

$$\frac{dC_i}{dV} = \int_0^{u_{max}} du \frac{f(u)}{u} w\Omega^-_{v,i}(w) \tag{30.11}$$

where $f(u)$ is the velocity distribution (normalized such that $\int_0^\infty f(u) = n_\chi$ where n_χ is the number density of WIMPs. The expression $\Omega^-_{v,i}(w)$ is related to the probability that we scatter to orbits below the escape velocity. w is the velocity at the given shell and it is related to the velocity at infinity u and the escape velocity v by

$$w = \sqrt{u^2 + v^2}. \tag{30.12}$$

The upper limit of integration is a priori set to $u_{max} = \infty$, but we will see below that due to kinematical reasons we can set it to a lower value (Eq. (30.17) below). If we allow for a form factor suppression of the form [119][Eq. (A3)]

$$|F(q^2)|^2 = \exp\left(-\frac{\Delta E}{E_0}\right) \tag{30.13}$$

with [119][Eq. (A4)]

$$E_0 = \frac{3\hbar^2}{2m_\chi R^2} \tag{30.14}$$

we can evaluate $w\Omega^-_{v,i}(w)$ and arrive at the expression [119][Eq. (A6)]

$$w\Omega^-_{v,i}(w) = \sigma_i n_i \frac{\mu_+^2}{\mu} 2E_0 \left[e^{-\frac{m_\chi u^2}{2E_0}} - e^{-\frac{\mu}{\mu_+^2} m_\chi \frac{u^2+v^2}{2E_0}}\right] \Theta\left(\frac{\mu}{\mu_+^2} - \frac{u^2}{u^2+v^2}\right) \tag{30.15}$$

where we have introduced

$$\mu = \frac{m_\chi}{m_i} \quad ; \quad \mu_\pm = \frac{\mu \pm 1}{2} \tag{30.16}$$

with m_i the mass of element i. The Heaviside step function Θ plays the role of only including WIMPs that can scatter to a velocity lower then the escape velocity v. To simplify our calculations we can drop this step function in Eq. (30.15) and instead set the upper limit of integration in Eq. (30.11) to

$$u_{max} = \sqrt{\frac{\mu}{\mu_-^2}} v \tag{30.17}$$

We also need the scattering cross section on element i, which can be written as [118][Eq. (9-25)]

$$\sigma_i = \sigma_p A_i^2 \frac{(m_\chi m_i)^2}{(m_\chi + m_i)^2} \frac{(m_\chi + m_p)^2}{(m_\chi m_p)^2} \tag{30.18}$$

where A_i is the atomic number of the element, m_p is the proton mass and σ_p is the scattering cross section on protons.

We now have what we need to calculate the capture rate. In Eq. (30.11) we integrate over the velocity for our chosen velocity distribution. We then integrate this equation over the radius of the Earth and sum over all the different elements in the Earth,

$$C = \int_0^{R_\oplus} dr \sum_i \frac{dC_i}{dV} 4\pi r^2 \tag{30.19}$$

Note that we have not assumed anything special about our velocity distribution, it doesn't even have to be isotropic since the distribution of elements evenly in the shells will make an anisotropic distribution on average to behave as an isotropic one.

The routines that calculate the capture rates with these general (and accurate) expressions are dsntcapearthnum and dsntcapsunnum. As these calculations are somewhat time-consuming, we have also added a possibility to tabulate the result and interpolate in these tables. To use (or create, if the table files are missing) instead call dsntcapearthtab and dsntcapsuntab. These last two routines are the default in DarkSUSY. The velocity distribution used is determined by a switch when the halo model is set (i.e. when dshm_set is called).

30.1.7 Accurate capture rates for the Earth for a Maxwell-Boltzmann velocity distribution

We will here give some more information on how the approximations introduced in the beginning of this chapter are derived from the general expressions in the preceeding section.

If the velocity distribution is of Maxwell-Boltzmann type we can greatly simplify our expressions above as we can perform the integration over velocity analytically. The integration over radius can also be further simplified by using the average mass fractions f_i and potentials ϕ_i in Tables 30.1–30.2.

If the velocity distribution in the halo is Maxwell-Boltzmann, it looks like

$$f_h(u)du = n_\chi \frac{4}{\sqrt{\pi}} \left(\frac{3}{2}\right)^{\frac{3}{2}} \frac{u^2}{\bar{v}^3} e^{-\frac{3}{2}\frac{u^2}{\bar{v}^2}} du \tag{30.20}$$

where $\bar{v}$ is the three-dimensional velocity dispersion and n_χ is the number density of WIMPs in the halo. However, the solar system moves through the halo with a velocity v_* and the distribution on observer with this velocity through the halo will see is

$$f_*(u) = f_h(u) e^{-\frac{3}{2}\frac{v_*^2}{\bar{v}^2}} \frac{\sinh\left(\frac{3uv_*}{\bar{v}^2}\right)}{\frac{3uv_*}{\bar{v}^2}} = n_\chi \sqrt{\frac{3}{2\pi}} \frac{u}{\bar{v}v_*} \left[e^{-\frac{3}{2}\frac{(u-v_*)^2}{\bar{v}^2}} - e^{-\frac{3}{2}\frac{(u+v_*)^2}{\bar{v}^2}} \right] \tag{30.21}$$

Now one would naively believe that this is not the distribution that an observer at the Earth will see. First of all, the Earth is moving with respect to the Sun and secondly, the WIMPs have gained speed by the gravitational attraction of the Sun when they reach the Earth. Both of these arguments are true and the distribution of WIMPs in the halo will not look like Eq. (30.21) to an observer on the Earth. However, Gould [123] showed that WIMPs from the halo can diffuse into the solar system due to gravitational interactions with the planets and this distribution of WIMPs will roughly look like as if the Earth was in free space moving through the halo with the velocity of the solar system, i.e. Eq. (30.21). We will later scrutinize this statement, as it turns out that it does not quite hold, but as a first guess it is a reasonable approximation. For the Sun, though, the velocity distribution give above is the correct one for a Maxwell-Boltzmann distribution.

With the distribution Eq. (30.21) we can analytically perform the integration over velocity in Eq. (30.11). After some algebra we arrive at [119][Eq. (A10)]

$$\begin{aligned} \frac{dC_i}{dV} &= \left(\frac{8}{3\pi}\right)^{\frac{1}{2}} \frac{\sigma_i n_i n_\chi \bar{v}}{2b\eta} \\ &\quad \left[\frac{e^{-a\hat{\eta}^2}}{\sqrt{1+a}} \left[2\widetilde{\mathrm{erf}}(\hat{\eta}) - \widetilde{\mathrm{erf}}(\hat{A}_+) + \widetilde{\mathrm{erf}}(\hat{A}_-) \right] \right. \\ &\quad \left. - \frac{e^{-b\check{\eta}^2}}{\sqrt{1+b}} e^{-(a-b)A^2} \left[2\widetilde{\mathrm{erf}}(\check{\eta}) - \widetilde{\mathrm{erf}}(\check{A}_+) + \widetilde{\mathrm{erf}}(\check{A}_-) \right] \right] \end{aligned} \tag{30.22}$$

where $\widetilde{\mathrm{erf}}$ is the modified error function,

$$\widetilde{\mathrm{erf}}(x) = \frac{\sqrt{\pi}}{2}\mathrm{erf}(x) \quad ; \quad \mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-y^2}dy. \tag{30.23}$$

Following Gould [119], we have in Eq. (30.22) introduced the following shorthand notation:

$$\begin{array}{lll} \eta = \sqrt{\frac{3}{2}\frac{v_*^2}{\bar{v}^2}} & ; \quad a = \frac{m_e h i \bar{v}^2}{3E_0} & ; \quad b = \frac{\mu}{\mu_+^2}a \\ \hat{\eta} = \frac{\eta}{\sqrt{1+a}} & ; \quad \check{\eta} = \frac{\eta}{\sqrt{1+b}} & \\ A^2 = \frac{3}{2}\frac{v_*^2}{\bar{v}^2}\frac{\mu}{\mu_-^2} & ; \quad \hat{A} = A\sqrt{1+a} & ; \quad \check{A} = A\sqrt{1+b} \\ \hat{A}_\pm = \hat{A} \pm \hat{\eta} & ; \quad \check{A}_\pm = \check{A} \pm \check{\eta} & \end{array} \tag{30.24}$$

If we wish, we can now integrate Eq. (30.22) over radius just like in the previous section, but we can without loosing too much accuracy, replace this integration with a sum over the elements in the Earth with their respective typical location. I.e. we can write

$$C = \sum_i \frac{dC_i}{dV}\frac{1}{n_i}\frac{f_i M_\oplus}{m_i} \tag{30.25}$$

where we instead of the number density n_i use the total number of atoms of the given type $f_i M_\oplus/m_i$. Note that for each element in the sum we should evaluate this expression at the given typical gravitational potential ϕ_i of the element, i.e. with the escape velocity given by Eq. (30.10). The mass fractions f_i and typical potentials ϕ_i are listed in Table 30.2 (and analogously in Table 30.1 for the Sun). This approximation introduces an error of no more than about 1–2% for a Maxwell-Boltzmann distribution*

The capture rate evaluated with the expressions shown here are encoded into the routine dsntcapearth2. Note that we have not coded the corresponding approximate expressions for the Sun. Instead, as given in the preceeding section, we now have more accurate expressions for both the Sun and the Earth.

30.1.8 Effects of WIMP diffusion in the solar system

As the Earth has a rather low escape velocity, the Earth will only be able to capture WIMPs that have a rather low velocity with respect to the Earth. However, WIMPs from the halo have gained speed in the gravitaional potential from the Sun and will essentially be impossible to capture by the Earth. Hence, the Earth will only capture WIMPs that have diffused around in the solar system (by gravitational interactions with the other planets). Gould showed [123] that effectively this diffusion will lead to the same phase space distribution at the Earth as if the Earth was in free space (i.e. neglecting the solar potential). However, numerical simulations of asteroids showed that they are thrown into the Sun due to perturbations of the orbits by other planets, see e.g. [124]. These analyses led to worried that maybe the population of WIMPs diffusing around in the solar system is not as big as thought [125]. In [126], Lundberg and Edsjö investigated this issue with detailed numerical simulations of WIMP orbits in the solar system, showing that the annihilation rate in the Earth is typically reduced by up to two orders of magnitude. In DarkSUSY, we include these results for the neutrino rates from the Earth by using the velocity distribution at the Earth (as obtained in [126]). This velocity distribution is then used as input for our numerical capture rate routines instead of the usual approximation of using the halo velocity distribution directly. Using these new velocity distributions for the Earth is the default in DarkSUSY.

*Note that it is not advisable to use this approximation for general velocity distributions. If one e.g. has a lower limit on possible velocities, u_{min}, for heavy WIMPs capture will then only be possible very close to the central core. Replacing the actual distribution of potentials $\phi(r)$ with the typical value ϕ_i may then introduce larger errors. We will encounter these kind of distributions shortly.

30.2 Neutrinos from Sun and Earth – routines

This folder contains routines to calculate the neutrino-induced muon flux from the Earth and the Sun in various models.

There are several different methods of calculation available (determined by `secalcmet` in `dssecom.h`). Method 1 uses the approximate formulae for the capture rates in the Earth/Sun from the Jungman, Kamionkowski and Griest review [118]. Method 2, uses the same expression for the Sun, but the full expression from Gould [119] for capture in the Earth. Method 4 uses a full numerical integration over the velocity distribution (instead of assuming that it is Gaussian) and method 5, finally, also performs a full numerical integration over the momentum transfer in the form factors (instead of assuming exponential form factors). The default is to use method 5, but instead of using the numerical routines directly, tables are used. One can always force using the numerical routines instead of the tables if one so wishes. There are several options regarding how many elements to incluce in the Sun summation over elements ('lo', 'med' and 'hi'). The easiest way to select method is by calling `dssenu_set`. For the Earth, method 5 is not expected to make a large difference due to the smaller momentum transfers in the Earth and method 5 reverts back to method 4 for the Earth.

A call to `dssenu_set('default')` is made in `dsinit`, but can be changed by the user by calling `dssenu_set` after `dsinit`.

To calculate the neutrino-induced muon flux from the Earth, you call **dssenu_rates**.

30.3 Routine headers – fortran files

dssenu_annrate.f

```
      subroutine dssenu_annrate(mx,rho,sigsip,sigsdp,sigma_v,calcmet,
     &  arateea,aratesu)
c--------------------------------------------------------------------------
c
c     wimp annihilation rate in the sun and in the earth
c     in units of 10^24 annihilations per year
c
c     also gives the capture rate and the annih/capt equilibration time
c
c    november, 1995
c    uses routines by p. gondolo and j. edsjo
c    modified by l. bergstrom and j. edsjo and p. gondolo
c    capture rate routines are written by l. bergstrom
c    input:  mx      - wimp mass [GeV]
c            rho - local halo density [GeV/cm^3]
c            sigsip  - spin-indep wimp-proton cross section in cm^2
c            sigsdp  - spin-dep wimp-proton cross section in cm^2
c            sigma_v  - wimp self-annihilation cross section in cm^3/s
c            calcmet - method for calculation (usually passed from secalcmet
c                   when called)
c               1 = JKG approximation
c               2 = Full Gould, analytic Gaussian vel dist approx
c               4 = Full Gould, numerical vel dist integration
c    output: arateea  - 10^24 annihilations per year, earth
c            aratesu  - 10^24 annihilations per year, sun
c    slightly modified by j. edsjo.
c    modified by j. edsjo 97-05-15 to match new inv. rate convention
c    modified by j. edsjo 97-12-03 to match muflux3.21 routines.
c    modified by p. gondolo 98-03-04 to detach it from susy routines.
c
c==========================================================================
```

dssenu_annrateff.f

```
      subroutine dssenu_annrateff(mx,rho,gps,gns,gpa,gna,sigma_v,arateea,
     &  aratesu)
c---------------------------------------------------------------------
c
c     wimp annihilation rate in the sun and in the earth
c     in units of 10^24 annihilations per year
c
c     also gives the capture rate and the annih/capt equilibration time
c
c    november, 1995
c    uses routines by p. gondolo and j. edsjo
c    modified by l. bergstrom and j. edsjo and p. gondolo
c    capture rate routines are written by l. bergstrom
c    input:  mx      - wimp mass in GeV
c            rho - local halo density in GeV/cm^3
c            sigsip  - spin-indep wimp-proton cross section in cm^2
c            sigsdp  - spin-dep wimp-proton cross section in cm^2
c            sigma_v  - wimp self-annihilation cross section in cm^3/s
c            rescale - rescale factor for local density
c    output: arateea  - 10^24 annihilations per year, earth
c            aratesu  - 10^24 annihilations per year, sun
c    slightly modified by j. edsjo.
c    modified by j. edsjo 97-05-15 to match new inv. rate convention
c    modified by j. edsjo 97-12-03 to match muflux3.21 routines.
c    modified by p. gondolo 98-03-04 to detach it from susy routines.
c    modified by J. Edsjo, 2010-05-28 to allow for numerical
c      form factor integration
c
c=====================================================================
```

dssenu_capcom.f

No header found.

dssenu_capearth.f

```
**********************************************************************
*** note. this routine assumes a maxwell-boltzmann velocity
*** distribution and uses approximations in the jkg review,
*** Jungman, Kamionkowski and Griest, Phys. Rep. 267 (1996) 195.
*** In particular, it is assumed that the Sun's velocity is
*** sqrt(2/3)*vd_3d.
*** for more accurate results, use dssenu_capearthfull instead.
*** for an arbitrary velocity distribution, use dssenu_capearthnumi instead.
**********************************************************************

      real*8 function dssenu_capearth(mx,rho,sigsi)
c---------------------------------------------------------------------
c         capture rate in the earth
c         based on jungman, kamionkowski, griest review
c       mx: WIMP mass
c       sigsi: spin independent cross section in units of cm^2
c       lars bergstrom 1995-12-14
c---------------------------------------------------------------------
```

dssenu_capearth2.f

```
**********************************************************************
*** note. this routine uses the full expressions for the capture
*** rate in the earth from gould, apj 521 (1987) 571.
```

```
*** this routine replaces dssenu_capearth which use the approximations
*** given in the jkg review.
**********************************************************************


       real*8 function dssenu_capearth2(mx,rho,sigsi)
c---------------------------------------------------------------------
c         capture rate in the earth
c       uses the full routines instead of jkg (as in dssenu_capearth).
c *** full: use formulas by gould as reported in jkg
c
c       mx: WIMP mass
c       sigsi: spin independent cross section in units of cm^2
c       vbar: 3D WIMP velocity dispersion in the halo
c       vstar: Sun's velocity through the halo
c       lars bergstrom 1998-09-15
c---------------------------------------------------------------------
```

dssenu_capearthfull.f

```
**********************************************************************
*** full capture rate routines for the earth.
*** this set of routines use the full expressions for the capture rate
*** in the earth as given in gould, apj 321 (1987) 571.
*** dssenu_capearthfull thus replaces dssenu_capearth which use the approximations
*** given in the jkg review. these routines assume a maxwell-
*** boltmann velocity distribution. for the damour-krauss population
*** of wimps, the routine dssenu_capearthnumi should be used instead.
**********************************************************************

      real*8 function dssenu_capearthfull(mx,sigsi,v_star,v_bar,rho_x)
c---------------------------------------------------------------------
c         capture rate in the earth
c *** full: use formulas by gould ap.j. 321 (1987) 571
c       mass fractions and phi_i from jkg review
c       mx: WIMP mass
c       sigsi: spin independent cross section in units of cm^2
c       v_star: solar system velocity through halo (220 km/s in standard case)
c       v_bar: 3D velocity dispersion of wimps (270 km/s in standard case)
c       rho_x: local wimp density (units of gev/cm**3)
c       lars bergstrom 1998-09-21
c       Modified by J. Edsjo, 2003-11-22
c References: gould: Gould ApJ 321 (1987) 571
c---------------------------------------------------------------------
```

dssenu_capearthnum.f

```
**********************************************************************
*** dssenu_capearthnum calculates the capture rate at present.
*** Intead of using the assumptions of Gould (i.e. capture as in
*** free space), a tabulted velocity distribution based on detailed
*** numerical simulations of Johan Lundberg is used.
*** A numerical intregration has to be performed instead of the
*** convenient expressions in jkg.
*** Input: mx = WIMP mass in GeV
***        rho = local halo density in GeV/cm^3
***        sigsi = spin-independent scattering cross section in cm^2
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: July 10, 2003
**********************************************************************
      real*8 function dssenu_capearthnum(mx,rho,sigsi)
```

dssenu_capearthnumi.f

```
c  dssenu_capearthnumi.f:
**********************************************************************
*** dssenu_capearthnumi gives the capture rate of WIMPs in the earth
*** given a specified velocity distribution. the integrations over
*** the earth's radius and over the velocity distribution are
*** performed numerically
*** input: mx [ gev ]
***        sigsi [ cm^2 ]
***        foveru [ cm^-3 (cm s^-1)^-2 ] external function f(u)/u with
***          velocity u [ km s^-1 ] as argument.
***        vt (velocity type): 1=general type, 2=DK-type (i.e. only
***          include non-zero parts, deprecated )
*** output: capture rate [ s^-1 ]
*** l.b. and j.e. 1999-04-06
*** Modified by Joakim Edsjo 2003-07-10 to allow for arbitrary external
*** velocity distributions foveru.
**********************************************************************

      real*8 function dssenu_capearthnumi(mx,sigsi,foveru,vt)
```

dssenu_capearthtab.f

```
**********************************************************************
*** This routine calculates the capture rates in the Earth.
*** It does the same thing as dssenu_capearthnum (i.e. dssenu_capearthnumi),
*** except that it uses tabulted versions of the results intead
*** of performing a numerical integration every time.
*** Inputs: mx - WIMP mass in GeV
***         rho - local halo density in GeV/cm^3
***         sigsi - spin-independent capture rate in cm^2
***         type - type of distribution (same as in dssenu_capearthnum)
*** Author: Joakim Edsjo
*** Date: 2003-11-27
**********************************************************************

      real*8 function dssenu_capearthtab(mx,rho,sigsi)
```

dssenu_capsun.f

```
       real*8 function dssenu_capsun(mx,rho,sigsi,sigsd)
c---------------------------------------------------------------------
c         capture rate in the sun
c         based on jungman, kamionkowski, griest review
c       mx: WIMP mass
c       sigsi: spin independent cross section in units of cm^2
c       sigsd: spin dependent cross section in units of cm^2
c       output:
c         capture rate in s^-1
c       lars bergstrom 1995-12-12
c---------------------------------------------------------------------
```

dssenu_capsunnum.f

```
**********************************************************************
*** dssenu_capsunnum calculates the capture rate at present.
*** Instead of using the approximations in jkg, i.e. a gaussian
*** velocity distribution and approximating all elements as being
*** at their typical radius, we here integrate numerically over
*** the actual velocity distribution and over the Sun's radius.
```

```
*** The velocity distribution used is the one set up by the
*** option veldf in dshmcom.h (see src/hm/dshmudf.f for details)
*** Input: mx = WIMP mass in GeV
***        rho = local halo density in GeV/cm^3
***        sigsi = spin-independent scattering cross section (cm^2)
***        sigsd = spin-dependent scattering cross section on protons (cm^2)
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: 2003-11-26
**********************************************************************
      real*8 function dssenu_capsunnum(mx,rho,sigsi,sigsd)
```

dssenu_capsunnumff.f

```
**********************************************************************
*** dssenu_capsunnumff calculates the capture rate at present.
*** Instead of using the approximations in jkg, i.e. a gaussian
*** velocity distribution and approximating all elements as being
*** at their typical radius, we here integrate numerically over
*** the actual velocity distribution and over the Sun's radius.
*** The velocity distribution used is the one set up by the
*** option veldf in dshmcom.h (see src/hm/dshmudf.f for details)
*** Compared to dssenu_capsunnum, this routine does not assume exponential
*** form factors, instead an actual integration over the form factors
*** is performed.
*** Input: mx = WIMP mass in GeV
***        rho = local halo density in GeV/cm^3
***        gps,gns,gpa,gna - WIMP-nucleon four-fermion couplings
***          (obtained from e.g. dsddgpgn), unit: GeV^-4
*** author: joakim edsjo (edsjo@fysik.su.se)
*** date: 2003-11-26
*** modified: je 2009-11-13
**********************************************************************
      real*8 function dssenu_capsunnumff(mx,rho,gps,gns,gpa,gna)
```

dssenu_capsunnumffi.f

```
**********************************************************************
*** dssenu_capsunnumffi gives the capture rate of WIMPs in the sun
*** given a specified velocity distribution. the integrations over
*** the sun's radius and over the velocity distribution are
*** performed numerically.
*** Compared to dssenu_capsunnumi, the form factors are not assumed to be
*** exponential, and an actual integration over the form factors is
*** performed
*** input: mx [ gev ]
***        gps,gns,gpa,gna - WIMP-nucleon four-fermion couplings
***          (obtained from e.g. dsddgpgn), [GeV^-4]
***        foveru [ cm^-3 (cm s^-1)^-2 ] external function f(u)/u with
***          velocity u [ km s^-1 ] as argument.
*** Author: Joakim Edsjo
*** Date: 2003-11-26
*** Modified: 2010-05-19 to allow for more elements and arbitrary form factors
**********************************************************************

      real*8 function dssenu_capsunnumffi(mx,gps,gns,gpa,gna,foveru)
```

dssenu_capsunnumi.f

```
c  dssenu_capsunnumi.f:
**********************************************************************
*** dssenu_capsunnumi gives the capture rate of WIMPs in the sun
*** given a specified velocity distribution. the integrations over
*** the sun's radius and over the velocity distribution are
```

```
*** performed numerically
*** input: mx [ gev ]
***        sigsi [ cm^2 ]
***        sgisd [ cm^2 ]
***        foveru [ cm^-3 (cm s^-1)^-2 ] external function f(u)/u with
***          velocity u [ km s^-1 ] as argument.
*** Author: Joakim Edsjo
*** Date: 2003-11-26
**********************************************************************

      real*8 function dssenu_capsunnumi(mx,sigsi,sigsd,foveru)
```

dssenu_capsuntab.f

```
**********************************************************************
*** This routine calculates the capture rates in the Sun.
*** It does the same thing as dssenu_capsunnum (i.e. dssenu_capsunnumi),
*** except that it uses tabulted versions of the results intead
*** of performing a numerical integration every time.
*** Inputs: mx - WIMP mass in GeV
***         rho - local halo density in GeV/cm^3
***         sigsi - spin-independent capture rate in cm^2
*** Author: Joakim Edsjo
*** Date: 2003-11-27
**********************************************************************

      real*8 function dssenu_capsuntab(mx,rho,sigsi,sigsd)
```

dssenu_capsuntabff.f

```
**********************************************************************
*** This routine calculates the capture rates in the Sun.
*** It does the same thing as dssenu_capsunnumff (i.e. dssenu_capsunnumffi),
*** except that it uses tabulted versions of the results intead
*** of performing a numerical integration every time.
*** Inputs: mx - WIMP mass in GeV
***         rho - local halo density in GeV/cm^3
***         gps,gns,gpa,gna - WIMP-nucleon four-fermion couplings
***           (obtained from e.g. dsddgpgn), unit: GeV^-4
*** Author: Joakim Edsjo
*** Date: 2015-06-12
**********************************************************************

      real*8 function dssenu_capsuntabff(mx,rho,gps,gns,gpa,gna)
```

dssenu_ceint.f

```
**********************************************************************
*** l.b. and j.e. 1999-04-06
*** auxiliary function for r-integration
*** input: radius in centimeters
*** output: integrand in cm^-1 s^-1
**********************************************************************

      real*8 function dssenu_ceint(r,foveru)
```

dssenu_ceint2.f

```
c...
```

```
c...auxiliary function for inner integrand
c...input: velocity relaitve to earth in km/s
c...output: integrand in cm^-4
c...We here follow the analysis in Gould, ApJ 321 (1987) 571 and more
c...specifically the more general expressions in appendix A.
      real*8 function dssenu_ceint2(u,foveru)
```

dssenu_csint.f

```
**************************************************************************
*** l.b. and j.e. 1999-04-06
*** auxiliary function for r-integration
*** input: radius in centimeters
*** output: integrand in cm^-1 s^-1
*** Adapted for the Sun by J. Edsjo, 2003-11-26
**************************************************************************

      real*8 function dssenu_csint(r,foveru)
```

dssenu_csint2.f

```
c...
c...auxiliary function for inner integrand
c...input: velocity relaitve to sun in km/s
c...output: integrand in cm^-4
c...We here follow the analysis in Gould, ApJ 321 (1987) 571 and more
c...specifically the more general expressions in appendix A.
      real*8 function dssenu_csint2(u,foveru)
```

dssenu_csintff.f

```
**************************************************************************
*** l.b. and j.e. 1999-04-06
*** auxiliary function for r-integration
*** input: radius in centimeters
*** output: integrand in cm^-1 s^-1
*** Adapted for the Sun by J. Edsjo, 2003-11-26
*** Modified to allow for numerical form factor integration by
*** J. Edsjo, 2010-05-28
**************************************************************************

      real*8 function dssenu_csintff(r,foveru)
```

dssenu_csintff2.f

```
c...
c...auxiliary function for inner integrand
c...Compared to dssenu_csint2.f, this routine integrates numerically over
c...the form factors
c...input: velocity relaitve to sun in km/s
c...output: integrand in cm^-4
c...We here follow the analysis in Gould, ApJ 321 (1987) 571 and more
c...specifically the more general expressions in appendix A, but further
c...generalized to allow for cut-off velocities at e.g. Jupiter and to
c...allow for numerial integration over form factors
      real*8 function dssenu_csintff2(u,foveru)
```

dssenu_csintff3.f

```
c...
c...auxiliary function for inner integrand
c...This function here is essentially the integrand in Gould (A5),
c...ApJ 321 (1987) 571, but with sigma added and a more general form factor
c...than the exponential one in Gould A5.
c...input: y=Delta-E / (mx*w**2/2)
c...output: integrand in units of cm^2

      real*8 function dssenu_csintff3(y)
```

dssenu_ctabcreate.f

```
      subroutine dssenu_ctabcreate(wh,i)

**********************************************************************
*** Creates tabulated capture rates (apart from cross section)
*** Input: wh = 'su' or 'ea' for sun or earth
***        i = table number to store the results in
*** Author: Joakim Edsjo
*** Date: 2003-11-27
**********************************************************************
```

dssenu_ctabffcreate.f

```
      subroutine dssenu_ctabffcreate(wh,i)

**********************************************************************
*** Creates tabulated capture rates (apart from couplings)
*** Input: wh = 'su' or 'ea' for sun or earth
***        i = table number to store the results in
*** Author: Joakim Edsjo
*** Date: 2015-06-12
**********************************************************************
```

dssenu_ctabffget.f

```
      real*8 function dssenu_ctabffget(wh,ct,mx)

**********************************************************************
*** Interpolates in capture rate tables and returns the capture
*** rate (apart from the coupling).
*** This routine is for the full form factor integration.
*** Input: wh ('su' or 'ea' for sun or earth)
***        ct, copuling type:
***           1 = gps**2
***           2 = gns**2
***           3 = gps*gns
***           4 = gpa**2
***           5 = gna**2
***           6 = gpa*gna
***        mx WIMP mass in GeV
*** Hidden input: velocity distribution model as given in
*** veldf (for the Sun) and veldfearth (for the Earth)
*** and possible escape velocity corrections (Jupiter effects)
*** Author: Joakim Edsjo
*** Date: 2015-06-12
**********************************************************************
```

dssenu_ctabffread.f

```
      subroutine dssenu_ctabffread(wh,i,file)

**********************************************************************
*** Reads in tabulated capture rates (apart from couplings)
*** Input: wh = 'su' or 'ea' for sun or earth (Earth not yet implemented)
***        i = table number to read
***        file = file name to read
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2015-06-12
**********************************************************************
```

dssenu_ctabffwrite.f

```
      subroutine dssenu_ctabffwrite(wh,i,file)

**********************************************************************
*** Writes out tabulated capture rates (apart from couplings)
*** Input: wh = 'su' or 'ea' for sun or earth
***        i = table number to write
***        file = file to write to
*** Author: Joakim Edsjo
*** Date: 2015-06-12
**********************************************************************
```

dssenu_ctabget.f

```
      real*8 function dssenu_ctabget(wh,st,mx)

**********************************************************************
*** Interpolates in capture rate tables and returns the capture
*** rate (apart from the cross section)
*** Input: wh ('su' or 'ea' for sun or earth)
***        st, spin-type (1:spin-independent, 2:spin-dependent)
***        mx WIMP mass in GeV
*** Hidden input: velocity distribution model as given in
*** veldf (for the Sun) and veldfearth (for the Earth)
*** and possible escape velocity corrections (Jupiter effects)
*** Author: Joakim Edsjo
*** Date: 2003-11-27
**********************************************************************
```

dssenu_ctabread.f

```
      subroutine dssenu_ctabread(wh,i,file)

**********************************************************************
*** Reads in tabulated capture rates (apart from cross section)
*** Input: wh = 'su' or 'ea' for sun or earth
***        i = table number to read
***        file = file name to read
*** Author: Joakim Edsjo
*** Date: 2003-11-27
*** Modified: 2004-02-01
**********************************************************************
```

dssenu_ctabwrite.f

```
      subroutine dssenu_ctabwrite(wh,i,file)

**********************************************************************
*** Writes out tabulated capture rates (apart from cross section)
*** Input: wh = 'su' or 'ea' for sun or earth
***        i = table number to write
***        file = file to write to
*** Author: Joakim Edsjo
*** Date: 2003-11-27
*** Modified: 2004-02-01
**********************************************************************
```

dssenu_dqagse.f

```
* ====================================================================
* nist guide to available math software.
* fullsource for module dqagse from package cmlib.
* retrieved from camsun on wed oct  8 08:26:30 1997.
* ====================================================================
      subroutine dssenu_dqagse(f,foveru,
     &   a,b,epsabs,epsrel,limit,result,abserr,neval,
     1   ier,alist,blist,rlist,elist,iord,last)
```

dssenu_dqagseb.f

```
* ====================================================================
* nist guide to available math software.
* fullsource for module dqagse from package cmlib.
* retrieved from camsun on wed oct  8 08:26:30 1997.
* ====================================================================
      subroutine dssenu_dqagseb(f,foveru,
     &   a,b,epsabs,epsrel,limit,result,abserr,neval,
     1   ier,alist,blist,rlist,elist,iord,last)
```

dssenu_dqk21.f

```
      subroutine dssenu_dqk21(f,foveru,a,b,result,abserr,resabs,resasc)
No header found.
```

dssenu_dqk21b.f

```
      subroutine dssenu_dqk21b(f,foveru,a,b,result,abserr,resabs,resasc)
No header found.
```

dssenu_foveru.f

```
**********************************************************************
*** input: velocity relative to Sun [ km s^-1 ]
*** output: f(u) / u [ cm^-3 (cm/s)^(-2) ]
*** Date: 2004-01-28
**********************************************************************

      real*8 function dssenu_foveru(u)
```

dssenu_foveruearth.f

```
**********************************************************************
*** input: velocity relative to Earth [ km s^-1 ]
*** output: f(u) / u [ cm^-3 (cm/s)^(-2) ]
*** Date: 2004-01-28
**********************************************************************

      real*8 function dssenu_foveruearth(u)
```

dssenu_hiprecint.f

```
      subroutine dssenu_hiprecint(fun,foveru,lowlim,upplim,result)
```

dssenu_hiprecint2.f

```
      subroutine dssenu_hiprecint2(fun,foveru,lowlim,upplim,result)
```

dssenu_litlf_e.f

```
       real*8 function dssenu_litlf_e(mx,vbar)
c----------------------------------------------------------------------
c
c    dssenu_litlf_e is used by capearth to calculate the capture rate
c    in the earth.
c    mx is the WIMP mass in gev
c    written by l. bergstrom 1995-12-12
c
c======================================================================
```

dssenu_litlf_s.f

```
      real*8 function dssenu_litlf_s(mx,vbar)
c----------------------------------------------------------------------
c       dssenu_litlf_s used by capsun
c       mx: WIMP mass
c       lars bergstrom 1995-12-12
c----------------------------------------------------------------------
```

dssenu_rates.f

```
      subroutine dssenu_rates(egev,theta,kind,rtype,ptype,rho,
     &  rateea,ratesu,istat)
c----------------------------------------------------------------------
c
c    This routine calculates
c       - the capture rate of WIMPs in the Sun/Earth
c       - the annihilation rate of WIMPs in the Sun/Earth
c       - the yield of neutrinos or muons from these annihilations
c       - the total flux of these particles at the detector at Earth
c
c    type : commonly used
c    desc : Rates of neutrinos, neutrino-induced leptons and hadronic
c    desc : showers from DM annihilations in the Sun/Earth
c
c    november, 1995
c    uses routines by p. gondolo and j. edsjo
c    modified by l. bergstrom and j. edsjo
```

```
c    capture rate routines are written by l. bergstrom
c    input:
c       egev  - energy in GeV
c       theta  - angular cut in degrees
c       kind    - 1 = integrated fluxes above energy egev and up to angle theta
c               - 2 = differential fluxes at egev and theta
c               - 3 = mixed, integrated up to theta, differential in energy
c       rtype   - 1 = flux of neutrinos (nu_mu and/or nu_mu-bar) km^-2 yr^-1
c               - 2 = contained mu- and/or mu+ events km^-3 yr^-1
c               - 3 = through-going mu- and/or mu+ events km^-2 yr^-1
c       ptype   - 1 = particles only (nu_mu or mu-)
c               - 2 = anti-particles only (nu_mu-bar or mu+)
c               - 3 = summed rates (both particles and anti-particles)
c       rho - local dark matter halo density [GeV/cm^3]
c    hidden input (set via dssenu_set, see that routine for details)
c       secalcmet - 1 = use jkg approximations
c                   2 = use jkg for sun, full gould for earth
c                   3 = use jkg for sun, full gould+dk for earth (deprecated)
c                   4 = use full numerical calculations for Sun, Earth,
c                       but analytical form factors (exponential)
c                   5 = use full numerical calculaitons for Sun, Earth,
c                       and numerical form factor integrations [default]
c    output: rateea  - events from earth ann. per km^2(3) per yr
c            ratesu  - events from sun ann. per km^2(3) per yr
c            the km^2 is for ptype=1,3 and km^3 for ptype=2
c    For kind=1, the units are as above
c            =2, the units have an additional GeV^-1 degree^-1
c            =3, the units have an additional GeV^-1
c    slightly modified by j. edsjo.
c    modified by j. edsjo 97-05-15 to match new inv. rate convention
c    modified by j. edsjo 97-12-03 to match muflux3.21 routines.
c    modified by p. gondolo 98-03-04 to detach dssenu_annrate from susy
c    routines.
c    modified by j. edsjo 98-09-07, corrected istat handling.
c    modified by j. edsjo 98-09-23 to use damour-krauss distributions
c      and full earth formulas.
c    modified by j. edsjo 99-03-17 to include better damour-krauss
c      velocity distributions and numerical capture rate integrations
c      for these non-gaussian distributions
c    modified by p. scott 11-04-23 to allow rates to be calculated
c      for only particles or only antiparticles
c    modified by j. edsjo 2014-12-10 to have both integrated, differential
c      and mixed fluxes in the same routine
c    modified by j. edsjo 2015-06-11 to clean up and avoid excessive use
c      of common block variables and options
c
c=======================================================================
```

dssenu_se.f

```
      real*8 function dssenu_se(x,vbar)
c---------------------------------------------------------------------
c     dssenu_se
c     x: mx/m(i)
c     vbar: three-dimensional velocity dispersion of WIMPs in the halo
c     lars bergstrom 1995-12-12
c---------------------------------------------------------------------
```

dssenu_sefull.f

```
      real*8 function dssenu_sefull(mx,m_a,v_star,v_bar,phi)
c-----------------------------------------------------------------------------
c the gould function for capture in the earth, as given by the expression
```

```
c (a10) in gould ap.j. 321 (1987) 571
c mx: WIMP mass in gev
c m_a: nuclear mass
c v_star: velocity of earth with respect to wimps
c v_bar: velocity dispersion of wimps
c output in natural units (power of gev)
c l. bergstrom 1998-09-21
c Modified by J. Edsjo, 2003-11-22
c References:
c   jkg: Jungman, Kamionkowski and Griest, Phys. Rep. 267 (1996) 195.
c    dk: Damour and Krauss, Phys. Rev. D59 (1999) 063509.
c gould: Gould, ApJ 321 (1987) 571.
c-----------------------------------------------------------------------
```

dssenu_selectelements.f

```
      subroutine dssenu_selectelements

*********************************************************************
*** After having read a solar model file, we here select which
*** elements to include in capture rate calculation. This is determined
*** by the common block variable sesunacc.
*** Author: Joakim Edsjo
*** Date: 2015-06-15 (based on dssem_sunread code)
*********************************************************************
```

dssenu_set.f

```
c...Subroutine dssenu_set
c...set parameters for neutrino telescope routines
c...  c - character string specifying choice to be made
c...author: joakim edsjo, 2000-08-16
c...modified: joakim Edsjo, 2015
c...Short description of options set here:
c...  secalcmet = 1 -> JKG approximation of capture rates
c...               2 -> JKG + similar approx for Earth, but w/ Gould
c...               4 -> Full numerical integration over velocity distribution
c...                    and sum over solar and terrestrial elements
c...               5 -> Full numerical integration over velocity distribution
c...                    and form factors (as taken from DD routines) and
c...                    sum over solar and terrestrial elements
c...              (3 -> Deprecated: Damour-Krauss population)
c...  setab = 0 -> no tables are used, instead the capture rates are
c...               calculated (integrated) directly
c...          1 -> the capture rates are read from tables (created from
c...               numerical integrations) to speed things up. This option
c...               is not available for all scenarios (e.g. full Jupiter
c...               cut-off, sejup=3, where numerical integrations need to
c...               be performed model for model).
c...  sejup = 0 -> no suppression due to Jupiter
c...          1 -> simple approximate of effects of Jupiter on solar capture
c...               if the WIMPs reach out to 5.2 AU after first scatter they
c...               are considered lost as Jupiter will most likely throw the
c...               WIMPs out of the solar system before being able to scatter
c...               in the Sun again.
c...          2 -> more complete treatment of Jupiter effects on solar capture.
c...               The maximal distance from the Sun where WIMPs are considered
c...               captured after first scatter is here calculated as a function
c...               of the scattering cross sections, based on results by
c...               Annika Peter. For very high cross sections, Jupiter effects
c...               are smaller than option 1, whereas the effects are larger
c...               for small cross sections. This option requires numerical
```

```
c...               integrations model by model.
c...          sejup options are available for secalcmet 4 and 5.
c...  sesunacc = 1 -> high accuracy for solar integration, i.e. all elements
c...                  and isotopes are included for both SI and SD scattering.
c...             2 -> medium accuracy for solar integration, i.e.
c...                  elements with very low abundances are ignored and some
c...                  heavier isotopes are clumped together element by element
c...             3 -> low accuracy for solar integration, i.e. elements with
c...                  low abundances are ignored and some
c...                  heavier isotopes are clumped together element by element
c...    In most cases, the low option is accurate enough (to within 1%).
c...  Note that the solar model used is set with a call to the routine
c...  dssem_sunset.
      subroutine dssenu_set(c)
```

dssenu_ss.f

```
       real*8 function dssenu_ss(x,vbar)
c---------------------------------------------------------------------
c       dssenu_ss used by capsun and litlf_s
c       x=mx/m(i)
c       lars bergstrom 1995-12-12
c---------------------------------------------------------------------
```

dssenu_summedyields.f

```
************************************************************************
*** function dssenu_summedyields gives the total yield of muons
*** (mu- and mu+) above threshold for
*** a given WIMP mass or the differential muon yield for a given
*** energy and a given angle.
*** kind =  1 for integrated yields (above emu and below theta)
***         2 for differential yileds (in energy and angle)
***         3 for mixed yields, diff. in energy, integrated up to theta
*** rtype = 1 for neutrino (nu_mu +/or nu_mu-bar) yields
***         2 for mu- +/or mu+ yields at neutrino-nucleon vertex (cont. events)
***         3 for mu- +/or mu+ yields at imaginary plane in detector.
***            (through-going events)
*** ptype = 1 for particles only (nu_mu or mu-)
***         2 for anti-particles only (nu_mu-bar or mu+)
***         3 for summed yields (both particles and anti-particles)
*** wh='su' corresponds to annihilation in the sun and wh='ea' corresponds
*** to annihilation in the earth. if istat=1 upon return,
*** some inaccesible parts the differential muon spectra has been wanted,
*** and the returned yield should then be treated as a lower bound.
*** if istat=2 energetically forbidden annihilation channels have been
*** wanted. if istat=3 both of these things has happened.
*** units: 1.0e-30 m**-2 (annihilation)**-1 for integrated yield.
***        1.0e-30 m**-2 gev**-1 (degree)^-1 (annihilation)**-1 for
***        differential yield.
*** An additionl unit of m**-1 for rtype 1-2.
*** author: joakim edsjo  edsjo@fysik.su.se
*** date: 96-03-19
*** modified: 96-09-03 to include new index order
*** modified: 97-12-03 to include new muyield routines (v3.21)
*** Modified: 08-04-02 to use new wimpsim routines with neutrino
*** oscillations etc.
*** Modified: 11-04-23 to allow rates to be calculated for only
***  particles or only antiparticles (pat scott, patscott@physics.mcgill.ca)
************************************************************************

      real*8 function dssenu_summedyields(emu,theta,wh,kind,rtype,
     & ptype,istat)
```

dssenu_veoutjupiter.f

```
************************************************************************
*** Function dssenu_veoutjupiter determines an approximate escape
*** velocity cut to be used instead of 0 to include effects of Jupiter
*** during capture in the Sun. The idea is that while the WIMPs are
*** being captured by the Sun, Jupiter can deflect the orbits and throw
*** the WIMPs out of the solar system, before they get captured again.
*** How efficient this is depends on the mass of the WIMP and its
*** scattering cross sections. We here use approximate results in
*** A. Peter, PR D79 (2009) 103532, arXiv:0902.1347.
*** More specifically, we interpolate in Fig. 2 to determine how far
*** out the WIMPs can safely reach after the first scatter not to be
*** disturbed by Jupiter before scattering again. We then translate this
*** to an escape velocity veout of this distance from the Sun, that is
*** the returned and used in the calculations. This is not superprecise,
*** as we do not e.g. distinguish the different regions a21 and a22 in
*** Fig. 2 (i.e. we treat long-lived orbits as captured quickly).
*** Input: mx = WIMP mass in GeV
***        sigsi = spin-independent scattering cross section (cm^2)
***        sigsd = spin-dependent scattering cross section (cm^2)
*** Output: escape velocity of orbits where to cut capture rate
***         calculation (km/s)
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: May 18, 2010
************************************************************************

      real*8 function dssenu_veoutjupiter(mx,sigsi,sigsd)
```

Chapter 31

se_yield: Yields from annihilation in the Sun/Earth

(JE: I am responsible for updating this part)

31.1 Muon yields from annihilation in the Earth/Sun – theory

We need to take into account all processes that yield muon neutrinos from annihilation in the Earth/Sun. To do this, we use a Monte Carlo, WimpSim [127], to simulate annihilations in the center of the Sun/Earth, neutrino oscillations and neutrino interactions on the way out of the Sun/Earth and to the detector.

31.1.1 Monte Carlo simulations with WimpSim

We need to evaluate the yield of different particles per neutralino annihilation. The hadronization and/or decay of the annihilation products are simulated with PYTHIA [10] 6.426 and we here describe how the simulations are done. For annihilation in the Sun/Earth the simulations are done for a set of 22 neutralino masses, m_χ = 3, 6, 10, 25, 50, 80.3, 91.2, 100, 150, 176, 200, 250, 350, 500, 750, 1000, 1500, 2000, 3000, 5000, 7500 and 10000 GeV. We tabulate the yields and then interpolate these tables in DarkSUSY.

We are mainly interested in the flux of high energy muon neutrinos and neutrino-induced muons at a neutrino telescope. We simulate 13 'fundamental' annihilation channels, for each mass (where kinematically allowed) above. In Table 31.1 we list the 'fundamental' channels for which simulations are run and the full set of more complex channels. Pions and kaons get stopped before they decay and are thus made stable in the PYTHIA simulations so that they don't produce any neutrinos. For annihilation channels containing Higgs bosons, we can calculate the yield from these fundamental channels by letting the Higgs bosons decaying in flight (see below). We also take into account the energy losses of B-mesons in the Sun and the Earth by following the approximate treatment of [110] but with updated B-meson interaction cross sections as given in [128]. We also take neutrino-interactions on the way out of the Sun into account by considering the charged-current interaction as a neutrino-loss and the neutral current interactions are simulated with nusigma [129]. The neutrino-nucleon charged current interactions close to the detector are also simulated with nusigma and finally the multiple Coulomb scattering of the muon on its way to the detector is calculated using distributions from [130]. We have used the CTEQ6 structure functions in these simulations. We also take into account neutrino oscillations with a full three-neutrino Monte Carlo. All of these

processes are put into the simulation package WimpSim [127] that can be downloaded separately. Results from simulation runs with this package are included with DarkSUSY. For more details on these simulations, see [131].

For each annihilation channel and mass we simulate 10^7 annihilations and tabulate the final results as a neutrino-yield, neutrino-to-lepton conversion rate, a muon yield and hadronic shower yields differential in energy and angle from the center of the Sun/Earth. We also tabulate the integrated yield above a given threshold and below an opening angle θ. We assumed throughout that the surrounding medium is water with a density of 1.0 g/cm^3. Hence, the neutrino-to-muon conversion rates have to be multiplied by the density of the medium. In the muon fluxes, the density cancels out (to within a few percent). All results are summarized as yield tables that can be loaded and interpolated in with DarkSUSY. This is done with the function dswayieldf. There are three kinds of yields (two-dimensional in opening angle θ and energy), kind=1 gives integrated yields, kind=2 gives differential yields and kind=3 gives yields integrated in angle, but differential in θ. For each kind, there are 26 different types of yield available according to Table 31.4. As a default, only type 3–4, 9–10 and 13–14 are included in the DarkSUSY download, as these are the most commonly used types. If you need any other types, download the auxiliary data files from `http://www.darksusy.org`, unpack them in share/DarkSUSY in the DarkSUSY root directory, and then do a usual configure and make to install them. Also note that the kind=3 yields are not tabulated directly, but are instead calculated and tabulated when the simulation tables are read in during DarkSUSY initialization.

With these simulations, we can calculate the yield for any of these particles for a given MSSM model. For the Higgs bosons, which decay in flight, an integration over the angle of the decay products with respect to the direction of the Higgs boson is performed. Given the branching ratios for different annihilation channels it is then straightforward to compute the yield above any given energy threshold and within any angular region around the Sun or the center of the Earth. The routine dswayieldone calculates the yield for one channel, i.e. even these complex channels containing Higgs bosons, whereas the main routine dswayield calculates the total yield for a given model. Note that the WIMP annihilation yield routines do not know about SUSY at all, so before they are called, a routine dswasetup is called to set up the annihilation branching ratios for the WIMP and decay channels for the Higgs bosons. In Tables 31.2 and 31.3, the Higgs decay width channels are given. If these routines are used with other particle physics models, replace dswasetup with a routine appropriate for your particle physics model and then call dswayield as usual.

31.2 Routine headers – fortran files

dsse_init.f

```
************************************************************************
*** subroutine dsse_init initializes and loads (from disk) the common
*** block variables needed by the WIMP annihilation yield routines.
*** Input: kind - 1 = integrated yields
***               2 = differential yields
***               3 = mixed yields (diff in energy, integrated in theta)
***        type - 1-26: type of yield
***        dataformat = 1 for old type (pythia 6)
***                     2 for new type (pythia 8)
*** author: joakim edsjo  edsjo@fysik.su.se
*** date: 96-10-23
*** Modified: 97-12-03, 08-04-02
************************************************************************

      subroutine dsse_init(kind,type,dataformat)
```

Annihilation channel ch	Particles	Internal channel chi	Internal channel array index chii
1	$S_1^0 S_1^0$	-	-
2	$S_1^0 S_2^0$	-	-
3	$S_2^0 S_2^0$	-	-
4	$S_3^0 S_3^0$	-	-
5	$S_1^0 S_3^0$	-	-
6	$S_2^0 S_3^0$	-	-
7	$S^- S^+$	-	-
8	$Z^0 S_1^0$	-	-
9	$Z^0 S_2^0$	-	-
10	$Z^0 S_3^0$	-	-
11	$W^- S^+ / W^+ W^-$	-	-
12	$Z^0 Z^0$	9	9
13	$W^+ W^-$	8	8
14	$\nu_e \bar{\nu}_e$	12	11
15	$e^+ e^-$	-	-
16	$\nu_\mu \bar{\nu}_\mu$	13	12
17	$\mu^+ \mu^-$	10	-
18	$\nu_\tau \bar{\nu}_\tau$	14	13
19	$\tau^+ \tau^-$	11	10
20	$u\bar{u}$	2	2
21	$d\bar{d}$	1	1
22	$c\bar{c}$	4	4
23	$s\bar{s}$	3	3
24	$t\bar{t}$	6	6
25	$b\bar{b}$	5	5
26	gg	7	7
27	qqg	-	-
28	$\gamma\gamma$	-	-
29	$Z^0\gamma$	-	-

Table 31.1: The annihilation channels ch used in dswayieldone. Also shown are the internal channel numbers chi used for the fundamental channels used in the simulations (used by routine dswayieldf). To save some additional space with the data files in memory, there are also array index channel numbers chii that are only used internally to access the right elements of the yield arrays. S denotes scalars (Higgs bosons).

Decay width channel dch	Particles
1–29	Same as the annihilation channels in Table 31.1.
30	Sfermions
31	Neutralinos
32	Charginos

Table 31.2: The neutral scalar (Higgs) decay width channels used. In DarkSUSY these are stored in the array hdwidth(i,j) where i is the decay channel index above and j is the Higgs number (1–3 for H_1^0, H_2^0 and H_3^0 respectively). For the wa routines, these decay branching ratios (partial width divided by total width) are stored in dswas0br(i,j).

Decay width channel dch	Particles
1	ud
2	$u\bar{s}$
3	$u\bar{b}$
4	$c\bar{d}$
5	$c\bar{s}$
6	$c\bar{b}$
7	$t\bar{d}$
8	$t\bar{s}$
9	$t\bar{b}$
10	$\nu_e e^+$
11	$\nu_\mu \mu^+$
12	$\nu_\tau \tau^+$
13	$W^+ S_1^0$
14	$W^+ S_2^0$
15	$W^+ S_3^0$
20	Sfermions
21	Neutralinos and charginos

Table 31.3: The (positively) charged scalar (Higgs) decay width channels used. In DarkSUSY these are stored in the array hdwidth(i,4) where i is the decay channel index above. For the wa routines, these decay branching ratios (partial width divided by total width) are stored in dswascbr(i).

dsse_init1.f

```
************************************************************************
*** subroutine dsse_init1 initializes and loads (from disk) the common
*** block variables needed by the WIMP annihilation yield routines.
*** For simulation data format version 1.
*** Input: kind - 1 = integrated yields
***               2 = differential yields
***               3 = mixed yields (diff in energy, integrated in theta)
***        type - 1-26: type of yield
*** author: joakim edsjo  edsjo@fysik.su.se
*** date: 96-10-23
*** Modified: 97-12-03, 08-04-02
************************************************************************

      subroutine dsse_init1(kind,type)
```

dsse_init2.f

```
************************************************************************
*** subroutine dsse_init2 initializes and loads (from disk) the common
*** block variables needed by the WIMP annihilation yield routines.
*** For simulation data format version 2.
*** Input: kind - 1 = integrated yields
***               2 = differential yields
***               3 = mixed yields (diff in energy, integrated in theta)
***        type - 1-26: type of yield
*** author: joakim edsjo  edsjo@fysik.su.se
*** date: 96-10-23
*** Modified: 97-12-03, 08-04-02
************************************************************************

      subroutine dsse_init2(kind,type)
```

Yield type type	Yield	Unit
1	ν_e	10^{-30} m^{-2} annihilation^{-1}
2	$\bar{\nu}_e$	10^{-30} m^{-2} annihilation^{-1}
3	ν_μ	10^{-30} m^{-2} annihilation^{-1}
4	$\bar{\nu}_\mu$	10^{-30} m^{-2} annihilation^{-1}
5	ν_τ	10^{-30} m^{-2} annihilation^{-1}
6	$\bar{\nu}_\tau$	10^{-30} m^{-2} annihilation^{-1}
7	e^- at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
8	e^+ at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
9	μ^- at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
10	μ^+ at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
11	τ^- at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
12	τ^+ at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
13	μ^- at an imaginary plane at detector	10^{-30} m^{-2} annihilation^{-1}
14	μ^+ at an imaginary plane at detector	10^{-30} m^{-2} annihilation^{-1}
15	hadronic shower from ν_e CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
16	hadronic shower from $\bar{\nu}_e$ CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
17	hadronic shower from ν_μ CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
18	hadronic shower from $\bar{\nu}_\mu$ CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
19	hadronic shower from ν_τ CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
20	hadronic shower from $\bar{\nu}_\tau$ CC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
21	hadronic shower from ν_e NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
22	hadronic shower from $\bar{\nu}_e$ NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
23	hadronic shower from ν_μ NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
24	hadronic shower from $\bar{\nu}_\mu$ NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
25	hadronic shower from ν_τ NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}
26	hadronic shower from $\bar{\nu}_\tau$ NC int. at neutrino-nucleon vertex	10^{-30} m^{-3} annihilation^{-1}

Table 31.4: The yield types available from the wa routines. All of these yields are at the detector (currently IceCube). Note that the units are for integrated yields (kind=1), for differential yields (kind=2), the units should be multiplied by GeV^{-1} degree^{-1}. CC int. = charged current interactions. NC int. = neutral current interactions.

dsseifind.f

```
*********************************************
*** routine to find the index of an entry   ***
*** the closest lowest hit is given          ***
*********************************************

      subroutine dsseifind(value,array,ipl,ii,imin,imax)
```

dsseyield_set.f

```
************************************************************************
*** subroutine dssseyield_set is used to choose which simulation
*** data tables to use.
************************************************************************

      recursive subroutine dsseyield_set(key,value)
```

dsseyield_sim.f

```
************************************************************************
*** function dsseyield_sim calculates the yield (including propagation
*** effects like oscillations and interactions) of particles from
*** WIMP annihilations in the Sun and the Earth.
*** This routine is a wrapper routine that calls dsseyield_sim1 for data
*** format version 1 and dsseyield_sim2 for data format version 2.
*** Date: March 17, 2022
*** Updates: 2022-12-08, allow for annihilation to two different
***    particles, added one more pdg argument, Joakim Edsjo
*** Author: Joakim Edsjo, edsjo@fysik.su.se
************************************************************************

      real*8 function dsseyield_sim(mwimp,e,theta,pdg1,pdg2,hel,wh,kind,
     &  type,istat)
```

dsseyield_sim1.f

```
************************************************************************
*** function dsseyield_sim calculates the yield (including propagation
*** effects like oscillations and interactions) of particles from
*** WIMP annihilations in the Sun and the Earth.
*** This routine is for simulation data format version 1.
*** This routine assumes that annihilation
*** takes place to two final state particles where the second is the
*** antiparticle of the first. For polarized states, the polarization
*** is assumed to be the same for both the particles.
***
*** Inputs:
***   - mwimp: mass of WIMP in GeV
***   - E:     kinetic energy of the neutrino, lepton or hadronic shower
***            where the yield is calculated (in GeV)
***   - theta: angle where the yield is calculated (in degrees)
***   - pdg:   PDG code of annihilation final state particle
***   Only the pdg code of the first particle is given, the second one
***   is assumed to have pdg code -pdg.
***   Only channels for which simulation data from Pythia simulations exist
***   are available here. More complex channels
***   like channels containing Higgs bosons etc that decay to standard model
***   particles are treated in each respective particle physics module
***   in src_model.
***   The currently implemented channels are
```

```
***
***      pdg  Channel           Channel No chi  Array Ch No chii
***                             (only listed temporarily)
***     ----  -------           --------------  ----------------
***        1  d d-bar            1              1
***        2  u u-bar            2              2
***        3  s s-bar            3              3
***        4  c c-bar            4              4
***        5  b b-bar            5              5
***        6  t t-bar            6              6
***       12  nu_e nu_e-bar     12              11
***       13  mu- mu+           10               - not simulated
***       14  nu_mu nu_mu-bar   13              12
***       15  tau- tau+         11              10
***       16  nu_tau nu_tau-bar 14              13
***       24  W+ W-              8              8
***       23  Z0 Z0               9              9
***       21  gluon gluon        7              7
***
***   Note: If a channel that is not simulated is asked for, the yield
***   0 is returned and a warning is issues (istat bit 3 set). Note that
***   in the internal numbering here, mu- mu+ is not included.
***   However when WimpSim is run, it is included (as number 10) to have
***   a consistent numbering with HaloAnn.
***
***   hel: helicity state of final state. Currently, only states where the
***        two final state particles are in the same helicity state (e.g.
***        both left, both right, both transverse, both longitudinal)
***        is included.
***        Note: right now, only unpolarized yields are given, will
***        change eventually
***
***   Possible values for hel:
***     'L': left-polarized fermion for fermions, longitudinal for vectors
***     'R': right-polarized fermion for fermions
***     'T': transverse for vectors
***     'O': unpolarized yields (unphysical, but included for comaprison)
***
***   - wh:    Flag determinging source body:
***            'su' for Sun and 'ea' for Earth
***   - kind:  The yields are of different kinds:
***      = 1:  integrated up to a given theta and above a given energy
***        2:  differetial in energy and angle
***        3:  differential in energy, but integrated in angle up to the
***            given angular cut
***   - type:  Type of yield:
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
```

```
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** If this routine is called outside of the kinematical regions
*** where tables exist, the following is done:
***   - for lower energies, than the lowest simulted ones,
***     extrapolations are used
***   - for masses below the lowest simulated, extrapolations
***     are used
***   - for energies above the highest simulated, the results
***     for the highest energy simulated are used
*** Output:
***   - istat (=0 if no warnings/errors are reported)
***     bit 0 is set if extrapolations below the lowest simulated mass
***       or above the highest simulated mass is needed
***     bit 3 is set if the requested channel is not simulated,
***       yield zero is returned
***     bit 4 is set if the requested polarization state is not available,
***       the yield of the simulated polarization yield (typically
***       unpolarized) is returned instead
***   - dsseyield_sim in units of
*** units: 1.0e-30 m**-2 (annihilation)**-1  integrated (kind 1)
*** units: 1.0e-30 m**-2 gev**-1 (degree)**-1 (annihilation)**-1 differential
***        (kind 2)
*** units: 1.0e-30 m**-2 gev**-1 (annihilation)**-1 mixed (kind 3)
*** types 7-12, 15-26 have an additional unit of m**-1.
*** author: joakim edsjo, edsjo@physics.berkeley.edu
*** date: 1995
*** modified: dec 03, 1997, April 3, 2008
*** Modified 2011-05-08 to included mixed yields (kind=3). Corrected a bug
*** for integrated yields below 0.2 degrees.
*** Modified: December 2014 to use PDG codes (edsjo)
***    December, 2022 to allow for two pdg codes (only one used yet
***      here though, but addition is made to be consistent with
***      format in dsseyield_sim2.
**************************************************************************

      real*8 function dsseyield_sim1(mwimp,e,theta,pdg1,pdg2,hel,wh,
     &   kind,type,istat)
*$      use omp_lib
```

dsseyield_sim2.f

```
**************************************************************************
*** function dsseyield_sim2 calculates the yield (including propagation
*** effects like oscillations and interactions) of particles from
*** WIMP annihilations in the Sun and the Earth.
*** This routine is for simulation data format version 2.
*** This routine assumes that annihilation
*** takes place to two final state particles where the second is the
*** antiparticle of the first. For polarized states, the polarization
*** is assumed to be the same for both the particles.
***
*** Inputs:
***   - mwimp: mass of WIMP in GeV
***   - E:     kinetic energy of the neutrino, lepton or hadronic shower
***            where the yield is calculated (in GeV)
***   - theta: angle where the yield is calculated (in degrees)
```

```
***   - pdg:   PDG code of annihilation final state particle
***   Only the pdg code of the first particle is given, the second one
***   is assumed to have pdg code -pdg.
***   Only channels for which simulation data from Pythia simulations exist
***   are available here. More complex channels
***   like channels containing Higgs bosons etc that decay to standard model
***   particles are treated in each respective particle physics module
***   in src_model.
***   The currently implemented channels are
***
***      pdg  Channel           Channel No chi  Array Ch No chii
***                             (only listed temporarily)
***     ----  -------           --------------  ----------------
***        1  d d-bar            1              1
***        2  u u-bar            2              2
***        3  s s-bar            3              3
***        4  c c-bar            4              4
***        5  b b-bar            5              5
***        6  t t-bar            6              6
***       12  nu_e nu_e-bar     12              11
***       13  mu- mu+           10               - not simulated
***       14  nu_mu nu_mu-bar   13              12
***       15  tau- tau+         11              10
***       16  nu_tau nu_tau-bar 14              13
***       24  W+ W-              8              8
***       23  Z0 Z0              9              9
***       21  gluon gluon        7              7
***       25  H H               15             14
***       23 25 Z0 H            16             15 - not implemented yet
***
***   Note: If a channel that is not simulated is asked for, the yield
***   0 is returned and a warning is issues (istat bit 3 set). Note that
***   in the internal numbering here, mu- mu+ is not included.
***   However when WimpSim is run, it is included (as number 10) to have
***   a consistent numbering with HaloAnn.
***
***   hel: helicity state of final state. Currently, only states where the
***        two final state particles are in the same helicity state (e.g.
***        both left, both right, both transverse, both longitudinal)
***        is included.
***        Note: right now, only unpolarized yields are given, will
***        change eventually
***
***   Possible values for hel:
***     'L': left-polarized fermion for fermions, longitudinal for vectors
***     'R': right-polarized fermion for fermions
***     'T': transverse for vectors
***     '0': unpolarized yields (unphysical, but included for comaprison)
***
***   - wh:    Flag determinging source body:
***            'su' for Sun and 'ea' for Earth
***   - kind:  The yields are of different kinds:
***      = 1:  integrated up to a given theta and above a given energy
***        2:  differetial in energy and angle
***        3:  differential in energy, but integrated in angle up to the
***            given angular cut
***   - type:  Type of yield:
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
```

```
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** If this routine is called outside of the kinematical regions
*** where tables exist, the following is done:
***   - for lower energies, than the lowest simulted ones,
***     extrapolations are used
***   - for masses below the lowest simulated, extrapolations
***     are used
***   - for energies above the highest simulated, the results
***     for the highest energy simulated are used
*** Output:
***   - istat (=0 if no warnings/errors are reported)
***     bit 0 is set if extrapolations below the lowest simulated mass
***       or above the highest simulated mass is needed
***     bit 3 is set if the requested channel is not simulated,
***       yield zero is returned
***     bit 4 is set if the requested polarization state is not available,
***       the yield of the simulated polarization yield (typically
***       unpolarized) is returned instead
***   - dsseyield_sim in units of
*** units: 1.0e-30 m**-2 (annihilation)**-1  integrated (kind 1)
*** units: 1.0e-30 m**-2 gev**-1 (degree)**-1 (annihilation)**-1 differential
***        (kind 2)
*** units: 1.0e-30 m**-2 gev**-1 (annihilation)**-1 mixed (kind 3)
*** types 7-12, 15-26 have an additional unit of m**-1.
*** author: joakim edsjo, edsjo@physics.berkeley.edu
*** date: 1995
*** modified: dec 03, 1997, April 3, 2008
*** Modified 2011-05-08 to included mixed yields (kind=3). Corrected a bug
*** for integrated yields below 0.2 degrees.
*** Modified: December 2014 to use PDG codes (edsjo)
**************************************************************************

      real*8 function dsseyield_sim2(mwimp,e,theta,pdg1,pdg2,hel,wh,
     &   kind,type,istat)
*$      use omp_lib
```

dsseyield_sim_ls.f

```
**************************************************************************
*** NOTE: This routine is not fully functional yet, but shows what we intend
*** to have eventually. The goal is to have a much more general structure
*** regarding which channels and polarization states that are avaialble.
*** This routine should eventually be able to return the yield for different
*** final state particles and polarization states, expressed as the final
*** state's quantum numbers j,P,l and s. Right now it just calls the old
```

```
*** routine, but eventually we want to add more stuff here
***
*** function dsseyield_sim_ls calculates the yield above threshold
*** (or differential at that energy and angle) for the requested
*** annihilation channel and the kind and type of yield.
*** This routine assumes that annihilation takes place to two final
*** state particles.
***
*** Inputs:
*** Inputs:
***   - mwimp: mass of WIMP in GeV
***   - E:     kinetic energy of the neutrino, lepton or hadronic shower
***            where the yield is calculated (in GeV)
***   - theta: angle where the yield is calculated (in degrees)
***
***   - pdg1, pdg2 = pdg codes of annihilation final state particles.
***   Only channels for which simulation data from Pythia simulations exist
***   are available here. More complex channels
***   like channels containing Higgs bosons etc that decay to standard model
***   particles are treated in each respective particle physics module
***   in src_model.
***   The currently implemented channels are
***
***     pdg1  pdg2  Channel          Channel No chi  Array Ch No chii
***                                  (only listed temporarily)
***     ----  ----  -------           --------------  ----------------
***        1    -1  d d-bar            1              1
***        2    -2  u u-bar            2              2
***        3    -3  s s-bar            3              3
***        4    -4  c c-bar            4              4
***        5    -5  b b-bar            5              5
***        6    -6  t t-bar            6              6
***       12   -12  nu_e nu_e-bar     12              11
***       13   -13  mu- mu+           10               - not simulated
***       14   -14  nu_mu nu_mu-bar   13              12
***       15   -15  tau- tau+         11              10
***       16   -16  nu_tau nu_tau-bar 14              13
***       24   -24  W+ W-              8              8
***       23    23  Z0 Z0              9              9
***       21    21  gluon gluon        7              7
***       25    25  H H               15             14
***       23    25  Z0 H              16             15
***
***   Note 1: the last two channels (HH and ZH) are only available
***   for Pythia 8 simulations.
***   Note 2: If a channel that is not simulated is asked for, the yield
***   0 is returned and a warning is issues (istat bit 3 set)
***
***   For the final state polarization, we need a few arguments to
***   describe it fully.
***     twoj: total angular momentum quantum number of final state particles
***         times 2.
***     p: parity quantum number (maybe switch to CP later)
***     twol: orbital angular momentum quantum number of final state times 2
***     twos: spin quantum number of final state times 2
***
***   - wh:    Flag determinging source body:
***            'su' for Sun and 'ea' for Earth
***   - kind:  The yields are of different kinds:
***      = 1:  integrated up to a given theta and above a given energy
***        2:  differetial in energy and angle
***        3:  differential in energy, but integrated in angle up to the
***            given angular cut
***   - type:  Type of yield:
***   type   Yield at detector
```

```
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** If this routine is called outside of the kinematical regions
*** where tables exist, the following is done:
***   - for lower energies, than the lowest simulted ones,
***     extrapolations are used
***   - for masses below the lowest simulated, extrapolations
***     are used
***   - for energies above the highest simulated, the results
***     for the highest energy simulated are used
*** Output:
***   - istat (=0 if no warnings/errors are reported)
***     bit 0 is set if extrapolations below the lowest simulated mass
***       or above the highest simulated mass is needed
***     bit 3 is set if the requested channel is not simulated,
***       yield zero is returned
***     bit 4 is set if the requested polarization state is not available,
***       the yield of the simulated polarization yield (typically
***       unpolarized) is returned instead
***   - dsseyield_sim in units of
*** units: 1.0e-30 m**-2 (annihilation)**-1  integrated (kind 1)
*** units: 1.0e-30 m**-2 gev**-1 (degree)**-1 (annihilation)**-1 differential
***        (kind 2)
*** units: 1.0e-30 m**-2 gev**-1 (annihilation)**-1 mixed (kind 3)
*** types 7-12, 15-26 have an additional unit of m**-1.
***
*** Note: at initialization of DarkSUSY, dsseinit should be called
*** to initialize these routines (done automatically in dsinit). This is
*** only needed once per run.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
**************************************************************************

      real*8 function dsseyield_sim_ls(mwimp,e,theta,pdg1,pdg2,twoj,p,
     &  twol,twos,wh,kind,type,istat)
```

Chapter 32

si: Self-interactions

32.1 Dark matter self-interactions (si) – theory

The CDM paradigm, resting on cold and collisionless dark matter, describes cosmological structure formation with remarkable accuracy at scales larger than about one Mpc. At smaller cosmological scales, on the other hand, this paradigm is less well tested, and currently still allows for DM self-interactions as strong as the interaction between nucleons – and thus much stronger than current limits on DM interacting with SM particles. The possibility that DM could be relatively strongly interacting with itself [132] and thereby leave imprints on cosmological observables related to structure formation has therefore seen significant interest in recent years, both from an astrophysical and a model-building perspective. Self-interacting DM (SIDM) could even address [133, 134, 135, 136, 137, 138, 139, 140] the most pressing potential small-scale problems of ΛCDM cosmology [141], referred to as 'core-vs.-cusp" [142, 143], 'too-big-to-fail' [144, 145], 'diversity' [146, 147] and 'missing satellites' problems [148, 149] (the latter problem is addressed for SIDM with late kinetic decoupling, which is a natural combination in many models [79]). Those specific discrepancies with the ΛCDM paradigm may of course well turn out to be either due to poorly modelled baryonic effects or observational uncertainties – but these examples serve as evidence that SIDM *can* leave observable imprints even in the absence of interactions with the SM. An unambiguous detection of such signatures, which are not expected in for example traditional WIMP models of DM, would significantly narrow down the range of possible particle explanations for the nature of DM.

The traditional reference quantity for the impact of DM self-interactions on the halo structure is the momentum transfer cross section,

$$\sigma_T \equiv \int d\Omega \, (1 - \cos\theta) \, \frac{d\sigma}{d\Omega} \,, \tag{32.1}$$

where σ is the standard cross section for DM-DM scattering. This has the advantage of regulating large forward-scattering amplitudes, which should not affect the DM distribution (the same goes for backward scattering which, however, is treated symmetrically in this prescription. See Ref. [150, 151] for a more detailed discussion). The size of this quantity that is of cosmological relevance is very roughly given by

$$\sigma_T / m_\chi \sim 1 \, \mathrm{cm}^2/\mathrm{g} \,. \tag{32.2}$$

Cross sections in this ballpark, in other words, may leave observable imprints and possibly address the various ΛCD; small-scale structure problems mentioned above, while much smaller cross sections have no impact on the structure of DM halos. Much larger values of σ_T/m_χ are ruled out, on

the other hand, in particular from observations of galaxy clusters. For a detailed review that summarizes both various constraints on DM self-interactions and models that have been discussed in the literature, we refer to Ref. [152].

The details of the DM self-interactions depend on the underlying particle theory. A simple contact interaction term in the Lagrangian, for example, would lead to a constant (velocity-independent) σ_T. Another well-motivated option is that the interaction is mediated by a massive particle with mass $m_{\rm med}$, which leads to a Yukawa potential between the two DM particles:

$$V(r) = \pm \frac{\alpha_\chi}{r} e^{-m_{\rm med} r} \,. \tag{32.3}$$

Here, $\alpha_\chi = g_\chi^2/(4\pi)$ describes the coupling strength between mediator and DM particles, and the different signs refer to attractive (-) and repulsive (+) potentials. Scalar mediators only generate attractive potentials, while for vector mediators this depends on the particles involved in the scattering: for (e.g. Dirac) DM scattering with anti-DM, the force is attractive, otherwise it is repulsive. An interesting phenomenological aspect for the scattering of nonrelativistic particles in such a potential is the resulting strong velocity-dependence of σ_T; this allows to achieve large scattering cross sections for the relatively small velocities of $\sim$30 km/s typically encountered in dwarf galaxies (where one observes potential discrepancies with the ΛCDM expectations) while evading the strong bounds at cluster scales for velocities of ~ 1000km/s. The transfer cross section resulting from scattering in a Yukawa potential has been extensively studied, and depends on the scattering regime ($v_{\rm rel}$ is the relative velocity between the DM particles):

- In the **Born regime** ($\alpha_\chi m_\chi \lesssim m_{\rm med}$), σ_T can be calculated perturbatively, leading (for both attractive and repulsive potentials) to [153]

$$\sigma_T^{\rm Born} = \frac{8\pi\alpha_\chi^2}{m_\chi^2 v_{\rm rel}^4} \left(\ln\left[1 + \frac{m_\chi^2 v_{\rm rel}^2}{m_{\rm med}^2}\right] - \frac{m_\chi^2 v_{\rm rel}^2}{m_{\rm med}^2 + m_\chi^2 v_{\rm rel}^2} \right) . \tag{32.4}$$

- In the **classical regime** ($m_\chi v_{\rm rel} \gtrsim m_{\rm med}$), a large numbers of partial waves contributes such that non-perturbative effects are important. In principle, this can be computed by numerically solving the Schrödinger equation for each case and then summing all contributions [150]. A computationally much more efficient method is to use parameterizations [154] of numerical results from the Plasma literature that have been obtained for screened Coulomb scattering [155, 156]. For an attractive potential, these are given by

$$\sigma_T^- = \frac{\pi}{m_{\rm med}^2} \times \begin{cases} 2\,\beta^2 \ln[1+\beta^{-2}] & \text{for } \beta \lesssim 10^{-2} \\ \frac{7\beta^{1.8} + 1960(\beta/10)^{10.3}}{1 + 1.4\beta + 0.006\beta^4 + 160(\beta/10)^{10}} & \text{for } 10^{-2} \lesssim \beta \lesssim 10^2 \\ 0.81\,(1 + \ln\beta - (2\ln\beta)^{-1})^2 & \text{for } \beta \gtrsim 10^2 \end{cases} \tag{32.5}$$

  where $\beta \equiv 2\alpha_\chi m_{\rm med}/(m_\chi v_{\rm rel}^2)$, while for a repulsive potential we have

$$\sigma_T^+ = \frac{\pi}{m_{\rm med}^2} \times \begin{cases} 2\beta^2 \ln[1+\beta^{-2}] & \text{for } \beta \lesssim 10^{-2} \\ \frac{8\beta^{1.8}}{1 + 5\beta^{0.9} + 0.85\beta^{1.6}} & \text{for } 10^{-2} \lesssim \beta \lesssim 10^4 \\ (\ln 2\beta - \ln\ln 2\beta)^2 & \text{for } \beta \gtrsim 10^4 \end{cases} \tag{32.6}$$

- Finally, there is the **resonant regime** (for $m_\chi v_{\rm rel} \lesssim m_{\rm med}$ or $\alpha_\chi m_\chi \gtrsim m_{\rm med}$). Full numerical solutions can be obtained by solving the Schrödinger equation explicitly in this regime. These are well described by the following analytic expressions that result from approximating the Yukawa potential with a Hulthén potential [150]

$$\sigma_T^{\rm Hulthn} = \frac{16\pi}{m_\chi v_{\rm rel}^2} \sin^2\left({\rm Arg}\left[\frac{i\Gamma[i\Theta v_{\rm rel}]}{\Gamma[\lambda_+]\,\Gamma[\lambda_-]} \right] \right) , \tag{32.7}$$

where $\Theta \equiv \frac{m_\chi}{\sqrt{2\zeta(3)}m_{\rm med}}$ and $\lambda_\pm \equiv 1 + i\Theta v_{\rm rel}/2 \pm \sqrt{\alpha_\chi\Theta - \Theta^2 v_{\rm rel}^2/4}$ for an attractive potential and $\lambda_\pm \equiv 1 + i\Theta v_{\rm rel}/2 \pm i\sqrt{\alpha_\chi\Theta + \Theta^2 v_{\rm rel}^2/4}$ for a repulsive potential.

32.2 Self-interactions – routines

In DarkSUSY, $\sigma_T(v_{\rm rel})/m_\chi$ is provided by an interface function dssigtm returned by the particle module which, from the perspective of the core library, can take any functional form. The function dssigtmav then computes the velocity average $\langle\sigma_T\rangle/m_\chi$, assuming a Maxwellian velocity distribution of the DM particles; this gives a better estimate of the effect of DM self-interactions than just evaluating dssigtm directly for a typical halo velocity. The core library furthermore provides several auxiliary routines, to be used by any particle module, for the commonly encountered specific transfer cross sections in the presence of a Yukawa potential as discussed above. In particular, dssisigmatborn returns the expression given in Eq. (32.4), dssisigmatclassical those given in Eqs. (32.5, 32.6), and dssisigmatres those given in Eq. (32.7). For the latter, we adopt the complex gamma function as implemented in the CERN library (based on Ref. [157]), and use analytic expansions where a naive usage of these routines is problematic due to limited numerical precision (which is relevant for a significant fraction of the physically interesting parameter range).

32.3 Routine headers – fortran files

dssisigmatborn.f

```
************************************************************************
*** Function dssisigmatborn provides the momentum-transfer cross section    ***
*** for a Yukawa potential in the Born regime.                               ***
***                                                                         ***
***  Input:                                                                 ***
***    mmed - mediator mass (in GeV)                                        ***
***    beta - ratio of potential to kinetic energy,                         ***
***           beta = 2*alpha*mmed/mdm/vrel**2                               ***
***    R    - ratio of interaction range to de Broglie wavelength           ***
***           R = vrel*mDM/mmed                                             ***
***                                                                         ***
***  Output: momentum-transfer cross section in units of GeV**-2.           ***
***                                                                         ***
***  This implements Eq. (5) in 0911.0422, and provides the correct         ***
***  description of \sigma_T for \alpha mdm/mmed << 1.                      ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2015-05-17                                                         ***
************************************************************************
      real*8 function dssisigmatborn(mmed,beta,R)
```

dssisigmatclassical.f

```
************************************************************************
*** Function dssisigmatclassical provides the momentum-transfer cross       ***
*** section for a Yukawa potential in the classical regime.                 ***
***                                                                         ***
***  Input:                                                                 ***
***    mmed - mediator mass (in GeV)                                        ***
***    beta - ratio of potential to kinetic energy,                         ***
***           beta = 2*alpha*mmed/mdm/vrel**2                               ***
***    rep  - specifies whether the potential is attractive (rep=0)         ***
***           or repulsive (rep=1)                                          ***
***                                                                         ***
***  Output: momentum-transfer cross section in units of GeV**-2.           ***
```

```
***                                                                          ***
***  This implements Eqs (45,46) in 1512.05344, based on numerical results  ***
***  by Khrapak et al. (2003,2004), and provides the correct description of ***
***  \sigma_T for vrel mdm/mmed >> 1.                                        ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                     ***
*** date 2015-05-17                                                          ***
********************************************************************************
      real*8 function dssisigmatclassical(mmed,beta,rep)
```

dssisigmatres.f

```
********************************************************************************
*** Function dssisigmatres provides the momentum-transfer cross section     ***
*** for a Hulthén potential in the resonant regime.                          ***
***                                                                          ***
***  Input:                                                                  ***
***    mdm   - dark matter mass (in GeV)                                     ***
***    mmed  - mediator mass (in GeV)                                        ***
***    vrel  - relative velocity of DM particles (in units of c)             ***
***    alpha - DM-mediator coupling (alpha=g**2/4/pi)                        ***
***            beta = 2*alpha*mmed/mdm/vrel**2                               ***
***    rep   - specifies whether the potential is attractive (rep=0)         ***
***            or repulsive (rep=1)                                          ***
***                                                                          ***
***  Output: momentum-transfer cross section in units of GeV**-2.            ***
***                                                                          ***
***  This implements Eq. (A4,A5) in 1302.3898, and provides the correct      ***
***  description of \sigma_T for vrel mdm/mmed < 1, i.e. outside the.        ***
***  classical regime.                                                       ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                     ***
*** date 2015-05-17                                                          ***
********************************************************************************
      real*8 function dssisigmatres(mdm,mmed,vrel,alpha,rep)
```

dssisigtmav.f

```
********************************************************************************
*** Function dssisigtmav provides the velocity-averaged momentum-transfer    ***
*** cross section per DM mass, assuming a Maxwellian velocity distribution   ***
*** of the DM particles.                                                     ***
***                                                                          ***
***  type : commonly used                                                    ***
***  desc : velocity-averaged momentum-transfer cross section                ***
***  desc : for dark matter self-interactions                                ***
***                                                                          ***
***  Input:                                                                  ***
***    v0 - most probable velocity of individual DM particles               ***
***         (i.e. the mean speed is given by 2/sqrt(pi)*v0)                  ***
***         units: km/s                                                      ***
***                                                                          ***
***  Output:                                                                 ***
```

$$\langle\sigma_T\rangle = \frac{4\pi}{(\sqrt{2\pi}v_0)^3}\int_0^\infty dv\, v^2 \exp -v^2/(2v_0^2)\sigma_T(v)\,,$$

where v is the *relative* (i.e. not individual) DM velocity.

```
***                                                                          ***
***  This function requires the particle module to provide the interface     ***
***  function dssisigtm.                                                     ***
***                                                                          ***
```

```
***  Units of output: cm**2 / g.                                          ***
***                                                                       ***
*** author: Torsten.Bringmann.fys.uio.no                                  ***
*** date 2015-05-17                                                       ***
*****************************************************************************
      real*8 function dssisigtmav(v0)
```

Chapter 33

ucmh: Ultra-compact mini-halos

33.1 Routine headers – fortran files

dsageatz.f

```
!Calculates the approximate age of the Universe at a given redshift.
!Assumes a flat Universe; accurate to a few percent for z ~< 1000.  Matches Eq 16, astro-ph/0003463.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: April 2015
!
!Input:   z          redshift
!Output:  dsageatz   age of the Universe at redshift z

      double precision function dsageatz(z)
```

dshmucmh_set.f

```
      subroutine dshmucmh_set(mucmh, rucmh, rcore, dist)
*****************************************************************
*** subroutine dshmucmh_set:                                  ***
*** Perform some initialisation required when the UCMH mass, ***
*** radii or distance have been set/reset.                    ***
***                                                           ***
*** author: pat scott (patscott@physics.mcgill.ca)           ***
*** date: 2010-10-17                                         ***
*****************************************************************
```

dshmucmhrho.f

```
*****************************************************************
*** Ultracompact primordial minihalo density profile,        ***
*** without adiabatic contraction.                           ***
***                                                          ***
*** radialdist = distance from centre of minihalo in kpc     ***
***                                                          ***
*** Input: radial distance in kpc                            ***
***                                                          ***
*** Output: density in gev/cm**3                             ***
***                                                          ***
*** Author: Pat Scott (pat@fysik.su.se)                      ***
```

```
*** Date: 2009-08-08                                          ***
****************************************************************

      real*8 function dshmucmhrho(radialdist)
```

dshmucmhsetprofile.f

```
**********************************************************************
*** dshmucmhsetprofile: store profile name for UCMH
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Adapted from Pat Scott's implementation with dshm_set
**********************************************************************
      subroutine dshmucmhsetprofile(c)
```

dspbh_init.f

```
!Initialisation routine for interpolator in limits on PBH fractions
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: Oct 2014

      subroutine dspbh_init
```

dspbh_limits.f

```
!Returns limit on beta in PBHs from arXiv:0912.5297
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: Oct 2014
!
!Input: k      wavenumber of perturbation (Mpc^-1)
!Output:    betamax   maximum allowed value of beta

      double precision function dspbh_limits(k)
```

dsucmh_alphaintegrand.f

```
!Integrand for calculating alpha^2 for power-law cosmological power spectra (Eq B9 in arXiv:1110.2484)
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: 2011

      double precision function alphaIntegrand(x,n,alpha,kratio)
```

dsucmh_aontsq.f

```
!Compute the square of the ratio of the scalefactor to the age of the Universe
!at a given redshift.
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: 2011
!
!Input:   z       redshift    (dimensionless)
!Output:  aontsq  a^2/(ct)^2  (Mpc^-2)
```

```
      double precision function dsucmh_aontsq(z)
```

dsucmh_beta_generalised.f

```
!Works out UCMH/PBH fraction at equality for generalised power spectra.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  log10P_R   log10(power in curvature perturbations) (dimensionless)
!        delta_min  lower limit of density contrasts that could form UCMHs (dimensionless)
!        delta_max  upper limit of density contrasts that could form UCMHs (dimensionless)
!
!Output: beta       fraction of perturbations leading to UCMHs/PBHs (dimensionless)

      double precision function dsucmh_beta_generalised(log10P_R,delta_min,delta_max)
```

dsucmh_beta_powerlaw_running.f

```
!Works out UCMH/PBH fraction at equality for running power-law power spectra.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  n          spectral index (dimensionless)
!        alpha      running of spectral index (dimensionless)
!        k          wavenumber corresponding to scale of perturbations (Mpc^-1)
!        delta_min  lower limit of density contrasts that could form UCMHs (dimensionless)
!        delta_max  upper limit of density contrasts that could form UCMHs (dimensionless)
!        mH         horizon mass (M_solar)
!        rough      use the rough Green-Liddle formula
!
!Output: beta       fraction of perturbations leading to UCMHs/PBHs (dimensionless)

      double precision function dsucmh_beta_powerlaw_running(n,alpha,k,delta_min,delta_max,mH,rough)
```

dsucmh_beta_powerlaw_step.f

```
!Works out UCMH/PBH fraction at equality for running power-law power spectra with a step.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  n          spectral index (dimensionless)
!        alpha      running of spectral index (dimensionless)
!        p          size of step in amplitude of power spectrum (dimensionless)
!        k_s        wavenumber at which the step occurs (Mpc^-1)
!        k          wavenumber corresponding to scale of perturbations (Mpc^-1)
!        delta_min  lower limit of density contrasts that could form UCMHs (dimensionless)
!        delta_max  upper limit of density contrasts that could form UCMHs (dimensionless)
!
!Output: beta       fraction of perturbations leading to UCMHs/PBHs (dimensionless)

      double precision function dsucmh_beta_powerlaw_step(n,alpha,p,k_s,k,delta_min,delta_max)
```

dsucmh_betaf16.f

```
!N=16 beta calculator for feeder scaling.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
!Input:  M     M_3 non-Gaussianty parameter (dimensionless)
!        nu    ratio delta/sigma of density contrast to mass variance (dimensionless)
!Output: beta  (dimensionless)

      double precision function dsucmh_BetaF16(M,v)
```

dsucmh_betaf17.f

```
!N=17 beta calculator for feeder scaling.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
!Input:  M     M_3 non-Gaussianty parameter (dimensionless)
!        nu    ratio delta/sigma of density contrast to mass variance (dimensionless)
!Output: beta  (dimensionless)

      double precision function dsucmh_BetaF17(M,v)
```

dsucmh_betah15.f

```
!N=15 beta calculator for hierarchical scaling.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
!Input:  M     M_3 non-Gaussianty parameter (dimensionless)
!        nu    ratio delta/sigma of density contrast to mass variance (dimensionless)
!Output: beta  (dimensionless)

      double precision function dsucmh_BetaH15(M,v)
```

dsucmh_betah16.f

```
!N=16 beta calculator for hierarchical scaling.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
!Input:  M     M_3 non-Gaussianty parameter (dimensionless)
!        nu    ratio delta/sigma of density contrast to mass variance (dimensionless)
!Output: beta  (dimensionless)

      double precision function dsucmh_BetaH16(M,v)
```

dsucmh_betaintegrand.f

```
!Integrand for dsucmh_beta
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  delta density contrast (dimensionless)
!Output: integrand  (dimensionless)

      double precision function betaintegrand(delta)
```

dsucmh_deltahsq.f

```
!Power spectrum normalisation delta_H^2
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  n          spectral index (dimensionless)
!        alpha      running of spectral index (dimensionless)
!        k          wavenumber corresponding to scale of perturbations (Mpc^-1)
!
!Output: beta       power spectrum normalisation (dimensionless)
!        kratio     ratio of k to reference scale (dimensionless)

      double precision function dsucmh_deltahsq(n,alpha,k,kratio)
```

dsucmh_dmin.f

```
!Works out the mimimum amplitude perturbation required to form a UCMH
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  k          wavenumber corresponding to scale of perturbations (Mpc^-1)
!        z          latest allowed redshift of collapse of UCMHs
!
!Output: dmin       minimum density contrast required to form a UCMH (dimensionless)

      double precision function dsucmh_dmin(k,z)
```

dsucmh_f_cs.f

```
!Function for calculating the theoretical differential fraction of DM in UCMHs from cosmic string loops
!
!Authors: Madeleine Anthonisen
!          maddyanthonisen@hotmail.com
!         Pat Scott
!          pscott@imperial.ac.uk
!Date: 2014/15
!
!Inputs:  r_cs               cosmic string loop radius (kpc)
!         Gmu                cosmic string tension (dimensionless)
!         z_c                latest allowed redshift of UCMH collapse (dimensionless)
```

```
!         zstop             redshift at which UCMHs stop growing (dimensionless)
!         alpha             ratio of loop length to horizon scale at formation (dimensionless)
!         beta              ratio of loop length to loop radius (dimensionless)
!         gamma             cosmic string loop decay constant (dimensionless)
!         N                 number of loops formed per Hubble 4-volume (dimensionless)
!         K                 number of loop radii loops can travel before UCMH formation is impossible (dimensionless)
!         sigma             stdev of loop velocity distribution / speed of light (dimensionless)
!
!Output:  dsucmh_f_cs       differential fraction of DM in UCMHs (M_Sun^-1)

      double precision function dsucmh_f_cs(r_cs, Gmu, z_c, zstop, alpha, beta, gamma, N, K, sigma)
```

dsucmh_f_generalised.f

```
!Calculates the fraction of mass in UCMHs today from generalised curvature perturbations
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: mostly 2011, in this form Oct 2014
!
!Input:   log10P_R        log_10 of the amplitude of primordial curvature perturbations at scale k
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs
!         zstop           redshift at which UCMHs cease to grow
!
!Output:  f (return val)  cosmological fraction of UCMHs today
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)
!         log10P_delta    log_10 of the amplitude of primordial density perturbations at scale k

      double precision function dsucmh_f_generalised(log10P_R, mucmh, z_i, z_c, zstop, k, log10P_delta)
```

dsucmh_f_ng_feeder.f

```
!Theoretical prediction of cosmological fraction of UCMHs from feeder non-Gaussianity.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
!Input:   log10P_R        log_10 of the Gaussian amplitude of primordial curvature perturbations at scale k (dimension
!         M_3             Dimensionless skewness of the perturbation amplitude probability distribution (ie. degree of
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs (dimensionless)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!
!Output:  f (return val)  cosmological fraction of UCMHs today (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)
!         log10P_delta    log_10 of the amplitude of primordial density perturbations at scale k (dimensionless)

      double precision function dsucmh_f_ng_feeder(log10P_R, M_3, mucmh, z_i, z_c, zstop, k, log10P_delta)
```

dsucmh_f_ng_hierarchical.f

```
!Theoretical prediction of cosmological fraction of UCMHs from hierarchical non-Gaussianity.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2012
!
```

```
!Input:   log10P_R        log_10 of the Gaussian amplitude of primordial curvature perturbations at scale k (dimension
!         M_3             Dimensionless skewness of the perturbation amplitude probability distribution (ie. degree of
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs (dimensionless)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!
!Output:  f (return val)  cosmological fraction of UCMHs today (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)
!         log10P_delta    log_10 of the amplitude of primordial density perturbations at scale k (dimensionless)

      double precision function dsucmh_f_ng_hierarchical(log10P_R, M_3, mucmh, z_i, z_c, zstop, k, log10P_delta)
```

dsucmh_f_powerlaw.f

```
!Theoretical prediction of cosmological fraction of UCMHs from power-law Gaussian power spectra.
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: 2011
!
!Input:   n               spectral index of primordial spectrum of perturbations (dimensionless)
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs (dimensionless)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!         rough           Rely on Green-Liddle approximation for mass variance (only to be used for back-comparison)
!
!Output:  f (return val)  cosmological fraction of UCMHs today (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)

      double precision function dsucmh_f_powerlaw(n,mucmh,z_i,z_c,zstop,rough,k)
```

dsucmh_f_powerlaw_running.f

```
!Theoretical prediction of cosmological fraction of UCMHs from running power-law Gaussian power spectra.
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: 2011
!
!Input:   n               spectral index of primordial spectrum of perturbations (dimensionless)
!         alpha           slope of running spectral index (dimensionless)
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs (dimensionless)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!
!Output:  f (return val)  cosmological fraction of UCMHs today (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)

      double precision function dsucmh_f_powerlaw_running(n,alpha,mucmh,z_i,z_c,zstop,k)
```

dsucmh_f_powerlaw_step.f

```
!Theoretical prediction of cosmological fraction of UCMHs from running power-law Gaussian power spectra with a step.
!
!Author:  Pat Scott
!         pscott@imperial.ac.uk
!Date: 2011
!
!Input:   n               spectral index of primordial spectrum of perturbations (dimensionless)
```

```
!         alpha           slope of running spectral index (dimensionless)
!         p               step size in primordial spectrum of density perturbations (dimensionless)
!         k_s             wavenumber at which spectrum has a step discontinuity (Mpc^-1)
!         mucmh           present-day UCMH mass (M_Sun)
!         z_c             latest allowed redshift of collapse in order to form UCMHs (dimensionless)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!
!Output:  f (return val)  cosmological fraction of UCMHs today (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)

      double precision function dsucmh_f_powerlaw_step(n,alpha,p,k_s,mucmh,z_i,z_c,zstop,k)
```

dsucmh_flux.f

```
!Calculates the gamma-ray flux expected from a single UCMH
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: mostly late 2009, in this form Oct 2014
!
!Input:   mucmh            UCMH mass (M_Sun)
!         rucmh            UCMH radius (kpc)
!         rcore            Core radius of the UCMH (kpc)
!         dist             distance at which UCMH is situated, for flux calculation (kpc)
!         ethreshold       Instrumental energy threshold (GeV)
!
!Output:  flux             flux from WIMP annihilation in a single UCMH (cm^-2 s^-1)
!         Eflux            'Energy flux' from WIMP annihilation in a single UCMH (GeV cm^-2 s^-1)

      subroutine dsucmh_flux(mucmh, rucmh, rcore, dist, ethreshold, flux, Eflux)
```

dsucmh_fmax_gamma_diffuse.f

```
!Limit on cosmological fraction of UCMHs from high-lattitude diffuse gamma-ray background
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   Eflux           energy-flux from WIMP annihilation in a single UCMH at distance dist (GeV cm^-2 s^-1)
!         dist            reference distance at which UCMH is situated (kpc)
!         skyfrac         Fraction of the sky accessible to the probe in question (dimensionless)
!         CL_fmax         The confidence level at which fmax should be computed (dimensionless)
!         quiet           Suppress warnings
!
!Output:  fmax            maximum cosmological fraction of UCMHs today (dimensionless)

      double precision function dsucmh_fmax_gamma_diffuse(Eflux, dist, skyfrac, CL_fmax, quiet)
```

dsucmh_fmax_gamma_exgalptsrc.f

```
!Limit on cosmological fraction of UCMHs from extragalactic point sources.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   flux                flux from WIMP annihilation in a single UCMH at distance dist (must have same units as
!         dist                reference distance at which UCMH is situated (kpc)
```

```
!         skyfrac              Fraction of the sky accessible to the probe in question (dimensionless)
!         gamma_pt_sensitivity Point source sensitivity of gamma-ray instrument (must have same units as flux)
!         CL_ptsrc             The confidence level with which gamma_pt_sensitivity is stated (dimensionless)
!         CL_fmax              The confidence level at which fmax should be computed (dimensionless)
!         quiet                Suppress warnings
!
!Output:  fmax                 maximum cosmological fraction of UCMHs today (dimensionless)

      double precision function dsucmh_fmax_gamma_exgalptsrc(flux, dist, skyfrac, gamma_pt_sensitivity, CL_ptsrc, CL_f
```

dsucmh_fmax_gamma_galptsrc.f

```
!Limit on cosmological fraction of UCMHs from galactic point sources.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   flux                 flux from WIMP annihilation in a single UCMH at distance dist (must have same units as
!         dist                 reference distance at which UCMH is situated (kpc)
!         skyfrac              Fraction of the sky accessible to the probe in question (dimensionless)
!         gamma_pt_sensitivity Point source sensitivity of gamma-ray instrument (must have same units as flux)
!         CL_ptsrc             The confidence level with which gamma_pt_sensitivity is stated (dimensionless)
!         CL_fmax              The confidence level at which fmax should be computed (dimensionless)
!         quiet                Suppress warnings
!
!Output:  fmax                 maximum cosmological fraction of UCMHs today (dimensionless)

      double precision function dsucmh_fmax_gamma_galptsrc(flux, dist, skyfrac, gamma_pt_sensitivity, CL_ptsrc, CL_fma
```

dsucmh_fmax_reion.f

```
!Limit on cosmological fraction of UCMHs from reionisation.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   mchi            WIMP mass (GeV)
!         zstop           redshift at which UCMHs cease to grow (dimensionless)
!         quiet           Suppress warnings
!
!Output:  fmax            maximum cosmological fraction of UCMHs today (dimensionless)

      double precision function dsucmh_fmax_reion(mchi, z_i, zstop, quiet)
```

dsucmh_init_profile.f

```
!Set up the density profile for UCMHs, returning core and outer radii.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: Oct 2014
!
!Input:   profile         chosen density profile for UCMHs
!                         General:
!                          "plain": uncontracted, analytical profile with mass as per mucmh
!                         For UCMHs from phase transition <trans> in {"EW", "QCD", "ee"}:
!                          <trans>_01: F=0.001, core radius 0.1%
```

```
!                          <trans>_11: F=0.01, core radius 0.1%
!                          <trans>_unc: uncontracted, numerical
!         z_i              redshift of formation of the the UCMH seed
!         z_c              latest allowed redshift of collapse in order to form UCMHs
!         zstop            redshift at which UCMHs cease to grow
!         mchi             WIMP mass (GeV)
!         sigchi           WIMP ann. cross-section (cm^3 s^-1)
!
!In/Out:  mucmh            UCMH mass (M_Sun); input if profile = "plain", output otherwise (as UCMHs are implied to hav
!                          originated in some specific phase transition in this case).
!
!Output:  rucmh            UCMH radius (kpc)
!         rmin             radius below which the radial infall assumption is violated (kpc)
!         rcut             radius at which WIMP self-annihilation cuts off the density profile (kpc)

      subroutine dsucmh_init_profile(profile, z_i, z_c, zstop, mchi, sigchi, mucmh, rucmh, rmin, rcut)
```

dsucmh_initdens_ptrans.f

```
*****************************************************************
*** Numerical density profile initialisation for UCMHs formed***
*** in phase transitions in the early Universe.              ***
***                                                          ***
*** Author: Pat Scott (pat@fysik.su.se)                      ***
*** Date: August 2009                                        ***
*****************************************************************

      subroutine dsucmh_initdens_ptrans
```

dsucmh_jpntsrc.f

```
!Calculate the J value for point source detection of a single UCMH at distance r_0
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Inputs: incut  inner radius of UCMH (kpc)
!        inmax  outer radius of UCMH (kpc)
!Output: J factor (cm^-5 GeV^2)

      double precision function dsucmh_jpntsrc(incut,inmax)
```

dsucmh_localmass.f

```
c.. Program som beraknar massan i vintergatan innanfor en sfar med
c.. radien dmin kpc runt solen (med antaganden att profilen ar NFW
c.. fran Battaglia et al, MNRAS 370:1055 2006 -- se dslocal.h).
c..
c.. (Mass of DM in sphere around Sun assuming NFW profile as per
c.. Battaglia+)
c..
c.. Input: dmin (kpc)                 radius of sphere
c.. Output: dsucmh_localmass (M_sun)  DM mass within that sphere
c..
c.. Author: Sofia Sivertsson August/September 2009
c.. Modified: Pat Scott (pat@fysik.su.se) 100529, 110127, 141011

      real*8 function dsucmh_localmass(dmin)
```

dsucmh_mass_cs.f

```
!Subroutine for calculating UCMH mass and regime of collapse (0-6) for a given cosmic string radius and value of Gmu
!
!Authors: Madeleine Anthonisen
!          maddyanthonisen@hotmail.com
!         Pat Scott
!          pscott@imperial.ac.uk
!Date: 2014/15
!
!All equation numbers refer to v1 of the arXiv posting:
!  Anthonisen, Brandenberger & Scott
!  'Constraints on cosmic strings from ultracompact minihalos', arXiv:1504.something
!
!Inputs:  r_cs    cosmic string radius (kpc)
!         Gmu     cosmic string tension (dimensionless)
!         z_c     latest allowed redshift of UCMH collapse (dimensionless)
!         zstop   redshift at which UCMHs stop growing (dimensionless)
!         alpha   ratio of loop length to horizon scale at formation (dimensionless)
!         beta    ratio of loop length to loop radius (dimensionless)
!         gamma   cosmic string loop decay constant (dimensionless)
!
!Outputs: mucmh   mass of UCMH (M_solar)
!         ronteq  r_cs / (c*teq) (dimensionless)
!         regime  0-VI: regimes of UCMH collapse (0=No UCMHs)
!         gmu_c1  minimal G\mu for regime I, Scenario A, Eq28
!         gmu_c2  minimal G\mu for regime I, Scenario B, Eq29
!         gmu_c3  G\mu separating regimes III & "No UCMHs", Scenario A, Eq49
!         gmu_c4  G\mu separating regimes V & IV, Scenario B, Eq59(x_f=x_decay)
!         gmu_c5  G\mu separating regimes IV & "No UCMHs", Scenario C, Eq59(x_f=x_c)
!         gmu_c6  G\mu separating regimes VI & VII, Scenario D, Eq63 (x_f=x_decay)
!         gmu_c7  G\mu separating regimes VI & "No UCMHs", Scenario E, Eq63 (x_f=x_c)
!         gmu_c8  G\mu separating regimes VII & "No UCMHs", Scenario D, Eq72
!         gmu_c9  G\mu separating regimes II & III, Scenario A, Eq32

      subroutine dsucmh_mass_cs(r_cs, Gmu, z_c, zstop, alpha, beta, gamma,
     &                          mucmh, ronteq, regime, gmu_c1,gmu_c2,gmu_c3,gmu_c4,gmu_c5,gmu_c6,gmu_c7,gmu_c8,gmu_c9)
```

dsucmh_mass_ptrans.f

```
!Returns the expected mass of a UCMH produced in a phase transition.
!Note that the result is DM+baryons, but the internal deltam is just DM as the baryons do not collapse with the DM
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Inputs: transition  EW, QCD or ee phase transition
!        zstop       redshift at which UCMHs cease to grow (dimensionless)
!Output: Mass of UCMH (M_sun)
      double precision function dsucmh_mass_ptrans(transition, zstop)
```

dsucmh_mathcalT.f

```
!Returns fitting function \mathcal{T} from Weinberg's cosmology book
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   k             wavenumber (Mpc^-1)
!Output   \mathcal{T}   fitting func (dimensionless)
```

```
      double precision function dsucmh_mathcalT(k)
```

dsucmh_midinf.f

```
!Midpoint integration to infinity for power-law power spectra
!Adapted from Numerical Recipes.
!Author: Pat Scott (p.scott@imperial.ac.uk)
!Date: Some time in 2010/11.
!Added to DarkSUSY: Oct 13 2014
      subroutine midinf(funk,n_spec,alpha,k,aa,bb,s,n)
```

dsucmh_midpnt.f

```
!Midpoint integration for power-law power spectra
!Adapted from Numerical Recipes.
!Author: Pat Scott (p.scott@imperial.ac.uk)
!Date: Some time in 2010/11.
!Added to DarkSUSY: Oct 13 2014

      subroutine midpnt(func,n_spec,alpha,k,a,b,s,n)
```

dsucmh_mvar_generalised.f

```
!Mass variance calculation for generalised power spectra.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input: log10(power in curvature perturbations) (dimensionless)
!Output: mass variance (dimensionless)

      double precision function dsucmh_mvar_generalised(log10P_R)
```

dsucmh_mvar_powerlaw_running.f

```
!Mass variance calculation for running power-law power spectra.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:   n               spectral index of primordial spectrum of perturbations (dimensionless)
!         alpha           slope of running spectral index (dimensionless)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)
!
!Output: mass variance (dimensionless)

      double precision function dsucmh_mvar_powerlaw_running(n,alpha,k)
```

dsucmh_mvar_powerlaw_step.f

```
!Mass variance calculation for running power-law power spectra with a step.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
```

```
!
!Input:   n               spectral index of primordial spectrum of perturbations (dimensionless)
!         alpha           slope of running spectral index (dimensionless)
!         logp2           ln(square of step size in primordial spectrum of density perturbations) (dimensionless)
!         k_s             wavenumber at which spectrum has a step discontinuity (Mpc^-1)
!         k               wavenumber of perturbation (Mpc^-1; 0.5d3/lambda)
!
!Output: mass variance (dimensionless)

      double precision function dsucmh_mvar_powerlaw_step(n,alpha,logp2,k_s,k)
```

dsucmh_numdens_ptrans.f

```
!Return the dark matter density a given height above the centre of a UCMH created in a phase transition
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  r  height above centre of UCMH (kpc)
!Output: dark matter density (GeV cm^-3)

      double precision function dsucmh_numdens_ptrans(r)
```

dsucmh_numerical_dens.f

```
!Return the dark matter density a given height above the centre of a UCMH.
!Replace this with a call to your own density function to
!include other profiles, like adiabatically-contracted ones.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2009
!
!Input:  r  height above centre of UCMH (kpc)
!Output: dark matter density (GeV cm^-3)

        double precision function dsucmh_numerical_dens(r)
```

dsucmh_qromo.f

```
!Open-interval Romberg integration for power-law power spectra
!Adapted from Numerical Recipes.
!Author: Pat Scott (p.scott@imperial.ac.uk)
!Date: Some time in 2010/11.
!Added to DarkSUSY: Oct 13 2014


      double precision function qromo(func,n,alpha,kratio,a,b,choose)
```

dsucmh_transfunc.f

```
!Dark matter transfer function.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
```

```
!Input:    x    wavenumber * horizon scale (dimensionless)
!Output:   T    transfer function (dimensionless)

      double precision function dsucmh_transFunc(x)
```

dsucmh_transfunc_rad.f

```
!Radiation transfer function.
!
!Author:  Pat Scott
!          pscott@imperial.ac.uk
!Date: 2011
!
!Input:    x    wavenumber * horizon scale (dimensionless)
!Output:   T    transfer function (dimensionless)

      double precision function dsucmh_transFunc_rad(x)
```

Part III

Particle physics modules in src_models

Chapter 34

Basic principles and common routines

The general concept of a particle physics module, and how it communicates with the core library via interface functions, was already introduced in Chapter 3 – see in particular Section 3.2, and Table 3.3 for an automatically updated list of all interface functions that the core library is aware of. Note that there is no *principal* restriction on which interface functions a particle module must provide: the main program will determine at compilation time whether it needs a functionality of the core library that requires certain interface functions to exist.

Every particle physics module must provide an independent representation of the full particle content of the respective particle theory. How this is done is fully flexible, and completely up to the module. In practice, however, there are common frameworks, like the Standard Model, that appear repeatedly. For convenience, we therefore store auxiliary routines and setups that *may* be used by more than just one particle module in src_models/common. Common blocks and header files included by more than one particle module are found in src_model/include. In order to keep up the modularity of the code, routines in src_models/common thus *only* include files in src_model/include.

34.1 common/aux: auxiliary routines

Here we collect various routines that belong to particle physics, and hence do not reside in ds_core, but are not only useful for one specific particle module. Currently, the most important are

- A set of functions to handle *unique model IDs* for each particle model, which is essential in order not to repeat identical calculations when changing the model and thus to optimize numerical performance. A new such model ID should be assigned with dsnewidnumber whenever a new particle model is initalized (for the modules provided with the DarkSUSY release, this is automatically done in dsmodelsetup). Any routine in src_models can test with dsidnumberset whether this ID number is indeed set, and retrieve its value with dsidnumber (which then can be compared to the locally stored value of the ID number that was valid at the time when the respective routine was called the previous time).

- The function dsanthreshold returns the correction factor to a 2-body rate close to a kinematical threshold, resulting from the fact that one or both of the particles may be slightly off-shell. This implements the simplified treatment presented in Ref. [31], c.f. their Fig. 5, and hence assumes that the decay products of the virtual particle are (effectively) massless.

34.2 common/sm: standard model

The routines collected in this folder provide a convenient shortcut to include the most basic properties of the standard model in a BSM particle module. Currently, the most important functionalites

collected here are given by

- An initialization routine dsinit_sm, which can be called directly from the corresponding dsinit_module. After a call to this routine, the functions dsmass(kPDG) and dswidth(kPDG) correctly return masses and widths of the standard model particles – kPDG being the PDG code [130] – as stored centrally in src_models/include/dssmparam.h.
- Running quark masses are provided up to the 4-loop level, with SM contributions only, both for pole and $\overline{MS}$ masses [158].
- The SM contributions to the strong coupling constant are also provided up to the 4-loop level.
- Another convenience function is dsgf2s2thw, which calculates $\sin^2(\theta_W)$ at the m_Z scale: in praxis, its most important application is to ensure that a particle module can adopt a consistent relation between m_Z and m_W, which for example can be crucial in order to keep large numerical cancellations under control.

Let us stress that standard model physics in DarkSUSY is by far not restricted to the routines collected in common/sm. Much is presently still contained in the mssm module. It will be (further) disentangled from the SUSY-specific parts as new modules are added to the code that require this functionality.

34.3 common/aux: src_models/common/aux

34.3.1 Routine headers – fortran files

dsanGbar.f

```
************************************************************************
*** Function dsanGbar returns Gbar as defined in Eq.(B.4) of 2111.14871.    ***
*** A vacuum decay rate - or cross section that only depends on the CMS     ***
*** energy - thus must be multiplied by this factor in order to obtain the  ***
*** corresponding rate/cross section if the final state particles are in    ***
*** equilibriun with a thermal bath of temperature T.                       ***
***                                                                         ***
*** Input:                                                                  ***
***    z        - Lorentz boost gamma from CMS to cosmic frame              ***
***    sqrts    - CMS energy [GeV]                                          ***
***    TSM      - temperature of heat bath [GeV]                            ***
***    m_1, m_2 - mass of final state particle [GeV]                        ***
***    eps_stat - 1 (-1) for final state fermions (bosons)                  ***
***                                                                         ***
*** Output: Gbar [dimensionless]                                           ***
***                                                                         ***
*** author: torsten.bringmann@fys.uio.no + Kristian Vangsnes, 2021-10-30    ***
************************************************************************
      real*8 function dsanGbar(z,sqrts,TSM,m_1,m_2,eps_stat)
```

dsansommerfeld.f

```
************************************************************************
*** function dsansommerfeld returns the Sommerfeld correction factor S      ***
*** for a p-wave and s-wave process that results from the multiple           ***
*** exchange of a single vector or scalar mediator type. The total          ***
*** crosss ection is thus given as sv = S * (sv)_0, where (sv)_0 is the     ***
*** tree level result.                                                      ***
*** This implements Eq. (34) in 1302.3898, based on S. Cassel,              ***
*** J. Phys. G 37, 105009 (2010) [arXiv:0903.5307 [hep-ph]]                 ***
```

```
***                                                                          ***
*** input:                                                                  ***
***                                                                          ***
***    mDM   - DM mass (in GeV)                                              ***
***    mmed  - vector mediator mass (in GeV)                                 ***
***    alpha - (DM-mediator coupling)^2 /(4pi)                               ***
***    vrel  - relative velocity between DM particles /in CMS)               ***
***    l     - determines partial wave [0:s-wave, 1:p-wave]                  ***
***                                                                          ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date: 2018-02-21                                                        ***
********************************************************************************

      real*8 function dsansommerfeld(mDM,mmed,alpha,vrel,l)
```

dsanthreshold.f

```
************************************************************************
*** function dsanthreshold returns the correction factor to a 2-body
*** rate close to a kinematical threshold. We consider a process
*** DM DM -> A B, and assume that just above threshold the cross
*** section scales as
***
***   (sigma v)_AB \propto [\lambda(1, m_A^2/s, m_B^2/s)]^(n+1/2)
***
*** For s-wave annihilation, the cross section including the
*** possibility to produce a virtual B (assumed to decay exclusively
*** to much lighter particles) is then given by
***
***   (sigma v)_AB^* = (sigma v)_red * dsanthreshold
***
*** where the reduced cross section is given by
***
***   (sigma v)_red = S (sigma v)_AB / [\lambda(1, m_A^2/s, m_B^2/s)]^(n+1/2)
***
*** which by construction is non-zero at threshold. The symmetry factor
*** is S=2 if A=B, otherwise S=1. [more complicated symmetry factors arise
*** if A is the same as one of the decay products of B, see 1705.03466]
***
***    Input:
***
***          s      - CMS energy squared
***          mA, mB - mass of particle A, B
***          width  - total decay width of B
***          n      - integer (definition see above)
***
***  The output is dimension-less.
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2017-05-11
************************************************************************

      real*8 function dsanthreshold(s,mA,mB,width,n)
```

dscheckmodule.f

```
********************************************************************************
*** subroutine dscheckmodule checks whether a routine or function is called
*** for the correct particle module. This internal consistency check should
*** (at least) be included for all required functions.
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2015-06-11
```

```
**************************************************************************

      subroutine dscheckmodule(mymodule,functionname)
```

dsddsigma.f

```
      subroutine dsddsigma(v,e,a,z,sigij,ierr)
c-----------------------------------------------------------------------
c
c  type : INTERFACE
c  desc : UNpolarized *equivalent* WIMP nucleus cross section including form factors
c      sigma = E_{max} d\sigma/dE_R
c  The calculation depends on the definition of the couplings G_a, i.e.,
c    the dd scheme,
c      sigma = \sum_{ij} sig_{ij}
c    where sig_{ij} = \sum_{NN'} G_i^{N*} G_j^{N'} P_{ij}^{NN'}.
c  input:
c    v : real*8    : WIMP-nucleus relative velocity in km/s
c    e : real*8    : nucleus recoil energy in keV
c    a : integer   : nucleus mass number
c    z : integer   : nucleus atomic numbers
c  output:
c    sigij(27,27) : real*8  : partial cross section array in cm^2
c                       note that the usual spin-independent scattering
c                       cross section is sigij(1,1) and the usual
c                       spin-dependent one is sigij(4,4)
c    ierr  : integer : error code (0=no error)
c
c  Note: If you call this routine with arguments
c    0.0d0,0.0d0,1,1 etc you get the usual scattering cross sections
c       on protons
c    0.0d0,0.0d0,1,0 etc you get the usual scattering cross sections
c       on neutrons
c
c  Note: This particular version of dsddsigma can be used (as an interface
c        function) by all particle modules that provide the function dsddgpgn.
c        Alternatively, a particle module can provide its own version of
c        dsddsigma, in which case this version will be overriden.
c
c  author: paolo gondolo (paolo.gondolo@utah.edu) 2016
c  mod: torsten bringmann (generalized and placed in /common, and hence
c                          usable by all particle modules) 2018
c=======================================================================
```

dsddsigmanucleon.f

```
      subroutine dsddsigmanucleon(v,e,sigsip,sigsin,sigsdp,sigsdn,ierr)
c-----------------------------------------------------------------------
c  dark matter - nucleon cross section.
c
c  type : commonly used
c  desc : Calculate nuclear cross sections
c
c  common:
c    'dssusy.h' - file with susy common blocks
c  output:
c    sigsip, sigsin : proton and neutron spin-independent cross sections
c    sigsdp, sigsdn : proton and neutron spin-dependent cross sections
c    units: cm^2
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995,2002
c     13-sep-94 pg no drees-nojiri twist-2 terms
c     22-apr-95 pg important bug corrected [ft -> ft mp/mq]
c     06-apr-02 pg drees-nojiri treatment added
c     06-feb-16 pg renamed dsddsigmanucleon (was dsddneunuc)
```

```
c     16-mar-19 tb bug fixes (catch errors independently)
c=======================================================================
```

dsidnumber.f

```
      integer function dsidnumber()
************************************************************************
*** This function returns an integer number that is required to be
*** unique for each new particle physics model. This routine is used
*** by other routines both in src_models/ and in src/ to be able to optimze
*** the code by only recalculting quantities when needed (e.g. for a new
*** model).
***
***    Type: interface function
***
*** The subroutine dsnewidnumber creates a new one and is called for each
*** new particle physics model
***
***  NB: You can first check with dsidnumberset whether this function can safely
***      be used!
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: December 10, 2014
************************************************************************
```

dsidnumberset.f

```
      logical function dsidnumberset()
************************************************************************
*** The function dsidnumber returns an integer number that is
*** unique for each new particle physics model. This function, dsidnumberset,
*** works as a check if the function dsidnumber is properly initialized in the
*** respective particle module. If it returns false, the output of dsidnumber
*** should not be used
***
***    Type: interface function
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: 10/10/20174
************************************************************************
```

dsnewidnumber.f

```
       subroutine dsnewidnumber()
************************************************************************
*** This subroutine creates a new unique id number and should be called
*** for each new particle physics model. The Function dsidnumber returns
*** the number and can be used by different functions for optimzation
*** purposes.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: December 10, 2014
*** mod: torsten.bringmann@fys.uio.no, 10/10/2017 (added checksum)
************************************************************************
```

34.4 common/sm: src_models/common/sm

34.4.1 Routine headers – fortran files

dsfindmtmt.f

```
      subroutine dsfindmtmt
No header found.
```

dsfindmtmt1loop.f

```
      subroutine dsfindmtmt1loop
No header found.
```

dsfindmtmt4loop.f

```
      subroutine dsfindmtmt4loop
No header found.
```

dsgf2s2thw.f

```
**********************************************************************
*** dsgf2s2thw takes the Fermi constant (as measured from mu decay)
*** and calculates the sin^2(theta_W) at the MZ scale.
*** Formulas from PDG 2007, chapter 10.
*** Input: opt = 1, calculate the on-shell value (use generally)
***        opt = 2, calculate s_Z
***        opt = 3, calculate the MS-bar value at MZ (use for GUT relations,
***                 and other places that require this value)
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01<
**********************************************************************

      real*8 function dsgf2s2thw(gf,alphmz,mz,mt,opt)
```

dshvev_finiteT.f

```
*****************************************************************************
*** Function dshvev_finiteT returns the vacuum expectation value (vev) of   ***
*** the standard model Higgs doublet.                                       ***
***                                                                         ***
*** Input: Heat bath temperature [GeV]                                      ***
***                                                                         ***
*** Output: Higgs vev [GeV]                                                 ***
***                                                                         ***
*** author: torsten.bringmann@fys.uio.no, 2021-08-21                        ***
*****************************************************************************
      real*8 function dshvev_finiteT(T)
```

dsinit_sm.f

```
*****************************************************************************
***  This is the recommended way to initialize SM physics, and typically    ***
***  called at the beginning of the dsinit_module subroutine of each        ***
***  particle physics module. It automatically sets basic properties of SM  ***
***  particles, running masses etc. Note that this function is just provided***
***  for convenience -- each particle physics model can either adopt its own***
***  way of representing SM physics, or override any individual settings    ***
***  made in this routine by a corresponding statement in dsinit_module     ***
***  after calling this basic initialization routine.                       ***
***                                                                         ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)               ***
***  Date: 2016-11-16                                                       ***
```

```
************************************************************************
      subroutine dsinit_sm
```

dsmass.f

```
************************************************************************
*** This function returns the mass of standard model particles, identified  ***
*** by their PDG code (http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf)   ***
***                                                                        ***
*** Input:                                                                 ***
***      kPDG  | particle                                                  ***
***    --------+----------                                                 ***
***         1  | d quark                                                   ***
***         2  | u quark                                                   ***
***         3  | s quark                                                   ***
***         4  | c quark                                                   ***
***         5  | b quark                                                   ***
***         6  | t quark                                                   ***
***        11  | electron                                                  ***
***        12  | nu_e                                                      ***
***        13  | muon                                                      ***
***        14  | nu_mu                                                     ***
***        15  | tau                                                       ***
***        16  | nu_tau                                                    ***
***        21  | gluon                                                     ***
***        22  | photon                                                    ***
***        23  | Z                                                         ***
***        24  | W                                                         ***
***        25  | h                                                         ***
***                                                                        ***
*** Output: particle mass [in GeV]                                         ***
***                                                                        ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                       ***
************************************************************************
      real*8 function dsmass(kPDG)
```

dsmass_finiteT.f

```
************************************************************************
*** Function dsmass_finiteT returns the mass of standard model particles at ***
*** finite temperature (in a SM plasma), identified by their PDG code       ***
*** (http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf).                    ***
***                                                                        ***
*** Input:                                                                 ***
***      T - SM heat bath temperature [GeV]                                ***
***      kPDG  | particle                                                  ***
***    --------+----------                                                 ***
***         1  | d quark                                                   ***
***         2  | u quark                                                   ***
***         3  | s quark                                                   ***
***         4  | c quark                                                   ***
***         5  | b quark                                                   ***
***         6  | t quark                                                   ***
***        11  | electron                                                  ***
***        12  | nu_e                                                      ***
***        13  | muon                                                      ***
***        14  | nu_mu                                                     ***
***        15  | tau                                                       ***
***        16  | nu_tau                                                    ***
***        21  | gluon                                                     ***
***        22  | photon                                                    ***
***        23  | Z                                                         ***
***        24  | W                                                         ***
***        25  | h                                                         ***
```

```
***                                                                          ***
*** Output: particle mass [in GeV]                                          ***
***                                                                          ***
*** Note: Thermal masses are presently approximations based on expressions  ***
***       from hep-ph/9307210 (for fermions) and hep-ph/9307210 (for gauge  ***
***       bosons). The Higgs thermal mass is tabulated from 2111.14871.     ***
***                                                                          ***
*** author: torsten.bringmann@fys.uio.no, 2021-08-11                        ***
********************************************************************************
      real*8 function dsmass_finiteT(kPDG,T)
```

dsmqpole1loop.f

```
      real*8 function dsmqpole1loop(mqmq)
No header found.
```

dsmqpole4loop.f

```
      function dsmqpole4loop(k,mqmq)
No header found.
```

dsmsbaralph3.f

```
      function dsmsbaralph3(alph3_1,mu_1,mu_2)
c
c     computes the running msbar alpha_s using
c     formulas collected in Chetyrkin, Kuehn, and
c     Steinhauser, hep-ph/0004189 (Thanks to Andre
c     Hoang for providing the reference)
c
c     Paolo Gondolo (paolo@physics.utah.edu) 2008-06-30
c
```

dsralph3.f

```
      real*8 function dsralph3(mscale)

c... TB: stripped off MSSM part: 11/2016
c...     added cut at confinement scale (divergent below): 11/2021
```

dsralph31loop.f

```
      real*8 function dsralph31loop(mscale)
No header found.
```

dsralph34loop.f

```
      function dsralph34loop(mscale)
c
c     computes the running msbar alpha_s starting from
c     the input value alph3mz at the scale mz using
c     formulas collected in Chetyrkin, Kuehn, and
c     Steinhauser, hep-ph/0004189 (Thanks to Andre
c     Hoang for providing the reference)
c
c     Paolo Gondolo (paolo@physics.utah.edu) 2008-06-30
c
c     Modified to be externally reset 2009-10-20 Pat Scott
c
```

dsrmq.f

```
      real*8 function dsrmq(mscale,kpart)
No header found.
```

dsrmq1loop.f

```
      real*8 function dsrmq1loop(mscale,kpart)
No header found.
```

dsrmq4loop.f

```
      function dsrmq4loop(mscale,k)
c
c     run the mass of particle k from the msbar mass
c     at the scale of the msbar mass, i.e. mqmq, to the
c     scale mscale
c     using 4 loop formulas in  Chetyrkin, Kuehn, and
c     Steinhauser, hep-ph/0004189 (Thanks to Andre
c     Hoang for providing the reference)
c
c     Paolo Gondolo (paolo@physics.utah.edu) 2008-06-30
c     updated header to correctly describe normalization/
c     reference point of running (TB, 2015-11-05)
c
```

dssmconst_ckm.f

```
      subroutine dssmconst_ckm
c-----------------------------------------------------------------------
c  useful constants for CKM mixing
c  common:
c    'dssm.h' - file with SM common blocks
c  author: paolo gondolo 1994-1999
c  modified: 031105 neutrino's yukawa corrected (pg)
c  modified: 13-04-2019 moved to SM part (tb)
c=======================================================================
```

dssmconst_couplings.f

```
      subroutine dssmconst_couplings
c-----------------------------------------------------------------------
c  useful constants
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo 1994-1999
c  modified: 031105 neutrino's yukawa corrected (pg)
c  modified: 13-04-2019 moved to SM part (tb)
c=======================================================================
```

dssmgammah.f

```
*************************************************************************
*** Function *dssmgammah returns the total Standard-Model Higgs decay width ***
***                                                                   ***
*** Input:                                                            ***
***    sqrts - CMS energy [GeV]                                       ***
***                                                                   ***
*** Output: (offshell) width in GeV                                   ***
***                                                                   ***
*** Tabulated width / analytical expressions according to flag        ***
```

```
*** gammahhow set in dsinit_sm [default: hdecay]                          ***
***                                                                       ***
*** Author: Paolo Gondolo 2016                                           ***
*** mod 2021       (tb): added options for how to handle Higgs width      ***
*** mod 2021-15-09 (tb): removed free quark/gluon, and added hadronic     ***
***                       expressions below confinement scale             ***
*****************************************************************************
      function dssmgammah(sqrts)
```

dssmgammah_Dittmaier_tab.f

```
      function dssmgammah_Dittmaier_tab(ichannel,sqrts)
***
*** Returns the standard model partial Higgs decay width for channel ichannel,
*** for the low and intermediate mass range (90-490 GeV), as tabulated in
*** Dittmaier et al 1101.0593.
*** (remember to update dsinit_sm if tabulation range is changed!)
***
*** List of channels:
***   ichallen = 0  :  total Width
***   ichannel = 1  :  nue + anti-nue [not tabulated]
***   ichannel = 2  :  e+ + e- [not tabulated]
***   ichannel = 3  :  numu + anti-numu [not tabulated]
***   ichannel = 4  :  mu+ + mu-s
***   ichannel = 5  :  nutau + anti-nutau [not tabulated]
***   ichannel = 6  :  tau+ + tau-
***   ichannel = 7  :  u + ubar [not tabulated]
***   ichannel = 8  :  d + dbar [not tabulated]
***   ichannel = 9  :  c + cbar
***   ichannel = 10  :  s + sbar
***   ichannel = 11  :  t + tbar
***   ichannel = 12  :  b + bbar
***   ichannel = 13  :  gamma + gamma
***   ichannel = 14  :  W+ + W-
***   ichannel = 15  :  Z + Z
***   ichannel = 16  :  g + g
***   ichannel = 17  :  H + H [not kinematically available !]
***   ichannel = 18  :  Z + gamma
***
*** Author: Paolo Gondolo 2016
*** mod: Torsten Bringmann 2021 (updated/clarified description,
***                              merged in total width tabulation)
***
```

dssmgammah_hadron_tab.f

```
*****************************************************************************
*** Function dssmgammah_hadron_tab returns the standard model (off-shell)   ***
*** Higgs decay width into hadrons. Tabulated from 1809.01876               ***
***                                                                         ***
*** Input:                                                                  ***
***   sqrts - CMS energy of off-shell Higgs                                 ***
***   ichannel - channel to consider                                        ***
***                                                                         ***
*** List of channels:                                                       ***
***   ichannel = 0  :  total decay width into hadrons                       ***
***   ichannel = 1  :  4pi, 2eta, 2rho,...                                  ***
***   ichannel = 2  :  2 K                                                  ***
***   ichannel = 3  :  2 pi                                                 ***
***                                                                         ***
*** Output is the total (ichannel=0) and partial width, respectively,       ***
*** in GeV. The total width is tabulated up to 7.9 GeV, partial widths      ***
*** for ichannel 1-3 (from a dispersive analysis) only up to 2 GeV.         ***
```

```
***                                                                          ***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no                  ***
*** Date: 2021-09-13                                                         ***
******************************************************************************
      real*8 function dssmgammah_hadron_tab(ichannel,sqrts)
```

dssmgammah_hdecay_tab.f

```
      function dssmgammah_hdecay_tab(ichannel,sqrts)
***
*** Returns the standard model partial Higgs decay width for channel ichannel
*** This tabulation is made with HDECAY. Currently tabulated range:
*** 0.81 GeV -- 4 TeV (remember to update dsinit_sm if this is changed!).
***
*** List of channels:
***   ichannel = 0  :  total Width
***   ichannel = 1  :  nue + anti-nue [not tabulated]
***   ichannel = 2  :  e+ + e- [not tabulated]
***   ichannel = 3  :  numu + anti-numu [not tabulated]
***   ichannel = 4  :  mu+ + mu-s
***   ichannel = 5  :  nutau + anti-nutau [not tabulated]
***   ichannel = 6  :  tau+ + tau-
***   ichannel = 7  :  u + ubar [not tabulated]
***   ichannel = 8  :  d + dbar [not tabulated]
***   ichannel = 9  :  c + cbar
***   ichannel = 10  :  s + sbar
***   ichannel = 11  :  t + tbar
***   ichannel = 12  :  b + bbar
***   ichannel = 13  :  gamma + gamma
***   ichannel = 14  :  W+ + W-
***   ichannel = 15  :  Z + Z
***   ichannel = 16  :  g + g
***   ichannel = 17  :  H + H [not kinematically available !]
***   ichannel = 18  :  Z + gamma
***
*** Author: Paolo Gondolo 2016
*** mod (Joakim Edsjö): implemented hdecay tables
*** mod 10/2021 (JE/TB): updated hdecay runs
***
```

dssmgammahpartial.f

```
******************************************************************************
*** Function dssmgammahpartial returns the standard model (off-shell)       ***
*** partial Higgs decay width, for channel ichannel.                         ***
***                                                                          ***
*** Input:                                                                   ***
***   sqrts - CMS energy of off-shell Higgs                                  ***
***   ichannel - channel to consider                                         ***
***                                                                          ***
*** List of channels:                                                        ***
***   ichannel = 1   :  nue + anti-nue                                       ***
***   ichannel = 2   :  e+ + e-                                              ***
***   ichannel = 3   :  numu + anti-numu                                     ***
***   ichannel = 4   :  mu+ + mu-                                            ***
***   ichannel = 5   :  nutau + anti-nutau                                   ***
***   ichannel = 6   :  tau+ + tau-                                          ***
***   ichannel = 7   :  u + ubar                                             ***
***   ichannel = 8   :  d + dbar                                             ***
***   ichannel = 9   :  c + cbar                                             ***
***   ichannel = 10  :  s + sbar                                             ***
***   ichannel = 11  :  t + tbar                                             ***
***   ichannel = 12  :  b + bbar                                             ***
***   ichannel = 13  :  gamma + gamma                                        ***
```

```
***   ichannel = 14  :  W+ + W-                                          ***
***   ichannel = 15  :  Z + Z                                            ***
***   ichannel = 16  :  g + g                                            ***
***   ichannel = 17  :  H + H                                            ***
***   ichannel = 18  :  gamma + Z                                        ***
***   ichannel = 19  :  hadrons (for sqrts<Ehadron)                      ***
***                                                                      ***
*** Default settings:                                                   ***
*** Tables are calculated with HDecay v. 6.51                           ***
*** Analytic expressions from Cline et al 1306.4710, Ilisie 2011 as     ***
*** fallback options (outside tabulated range/channels)                 ***
*** [set gammahhow to a different value in dsinit_sm for alternatives, e.g. ***
***  using tables by Dittmaier instead]                                 ***
***                                                                      ***
*** NB: Channels 7-12 & 16 assume free quarks and gluons, and hence return  ***
***     zero below the confinement scale ('Ehadron', set in dsinit_sm)!  ***
***     Conversely, ichannel=19 returns the total hadronic widths below the ***
***     confinement scale, and zero otherwise.                           ***
***                                                                      ***
*** Author: Paolo Gondolo 2016                                          ***
*** mod ??/2016 (Joakim Edsjö): allow using Hdecay instead of Dittmaier     ***
***                            tables                                    ***
*** mod 08/2021 (Saniya Heeba): 0503172 for gluon channel               ***
*** mod 10/2021 (tb): added hadronization scale                         ***
**************************************************************************
      real*8 function dssmgammahpartial(ichannel,sqrts)
```

dswidth.f

```
**************************************************************************
*** This function returns the (SM) width of standard model particles,   ***
*** identified by their PDG code                                        ***
*** (http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf)                  ***
***                                                                      ***
*** Input:                                                              ***
***      kPDG  | particle                                                ***
***    --------+----------                                             ***
***         1  | d quark                                                 ***
***         2  | u quark                                                 ***
***         3  | s quark                                                 ***
***         4  | c quark                                                 ***
***         5  | b quark                                                 ***
***         6  | t quark                                                 ***
***        11  | electron                                                ***
***        12  | nu_e                                                    ***
***        13  | muon                                                    ***
***        14  | nu_mu                                                   ***
***        15  | tau                                                     ***
***        16  | nu_tau                                                  ***
***        21  | gluon                                                   ***
***        22  | photon                                                  ***
***        23  | Z                                                       ***
***        24  | W                                                       ***
***        25  | h                                                       ***
***                                                                      ***
*** Output: particle width [in GeV]                                     ***
***                                                                      ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                    ***
**************************************************************************

      real*8 function dswidth(kPDG)
```

Chapter 35

The empty model

The empty module cannot be used to perform any real calculations. Its purpose is for debugging and testing only, with the main advantage being that it contains (empty versions of) all interface functions that the core library is aware of (for a full list, see Table 3.3). In some cases, this makes it a good starting point to create new particle physics modules.

35.1 empty/ac: Accelerator constraints

35.1.1 Routine headers – fortran files

dsacbnd.f

```
*************************************************************************
***  subroutine dsacbnd checks collider bounds                        ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
***  desc : Check collider bounds                                     ***
***                                                                   ***
*** author: Torsten.Bringmann@fys.uio.no                              ***
*** date  : 2015-06-11                                                ***
*************************************************************************
      subroutine dsacbnd(excl)
```

35.2 empty/an: Annihilation routines

35.2.1 Routine headers – fortran files

dsanwx.f

```
*************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.  ***
***                                                                   ***
***  type : interface                                                 ***
***  desc : Self-annihilation invariant rate                          ***
***                                                                   ***
***  Input:                                                           ***
***    p - initial cm momentum (real) for DM annihilations            ***
***  Output:                                                          ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  passed to dsrdens by dsrdomega.                                        ***
***                                                                         ***
*** author: Torsten.Bringmann@fys.uio.no                                    ***
*** date 2015-06-11                                                         ***
*******************************************************************************
      real*8 function dsanwx(p)
```

dsanwx_finiteT.f

```
*******************************************************************************
*** Function dsanwx_finiteT provides the effective, invariant rate for DM   ***
*** self-annihilation. Same as dsanwx, but including finite temperature     ***
*** effects in the cross section, as well as quantum statistics effects     ***
*** from the final state (and/or virtual heat bath) particles.              ***
***                                                                         ***
***  type : INTERFACE                                                       ***
***  desc : Self-annihilation invariant rate at finite temperature          ***
***                                                                         ***
***  Input:                                                                 ***
***    p   - cm momentum of each of the initial DM particles [GeV]          ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma          ***
***          temperature [both in GeV] at which sigma v is evaluated        ***
***                                                                         ***
***  Author: Torsten Bringmann                                              ***
***  Date: 21/09/2021                                                       ***
*******************************************************************************
      real*8 function dsanwx_finiteT(p,x)
```

dssigmav0tot.f

```
*******************************************************************
*** function dssigmav0tot returns the *total* annihilation cross section
*** sigma v at p=0 for neutralino-neutralino annihilation.
*** This is obtained by summing over all implemented 2-body channels
*** (as returned by dssigmav) plus contributions from final states with
*** more particles.
***
***  type : interface
***
***  desc : Total annihilation cross section at $v=0$
***
*** Units of returned cross section: cm^3 s^-1
***
*** author: Torsten.Bringmann@fys.uio.no
*** date: 2014-11-14
*******************************************************************

      real*8 function dssigmav0tot()
```

35.3 empty/cr: Cosmic rays

35.3.1 Routine headers – fortran files

dscrsource.f

```
************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : interface
***
***   desc : Source term for dark matter induced cosmic rays
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann@fys.uio.no
*** date  : 2015-06-11
************************************************************************

      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***  desc : Source term for monochromatic contributions from dark matter
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***
```

```
*** author: Torsten.Bringmann@fys.uio.no
*** date  : 2015-06-11
**************************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

35.4 empty/dd: Direct detection

35.4.1 Routine headers – fortran files

dsddgpgn.f

```
**************************************************************************
***  subroutine dsddgpgn returns DM nucleon four-fermion couplings          ***
***                                                                         ***
***  type : interface                                                       ***
***         [NB: this routine will eventually be phased out as *interface*  ***
***              routine. Currently, neutrino telescope routines are the    ***
***              only routines in /src that access it. ]                    ***
***                                                                         ***
***  desc : Four fermion couplings.                                         ***
***                                                                         ***
***   output:                                                               ***
***     gg complex*16 array of couplings                                     ***
***          first index is operator index (1:ddng), with ddng in dsddcom.h ***
***          second index is proton/neutron (1:2)                           ***
***     units: GeV^-2                                                       ***
***                                                                         ***
***  author: torsten.bringmann@fys.uio.no 18-11-2016                        ***
***    2016-11-20 Paolo Gondolo abandoned scheme                            ***
**************************************************************************
      subroutine dsddgpgn(gg,ierr)
```

dsddsigma.f

```
      subroutine dsddsigma(v,e,a,z,sigij,ierr)
c--------------------------------------------------------------------------
c
c  type : INTERFACE
c  desc : UNpolarized *equivalent* WIMP nucleus cross section including form factors
c       sigma = E_{max} d\sigma/dE_R
c  The calculation depends on the definition of the couplings G_a, i.e.,
c    the dd scheme,
c      sigma = \sum_{ij} sig_{ij}
c    where sig_{ij} = \sum_{NN'} G_i^{N*} G_j^{N'} P_{ij}^{NN'}.
c  input:
c    v : real*8    : WIMP-nucleus relative velocity in km/s
c    e : real*8    : nucleus recoil energy in keV
c    a : integer   : nucleus mass number
c    z : integer   : nucleus atomic numbers
c  output:
c    sigij(27,27) : real*8  : partial cross section array in cm^2
c                        note that the usual spin-independent scattering
c                        cross section is sigij(1,1) and the usual
c                        spin-dependent one is sigij(4,4)
c    ierr  : integer : error code (0=no error)
c
c  Note: If you call this routine with arguments
c    0.0d0,0.0d0,1,1 etc you get the usual scattering cross sections
c       on protons
```

```
c    0.0d0,0.0d0,1,0 etc you get the usual scattering cross sections
c       on neutrons
c
c  author: torsten bringmann, 2018
c======================================================================
```

35.5 empty/ge: General routines

35.5.1 Routine headers – fortran files

dsdmspin.f

```
*************************************************************************
*** Function dsdmspin returns the dark matter spin (in hbar)          ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
***  desc : dark matter spin                                          ***
***                                                                   ***
*** author: Paolo Gondolo 2016-11-20                                  ***
*************************************************************************

      real*8 function dsdmspin()
```

dsmwimp.f

```
*************************************************************************
*** Function dsmwimp returns the WIMP mass (in GeV)                   ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
***  desc : Dark matter mass                                          ***
***                                                                   ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                  ***
*************************************************************************

      real*8 function dsmwimp()
```

35.6 empty/ini: Initialization routines

35.6.1 Routine headers – fortran files

dsinit_module.f

```
*************************************************************************
***  This is the initialization subroutine for the "empty" module which ***
***  essentially does not contain any actual particle model information at  ***
***  all.                                                             ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
***  desc : Intialzation of module                                    ***
***                                                                   ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)         ***
***  Date: 04/2014                                                    ***
*************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
*************************************************************************
***  subroutine dsmodelsetup sets up a particle model                   ***
***                                                                     ***
***  type : interface                                                   ***
***                                                                     ***
***  desc : Sets up a new particle physics model                        ***
***                                                                     ***
***  output:                                                            ***
***    ierr  -  0 no errors                                             ***
***    iwarn -  0 no warnings                                           ***
***                                                                     ***
*** author: Torsten.Bringmann@fys.uio.no                                ***
*** date 2014-05-09                                                     ***
*************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

35.7 empty/kd: Kinetic decoupling

35.7.1 Routine headers – fortran files

dskdm2.f

```
*********************************************************************
*** dskdm2 returns the full scattering amplitude for DM at zero
*** momentum transfer squared in the EMPTY model, SUMMED over both initial
*** and final spin and other internal states, and then divided by the DM
*** internal degrees of freedom.
***
***  type : INTERFACE
***  desc : Full scattering amplitude squared, averaged over momentum transfer
***
***  input: omega -- CMS MOMENTUM of scattering partner
***  output: omega -- CMS ENERGY of scattering partner
***
***  input:   SMtype - SM scattering partners
***             "    = 7,8,9 - u,d,s quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09
*********************************************************************

      real*8 function dskdm2(omega,SMtype)
```

dskdm2simp.f

```
*********************************************************************
*** dskdm2simp returns the scattering amplitude for DM at zero
*** momentum transfer squared in the EMPTY model, SUMMED over both initial
*** and final spin and other internal states, and then divided by the DM
*** internal degrees of freedom. This is only valid in the limit of
*** relativistic scattering partners with small energies omega, where we
*** can expand as
***
***    |M|**2 = cn*(omega/m0)**n + O( (omega/m0)**(n+1) )
***
***  type : INTERFACE
***  desc : Scattering amplitude squared, expanded in powers of energy
***
```

```
***  input: SMtype   - SM scattering partners:
***                    7,8,9 - u,d,s quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09
**********************************************************************

      subroutine dskdm2simp(SMtype,cn,n)
```

dskdparticles.f

```
*****************************************************************************
*** Prepares integration of Boltzmann equation and has to be called        ***
*** before dskdboltz (called by dskdtkd)                                   ***
***                                                                        ***
***  type : interface                                                      ***
***                                                                        ***
***  desc : Initalization of kinetic decoupling for module                 ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date  : 2015-06-11                                                     ***
*****************************************************************************
      subroutine dskdparticles
```

35.8 empty/rd: Relic density

35.8.1 Routine headers – fortran files

dsrdparticles.f

```
      subroutine dsrdparticles(option,nsize,TSM,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
**********************************************************************
*** subroutine dsrdparticles returns which particles are included in
***   the calculation of the WIMP relic density (coannihilations,
***   resonances, thresholds)
***
***  type : interface
***
***  desc : Particles included in relic density calculation
***  desc : (coannihilations, resonances and thresholds)
***
*** input:
***   option = 0 - default
***            !=0 - include various subsets of coannihilations
***   nsize = size of arrays, do not fill larger than this
***    TSM  = finite temperature of the SM heat bath
***            (if TSM=0d0, finite-T effects will be ignored)
***
*** output
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)
***    ncoann   - number of particles coannihilating
***    mcoann   - relic and coannihilating mass in gev
***    dof      - internal degrees of freedom of the particles
***    nrs      - number of resonances to take special care of
***    rm       - mass of resonances in gev
***    rw       - width of resonances in gev
***    nt       - number of thresholds to take special care of
***               do not include coannihilation thresholds (that's automatic)
***    tm       - sqrt(s) of the thresholds in gev
```

```
*** This output is used by the RD routines in src/rd/
**** author: Torsten Bringmann
*** (derived from mssm/dsrdparticles)
*** date: 14-04-06
**********************************************************************
```

35.9 empty/se_yield: Yields from annihilation in the Sun/Earth

35.9.1 Routine headers – fortran files

dsseyield.f

```
**********************************************************************
*** Function dsseyield calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth summing
*** over all annihilation channels.
***
***  type : interface
***
***  desc : Yields from annihilation in the Sun/Earth
***
*** Inputs:
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
```

```
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1.
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
***
*** Author: Torsten.Bringmann@fys.uio.no
*** Date: 11/06/2015
**************************************************************************

      real*8 function dsseyield(e,theta,wh,kind,type,istat)
```

35.10 empty/si: Dark matter self-interactions

35.10.1 Routine headers – fortran files

dssisigtm.f

```
**************************************************************************
*** Function dssisigtm provides the momentum-transfer cross section per DM  ***
*** mass, in conventional units of cm**2/g.                                 ***
***                                                                         ***
***  type : interface                                                       ***
***  desc : Self-interaction momentum-transfer cross section                ***
***                                                                         ***
***  Input:                                                                 ***
***    vkms - relative velocity of scattering DM particles (in km/s)        ***
***                                                                         ***
***  This interfacte function is required by dssisigtmav in src/.           ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2015-05-18                                                         ***
**************************************************************************
      real*8 function dssisigtm(vkms)
```

Chapter 36

Generic decaying dark matter

This module provides a simple phenomenological template for a generic decaying dark matter model. It is fully specified by the DM mass m_χ and the total decay rate Γ, along with a list of decay channels. The latter are specified by their branching ratios and the PDG codes of the final state standard model particles. Non-standard model final states can also be included, and are treated as invisible decays.

The phenomenology of this module is restricted to indirect DM searches, with full access to the corresponding routines in ds_core. DM scattering with nuclei is assumed to be negligible, as well thermal production of DM in the early universe.

36.1 generic_decayingDM/cr: Cosmic rays

36.1.1 Routine headers – fortran files

dscrsource.f

```
************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2016-02-06
************************************************************************
```

```
      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2015-06-21
************************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

36.2 generic_decayingDM/dec: Decay routines

36.2.1 Routine headers – fortran files

dsGammatot.f

```
************************************************************************
*** function dsGammatot returns the *total* decay rate / width of the
*** DM particle.
***
***  type : interface
***
*** Units of returned width: 1/s
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2016-02-06
************************************************************************

      real*8 function dsGammatot()
```

36.3 generic_decayingDM/ge: General routines

36.3.1 Routine headers – fortran files

dsdmspin.f

```
************************************************************************
*** Function dsdmspin returns the dark matter spin (in hbar)         ***
***                                                                  ***
***  type : interface                                                ***
***                                                                  ***
***  desc : dark matter spin                                         ***
***                                                                  ***
*** author: Paolo Gondolo 2016-11-20                                 ***
************************************************************************

      real*8 function dsdmspin()
```

dsmwimp.f

```
************************************************************************
*** Function dsmwimp returns the WIMP mass                           ***
***                                                                  ***
***  type : interface                                                ***
***                                                                  ***
*** author: torsten.bringmann@fys.uio.no, 2016-02-06                 ***
************************************************************************

      real*8 function dsmwimp()
```

36.4 generic_decayingDM/ini: Initialization routines

36.4.1 Routine headers – fortran files

dsgivemodel_decayingDM.f

```
************************************************************************
***  subroutine dsgivemodel_decayingDM reads in the parameters to describe  ***
***  a generic decaying DM particle and transfers them to common blocks     ***
***                                                                  ***
***  input:                                                          ***
***                                                                  ***
***    mDM        - DM mass (in GeV)                                 ***
***    DecRate    - Total decay rate Gamma_tot (in 1/s)              ***
***    n          - number of (2-body) decay channels                ***
***    BR         - array of size n containg the branching ratios    ***
***                 Gamma_i/Gamma_tot                                ***
***    PDG1, PDG2 - array of size n with PDG code for final state particles ***
***                 (e.g. 5 for bbar, 24 for W^+W^-)                 ***
***                                                                  ***
*** author: Torsten.Bringmann.fys.uio.no                             ***
*** date 2016-02-06                                                  ***
************************************************************************
      subroutine dsgivemodel_decayingDM(mDM,DecRate,n,BR,PDG1,PDG2)
```

dsinit_module.f

```
************************************************************************
***  This is the initialization subroutine for the "generic_decayingDM"     ***
***  module which just contains a simple DM model with a mass,              ***
***  toatl decay rate and branching fractions.                              ***
***                                                                         ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)               ***
***  Date: 2016-02-06                                                       ***
************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
************************************************************************
***  subroutine dsmodelsetup sets up a particle model                   ***
***                                                                     ***
***  type : interface                                                   ***
***                                                                     ***
***  desc : Sets up a new particle physics model                        ***
***                                                                     ***
***  output:                                                            ***
***    ierr  -  0 no errors                                             ***
***          - <0 if an error is encountered (theoretically inconsistent  ***
***               model parameters)                                     ***
***    iwarn -  0 no warnings                                             ***
***             1 kinematically disallowed channel has been specified;  ***
***               will be treated as invisible                          ***
***            >1 (further details about error)                         ***
***                                                                     ***
*** author: Torsten.Bringmann.fys.uio.no                                ***
*** date 2016-02-06                                                     ***
************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

Chapter 37

Generic FIMPs

Similar to the case of generic_wimp, the module generic_fimp provides a simple phenomenological template for a large class of DM candidates. Rather than being based on an actual particle physics model, it mostly serves to provide an illustration of how the functionalities of DarkSUSY can be used in phenomenological freeze-in studies of 'vanilla' FIMP DM, when only providing the absolute minimum of input parameters.

A generic FIMP model in DarkSUSY is set up by a call to dsgivemodel_generic_fimp, followed as usual by a call to dsmodelsetup. The model is thus fully defined by the input parameters of dsgivemodel_generic_fimp: the mass m_χ of the DM particle and a flag stating whether the DM particle is its own anti-particle or not; a *single* SM species that the DM particle is assumed to couple to (defined by its PDG code, PDGSM) and three parameters (c, Λ, n) that specify strength and type of the interaction. Concretely, the squared amplitude for $2 \to 2$ production in this model is assumed to take the form

$$|\mathcal{M}|^2 \equiv c \left(\frac{s}{\Lambda^2}\right)^n , \tag{37.1}$$

where s is the CMS energy squared.* Very roughly, this would, *e.g.*, correspond to the amplitude expected for contact interactions in an effective field theory that results from integrating out heavy d.o.f. (where 'heavy' in this context means higher energies than the temperature where freeze-in production dominates.

Presently, the generic_fimp module only allows the user to calculate the relic density from freeze-in production (as well as, in principle, the would-be density from freeze-out), since rates are expected to be too small to be relevant for indirect and direct detection signals. For phenomenological studies of the latter, the generic_wimp module is thus better suited.

*__Technical note.__ Allowed values for PDGSM are all elementary SM particles in the EW broken phase. Strictly speaking, these choices are ill-defined above the EW phase transition and, for gluons and quarks, below the hadronization scale; results of freeze-in calculations must hence be interpreted with care if the dominant freeze-in production happens outside this range.

For reference, we treat situations outside this range as follows:

- For quark and gluon final states, we use Eq. (37.1) only for $T > T_{\rm qcd}$ and for $\sqrt{s} > E_{\rm hadron}$ (global parameters set in dsinit and dsinit_sm, respectively); outside this range, we set $|\mathcal{M}|^2 = 0$.
- For $T > T_{\rm EW}$, fermions are chiral; we model their thermal masses as the *average* of the contributions from left- and right handed states – see dsmass_finiteT. Photons we continue to describe as a $U(1)$ particle with electromagnetic couplings, while we describe both Z and $W^\pm$ as unbroken $SU(2)$ bosons (with temperature-dependent transverse masses).

We stress that this implementation is just a heuristic way of gauging the impact of phase transitions on the result of freeze-in calculations when insisting on the possibility of specifying the interacting SM species with a single (integer) parameter PDGSM (in analogy to the generic_wimp module). A more realistic implementation is necessarily model-dependent (see, e.g., the silveira_zee module described in Chapter 40).

37.1 generic_fimp/an: Annihilation routines

37.1.1 Routine headers – fortran files

dsanwx.f

```
************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.    ***
***                                                                          ***
***  type : interface                                                        ***
***                                                                          ***
***  Input:                                                                  ***
***    p - initial cm momentum (real) for DM annihilations                   ***
***  Output:                                                                 ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  passed to dsrdens by dsrdomega.                                         ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2021-11-19                                                         ***
************************************************************************
      real*8 function dsanwx(p)
```

dsanwx_finiteT.f

```
************************************************************************
*** Function dsanwx_finiteT provides the effective, invariant rate for DM   ***
*** self-annihilation. Same as dsanwx, but including finite temperature     ***
*** effects in the cross section, as well as quantum statistics effects     ***
*** from the final state (and/or virtual heat bath) particles.              ***
***                                                                         ***
***  type : INTERFACE                                                       ***
***  desc : Self-annihilation invariant rate at finite temperature          ***
***                                                                         ***
***  Input:                                                                 ***
***    p   - cm momentum of each of the initial DM particles [GeV]          ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma          ***
***          temperature [both in GeV] at which sigma v is evaluated        ***
***                                                                         ***
***  Author: Torsten Bringmann                                              ***
***  Date: 2021-11-19                                                       ***
************************************************************************
      real*8 function dsanwx_finiteT(p,x)
```

dssigmav_eff.f

```
************************************************************************
*** function dssigmav_eff returns the *effective*, total annihilation cross ***
*** section sigma v for DM-DM annihilation in the plasma (or COSMIC) frame  ***
*** and at finite temperature, as it appears on the right-hand side of the  ***
*** Boltzmann equation. Here v is defined to be the relative velocity of    ***
*** one particle in the frame of the other. For a full definition see       ***
*** (A.1,A.2) in 2111.14871.                                               ***
***                                                                        ***
```

```
***  Input:                                                                 ***
***    p   - cm momentum of each of the initial DM particles [GeV]          ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma          ***
***          temperature [both in GeV] at which sigma v is evaluated        ***
***                                                                         ***
*** Units of returned cross section: cm^3 s^-1                             ***
***                                                                         ***
*** Author: Torsten Bringmann                                              ***
*** Date: 2021-11-19                                                       ***
*****************************************************************************
      real*8 function dssigmav_eff(p,x)
```

dssigmav_finiteT.f

```
*****************************************************************************
*** function dssigmav_finiteT returns the *total* annihilation cross       ***
*** section sigma v at p=0 for WIMP-WIMP annihilation, evaluated in the     ***
*** CMS frame and at finite temperature. Here v is defined to be the       ***
*** relative velocity of one particle in the frame of the other.           ***
***                                                                        ***
***  Input:                                                                ***
***    p     - cm momentum of each of the initial DM particles [GeV]       ***
***    gamma - Lorentz boost wrt the frame of the heat bath                ***
***    TSM   - SM/plasma temperature [in GeV] at which sigma v is evaluated ***
***                                                                        ***
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                        ***
*** Author: Torsten Bringmann (2021)                                      ***
*** Date: 2021-11-19                                                      ***
*****************************************************************************
      real*8 function dssigmav_finiteT(p,gamma,TSM)
```

37.2 generic_fimp/fi: Freeze-in routines

A call to dsfiset_generic_FIMP prior to dsfi2to2rhs allows to perform freeze-in calculations with modified default settings. For example, call dsfiset_generic_FIMP('quantum_statistics','off') will lead to a freeze-in calculation without taking into account the effect of quantum statistics for the heat bath particles, and call dsfiset_generic_FIMP('finiteT','off') will switch off other finite-temperature effects (thermal masses and further temperature-dependence due to phase transitions). Default settings are restored with corresponding calls to the same routine, replacing 'off' with 'default'.

37.2.1 Routine headers – fortran files

dsfiset_generic_FIMP.f

```
      recursive subroutine dsfiset_generic_FIMP(key,value)
c...  desc : Set parameters for freeze-in routines
c...  key,value - character strings specifying choice to be made
c...  author: torsten bringmann 2021-11-19
```

37.3 generic_fimp/ge: General routines

37.3.1 Routine headers – fortran files

dsmwimp.f

```
************************************************************************
*** Function dsmwimp returns the WIMP mass                           ***
***                                                                  ***
***  type : interface                                                ***
***                                                                  ***
*** author: torsten.bringmann@fys.uio.no, 2021-11-18                 ***
************************************************************************

      real*8 function dsmwimp()
```

37.4 generic_fimp/ini: Initialization routines

37.4.1 Routine headers – fortran files

dsgivemodel_generic_fimp.f

```
************************************************************************
***  subroutine dsgivemodel_generic_fimp reads in the parameters to  ***
***  describe a generic FIMP and transfers them to common blocks.    ***
***  For the purpose of this module, a 'generic FIMP' is defined as an ***
***  effective toy model where a DM particle feably couples to one species ***
***  of SM particle, reasulting in an amplitude                      ***
***                                                                  ***
***        |M|^2 = c * (s/Lambda**2)**n                             ***
***                                                                  ***
***  for 2->2 production, where s is the CMS energy squared.         ***
***                                                                  ***
***  input:                                                          ***
***                                                                  ***
***    mgenfimp    - FIMP mass [GeV]                                 ***
***    genselfconj - true for self-conjugated DM, false for not      ***
***    PDGSM       - PDG code of SM particle                         ***
***    c           - coupling strength (dimensionless)               ***
***    Lambda      - EFT scale [GeV]                                 ***
***    n           - energy-dependence of matrix element (integer, as above)***
***                                                                  ***
*** NB: PDGSM refers to the particle code in the EW broken phase, above the ***
***     hadronization scale. For model parameters where freeze-in    ***
***     production dominantly happens outside this regime, i.e. for T>TEW ***
***     (or T<TQCD for couplings to quarks/gluons), the results of   ***
***     dsfi2to2oh2 hence must be interpreted with care.             ***
***     See the manual for more details about the actual implementation ***
***     for T>TEW and T<TCD (and/or E<Ehadron).                      ***
***                                                                  ***
***                                                                  ***
*** author: Torsten.Bringmann.fys.uio.no                             ***
*** date 2021-11-18                                                  ***
************************************************************************
      subroutine dsgivemodel_generic_fimp(mgenfimp,genselfconj,PDGSM,c,Lambda,n)
```

dsinit_module.f

```
************************************************************************
***  This is the initialization subroutine for the "generic_fimp" module ***
***  which just contains a simple FIMP model defined by the DM mass,  ***
***  a single SM state the DM particles couple to, as well as 3 parameters ***
```

```
***  describing the interaction rate (see dsgivemodel_generic_fimp).        ***
***                                                                          ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)                ***
***  Date: 2021-11-19                                                        ***
********************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
********************************************************************************
***  subroutine dsmodelsetup sets up a particle model                        ***
***                                                                          ***
***  type : interface                                                        ***
***                                                                          ***
***  desc : Sets up a new particle physics model                             ***
***                                                                          ***
***  output:                                                                 ***
***    ierr  -  0 no errors                                                  ***
***          - <0 if an error is encountered (theoretically inconsistent     ***
***               model parameters)                                          ***
***    iwarn -  0 no iwarns                                                  ***
***             1 WDM bound likely violated                                  ***
***                                                                          ***
*** author: Torsten.Bringmann.fys.uio.no                                     ***
*** date 2016-05-09                                                          ***
********************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

37.5 generic_fimp/rd: Relic density

37.5.1 Routine headers – fortran files

dsrdparticles.f

```
********************************************************************************
*** subroutine dsrdparticles returns kinematic information about the         ***
*** particles that are included in the calculation of the DM relic density   ***
*** (coannihilations, resonances  thresholds, thermal effects).              ***
***                                                                          ***
***  type : interface                                                        ***
***                                                                          ***
***  desc : Particles included in relic density calculation                  ***
***  desc : (coannihilations, resonances and thresholds)                     ***
***                                                                          ***
*** Input:                                                                   ***
***    option - 0        = default                                           ***
***             1..99    = include various subsets of coannihilations        ***
***                        (not relevant in generic FIMP module)             ***
***                                                                          ***
***    nsize - size of arrays, do not fill larger than this                  ***
***    TSM   - finite temperature of the SM heat bath                        ***
***            (if TSM=0d0, finite-T effects will be ignored)                ***
***                                                                          ***
*** Output                                                                   ***
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)       ***
***    ncoann   - number of particles coannihilating                         ***
***    mcoann   - relic and coannihilating mass [GeV]                        ***
***    dof      - internal degrees of freedom of the particles               ***
***    nrs      - number of resonances to take special care of               ***
***    rm       - mass of resonances [GeV]                                   ***
***    rw       - width of resonances [GeV]                                  ***
```

```
***    nt       - number of thresholds to take special care of             ***
***               (NB: coannihilation thresholds are automatic             ***
***                -- do not include here!)                                ***
***    tm       - sqrt(s) of the thresholds [GeV]                          ***
***                                                                        ***
*** This output is used by the RD routines in src/rd/ and src/fi           ***
***                                                                        ***
*** author: Torsten Bringmann                                             ***
*** date: 2021-11-19                                                      ***
*** mod 2021-12-20 [tb]: added TSM as argument                            ***
******************************************************************************
      subroutine dsrdparticles(option,nsize,TSM,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
```

Chapter 38

Generic WIMPs

Similar to the case of generic_decayingDM, the module generic_wimp provides a simple phenomenological template for a large class of DM candidates. Rather than being based on an actual particle physics model, it mostly serves to provide an illustration of how the functionalities of DarkSUSY can be used in phenomenological studies of 'vanilla' WIMP DM, when only providing the absolute minimum of input parameters.

38.1 Generic WIMP and simple extensions

A generic WIMP model in DarkSUSY is set up by a call to dsgivemodel_generic_wimp, and hence fully defined by the input parameters of that routine:

- The mass of the DM particle, $m_{\rm WIMP}$.
- The annihilation cross section $\langle\sigma v\rangle$.
- The PDG code of the particle (and antiparticle) that the WIMP annihilates to. It is assumed to annihilate to a two-body final state with this set of final state particles.
- The spin-independent WIMP-nucleon scattering cross section, $\sigma_{\rm SI}$. This is assumed to apply for scattering off protons and neutrons.

In addition the DM particle is in this simple setup assumed to be self-conjugate (its own antiparticle), with spin 0, with no spin-dependent scattering cross sections for WIMP-nucleon scatterings and with one internal degree of freedom. In most cases, this is what is usually called a generic WIMP.

However, at times it can be useful to extend this model to also include other properties of the DM particle. The routine dsgivemodel_generic_wimp_opt allows to set additional properties. It takes two arguments, a key to determine what property to set and a value, according to the following

Optional parameters in dsgivemodel_generic_wimp_opt.

spin	r8	Allows to set the spin of the DM particle. The internal degrees of freedom is updated accordingly.
sigsip	r8	Spin-independent WIMP-proton scattering cross section.
sigsin	r8	Spin-independent WIMP-neutron scattering cross section.
sigsdp	r8	Spin-dependent WIMP-proton scattering cross section.
sigsdn	r8	Spin-dependent WIMP-neutron scattering cross section.

This simple setup allows to use essentially the full functionality of the core library, from relic density calculation to direct detection rates and cosmic ray propagation and indirect detection signals in general. If needed, it is relatively simple to extend the generic WIMP model if you wish

to explore other similar models. Please use the script **scr/make_module.pl** to create a copy of the generic WIMP module and make your changes there.

38.2 Asymmetric dark matter

While a call to dsgivemodel_generic_wimp (followed by dsmodelsetup) always initializes a DM particle that is treated as its own antiparticle, the generic_wimp model also provides the possibility of initializing a DM particle that is *not* self-conjugate. This is done by a call to dsgivemodel_generic_wimp instead, which takes the primordial DM asymmetry, $(n_+ - n_-)/s > 0$, as additional input parameter $\eta_{\rm model}$.

The main effect of $\eta_{\rm model}$ appears in the source term entering in indirect detection yields, provided by the interface function dscrsource. Concretely, the source term will be reduced by a factor of

$$\xi = \frac{1}{2} r(2-r) \tag{38.1}$$

compared to the case of self-conjugate DM, see also Eq. (14.3). Here, r describes the ratio of the symmetric to the total DM component as given by Eq. (27.74):

$$r = \frac{2n_-}{n_+ + n_-} = 1 - \eta_{\rm model} \frac{c}{\Omega_{\rm DM,obs} h^2}\,, \tag{38.2}$$

with $c = 2.755 \times 10^{10}\,(m_{\rm DM}/100\,{\rm GeV})$ and $\Omega_{\rm DM,obs}h^2 \approx 0.112$ being the total cosmological DM density (stored in the common block variable omegacdmh2).

The value of r given in Eq. (38.2) assumes that all of the observed DM is made up of particles with the properties given in this specific model. If, instead, this only provides a subdominant DM fraction, or if the value of r following from the relic density calculation, cf. section 27.3, is inconsistent with the value obtained by Eq. (38.2), the default rescaling given by Eq. (38.1) may not be appropriate. For that purpose, we provide the auxiliary function dscrrescale_asym, see section 12.1, to help users rescale the default output of indirect detection routines as intended.

38.3 generic_wimp/an: Annihilation routines

38.3.1 Routine headers – fortran files

dsanwx.f

```
**************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.    ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  Input:                                                                 ***
***    p - initial cm momentum (real) for DM annihilations                  ***
***  Output:                                                                ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  passed to dsrdens by dsrdomega.                                        ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2015-06-11                                                         ***
*** mod  2020-04-12 TB changed to Møller velocity as reference              ***
*******************************************************************************
      real*8 function dsanwx(p)
```

dssigmav0tot.f

```
*****************************************************************
*** function dssigmav0tot returns the *total* annihilation cross section
*** sigma v at p=0 for WIMP-WIMP annihilation.
*** This is obtained by summing over all implemented 2-body channels
*** (as returned by dssigmav) plus contributions from final states with
*** more particles.
***
***  type : interface
***
*** Units of returned cross section: cm^3 s^-1
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2014-11-14
*****************************************************************

      real*8 function dssigmav0tot()
```

38.4 generic_wimp/cr: Cosmic rays

38.4.1 Routine headers – fortran files

dscrsource.f

```
*******************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2015-06-11
*** modified TB 2019-12-13: checked FSR contributions
*******************************************************************************
```

```
      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2015-06-21
************************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

38.5 generic_wimp/dd: Direct detection

38.5.1 Routine headers – fortran files

dsddgpgn.f

```
************************************************************************
***  subroutine dsddgpgn returns DM nucleon four-fermion couplings          ***
***                                                                         ***
***  type : interface                                                       ***
***         [NB: this routine will eventually be phased out as *interface*  ***
***              routine. Currently, neutrino telescope routines are the    ***
***              only routines in /src that access it. ]                    ***
***                                                                         ***
***   output:                                                               ***
***     gg complex*16 array of couplings                                     ***
***          first index is operator index (1:ddng), with ddng in dsddcom.h ***
***          second index is proton/neutron (1:2)                           ***
***     units: GeV^-2                                                       ***
***                                                                         ***
***  author: paolo gondolo (paolo@physics.utah.edu) 2004                    ***
***    2008-02-16 paolo gondolo double spin-dependent squark contribution   ***
***    2008-02-16 paolo gondolo undo running quark masses in couplings      ***
***    2016-05-19 Paolo Gondolo, Joakim Edsjo, scheme setup                 ***
***    2016-11-20 Paolo Gondolo abandoned scheme                            ***
```

```
***    2018-10-24 Torsten Bringmann, nucleon x-sections now in common block ***
************************************************************************
      subroutine dsddgpgn(gg,ierr)
```

38.6 generic_wimp/ge: General routines

38.6.1 Routine headers – fortran files

dsdmspin.f

```
************************************************************************
*** Function dsdmspin returns the dark matter spin (in hbar)          ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
***  desc : dark matter spin                                          ***
***                                                                   ***
*** author: Paolo Gondolo 2016-11-20                                  ***
************************************************************************

      real*8 function dsdmspin()
```

dsmwimp.f

```
************************************************************************
*** Function dsmwimp returns the WIMP mass                            ***
***                                                                   ***
***  type : interface                                                 ***
***                                                                   ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                  ***
************************************************************************

      real*8 function dsmwimp()
```

38.7 generic_wimp/ini: Initialization routines

38.7.1 Routine headers – fortran files

dsgivemodel_generic_wimp.f

```
************************************************************************
***  subroutine dsgivemodel_generic_wimp reads in the parameters to   ***
***  describe a generic WIMP and transfers them to common blocks      ***
***                                                                   ***
***  input:                                                           ***
***                                                                   ***
***    mgenwimp - WIMP mass (in GeV)                                  ***
***    svann    - constant annihilation cross section (in cm^3/s)     ***
***    pdgann   - PDG code for particle in dominant annihilation channel ***
***               (e.g. 5 for bbar, 24 for W^+W^-)                    ***
***    SI       - cross section for spin-indep. WIMP-nucleon scattering (pb)***
***                                                                   ***
***   Note that further model details can be added / changed by a     ***
***   subsequent call to dsgivemodel_generic_wimp_opt!                ***
***                                                                   ***
***   NB: From DS version 6.3.2, the DM particle initialized with this ***
```

```
***       function is always its own anti-particle. See                    ***
***       dsgivemodel_generic_wimp_aDM for the more general case where this ***
***       is not the case.                                                 ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2014-06-20                                                        ***
*** mod  2022-10-18 (enforced self-conjugate DM)                           ***
******************************************************************************
      subroutine dsgivemodel_generic_wimp(mgenwimp,svann,pdgann,SI)
```

dsgivemodel_generic_wimp_aDM.f

```
******************************************************************************
***  subroutine dsgivemodel_generic_wimp_aDM reads in the parameters to     ***
***  describe a generic WIMP that is NOT its own antiparticle and transfers ***
***  them to common blocks                                                 ***
***                                                                        ***
***  input:                                                                ***
***                                                                        ***
***    mgenwimp   - WIMP mass (in GeV)                                     ***
***    etagenwimp - DM asymmetry = (n+ - n-)/s > 0                         ***
***                 (Note the conventional assumption about the sign!)     ***
***    svann      - constant annihilation cross section (in cm^3/s)        ***
***    pdgann     - PDG code for particle in dominant annihilation channel ***
***                 (e.g. 5 for bbar, 24 for W^+W^-)                       ***
***    SI         - cross section for spin-independent WIMP-nucleon        ***
***                 scattering (pb)                                        ***
***                                                                        ***
***   Note that further model details can be added / changed by a          ***
***   subsequent call to dsgivemodel_generic_wimp_opt!                     ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2022-10-18                                                        ***
******************************************************************************
      subroutine dsgivemodel_generic_wimp_aDM(mgenwimp,etagenwimp,svann,pdgann,SI)
```

dsgivemodel_generic_wimp_opt.f

```
******************************************************************************
***  subroutine dsgivemodel_generic_wimp_opt sets optional inputs not      ***
***  covered by the default set of input parameters given in               ***
***  dsgivemodel_generic_wimp (and overrides those specified there).       ***
***                                                                        ***
***  input:                                                                ***
***                                                                        ***
***    key   - character string specifying input to be provided            ***
***    value - input value assigned (real)                                 ***
***                                                                        ***
***   This function should be called *after* dsgivemodel_generic_wimp (and) ***
***   before dsmodelsetup). For details about which values of "key" can be ***
***   provided, see below.                                                 ***
***                                                                        ***
***                                                                        ***
*** author: Torsten Bringmann, torsten.bringmann@fys.uio.no                ***
*** date: 2018-10-24                                                       ***
******************************************************************************
      subroutine dsgivemodel_generic_wimp_opt(key,value)
```

dsinit_module.f

```
******************************************************************************
***  This is the initialization subroutine for the "generic_wimp" module   ***
***  which just contains a simple WIMP model with a mass, annihilation     ***
```

```
***  branching fractions, cross sections etc.                              ***
***                                                                        ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)              ***
***  Date: 21/06/2016                                                      ***
******************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
******************************************************************************
***  subroutine dsmodelsetup sets up a particle model                      ***
***                                                                        ***
***  type : interface                                                      ***
***                                                                        ***
***  desc : Sets up a new particle physics model                           ***
***                                                                        ***
***  output:                                                               ***
***    ierr  -  0 no errors                                                ***
***          - <0 if an error is encountered (theoretically inconsistent   ***
***               model parameters)                                        ***
***    iwarn -  0 no warnings                                              ***
***             1 spin-dependent couplings reset to zero                   ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2016-05-09                                                        ***
***  mod 2021-10-18 (added option for asymmetric component)                ***
******************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

38.8 generic_wimp/kd: Kinetic decoupling

38.8.1 Routine headers – fortran files

dskdm2.f

```
**********************************************************************
*** dskdm2 returns the full scattering amplitude for DM at zero
*** momentum transfer squared in the generic WIMP model, SUMMED over both initial
*** and final spin and other internal states, and then divided by the DM
*** internal degrees of freedom.
***
***  type : INTERFACE
***  desc : Full scattering amplitude squared, averaged over momentum transfer
***
***  input: omega -- CMS MOMENTUM of scattering partner
***  output: omega -- CMS ENERGY of scattering partner
***
***  input:   SMtype - SM scattering partners
***             "    = 7,8,9 - d,u,s quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***                  = 13    - photons
***
*** author: torsten.bringmann@fys.uio.no, 2015-06-22
*** updated 2018-05-14 : moved momentum->energy conversion to src_models/
*** updated 2019-12-13 : added photon scattering
**********************************************************************

      real*8 function dskdm2(omega,SMtype)
```

dskdm2simp.f

```
************************************************************************
*** dskdm2simp returns the scattering amplitude for DM at zero
*** momentum transfer squared in the geenric WIMP model, SUMMED over both initial
*** and final spin and other internal states, and then divided by the DM
*** internal degrees of freedom. This is only valid in the limit of
*** relativistic scattering partners with small energies omega, where we
*** can expand as
***
***    |M|**2 = cn*(omega/m0)**n + O( (omega/m0)**(n+1) )
***
***  type : INTERFACE
***  desc : Scattering amplitude squared, expanded in powers of energy
***
***  input: SMtype   - SM scattering partners:
***                    7,8,9 - u,d,s quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten.bringmann@fys.uio.no, 2015-06-22
*** updated 2019-12-13 : added photon scattering
***         2020-05-27 : corrected spin- and symmetry factors
************************************************************************

      subroutine dskdm2simp(SMtype,cn,n)
```

dskdparticles.f

```
************************************************************************************
*** Prepares integration of Boltzmann equation and has to be called         ***
*** before dskdboltz (called by dskdtkd)                                    ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date  : 2015-06-11                                                      ***
************************************************************************************
      subroutine dskdparticles()
```

38.9 generic_wimp/rd: Relic density

38.9.1 Routine headers – fortran files

dsrdparticles.f

```
      subroutine dsrdparticles(option,nsize,TSM,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
************************************************************************
*** subroutine dsrdparticles returns which particles are included in
***   the calculation of the WIMP relic density (coannihilations,
***   resonances, thresholds)
***
***  type : interface
***
***  desc : Particles included in relic density calculation
***  desc : (coannihilations, resonances and thresholds)
***
*** input:
***   option - 0 = default
***            !=0 - include various subsets of coannihilations
```

```
***   nsize  - size of arrays, do not fill larger than this
***   TSM    - finite temperature of the SM heat bath
***            (not supported in module generic_wimp)
***
*** output
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)
***    ncoann   - number of particles coannihilating
***    mcoann   - relic and coannihilating mass in gev
***    dof      - internal degrees of freedom of the particles
***    nrs      - number of resonances to take special care of
***    rm       - mass of resonances in gev
***    rw       - width of resonances in gev
***    nt       - number of thresholds to take special care of
***               do not include coannihilation thresholds (that's automatic)
***    tm       - sqrt(s) of the thresholds in gev
*** This output is used by the RD routines in src/rd/
*** author: paolo gondolo
*** derived from dsrdomega
*** date: 13-10-04
*** Modified: Joakim Edsjo, edsjo@fysik.su.se, 2015-12-08
*** Modified: Torsten Bringmann 2016-06-28
*** Modified: Torsten Bringmann 2018-02-28 (added selfcon)
*** mod 2021-12-20 [tb]: added TSM as argument
**********************************************************************
```

38.10 generic_wimp/se_yield: Yields from annihilation in the Sun/Earth

38.10.1 Routine headers – fortran files

dsseyield.f

```
**********************************************************************
*** Function dsseyield calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth summing
*** over all annihilation channels.
***
***  type : interface
***
*** Inputs:
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
```

```
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1.
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
***
*** Author: Torsten.Bringmann@fys.uio.no
*** Date: 22/06/2015
**********************************************************************

      real*8 function dsseyield(e,theta,wh,kind,type,istat)
```

Chapter 39

The Minimal Supersymmetric Standard Model

The implementation of the minimal supersymmetric standard model (MSSM) in the module mssm mainly follows that of DarkSUSYversions 4 [1] (see also Section 39.16 in this manual). In particular, the conventions for the superpotential and soft supersymmetry-breaking potential are the same as implemented in [25], and thus similar to [159, 160]. The full set of input parameters to be provided at the weak scale thus consists of the pseudoscalar mass (m_A), the ratio of Higgs vacuum expectation values ($\tan\beta$), the Higgsino (μ) and gaugino (M_1, M_2, M_3) mass parameters, trilinear couplings (A_{Eaa}, A_{Uaa}, A_{Daa}, with $a = 1, 2, 3$) as well as soft sfermion masses (M^2_{Qaa}, M^2_{Laa}, M^2_{Uaa}, M^2_{Daa}, M^2_{Eaa}, with $a = 1, 2, 3$).* Internally, those values are stored in mssm common blocks. The user may either provide them directly or by setting up pre-defined phenomenological MSSM models with a reduced number of parameters through a call to a routine like dsgive_model or dsgive_model25 (followed by a call to dsmodelsetup). The former sets up the simplest of those models, defined by the input parameters μ, M_2, m_A, $\tan\beta$, a common scalar mass m_0, and trilinear parameters A_t and A_b; M_1 and M_3 are then calculated by assuming the GUT condition, and the remaining MSSM parameters are given by $\mathbf{M}_Q = \mathbf{M}_U = \mathbf{M}_D = \mathbf{M}_E = \mathbf{M}_L = m_0\mathbf{1}$, $\mathbf{A}_U = \mathrm{diag}(0, 0, A_t)$, $\mathbf{A}_D = \mathrm{diag}(0, 0, A_b)$, $\mathbf{A}_E = \mathbf{0}$. Similarly, dsgive_model25 sets up a pMSSM model with 25 free parameters (see the header of that file for details). Alternatively, all those values can be set by reading in an SLHA file, or providing GUT scale parameters in the case of cMSSM models (via an interface to the ISASUGRA code, as included in ISAJET [161, 162]).

Compared to previous versions of the code, DarkSUSY 6 has a new interface to read and write SUSY Les Houches Accord (SLHA) files [163, 164].

All particle and sparticle masses are stored in a common block array mass(). For neutralino masses, we include the leading loop corrections [165, 166, 167] but neglect the relatively small corrections for charginos [165] (in both cases, masses cannot be negative in our convention).† Likewise, all mixing matrices and decay widths are available as common block arrays. The latter are currently only computed for the Higgs particles (via an interface to the FeynHiggs [168, 169, 170, 171, 172] package), while the other sparticles have fictitious widths of 0.5% of the sparticle mass (for the sole purpose of regularizing annihilation amplitudes close to poles). Again, the conventions for masses and mixings follow exactly those of Ref. [1], to which we refer for further details.

*Note that currently only diagonal matrices are allowed. While not being the most general ansatz possible, this implies the absence of flavour changing neutral currents at tree-level in all sectors of the model.

†Unless, of course, those values are provided by an SLHA file. This comment also applies to the following simplifications concerning both sparticle masses and widths.

39.1 mssm/ac: Accelerator constraints

39.1.1 Accelerator bounds

DarkSUSY contains a set of routines for a rough check whether a given model is excluded by accelerator constraints. These routines are called **dsacbnd[number]**. The policy is that when we update DarkSUSY with new accelerator constraints, we keep the old routine, and add a new routine with the last number incremented by one. Which routine that is called is determined by calling **dsacset** with a tag determining which routine to call. To check the accelerator constraints, then call **dsacbnd** which calls the right routine for you. Upon return, **dsacbnd** returns an exclusion flag, warning. If zero, the model is OK, if non-zero, the model is likely excluded. The cause for the exclusion is coded in the bits of excl according to table 39.1.

We stress that these bounds are in most cases only *approximate* limits: DarkSUSY generally focusses on theoretical predictions for such observables, given a DM model realization, rather than on the implementation of experimental likelihoods and the possibility to derive statistically well-defined limits from those. For the latter, we instead refer to packages like DarkBit [173] (or ColliderBit [174] for accelerator-based constraints)

	excl		
Bit set	Octal value	Decimal value	Reason for exclusion
0	1	1	Chargino mass
1	2	2	Gluino mass
2	4	4	Squark mass
3	10	8	Slepton mass
4	20	16	Invisible Z width
5	40	32	Higgs mass in excluded region
6	100	64	Neutralino mass
7	200	128	$b \to s\gamma$
8	400	256	ρ parameter
9	1000	512	$(g-2)_\mu$
10	2000	1024	$B_s \to \mu^+\mu^-$
11	4000	2048	squark-gluino
12	10000	4096	Higgs mass does not fit observed Higgs

Table 39.1: The bits of excl are set to indicate by which process this particular model is excluded. Check if a bit is set with btest(excl,bit).

Compared to previous DarkSUSY versions, we use in particular updated limits from HiggsBounds [175] on the mass of the MSSM Higgs bosons, as well as approximate bounds on squark and gluino masses from LHC 8 TeV data [176]. For $b \to s\gamma$, we keep our genuine routines in mssm/ac_bsg for this rare decay (see Ref. [1] for a more detailed description) but now use as a default the result from SuperIso [177]; we compare this to the current limit of $\mathcal{B}(B \to X_s\gamma) = (3.27 \pm 0.14) \times 10^{-4}$ as adopted in FlavBit [178], based on data from BarBar and Belle [179, 180, 181]. SuperIso also computes the rate for the rare leptonic decay $B_s^0 \to \mu^+\mu^-$, which we compare to the LHCb measurement of $\mathcal{B}(B_s^0 \to \mu^+\mu^-) = (3.0 \pm 0.6^{+0.3}_{-0.2}) \times 10^{-9}$ [182]. Finally, $a_\mu \equiv (g-2)_\mu/2$ is calculated by dsgm2muon, based on [183]; in dsacbnd, this is compared to the observed valued of $a_{\mu,\,\mathrm{obs}} = (11659208.9 \pm 6.3) \times 10^{-10}$ [184] after subtracting the SM expectation as specified in PrecisionBit [185].

39.1.2 Routine headers – fortran files

dsacbnd.f

```
************************************************************************
***  subroutine dsacbnd checks collider bounds                       ***
***                                                                  ***
***                                                                  ***
************************************************************************
      subroutine dsacbnd(warning)
```

dsacbnd11.f

```
      subroutine dsacbnd11(excl)
c-----------------------------------------------------------------------
c  check if accelerator data exclude the present point.
c  output:
c    excl - code of the reason for exclusion (integer); 0 if allowed
c    if not allowed, the reasons are coded as follows
c      bit set   dec.   oct.   reason
c      -------   ----   ----   ------
c            0      1      1   chargino mass
c            1      2      2   gluino mass
c            2      4      4   squark mass
c            3      8     10   slepton mass
c            4     16     20   invisible z width
c            5     32     40   higgs mass
c            6     64    100   neutralino mass
c            7    128    200   b -> s gamma
c            8    256    400   rho parameter
c            9    512   1000   (g-2)_mu
c           10   1024   2000   B_s -> mu+ mu-
c  author: paolo gondolo 1994-1999
c  history:
c     940407 first version paolo gondolo
c     950323 update paolo gondolo
c     971200 partial update joakim edsjo
c     980428 update joakim edsjo
c     990719 update paolo gondolo
c     000310 update piero ullio
c     000424 added delrho joakim edsjo
c     000904 update according to pdg2000 lars bergstrom
c     010214 mh2 limits corrected, joakim edsjo
c     020927 higgs limits update according to pdg2002 mia schelke
c     021001 susy part. mass limits update to pdg2002 je/ms
c     031204 standard model higgs like mh2 limit for msugra models
c     070529 calling new bsg routine+new experim bsg value, ms
c     090105 corrected do-loop for mh3 mass bounds
c     090316 higgs limits by HiggsBounds, erik lundstrom
c     110905 new higgs limits with HiggsBounds, paolo gondolo
c     130412 Added B_s -> mu+ mu- (calculated with SuperIso)
c=======================================================================
```

dsacbnd12.f

```
      subroutine dsacbnd12(excl)
c-----------------------------------------------------------------------
c  check if accelerator data exclude the present point.
c  This routine uses simplified models and approximate constraints.
c  output:
c    excl - code of the reason for exclusion (integer); 0 if allowed
c    if not allowed, the reasons are coded as follows
c      bit set   dec.   oct.   reason
c      -------   ----   ----   ------
```

```
c            0      1      1   chargino mass
c            1      2      2   gluino mass
c            2      4      4   squark mass
c            3      8     10   slepton mass
c            4     16     20   invisible z width
c            5     32     40   higgs mass in excluded region
c            6     64    100   neutralino mass
c            7    128    200   b -> s gamma
c            8    256    400   rho parameter
c            9    512   1000   (g-2)_mu
c           10   1024   2000   B_s -> mu+ mu-
c           11   2048   4000   squark-gluino
c           12   4096  10000   Higgs mass does not fit observed Higgs
c  author: paolo gondolo 1994-1999
c  history:
c     940407 first version paolo gondolo
c     950323 update paolo gondolo
c     971200 partial update joakim edsjo
c     980428 update joakim edsjo
c     990719 update paolo gondolo
c     000310 update piero ullio
c     000424 added delrho joakim edsjo
c     000904 update according to pdg2000 lars bergstrom
c     010214 mh2 limits corrected, joakim edsjo
c     020927 higgs limits update according to pdg2002 mia schelke
c     021001 susy part. mass limits update to pdg2002 je/ms
c     031204 standard model higgs like mh2 limit for msugra models
c     070529 calling new bsg routine+new experim bsg value, ms
c     090105 corrected do-loop for mh3 mass bounds
c     090316 higgs limits by HiggsBounds, erik lundstrom
c     110905 new higgs limits with HiggsBounds, paolo gondolo
c     130412 Added B_s -> mu+ mu- (calculated with SuperIso), Joakim Edsjo
c     160209 Added LHC 8 TeV mgluino msquark bounds, lars bergstrom
c=======================================================================
```

dsacbnd13.f

```
      subroutine dsacbnd13(warning)
c-----------------------------------------------------------------------
c  check if accelerator data warningude the present point.
c  This routine uses simplified models and approximate constraints.
c  output:
c    warning - code of the reason for warningusion (integer); 0 if allowed
c    if not allowed, the reasons are coded as follows
c      bit set   dec.   oct.   reason
c      -------   ----   ----   ------
c            0      1      1   chargino mass
c            1      2      2   gluino mass
c            2      4      4   squark mass
c            3      8     10   slepton mass
c            4     16     20   invisible z width
c            5     32     40   higgs mass in excluded region
c            6     64    100   neutralino mass
c            7    128    200   b -> s gamma
c            8    256    400   rho parameter
c            9    512   1000   (g-2)_mu
c           10   1024   2000   B_s -> mu+ mu-
c           11   2048   4000   squark-gluino
c           12   4096  10000   Higgs mass does not fit observed Higgs
c  author: paolo gondolo 1994-1999
c  history:
c     940407 first version paolo gondolo
c     950323 update paolo gondolo
c     971200 partial update joakim edsjo
c     980428 update joakim edsjo
```

```
c     990719 update paolo gondolo
c     000310 update piero ullio
c     000424 added delrho joakim edsjo
c     000904 update according to pdg2000 lars bergstrom
c     010214 mh2 limits corrected, joakim edsjo
c     020927 higgs limits update according to pdg2002 mia schelke
c     021001 susy part. mass limits update to pdg2002 je/ms
c     031204 standard model higgs like mh2 limit for msugra models
c     070529 calling new bsg routine+new experim bsg value, ms
c     090105 corrected do-loop for mh3 mass bounds
c     090316 higgs limits by HiggsBounds, erik lundstrom
c     110905 new higgs limits with HiggsBounds, paolo gondolo
c     130412 Added B_s -> mu+ mu- (calculated with SuperIso), Joakim Edsjo
c     160209 Added LHC 8 TeV mgluino msquark bounds, lars bergstrom
c     171212 Updated b -> s\gamma, B_s -> mu+ mu-, (g-2), torsten bringmann
c=======================================================================
```

dsacset_mssm.f

```
**********************************************************************
*** This routine selects which set of accelerator constraints to use
*** when dsacbnd is called. For available options, see dsacbnd.f
**********************************************************************
      subroutine dsacset_mssm(a)
```

dsbsgamma.f

```
      subroutine dsbsgamma(ratio,flag)
c-----------------------------------------------------------------------
c  b -> s + gamma branching ratio
c  input:
c    flag : 0 no qcd correction -- 1 qcd corrections
c  output:
c    ratio : branching ratio
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995
c     28-nov-94 formulas in bertolini et al, nucl phys b353 (1991) 591
c     modified: joakim edsjo, 2000-09-03, vertices from dsvertx.f
c     correctly implemented
c=======================================================================
```

dsbsgf1.f

```
      function dsbsgf1(x)
c-----------------------------------------------------------------------
c  function in bertolini et al, nucl phys b353 (1991) 591
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dsbsgf2.f

```
      function dsbsgf2(x)
c-----------------------------------------------------------------------
c  function in bertolini et al, nucl phys b353 (1991) 591
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dsbsgf3.f

```
      function dsbsgf3(x)
c-----------------------------------------------------------------------
c  function in bertolini et al, nucl phys b353 (1991) 591
```

```
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c======================================================================
```

dsbsgf4.f

```
      function dsbsgf4(x)
c----------------------------------------------------------------------
c  function in bertolini et al, nucl phys b353 (1991) 591
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c======================================================================
```

dsflavour.f

```
      subroutine dsflavour(brbsg,delta0,brbmumu,brbmumuuntag,
     &   brbtaunu,brbdtaunu,amu,rmu23,iflag)
c----------------------------------------------------------------------
c  Routine that calls SuperIso and calculates various flavour
c  observables. More are easily added here.
c  The interface goes via a SLHA file. For SuperIso to be able to read it
c  we need to set it to no-CP-violation and MFV.
c  output:
c    brbsg:   Br(b -> s gamma)
c    brbmumu: Br(B_s -> mu+ mu-)
c    iflag: flag that if non-zero indicates a problem
c  author: Joakim Edsjo, edsjo@fysik.su.se
c  date: 2013-05-01
c======================================================================
```

dsgm2muon.f

```
      function dsgm2muon()
No header found.
```

dswexcl.f

```
      subroutine dswexcl(unit,excl)
c----------------------------------------------------------------------
c  write reasons for exclusion to specified unit.
c  input:
c    unit - logical unit to write on (integer)
c    excl - code of the reason for exclusion (integer); 0 if allowed
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c======================================================================
```

39.2 mssm/ac_bsg: src_models/mssm/ac_bsg

39.2.1 Routine headers – fortran files

dsbsg2007full.f

```
      subroutine dsbsg2007full(ratio,flag)

**********************************************************************
* Routine that calculates the b-->s+gamma branching rate             *
* This routine was written after a big shift in the theoretical      *
* calculation of the SM Branching ratio in 2006.                     *
* From 3.7e-4 to 3.15e-4.                                            *
* This routine implements an approximate solution which works when   *
```

```
* there are only small Beyond-SM (BSM) corrections                    *
* For the implemented solution we refer to:                           *
*   M. Misiak et al: hep-ph/0609232                                   *
* + M. Misiak and M. Steinhauser: hep-ph/0609241                      *
* + M. Misiak and M. Steinhauser, private communication               *
* + P. Gambino, private communication.                                *
* (+ref for the SUSY contributions, see the individual files)         *
* The implemented solution is:                                        *
* BR * 10^4  = (3.15 +_ 0.23) - 8.0 * deltaBSM[C7]                    *
*              - 1.9 * deltaBSM[C8] + O(deltaBSM^2)                   *
* with a matching scale m_0 = 2 m_W = 160 GeV                         *
* for other parameters see hep-ph/0609241 appendix A.                 *
* but we use the old matching scale mu_0 = m_t(m_t) = 166.6 GeV       *
* Input: flag:  0 = only standard model                               *
*               1 = standard model plus SUSY corrections              *
* Output: ratio = BR(b -> s gamma)                                    *
* author:Mia Schelke, schelke@physto.se, 2007                         *
**********************************************************************
```

dsbsgalpha3.f

```
      function dsbsgalpha3(m)

**********************************************************************
* The coupling constant alpha_3 evaluated at the scale m              *
* using nf effective quark flavours (usually taken to be nf=5)        *
* Uses eq. (42) of Ciuchini et al. hep-ph/9710335                     *
* for the calculation of b --> s gamma                                *
* Note: This routines is strictly speaking only valid for mass scales *
* between mb and mt where nf=5 should be used.                        *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                   *
**********************************************************************
```

dsbsgalpha3int.f

```
      function dsbsgalpha3int(al,mstart,m,nf)

**********************************************************************
* The coupling constant alpha_3 evaluated at the scale m              *
* given the value at a given scale mstart. nf effective quark flavours*
* are used in the running (if nf=7, 6 quark flavours and one squark   *
* flavor are used)                                                    *
* Uses eq. (42) of Ciuchini et al. hep-ph/9710335                     *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                   *
**********************************************************************
```

dsbsgammafull.f

```
**********************************************************************
*** Wrapper routine kept for backwards compatibility. This routine
*** calls the latest expressions for b->s gamma
**********************************************************************
      subroutine dsbsgammafull(ratio,flag)
```

dsbsgat0.f

```
      function dsbsgat0(x,flag)

**********************************************************************
```

```
* Function A^t_0(x) in (2.7) of Gambino and Misiak,                  *
* Nucl. Phys. B611 (2001) 338                                        *
* x must be a positive number                                        *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-03-12                  *
* updated by Mia Schelke 2003-04-10 to include the susy contribution  *
* as explained in eq. (5.1)                                          *
* (the references used for the susy contributions can be found in     *
*  the different fortran codes)                                      *
* Input: flag:  0 = only standard model                              *
*               1 = standard model plus SUSY corrections             *
**********************************************************************
```

dsbsgat1.f

```
      function dsbsgat1(x,flag)

**********************************************************************
* Function A^t_1(x) p. 11 in Gambino and Misiak,                     *
* Nucl. Phys. B611 (2001) 338                                        *
* x must be a positive number                                        *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-03-13                  *
* updated by Mia Schelke 2003-04-10 to include the susy contribution  *
* as explained in eq. (5.1)                                          *
* (the references used for the susy contributions can be found in     *
*  the different fortran codes)                                      *
* Input: flag:  0 = only standard model                              *
*               1 = standard model plus SUSY corrections             *
**********************************************************************
```

dsbsgbofe.f

```
      function dsbsgbofe(flag)
c      program dsacbofe
**********************************************************************
* Program that calculates B(E_0) in (E.1) of Gambino and Misiak,      *
* Nucl. Phys. B611 (2001) 338                                        *
* for the calculation of b --> s gamma                               *
* Input: flag:  0 = only standard model                              *
*               1 = standard model plus SUSY corrections             *
* author:Mia Schelke, schelke@physto.se, 2003-03-12                  *
**********************************************************************
```

dsbsgc41h2.f

```
      function dsbsgc41h2()

**********************************************************************
* The next to leading order contribution to the Wilson coefficient C_4*
* from the two-Higgs doublet model                                   *
* Eq. (58) of Ciuchini et al.,                                       *
* hep-ph/9710335                                                     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                  *
**********************************************************************
```

dsbsgc41susy.f

```
      function dsbsgc41susy()

************************************************************************
* The next to leading order contribution to                            *
* the Wilson coefficient C_4 from susy                                 *
* Eq (10) of Ciuchini et al.,                                          *
* hep-ph/9806308                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgc70h2.f

```
      function dsbsgc70h2()

************************************************************************
* The leading order contribution to the Wilson coefficient C_7        *
* from the two-Higgs doublet model                                     *
* Eq. (53) of Ciuchini et al.,                                         *
* hep-ph/9710335                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgc70susy.f

```
      function dsbsgc70susy()

************************************************************************
* The leading order contribution to the Wilson coefficient C_7        *
* from susy                                                            *
* Eq (31) of Degrassi et al.,                                          *
* hep-ph/0009337                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-04                    *
************************************************************************
```

dsbsgc71chisusy.f

```
      function dsbsgc71chisusy()

************************************************************************
* The next to leading order contribution to                            *
* the Wilson coefficient C_7 from chargino (susy)                      *
* Eq (13) of Ciuchini et al.,                                          *
* hep-ph/9806308                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgc71h2.f

```
      function dsbsgc71h2()  ! (muw)

************************************************************************
* The next to leading order contribution to the Wilson coefficient C_7*
```

```
* from the two-Higgs doublet model                                    *
* Eq. (59) of Ciuchini et al.,                                        *
* hep-ph/9710335                                                      *
* for the calculation of b --> s gamma                                *
* The input parameter muw=\mu_W is the matching scale                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                   *
**********************************************************************
```

dsbsgc71phi1susy.f

```
      function dsbsgc71phi1susy()

**********************************************************************
* The next to leading order contribution to                          *
* the Wilson coefficient C_7 from the susy renormalization effect in  *
* the charged scalar,\phi_1, coupling                                 *
* Eq (25) of Ciuchini et al.,                                         *
* hep-ph/9806308                                                      *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgc71phi2susy.f

```
      function dsbsgc71phi2susy()

**********************************************************************
* The next to leading order contribution to                          *
* the Wilson coefficient C_7 from the susy renormalization effect in  *
* the unphysical charged scalar,\phi_2, coupling                      *
* Eq (26) of Ciuchini et al.,                                         *
* hep-ph/9806308                                                      *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgc71wsusy.f

```
      function dsbsgc71wsusy()

**********************************************************************
* The next to leading order contribution to                          *
* the Wilson coefficient C_7 from the susy renormalization effect in  *
* the W coupling                                                      *
* Eq (23) of Ciuchini et al.,                                         *
* hep-ph/9806308                                                      *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgc80h2.f

```
      function dsbsgc80h2()

**********************************************************************
* The leading order contribution to the Wilson coefficient C_8       *
* from the two-Higgs doublet model                                    *
* Eq. (53) of Ciuchini et al.,                                        *
```

```
* hep-ph/9710335                                                     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                  *
**********************************************************************
```

dsbsgc80susy.f

```
      function dsbsgc80susy()

**********************************************************************
* The leading order contribution to the Wilson coefficient C_8       *
* from susy                                                          *
* Eq (31) of Degrassi et al., hep-ph/0009337                         *
* Differs from dsbsgc70susy only by a few changes described p.11     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-04                  *
**********************************************************************
```

dsbsgc81chisusy.f

```
      function dsbsgc81chisusy()

**********************************************************************
* The next to leading order contribution to                          *
* the Wilson coefficient C_8 from chargino (susy)                    *
* Eq (14) of Ciuchini et al.,                                        *
* hep-ph/9806308                                                     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                  *
**********************************************************************
```

dsbsgc81h2.f

```
      function dsbsgc81h2()  ! (muw)

**********************************************************************
* The next to leading order contribution to the Wilson coefficient C_8*
* from the two-Higgs doublet model                                   *
* Eq. (59) of Ciuchini et al.,                                       *
* hep-ph/9710335                                                     *
* for the calculation of b --> s gamma                               *
* The input parameter muw=\mu_W is the matching scale                *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                  *
**********************************************************************
```

dsbsgc81phi1susy.f

```
      function dsbsgc81phi1susy()

**********************************************************************
* The next to leading order contribution to                          *
* the Wilson coefficient C_8 from the susy renormalization effect in  *
* the charged scalar,\phi_1, coupling                                *
* Eq (25) of Ciuchini et al.,                                        *
* hep-ph/9806308                                                     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                  *
**********************************************************************
```

dsbsgc81phi2susy.f

```
      function dsbsgc81phi2susy()

************************************************************************
* The next to leading order contribution to                            *
* the Wilson coefficient C_8 from the susy renormalization effect in   *
* the unphysical charged scalar,\phi_2, coupling                       *
* Eq (26) of Ciuchini et al.,                                          *
* hep-ph/9806308                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgc81wsusy.f

```
      function dsbsgc81wsusy()

************************************************************************
* The next to leading order contribution to                            *
* the Wilson coefficient C_8 from the susy renormalization effect in   *
* the W coupling                                                       *
* Eq (24) of Ciuchini et al.,                                          *
* hep-ph/9806308                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgckm.f

```
      function dsbsgckm()

************************************************************************
* The ratio,|V_ts^* V_tb/V_cb|^2, of ckm elements                      *
* with susy corrections                                                *
* Eq (34) of Ciuchini et al., hep-ph/9806308                           *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-22                    *
************************************************************************
```

dsbsgd1td.f

```
      function dsbsgd1td(x1)

************************************************************************
* Function \Delta^(1)_{t,d}(x_1) in app A p. 17 of                     *
* Ciuchini et al., hep-ph/9806308                                      *
* x1 must be a positive number                                         *
* x1=m^2(sq_1 of flavour d)/m^2(kgluin)                                *
* Note that this function can also be used for \Delta^(1)_d(x_2)       *
* but in that case x1 should be identified with                        *
* x2=m^2(sq_2 of flavour d)/m^2(kgluin)                                *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgd2d.f

```
      function dsbsgd2d()

************************************************************************
* Function \Delta^(2)_d (with d=b) in app A p. 17 of                   *
* Ciuchini et al., hep-ph/9806308                                      *
* Note that we have inserted d=b                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgd2td.f

```
      function dsbsgd2td(famd)

************************************************************************
* Function \Delta^(2)_{t,d} in app A p. 17 of                          *
* Ciuchini et al., hep-ph/9806308                                      *
* The input parameter famd is the family of the down sector            *
* it should be 1 for the first family, 2 for the 2nd and 3 for the 3rd*
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgd7chi1.f

```
      function dsbsgd7chi1(x)

************************************************************************
* Function \Delta_7^{\chi,1} eq (20) of                                *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgd7chi2.f

```
      function dsbsgd7chi2(x)

************************************************************************
* Function \Delta_7^{\chi,2} eq (20) of                                *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgd7h.f

```
      function dsbsgd7h(y)

************************************************************************
* Function \Delta_7^H(y) in (61) of Ciuchini et al.,                   *
* hep-ph/9710335                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgd8chi1.f

```
      function dsbsgd8chi1(x)

************************************************************************
* Function \Delta_8^{\chi,1} eq (21) of                                *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgd8chi2.f

```
      function dsbsgd8chi2(x)

************************************************************************
* Function \Delta_8^{\chi,2} eq (21) of                                *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsgd8h.f

```
      function dsbsgd8h(y)

************************************************************************
* Function \Delta_8^H(y) in (63) of Ciuchini et al.,                   *
* hep-ph/9710335                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgeb.f

```
      function dsbsgeb()

************************************************************************
* Function \epsilon_b in (10) of Degrassi et al.,                      *
* hep-ph/0009337                                                       *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                    *
************************************************************************
```

dsbsgechi.f

```
      function dsbsgechi(y)

************************************************************************
* Function E_\chi(y) in (11) of Ciuchini et al.,                       *
* hep-ph/9806308                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-04                    *
```

```
**************************************************************************
```

dsbsgeh.f

```
      function dsbsgeh(y)

**************************************************************************
* Function E^H(y) in (64) of Ciuchini et al.,                            *
* hep-ph/9710335                                                         *
* y must be a positive number                                            *
* for the calculation of b --> s gamma                                   *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                      *
**************************************************************************
```

dsbsget0.f

```
      function dsbsget0(x,flag)

**************************************************************************
* Function E^t_0(x) p. 11 in Gambino and Misiak,                         *
* Nucl. Phys. B611 (2001) 338                                            *
* x must be a positive number                                            *
* for the calculation of b --> s gamma                                   *
* author:Mia Schelke, schelke@physto.se, 2003-03-13                      *
* updated by Mia Schelke 2003-04-10 to include the susy contribution     *
* as explained in eq. (5.1)                                              *
* (the references used for the susy contributions can be found in        *
*  the different fortran codes)                                          *
* Input: flag:  0 = only standard model                                  *
*               1 = standard model plus SUSY corrections                 *
**************************************************************************
```

dsbsgf71.f

```
      function dsbsgf71(y)

**************************************************************************
* Function F_7^(1)(y) in (29) of Ciuchini et al.,                        *
* hep-ph/9710335                                                         *
* y must be a positive number                                            *
* for the calculation of b --> s gamma                                   *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                      *
**************************************************************************
```

dsbsgf72.f

```
      function dsbsgf72(y)

**************************************************************************
* Function F_7^(2)(y) in (54) of Ciuchini et al.,                        *
* hep-ph/9710335                                                         *
* y must be a positive number                                            *
* for the calculation of b --> s gamma                                   *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                      *
**************************************************************************
```

dsbsgf73.f

```
      function dsbsgf73(y)

************************************************************************
* Function F_7^(3)(y) in (21) of Degrassi et al.,                      *
* hep-ph/0009337                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                    *
************************************************************************
```

dsbsgf81.f

```
      function dsbsgf81(y)

************************************************************************
* Function F_8^(1)(y) in (30) of Ciuchini et al.,                      *
* hep-ph/9710335                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgf82.f

```
      function dsbsgf82(y)

************************************************************************
* Function F_8^(2)(y) in (55) of Ciuchini et al.,                      *
* hep-ph/9710335                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgf83.f

```
      function dsbsgf83(y)

************************************************************************
* Function F_8^(3)(y) in (22) of Degrassi et al.,                      *
* hep-ph/0009337                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                    *
************************************************************************
```

dsbsgft0.f

```
      function dsbsgft0(x,flag)

************************************************************************
* Function F^t_0(x) in (2.7) of Gambino and Misiak,                    *
* Nucl. Phys. B611 (2001) 338                                          *
* x must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-12                    *
```

```
* updated by Mia Schelke 2003-04-10 to include the susy contribution  *
* as explained in eq. (5.1)                                           *
* (the references used for the susy contributions can be found in     *
*  the different fortran codes)                                       *
* Input: flag:  0 = only standard model                               *
*               1 = standard model plus SUSY corrections              *
**********************************************************************
```

dsbsgft1.f

```
      function dsbsgft1(x,flag)

**********************************************************************
* Function F^t_1(x) p. 11 in Gambino and Misiak,                      *
* Nucl. Phys. B611 (2001) 338                                         *
* x must be a positive number                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-13                   *
* updated by Mia Schelke 2003-04-10 to include the susy contribution  *
* as explained in eq. (5.1)                                           *
* (the references used for the susy contributions can be found in     *
*  the different fortran codes)                                       *
* Input: flag:  0 = only standard model                               *
*               1 = standard model plus SUSY corrections              *
**********************************************************************
```

dsbsgfxy.f

```
      function dsbsgfxy(x,y)

**********************************************************************
* Function F'(x,y) in app. B p. 18 of                                 *
* Ciuchini et al., hep-ph/9806308                                     *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-22                   *
**********************************************************************
```

dsbsgg.f

```
      function dsbsgg(t)

**********************************************************************
* Function G(t) in (E.8) of Gambino and Misiak,                       *
* Nucl. Phys. B611 (2001) 338                                         *
* t must be a positive number                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-05                   *
**********************************************************************
```

dsbsgg7chi2.f

```
      function dsbsgg7chi2(x)

**********************************************************************
* Function G^{chi,2}_7(x) in eq. (16) of                              *
* Ciuchini et al., hep-ph/9806308                                     *
```

```
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                   *
**********************************************************************
```

dsbsgg7chij1.f

```
      function dsbsgg7chij1(x,j)

**********************************************************************
* Function G^{chi,1}_7(x) in eq. (15) of                              *
* Ciuchini et al., hep-ph/9806308                                     *
* The expression has been extended to large tanbe, by replacing       *
* ln(m^2(kgluin)/m^2(\chi_j)) by ln((mu_w)^2)/m^2(\chi_j))            *
* as explained in Degrassi et al., hep-ph/0009337 p.11                *
* The input, j, is the nb of the chargino, it must be 1 or 2          *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                   *
**********************************************************************
```

dsbsgg7h.f

```
      function dsbsgg7h(y)

**********************************************************************
* Function G_7^H(y) in (60) of Ciuchini et al.,                       *
* hep-ph/9710335                                                      *
* y must be a positive number                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                   *
**********************************************************************
```

dsbsgg7w.f

```
      function dsbsgg7w(x)

**********************************************************************
* Function G^W_7(x) in eq. (27) of                                    *
* Ciuchini et al., hep-ph/9806308                                     *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgg8chi2.f

```
      function dsbsgg8chi2(x)

**********************************************************************
* Function G^{chi,2}_8(x) in eq. (18) of                              *
* Ciuchini et al., hep-ph/9806308                                     *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                   *
**********************************************************************
```

dsbsgg8chij1.f

```
      function dsbsgg8chij1(x,j)
```

```
************************************************************************
* Function G^{chi,1}_8(x) in eq. (17) of                               *
* Ciuchini et al., hep-ph/9806308                                      *
* The expression has been extended to large tanbe, by replacing        *
* ln(m^2(kgluin)/m^2(\chi_j)) by ln((mu_w)^2)/m^2(\chi_j))             *
* as explained in Degrassi et al., hep-ph/0009337 p.11                 *
* The input, j, is the nb of the chargino, it must be 1 or 2           *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgg8h.f

```
      function dsbsgg8h(y)

************************************************************************
* Function G_8~H(y) in (62) of Ciuchini et al.,                        *
* hep-ph/9710335                                                       *
* y must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-03-31                    *
************************************************************************
```

dsbsgg8w.f

```
      function dsbsgg8w(x)

************************************************************************
* Function G^W_8(x) in eq. (27) of                                     *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                    *
************************************************************************
```

dsbsggxy.f

```
      function dsbsggxy(x,y)

************************************************************************
* Function G'(x,y) in app. B p. 18 of                                  *
* Ciuchini et al., hep-ph/9806308                                      *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-22                    *
************************************************************************
```

dsbsgh1x.f

```
      function dsbsgh1x(x)

************************************************************************
* Function H_1(x) in app A p. 16 of Ciuchini et al., hep-ph/9806308    *
* x must be a positive number                                          *
* for the calculation of b --> s gamma                                 *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                    *
************************************************************************
```

dsbsgh2xy.f

```
      function dsbsgh2xy(x,y)

**********************************************************************
* Function H_2(x,y) in (12) of Degrassi et al.,                      *
* hep-ph/0009337                                                     *
* x and y must be  positive numbers                                  *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                  *
**********************************************************************
```

dsbsgh3x.f

```
      function dsbsgh3x(x)

**********************************************************************
* Function H_3(x) in app A p. 17 of Ciuchini et al., hep-ph/9806308  *
* x must be a positive number                                        *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                  *
**********************************************************************
```

dsbsghd.f

```
      function dsbsghd()

**********************************************************************
* Function H_d (with d=b) in app A p. 16 of                          *
* Ciuchini et al., hep-ph/9806308                                    *
* Note that we have inserted d=b                                     *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                  *
**********************************************************************
```

dsbsghtd.f

```
      function dsbsghtd(famd)

**********************************************************************
* Function H_t^d in app A p. 16 of                                   *
* Ciuchini et al., hep-ph/9806308                                    *
* The input parameter famd is the family of the down sector          *
* it should be 1 for the first family, 2 for the 2nd and 3 for the 3rd*
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                  *
**********************************************************************
```

dsbsgkc.f

```
      function dsbsgkc()
**********************************************************************
* Program that calculates K_c in (3.7) of Gambino and Misiak,        *
* Nucl. Phys. B611 (2001) 338                                        *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-03-25                  *
**********************************************************************
```

dsbsgkt.f

```
      function dsbsgkt(flag)
c      program dsackt
**********************************************************************
* Program that calculates K_t in (3.8) of Gambino and Misiak,        *
* Nucl. Phys. B611 (2001) 338                                        *
* for the calculation of b --> s gamma                               *
* Input: flag:  0 = only standard model                              *
*               1 = standard model plus SUSY corrections             *
* author:Mia Schelke, schelke@physto.se, 2003-03-24                  *
**********************************************************************
```

dsbsgmtmuw.f

```
      function dsbsgmtmuw(m)

**********************************************************************
* The running top mass evaluated at a weak scale m=mu_w              *
* using nf effective quark flavours (taken to be nf=5 here)          *
* Uses eq. (32) of Ciuchini et al. hep-ph/9710335                    *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                  *
**********************************************************************
```

dsbsgmtmuwint.f

```
      function dsbsgmtmuwint(mtstart,mstart,m,nf)

**********************************************************************
* The running top mass from value mtstart at scale mstart.           *
* The running is done with nf effective active quark flavours.       *
* Uses eq. (32) of Ciuchini et al. hep-ph/9710335                    *
* for the calculation of b --> s gamma                               *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                  *
**********************************************************************
```

dsbsgpre2007full.f

```
      subroutine dsbsgpre2007full(ratio,flag)

**********************************************************************
* Routine that calculates the b-->s+gamma branching rate             *
* The Standard Model contribution is taken from                      *
* Gambino and Misiak,Nucl. Phys. B611 (2001) 338                     *
* with new 'magic numbers' from Buras et al., hep-ph/0203135         *
* Input: flag:  0 = only standard model                              *
*               1 = standard model plus SUSY corrections             *
* Output: ratio = BR(b -> s gamma)                                   *
* author:Mia Schelke, schelke@physto.se, 2003-03-27                  *
**********************************************************************
```

dsbsgri.f

```
      function dsbsgri(famd)

**********************************************************************
* Function R_i in eq. (19) of                                        *
* Ciuchini et al., hep-ph/9806308                                    *
```

```
* The expression has been extended to large tanbe, by dropping        *
* ln((mu_w)^2)/m^2(kgluin))                                           *
* as explained in Degrassi et al., hep-ph/0009337 p.11                *
* The input parameter famd is the family of the down sector           *
* it should be 1 for the first family, 2 for the 2nd and 3 for the 3rd*
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-07                   *
**********************************************************************
```

dsbsgud.f

```
      function dsbsgud()

**********************************************************************
* Function U_d (with d=b) in app A p. 15-6 of                         *
* Ciuchini et al., hep-ph/9806308                                     *
* Note that we have inserted d=b                                      *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgutd.f

```
      function dsbsgutd(famd)

**********************************************************************
* Function U_t^d in app A p. 16 of                                    *
* Ciuchini et al., hep-ph/9806308                                     *
* The input parameter famd is the family of the down sector           *
* it should be 1 for the first family, 2 for the 2nd and 3 for the 3rd*
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgwud.f

```
      function dsbsgwud(famu,famd)

**********************************************************************
* Function W_u^d in app A p. 15 of                                    *
* Ciuchini et al., hep-ph/9806308                                     *
* The input parameters famu and famd are the families of the up- and  *
* down sector respectively, and they should be 1 for the first family,*
* 2 for the 2nd and 3 for the 3rd                                     *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgwxy.f

```
      function dsbsgwxy(x,y)

**********************************************************************
* Function W[x,y] in app. A p. 15 of                                  *
* Ciuchini et al., hep-ph/9806308                                     *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-08                   *
**********************************************************************
```

dsbsgyt.f

```
      function dsbsgyt(m)

************************************************************************
* Function that calculates y_t(m=mu_susy), the top Yukawa coupling    *
* at the scale m=mu_susy                                              *
* Note that m=mu_susy should be of the order of 1TeV, e.g. m_gluino   *
* Uses eq (23) of Degrassi et al.,  hep-ph/0009337                    *
* Note that y_t(susy)=\tilde{y}_t(susy) (see text in the ref.)        *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-04-03                   *
************************************************************************
```

dsphi22a.f

```
      function dsphi22a(t)
************************************************************************
* The first integrand of \phi_22 in (E.2) of Gambino and Misiak,      *
* Nucl. Phys. B611 (2001) 338                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-10                   *
************************************************************************
```

dsphi22b.f

```
      function dsphi22b(t)
************************************************************************
* The 2nd integrand of \phi_22 in (E.2) of Gambino and Misiak,        *
* Nucl. Phys. B611 (2001) 338                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-10                   *
************************************************************************
```

dsphi27a.f

```
      function dsphi27a(t)
************************************************************************
* The first integrand of \phi_27 in (E.3) of Gambino and Misiak,      *
* Nucl. Phys. B611 (2001) 338                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-10                   *
************************************************************************
```

dsphi27b.f

```
      function dsphi27b(t)
************************************************************************
* The 2nd integrand of \phi_27 in (E.3) of Gambino and Misiak,        *
* Nucl. Phys. B611 (2001) 338                                         *
* for the calculation of b --> s gamma                                *
* author:Mia Schelke, schelke@physto.se, 2003-03-10                   *
************************************************************************
```

39.3 mssm/an: (Co-)annihilation cross sections

39.3.1 Annihilation cross sections – theory

For the relic density calculations, we need all possible (co)annihilation cross sections between neutralinos, charginos and sfermions.

39.3.1.1 Annihilation cross sections

We have calculated all two-body final state cross sections at tree level involving neutralinos, charginos, sneutrinos, sleptons and squarks in the initial state. A complete list is given below.

Since we have so many different diagrams contributing, we have to use some method where the diagrams can be calculated efficiently. To achive this, we calculate the diagrams with general expressions for vertices, masses etc so that they can be reused for other processes. How we do this in practice differs a bit between different sets of annihilation diagrams.

For neutralino-neutralino, neutralino-chargino and chargino-chargino annihilation, we classify the diagrams according to their topology (s-, t- or u-channel) and to the spin of the particles involved. We then compute the helicity amplitudes for each type of diagram analytically with REDUCE [186] using general expressions for the vertex couplings.

The strength of the helicity amplitude method is that the analytical calculation of a given type of diagram has to be performed only once and the sum of the contributing diagrams for each set of initial and final states can be done numerically afterwards.

For the diagrams involving sfermions, FORM is used to analytically calculate the amplitudes. This output is then converted into Fortran with a PERL script, **form2f** [187].

39.3.1.2 Coannihilation diagrams

All Feynman diagrams for which we calculate the annihilation cross section are listed in the coming sections. $s(x)$, $t(x)$ and $u(x)$ denote a tree-level Feynman diagram in which particle x is exchanged in the s-, t- and u-channel respectively.

The convention used in this list of included coannihilation diagrams is that if a sfermion is denoted $\tilde{f}$, then it's antiparticle is denoted $\tilde{f}^*$.

39.3.1.3 Neutralino and chargino annihilation

Indices i, j, k run from 1 to 4, and indices c, d, e from 1 to 2. u, $\tilde{u}$, d, $\tilde{d}$, ν, $\tilde{\nu}$, ℓ, $\tilde{\ell}$, f and $\tilde{f}$ are generic notations for up-type quarks, up-type squarks, down-type quarks, down-type squarks, neutrinos, sneutrinos, leptons, sleptons, fermions and sfermions. A sum of diagrams over (s)fermion generation indices and over the neutralino and chargino indices k and e is understood (no sum over indices i, j, c, d).

Neutralino-neutralino annihilation

Initial state	Final state	Feynman diagrams
$\chi_i^0\chi_j^0$	H_1H_1, H_1H_2, H_2H_2, H_3H_3	$t(\chi_k^0)$, $u(\chi_k^0)$, $s(H_{1,2})$
	H_1H_3, H_2H_3	$t(\chi_k^0)$, $u(\chi_k^0)$, $s(H_3)$, $s(Z^0)$
	H^-H^+	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2})$, $s(Z^0)$
	Z^0H_1, Z^0H_2	$t(\chi_k^0)$, $u(\chi_k^0)$, $s(H_3)$, $s(Z^0)$
	Z^0H_3	$t(\chi_k^0)$, $u(\chi_k^0)$, $s(H_{1,2})$
	W^-H^+, W^+H^-	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2,3})$
	Z^0Z^0	$t(\chi_k^0)$, $u(\chi_k^0)$, $s(H_{1,2})$
	W^-W^+	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2})$, $s(Z^0)$
	$f\bar{f}$	$t(\tilde{f}_{L,R})$, $u(\tilde{f}_{L,R})$, $s(H_{1,2,3})$, $s(Z^0)$

Neutralino-chargino annihilation

Initial state	Final state	Feynman diagrams
$\chi_c^+\chi_i^0$	H^+H_1, H^+H_2	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(H^+)$, $s(W^+)$
	H^+H_3	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(W^+)$
	W^+H_1, W^+H_2	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(H^+)$, $s(W^+)$
	W^+H_3	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(H^+)$
	H^+Z^0	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(H^+)$
	γH^+	$t(\chi_c^+)$, $s(H^+)$
	W^+Z^0	$t(\chi_k^0)$, $u(\chi_e^+)$, $s(W^+)$
	γW^+	$t(\chi_c^+)$, $s(W^+)$
	$u\bar{d}$	$t(\tilde{d}_{L,R})$, $u(\tilde{u}_{L,R})$, $s(H^+)$, $s(W^+)$
	$\nu\bar{\ell}$	$t(\tilde{\ell}_{L,R})$, $u(\tilde{\nu}_L)$, $s(H^+)$, $s(W^+)$

Chargino-chargino annihilation

Initial state	Final state	Feynman diagrams
$\chi_c^+\chi_d^-$	H_1H_1, H_1H_2, H_2H_2, H_3H_3	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2})$
	H_1H_3, H_2H_3	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_3)$, $s(Z^0)$
	H^+H^-	$t(\chi_k^0)$, $s(H_{1,2})$, $s(Z^0,\gamma)$
	Z^0H_1, Z^0H_2	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_3)$, $s(Z^0)$
	Z^0H_3	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2})$
	H^+W^-, W^+H^-	$t(\chi_k^0)$, $s(H_{1,2,3})$
	Z^0Z^0	$t(\chi_e^+)$, $u(\chi_e^+)$, $s(H_{1,2})$
	W^+W^-	$t(\chi_k^0)$, $s(H_{1,2})$, $s(Z^0,\gamma)$
	$\gamma\gamma$ (only for $c=d$)	$t(\chi_c^+)$, $u(\chi_c^+)$
	$Z^0\gamma$	$t(\chi_d^+)$, $u(\chi_c^+)$
	$u\bar{u}$	$t(\tilde{d}_{L,R})$, $s(H_{1,2,3})$, $s(Z^0,\gamma)$
	$\nu\bar{\nu}$	$t(\tilde{\ell}_{L,R})$, $s(Z^0)$
	$\bar{d}d$	$t(\tilde{u}_{L,R})$, $s(H_{1,2,3})$, $s(Z^0,\gamma)$
	$\bar{\ell}\ell$	$t(\tilde{\nu}_L)$, $s(H_{1,2,3})$, $s(Z^0,\gamma)$
$\chi_c^+\chi_d^+$	H^+H^+	$t(\chi_k^0)$, $u(\chi_k^0)$
	H^+W^+	$t(\chi_k^0)$, $u(\chi_k^0)$
	W^+W^+	$t(\chi_k^0)$, $u(\chi_k^0)$

39.3.1.4 Squark-squark annihilation

Note: The tables below are not entirely up to date, more processes are included than shown in the tables.

We will here denote squarks as $\tilde{q}_a^i$ and $\tilde{q}_b^j$ where i and j are the family indices and a and b are the mass eigenstate indices (running from 1 to 2). k and l will also be used as family indices for

processes including more squarks. Colour indices are suppressed. $\tilde{u}^i$ is used as a generic notation for any up-type squark where i denotes the family index. Down-type squarks are denoted analogously.

Note that we will not (except in rare occations) show processes for $\tilde{\nu}$ and $\tilde{\ell}$ separately since they can easily be obtained from the squark processes by replacing $\tilde{u}$ with $\tilde{\nu}$ and $\tilde{d}$ with $\tilde{\ell}$ (and noting that we only have one mass eigenstate for the $\tilde{\nu}$. Also note that the $\tilde{\nu} - \tilde{\ell}$–sector is assumed not to be flavour-changing.

$\tilde{d}_a^i \tilde{d}_b^{i*}$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i \tilde{d}_b^{i*}$	$\gamma\gamma$, $Z\gamma$	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, p	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	ZZ	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, p, $s(H_1, H_2)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	W^-W^+	p, $s(H_1, H_2, Z, \gamma)$, $t(\tilde{u}_{1,2}^k)$	$k = 1, 2, 3$
$\tilde{d}_a^i \tilde{d}_b^{i*}$	ZH_2, ZH_1	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, $s(Z, H_3)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	ZH_3	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, $s(H_1, H_2)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	γH_2, γH_1, γH_3	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	H_2H_2, H_1H_1, H_1H_2	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, p, $s(H_1, H_2)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	H_2H_3, H_1H_3	$s(Z, H_3)$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	H_3H_3	$s(H_1, H_2)$, p, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$	
$\tilde{d}_a^i \tilde{d}_b^{i*}$	W^-H^+	$s(H_1, H_2, H_3)$, $t(\tilde{u}_{1,2}^k)$	$k = 1, 2, 3$
$\tilde{d}_a^i \tilde{d}_b^{i*}$	H^-H^+	$s(H_1, H_2, Z, \gamma)$, p, $t(\tilde{u}_{1,2}^k)$	$k = 1, 2, 3$
$\tilde{d}_a^i \tilde{d}_b^{i*}$	$f\bar{f}$ $(f \neq d^i)$	$s(H_1^\star, H_2^\star, H_3^\star, Z, \gamma^\star, g^\ddagger)$, $t(\chi_c^+)^\dagger$	†) $f = u^k$ $(k = 1, 2, 3)$, $\star$) Not for $f = \nu$, ‡) Only for squarks/quarks
$\tilde{d}_a^i \tilde{d}_b^{i*}$	$d^i\bar{d}^i$	$s(H_1, H_2, H_3, Z, \gamma, g^\dagger)$, $t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks
$\tilde{d}_a^i \tilde{d}_b^{i*}$	Zg	$t(\tilde{d}_{1,2}^i), u(\tilde{d}_{1,2}^i), p$	Only for squarks
$\tilde{d}_a^i \tilde{d}_b^{i*}$	gg	$t(\tilde{d}_{1,2}^i), u(\tilde{d}_{1,2}^i), s(g), p$	Only for squarks
$\tilde{d}_a^i \tilde{d}_b^{i*}$	$g\gamma$	$t(\tilde{d}_{1,2}^i), u(\tilde{d}_{1,2}^i), p$	Only for squarks
$\tilde{d}_a^i \tilde{d}_b^{i*}$	gH_1, gH_2, gH_3	$t(\tilde{d}_{1,2}^i), u(\tilde{d}_{1,2}^i)$	Only for squarks

$\tilde{d}_a^i \tilde{d}_b^{j*}$ annihilation $(i \neq j)$

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i \tilde{d}_b^{j*}$	W^+W^-	$t(\tilde{u}_{1,2}^k)^\dagger$	Not included at present
$\tilde{d}_a^i \tilde{d}_b^{j*}$	W^+H^-	$t(\tilde{u}_{1,2}^k)^\dagger$	Not included at present
$\tilde{d}_a^i \tilde{d}_b^{j*}$	H^+H^-	$t(\tilde{u}_{1,2}^k)^\dagger$	Not included at present
$\tilde{d}_a^i \tilde{d}_b^{*j}$	$d^i\bar{d}^j$	$t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks
$\tilde{d}_a^i \tilde{d}_b^{*j}$	$u^k\bar{u}^l$	$t(\tilde{\chi}_c^+)$	Only $k = i, l = j$ at present

$\tilde{d}_a^i \tilde{d}_b^i$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i \tilde{d}_b^i$	$d^i d^i$	$t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$, $u(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks

$\tilde{d}_a^i \tilde{d}_b^j$ annihilation $(i \neq j)$

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i \tilde{d}_b^j$	$d^i d^j$	$t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks

$\tilde{u}_a^i \tilde{u}_b^{i*}$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{u}_b^{i*}$	$\gamma\gamma^\dagger$, $Z\gamma^\dagger$	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, p	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	ZZ	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, p, $s(H_1, H_2)$	
$\tilde{u}_a^i \tilde{u}_b^{i*}$	W^-W^+	p, $s(H_1, H_2, Z, \gamma^\dagger)$, $u(\tilde{d}_{1,2}^k)$	$k = 1, 2, 3$, †) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	ZH_2, ZH_1	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, $s(Z, H_3^\dagger)$	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	ZH_3	$t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{u}_{1,2}^i)^\dagger$, $s(H_1, H_2)$	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	$\gamma H_2^\dagger$, $\gamma H_1^\dagger$, $\gamma H_3^\dagger$	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	H_2H_2, H_1H_1, H_1H_2	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, p, $s(H_1, H_2)$	
$\tilde{u}_a^i \tilde{u}_b^{i*}$	H_2H_3, H_1H_3	$s(Z, H_3^\dagger)$, $t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{u}_{1,2}^i)^\dagger$	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	H_3H_3	$s(H_1, H_2)$, p, $t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{u}_{1,2}^i)^\dagger$	†) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	W^-H^+	$s(H_1, H_2, H_3^\dagger)$, $u(\tilde{d}_{1,2}^k)$	$k = 1, 2, 3$, †) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	H^+H^-	$s(H_1, H_2, Z, \gamma^\dagger)$, p, $t(\tilde{d}_{1,2}^k)$	$k = 1, 2, 3$, †) Not for $\tilde{\nu}$
$\tilde{u}_a^i \tilde{u}_b^{i*}$	$f\bar{f}$ $(f \neq u^i)$	$s(H_1^\times, H_2^\times, H_3^{\dagger\times}, Z, \gamma^{\dagger\times}, g^\ddagger), t(\chi_c^+)^\star$	†) Not for $\tilde{\nu}$, $\star$) If $f = d^k$ $(k = 1, 2, 3)$, ‡) Only for squarks/quarks, $\times$) Not for ν
$\tilde{u}_a^i \tilde{u}_b^{i*}$	$u^i\bar{u}^i$	$s(H_1^\times, H_2^\times, H_3^\times, Z, \gamma^\times, g^\ddagger)$, $t(\tilde{\chi}_k^0, \tilde{g}^\ddagger)$	$\times$) Not for ν, ‡) Only for squarks
$\tilde{u}_a^i \tilde{u}_b^{i*}$	Zg	$t(\tilde{u}_{1,2}^i), u(\tilde{u}_{1,2}^i), p$	Only for squarks
$\tilde{u}_a^i \tilde{u}_b^{i*}$	gg	$t(\tilde{u}_{1,2}^i), u(\tilde{u}_{1,2}^i), s(g), p$	Only for squarks
$\tilde{u}_a^i \tilde{u}_b^{i*}$	$g\gamma$	$t(\tilde{u}_{1,2}^i), u(\tilde{u}_{1,2}^i), p$	Only for squarks
$\tilde{u}_a^i \tilde{u}_b^{i*}$	gH_1, gH_2, gH_3	$t(\tilde{u}_{1,2}^i), u(\tilde{u}_{1,2}^i)$	Only for squarks

$\tilde{u}_a^i \tilde{u}_b^{j*}$ annihilation $(i \neq j)$

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{u}_b^{j*}$	W^+W^-	$t(\tilde{d}_{1,2}^k)^\dagger$	Not included at present, †) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{u}_b^{j*}$	W^+H^-	$t(\tilde{d}_{1,2}^k)^\dagger$	Not included at present, †) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{u}_b^{j*}$	H^+H^-	$t(\tilde{d}_{1,2}^k)^\dagger$	Not included at present, †) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{u}_b^{j*}$	$u^i\bar{u}^j$	$t(\tilde{\chi}_k^0, g^\dagger)$	†) Only for squarks
$\tilde{u}_a^i \tilde{u}_b^{j*}$	$d^k\bar{d}^l$	$t(\tilde{\chi}_c^+)$	Only $k = i, l = j$ at present

$\tilde{u}_a^i \tilde{u}_b^i$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{u}_b^i$	u^iu^i	$t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$, $u(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks

$\tilde{u}_a^i \tilde{u}_b^j$ annihilation $(i \neq j)$

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{u}_b^j$	u^iu^j	$t(\tilde{\chi}_k^0, \tilde{g}^\dagger)$	†) Only for squarks

$\tilde{u}_a^i \tilde{d}_b^{i*}$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{d}_b^{i*}$	H^+H_1, H^+H_2	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, p, $s(W^+, H^+)$	
$\tilde{u}_a^i \tilde{d}_b^{i*}$	H^+H_3	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)^\dagger$, p, $s(W^+)$	†) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{d}_b^{i*}$	γH^+	$t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{d}_{1,2}^i)$, $s(H^+)$	†) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{d}_b^{i*}$	ZH^+	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$, $s(H^+)$	
$\tilde{u}_a^i \tilde{d}_b^{i*}$	W^+H_1, W^+H_2	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, $s(W^+, H^+)$	
$\tilde{u}_a^i \tilde{d}_b^{i*}$	W^+H_3	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)^\dagger$, $s(H^+)$	†) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{d}_b^{i*}$	$W^+\gamma$	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)^\dagger$, $s(W^+)$, p	†) Not for $\tilde{\ell}$
$\tilde{u}_a^i \tilde{d}_b^{i*}$	W^+Z	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, $s(W^+)$, p	
$\tilde{u}_a^i \tilde{d}_b^{i*}$	$u^k \bar{d}^l$	$s(H^+, W^+)^\star$, $t(\tilde{\chi}_m^0, \tilde{g}^\dagger)\delta^{ik}\delta^{il}$	†) Not for $\tilde{\ell}$, $\star$) Only $k = l$ at present
$\tilde{u}_a^i \tilde{d}_b^{i*}$	W^+g	$t(\tilde{d}_{1,2}^i)$, $u(\tilde{u}_{1,2}^i)$, p	Only for squarks
$\tilde{u}_a^i \tilde{d}_b^{i*}$	gH^+	$t(\tilde{u}_{1,2}^i)$, $u(\tilde{d}_{1,2}^i)$	Only for squarks

$\tilde{u}_a^i \tilde{d}_b^{j*}$ annihilation ($i \neq j$) For squarks we can have the following processes

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{d}_b^{j*}$	H^+H_1, H^+H_2	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, p, $s(W^+, H^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	H^+H_3	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, p, $s(W^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	$H^+\gamma$	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(H^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	H^+Z	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(H^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	W^+H_1, W^+H_2	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(W^+, H^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	W^+H_3	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(H^+)$	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	$W^+\gamma$	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(W^+)$, p	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	W^+Z	$t(\tilde{d}_{1,2}^j)$, $u(\tilde{u}_{1,2}^i)$, $s(W^+)$, p	Not included at present
$\tilde{u}_a^i \tilde{d}_b^{j*}$	$u^k \bar{d}^l$	$s(H^+, W^+)^\dagger$, $t(\tilde{\chi}_m^0, \tilde{g})\delta^{ik}\delta^{jl}$	†) Not included at present

whereas for sneutrinos and sleptons, we can only have the process

Initial state	Final state	Diagrams	Note
$\tilde{\nu}^i \tilde{\ell}_b^{j*}$	$\nu^i \bar{\ell}^j$	$t(\tilde{\chi}_k^0)$	

$\tilde{u}_a^i \tilde{d}_b^i$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{d}_b^i$	$u^k d^l$	$t(\tilde{\chi}_m^0, \tilde{g}^\dagger)\delta^{ik}\delta^{il}$, $u(\tilde{\chi}_c^+)^\star$	†) Only for squarks, $\star$) Only $i = k = l$ at present

$\tilde{u}_a^i \tilde{d}_b^j$ annihilation ($i \neq j$)

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i \tilde{d}_b^j$	$u^k d^l$	$t(\tilde{\chi}_m^0, \tilde{g}^\dagger)\delta^{ik}\delta^{jl}$, $u(\tilde{\chi}_c^+)^{\times\star}$	†) Only for squarks, $\times$) For $\tilde{\nu}\tilde{\ell}$ only when $i = l, j = k$, $\star$) Only included when $i = l, j = k$ at present

39.3.1.5 Squark-neutralino annihilation

Note: The tables below are not entirely up to date, more processes are included than shown in the tables.

We will here denote squarks as $\tilde{u}_a^i$ and $\tilde{d}_a^i$ where i is the family index and a is the mass eigenstate index (running from 1 to 2).

$\tilde{u}_a^i\tilde{\chi}_j^0$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i\tilde{\chi}_j^0$	γu^i	$s(u^i)$, $t(\tilde{u}_{1,2}^i)$	Only for squarks
$\tilde{u}_a^i\tilde{\chi}_j^0$	Zu^i	$s(u^i)$, $t(\tilde{u}_{1,2}^i)$, $u(\tilde{\chi}_k^0)$	
$\tilde{u}_a^i\tilde{\chi}_j^0$	H_1u^i, H_2u^i	$s(u^i)^\dagger$, $t(\tilde{u}_{1,2}^i)$, $u(\tilde{\chi}_k^0)$	†) Only for squarks
$\tilde{u}_a^i\tilde{\chi}_j^0$	H_3u^i	$s(u^i)^\dagger$, $t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{\chi}_k^0)$	†) Only for squarks
$\tilde{u}_a^i\tilde{\chi}_j^0$	W^+d^k	$s(u^i)$, $t(\tilde{d}_{1,2}^k)$, $u(\tilde{\chi}_c^+)$	$k = 1,2,3$, $\tilde{u}_a^{i*}\tilde{\chi}_j^0 \to W^-\bar{d}^k$ in the code
$\tilde{u}_a^i\tilde{\chi}_j^0$	H^+d^k	$s(u^i)$, $t(\tilde{d}_{1,2}^k)$, $u(\tilde{\chi}_c^+)$	$k = 1,2,3$, $\tilde{u}_a^{i*}\tilde{\chi}_j^0 \to H^-\bar{d}^k$ in the code
$\tilde{u}_a^i\tilde{\chi}_j^0$	gu^i	$s(u^i)$, $t(\tilde{u}_{1,2}^i)$	

$\tilde{d}_a^i\tilde{\chi}_j^0$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i\tilde{\chi}_j^0$	γd^i	$s(d^i)$, $t(\tilde{d}_{1,2}^i)$	
$\tilde{d}_a^i\tilde{\chi}_j^0$	Zd^i	$s(d^i)$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_k^0)$	
$\tilde{d}_a^i\tilde{\chi}_j^0$	H_1d^i, H_2d^i	$s(d^i)$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_k^0)$	
$\tilde{d}_a^i\tilde{\chi}_j^0$	H_3d^i	$s(d^i)$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_k^0)$	
$\tilde{d}_a^i\tilde{\chi}_j^0$	W^-u^k	$s(d^i)$, $t(\tilde{u}_{1,2}^k)$, $u(\tilde{\chi}_c^+)$	$k = 1,2,3$
$\tilde{d}_a^i\tilde{\chi}_j^0$	H^-u^k	$s(d^i)$, $t(\tilde{u}_{1,2}^k)$, $u(\tilde{\chi}_c^+)$	$k = 1,2,3$
$\tilde{d}_a^i\tilde{\chi}_j^0$	gd^i	$s(d^i)$, $t(\tilde{d}_{1,2}^i)$	

39.3.1.6 Squark-chargino annihilation

Note: The tables below are not entirely up to date, more processes are included than shown in the tables.

We will here denote squarks as $\tilde{q}_a^i$ where i is the family index and a is the mass eigenstate index (running from 1 to 2).

$\tilde{u}_a^i\tilde{\chi}_c^+$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^i\tilde{\chi}_c^+$	W^+u^k	$t(\tilde{d}_{1,2}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{u}_a^i\tilde{\chi}_c^+$	H^+u^k	$t(\tilde{d}_{1,2}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present

$\tilde{u}_a^{i*}\tilde{\chi}_c^+$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$Z\bar{d}^k$	$s(\bar{d}^k)$, $t(\tilde{u}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	$k = 1, 2, 3$
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$\gamma\bar{d}^k$	$s(\bar{d}^k)$, $t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{\chi}_c^+)$	$k = 1, 2, 3$, †) Only for squarks
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$H_1\bar{d}^k$, $H_2\bar{d}^k$	$s(\bar{d}^k)$, $t(\tilde{u}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	$k = 1, 2, 3$
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$H_3\bar{d}^k$	$s(\bar{d}^k)$, $t(\tilde{u}_{1,2}^i)^\dagger$, $u(\tilde{\chi}_c^+)$	$k = 1, 2, 3$, †) Only for squarks
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$W^+\bar{u}^k$	$s(\bar{d}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$H^+\bar{u}^k$	$s(\bar{d}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{u}_a^{i*}\tilde{\chi}_c^+$	$g\bar{d}^k$	$s(\bar{d}^k)$, $t(\tilde{u}_a^i)$	$k = 1, 2, 3$, only for squarks

$\tilde{d}_a^i\tilde{\chi}_c^+$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^i\tilde{\chi}_c^+$	Zu^k	$s(u^k)$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	Only $k = i$ at present
$\tilde{d}_a^i\tilde{\chi}_c^+$	γu^k	$s(u^k)^\dagger$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	Only $k = i$ at present, †) Only for squarks
$\tilde{d}_a^i\tilde{\chi}_c^+$	H_1u^k, H_2u^k	$s(u^k)^\dagger$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	Only $k = i$ at present, †) Only for squarks
$\tilde{d}_a^i\tilde{\chi}_c^+$	H_3u^k	$s(u^k)^\dagger$, $t(\tilde{d}_{1,2}^i)$, $u(\tilde{\chi}_c^+)$	Only $k = i$ at present, †) Only for squarks
$\tilde{d}_a^i\tilde{\chi}_c^+$	W^+d^k	$s(u^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{d}_a^i\tilde{\chi}_c^+$	H^+d^k	$s(u^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{d}_a^i\tilde{\chi}_c^+$	gu^k	$s(u^k)$, $t(\tilde{d}_a^i)$	Only $k = i$ at present, only for squarks

$\tilde{d}_a^{i*}\tilde{\chi}_c^+$ annihilation

Initial state	Final state	Diagrams	Note
$\tilde{d}_a^{i*}\tilde{\chi}_c^+$	$W^+\bar{d}^k$	$t(\tilde{u}_{1,2}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present
$\tilde{d}_a^{i*}\tilde{\chi}_c^+$	$H^+\bar{d}^k$	$t(\tilde{u}_{1,2}^l)$, $u(\tilde{\chi}_c^0)\delta^{ik}$	Only $k = l = i$ at present

39.3.1.7 Degrees of freedom

We have to be careful with the internal degrees of freedom, g, of the particles. We can either treat e.g. a χ_i^+ and a χ_i^- as two separate particles with two degrees of freedom each, or we can treat them as one particle $\chi_i^\pm$ with four degrees of freedom. The latter approach has an advantage that we simplify our expressions for the effective annihilation cross sections when coannihilations are needed. Hence, we use that approach here. For a more detailed discussion about this, see Section 39.21.2.

39.3.2 Annihilation routines - general remarks

The annihilation cross section routines is divided into several parts, mostly for historical reasons. The layout is roughly as follows:

src/an Here we keep the main routins for both neutralino- neutralino annihilation cross sections and the effective annihilation cross section in the relic density calculations. The steering routines for neutralino and chargino coannihilations are also kept here.

src/anstu Here keep the $t-$, $u-$ and $s-$ diagram expressions for fermion-fermion coannihilations (i.e. neutralino and chargino coannihilations).

src/as Here all the coannihilation cross sections including sfermions are kept.

We will here describe the src/an-routines.

39.3.2.1 General routines

The general routine to call for an effective annihilation cross section (to be used for relic density calculations) is **dsanwx**, which returns the invariant annihilation rate (integrated over $\cos\theta$). The actual cross section, differential in $\cos\theta$ is calculated by **dsandwdcos** which includes all the coannihilations needed. This is set up in **mssm/rd/dsrdparticles** (typically called from dsrdomega in the core library) which determines which coannihilating particles to include.

For other applications where the annihilation rate is needed, e.g. annihilation in the galactic halo, one can call the specific annihilation rate routine directly. The main one is **dsandwdcosnn** for neutralino-neutralino annihilation. To simplify this task, we supply a routine **dssigmav0** which calls **dsandwdcosnn** for neutralino-neutralino annihilation at zero relative velocity and returns the result, either as the total annihilation cross section, or the cross section for a specific channel. See the header of **dssigmav** for details.

39.3.2.2 Neutralino and chargino (co)annihilation cross sections

The routines **dsandwdcosnn**, **dsandwdcoscn** and **dsandwdcoscc** calculate the annihilation cross sections (returning the invariant annihilation rate) for neutralino-neutralino, neutralino-chargino and chargino-chargino annihilations. Which particles the cross section is calculated for is given by particle indices as defined in **mssm/include/dsmssm.h**.

All the annihilation routines return the invariant rate instead of the cross section. The invariant annihilation rate between particle i and j is defined as

$$W_{ij} = 4p_{ij}\sqrt{s}\sigma_{ij} = 4\sigma_{ij}\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4E_i E_j \sigma_{ij} v_{ij}. \tag{39.1}$$

See Chapter 27 for more details.

39.3.3 Routine headers – fortran files

dsanalbe.f

```
      subroutine dsanalbe(alph,bet)
c-----------------------------------------------------------------------
c  determine alph and bet for the integration.
c  modified: joakim edsjo (edsjo@fysik.su.se) 97-09-09
c=======================================================================
```

dsanclearaa.f

```
      subroutine dsanclearaa
c-----------------------------------------------------------------------
c  clear the amplitude matrix
c  author: joakim edsjo (edsjo@physto.se) 95-10-25
c          paolo gondolo 99-1-15 factor of 3.7 faster
c          torsten bringmann: added OMP compatibility
c  called by: dwdcos
c=======================================================================
```

dsandwdcos.f

```
      function dsandwdcos(p,costheta)
c-----------------------------------------------------------------------
c  annihilation differential invariant rate.
c  input:
c    p - initial cm momentum (real) for lsp annihilations
c    costheta - cosine of c.m. annihilation angle
c  Output:
```

$$\frac{dW_{ij}}{d\cos\theta}$$

where

$$W_{ij} = 4p_{ij}\sqrt{s}\sigma_{ij} = 4\sigma_{ij}\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4E_i E_j \sigma_{ij} v_{ij}$$

```
c  The returned dW/dcos(theta) is dimensionless
c  uses dsandwdcosnn, dsandwdcoscn and dsandwdcoscc and
c  routines in src/as
c  called by dsanwx.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 96-02-21
c  modified: 97-05-12 Joakim Edsjo (edsjo@fysik.su.se)
c  modified: 01-01-30 paolo gondolo (paolo@mamma-mia.phys.cwru.edu)
c  modified: 02-03-09 Joakim Edsjo (edsjo@fysik.su.se)
c  modified: 06-02-22 Paolo Gondolo (paolo@physics.utah.edu)
c=======================================================================
```

dsandwdcoscc.f

```
      function dsandwdcoscc(p,costheta,kp1,kp2)
c-----------------------------------------------------------------------
c  annihilation differential invariant rate between particle kp1
c  and kp2 where kp1 and kp2 are charginos
c  input:
c    p - initial cm momentum (real)
c    costheta - cosine of c.m. annihilation angle
c    kp1 - particle code, particle 1
c    kp2 - particle code, particle 2
c  Output:
```

$$\frac{dW_{ij}}{d\cos\theta}$$

where

$$W_{ij} = 4p_{ij}\sqrt{s}\sigma_{ij} = 4\sigma_{ij}\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4E_i E_j \sigma_{ij} v_{ij}$$

```
c  The returned dW/dcos(theta) is unitless
c  uses dsanclearaa,dsansumaa
c  called by dsandwdcos.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 96-08-06
c  modified: 01-09-12
c=======================================================================
```

dsandwdcoscn.f

```
      function dsandwdcoscn(p,costheta,kp1,kp2)
c-----------------------------------------------------------------------
c  annihilation differential invariant rate between particle kp1
c  and kp2 where kp1 is a chargino and kp2 is a neutralino.
c  input:
c    p - initial cm momentum (real)
c    costheta - cosine of c.m. annihilation angle
c    kp1 - particle code, particle 1
c    kp2 - particle code, particle 2
c  Output:
```

$$\frac{dW_{ij}}{d\cos\theta}$$

where

$$W_{ij} = 4p_{ij}\sqrt{s}\sigma_{ij} = 4\sigma_{ij}\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4E_i E_j \sigma_{ij} v_{ij}$$

```
c  The returned dW/dcos(theta) is unitless
c  uses dsanclearaa,dsansumaa
c  called by dsandwdcos.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 96-08-06
c  modified: 01-09-12
c=======================================================================
```

dsandwdcosd.f

```
      function dsandwdcosd(costheta)
c-----------------------------------------------------------------------
c  10^15*annihilation differential invariant rate.
c  input:
c    p - initial cm momentum (real) for lsp annihilations via common
c    costheta - cosine of c.m. annihilation angle
c  uses dwdcos
c  used for gaussian integration with gadap.f
c  author: joakim edsjo (edsjo@physto.se)
c  date: 97-01-09
c... mod TB 2016-02-08: added OMP compatibility
c=======================================================================
```

dsandwdcosij.f

```
      function dsandwdcosij(p,costheta,kp1,kp2)
No header found.
```

dsandwdcosnn.f

```
      function dsandwdcosnn(p,costheta,kp1,kp2)
c-----------------------------------------------------------------------
c  Annihilation differential invariant rate between particle kp1
c  and kp2 where kp1 and kp2 are neutralinos
c  Input:
c    p - initial cm momentum (real)
c    costheta - cosine of c.m. annihilation angle
c    kp1 - particle code for particle 1
c    kp2 - particle code for particle 2
c  Output:
```

$$\frac{dW_{ij}}{d\cos\theta}$$

where

$$W_{ij} = 4p_{ij}\sqrt{s}\sigma_{ij} = 4\sigma_{ij}\sqrt{(p_i \cdot p_j)^2 - m_i^2 m_j^2} = 4E_i E_j \sigma_{ij} v_{ij}$$

```
c  The returned dW/dcos(theta) is unitless.
c
c  uses dsanclearaa,dsansumaa
c  called by dsandwdcos.
c  note: the 32pi in the partial cross sections is 8pi g_1^2, g_1=2
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 96-02-21
c  modified: 01-09-12
c  modifed: 15-10-29 added QCD NLO corrections (torsten bringmann)
c  modifed: 2022-04-29 updated QCD NLO implementation; full s-wave (tb)
c=======================================================================
```

dsandwdcoss.f

```
      function dsandwdcoss(costheta)
c-----------------------------------------------------------------------
c  10^15*annihilation differential invariant rate.
c  input:
c    p - initial cm momentum (real) for lsp annihilations via common
c    costheta - cosine of c.m. annihilation angle
c  uses dwdcos
c  used for gaussian integration with gadap.f
c  author: joakim edsjo (edsjo@physto.se)
c  date: 97-01-09
c  mod TB 2016-02-08: added OMP compatibility
c=======================================================================
```

dsandwdcosy.f

```
      function dsandwdcosy(y)
c-----------------------------------------------------------------------
c  10^15*annihilation differential invariant rate.
c  the integration variable is changed from cos(theta) to
c    y=1/(mx^2+2p^2(1-cos(theta))) for cos(theta)>0 and to
c    y=1/(mx^2+2p^2(1-cos(theta))) for cos(theta)<0.
c  this avoids the poles at cos(theta)=+-1
c  integrate this function from 1/(mx^2+2p^2) to 1/mx^2 to get
c  1d15 times the integral from cos(theta)=0 to 1.
c  input:
c    y - initial cm momentum (real) for lsp annihilations via common
c  uses dwdcos
c  used for gaussian integration with gadap.f
c  author: joakim edsjo (edsjo@physto.se)
c  date: 98-05-03
c mod TB 2016-02-08: added OMP compatibility
c=======================================================================
```

dsankinvar.f

```
      subroutine dsankinvar(p,costheta,kp1,kp2,kpk,kp3,kp4)

c=======================================================================
c  calculate kinematical variables for s-, t-, and u-diagram
c  author: joakim edsjo, edsjo@fysik.su.se
c  date: 97-01-09
c mod TB 2016-02-08: added OMP compatibility
c mod TB 2022-04-29: added switch to extract s-wave
c=======================================================================
```

dsanset_mssm.f

```
      subroutine dsanset_mssm(c)
c...set parameters for annihilation routines
c...  c - character string specifying choice to be made
c...author: joakim edsjo, 2001-09-12
```

dsansumaa.f

```
      real*8 function dsansumaa()
c-----------------------------------------------------------------------
c  sum the amplitude matrix
c  author: joakim edsjo (edsjo@physto.se) 96-02-02
c          paolo gondolo 99-1-15 factor of 3 faster
```

```
c  called by: dwdcos
c mod TB 2016-02-08: added OMP compatibility
c=======================================================================
```

dsansumaa_swave.f

```
      real*8 function dsansumaa_swave()
c-----------------------------------------------------------------------
c  only sum helicity amplitudes (squared) that contribute to the s-wave
c  NB: The result of the sum implemented here is only the s-wave contribution
c  if the contribution from all theta-dependent wigner functions has been
c  set to zero [see use of flag wdzero in dsandwdcosnn and dsankinvar]
c  author: torsten.bringmann@fys.uio.no 2022-04-29
c  called by: dsandwdcosnn
c=======================================================================
```

dsantucc.f

```
      subroutine dsantucc(p,ind1,ind2)
c-----------------------------------------------------------------------
c  routine to check for t- or u-channel resonances.
c  called by dwdcosopt.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 97-09-17
c=======================================================================
```

dsantucn.f

```
      subroutine dsantucn(p,ind1,ind2)
c-----------------------------------------------------------------------
c  routine to check when t- or u-resonances occur.
c  called by dwdcosopt.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 97-09-17
c=======================================================================
```

dsantunn.f

```
      subroutine dsantunn(p,ind1,ind2)
c-----------------------------------------------------------------------
c  routine to check if t- or u-resonances occur for neutralino-neutralino
c  annihilation
c  called by dwdcosopt.
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 97-09-17
c=======================================================================
```

dsantures.f

```
      integer function dsantures(kp1,kp2,kp3,kp4,p)

c=======================================================================
c  determine if t- or u-resonances can occur for a given 2 ->2
c  scattering. if t_max>0 or u_max>0, then tures=1, otherwise tures=0
c  author: joakim edsjo, edsjo@fysik.su.se
c  date: 97-09-17
c=======================================================================
```

dsanwriteaa.f

```
      subroutine dsanwriteaa
c--------------------------------------------------------------------
c  write out the amplitude matrix
c  author: joakim edsjo (edsjo@fysik.su.se) 95-10-25
c  called by: different routines during debugging
c====================================================================
```

dsanwx.f

```
************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.    ***
***                                                                          ***
***  type : interface                                                        ***
***                                                                          ***
***  Input:                                                                  ***
***    p - initial cm momentum (real) for DM annihilations                   ***
***  Output:                                                                 ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  uses dsabsq.
***  passed to dsrdens by dsrdomega.                                    ***
***                                                                      ***
***  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
***  modified: joakim edsjo (edsjo@fysik.su.se) 97-09-09
***  mod TB 2016-02-08: added OMP compatibility
***  mod TB 2018-04-27, 20-10-30: caught NAN, negative output
************************************************************************
      real*8 function dsanwx(p)
```

dsanwxint.f

```
      function dsanwxint(p,a,b)
c--------------------------------------------------------------------
c  neutralino self-annihilation invariant rate integrated between
c  cos(theta)=a and cos(theta)=b.
c  input:
c    p - initial cm momentum (real) for lsp annihilations
c    integration limits a and b
c  called by wx.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c  modified slightly by joakim edsjo (edsjo@fysik.su.se) 96-04-10
c====================================================================
```

dsanyieldgammaline.f

```
************************************************************************
*** Subroutine dsanyieldgammaline returns observables related to
*** gamma ray lines (or almost lines).
*** Output
***   n =  number of gamma lines
***   nbr[n] = N_gamma * branching fraction to this channel
***              [unitless]
***   eline[n] = energy of line [GeV]
***   wline[n] = width of line [GeV]
```

```
***   pdg_second[n] = PDG code of second particle
*** NOTE: nbr, eline and wline have to be defined as real*8 with
*** dimension 10, i.e. real*8 nbr(10),eline(10),wline(10)
*** pdg_second has to be defined as integer with dimension 10.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: May, 2014
**********************************************************************

      subroutine dsanyieldgammaline(n,nbr,eline,wline,pdg_second)
```

dsanyieldline.f

```
**********************************************************************
*** function dsanyieldline returns the branching fraction for some
*** line features.
*** The calculation is done at v=0 and does NOT include gamma rays,
*** these are instead handled by dsanyieldgammaline.
*** Input:
***   pdg = PDG code for final state particle (annihilation to pdg and -pdg
***         is assumed).
***         11 e+ e-
***         12 nu_e nu_e-bar
***         14 nu_mu nu_nu-bar
***         16 nu_tau nu_tau-bar
*** Output: branching fraction to channel
*** Units: unitless
**********************************************************************

      real*8 function dsanyieldline(pdg)
```

dsdecratewimp.f

```
**********************************************************************
*** function dsdecratewimp returns the decay rate for the WIMP
*** Units of returned decay rate: s^-1
**********************************************************************

      real*8 function dsdecratewimp()
```

dssigmav0.f

```
**********************************************************************
*** function dssigmav0 returns the annihilation cross section
*** sigma v at p=0 for neutralino-neutralino annihilation into 2-body
*** final states.
***
*** input: pdg1, pdg2 -- PDG codes of final state particles
***
*** Units of returned cross section: cm^3 s^-1
***
*** author: Torsten.Bringmann.fys.uio.no
*** (based on old dssigmav with channel codes)
*** date: 2014-11-14
**********************************************************************

      real*8 function dssigmav0(pdg1,pdg2)
```

dssigmav0tot.f

```
**********************************************************************
*** function dssigmav0tot returns the *total* annihilation cross section
*** sigma v at p=0 for neutralino-neutralino annihilation.
```

```
*** This is obtained by summing over all implemented 2-body channels
*** (as returned by dssigmav) plus contributions from final states with
*** more particles.
***
***  type : interface
***
*** Units of returned cross section: cm^3 s^-1
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2014-11-14
**********************************************************************

      real*8 function dssigmav0tot()
```

39.4 mssm/an_1l: Annihilation cross sections (1-loop)

39.4.1 Annihilation cross sections at 1-loop – general

The annihilation cross sections at 1-loop that we have implemented in DarkSUSY are those to $\gamma\gamma$, $Z\gamma$ and gg. The derivation of these is described in the works [26, 27]. Let us mention that these one-loop expressions formally violate unitarity for diagrams with electroweak gauge-boson exchange because they do not take into account the nonperturbative effects related to multiple electroweak gauge boson exchange ('Sommerfeld effect') as described in [188, 189]. In practice, one can still trust the result for sub-TeV neutralinos – but should keep in mind that the implemented cross sections become unphysical in the limit $m_W/m_\chi \to 0$.

To see how these routines are called, see the file **src_models/mssm/an/dsandwdcosnn.f** where the $\gamma\gamma$, $Z\gamma$ and gg contributions are added to the annihilation cross section at the end.

39.4.2 Routine headers – fortran files

dsanggim.f

```
c====================================================================
c
c   this subroutine gives the imaginary part of the amplitude of the
c   process of neutralino annihilation into two photons in the limit
c   of vanishing relative velocity of the neutralino pair
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
c
c   imres: imaginary part
c   imfbxg: contribution from diagram 1a  divided by imres
c   imftxg: contribution from diagrams 1c & 1d  divided by imres
c   imgbxg: contribution from diagram 3a  divided by imres
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      subroutine dsanggim(imres)
```

dsanggimpar.f

```
c====================================================================
c
c   this subroutine gives the imaginary part of the amplitude of the
c   process of neutralino annihilation into two photons in the limit
c   of vanishing relative velocity of the neutralino pair
```

```
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
c
c   see header of dsanggim.f for details
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      subroutine dsanggimpar(imres,imfbx,imftx,imgbx)
```

dsanggre.f

```
c====================================================================
c
c   this subroutine gives the real part of the amplitude of the
c   process of neutralino annihilation into two photons in the limit
c   of vanishing relative velocity of the neutralino pair
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
c
c   reres: real part
c   refbxg: contribution from diagram 1a & 1b  divided by reres
c   reftxg: contribution from diagrams 1c & 1d  divided by reres
c   rehbxg: contribution from diagram 2a & 2b  divided by reres
c   rehtxg: contribution from diagrams 2c & 2d  divided by reres
c   regbxg: contribution from diagram 3a - 3c & 4a -4b divided by
c     reres
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------


      subroutine dsanggre(reres)
```

dsanggrepar.f

```
c====================================================================
c
c   this subroutine gives the imaginary part of the amplitude of the
c   process of neutralino annihilation into two photons in the limit
c   of vanishing relative velocity of the neutralino pair
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
c
c   see header of dsanggre.f for details
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      subroutine dsanggrepar(reres,refbx,reftx,rehbx,rehtx,regbx)
```

dsanglglim.f

```
c====================================================================
c
c   this subroutine gives the imaginary part of the amplitude of the
c   process of neutralino annihilation into two gluons in the limit
c   of vanishing relative velocity of the neutralino pair
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
```

```
c
c   imres: imaginary part
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c----------------------------------------------------------------------

      subroutine dsanglglim(imres)
```

dsanglglre.f

```
c======================================================================
c
c   this subroutine gives the real part of the amplitude of the
c   process of neutralino annihilation into two gluons in the limit
c   of vanishing relative velocity of the neutralino pair
c
c   l. bergstrom & p. ullio, nucl. phys. b 504 (1997) 27
c
c   reres: real part
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c----------------------------------------------------------------------

      subroutine dsanglglre(reres)
```

dsanzg.f

```
c======================================================================
c
c   this subroutine gives the real and imaginary parts of the
c   amplitude of the process of neutralino annihilation into
c   one photon and one z boson in the limit of vanishing relative
c   velocity of the neutralino pair
c
c   p. ullio & l. bergstrom, phys. rev. d 57 (1998) 1962
c
c   the present version assumes equal sfermion mass eigenstates in
c   fermion - sfermion loop diagrams
c
c   imres: imaginary part
c   imfbxz: contribution to imres from diagram 1a - 1c divided by
c     imres
c   imftxz: contribution to imres from diagrams 1e - 1h  divided by
c     imres
c   imgbxz: contribution to imres from diagram 3a & 3b  divided by
c     imres
c   reres: real part
c   refbxz: contribution to reres from diagram 1a - 1d  divided by
c     reres
c   reftxz: contribution to reres from diagrams 1e - 1h  divided by
c     reres
c   rehbxz: contribution to reres from diagram 2a - 2d  divided by
c     reres
c   rehtxz: contribution to reres from diagrams 2e - 2h  divided by
c     reres
c   regbxz: contribution from diagram 3a - 3f & 4a -4f divided by
c     reres
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c----------------------------------------------------------------------
```

```
      subroutine dsanzg(reres,imres)
```

dsanzgpar.f

```
c=====================================================================
c
c   this subroutine gives the real and imaginary parts of the
c   amplitude of the process of neutralino annihilation into
c   one photon and one z boson in the limit of vanishing relative
c   velocity of the neutralino pair
c
c   p. ullio & l. bergstrom, phys. rev. d 57 (1998) 1962
c
c   see header of dsanzg.f for details
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      subroutine dsanzgpar(imres,imfbx,imftx,imgbx,reres,refbx,reftx,
     &  rehbx,rehtx,regbx)
```

dsfl1c1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl1c1(r1,r2,r3,r4,r5)
```

dsfl1c2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl1c2(r1,r2,r3,r4,r5)
```

dsfl2c1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl2c1(r1,r2,r3,r4,r5)
```

dsfl2c2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl2c2(r1,r2,r3,r4,r5)
```

dsfl3c1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl3c1(r1,r2,r3,r4,r5)
```

dsfl3c2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl3c2(r1,r2,r3,r4,r5)
```

dsfl4c1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl4c1(r1,r2,r3,r4,r5)
```

dsfl4c2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsti_5.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsfl4c2(r1,r2,r3,r4,r5)
```

dsi_12.f

```
c=====================================================================
c
c   auxiliary function used in:
c   not used, this is here just for notation
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_12(r1,r2)
```

dsi_13.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f  dsi_13.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_13(r1,r2,r3)
```

dsi_14.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f  dsi_13.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_14(r1,r2,r3,r4)
```

dsi_22.f

```
c=====================================================================
c
c   auxiliary function used in:
c   not used, this is equivalent to dspiw2.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_22(r1,r2)
```

dsi_23.f

```
      real*8 function dsi_23(r1,r2,r3)
No header found.
```

dsi_24.f

```
      real*8 function dsi_24(r1,r2,r3,r4)
No header found.
```

dsi_32.f

```
c=====================================================================
c
c   auxiliary function used in:
c   not used, this is equivalent to dspiw3.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_32(r1,r2)
```

dsi_33.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_33(r1,r2,r3)
```

dsi_34.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f  dsi_32.f  dsi_33.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_34(r1,r2,r3,r4)
```

dsi_41.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsi_41(a,b,c)
```

dsi_42.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------
```

```
      real*8 function dsi_42(a,b,d,c)
```

dsilp2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dslp2.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsilp2(x)
```

dsj_1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsj_1(a,b)
```

dsj_2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsj_2(b,c)
```

dsj_3.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsj_3(a,b,c)
```

dslp2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_14.f  dsi_24.f  dsi_34.f  dsilp2.f dsti_5.f
```

```
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dslp2(c1,c2)
c this function compute the integral between 0 and 1 of 1/x*log(1+c1*x+c2*x**2)
```

dspi1.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanggrepar.f  dsanglglre.f  dsi_12.f  dsrepfbox.f  dsrepgh.f  dsrepw.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dspi1(a,b)
```

dspiw2.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanglglre.f  dsrepfbox.f  dsrepgh.f  dsrepw.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dspiw2(a,b)
```

dspiw2i.f

```
c====================================================================
c
c   auxiliary function used in:
c   dspiw2.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dspiw2i(x)
```

dspiw3.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanglglre.f  dsrepfbox.f  dsrepgh.f  dsrepw.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dspiw3(a,b)
```

dspiw3i.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dspiw3.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dspiw3i(x)
```

dsrepfbox.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanggrepar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsrepfbox(a,b,sq,dq,signm)
```

dsrepgh.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanggrepar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsrepgh(a,b,sq,dq,signm)
```

dsrepw.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanggrepar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsrepw(a,b,sq,dq,signm)
```

dsslc1.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_14.f  dsi_24.f  dsi_34.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------
```

```
      real*8 function dsslc1(r1,r2,r3)
c this function gives the coefficient c1 of slog
```

dsslc2.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_14.f  dsi_24.f  dsi_34.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      real*8 function dsslc2(r1,r2,r3)
c this function gives the coefficient c2 of slog
```

dssubka.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_41.f  dsi_42.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------


      real*8 function dssubka(r,delta)
```

dssubkb.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_41.f  dsi_42.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      subroutine dssubkb(r1,r2,delta,res1,res2)
```

dssubkc.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsi_42.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c---------------------------------------------------------------------

      subroutine dssubkc(r1,delta,res1,res2)
```

dsti_214.f

```
c=====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
```

```
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dsti_214(r1,r2,r3,r4)
```

dsti_224.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dsti_224(r1,r2,r3,r4)
```

dsti_23.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dsti_23(r1,r2,r3)
```

dsti_33.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsanzgpar.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dsti_33(r1,r2,r3)
```

dsti_5.f

```
c====================================================================
c
c   auxiliary function used in:
c   dsti_214.f  dsti_224.f  dsti_23.f dsti_33.f
c
c   author: piero ullio (piero@tapir.caltech.edu)
c
c--------------------------------------------------------------------

      real*8 function dsti_5(r1,r2,r3,r4,r5)
```

39.5 mssm/an_ib: Internal bremsstrahlung

39.5.1 Internal Bremsstrahlung (IB) – theory

Whenever WIMPs annihilate into pairs of charged particles $X\bar{X}$, this process will inevitably be accompanied by internal bremsstrahlung (IB), i.e. the emission of an additional photon in the final state (note that in contrast to ordinary, or external, bremsstrahlung no external electromagnetic field is required for the emission of the photon). In many cases, IB photons completely dominate the annihilation spectrum at the highest accessible energies. The resulting, characteristic sharp step at an energy corresponding to the dark matter particle's mass, often accompanied by a bump-like feature at slightly smaller energies, is a spectral signature that is hard to mimic by astrophysical sources and in that respect similar to monochromatic photons. In fact, being an $\mathcal{O}(\alpha_{\rm em})$ correction to the tree-level annihilation rate, one would generically expect IB photons to be more copiously produced than the loop-suppressed [i.e. $\mathcal{O}(\alpha_{\rm em}^2)$] monochromatic photons; this has been confirmed in [28] for a large part of the supersymmetric parameter space, not the least due to the appearance of efficient enhancement mechanisms.

39.5.1.1 General considerations

For didactic purposes, one may follow [28] and distinguish between photons directly radiated off the external legs (*final state radiation*, FSR) and photons radiated from virtual charged particles (*virtual internal bremsstrahlung*, VIB). The IB photons are thus the total contribution from both FSR and VIB photons.

For relativistic charged final states, FSR diagrams are always dominated by photons emitted *collinearly* with X or $\bar{X}$. This is a purely kinematical effect and related to the fact that the propagator of the corresponding outgoing particle,

$$D(p) \propto \left((k+p)^2 - m_X^2\right)^{-1}, \tag{39.2}$$

diverges in this situation. Here, k and p denote the momenta of the photon and the outgoing particle, respectively. The resulting photon spectrum turns out to be of a universal form, almost independent of the underlying particle physics model (see, e.g., [190]):

$$\frac{dN_{\gamma,\rm FSR}^{X\bar{X}}}{dx} \approx \frac{\alpha Q_X^2}{\pi} \mathcal{F}_X(x) \log\left(\frac{s(1-x)}{m_X^2}\right). \tag{39.3}$$

Here, Q_X and m_X are the electric charge and mass of X; the splitting function $\mathcal{F}(x)$ depends only on the spin of the final state particles and takes the form

$$\mathcal{F}_{\rm fermion}(x) = \frac{1+(1-x)^2}{x} \tag{39.4}$$

for fermions and

$$\mathcal{F}_{\rm boson}(x) = \frac{1-x}{x} \tag{39.5}$$

for bosons. Due to the logarithmic enhancement that becomes apparent in Eq. (39.3), FSR photons are often the main source for IB. A prominent example where FSR in this universal form not only dominates IB but in fact the total gamma-ray spectrum from WIMP annihilations, is the case of Kaluza-Klein dark matter [191].

In general, one can single out two situations where photons emitted from virtual charged particles may give an even more important contribution to the total IB spectrum than FSR: i) the three-body final state $X\bar{X}\gamma$ satisfies a symmetry of the initial state that cannot be satisfied by the two-body final state $X\bar{X}$ or ii) X is a boson and the annihilation into $X\bar{X}$ is dominated by t-channel diagrams,

with the t-channel particle almost degenerate in mass with the annihilating WIMP. In contrast to FSR, the contribution from VIB photons can not be given in a model-independent way but is very sensitive to the underlying short-distance physics. For more details, see [28].

39.5.1.2 IB from neutralino annihilations

For supersymmetric dark matter annhilations, the relevant final states for IB are W^+W^-, $W^\pm H^\mp$, H^+H^- and $f\bar{f}$; both of the situations just mentioned above can arise, and *VIB* contributions become important in considerable regions of the parameter space.

Let us first note that for neutralino annihilations, in contrast to the situation for, e.g., Kaluza-Klein dark matter, we cannot in general expect very large *FSR* contributions. This is because the lightest charged final states, for which the logarithmic enhancement shown in Eq. (39.3) would be most effective, are fermionic and therefore strongly helicity suppressed. Fermion final states containing an additional photon, however, are not subject to such a suppression [192]. In the limit of vanishing fermion mass, and assuming that both corresponding sfermions have the same mass, the photon multiplicity is given by [28]

$$\frac{\mathrm{d}N^{f^+f^-}}{\mathrm{d}x} = \alpha_{\mathrm{em}} Q_f^2 \frac{|\tilde{g}_R|^4 + |\tilde{g}_L|^4}{64\pi^2} \left(m_\chi^2 \langle\sigma v\rangle_{\chi\chi\to f\bar{f}} \right)^{-1} \tag{39.6}$$
$$\times (1-x) \left\{ \frac{4x}{(1+\mu)(1+\mu-2x)} - \frac{2x}{(1+\mu-x)^2} - \frac{(1+\mu)(1+\mu-2x)}{(1+\mu-x)^3} \log\frac{1+\mu}{1+\mu-2x} \right\},$$

where $\mu \equiv m_{\tilde{f}_R}^2/m_\chi^2 = m_{\tilde{f}_L}^2/m_\chi^2$ and $\tilde{g}_R P_L$ ($\tilde{g}_L P_R$) denotes the coupling between neutralino, fermion and right-handed (left-handed) sfermion. In the above expression, a large factor m_χ^2/m_f^2 due to the lifted helicity suppression (from $\langle\sigma v\rangle_{\chi\chi\to f\bar{f}} \propto m_f^2 m\chi^{-4}$) appears, and another enhancement at high photon energies for sfermions degenerate with the neutralino.

For large neutralino masses $m_\chi \gg m_W$ and charginos almost degenerate with the neutralino, W^+W^- and $W^\pm H^\mp$ final states are affected by the second of the mechanisms for VIB enhancement that were discussed in the previous subsection. Of these two channels, IB from W^+W^- nearly always dominates; for pure Higgsinos (or Winos), the resulting photon multiplicity in this limit is well approximated by [28, 193]:

$$\frac{\mathrm{d}N^{W^+W^-}}{\mathrm{d}x} \approx \frac{\alpha_{\mathrm{em}}}{\pi} \frac{4(1-x+x^2)^2}{(1-x+\epsilon/2)x} \left[\log\left(2\frac{1-x+\epsilon/2}{\epsilon} \right) - 1/2 + x - x^3 \right], \tag{39.7}$$

where $\epsilon \equiv m_W/m_\chi$.

Charged Higgs pairs H^+H^-, finally, provide yet another interesting example where the two-body final state is not allowed due to symmetry restrictions; in this case, in the limit $v \to 0$, enforced by CP conservation. The annihilation into $H^+H^-\gamma$, on the other hand, *is* possible. However, since charged Higgs bosons in most models have considerably larger masses than gauge bosons, they are expected to give negligible IB contributions compared to the latter.

39.5.1.3 The implementation in DarkSUSY

Let us now briefly describe how IB is implemented in DarkSUSY. The total gamma-ray spectrum from WIMP annihilations is given by

$$\frac{dN_{\gamma,\mathrm{tot}}}{dx} = \sum_f B_f \left(\frac{dN_{\gamma,\mathrm{sec}}^f}{dx} + \frac{dN_{\gamma,\mathrm{IB}}^f}{dx} + \frac{dN_{\gamma,\mathrm{line}}^f}{dx} \right), \tag{39.8}$$

where B_f denotes the branching ratio into the annihilation channel f. The first term encodes the contribution from secondary photons, mainly produced through the decay of neutral pions,

as described in Chapter 14. We recall here that these contributions are included by using the Monte Carlo code Pythia [10] to simulate the decay of a hypothetical particle with mass $2m_\chi$ and user-specified branching ratios B_f. In this way, also FSR associated to this decay is automatically included in $dN^f_{\gamma,\mathrm{sec}}/dx$ (the main contribution here comes from photons directly radiated off the external legs, but also photons radiated from other particles in the decay cascade are taken into account).

Of course, IB from the decay of such a hypothetical particle cannot in general be expected to show the same characteristics as IB from the actual annihilation of two WIMPs and for this reason an additional term $dN^f_{\gamma,\mathrm{IB}}/dx$ is included that accounts for the difference between the full IB contribution and the FSR part already taken into account for by Pythia. For details regarding the implemented procedure of separating these two contributions in a consistent way, see [28]. The contributions $dN^f_{\gamma,\mathrm{IB}}/dx$, in contrast to $dN^f_{\gamma,\mathrm{sec}}/dx$, are generically highly model-dependent. At the moment, they are fully implemented only for neutralino annihilations and included by default. However, one may easily switch to a user-defined contribution, choose to neglect IB completely or, for comparison, to only include the FSR part by replacing the call to dsibyield in mssm/an_yield/dsanyield. For example, a version of dsibyield containing only the FSR part of the yield is provided in mssm/an_ib/dsanyield_fsr.

For the supersymmetric case, the full expressions for all relevant three-body final states are implemented, i.e. not just the approximations given in the last subsection, which only apply to the limits described there. For performance reasons, IB is only included when virtual t-channel particles are sufficiently degenerate in mass that large IB contributions to the total spectrum can be expected. With a call to **dsIBset**, this default behaviour can be customized. Finally, radiative corrections to the annihilation into charged particles $X\bar{X}$ of course also change the number of $X\bar{X}$ pairs per annihilation and thus the corresponding yield of particles in the further decay of the annihilation products. At the moment, apart from the photon yield, only the IB positron yield is implemented in DarkSUSY. By default, only the annihilation into the dominating channel $e^+e^-\gamma$ is taken into account in this case; again, this behaviour can be modified by a call to **dsIBset**.

39.5.2 Routine headers – fortran files

dsIBf_intdE.f

```
************************************************************************
*** computes the envelope of the yield in channel IBch with the yield of
*** type yieldk due to the further decay of the particles in IBch
***
*** author: Torsten Bringmann (troms@physto.se)
*** date  : 2007-11-20
*** update: 2008-03-10 - better numerical convergence
*** update: 2016-04-12 - added light quark channels
************************************************************************

      real*8 function dsIBf_intdE(IBch,yieldk,x,mwimp,mp1,mp2)
```

dsIBf_intdxdy.f

```
************************************************************************
*** integrates a function f(x,y) over the kinematic variables
***
*** x = E_gamma/mx
*** y = (p+k)^2/(4 mx^2)
***
*** that appear for 3-body final states. (k is the photon momentum and p
*** that of one of the other final sates). The result is the
*** total photon yield per annihilation
```

```
***
*** called by dsIByield
***
*** based on paolo gondolos wxint.f routine.
*** author: Torsten Bringmann 2007-04-18
************************************************************************
      real*8 function dsIBf_intdxdy(IBch,x,mwimp,mp1,mp2)
```

dsIBf_intdxdy2.f

```
************************************************************************
*** integrates a function f(z,y) over the kinematic variables
***
*** z = E_p/mx
*** y = (p+k)^2/(4 mx^2)
***
*** that appear for 3-body final states. (k is the photon momentum and p
*** that of one of the other final sates). The result is the
*** total p yield per annihilation
***
*** called by dsIByieldone
***
*** author: Torsten Bringmann (troms@physto.se)
*** date  : 2007-10-20
************************************************************************
      real*8 function dsIBf_intdxdy2(IBch,z,mwimp,mp1,mp2)
```

dsIBf_intdy.f

```
************************************************************************
*** integrates a function f(x,y) over the kinematic variable
***
*** y = (p+k)^2/(4 mx^2)
***
*** that appears for 3-body final states. (k is the photon momentum and p
*** that of one of the other final sates). The result is the
*** (dimensionless) differential photon yield per annihilation
***
*** called by dsIByield
***
*** author: Torsten Bringmann (bringman@sissa.it)
*** date:   2007-04-18
*** update: 2008-03-10 - better numerical convergence
************************************************************************

      real*8 function dsIBf_intdy(IBch,x,mwimp,mp1,mp2)
```

dsIBf_intdy2.f

```
************************************************************************
*** integrates a function f(z,y) over the kinematic variable
***
*** y = (p+k)^2/(4 mx^2)
***
*** that appears for 3-body final states. (k is the photon momentum and p
*** that of one of the other final sates). The result is the
*** (dimensionless) differential p yield per annihilation
***
*** called by dsIByieldone
***
*** author: Torsten Bringmann (troms@physto.se)
*** date:   2007-10-20
```

```
*** update: 2008-03-10 - better numerical convergence
**************************************************************************

      real*8 function dsIBf_intdy2(IBch,z,mwimp,mp1,mp2)
```

dsIBffdxdy.f

```
**************************************************************************
*** The function dsIBffdxdy gives the full analytical expressions for
*** the differential IB photon yield (dxdy) from fermion final states,
*** normalized to the annihilation rate into fermion pairs f fbar
***
*** The kinematic variables x,y are
***
*** x = E_gamma/mx
*** y = (p+k)^2/(4 mx^2),
***
*** where p denotes the fermion momentum and k the photon momentum.
*** (note that the expressions above and below only apply to the v->0 limit)
***
*** author: Torsten Bringmann (bringman@sissa.it)
*** date: 2007-07-05
*** update: 2016-04-04 (switched to analytic tree-level expressions to avoid
***                     recursive call to dssigmav0)
***         2020-10-23 bug-fix for light fermions (corrected FSR-subtraction)
**************************************************************************

      real*8 function dsIBffdxdy(IBch,x,y)
```

dsIBffdxdy_1.f

```
**************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
**************************************************************************

      real*8 function dsIBffdxdy_1(x,y,m0,ml,msl1,msl2)
```

dsIBffdxdy_2.f

```
**************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
**************************************************************************

      real*8 function dsIBffdxdy_2(x,y,m0,ml,msl1,msl2)
```

dsIBffdxdy_3.f

```
**************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
**************************************************************************

      real*8 function dsIBffdxdy_3(x,y,m0,ml,msl1,msl2)
```

dsIBffdxdy_4.f

```
**************************************************************************
```

```
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
************************************************************************

      real*8 function dsIBffdxdy_4(x,y,m0,ml,msl1,msl2)
```

dsIBffdxdy_5.f

```
************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
************************************************************************

      real*8 function dsIBffdxdy_5(x,y,m0,ml,msl1,msl2)
```

dsIBffdxdy_6.f

```
************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
************************************************************************

      real*8 function dsIBffdxdy_6(x,y,m0,ml,msl1)
```

dsIBffdxdy_7.f

```
************************************************************************
*** auxiliary routine called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
************************************************************************

      real*8 function dsIBffdxdy_7(x,y,m0,ml,msl1)
```

dsIBffdxdy_8.f

```
************************************************************************
*** called by dsIBffdxdy
*** author: Torsten Bringmann, 2007-07-05
************************************************************************

      real*8 function dsIBffdxdy_8(x,y,CZ5,CH,m0,ml,mz,mh03,GZ,Gh03)
```

dsIBfsrdxdy.f

```
************************************************************************
*** The function dsIBfsrdxdy gives the full analytical expressions for
*** the differential FSR photon yield (dxdy) from fermion final states,
*** normalized to the annihilation rate into fermion pairs f fbar.
*** This contribution is already included in the Pythia results and hence has
*** to be subtracted when adding the full IB results. Note that Pythia does
*** not include FSR contributions from bosonic final states.
***
*** FSRch denotes the annihilation channels (see dsIByieldone_fsr.f)
***
*** the kinematical input variables are
*** x  = E_gamma/m0
*** y  = (p+k)^2/(4 mx^2),
***
*** where p denotes the fermion and k the photon momentum.
*** (note that the expressions above and below only apply to the v->0 limit)
***
```

```
*** author: Torsten Bringmann (troms@physto.se)
*** date: 2008-02-10

********************************************************************************

      real*8 function dsIBfsrdxdy(FSRch,x,y)
```

dsIBhhdxdy.f

```
********************************************************************************
*** The function dsIBhhdxdy gives the full analytical expressions for
*** the differential IB photon yield (dxdy) from H+H- final states,
*** normalized to the  *total* neutralino annihilation rate
***
*** The kinematic variables x,y are
***
*** x = E_gamma/mx
*** y = (p+k)^2/(4 mx^2),
***
*** where p denotes the H+ momentum and k the photon momentum.
*** (note that the expressions above and below only apply to the v->0 limit;
***  in this limit the lowest order result vanishes)
***
*** author: Torsten Bringmann (bringman@sissa.it)
*** date: 2007-05-01
********************************************************************************

      real*8 function dsIBhhdxdy(IBch,x,y)
```

dsIBintsel.f

```
********************************************************************************
*** auxiliary function that selects integrand for integration routines
*** author: Torsten Bringmann 2008-03-10
********************************************************************************

      real*8 function dsIBintsel(yint)
```

dsIBintsel2.f

```
********************************************************************************
*** auxiliary function that selects integrand for integration routines
*** author: Torsten Bringmann 2008-03-12
*** update 2014-11-12: changed call to dsanyield_sim to PDG codes
*** update 2016-04-12: added light quark channels
********************************************************************************

      real*8 function dsIBintsel2(xint)
```

dsIBselect.f

```
************************************************************************
*** Routine dsIBselect selects which channels to include based on the
*** mass degeneracies. Used when ibhow=2.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2007-10-19
************************************************************************
```

```
      subroutine dsibselect
```

dsIBset.f

```
************************************************************************
*** Routine dsibset sets default settings for internal bremsstrahlung
*** (IB) contributions to (total) yield calculations.
***
***  c - character string specifying choice to be made
***  (currently for photon and positron yield only)
***  possible values for c:
***  'accurate'             - all channels, high prec. integration
***  'medium'               - all channels, med. prec. integration
***  'fast'                 - most important channels, med. prec. integration
***  'dynamic' or 'default' - decide model by model which channels to include,
***                           med acc of integration
***  'off'                  - no IB contribution
***
*** author: joakim edsjo, 2007-10-19
*** positron switches added: torsten bringmann, 2008-02-29
************************************************************************

      subroutine dsibset(c)
```

dsIBwhdxdy.f

```
************************************************************************
*** The function dsIBwhdxdy gives the full analytical expressions for
*** the differential IB photon yield (dxdy) from W+H- and W-H+ final states,
*** normalized to the  annihilation rate into W+H- and W-H+
***
*** The kinematic variables x,y are
***
*** x = E_gamma/mx
*** y = (p+k)^2/(4 mx^2),
***
*** where p denotes the W momentum and k the photon momentum.
*** (note that the expressions above and below only apply to the v->0 limit)
***
*** author: Torsten Bringmann (bringman@sissa.it)
*** date: 2007-05-01
************************************************************************


      real*8 function dsIBwhdxdy(IBch,x,y)
```

dsIBwhdxdy_1.f

```
************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
************************************************************************

      real*8 function dsIBwhdxdy_1(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwhdxdy_2.f

```
************************************************************************
*** auxiliary routine called by dsIBwhdxdy
```

```
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_2(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwhdxdy_3.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_3(x,y,m0,mw,mhc,mc1)
```

dsIBwhdxdy_4.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_4(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwhdxdy_5.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_5(x,y,m0,mw,mhc,mc1)
```

dsIBwhdxdy_6.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_6(x,y,m0,mw,mhc)
```

dsIBwhdxdy_7.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_7(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwhdxdy_8.f

```
********************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
********************************************************************************

      real*8 function dsIBwhdxdy_8(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwhdxdy_9.f

```
************************************************************************
*** auxiliary routine called by dsIBwhdxdy
*** author: Torsten Bringmann, 2007-04-18
************************************************************************

      real*8 function dsIBwhdxdy_9(x,y,m0,mw,mhc,mc1,mc2)
```

dsIBwwdxdy.f

```
************************************************************************
*** The function dsIBwwdxdy gives the full analytical expressions for
*** the differential IB photon yield (dxdy) from W+W- final states,
*** normalized to the  annihilation rate into W+W-
***
*** The kinematic variables x,y are
***
*** x = E_gamma/mx
*** y = (p+k)^2/(4 mx^2),
***
*** where p denotes the W+ momentum and k the photon momentum.
*** (note that the expressions above and below only apply to the v->0 limit)
***
*** author: Torsten Bringmann (bringman@sissa.it)
*** date: 2007-05-01
************************************************************************

      real*8 function dsIBwwdxdy(IBch,x,y)
```

dsIBwwdxdy_1.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_1(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_2.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_2(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_3.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_3(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_4.f

```
************************************************************************
```

```
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_4(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_5.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_5(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_6.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_6(x,y,m0,mw,mc1,mc2)
```

dsIBwwdxdy_7.f

```
************************************************************************
*** auxiliary routine called by dsIBwwdxdy
*** author: Torsten Bringmann, 2007-02-16
************************************************************************

      real*8 function dsIBwwdxdy_7(x,y,m0,mw,mh01,mh02)
```

dsIByield.f

```
************************************************************************
***   function dsIByield gives the yield from internal
***   bremsstrahlung (IB) for SUSY models.
***   Input: egev - energy in GeV
***          yieldk - which yield to calculate
***             (see dsanyield for an explanation,
***              currently only photon and positron yield are implemented)
***   Output: yield (annihilation)**1, differential also GeV**-1
***           istat is set as follows in case of errors
***   bit  decimal  reason
***     0        1  dsIBf_intdy (integration for photon yield) failed
***     1        2  dsIBf_intdy2 (integration for positron yield) failed
***                              -- for direct annihilation into positrons (channel eeg)
***     2        4  dsIBf_intdy2 (integration for positron yield) failed
***                              -- for annihilation channel different from eeg
*** Author: Joakim Edsjo, edsjo@fysik.su.se
***         Torsten Bringmann, bringman@sissa.it
*** Date: 2008-01-15
************************************************************************

      real*8 function dsIByield(egev,yieldk,istat)
```

dsIByield_fsr.f

```
************************************************************************
***   function dsIByield_fsr gives the photon yield from final state radiation
```

```
***   (FSR) from the deacy of a hypothetical partical with mass 2*m0, such as
***   included in the Pythia runs. Just like in Pythia, only FSR from fermionic
***   final states is included
***   Input: egev - energy in GeV
***          yieldk - 52 for integrated or 152 for differential yield
***   Output: yield (annihilation)**1, differential also GeV**-1
***           istat is set as follows in case of errors
***   bit  decimal  reason
***     0        1  dsIBf_intdxdy failed
***     1        2  dsIBf_intdy failed
*** Author: Torsten Bringmann, troms@physto.se
*** Date: 2008-02-10
**************************************************************************

      real*8 function dsIByield_fsr(egev,yieldk,istat)
```

dsIByieldone.f

```
**************************************************************************
*** function dsIByieldone calculates the IB yield from one given
*** annihilation channel.
***
*** Currently included are:
***   yieldk =  51 - positron yield above threshold emuthr
***             151 - differential positron yield at emuthr
***              52 - photon yield above threshold emuthr
***             152 - differential photon yield at emuthr
***
*** The annihilation channels are:
***  IBch =  1 - W+W-
***          2 - W+H- and W-H+
***          3 - H+H-
***          4 - e+e-
***          5 - mu+mu-
***          6 - tau+tau-
***          7 - u u-bar
***          8 - d d-bar
***          9 - c c-bar
***         10 - s s-bar
***         11 - t t-bar
***         12 - b b-bar
*** Note that FSR photons from quark (and tau) final states are already
*** implicitly included in the fragmentation functions implemented
*** in dsanyield. Hence that contribution is subtracted here (from an
*** analytical calculation).
*** See arXiv: 0710.3169 (hep-ph) for more details.
***
*** the units are (tree-level annihilation into IBch)**-1
*** for the differential yields, the units are the same times GeV**-1.
***
*** istat will set upon return in case of errors
***   bit  decimal  reason
***     0        1  dsIBf_intdxdy failed
***     1        2  dsIBf_intdy failed
*** author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** date: 2007-05-01
*** mod: 2015-10-26 (assume that all couplings are
***                  already calculated, e.g. by previous call of dssigmav0!)
**************************************************************************

      real*8 function dsIByieldone(emuthr,IBch,yieldk,istat)
```

dsIByieldone_fsr.f

```
************************************************************************
*** function dsIByieldone_fsr analytically calculates the FSR photon
*** yield above threshold (yieldk=52) or the differential yield (yieldk=152)
*** from one given annihilation channel
***
*** The annihilation channels are:
***  FSRch = 4 - e+e-
***          5 - mu+mu-
***          6 - tau+tau-
***          7 - u u-bar
***          8 - d d-bar
***          9 - c c-bar
***         10 - s s-bar
***         11 - t t-bar
***         12 - b b-bar
*** Note that FSR photons from fermion final states are already
*** implicitly included in the fragmentation functions implemented
*** in dsanyield.
***
*** the units are (annihilation into IBch)**-1
*** for the differential yields, the units are the same times gev**-1.
***
*** author: Torsten Bringmann (troms@physto.se)
*** date: 2008-02-10
************************************************************************

      real*8 function dsIByieldone_fsr(mwimp,emuthr,FSRch,yieldk,istat)
```

39.6 mssm/an_ib2: Internal bremsstrahlung of Higgs and $SU(2)$ gauge bosons

39.6.1 Routine headers – fortran files

dsib2BRtree.f

```
************************************************************************
*** function dsIB2BRtree returns the branching ratio for the decay rate
*** of possible on-shell states into fermions (at tree level).
***
***   Input: decay - string that describes the decay process (see below)
***             kf - particle code of one of the final state fermions
***                  (integer; see description in dsib2sigmav or dmssm.h)
***
***   Output: partial decay rate at tree level, divided by total width
***           (not necessarily at tree level!)
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-12-15, update 2105-03-22 (streamlined handling of k2)
*** MG 2015-01-20: added k2 as argument
***                output for ff=k2,k2 or fF=k2,k2+1, resp.
***                sign flip in matrix element for V->ff (see comments)
************************************************************************

      real*8 function dsIB2BRtree(decay,kf)
```

dsib2chinit.f

```
************************************************************************
*** subroutine dsIB2chinit initializes masses and other relevant
*** parameters for a given IB2 channel and writes them to common block
```

```
*** variables in dsib2com.h
***
***   Input: IB2ch  - 3-body channel,
***                   see dsIByieldone or dsib2sigmav for details
***
***   Output: err=1 for unknown channel IB2ch, err=0 otherwise
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2015-11-19
**********************************************************************

      subroutine dsib2chinit(IB2ch,err)
```

dsib2convint.f

```
**********************************************************************
*** auxiliary function that provides integrand for convolution of 3-body
*** final state particle distribution and MC fragmentation functions
***
*** Author: Torsten Bringmann, 2012-12-03 (2015-11_19: added Higgs final states)
***         2016-02-05 updated to DS6 conventions
**********************************************************************

      real*8 function dsIB2convint(zint)
```

dsib2dnde.f

```
**********************************************************************
*** function dsIB2dnde returns the normalized energy spectrum of a given
*** final state particle:
```

$$\frac{dN}{dE} \equiv \frac{1}{(\sigma v)_f} \frac{d(\sigma v)_f}{dE_P}$$

```
***
***   Input: IB2chinput  - selects channel f for 3-body final state:
***                        1XX for ffZ
***                        2XX for fFW
***                        3XX for ffh (XX like for ffZ)
***                        4XX for ffH (XX like for ffZ)
***                        5XX for ffA (XX like for ffZ)
***                        6XX for ffH+- (XX like for fFW)
***
***                    XX   | \bar f f Z      |   \bar f f W
***                   ------+-----------------+------------------
***                    01   | nu_e   nu_e   Z |   e+     nu_e   W-
***                    02   | e      e      Z |   nu_e   e-     W+
***                    03   | nu_mu  nu_mu  Z |   mu+    nu_mu  W-
***                    04   | mu     mu     Z |   nu_mu  mu-    W+
***                    05   | nu_tau nu_tau Z |   tau+   nu_tau W-
***                    06   | tau    tau    Z |   nu_tau tau-   W+
***                    07   | u      u      Z |   dbar   u      W-
***                    08   | d      d      Z |   ubar   d      W+
***                    09   | c      c      Z |   sbar   c      W-
***                    10   | s      s      Z |   cbar   s      W+
***                    11   | t      t      Z |   bbar   t      W-
***                    12   | b      b      Z |   tbar   b      W+
***                    (note that this numbering must be consistent
***                     with dmssm.h !!! )
***
***          pfinal - type of considered final state particle:
***                   'f' for fermion
***                   'fbar' for antifermion
```

```
***                   'B' for vectorboson or scalar
***          Efinal - energy of considered final state particle [in GeV]
***
***   Output: differential energy spectrum [1/GeV]
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-12-07
**********************************************************************

      real*8 function dsib2dnde(IB2chinput,pfinal,Efinal)
```

dsib2dsde_aux.f

```
**********************************************************************
*** Function dsIB2dsde_aux returns the full SUSY differential cross
*** section, as a function of one of the 3-body final state particles P.
*** WARNING: If you want to call this function, you need to call
***          dsib2chinit() first to specify the annihilation channel
***          and set common block c_pfinal to choose the particle P
***          ('B', 'f' and 'fbar' currently implemented)
***
***   Input: zint  - energy of particle c_pfinal, divided by mx
***
***   Output: (d(sv)_{3-body}/dz) as function of z = E_P/mx.
***           [units: 1/GeV**2]
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-03-05
**********************************************************************
      real*8 function dsIB2dsde_aux(zint)
```

dsib2dsde_auxfsr.f

```
**********************************************************************
*** Function dsIB2dndz_fsr returns the energy distribution of one of
*** the 3-body final state particles, implementing the model-
*** independent result for final-state radiation from 1104.2996 (1009.0224)
*** WARNING: If you want to call this function, you need to call
***          dsib2chinit() first to specify the channel.
***
***   Input: pconv - final state particle type
***                  ('B', 'f' and 'fbar' currently implemented)
***          zint  - energy of particle pconv, divided by mx
***          Vtype - 1 for Z, 2 for W
***          ch    - fermion pair as defined in dsIB2yieldone
***
***   Output: (d(sv)_{3-body}/dz)
***
*** Author: Francesca Calore, 2012-09-12
**********************************************************************
      real*8 function dsIB2dsde_auxfsr(zint,pconv, Vtype, ch)
```

dsib2ffA0amps.f

```
**********************************************************************
*** subroutine dsib2ffA0amps calculates helicity amplitudes for fFA final
*** states with a neutral, CP-odd scalar.
***
***   Input: f                    - final state fermion
***                                 (see header of dsib2sigmav)
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
```

```
      subroutine dsib2ffA0amps(f)
```

dsib2ffA0sFSR.f

```
**********************************************************************
*** subroutine dsib2ffH3sFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel FSR diagrams.
*** The H3 is emitted from the final legs of the s channel diagrams.
*** H3 and Z diagrams are considered separately and then
*** added to the corresponding entries of amp(0, 0:3).
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0sFSR(f, leg)
```

dsib2ffA0sISR.f

```
**********************************************************************
*** subroutine dsib2ffH3sISR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel ISR diagrams.
*** H3 emitted from the initial legs of the  s channel diagrams.
*** The exchanged particle index ineu stays for neutralino.
***
*** Notice:  ISR1 = ISR2 ->  Each amplitudes is multiplied by 2
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0sISR(f, ineu)
```

dsib2ffA0sVIB.f

```
**********************************************************************
*** subroutine dsib2ffH3sVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel VIB diagrams.
*** The H3 is emitted from the mediators (H3, Z), becoming H_{i}, i= 1, 2. H_{i}.
*** The sum is performed only over ih.
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0sVIB(f, ih)
```

dsib2ffA0tuFSR.f

```
**********************************************************************
*** subroutine dsib2ffH3tuFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel FSR diagrams.
*** The H3 is emitted by final legs 1 (FSR1) or 2 (FSR2) of the t+u channel diagrams.
*** isf1 is the index if the exchanged sfermion in th t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0tuFSR(f, leg, isf1)
```

dsib2ffA0tuISR.f

```
**********************************************************************
*** subroutine dsib2ffH3tuISR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** H3 emitted by initial legs 1 (ISR1) or 2 (ISR2) the neutralino
*** radiates a H3 by becoming a "second" neutralino - ineu -
```

```
*** isf1 is the index if the exchanged sfermion in the t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0tuISR(f, leg, isf1, ineu)
```

dsib2ffA0tuVIB.f

```
**********************************************************************
*** subroutine dsib2ffH3tuVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel VIB diagrams.
*** H3 emitted by internal legs: the sfermion - isf1 - radiates a Z by becoming a "second" sfermion - isf2 -
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffA0tuVIB(f, isf1, isf2)
```

dsib2ffH0amps.f

```
**********************************************************************
*** subroutine dsib2ffH0amps calculates helicity amplitudes for fFH final
*** states with neutral, CP-even scalars.
***
***   Input: f                    - final state fermion
***                                 (see header of dsib2sigmav)
***          hi=1,2               - SM Higgs (1) or heavy Higgs (2)
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffH0amps(f,hi)
```

dsib2ffH0sFSR.f

```
**********************************************************************
*** subroutine dsib2ffHisFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel FSR diagrams.
*** The Hi is emitted from the final legs of the s channel diagrams.
*** Hi and Z diagrams are considered separately and then
*** added to the corresponding entries of amp(0, 0:3).
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffH0sFSR(f, leg, hi)
```

dsib2ffH0sISR.f

```
**********************************************************************
*** subroutine dsib2ffHisISR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel ISR diagrams.
*** Hi emitted from the initial legs of the  s channel diagrams.
*** The exchanged particle index ineu stays for neutralino.
***
*** Notice:  ISR1 = ISR2 ->  Each amplitudes is multiplied by 2
***
*** Author: Francesca Calore, 2014-02-20
**********************************************************************
      subroutine dsib2ffH0sISR(f, ineu, hi)  ! hi refers to the Hi radiated off
```

dsib2ffH0sVIB.f

```
*****************************************************************
*** subroutine dsib2ffHisVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel VIB diagrams.
***
*** Author: Francesca Calore, 2014-02-20
*****************************************************************
      subroutine dsib2ffH0sVIB(f, ih) ! ih refers to the Hi radiated off
```

dsib2ffH0tuFSR.f

```
*****************************************************************
*** subroutine dsib2ffHituFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel FSR diagrams.
*** The Hi is emitted by final legs 1 (FSR1) or 2 (FSR2) of the t+u channel diagrams.
*** isf1 is the index if the exchanged sfermion in th t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
*****************************************************************
      subroutine dsib2ffH0tuFSR(f, leg, isf1, hi)
```

dsib2ffH0tuISR.f

```
*****************************************************************
*** subroutine dsib2ffHituISR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** Hi emitted by initial legs 1 (ISR1) or 2 (ISR2) the neutralino
*** radiates a H3 by becoming a "second" neutralino - ineu -
*** isf1 is the index if the exchanged sfermion in the t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
*****************************************************************
      subroutine dsib2ffH0tuISR(f, leg, isf1, ineu, hi)
```

dsib2ffH0tuVIB.f

```
*****************************************************************
*** subroutine dsib2ffHituVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel VIB diagrams.
*** Hi emitted by internal legs: the sfermion - isf1 - radiates a Z by becoming a "second" sfermion - isf2 -
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-20
*****************************************************************
      subroutine dsib2ffH0tuVIB(f, isf1, isf2, hi)
```

dsib2ffHPamps.f

```
*********************************************************************
*** subroutine dsib2ffHPamps calculates helicity amplitudes for fFH final
*** states with charged scalars.
***
***   Input: f                    - final state fermion
***                                 (see header of dsib2sigmav)
```

```
***
*** Author: Francesca Calore, 2014-02-XX
**********************************************************************
      subroutine dsib2ffHPamps(f)
```

dsib2ffHPsFSRH3.f

```
**********************************************************************
*** subroutine dsib2ffHPsFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel FSR diagrams.
*** The Hp is emitted from the final legs of the s channel diagrams.
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
**********************************************************************
c...  H3 mediated diagrams
      subroutine dsib2ffHPsFSRH3(f, leg)
```

dsib2ffHPsFSRZ.f

```
**********************************************************************
*** subroutine dsib2ffHPsFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel FSR diagrams.
*** The Hp is emitted from the final legs of the s channel diagrams.
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
**********************************************************************
c...  Z mediated diagrams
      subroutine dsib2ffHPsFSRZ(f, leg)
```

dsib2ffHPsISRH.f

```
**********************************************************************
*** subroutine dsib2ffWsISRX add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel ISR diagrams.
*** W emitted from the initial legs of the s channel diagrams.
*** Exchanged particle index icha stays for chargino.
***
*** Subroutines: dsib2ffHPsISRH: Hc exchanged; ISR1 & ISR2
***              dsib2ffHPsISRW: W wxhanged; ISR1 & ISR2
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
**********************************************************************
c     Hc exchanged ISR1 & ISR2
      subroutine dsib2ffHPsISRH(f, icha)
```

dsib2ffHPsISRW.f

```
**********************************************************************
*** subroutine dsib2ffWsISRX add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel ISR diagrams.
*** W emitted from the initial legs of the s channel diagrams.
*** Exchanged particle index icha stays for chargino.
***
*** Subroutines: dsib2ffHPsISRH: Hc exchanged; ISR1 & ISR2
***              dsib2ffHPsISRW: W wxhanged; ISR1 & ISR2
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
**********************************************************************
c     W exchanged ISR1 & ISR2
      subroutine dsib2ffHPsISRW(f, icha)
```

dsib2ffHPsVIB.f

```
************************************************************************
*** subroutine dsib2ffHPsVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the s channel VIB diagrams.
*** The HP is emitted from the mediators (H3, Z).
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
************************************************************************
      subroutine dsib2ffHPsVIB(f)
```

dsib2ffHPtuFSR.f

```
************************************************************************
*** subroutine dsib2ffHPtuFSR add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel FSR diagrams.
*** The Hp is emitted by final legs 1 (FSR1) or 2 (FSR2) of the t+u channel diagrams.
*** isf1 is the index if the exchanged sfermion in th t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
************************************************************************
      subroutine dsib2ffHPtuFSR(f, leg, isf1)
```

dsib2ffHPtuISR1.f

```
************************************************************************
*** subroutine dsib2ffHPtuISRX adds to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** HP emitted by initial legs 1 (ISR1) or 2 (ISR2). isf is the index of the exchanged sfermion.
***
*** Subroutines: dsib2ffHPtuISR1:  HP emitted from p1 leg
***               dsib2ffHPtuISR2:  HP emitted from p2 leg
***
*** IMPORTANT: do NOT use separately dsib2ffHPtuISR1 and dsib2ffHPtuISR2.
***            The correct t+u channel for the 4 involved diagrams is given by ISR1 + ISR2.
***            Separately, the result is NOT the same as ISR1_t + ISR1_u (ISR2_t + ISR2_u),
***            being ISR1(t) \equiv ISR2(u) and ISR2(t) \equiv ISR1(u)
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
************************************************************************
      subroutine dsib2ffHPtuISR1(f, isf, icha)
```

dsib2ffHPtuISR2.f

```
************************************************************************
*** subroutine dsib2ffHPtuISRX adds to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** HP emitted by initial legs 1 (ISR1) or 2 (ISR2). isf is the index of the exchanged sfermion.
***
*** Subroutines: dsib2ffHPtuISR1:  HP emitted from p1 leg
***               dsib2ffHPtuISR2:  HP emitted from p2 leg
***
*** IMPORTANT: do NOT use separately dsib2ffHPtuISR1 and dsib2ffHPtuISR2.
***            The correct t+u channel for the 4 involved diagrams is given by ISR1 + ISR2.
***            Separately, the result is NOT the same as ISR1_t + ISR1_u (ISR2_t + ISR2_u),
```

```
***            being ISR1(t) \equiv ISR2(u) and ISR2(t) \equiv ISR1(u)
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
*********************************************************************
      subroutine dsib2ffHPtuISR2(f, isf, icha)
```

dsib2ffHPtuVIB.f

```
*********************************************************************
*** subroutine dsib2ffHPtuVIB add to the common block array amp(i,j), i = 0,
*** the helicity amplitudes of the t+u channel VIB diagrams.
*** Hp emitted by internal legs: the sfermion - isf or isff - radiates Hp by becoming a "second" sfermion - isff or is
*** f stays for the fermion, ff the antifermion
*** sf: sfermion of the fermion; sff: sfermion of the antifermion
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2014-02-22
***         Francesca Calore, 2015-02-15 (bug solved cf with CH)
***         Mathias Garny     2016-01-15 (bug MB*2 -> MB**2 l 86)
*********************************************************************
      subroutine dsib2ffHPtuVIB(f, isf, isff)
```

dsib2ffWamps.f

```
*********************************************************************
*** subroutine dsib2ffWamps calculates helicity amplitudes for fFW final
*** states.
***
***   Input: f                    - final state fermion
***                                 (see header of dsib2sigmav)
***
*** Author: Francesca Calore, 2013-04-10
***         Torsten Bringmann, 2014-02-20
*********************************************************************
      subroutine dsib2ffWamps(f)
```

dsib2ffWsFSR.f

```
*********************************************************************
*** subroutine dsib2ffWsFSR add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel FSR diagrams.
*** The W is emitted from the final legs of the s channel diagrams.
*** H3 and Z diagrams are considered separately and then
*** added to the corresponding entries of amp(-1:1, 0:3).
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-24-11 (modified ff)
*********************************************************************
      subroutine dsib2ffWsFSR(f, leg)
```

dsib2ffWsISR.f

```
*********************************************************************
*** subroutine dsib2ffWsISRX add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel ISR diagrams.
*** W emitted from the initial legs of the s channel diagrams.
*** Exchanged particle index icha stays for chargino.
```

```
***
*** Subroutines: dsib2ffWsISRH: Hc exchanged; ISR1 & ISR2
***              dsib2ffWsISRW: W wxhanged; ISR1 & ISR2
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-01-28
**********************************************************************
c     Hc exchanged ISR1 & ISR2
      subroutine dsib2ffWsISRH(f, icha)
```

dsib2ffWsVIB.f

```
**********************************************************************
*** subroutine dsib2ffWsVIB add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel VIB diagrams.
*** The W is emitted from the mediators (H3, Z), becoming Hc or W respectively.
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffWsVIB(f)
```

dsib2ffWtuFSR.f

```
**********************************************************************
*** subroutine dsib2ffWtuFSR add to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel FSR diagrams.
*** The W is emitted by final legs 1 (FSR1) or 2 (FSR2) of the t+u channel diagrams.
*** isf1 is the index if the exchanged sfermion in th t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffWtuFSR(f, leg, isf1)
```

dsib2ffWtuISR1.f

```
**********************************************************************
*** subroutine dsib2ffWtuISRX adds to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** W emitted by initial legs 1 (ISR1) or 2 (ISR2). isf is the index of the exchanged sfermion.
***
*** Subroutines: dsib2ffWtuISR1:  W emitted from p1 leg
***              dsib2ffWtuISR2:  W emitted from p2 leg
***
*** IMPORTANT: do *NOT* use separately dsib2ffWtuISR1 and dsib2ffWtuISR2.
***            The correct t+u channel for the 4 involved diagrams is given by ISR1 + ISR2.
***            Separately, the result is NOT the same as ISR1_t + ISR1_u (ISR2_t + ISR2_u),
***            being ISR1(t) \equiv ISR2(u) and ISR2(t) \equiv ISR1(u)
***
***            ISR1 and ISR2 are not invariant under CP symmetry serately, but
***            their sum is!
***
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-01-30
**********************************************************************
      subroutine dsib2ffWtuISR1(f, isf, icha)
```

dsib2ffWtuISR2.f

```
*****************************************************************
*** subroutine dsib2ffWtuISRX adds to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** W emitted by initial legs 1 (ISR1) or 2 (ISR2). isf is the index of the exchanged sfermion.
***
*** Subroutines: dsib2ffWtuISR1:  W emitted from p1 leg
***              dsib2ffWtuISR2:  W emitted from p2 leg
***
*** IMPORTANT: do *NOT* use separately dsib2ffWtuISR1 and dsib2ffWtuISR2.
***            The correct t+u channel for the 4 involved diagrams is given by ISR1 + ISR2.
***            Separately, the result is NOT the same as ISR1_t + ISR1_u (ISR2_t + ISR2_u),
***            being ISR1(t) \equiv ISR2(u) and ISR2(t) \equiv ISR1(u)
***
***            ISR1 and ISR2 are not invariant under CP symmetry serately, but
***            their sum is!
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
***         Francesca Calore, 2015-01-28
*****************************************************************
      subroutine dsib2ffWtuISR2(f, isf, icha)
```

dsib2ffWtuVIB.f

```
*****************************************************************
*** subroutine dsib2ffWtuVIB add to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel VIB diagrams.
*** W emitted by internal legs: the sfermion - isf or isff - radiates a Z by becoming a "second" sfermion - isff or is
*** f stays for the fermion, ff the antifermion
*** sf: sfermion of the fermion; sff: sfermion of the antifermion
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
*****************************************************************
      subroutine dsib2ffWtuVIB(f, isf, isff)
```

dsib2ffZamps.f

```
*************************************************************************
*** subroutine dsib2ffZamps calculates helicity amplitudes for fFW final
*** states.
***
***   Input: f                    - final state fermion
***                                 (see header of dsib2sigmav)
***
*** Author: Francesca Calore, 2013-04-10
***         Torsten Bringmann, 2014-02-20
*************************************************************************
      subroutine dsib2ffZamps(f)
```

dsib2ffZsFSR.f

```
*****************************************************************
*** subroutine dsib2ffZsFSR add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel FSR diagrams.
*** The Z is emitted from the final legs of the s channel diagrams.
*** H3 and Z diagrams are considered separately and then
*** added to the corresponding entries of amp(-1:1, 0:3).
```

```
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffZsFSR(f, leg)
```

dsib2ffZsISR.f

```
**********************************************************************
*** subroutine dsib2ffZsISR add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel ISR diagrams.
*** Z emitted from the initial legs of the  s channel diagrams.
*** H_{i}, i = 1, 2 and the sum is performed over the 2  Higges (ih index) already at the amplitude level.
*** The exchanged particle index ineu stays for neutralino.
***
*** Notice:  ISR1 = ISR2 ->  Each amplitudes is multiplied by 2
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffZsISR(f, ineu)
```

dsib2ffZsVIB.f

```
**********************************************************************
*** subroutine dsib2ffZsVIB add to the common block array amp(i,j)
*** the helicity amplitudes of the s channel VIB diagrams.
*** The Z is emitted from the mediators (H3, Z), becoming H_{i}, i= 1, 2. H_{i}.
*** The sum is performed only over ih.
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffZsVIB(f, ih)
```

dsib2ffZtuFSR.f

```
**********************************************************************
*** subroutine dsib2ffZtuFSR add to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel FSR diagrams.
*** The Z is emitted by final legs 1 (FSR1) or 2 (FSR2) of the t+u channel diagrams.
*** isf1 is the index if the exchanged sfermion in th t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffZtuFSR(f, leg, isf1)
```

dsib2ffZtuISR.f

```
**********************************************************************
*** subroutine dsib2ffZtuISR add to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel ISR diagrams.
*** Z emitted by initial legs 1 (ISR1) or 2 (ISR2) the neutralino
*** radiates a Z by becoming a "second" neutralino - ineu -
*** isf1 is the index if the exchanged sfermion in the t-channel
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
**********************************************************************
      subroutine dsib2ffZtuISR(f, leg, isf1, ineu)
```

dsib2ffZtuVIB.f

```
*****************************************************************
*** subroutine dsib2ffZtuVIB add to the common block array amp(i,j)
*** the helicity amplitudes of the t+u channel VIB diagrams.
*** Z emitted by internal legs: the sfermion - isf1 - radiates a Z by becoming a "second" sfermion - isf2 -
***
*** Notice: in the v->0 limit, to get the t+u amplitude a factor of 2* is required!
***         Helicity amplitudes are multiplied by 2.
***
*** Author: Francesca Calore, 2013-04-10
*****************************************************************
      subroutine dsib2ffZtuVIB(f, isf1, isf2)
```

dsib2getsfermion.f

```
*************************************************************************
*** subroutine dsib2getsfermion returns the sfermions corresponding to a     ***
*** given fermion f. If the global parameter IB2sfgen (set in dsib2set)      ***
*** equals 1, only the two sfermions states are returned that have the       ***
*** largest flavour component in common with f. Otherwise (for IB2sfgen=3),  ***
*** all sfermions with identical charges are returned (without assuming      ***
*** anything about their flavour ordering).                                  ***
***                                                                          ***
***      input: f - particle code of fermion                                 ***
***             i - index of sfermion to be returned                         ***
***                                                                          ***
***      output: particle code of sfermion number i                          ***
***                                                                          ***
*** Author: Torsten Bringmann, 2019-05-31                                    ***
*************************************************************************
      integer function dsib2getsfermion(f, i)
```

dsib2intres.f

```
*********************************************************************
*** function dsIB2intres returns the energy integral over the supplied
*** function, assuming that this function features the same relevant
*** resonances as the amplitude.
***
***   Input:  intfunction - function to be integrated over
***           ptype       - type of final state particle for which the
***                         energy integral is performed
***                         ('B','f' or 'fbar')
***           xmin,xmax   - energy range considered [in units of m_chi]
***           acc         - required accuracy
***   Output: ier         - error returned by dqagse
***                         (corresponding contributions to value of
***                          integral are always set to 0)
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-03-06
*********************************************************************

      real*8 function dsib2intres(intfunction,ptype,xmin,xmax,acc,ier)
```

dsib2kinematics.f

```
*********************************************************************
*** subroutines dsib2kinematics writes relevant dimensionless kinematical
*** quantities to the common block (needed by helicity amplitude routines).
***
***    Input: QB                  - Boson is neutral (0) or charged (1)
***           E1 \equiv x1 = E1/mx - CMS fermion energy
```

```
***           EB \equiv xB = Eb/mx - CMS boson energy
***
***    Output: istat - error code (0 if everything OK)
***
*** Author: Francesca Calore, 2013-04-10
***         2014-02-20 (Torsten Bringmann, adapted to include scalars)
**********************************************************************
      subroutine dsib2kinematics(QB, EB, E1, istat)
```

dsib2Msqaux.f

```
**********************************************************************
*** Function dsib2Msqaux returns the 3-body amplitude squared, as a
*** function of the energy of one of the final state particles.
*** Channel and final state particle set in common blocks
*** (auxiliary function needed for integration routines, do not call
*** directly!)
***
*** Author: Torsten Bringmann, 2014-02-20
**********************************************************************
      real*8 function dsib2Msqaux(Efinal)
```

dsib2set.f

```
**********************************************************************
*** Routine dsib2set sets default settings for electroweak internal
*** bremsstrahlung (IB2) calculations. Must be called before any of the
*** IB2 routines can be used (typically done in dsinit)!
***
***  c - character string specifying choice to be made
***  possible values for c:
***
***  'default' - include all ffB channels (where B is any Boson)
***  'off'     - no electroweak IB contribution
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-03-06
**********************************************************************

      subroutine dsib2set(c)
```

dsib2sigmav.f

```
**********************************************************************
*** function dsIB2sigmav returns the 3-body annihilation rate (sigma v),
*** in the v->0 limit, for the following channels:
*** (calling it with IB2chinput=0, the sum over all these channels is
*** returned)
***
***   Input: IB2chinput  - 0   sum over all states
***                        1XX for ffZ
***                        2XX for fFW
***                        3XX for ffh (XX like for ffZ)
***                        4XX for ffH (XX like for ffZ)
***                        5XX for ffA (XX like for ffZ)
***                        6XX for fFH+- (XX like for fFW)
***
***                         XX   | f \bar f Z      |   f \bar f W
***                        ------+-----------------+------------------
***                         01   | nu_e   nu_e   Z |   nu_e   e      W-
***                         02   | e      e      Z |   e      nu_e   W+
***                         03   | nu_mu  nu_mu  Z |   nu_mu  mu     W-
***                         04   | mu     mu     Z |   mu     nu_mu  W+
```

```
***                          05   | nu_tau nu_tau Z |   nu_tau tau    W-
***                          06   | tau    tau    Z |   tau    nu_tau W+
***                          07   | u      u      Z |   u      d      W-
***                          08   | d      d      Z |   d      u      W+
***                          09   | c      c      Z |   c      s      W-
***                          10   | s      s      Z |   s      c      W+
***                          11   | t      t      Z |   t      b      W-
***                          12   | b      b      Z |   b      t      W+
***                          (note that this numbering must be consistent
***                           with dsmssm.h !!!)
***
***          onshell - 0 subtracts the on-shell contributions ("2-body final
***                      states")in the narrow width approximation [default]
***                      (note that the result can be negative!)
***                    1 returns the result including on-shell contributions
***
***   Output: annihilation rate in units cm**3 s**-1
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-12-07, update 2105-03-22 (streamlined handling of kf)
***         update 2016-02-05 updated to DS6 conventions
**********************************************************************

      real*8 function dsib2sigmav(IB2chinput,onshell)
```

dsib2yield.f

```
**********************************************************************
*** function dsIB2yield returns the yield from electroweak internal
*** bremsstrahlung (IB) for SUSY models.
***
***   Input: egev    - energy in GeV
***          yieldk  - which yield to calculate
***                    (see dshaloyield for an explanation)
***          onshell - 0 subtracts the on-shell contributions ("2-body final
***                      states")in the narrow width approximation [default]
***                      (note that the result can be negative!)
***                    1 returns the result including on-shell contributions
***   Output: yield [total tree-level(!) annihilation]**-1,
***                  differential also [GeV]**-1
***           istat - 0 if everything went well
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-03-03
**********************************************************************

      real*8 function dsIB2yield(egev,yieldk,onshell,istat)
```

dsib2yieldone.f

```
**********************************************************************
*** function dsIB2yieldone calculates the electroweak IB yield from ONE
*** individual 3-body annihilation channel. NB: For the default case
*** (on-shell constribution already subtracted), only the *sum* over all
*** these channels gives a well-defined yield (as returned by dsib2yield)!!!
***
***   Input: egev  - energy in GeV
***          yieldk  - which yield to calculate
***                    (see dsanyield_ch for an explanation)
***          IB2ch   - 1XX for ffZ
***                    2XX for WfF
***                    3XX for ffh (XX like for ffZ)
***                    4XX for ffH (XX like for ffZ)
***                    5XX for ffA (XX like for ffZ)
```

```
***                    6XX for ffH+- (XX like for fFW)
***
***                    XX   | \bar f f Z      |   \bar f f W
***                   ------+-----------------+------------------
***                    01   | nu_e   nu_e   Z |   e+     nu_e   W-
***                    02   | e      e      Z |   nu_e   e-     W+
***                    03   | nu_mu  nu_mu  Z |   mu+    nu_mu  W-
***                    04   | mu     mu     Z |   nu_mu  mu-    W+
***                    05   | nu_tau nu_tau Z |   tau+   nu_tau W-
***                    06   | tau    tau    Z |   nu_tau tau-   W+
***                    07   | u      u      Z |   dbar   u      W-
***                    08   | d      d      Z |   ubar   d      W+
***                    09   | c      c      Z |   sbar   c      W-
***                    10   | s      s      Z |   cbar   s      W+
***                    11   | t      t      Z |   bbar   t      W-
***                    12   | b      b      Z |   tbar   b      W+
***                    (note that this numbering must be consistent
***                     with dmssm.h !!! )
***
***
***          onshell - 0 subtracts the on-shell contributions ("2-body final
***                       states")in the narrow width approximation [default]
***                       (note that the result can be negative!)
***                     1 returns the result including on-shell contributions
***
***   Output: yield [total tree-level(!) annihilation]**-1,
***                  differential also [GeV]**-1
***           -> use dssigmav() & dsib2sigmav() to manually adjust
***              normalization to refer to individual channels instead.
***
*** Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)
*** Date:   2014-12-16, update 2015-03-22 (streamlined handling of c_ftype),
***         update 2015-11-19 (added neutrino and positron spectra)
***         update 2016-02-05 updated to DS6 conventions
**********************************************************************

      real*8 function dsIB2yieldone(egev,IB2ch,yieldk,onshell,istat)
```

39.7 mssm/an_ib3: Internal bremsstrahlung of gluons

39.7.1 Routine headers – fortran files

dsIB3importdata.f

```
**********************************************************************
*** subroutine dsIB3importdata reads data files that contain tabulated
*** yields from qqg final states and stores information in common
*** blocks.
***
***   input:   qch   -- quark channel (7-up to 12-bottom)
***            spectype -- see beginning of dsIB3yieldtab for definitions
***
*** author: torsten.bringmann@fys.uio.no, 2016-04-02
**********************************************************************

      subroutine dsIB3importdata(qch,spectype)
```

dsIB3set.f

```
**********************************************************************
*** Routine dsib3set sets default settings for gluon internal
```

```
*** bremsstrahlung (IB) contributions to (total) yield calculations.
***
***  c - character string specifying choice to be made
***  (currently for photon and positron yield only)
***  possible values for c:
***  'photonIB' or 'default' - same settings as for photon IB
***                            (see header of dsIBset)
***  'off'                   - no gluon IB contribution
***
*** author: torsten bringmann, 2021-08-04
**********************************************************************

      subroutine dsib3set(c)
```

dsib3svqqgratio.f

```
**********************************************************************
*** Function dsib3svqqgratio returns the ratio between (FSR-subtracted)
*** qqg and tree-level cross section of a neutralino pair annihilating
*** via an s-wave into a quark pair. See Bringmann, Galea & Walia (2015).
***
***  input:   kpn -- particle code for neutralino
***           kpq -- particle code for quark
***
*** author: torsten bringmann
*** date: 2015-10-30
**********************************************************************
      real*8 function dsib3svqqgratio(kpn,kpq)
```

dsib3svqqratio.f

```
**********************************************************************
*** Function dsib3svqqratio returns the ratio between QCD-corrected and
*** tree-level cross section of a neutralino pair annihilating via an
*** s-wave into a quark pair. See Eqs. (A17, A21) of
*** Bringmann, Galea & Walia (2015).
***
***  input:   Ecm   -- total CMS energy
***           kpn -- particle code for neutralino
***           kpq -- particle code for quark
***
*** author: torsten bringmann & parampreet walia
*** date: 2015-10-30
**********************************************************************
      real*8 function dsib3svqqratio(Ecm,kpn,kpq)
```

dsIB3yield.f

```
**************************************************************************
***   function dsIB3yield gives the yield resulting from from gluon internal
***   bremsstrahlung (IB) for SUSY models, i.e. the *difference* to the 2-body
***   yield when taking into account qqg final states.
***   Input: egev - energy in GeV
***          yieldk - which yield to calculate
***             (see dshaloyield for an explanation,
***              currently only photon and antiproton yields are implemented)
***   Output: yield (annihilation)**1, differential also GeV**-1
***           istat is set as follows in case of errors
***   bit  decimal  reason
***     0        1  dsIBf_intdy (kinematic integration for 3-body rate) failed
***
```

```
*** Author: torsten.bringmann@fys.uio.no
*** Date: 2015-06-01
**********************************************************************

      real*8 function dsIB3yield(egev,yieldk,istat)
```

dsIB3yieldfit.f

```
**********************************************************************
*** subroutine dsIB3yieldfit returns the fit parameters necessary to approximate
*** the full 3-body spectrum as a linear combination of the most extreme
*** spectra that are possible, following the procedure described in 1510.02473.
***
***   input:   qch    -- quark channel (7-up to 12-bottom)
***            yieldk -- as in dsanyield_ch. For the moment, only gamma
***                      rays and antiprotons are implemented.
***   output:  mix      -- 1 (0) for large (small) squark mixing
***            y        -- interpolation parameter, c.f. Eq.(3,4) in 1510.02473.
***
*** Author: torsten.bringmann@fys.uio.no
*** Date: 2016-04-02
**********************************************************************

      subroutine dsIB3yieldfit(qch,yieldk,mix,y)
```

dsIB3yieldone.f

```
**********************************************************************
*** function dsIB3yieldone calculates the IB3 yield from one given
*** annihilation channel, i.e. the *difference* to the corresponding qq
*** yield when taking into account qqg final states.
***
*** Currently included are:
***   yieldk =  54 - antiproton yield above threshold emuthr
***             154 - differential antiproton yield at emuthr
***              52 - photon yield above threshold emuthr
***             152 - differential photon yield at emuthr
***
*** The annihilation channels are:
***  qch  =  7 - u u-bar
***          8 - d d-bar
***          9 - c c-bar
***         10 - s s-bar
***         11 - t t-bar
***         12 - b b-bar
***
*** See arXiv: 1510.02473 for more details on the scheme implemented here.
***
*** the units are (annihilation into IBch)**-1
*** for the differential yields, the units are the same times gev**-1.
***
*** istat will set upon return in case of errors
***   bit  decimal  reason
***     0        1  dsIBf_intdxdy failed
***     1        2  dsIBf_intdy failed
*** Author: torsten.bringmann@fys.uio.no
*** Date: 2015-06-01
**********************************************************************

      real*8 function dsIB3yieldone(emuthr,qch,yieldk,istat)
```

dsIB3yieldtab.f

```
************************************************************************
*** returns tabulated yield resulting from the subtracted 3-body qqg
*** spectrum
***
***   input:   TGeV   -- Kinetic energy in GeV
***            mdm    -- dark matter mass in GeV
***            qch    -- quark channel (7-up to 12-bottom)
***            threebodytype -- 1: VIB, maximal mixing
***                             2: VIB, minimal mixing
***                             3: heavy squark limit, maximal mixing
***                             4: heavy squark limit, minimal mixing
***                                (for definitions, see arxiv: 1510.02473)
***            yieldk -- as in dsanyield_ch. For the moment, only gamma
***                      rays and antiprotons are implemented.
***
*** the units are (annihilation into IBch)**-1
*** for the differential yields, the units are the same times gev**-1.
***
*** author: torsten.bringmann@fys.uio.no, 2015-08-07
************************************************************************

      real*8 function dsIB3yieldtab(TGev,mdm,qch,threebodytype,yieldk)
```

39.8 mssm/an_sf: Annihilation cross sections (with sfermions)

39.8.1 Annihilation cross sections with sfermions – general

In this directory, all the (co)annihilation cross sections involving one or more sfermions in the initial state are calculated. The code here is based upon the work described in [14]. All the cross sections are calculated with Form and converted to Fortran with a script **form2f** [187].

The main routines here are

Routine	Purpose
dsasdwdcossfsf	Calculates the invariant annihilation rate between two sfermions in the initial state.
dsasdwdcossfchi	Calculates the invariant annihilation rate between one sfermion and one fermion (neutralino or chargino) in the iniital state.

39.8.2 Routine headers – fortran files

dsaschicasea.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.34, October 8, 2001, edsjo@fysik.su.se)
c....Template file for dsaschicasea begins here

************************************************************
*** SUBROUTINE dsaschicasea                              ***
*** computes dW_{ij}/dcostheta                           ***
***                                                      ***
*** sfermion(i) + neutralino(j)/chargino~+(j)            ***
*** -> gauge-boson + fermion                             ***
***                                                      ***
*** The sfermion must be the first mentioned             ***
*** particle (kp1) and the neutralino/chargino           ***
*** the other (kp2) -- not the opposite.                 ***
```

```
*** For the final state the gauge boson must be mentioned  ***
*** first (i.e. kp3) and next the fermion (kp4) --         ***
*** not the opposite.                                      ***
***                                                        ***
***                                                        ***
*** Author:Mia Schelke, schelke@physto.se                  ***
*** Date: 01-10-05                                         ***
*** QCD included: 02-03-21                                 ***
*** comment added by Piero Ullio, 02-07-01                 ***
*** added flavour changing charged exchange f the 2nd case ***
*** 3rd, 4th and the 5th case                              ***
*** added by Mia Schelke June 2006                         ***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, 08-05-30                                       ***
*** sfermion particle codes corrected by JE 090403         ***
************************************************************

***** Note that it is assumed that coupling constants that do
***** not exist have already been set to zero!!!!!
***** Thus, many of the coefficients defined in this code
***** simplify when the diagrams contain sneutrinos
***** or neutrinos.

      subroutine dsaschicasea(kp1,kp2,kp3,kp4,icase,par)
```

dsaschicaseb.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.34, October 8, 2001, edsjo@fysik.su.se)
c....Template file for dsaschicaseb begins here

************************************************************
*** SUBROUTINE dsaschicaseb                              ***
*** computes dW_{ij}/dcostheta                           ***
***                                                      ***
*** anti-sfermion(i) + neutralino(j)/chargino^+(j)       ***
*** -> gauge-boson + anti-fermion                        ***
***                                                      ***
*** The anti-sfermion must be the first mentioned        ***
*** particle (kp1) and the neutralino/chargino           ***
*** the other (kp2) -- not the opposite.                 ***
*** For the final state the gauge boson must be mentioned  ***
*** first (i.e. kp3) and next the anti-fermion (kp4) --  ***
*** not the opposite.                                    ***
***                                                      ***
***                                                      ***
*** Author:Mia Schelke, schelke@physto.se                ***
*** Date: 01-10-03                                       ***
*** QCD included: 02-03-21                               ***
*** comment added by Piero Ullio, 02-07-01               ***
*** added flavour changing charged exchange:             ***
*** for the 1st,2nd, 3rd and 4th case                    ***
*** added by Mia Schelke January 2007                    ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

***** Note that it is assumed that coupling constants that do
***** not exist have already been set to zero!!!!!
***** Thus, many of the coefficients defined in this code
***** simplify when the diagrams contain sneutrinos
***** or neutrinos.

      subroutine dsaschicaseb(kp1,kp2,kp3,kp4,icase,par)
```

dsaschicasec.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.34, October 8, 2001, edsjo@fysik.su.se)
c....Template file for dsaschicasec begins here

************************************************************
*** SUBROUTINE dsaschicasec                              ***
*** computes dW_{ij}/dcostheta                           ***
***                                                      ***
*** sfermion(i) + neutralino(j)/chargino^+(j)            ***
*** -> higgs-boson + fermion                             ***
***                                                      ***
*** The sfermion must be the first mentioned             ***
*** particle (kp1) and the neutralino/chargino           ***
*** the other (kp2) -- not the opposite.                 ***
*** For the final state the higgs boson must be mentioned  ***
*** first (i.e. kp3) and next the fermion (kp4) --       ***
*** not the opposite.                                    ***
***                                                      ***
***                                                      ***
*** Author:Mia Schelke, schelke@physto.se                ***
*** Date: 01-10-10                                       ***
*** Trivial color factors included: 02-03-21             ***
*** Added flavour changing charged exchange:             ***
*** for the 3rd, 4th, 5th and 6th case                   ***
*** added by Mia Schelke January 2007                    ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

***** Note that it is assumed that coupling constants that do
***** not exist have already been set to zero!!!!!
***** Thus, many of the coefficients defined in this code
***** simplify when the diagrams contain sneutrinos
***** or neutrinos.

      subroutine dsaschicasec(kp1,kp2,kp3,kp4,icase,par)
```

dsaschicased.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.34, October 8, 2001, edsjo@fysik.su.se)
c....Template file for dsaschicased begins here

************************************************************
*** SUBROUTINE dsaschicased                              ***
*** computes dW_{ij}/dcostheta                           ***
***                                                      ***
*** anti-sfermion(i) + neutralino(j)/chargino^+(j)       ***
*** -> higgs-boson + anti-fermion                        ***
***                                                      ***
*** The anti-sfermion must be the first mentioned        ***
*** particle (kp1) and the neutralino/chargino           ***
*** the other (kp2) -- not the opposite.                 ***
*** For the final state the higgs boson must be mentioned  ***
*** first (i.e. kp3) and next the anti-fermion (kp4) --  ***
*** not the opposite.                                    ***
***                                                      ***
***                                                      ***
*** Author:Mia Schelke, schelke@physto.se                ***
*** Date: 01-10-11                                       ***
*** Trivial color factors included: 02-03-21             ***
*** Added flavour changing charged exchange:             ***
*** for the 1st, 2nd, 3rd, 4th and 5th case              ***
```

```
*** added by Mia Schelke January 2007                      ***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, 08-05-30                                       ***
*************************************************************

***** Note that it is assumed that coupling constants that do
***** not exist have already been set to zero!!!!!
***** Thus, many of the coefficients defined in this code
***** simplify when the diagrams contain sneutrinos
***** or neutrinos.

      subroutine dsaschicased(kp1,kp2,kp3,kp4,icase,par)
```

dsaschizero.f

```
*************************************************************
*** SUBROUTINE dsaschizero                                ***
*** computes dW_{ij}/dcostheta                            ***
*** sfermion(i) + neutralino(j) -> gamma/gluon + fermion   ***
*** ampl2 obtained with sum over physical polarizations    ***
***                                                       ***
*** input askin variables: p12,costheta                   ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                   ***
*** Date: 02-06-13                                        ***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, 08-05-30                                       ***
*************************************************************

      SUBROUTINE dsaschizero(kp1,kp2,kp3,kp4,icase,result)
```

dsascolset.f

```
      subroutine dsascolset(iloctype)
No header found.
```

dsasdepro.f

```
*************************************************************
*** FUNCTION dsasdepro                                    ***
*** computes the denominator of a propagator              ***
***                                                       ***
*** input: mom2 is S,T,U;                                 ***
*** kkpp is the number of the particle in the propagator  ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                   ***
*** Date: 01-02-28                                        ***
*************************************************************

      complex*16 function dsasdepro(mom2,kkpp)
```

dsasdwdcossfchi.f

```
*****************************************************************************
*** SUBROUTINE dsasdwdcossfchi                                            ***
*** computes dW_{ij}/dcostheta                                            ***
*** for sfermion(1) + neutralino(2) (or chargino(2))                      ***
*** plus sfermion(1) + neutralino(2) (or chargino(2))                     ***
***                                                                       ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                                   ***
*** Date: 01-11-04                                                        ***
*** Modified: Joakim Edsjo, Mia Schelke                                   ***
***   to include gluon final states, 2002-03-21                          ***
*** Modified: Piero Ullio                                                 ***
***   to switch to ampl2 with physical polarizations, 02-07-01           ***
```

```
*** Modified: Mia Schelke                                              ***
***   to allow for initial state squarks of different family than      ***
***   final state quarks in selected cases, June 2006, Jan 2007        ***
*** modified: Piero Ullio to include a new labelling of states,        ***
***   08-05-30                                                         ***
*** modified: Joakim Edsjo to properly include inter-family mixing     ***
***   2016-10-29                                                       ***
************************************************************************

      real*8 function dsasdwdcossfchi(p,costhe,kp1,kp2)
```

dsasdwdcossfsf.f

```
************************************************************************
*** SUBROUTINE dsasdwdcossfsf                                        ***
*** computes dW_{ij}/dcostheta                                       ***
*** for sfermion(1) + antisfermion(2) plus sfermion(1) + sfermion(2) ***
***                                                                  ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                              ***
*** Date: 01-08-10                                                   ***
*** modified: Joakim Edsjo, Mia Schelke to include squarks with      ***
***   gauge and Higgs boson final states and gluons                  ***
***   02-05-22                                                       ***
***   corrected switching of initial states fixed 020613 (edsjo)     ***
*** modified: Piero Ullio                                            ***
***   02-03-22                                                       ***
*** modified: Piero Ullio                                            ***
***   02-07-01                                                       ***
*** modified: Mia Schelke to include squark mixing:                  ***
***   diff family i.s. sq's with bosons in final state               ***
***   Jan 2007                                                       ***
*** modified: Piero Ullio to include mixing in fermion final states  ***
***   and a new labelling of states                                  ***
***   08-05-30                                                       ***
***   Joakim Edsjo, correction in slepton-squark part (case 3)       ***
***   2015-04-08                                                     ***
***   Joakim Edsjo, made more general (full 6x6 mixing),             ***
***   Cleaned up a bit (not as many if's needed with full 6x6        ***
***   mixing). 2016-11-16                                            ***
************************************************************************

      real*8 function dsasdwdcossfsf(p,costhe,kp1,kp2)
```

dsasfer.f

```
************************************************************
*** SUBROUTINE dsasfer                                   ***
*** computes dW_{ij}/dcostheta                           ***
*** sfermion(i) + antisfermion(j)                        ***
*** -> fermion(k1) + antifermion(k2)                     ***
*** version to be used if i or j has non-zero lepton #   ***
***                                                      ***
*** input askin variables: p12,costheta                  ***
*** kpk1 and kpk2 are the fermion code                   ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-08-10                                       ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

      SUBROUTINE dsasfer(kpi,kpj,kpk1,kpk2,icase,result)
```

dsasfercol.f

```
************************************************************
*** SUBROUTINE dsasfercol                                ***
*** computes dW_{ij}/dcostheta                           ***
*** sfermion(i) + antisfermion(j)                        ***
*** -> fermion(k1) + antifermion(k2)                     ***
*** version to be used if i or j are both squarks        ***
***                                                      ***
*** input askin variables: p12,costheta                  ***
*** kpk1 and kpk2 are the fermion code                   ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-08-10                                       ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

      SUBROUTINE dsasfercol(kpi,kpj,kpk1,kpk2,icase,result)
```

dsasfere.f

```
************************************************************
*** SUBROUTINE dsasfere                                  ***
*** computes dW_{ij}/dcostheta                           ***
*** sfermion(i) + sfermion(j)                            ***
*** -> fermion(k1) + fermion(k2)                         ***
*** version to be used if i or j has non-zero lepton #   ***
***                                                      ***
*** input askin variables: p12,costheta                  ***
*** kpk1 and kpk2 are the fermion code                   ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-08-10                                       ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

      SUBROUTINE dsasfere(kpi,kpj,kpk1,kpk2,icase,result)
```

dsasferecol.f

```
************************************************************
*** SUBROUTINE dsasferecol                               ***
*** computes dW_{ij}/dcostheta                           ***
*** sfermion(i) + sfermion(j)                            ***
*** -> fermion(k1) + fermion(k2)                         ***
*** version to be used if i or j are both squarks        ***
***                                                      ***
*** input askin variables: p12,costheta                  ***
*** kpk1 and kpk2 are the fermion code                   ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-08-10                                       ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
************************************************************

      SUBROUTINE dsasferecol(kpi,kpj,kpk1,kpk2,icase,result)
```

dsasff.f

```
************************************************************
*** SUBROUTINE dsasff                                    ***
*** computes the amplitude squared of the process        ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)     ***
***                                                      ***
```

```
*** input:                                                      ***
*** asparmass, askin, askinder variables                       ***
*** complex vectors ASxpl(i), ASxpr(i), ASyl, ASyr             ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-02-28                                             ***
*****************************************************************

      SUBROUTINE dsasff(ampl2)
```

dsasffcol.f

```
*****************************************************************
*** SUBROUTINE dsasffcol                                       ***
*** computes the amplitude squared of the process              ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)           ***
***                                                            ***
*** input:                                                      ***
*** asparmass, askin, askinder variables                       ***
*** complex vectors:                                           ***
***   ASxplc(j,i), ASxprc(j,i), ASylc(j), ASyrc(j)             ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-02-28                                             ***
*****************************************************************


      SUBROUTINE dsasffcol(ampl2)
```

dsasgbgb.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.35, May 23, 2002, edsjo@fysik.su.se)
c....Template file for dsasgbgb begins here

*****************************************************************
*** SUBROUTINE dsasgbgb                                        ***
*** computes dW_{ij}/dcostheta                                 ***
***                                                            ***
*** sfermion(i) + anti-sfermion(j)                             ***
*** -> MASSIVE gauge-boson + MASSIVE gauge-boson               ***
***                                                            ***
*** for one massive and one massless gb use dsasgbgb1exp       ***
*** for two massless gb use code dsasgbgb2exp                  ***
***                                                            ***
*** The first mentioned particle (kp1) will be taken as        ***
*** a sfermion and the second particle (kp2) as an             ***
*** anti-sfermion -- not the opposite.                         ***
***                                                            ***
*** When kp1 and kp2 have different                            ***
*** weak isospin (T^3=+,-1/2), then kp1 must be an             ***
*** up-type-sfermion and kp2 a down-type-anti-sfermion.        ***
***                                                            ***
*** When one gauge boson have electric charge while the        ***
*** other is neutral, then the charged one must be             ***
*** mentioned first (kp3) and then the neutral one (kp4)       ***
*** -- not the opposite.                                       ***
***                                                            ***
*** Author:Mia Schelke, schelke@physto.se                      ***
*** Date: 01-10-23  rewritten:02-03-12                         ***
*** QCD included: 02-03-20                                     ***
*** Ghost term excluded: 02-05-22                              ***
*** rewritten: 02-07-04 (now only massive gb)                  ***
*** added flavour changing charged exchange for W^-W^+:        ***
*** added by Mia Schelke 2005-06-14 + Jan 2007                 ***
*** added flavour changing charged exchange for W^+Z:          ***
```

```
*** added by Mia Schelke Jan 2007                          ***
*** terms rearranged by Paolo Gondolo, 2005-06             ***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, plus new factorization of first terms          ***
*** 08-05-30                                              ***
*************************************************************

      subroutine dsasgbgb(kp1,kp2,kp3,kp4,icase,par)
```

dsasgbgb1exp.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.35, May 23, 2002, edsjo@fysik.su.se)
c....Template file for dsasgbgb1exp begins here

*************************************************************
*** SUBROUTINE dsasgbgb1exp                               ***
*** computes dW_{ij}/dcostheta                            ***
***                                                       ***
*** sfermion(i) + anti-sfermion(j)                        ***
*** -> massive gauge-boson + massless gauge-boson         ***
***                                                       ***
***                                                       ***
*** The first mentioned particle (kp1) will be taken as   ***
*** a sfermion and the second particle (kp2) as an        ***
*** anti-sfermion -- not the opposite.                    ***
***                                                       ***
*** When kp1 and kp2 have different                       ***
*** weak isospin (T^3=+,-1/2), then kp1 must be an        ***
*** up-type-sfermion and kp2 a down-type-anti-sfermion.   ***
***                                                       ***
*** NOTE: for the gauge bosons, the MASSIVE must be       ***
*** mentioned first (kp3) and then the MASSLESS one (kp4) ***
*** -- not the opposite.                                  ***
***                                                       ***
*** Author:Mia Schelke, schelke@physto.se                 ***
*** Date: 01-10-23  rewritten:02-03-12                    ***
*** QCD included: 02-03-20                                ***
*** rewritten: 02-07-04 (to have exactly one massless gb) ***
*** explicite pol. vectors introduced: 02-07-05           ***
*** sum over massless pol. moved from                     ***
*** fortran to form: 02-07-09                             ***
***                                                       ***
*** addition: Jan 2007 by Mia Schelke                     ***
*** squark mixing allowed for W+ gamma,gluon              ***
*** modified: Piero Ullio to include a new labelling of   ***
*** states, 08-05-30                                      ***
*************************************************************

      subroutine dsasgbgb1exp(kp1,kp2,kp3,kp4,icase,par)
```

dsasgbgb2exp.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.35, May 23, 2002, edsjo@fysik.su.se)
c....Template file for dsasgbgb2exp begins here

*************************************************************
*** SUBROUTINE dsasgbgb2exp                               ***
*** computes dW_{ij}/dcostheta                            ***
***                                                       ***
*** sfermion(i) + anti-sfermion(j)                        ***
*** -> gluon+gluon, photon+photon, photon+gluon           ***
***                                                       ***
```

```
*** The first mentioned particle (kp1) will be taken as    ***
*** a sfermion and the second particle (kp2) as an         ***
*** anti-sfermion -- not the opposite.                     ***
***                                                        ***
***                                                        ***
***                                                        ***
*** Author:Mia Schelke, schelke@physto.se                  ***
*** Date: 02-05-21                                         ***
*** Rewritten: 02-07-03 (to have exactly two massless gb)  ***
*** explicite pol. vectors introduced: 02-07-08            ***
*** sum over pol. moved from fortran to form: 02-07-09     ***
*** two colour factors(c.f.) made complex: 02-07-10        ***
***(+these c.f. changed for g+g as ggg vertex code changed)***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, 08-05-30                                       ***
************************************************************

      subroutine dsasgbgb2exp(kp1,kp2,kp3,kp4,icase,par)
```

dsasgbhb.f

```
************************************************************
*** SUBROUTINE dsasgbhb                                  ***
*** computes dW_{ij}/dcostheta                           ***
*** up-sfermion(i) + down-antisfermion(j) ->             ***
***    gauge boson + Higgs boson                         ***
*** ampl2 obtained summing over physical polarizations   ***
***                                                      ***
*** input askin variables: p12,costheta                  ***
***    itype(1),itype(2),mass1,mass2                     ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 02-06-13                                       ***
*** This routine has been compared with the routine of   ***
*** Mia Schelke and the agreement is perfect, except in  ***
*** the low-p limit (due to widths in propagators)       ***
*** as expected.                                         ***
*** added flavour changing charged exchange for W^-H^+:  ***
*** added by Mia Schelke 2006-06-07  + Jan 2007          ***
*** added flavour changing charged exchange for:         ***
*** W^+ H^0_1,2,3 :  H^+ gamma,Z,gluon                   ***
*** added by Mia Schelke Jan 2007                        ***
*** modified: Piero Ullio to include a new labelling of  ***
*** states, 08-05-30                                     ***
*** Modified: Joakim Edsjo to allow for family mixing    ***
*** (assumptions were made on family belonging that were ***
*** not general enough). 2016-11-13                      ***
************************************************************

      SUBROUTINE dsasgbhb(kp1,kp2,kp3,kp4,icase,result)
```

dsashbhb.f

```
c...This subroutine is automatically generated from form output by
c...parsing it through form2f (version 1.35, May 23, 2002, edsjo@fysik.su.se)
c....Template file for dsashbhb begins here

************************************************************
*** SUBROUTINE dsashbhb                                  ***
*** computes dW_{ij}/dcostheta                           ***
***                                                      ***
*** sfermion(i) + anti-sfermion(j)                       ***
*** -> higgs-boson + higgs-boson                         ***
***                                                      ***
*** The first mentioned particle (kp1) will be taken as  ***
```

```
*** a sfermion and the second particle (kp2) as an         ***
*** anti-sfermion -- not the opposite.                     ***
***                                                        ***
*** When kp1 and kp2 have different                        ***
*** weak isospin (T^3=+,-1/2), then kp1 must be an         ***
*** up-type-sfermion and kp2 a down-type-anti-sfermion.    ***
***                                                        ***
*** For the cases with one charged and one neutral higgs   ***
*** in the final state, the charged higgs must be          ***
*** mentioned first (i.e. kp3) and next the neutral        ***
*** higgs-boson (kp4) -- not the opposite.                 ***
***                                                        ***
***                                                        ***
*** Author:Mia Schelke, schelke@physto.se                  ***
*** Date: 01-10-19                                         ***
*** Rewritten: 02-07-03 (because FORM input file now       ***
***  only gives the amplitude)                             ***
*** added flavour changing charged exchange for H^+H^-:    ***
*** added by Mia Schelke 2006-06-08  + Jan 2007            ***
*** added flavour changing charged exchange for:H^+H_1,2,3 ***
*** added by Mia Schelke Jan 2007                          ***
*** modified: Piero Ullio to include a new labelling of    ***
*** states, 08-05-30                                       ***
************************************************************
***                                                        ***
*** NOTE: The FORM input file only gives the amplitude     ***
***       not the amplitude squared                        ***
*** THE FOLLOWING THEREFORE HAS TO BE CHANGED BY HAND      ***
*** after running of the PERL script                       ***
***                                                        ***
*** the form result should be denoted amplitude instead    ***
*** of dsashbhbas                                          ***
*** the amplitude squared calculation should be added      ***
*** -- the lines are written in the end of                 ***
*** the template file                                      ***
************************************************************

      subroutine dsashbhb(kp1,kp2,kp3,kp4,icase,par)
```

dsaskinset.f

```
************************************************************
*** subroutine dsaskinset                                ***
*** set askinder variables: the kinematic variables and  ***
*** the scalar products of the four vectors              ***
*** p(1), p(2), k(3) and k(4) related by                 ***
*** p(1) + p(2) = k(3) + k(4)                            ***
*** in the center of mass frame                          ***
*** input: asparmass, askin variables                    ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-02-28                                       ***
************************************************************

      subroutine dsaskinset
```

dsaskinset1.f

```
************************************************************
*** subroutine dsaskinset1                               ***
*** sets: ep1, ep2, and Svar                             ***
*** input: mass1, mass2, p12                             ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
```

```
*** Date: 01-02-28                                              ***
****************************************************************

      subroutine dsaskinset1
```

dsaskinset2.f

```
****************************************************************
*** subroutine dsaskinset2                                     ***
*** sets: k34, ek3, ek4, Tvar, Uvar                            ***
*** you must call dsaskinset1 before calling dsaskinset2       ***
*** input: mass3, mass4, costheta                              ***
***                                                            ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-02-28                                             ***
****************************************************************

      subroutine dsaskinset2
```

dsaskinset3.f

```
****************************************************************
*** subroutine dsaskinset3                                     ***
*** sets the scalar products of the four vectors               ***
*** p(1), p(2), k(3) and k(4) related by                       ***
*** p(1) + p(2) = k(3) + k(4)                                  ***
*** in the center of mass frame                                ***
*** you must call dsaskinset1 and dsaskinset2                  ***
*** before calling dsaskinset3                                 ***
***                                                            ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-02-28                                             ***
****************************************************************

      subroutine dsaskinset3
```

dsasscscsSHffb.f

```
****************************************************************
*** SUBROUTINE dsasscscsSHffb                                  ***
*** computes ASx and ASy coefficients for                      ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)       ***
*** for a Higgs boson in the S channel                         ***
***                                                            ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-03-03                                             ***
****************************************************************

      SUBROUTINE dsasscscsSHffb(kp1,kp2,kp3,kp4,kph)
```

dsasscscsSHffbcol.f

```
****************************************************************
*** SUBROUTINE dsasscscsSHffbcol                               ***
*** computes ASx and ASy coefficients for                      ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)       ***
*** for a Higgs boson in the S channel                         ***
***                                                            ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                        ***
*** Date: 01-03-03                                             ***
****************************************************************

      SUBROUTINE dsasscscsSHffbcol(kp1,kp2,kp3,kp4,kph)
```

dsasscscsSVffb.f

```
************************************************************
*** SUBROUTINE dsasscscsSVffb                            ***
*** computes ASx and ASy coefficients for                ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)  ***
*** for a vector boson in the S channel                  ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-03-03                                       ***
************************************************************

      SUBROUTINE dsasscscsSVffb(kp1,kp2,kp3,kp4,kpv)
```

dsasscscsSVffbcol.f

```
************************************************************
*** SUBROUTINE dsasscscsSVffbcol                         ***
*** computes ASx and ASy coefficients for                ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)  ***
*** for a vector boson in the S channel                  ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-03-03                                       ***
************************************************************

      SUBROUTINE dsasscscsSVffbcol(kp1,kp2,kp3,kp4,kpv)
```

dsasscscsTCffb.f

```
************************************************************
*** SUBROUTINE dsasscscsTCffb                            ***
*** computes ASx and ASy coefficients for                ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)  ***
*** for a neutralino or chargino in the T channel        ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-03-03                                       ***
************************************************************

      SUBROUTINE dsasscscsTCffb(kp1,kp2,kp3,kp4,kpchi)
```

dsasscscsTCffbcol.f

```
************************************************************
*** SUBROUTINE dsasscscsTCffbcol                         ***
*** computes ASx and ASy coefficients for                ***
*** scalar(1) + scalar*(2) -> fermion(3) + fermionbar(4)  ***
*** for a neutralino or chargino in the T channel        ***
***                                                      ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                  ***
*** Date: 01-03-03                                       ***
************************************************************

      SUBROUTINE dsasscscsTCffbcol(kp1,kp2,kp3,kp4,kpchi)
```

dsasscscTCff.f

```
************************************************************
*** SUBROUTINE dsasscscTCff                              ***
*** computes ASx and ASy coefficients for                ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)      ***
*** for a neutralino or chargino in the T channel        ***
***                                                      ***
```

```
*** AUTHOR: Piero Ullio, ullio@sissa.it                    ***
*** Date: 01-03-03                                         ***
*************************************************************

      SUBROUTINE dsasscscTCff(kp1,kp2,kp3,kp4,kpchi)
```

dsasscscTCffcol.f

```
*************************************************************
*** SUBROUTINE dsasscscTCffcol                             ***
*** computes ASx and ASy coefficients for                  ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)       ***
*** for a neutralino or chargino in the T channel          ***
***                                                        ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                    ***
*** Date: 01-03-03                                         ***
*************************************************************

      SUBROUTINE dsasscscTCffcol(kp1,kp2,kp3,kp4,kpchi)
```

dsasscscUCff.f

```
*************************************************************
*** SUBROUTINE dsasscscTCff                                ***
*** computes ASx and ASy coefficients for                  ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)       ***
*** for a neutralino or chargino in the U channel          ***
***                                                        ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                    ***
*** Date: 01-03-03                                         ***
*************************************************************

      SUBROUTINE dsasscscUCff(kp1,kp2,kp3,kp4,kpchi)
```

dsasscscUCffcol.f

```
*************************************************************
*** SUBROUTINE dsasscscTCffcol                             ***
*** computes ASx and ASy coefficients for                  ***
*** scalar(1) + scalar(2) -> fermion(3) + fermion(4)       ***
*** for a neutralino or chargino in the U channel          ***
***                                                        ***
*** AUTHOR: Piero Ullio, ullio@sissa.it                    ***
*** Date: 01-03-03                                         ***
*************************************************************

      SUBROUTINE dsasscscUCffcol(kp1,kp2,kp3,kp4,kpchi)
```

dsaswcomp.f

```
      subroutine dsaswcomp(p,costheta,kp1,kp2,dwbsmax,dmbsmax)
c----------------------------------------------------------------------
c  Routine to compare annihilation cross sections to find
c  out how big they are
c    p - initial cm momentum (real) for lsp annihilations
c    costheta - cosine of c.m. annihilation angle
c  uses dsandwdcosnn, dsandwdcoscn and dsandwdcoscc
c  author: joakim edsjo (edsjo@fysik.su.se)
c  date: 02-05-23
c  modified: Joakim Edsjo, 02-10-22
c======================================================================
```

39.9 mssm/an_stu: t, u and s diagrams for ff-annihilation

39.9.1 Annihilation amplitudes for fermion-fermion annihilation

In this directory, all the helicity amplitudes needed for neutralino-neutralino, neutralino-chargino and chargino-chargino annihilation are calculated. The helicity amplitudes have been calculated with general expressions for vertices, masses etc. in Reduce and converted to Fortran files. The calculation of these are described in more detail in [13].

Each routine here adds the contribution to the helicity amplitudes from one particular diagram and the sum over contributed diagrams is done in the routines **an/dsandwdcosnn**, **an/dsandwdcoscn** and **an/dsandwdcoscc**. The naming convention for the routines here is the following: The first part of the routine name is **dsan** to indicate that they deal with annihilations in DarkSUSY. The next character tells which kind of process it is s-, t- or u-channel and the next two caracters tell which initial state particles we have (**f** for fermion), the next character is the kind of propagating particle (**f** for fermion, **s** for scalar and **v** for vector boson), and finally, the last two characters tell the kind of final state particles. So, to take an example, the routine **dsansffsvv** calculates the helicity amplitudes for annihilation of two fermions to two vector bosons via s-channel exchange of a scalar. There are also a few special cases (routines ending in **ex** or **in**) for diagrams with clashing arrows.

39.9.2 Routine headers – fortran files

dsansffsff.f

```
*****************************************************
*** subroutine dsansffsff                         ***
*** fermion + fermion -> fermion + fermion in     ***
*** s-channel scalar exchange (index k)           ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsansffsff(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffsss.f

```
*****************************************************
*** subroutine dsansffsss                         ***
*** fermion + fermion -> scalar + scalar in       ***
*** s-channel scalar exchange (index k)           ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsansffsss(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffssv.f

```
*****************************************************
*** subroutine dsansffssv                         ***
*** fermion + fermion -> scalar + gauge boson in  ***
*** s-channel scalar exchange (index k)           ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
```

```
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsansffssv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffsvs.f

```
*****************************************************
*** subroutine dsansffsvs                         ***
*** fermion + fermion -> scalar + gauge boson in ***
*** s-channel scalar exchange (index k)           ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se         ***
*****************************************************

      subroutine dsansffsvs(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffsvv.f

```
*******************************************************
*** subroutine dsansffsvv                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** s-channel scalar exchange (index k)             ***
*** 1 - arrow in, 2 - arrow out, k intermediate     ***
*** this code is computer generated by reduce       ***
*** and gentran.                                    ***
*** author: joakim edsjo, edsjo@fysik.su.se           ***
*******************************************************

      subroutine dsansffsvv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffvff.f

```
*******************************************************
*** subroutine dsansffvff                           ***
*** fermion + fermion -> fermion + fermion in       ***
*** s-channel gauge boson exchange (index k)        ***
*** 1 - arrow in, 2 - arrow out, k intermediate     ***
*** this code is computer generated by reduce       ***
*** and gentran.                                    ***
*** author: joakim edsjo, edsjo@fysik.su.se           ***
*******************************************************

      subroutine dsansffvff(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffvss.f

```
*******************************************************
*** subroutine dsansffvss                           ***
*** fermion + fermion -> scalar + scalar in         ***
*** s-channel gauge boson exchange (index k)        ***
*** 1 - arrow in, 2 - arrow out, k intermediate     ***
*** this code is computer generated by reduce       ***
*** and gentran.                                    ***
*** author: joakim edsjo, edsjo@fysik.su.se           ***
*******************************************************

      subroutine dsansffvss(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffvsv.f

```
*******************************************************
*** subroutine dsansffvsv                           ***
*** fermion + fermion -> scalar + gauge boson in     ***
*** s-channel gauge boson exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate      ***
*** this code is computer generated by reduce        ***
*** and gentran.                                     ***
*** author: joakim edsjo, edsjo@fysik.su.se            ***
*******************************************************

      subroutine dsansffvsv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffvvs.f

```
*******************************************************
*** subroutine dsansffvvs                           ***
*** fermion + fermion -> scalar + gauge boson in     ***
*** s-channel gauge boson exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate      ***
*** this code is computer generated by reduce        ***
*** and gentran.                                     ***
*** author: joakim edsjo, edsjo@fysik.su.se            ***
*******************************************************

      subroutine dsansffvvs(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsansffvvv.f

```
*******************************************************
*** subroutine dsansffvvv                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** s-channel gauge boson exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate      ***
*** this code is computer generated by reduce        ***
*** and gentran.                                     ***
*** author: joakim edsjo, edsjo@fysik.su.se            ***
*******************************************************

      subroutine dsansffvvv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffss.f

```
*************************************************
*** subroutine dsantfffss                     ***
*** fermion + fermion -> scalar + scalar in   ***
*** t-channel fermion exchange (index k)      ***
*** 1 - arrow in, 2 - arrow out, k intermediate ***
*** this code is computer generated by reduce ***
*** and gentran.                              ***
*** author: joakim edsjo, edsjo@fysik.su.se     ***
*************************************************

      subroutine dsantfffss(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffssex.f

```
*************************************************
*** subroutine dsantfffssex                   ***
*** fermion + fermion -> scalar + scalar in   ***
*** t-channel fermion exchange (index k)      ***
*** label 3 & 4 exchanged compared to tfffss  ***
*** 1 - arrow in, 2 - arrow out, k intermediate ***
```

```
*** this code is computer generated by reduce    ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffssex(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffssin.f

```
*****************************************************
*** subroutine dsantfffssin                       ***
*** fermion + fermion -> scalar + scalar in       ***
*** t-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** both fermion arrows point inwards             ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffssin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffsv.f

```
*****************************************************
*** subroutine dsantfffsv                         ***
*** fermion + fermion -> scalar + gauge boson in  ***
*** t-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffsv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffsvin.f

```
*****************************************************
*** subroutine dsantfffsvin                       ***
*** fermion + fermion -> scalar + gauge boson in  ***
*** t-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** both fermion arrows point inwards             ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffsvin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffvs.f

```
*****************************************************
*** subroutine dsantfffvs                         ***
*** fermion + fermion -> scalar + gauge boson in  ***
*** t-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffvs(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffvsex.f

```
*****************************************************
*** subroutine dsantfffvsex                       ***
*** fermion + fermion -> gauge boson + scalar in ***
*** t-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** label 3 & 4 exchanged compared to tfffsv      ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
*****************************************************

      subroutine dsantfffvsex(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffvv.f

```
*********************************************************
*** subroutine dsantfffvv                             ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** t-channel fermion exchange (index k)              ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsantfffvv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffvvex.f

```
*********************************************************
*** subroutine dsantfffvvex                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** t-channel fermion exchange (index k)              ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** label 3 & 4 exchanged compared to tfffvv          ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsantfffvvex(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantfffvvin.f

```
*********************************************************
*** subroutine dsantfffvvin                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** t-channel fermion exchange (index k)              ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** both fermion arrows point inwards                 ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsantfffvvin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsantffsff.f

```
*****************************************************
*** subroutine dsantffsff                         ***
*** fermion + fermion -> fermion + fermion in     ***
```

```
*** t-channel scalar exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** this code is computer generated by reduce    ***
*** and gentran.                                 ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
**************************************************

      subroutine dsantffsff(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffss.f

```
**************************************************
*** subroutine dsanufffss                        ***
*** fermion + fermion -> scalar + scalar in      ***
*** u-channel fermion exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** this code is computer generated by reduce    ***
*** and gentran.                                 ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
**************************************************

      subroutine dsanufffss(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffssin.f

```
**************************************************
*** subroutine dsanufffssin                      ***
*** fermion + fermion -> scalar + scalar in      ***
*** u-channel fermion exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** both fermion arrows point inwards            ***
*** this code is computer generated by reduce    ***
*** and gentran.                                 ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
**************************************************

      subroutine dsanufffssin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffsv.f

```
**************************************************
*** subroutine dsanufffsv                        ***
*** fermion + fermion -> scalar + gauge boson in ***
*** u-channel fermion exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** this code is computer generated by reduce    ***
*** and gentran.                                 ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
**************************************************

      subroutine dsanufffsv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffsvin.f

```
**************************************************
*** subroutine dsanufffsvin                      ***
*** fermion + fermion -> scalar + gauge boson in ***
*** u-channel fermion exchange (index k)         ***
*** 1 - arrow in, 2 - arrow out, k intermediate  ***
*** both fermion arrows point inwards            ***
*** this code is computer generated by reduce    ***
*** and gentran.                                 ***
*** author: joakim edsjo, edsjo@fysik.su.se        ***
```

```
*****************************************************

      subroutine dsanufffsvin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffvs.f

```
*****************************************************
*** subroutine dsanufffvs                         ***
*** fermion + fermion -> scalar + gauge boson in ***
*** u-channel fermion exchange (index k)          ***
*** 1 - arrow in, 2 - arrow out, k intermediate   ***
*** this code is computer generated by reduce     ***
*** and gentran.                                  ***
*** author: joakim edsjo, edsjo@fysik.su.se         ***
*****************************************************

      subroutine dsanufffvs(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffvv.f

```
*********************************************************
*** subroutine dsanufffvv                             ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** u-channel fermion exchange (index k)              ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsanufffvv(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffvvex.f

```
*********************************************************
*** subroutine dsanufffvvex                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** u-channel fermion exchange (index k)              ***
*** label 3 & 4 exchanged compared to ufffvv          ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsanufffvvex(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanufffvvin.f

```
*********************************************************
*** subroutine dsanufffvvin                           ***
*** fermion + fermion -> gauge boson + gauge boson in ***
*** u-channel fermion exchange (index k)              ***
*** 1 - arrow in, 2 - arrow out, k intermediate       ***
*** both fermion arrows point inwards                 ***
*** this code is computer generated by reduce         ***
*** and gentran.                                      ***
*** author: joakim edsjo, edsjo@fysik.su.se             ***
*********************************************************

      subroutine dsanufffvvin(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

dsanuffsff.f

```
************************************************
*** subroutine dsanuffsff                    ***
*** fermion + fermion -> fermion + fermion in ***
*** u-channel scalar exchange (index k)       ***
*** 1 - arrow in, 2 - arrow out, k intermediate ***
*** this code is computer generated by reduce ***
*** and gentran.                              ***
*** author: joakim edsjo, edsjo@fysik.su.se     ***
************************************************

      subroutine dsanuffsff(p,costheta,kp1,kp2,kpk,kp3,kp4)
```

39.10 mssm/an_yield: Annihilation yields

39.10.1 Routine headers – fortran files

dsaninit_mssm.f

```
************************************************************************
***   subroutine dsaninitmodel initializes model dependent parts of
***   annihilation routines. Specifically channel numbers, scalar decays
***   etc. Needs to be called once (typically from dsinit_module)
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: May, 2014
*** modified: 2014-11-11 Torsten Bringmann
***           (added explicit annihilation channel list with PDG codes)
***           2016-02-06 Torsten Bringmann
***           (added contributions from electroweak internal bremsstrahlung)
************************************************************************

      subroutine dsaninit_mssm
```

dsanyield.f

```
************************************************************************
***   function dsanyield gives the total yield of positrons, cont. gammas
***   or neutrinos coming from WIMP annihilation in the halo
***   the yields are given as number / annihilation. the energy egev
***   is the threshold for integrated yields and the energy for
***   differential yields. the yields are
***     yieldk =  51: integrated positron yields
***     yieldk =  52: integrated cont. gammas
***     yieldk =  53: integrated muon neutrinos
***     yieldk =  54: integrated antiproton yields
***     yieldk =  59: integrated anti-deuteron yields (old. sph. coal. model)
***                   obsolete, use 61 instead. Only works for differential
***                   yields
***     yieldk =  61: integrated anti-deuteron yields (Pythia6 MC, default dbar)
***     yieldk =  62: integrated electron neutrinos
***     yieldk =  63: integrated tau neutrinos
***     yieldk =  71: integrated neutrino yields (same as 53, but older version)
***     yieldk =  72: integrated muon yields at creation
***     yieldk =  73: integrated muon yields in ice
***     yieldk = above+100: differential in energy
***
*** The scalar decay properties are setup in this routine before
*** calling the cascade routines in an_yield_casc
***
*** istat will be zero in case of no errors. Otherwise its bits are set as
```

```
***   bit  decimal   reason
***     0        1   some inaccesible parts the differential spectra
***                  has been wanted, and the returned yield should then
***                  be treated as a lower bound.
***     1        2   energetically forbidden annihilation channels have been
***                  wanted.
***     2        4   problems with dsIBf_intdxdy integration for IB yields
***                  (only used for MSSM)
***     3        8   problems with dsIBf_intdy integration for IB yields
***                  (only used for MSSM)

*** author: joakim edsjo  edsjo@fysik.su.se
*** date: 98-01-29
*** modified: 98-04-15
*** modified: 2007-05-01 Torsten Bringmann (IB contribution added)
*** modified: 2008-01-15 Joakim Edsjo, made more modular
*** modified: April, 2014. Joakim Edsjo, split simulated part into
*** src/an_yield and model-dependent cascade decays into
*** src_models/mssm/an_yield_casc.
*** modified: 2011-11-11 Torsten Bringmann (changed to PDG code sum)
*** modified: 2016-02-06 Torsten Bringmann (added electroweak IB)
*** modified: 2016-04-06 Torsten Bringmann (added gluon IB)
*** modified: 2016-04-12 Joakim Edsjo (added dbars, both MC and old sph. coal)
*** modified: 2022-04-28 Torsten Bringmann (added channel 62,63 = other nu flavours)
**********************************************************************

      real*8 function dsanyield(egev,yieldk,istat)
```

dsanyield_ch.f

```
**********************************************************************
*** function dsanyield_ch calculates the yield above threshold
*** or the differential flux for a given fluxtype
***
***  input -- mwimp      : DM mass (in GeV)
***           egev       : energy of yield particle
***           pdg1, pdg2 : PDG codes of final state particles
***           yieldk     : fluxtype according to the following table:
***
***                        particle       integrated yield     differential yield
***                        --------       ----------------     ------------------
***                        positron                     51                     151
***                        cont. gamma                  52                     152
***                        nu_e and nu_e-bar            62                     162
***                        nu_mu and nu_mu-bar          53                     153
***                        nu_tau and nu_tau-bar        63                     164
***                        antiproton                   54                     154
***                        cont. gamma w/o pi0          55                     155
***                        nu_e and nu_e-bar            56                     156
***                        nu_tau and nu_tau-bar        57                     157
***                        pi0                          58                     158
***                        nu_mu and nu_mu-bar          71                     171
***                                                     (same as 53/153)
***                        muons from nu at creation    72                     172
***                        muons from nu at detector    73                     173
***
***   output -- istat : zero if everything went OK
***
*** the units are (annihilation)**-1
*** for the differential yields, the units are the same plus gev**-1.
***
*** Note 1. The correct data files need to be loaded. This is handled by
*** a call to dsaninit. It is done automatically here upon first call.
***
*** Note 2. These routines do not contain internal bremsstrahlung (IB)
```

```
*** contributions (except those final state radiations (FSR) that are
*** included in the Pythia runs). The full IB contributions are added
*** in dsanyield.f.
***
*** author: joakim edsjo (edsjo@physto.se)
*** date: 98-01-26
*** modified: 08-01-15
*** modified: 08-11-27 pat scott
*** modified: 09-10-20 pat scott
*** modified: 2014-11-11 torsten bringmann (added pdg codes)
*** modified: 2022-04-28 Torsten Bringmann (added channel 62,63 = other nu flavours)
************************************************************************

      real*8 function dsanyield_ch(mwimp,egev,pdg1,pdg2,yieldk,istat)
```

dsanyieldset.f

```
************************************************************************
***   subroutine dsanyieldsetup prepares the common blocks with annihilation
***   channel branching ratios and Higgs decay widths for annihilation
***   yield calculations. These common block variables are used by
***   by both the an_yield and the se_yield routines.
***   Note: This routine should be called whenever a new model is generated,
***         hence typically from dsmodelsetup
***   up things for  cascade decays.
***   This routine is the interface between SUSY and the halo
***   and Earth/Sun annihilation routines.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 08-01-15
*** Modified: December, 2014
*** modified: torsten bringmann, Dec 2013: removed some common block variables
************************************************************************

      subroutine dsanyieldset
```

39.11 mssm/an_yield_casc: Annihilation yields from cascade decays

39.11.1 Routine headers – fortran files

dsandydth.f

```
************************************************************************
*** function dsandydth is the differential yield dyield/dcostheta in the
*** cm system boosted to the lab system (including proper jacobians if
*** we are dealing with a differential yield.
*** the function should be integrated from -1 to 1, by e.g.
*** the routine gadap.
*** units: (annihilation)**-1
************************************************************************

      real*8 function dsandydth(cth)
```

dsandys.f

```
************************************************************************
*** function dsandys is the differential yield dyield/dcostheta in the
*** cm system boosted to the lab system (including proper jacobians if
*** we are dealing with a differential yield. all decay channels of the
*** scalar boson in question are summed.
```

```
*** the function should be integrated from -1 to 1, by e.g.
*** the routine gadap.
*** units: (annihilation)**-1
*** author: joakim edsjo (edsjo@physto.se)
*** date: 1998
*** modified: 98-04-15
************************************************************************

      real*8 function dsandys(cth)
```

dsanemean.f

```
************************************************************************
*** function dsanemean is used to calculate the mean energy of a decay product
*** when a moving particle decays. e0 and m0 are the energy and mass of
*** the moving particle and m1 and m2 are the masses of the decay products.
*** it is the mean energy of m1 that is returned. all energies and masses
*** should be given in gev.
************************************************************************

      real*8 function dsanemean(e0,m0,m1,m2)
```

dsanwspec.f

```
**********************************************************************
*** subroutine dsanwspec dumps the spectrum for which dsanyield_int failed
*** to the file haspec.dat
**********************************************************************

      subroutine dsanwspec(f,a,b,n)
```

dsanyield_int.f

```
      real*8 function dsanyield_int(f,a,b)
c---------------------------------------------------------------------
c  integrate function f between a and b
c  input
c    integration limits a and b
c  called by dsanyieldfth
c  author: joakim edsjo (edsjo@physto.se) 96-05-16
c  based on paolo gondolos wxint.f routine.
c  integration in log, paolo gondolo 99
c=====================================================================
```

dsanyielddec.f

```
************************************************************************
*** function dsanyielddec integrates dsandys over cos theta.
*** the scalar decay channels are summed in dsandys
*** This routine is for the fundamental channels from S1, S2 and S3 decay.
*** units: (annihilation)**-1
************************************************************************

      real*8 function dsanyielddec(eh,hno,egev,yieldk,istat)
```

dsanyieldfth.f

```
************************************************************************
*** function dsanyieldfth integrates dsandydth over cos theta.
*** it is the yield from particle 1 (which decays from m0)
*** that is calculated. particle one corresponds to channel ch.
```

```
*** units: (annihilation)**-1
*****************************************************************************

      real*8 function dsanyieldfth(e0,m0,mp1,mp2,emuthr,ch,yieldk,
     &  istat)
```

dsanyields.f

```
*****************************************************************************
*** function dsanyields calculates the yield above threshold (yieldk=1) or the
*** differential yield (yieldk=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: (annihilation)**-1
*****************************************************************************

      real*8 function dsanyields(eh,egev,hno,yieldk,istat)
```

dsanyields2.f

```
*****************************************************************************
*** function dsanyields2 calculates the yield above threshold (yieldk=1) or the
*** differential yield (yieldk=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
*****************************************************************************

      real*8 function dsanyields2(eh,egev,hno,yieldk,istat)
```

dsanyields3.f

```
*****************************************************************************
*** function dsanyields3 calculates the yield above threshold (yieldk=1) or the
*** differntial yield (yieldk=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
*****************************************************************************

      real*8 function dsanyields3(eh,egev,hno,yieldk,istat)
```

dsanyields4.f

```
*****************************************************************************
*** function dsanyields4 calculates the yield above threshold (yieldk=1) or the
*** differential yield (yieldk=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
*****************************************************************************

      real*8 function dsanyields4(eh,egev,hno,yieldk,istat)
```

39.12 mssm/cr: Cosmic rays

39.12.1 Routine headers – fortran files

dscrsource.f

```
************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : INTERFACE
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***                   Energy is per CR particle (i.e. not per nucleon)
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2014-11-13
************************************************************************

      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***
```

```
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2014-11-13
************************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

39.13 mssm/dd: Direct detection

39.13.1 Direct detection – theory

NOTE: The description below is outdated. DarkSUSY now includes much better form factors.

If neutralinos are indeed the CDM needed on galaxy scales and larger, there should be a substantial flux of these particles in the Milky Way halo. Since the interaction strength is essentially given by the same weak couplings as, e.g., for neutrinos there is a non-negligible chance of detecting them in low-background counting experiments [194]. Due to the large parameter space of MSSM, even with the simplifying assumptions above, there is a rather wide span of predictions for the event rate in detectors of various types. It is interesting, however, that the models giving the largest rates are already starting to be probed by present direct detection experiments [25, 195].

The rate for direct detection of galactic neutralinos, integrated over deposited energy assuming no energy threshold, is

$$R = \sum_i N_i n_\chi \langle \sigma_{i\chi} v \rangle, \tag{39.9}$$

where N_i is the number of nuclei of species i in the detector, n_χ is the local galactic neutralino number density, $\sigma_{i\chi}$ is the neutralino-nucleus elastic cross section, and the angular brackets denote an average over v, the neutralino speed relative to the detector as described in Chapter 21.

In DarkSUSY, the basic quantities computed are the neutralino-nucleon cross sections, which are free of complications related to nuclear structure, and various experimental details like energy threshold, efficiencies etc. However, as a crude estimate of the expected rates in realistic detectors, the total neutralino-nucleus scattering rates can be obtained for ^{76}Ge, Al_2O_3 (sapphire) and NaI. Usually, it is the spin-independent interaction that gives the most important contribution in target materials such as Na, Cs, Ge, I, or Xe, due to the enhancement caused by the coherence of all nucleons in the target nucleus.

The neutralino-nucleus elastic cross section can be written as

$$\sigma_{i\chi} = \frac{1}{4\pi v^2} \int_0^{4m_{i\chi}^2 v^2} \mathrm{d}q^2 G_{i\chi}^2(q^2), \tag{39.10}$$

where $m_{i\chi}$ is the neutralino-nucleus reduced mass, q is the momentum transfer and $G_{i\chi}(q^2)$ is the effective neutralino-nucleus vertex. We write

$$G_{i\chi}^2(q^2) = A_i^2 F_S^2(q^2) G_S^2 + 4\Lambda_i^2 F_A^2(q^2) G_A^2, \tag{39.11}$$

which shows the coherent enhancement factor A_i^2 for the spin-independent cross section (often Λ_i^2 is written as $\lambda^2 J(J+1)$). We assume gaussian nuclear form factors [119]

$$F_S(q^2) = F_A(q^2) = \exp(-q^2 R_i^2/6\hbar^2), \tag{39.12}$$

$$R_i = (0.3 + 0.89 A_i^{1/3})\mathrm{fm}, \tag{39.13}$$

which should provide us with a good approximation of the integrated detection rate [196, 197], in which we are only interested. (To obtain the differential rate with reasonable accuracy, better approximations are needed [198].)

Using heavy-squark effective lagrangians [199, 200, 201, 103, 202], we get

$$G_S = \sum_{q=\mathrm{u,d,s,c,b,t}} \langle \bar{q}q \rangle \left(\sum_{h=H_1,H_2} \frac{g_{h\chi\chi} g_{hqq}}{m_h^2} - \frac{1}{2} \sum_{k=1}^{6} \frac{g_{L\tilde{q}_k \chi q} g_{R\tilde{q}_k \chi q}}{m_{\tilde{q}_k}^2} \right) \tag{39.14}$$

and

$$G_A = \sum_{q=\mathrm{u,d,s}} \Delta q \left(\frac{g_{Z\chi\chi} g_{Zqq}}{m_Z^2} + \frac{1}{8} \sum_{k=1}^{6} \frac{g_{L\tilde{q}_k \chi q}^2 + g_{R\tilde{q}_k \chi q}^2}{m_{\tilde{q}_k}^2} \right). \tag{39.15}$$

The g's are elementary vertices involving the particles indicated by the indices, and they read

$$g_{h\chi\chi} = \begin{cases} (gZ_{\chi 2} - g_y Z_{\chi 1})(-Z_{\chi 3} \cos\alpha + Z_{\chi 4} \sin\alpha), & \text{for } H_1, \\ (gZ_{\chi 2} - g_y Z_{\chi 1})(Z_{\chi 3} \sin\alpha + Z_{\chi 4} \cos\alpha), & \text{for } H_2, \end{cases} \tag{39.16}$$

$$g_{hqq} = \begin{cases} -Y_q \cos\alpha/\sqrt{2}, & \text{for } H_1, \\ +Y_q \sin\alpha/\sqrt{2}, & \text{for } H_2, \end{cases} \tag{39.17}$$

$$g_{Z\chi\chi} = \frac{g}{2\cos\theta_W} (Z_{\chi 3}^2 - Z_{\chi 4}^2) \tag{39.18}$$

$$g_{Zqq} = -\frac{g}{2\cos\theta_W} T_{3q}, \tag{39.19}$$

$$g_{L\tilde{q}_k \chi q} = g_{LL} \Gamma_{QL}^{kq} + g_{RL} \Gamma_{QR}^{kq}, \tag{39.20}$$

$$g_{R\tilde{q}_k \chi q} = g_{LR} \Gamma_{QL}^{kq} + g_{RR} \Gamma_{QR}^{kq}, \tag{39.21}$$

with

$$g_{LL} = -\frac{1}{\sqrt{2}} \left(T_{3q} g Z_{\chi 2} + \frac{1}{3} g_y Z_{\chi 1} \right), \tag{39.22}$$

$$g_{RR} = \sqrt{2} e_q g_y Z_{\chi 1}, \tag{39.23}$$

$$g_{LR} = g_{RL} = \begin{cases} -Y_q Z_{\chi 3}, & \text{for } q = \mathrm{u,c,t}, \\ -Y_q Z_{\chi 4}, & \text{for } q = \mathrm{d,s,b}, \end{cases} \tag{39.24}$$

and

$$Y_q = \begin{cases} m_q/v_2, & \text{for } q = \mathrm{u,c,t}, \\ m_q/v_1, & \text{for } q = \mathrm{d,s,b}. \end{cases} \tag{39.25}$$

Defining ($N = n, p$)

$$f_{Tq}^N \equiv \frac{\langle N | m_q \bar{q} q | N \rangle}{m_N}, \tag{39.26}$$

we take in DarkSUSY the numerical values [203]

$$\begin{aligned} f_{Tu}^p &= 0.023, & f_{Td}^p &= 0.034, \\ f_{Tc}^p &= 0.0595, & f_{Ts}^p &= 0.14, \\ f_{Tt}^p &= 0.0595, & f_{Tb}^p &= 0.0595 \end{aligned} \tag{39.27}$$

and

$$\begin{aligned} f_{Tu}^n &= 0.019, & f_{Td}^n &= 0.041, \\ f_{Tc}^n &= 0.0592 & f_{Ts}^n &= 0.14, \\ f_{Tt}^n &= 0.0592, & f_{Tb}^n &= 0.0592. \end{aligned} \tag{39.28}$$

For the quark contributions to the nucleon spin we take [204]

$$\Delta\mathrm{u} = 0.77, \qquad \Delta\mathrm{d} = -0.40, \qquad \Delta\mathrm{s} = -0.12. \tag{39.29}$$

However, the older set of data [205]

$$\Delta u = 0.77, \qquad \Delta d = -0.49, \qquad \Delta s = -0.15 \tag{39.30}$$

can optionally be used.

Moreover, we take for the Λ factors

$$\Lambda^2_{Al} = 0.087, \qquad \Lambda^2_{Na} = 0.041 \quad \text{and} \quad \Lambda^2_{I} = 0.007, \tag{39.31}$$

according to the odd-group model [206].

39.13.2 Routine headers – fortran files

dsdddn1.f

```
      function dsdddn1(ms,mq,mx)
No header found.
```

dsdddn2.f

```
      function dsdddn2(ms,mq,mx)
No header found.
```

dsdddn3.f

```
      function dsdddn3(ms,mq,mx)
No header found.
```

dsdddn4.f

```
      function dsdddn4(ms,mq,mx)
No header found.
```

dsddgpgn.f

```
************************************************************************
***  function dsddgpgn returns DM nucleon four-fermion couplings      ***
***                                                                   ***
***  type : interface                                                 ***
***         [NB: this routine will eventually be phased out as *interface*  ***
***              routine. Currently, neutrino telescope routines are the    ***
***              only routines in /src that access it. ]                    ***
***                                                                   ***
***   output:                                                         ***
***     gg complex*16 array of couplings                               ***
***          first index is operator index (1:ddng), with ddng in dsddcom.h ***
***          second index is proton/neutron (1:2)                     ***
***     units: GeV^-2                                                 ***
***                                                                   ***
***  author: paolo gondolo (paolo@physics.utah.edu) 2004              ***
***    2008-02-16 paolo gondolo double spin-dependent squark contribution   ***
***    2008-02-16 paolo gondolo undo running quark masses in couplings      ***
***    2016-05-19 Paolo Gondolo, Joakim Edsjo, scheme setup                 ***
***    2016-11-20 Paolo Gondolo abandoned scheme                            ***
************************************************************************
      subroutine dsddgpgn(gg,ierr)
```

dsddo.f

```
      function dsddo(k,z,qsq)
c     k=0 Og, k=1 Ou, k=2 Od, ..... k=6 Ob
c     z=1 proton, z=2 neutron
c
c     This function is somehow related to the second moment of the
c     twist-2 parton matrix element. In the notation of Drees and
c     Nojiri, and dropping the momentum dependence and the index 2
c     indicating the second moment, one has Sigma+G=1, where Sigma
c     is the sum of the moments for each quark, Sigma=\sum q.
c     One has
c     <N|O^{(2)}_{q mu nu}|N> = (p_mu p_nu-m_N^2 g_{mu nu}/4)/m_N q,
c     and one needs \sum g_q q. From DN eq. (32),
c     with Op=O_{+}, Om=O_{-}, and a=alpha(m0^2)/alpha(Q0^2),
c     \sum g_q q =
c       g_u [2 (15/31+Op a^{62/69})/5 + (Ou-Od)/2 + Sigma/10 a^{32/69}]
c     + g_d [2 (15/31+Op a^{62/69})/5 - (Ou-Od)/2 + Sigma/10 a^{32/69}]
c     + g_b [(15/31+Op a^{62/69})/5 - 2 Sigma/10 a^{32/69}]
c     + top quark (treated using Dn (17), (18), (33))
c
```

dsddset_mssm.f

```
      subroutine dsddset_mssm(cs)
c...set parameters for mssm scattering cross section
c...  cs - character string specifying choice to be made
c...author: paolo gondolo 2000-07-07
c...modified by Gintaras Duda 2007-06-27 for new FF options
c...modified by Paolo Gondolo 2008-02-18
c...modified by Paolo Gondolo 2013-11-29
c...modified by torsten bringmann 2018-06-13: changed default, and allowed to set
c...                                          poles and DN independently
```

39.14 mssm/examples: src_models/mssm/examples

39.14.1 Routine headers – fortran files

dsslha2slha.f

```
c...The include below includes the standard templates for user-definable
c...functions. The templates are in src/templates/. If you want to use your
c...own functions, copy one of these templates and the templates-standard.f
c...and modify templates-standard.f to load your new modified user function.
```

dstemp.f

```
c...The include below includes the standard templates for user-definable
c...functions. The templates are in src/templates/. If you want to use your
c...own functions, copy one of these templates and the templates-standard.f
c...and modify templates-standard.f to load your new modified user function.
```

dstest-isasugra.f

```
c...The include below includes the standard templates for user-definable
c...functions. The templates are in src/templates/. If you want to use your
c...own functions, copy one of these templates and the templates-standard.f
c...and modify templates-standard.f to load your new modified user function.
```

39.15 mssm/examples_aux: src_models/mssm/examples_aux

39.15.1 Routine headers – fortran files

random_model.f

```
      subroutine random_model()
c
c     To generate MSSM7 model parameters in a random way
c
```

write_model.f

```
      subroutine write_model(lunit)
c
c     To write MSSM7 model parameters to unit lunit
c
```

39.16 mssm/ge: General SUSY model setup: masses, vertices etc

39.16.1 Supersymmetric model

We will here review the definition of the MSSM as given in [1].

39.16.1.1 Parameters

In our notation, the superpotential and the soft supersymmetry-breaking scalar potential minimal supersymmetric standard model (MSSM) with R-parity conservation [159, 160, 207] read respectively

$$W = \epsilon_{ij}\left(-\hat{\mathbf{e}}_R^*\mathbf{h}_E\hat{\mathbf{l}}_L^i\hat{H}_1^j - \hat{\mathbf{d}}_R^*\mathbf{h}_D\hat{\mathbf{q}}_L^i\hat{H}_1^j + \hat{\mathbf{u}}_R^*\mathbf{h}_U\hat{\mathbf{q}}_L^i\hat{H}_2^j - \mu\hat{H}_1^i\hat{H}_2^j\right), \tag{39.32}$$

$$\begin{aligned} V_{\rm soft} = {} & \epsilon_{ij}\Big(-\tilde{\mathbf{e}}_R^*\mathbf{A}_E\mathbf{h}_E\tilde{\mathbf{l}}_L^iH_1^j - \tilde{\mathbf{d}}_R^*\mathbf{A}_D\mathbf{h}_D\tilde{\mathbf{q}}_L^iH_1^j + \tilde{\mathbf{u}}_R^*\mathbf{A}_U\mathbf{h}_U\tilde{\mathbf{q}}_L^iH_2^j - B\mu H_1^iH_2^j \\ & \quad +\text{h.c.}) \\ & +H_1^{i*}m_1^2H_1^i + H_2^{i*}m_2^2H_2^i \\ & +\tilde{\mathbf{q}}_L^{i*}\mathbf{M}_Q^2\tilde{\mathbf{q}}_L^i + \tilde{\mathbf{l}}_L^{i*}\mathbf{M}_L^2\tilde{\mathbf{l}}_L^i + \tilde{\mathbf{u}}_R^*\mathbf{M}_U^2\tilde{\mathbf{u}}_R + \tilde{\mathbf{d}}_R^*\mathbf{M}_D^2\tilde{\mathbf{d}}_R + \tilde{\mathbf{e}}_R^*\mathbf{M}_E^2\tilde{\mathbf{e}}_R. \end{aligned} \tag{39.33}$$

Here i and j are SU(2) indices ($\epsilon_{12} = +1$), $\mathbf{h}$'s, $\mathbf{A}$'s and $\mathbf{M}$'s are 3×3 matrices in generation space, and the other boldface letters are vectors in generation space.

The current version of DarkSUSY uses only a restricted set of parameters. Namely the number of free parameters (a grand total of 124 [208]) is reduced by setting the off-diagonal elements of the $\mathbf{A}$'s and $\mathbf{M}$'s to zero and imposing CP conservation (except in the CKM matrix).

39.16.1.2 Mass spectrum

For easy reference, we now give the particle mass matrices, together with our convention for the mixing matrices.

Concerning the Higgs sector, we choose as independent parameters $\tan\beta$ and the mass m_A of the CP-odd Higgs boson. The code provides five options for the calculation of the Higgs masses:

	Higgs boson			
Channel	H_1^0	H_2^0	H_3^0	H^+
i=	j=1	j=2	j=3	j=4
1	$c\bar{c}$	$c\bar{c}$	$c\bar{c}$	ud
2	$b\bar{b}$	$b\bar{b}$	$b\bar{b}$	$u\bar{s}$
3	$t\bar{t}$	$t\bar{t}$	$t\bar{t}$	$u\bar{b}$
4	$\tau^+\tau^-$	$\tau^+\tau^-$	$\tau^+\tau^-$	$c\bar{d}$
5	W^+W^-	W^+W^-	–	$c\bar{s}$
6	Z^0Z^0	Z^0Z^0	–	$c\bar{b}$
7	–	$H_1^0H_1^0$	–	$t\bar{d}$
8	$H_2^0H_2^0$	–	–	$t\bar{s}$
9	$H_3^0H_3^0$	$H_3^0H_3^0$	–	$t\bar{b}$
10	H^+H^-	H^+H^-	–	$\nu_e e^+$
11	–	–	ZH_1^0	$\nu_\mu\mu^+$
12	–	–	ZH_2^0	$\nu_\tau\tau^+$
13	ZH_3^0	ZH_3^0	–	$W^+H_1^0$
14	W^+H^-/W^-H^+	W^+H^-/W^-H^+	W^+H^-/W^-H^+	$W^+H_2^0$
15	$\mu^+\mu^-$	$\mu^+\mu^-$	$\mu^+\mu^-$	$W^+H_3^0$
16	$s\bar{s}$	$s\bar{s}$	$s\bar{s}$	–
17	gg	gg	gg	–
18	$\gamma\gamma$	$\gamma\gamma$	$\gamma\gamma$	–
19	$Z^0\gamma$	$Z^0\gamma$	$Z^0\gamma$	–
20	$\tilde{f}\tilde{f}'$	$\tilde{f}\tilde{f}'$	$\tilde{f}\tilde{f}'$	$\tilde{f}\tilde{f}'$

Table 39.2: Higgs partial widths hdwidth(i,j). Index i refers to the decay channel and index j to the Higgs boson. All widths are given in GeV. Note that typically we have that $m_{H_2} < m_{H_3} < m_{H^+} < m_{H_1}$ so many of these decay channels are not kinematically allowed, but included for completeness. If the HDECAY interface is used, the channels where $m_{H_2} < m_{H_3} < m_{H^+} < m_{H_1}$ is not satisfied are not included. Channels 16–19 are only included if HDECAY is used.

higloop=0: tree level formulas; higloop=1: the effective potential approach in [209, 210, 211] (correcting the sign of μ in eq. (4) of [211]); higloop=2: the effective potential approach in [212] with addition of D-terms and correction of some signs and numerical factors; higloop=3: the analytical approximations to the RGE-improved effective potential in [213]; higloop=4: the pole mass calculation in [214]; higloop=5: FeynHiggs (requires FeynHiggs to be installed, default) [215]

The masses of the Higgs bosons are obtained from

$$\mathcal{M}_H^2 = \begin{pmatrix} m_Z^2\cos^2\beta + m_A^2\sin^2\beta + \Delta_{11} & -\sin\beta\cos\beta(m_Z^2+m_A^2)+\Delta_{12} \\ -\sin\beta\cos\beta(m_Z^2+m_A^2)+\Delta_{21} & m_Z^2\sin^2\beta + m_A^2\cos^2\beta + \Delta_{22} \end{pmatrix} \tag{39.34}$$

$$m_{H^\pm}^2 = m_A^2 + m_W^2 + \Delta_\pm. \tag{39.35}$$

The quantities Δ_{ij} and $\Delta_\pm$ are the one-loop radiative corrections, calculated acording to the value of higloop as described above. Diagonalization of $\mathcal{M}_H^2$ gives the two CP-even Higgs boson masses, $m_{H_{1,2}}$, and their mixing angle α ($-\pi/2 < \alpha < 0$). For higloop=4, the pole masses are then obtained solving $m_{H_i}^{2\text{pole}} = m_{H_i}^2 + \Pi_{ii}(m_{H_i}^{2\text{pole}}) - \Pi_{ii}(0)$, where $\Pi_{ii}(p^2)$ is H_iH_i the self-energy. In this case, m_{H_3} is the pole mass and m_A is the running mass.

The Higgs widths are calculated at tree level, but with QCD corrections. The decays to supersymmetric particles are also included in the total width, so the sum of the partial widths in Table 39.2 does not necessarily sum up to the total width given in width(k). The loop corrections are also available via an interface to HDECAY.

The neutralinos $\tilde{\chi}_i^0$ are linear combinations of the neutral gauginos $\tilde{B}$, $\tilde{W}_3$ and of the neutral

higgsinos $\tilde{H}_1^0$, $\tilde{H}_2^0$. In this basis, we write their mass matrix as

$$\mathcal{M}_{\tilde{\chi}^0_{1,2,3,4}} = \begin{pmatrix} M_1 & 0 & -m_Z s_W c_\beta & +m_Z s_W s_\beta \\ 0 & M_2 & +m_Z c_W c_\beta & -m_Z c_W s_\beta \\ -m_Z s_W c_\beta & +m_Z c_W c_\beta & \delta_{33} & -\mu \\ +m_Z s_W s_\beta & -m_Z c_W s_\beta & -\mu & \delta_{44} \end{pmatrix}, \tag{39.36}$$

with $c_W = \cos\theta_W$, $s_W = \sin\theta_W$, $c_\beta = \cos\beta$, and $s_\beta = \sin\beta$. Here δ_{33} and δ_{44} are radiative corrections important when two higgsinos are close in mass. Their explicit expressions are from ref. [165]. To neglect these radiative corrections set neuloop=0 instead of neuloop=1 (default). The neutralino mass eigenstates are written as

$$\tilde{\chi}_i^0 = N_{i1}\tilde{B} + N_{i2}\tilde{W}^3 + N_{i3}\tilde{H}_1^0 + N_{i4}\tilde{H}_2^0. \tag{39.37}$$

The phases of N_{ij} are chosen so that the neutralino masses $m_{\tilde{\chi}_i^0} \geq 0$.

The charginos are linear combinations of the charged gauge bosons $\tilde{W}^\pm$ and of the charged higgsinos $\tilde{H}_1^-$, $\tilde{H}_2^+$. Their mass matrix,

$$\mathcal{M}_{\tilde{\chi}^\pm} = \begin{pmatrix} M_2 & \sqrt{2} m_W \sin\beta \\ \sqrt{2} m_W \cos\beta & \mu \end{pmatrix}, \tag{39.38}$$

is diagonalized by the following linear combinations

$$\tilde{\chi}_i^- = U_{i1}\tilde{W}^- + U_{i2}\tilde{H}_1^-, \tag{39.39}$$

$$\tilde{\chi}_i^+ = V_{i1}\tilde{W}^+ + V_{i2}\tilde{H}_1^+. \tag{39.40}$$

We choose $\det(U) = 1$ and $U^* \mathcal{M}_{\tilde{\chi}^\pm} V^\dagger = \mathrm{diag}(m_{\tilde{\chi}_1^\pm}, m_{\tilde{\chi}_2^\pm})$ with non-negative chargino masses $m_{\tilde{\chi}_i^\pm} \geq 0$.

When discussing the squark mass matrix including mixing, it is convenient to choose a basis where the squarks are rotated in the same way as the corresponding quarks in the standard model. We follow the conventions of the particle data group [216] and put the mixing in the left-handed d-quark fields, so that the definition of the Cabibbo-Kobayashi-Maskawa matrix is $\mathbf{K} = \mathbf{V}_1 \mathbf{V}_2^\dagger$, where $\mathbf{V}_1$ ($\mathbf{V}_2$) rotates the interaction left-handed u-quark (d-quark) fields to mass eigenstates. For sleptons we choose an analogous basis, but due to the masslessness of neutrinos no analog of the CKM matrix appears.

We then obtain the general 6×6 $\tilde{u}$- and $\tilde{d}$-squark mass matrices:

$$\mathcal{M}_{\tilde{u}}^2 = \begin{pmatrix} \mathbf{M}_Q^2 + \mathbf{m}_u^\dagger \mathbf{m}_u + D_{LL}^u \mathbf{1} & \mathbf{m}_u^\dagger (\mathbf{A}_U^\dagger - \mu^* \cot\beta) \\ (\mathbf{A}_U - \mu \cot\beta)\mathbf{m}_u & \mathbf{M}_U^2 + \mathbf{m}_u \mathbf{m}_u^\dagger + D_{RR}^u \mathbf{1} \end{pmatrix}, \tag{39.41}$$

$$\mathcal{M}_{\tilde{d}}^2 = \begin{pmatrix} \mathbf{K}^\dagger \mathbf{M}_Q^2 \mathbf{K} + \mathbf{m}_d \mathbf{m}_d^\dagger + D_{LL}^d \mathbf{1} & \mathbf{m}_d^\dagger (\mathbf{A}_D^\dagger - \mu^* \tan\beta) \\ (\mathbf{A}_D - \mu \tan\beta)\mathbf{m}_d & \mathbf{M}_D^2 + \mathbf{m}_d^\dagger \mathbf{m}_d + D_{RR}^d \mathbf{1} \end{pmatrix}, \tag{39.42}$$

and the general sneutrino and charged slepton mass matrices

$$\mathcal{M}_{\tilde{\nu}}^2 = \mathbf{M}_L^2 + D_{LL}^\nu \mathbf{1} \tag{39.43}$$

$$\mathcal{M}_{\tilde{e}}^2 = \begin{pmatrix} \mathbf{M}_L^2 + \mathbf{m}_e \mathbf{m}_e^\dagger + D_{LL}^e \mathbf{1} & \mathbf{m}_e^\dagger (\mathbf{A}_E^\dagger - \mu^* \tan\beta) \\ (\mathbf{A}_E - \mu \tan\beta)\mathbf{m}_e & \mathbf{M}_E^2 + \mathbf{m}_e^\dagger \mathbf{m}_e + D_{RR}^e \mathbf{1} \end{pmatrix}. \tag{39.44}$$

Here

$$D_{LL}^f = m_Z^2 \cos 2\beta (T_{3f} - e_f \sin^2\theta_w), \tag{39.45}$$

$$D_{RR}^f = m_Z^2 \cos 2\beta e_f \sin^2\theta_w. \tag{39.46}$$

Table 39.3: Particle codes (synonyms are separated by commas).

ν_e	knue,knu(1)	γ	kgamma	$\tilde{\chi}_i^0$	kn(i) $i=1\ldots4$	$\tilde{u}_1$	ksu(1),ksqu(1)
e	ke,kl(1)	$W^\pm$	kw	$\tilde{\chi}_k^\pm$	kcha(k) $k=1,2$	$\tilde{u}_2$	ksu(2),ksqu(4)
ν_μ	knumu,knu(2)	Z^0	kz	$\tilde{g}$	kgluin	$\tilde{d}_1$	ksd(1),ksqd(1)
μ	kmu,kl(2)	g	kgluon	$\tilde{\nu}_e$	ksnue,ksnu(1)	$\tilde{d}_2$	ksd(2),ksqd(4)
ν_τ	knutau,knu(3)			$\tilde{e}_1$	kse(1),ksl(1)	$\tilde{c}_1$	ksc(1),ksqu(2)
τ	ktau,kl(3)			$\tilde{e}_2$	kse(2),ksl(4)	$\tilde{c}_2$	ksc(2),ksqu(5)
u	ku,kqu(1)	H^0	kh1	$\tilde{\nu}_\mu$	ksnumu,ksnu(2)	$\tilde{s}_1$	kss(1),ksqd(2)
d	kd,kqd(1)	h^0	kh2	$\tilde{\mu}_1$	ksmu(1),ksl(2)	$\tilde{s}_2$	kss(2),ksqd(5)
c	kc,kqu(2)	A^0	kh3	$\tilde{\mu}_2$	ksmu(2),ksl(5)	$\tilde{b}_1$	ksb(1),ksqd(3)
s	ks,kqd(2)	$H^\pm$	khc	$\tilde{\nu}_\tau$	ksnuta,ksnu(3)	$\tilde{b}_2$	ksb(2),ksqd(6)
b	kb,kqd(3)	G^0	kgold0	$\tilde{\tau}_1$	kstau(1),ksl(3)	$\tilde{t}_1$	kst(1),ksqu(3)
t	kt,kqu(3)	$G^\pm$	kgoldc	$\tilde{\tau}_2$	kstau(2),ksl(6)	$\tilde{t}_2$	kst(2),ksqu(6)

In the chosen basis, $\mathbf{m}_u = \mathrm{diag}(m_\mathrm{u}, m_\mathrm{c}, m_\mathrm{t})$, $\mathbf{m}_d = \mathrm{diag}(m_\mathrm{d}, m_\mathrm{s}, m_\mathrm{b})$ and $\mathbf{m}_e = \mathrm{diag}(m_e, m_\mu, m_\tau)$.

The slepton and squark mass eigenstates $\tilde{f}_k$ ($\tilde{\nu}_k$ with $k=1,2,3$ and $\tilde{e}_k$, $\tilde{u}_k$ and $\tilde{d}_k$ with $k=1,\ldots,6$) diagonalize the previous mass matrices and are related to the current sfermion eigenstates $\tilde{\mathbf{f}}_L$ and $\tilde{\mathbf{f}}_R$ via ($a=1,2,3$)

$$\tilde{f}_{La} = \sum_{k=1}^{6} \tilde{f}_k \mathbf{\Gamma}^{*ka}_{FL}, \tag{39.47}$$

$$\tilde{f}_{Ra} = \sum_{k=1}^{6} \tilde{f}_k \mathbf{\Gamma}^{*ka}_{FR}. \tag{39.48}$$

The squark and charged slepton mixing matrices $\mathbf{\Gamma}_{UL,R}$, $\mathbf{\Gamma}_{DL,R}$ and $\mathbf{\Gamma}_{EL,R}$ have dimension 6×3, while the sneutrino mixing matrix $\mathbf{\Gamma}_{\nu L}$ has dimension 3×3.

This version of DarkSUSY allows only for diagonal matrices $\mathbf{A}_U$, $\mathbf{A}_D$, $\mathbf{A}_E$, $\mathbf{M}_Q$, $\mathbf{M}_U$, $\mathbf{M}_D$, $\mathbf{M}_E$, and $\mathbf{M}_L$. This ansatz, while not being the most general one, implies the absence of tree-level flavor changing neutral currents in all sectors of the model. In this case, the squark mass matrices can be diagonalized analytically. For example, for the top squark one has, in terms of the top squark mixing angle $\theta_{\tilde{t}}$,

$$\Gamma^{\tilde{t}_1\tilde{t}}_{UL} = \Gamma^{\tilde{t}_2\tilde{t}}_{UR} = \cos\theta_{\tilde{t}}, \qquad \Gamma^{\tilde{t}_2\tilde{t}}_{UL} = -\Gamma^{\tilde{t}_1\tilde{t}}_{UR} = \sin\theta_{\tilde{t}}. \tag{39.49}$$

Special values of the sfermion masses can be set with the parameters `msquarks`, and msleptons. If msquarks=msleptons=0, the sfermion masses are obtained with the diagonalization described above. If `msquarks`>0 (or msleptons>0), all squark masses are set to `msquarks` (or all slepton masses to msleptons). Finally, if `msquarks`<0 (or msleptons<0), the squark (or slepton) masses are set equal to the neutralino mass but never less than |msquarks| (or |msleptons|). This is to provide the lightest possible sfermions compatible with a neutralino LSP. In all of these cases, there is no mixing between sfermions.

The particle masses are available in an array mass(p), where p is the particle code from table **??**. Similarly, particle decay width are available as width(p), but currently only the width of the Higgs bosons are calculated, the other particles having fictitious widths of 1 or 5 GeV (for the sole purpose of regularizing annihilation amplitudes close to poles).

39.16.1.3 Three-particle vertices

We define three-particle vertices gl(i,j,k)= g^L_{ijk} and gr(i,j,k)= g^R_{ijk} as follows. We adopt the convention that the order of the particles in the indices is the order in which they appear in the

corresponding lagrangian term, so the last particle is always entering. If there are charged particles in the vertex, they are both assumed positively charged, and the particle that exits the vertex is indexed before the particle that enters.

- Three scalar bosons:
$$\mathcal{L}_{\rm int} = g_{\phi_i\phi_j\phi_k} m_W \phi_i\phi_j\phi_k \tag{39.50}$$
where ϕ_i is a Higgs or a Goldstone boson. In this case, $\mathtt{gl}$=$\mathtt{gr}$=g. Available vertices are $\phi_i\phi_j\phi_k$ = $H_i^0H_j^0H_k^0$, $H_i^0H^-H^+$, $H_i^0A^0A^0$, $H_i^0G^0G^0$, $H_i^0G^-G^+$, $H_i^0G^-H^+$, $H_i^0G^-G^+$, $A^0G^-H^+$, $A^0G^0H_i^0$, and permutations.

- Two scalar and one vector bosons:
$$\mathcal{L}_{\rm int} = g_{V\phi_1\phi_2} V^\mu \phi_1 i\overleftrightarrow{\partial}_\mu \phi_2. \tag{39.51}$$
Available vertices are $V\phi_1\phi_2$ = $Z^0H_i^0A^0$, $Z^0H^-H^+$, γH^-H^+, $W^-H^+A^0$, $W^-H^+H_i^0$, and permutations.

- One scalar and two vector bosons:
$$\mathcal{L}_{\rm int} = g_{\phi V_1V_2} m_W g_{\mu\nu} \phi V_1^\mu V_2^\nu \tag{39.52}$$
Available vertices are ϕV_1V_2 = $H_i^0W^-W^+$, $H_i^0Z^0Z^0$.

- Three vector bosons:
$$ig_{V_1V_2V_3}\left[(k_1-k_3)_\nu g_{\mu\lambda} + (k_3-k_2)_\mu g_{\lambda\nu} + (k_2-k_1)g_{\mu\nu}\right] \tag{39.53}$$
with all momenta incoming and assigned as $V_1^\mu(k_1)$, $V_2^\nu(k_2)$ and $V_3^\lambda(k_3)$. Available vertices are $Z^0W^-W^+$ and γW^-W^+.

- One scalar boson and two Dirac fermions:
$$\mathcal{L}_{\rm int} = \phi\overline{\psi}_1(g^L_{\phi\psi_1\psi_2}P_L + g^R_{\phi\psi_1\psi_2}P_R)\psi_2 \tag{39.54}$$
Available vertices are $\phi\psi_1\psi_2$ =

- One vector boson and two Dirac fermions:
$$\mathcal{L}_{\rm int} = V_\mu\overline{\psi}_1\gamma^\mu(g^L_{V\psi_1\psi_2}P_L + g^R_{V\psi_1\psi_2}P_R)\psi_2 \tag{39.55}$$
Available vertices are $V\psi_1\psi_2$ =

- One scalar boson, one Dirac and one Majorana fermion:
$$\mathcal{L}_{\rm int} = \phi\overline{\psi}(g^L_{\phi\psi\chi}P_L + g^R_{\phi\psi\chi}P_R)\chi \tag{39.56}$$
Available vertices are $\phi\psi\chi$ =

- One vector boson, one Dirac and one Majorana fermion:
$$\mathcal{L}_{\rm int} = V_\mu\overline{\psi}\gamma^\mu(g^L_{V\psi\chi}P_L + g^R_{V\psi\chi}P_R)\chi \tag{39.57}$$
Available vertices are $V\psi\chi$ =

- One scalar boson and two Majorana fermions:
$$\mathcal{L}_{\rm int} = \tag{39.58}$$
Available vertices are...

- One vector boson and two Majorana fermions:

$$\mathcal{L}_{\rm int} = \tag{39.59}$$

Explicit expressions for the coupling constants g_{ijk} can be obtained in [159, 160, 207], with radiative corrections to trilinear scalar couplings in [217]. We have rederived from the superpotential all vertices we have implemented.

Implemented vertices: those listed above plus $Z^0W^\pm W^\mp$, $Z^0H_i^0H_i^0$, $W^\pm H^\mp A^0$, $W^\pm H^\mp H_i^0$, $H_i^0W^\pm W^\mp$, $H_i^0Z^0Z^0$, Z^0A^0H, $H_i^0A^0A^0$, A^0ff, H_i^0ff, Z^0ff, $Z^0\tilde{\chi}^0\tilde{\chi}^0$, $H_i^0\tilde{\chi}^0\tilde{\chi}^0$, $Z^0\tilde{\chi}^0\tilde{\chi}^0$, $W^\mp\tilde{\chi}^0\tilde{\chi}^\pm$, $H^\mp\tilde{\chi}^0\tilde{\chi}^\pm$, $\tilde{q}\tilde{g}q$, $\tilde{f}\tilde{\chi}^0f$, $H_i^0\tilde{\chi}^\pm\tilde{\chi}^\mp$, $A^0\tilde{\chi}^\pm\tilde{\chi}^\mp$, $W^\pm ff'$, $H^\pm ff'$, $\gamma W^\pm W^\mp$, $\gamma H^\pm H^\mp$, $Z^0\tilde{\chi}^\pm\tilde{\chi}^\mp$, $\gamma\tilde{\chi}^\pm\tilde{\chi}^\mp$, γff, GHH, GGH, $G^\mp\tilde{\chi}^0\tilde{\chi}^\pm$.

39.16.2 General supersymmetry – routines

Input parameters, options, results, etc. are contained in common blocks in the file dsmssm.h, which the user has to include. The input parameters are ($a = 1, 2, 3$)

ma= m_A,	tanbe= $\tan\beta$,	mu= μ,	m1= M_1,
m2= M_2,	m3= M_3,	asofte(a)= A_{Eaa},	asoftu(a)= A_{Uaa},
asoftd(a)= A_{Daa},	mass2q(a)= M^2_{Qaa},	mass2l(a)= M^2_{Laa},	mass2u(a)= M^2_{Uaa},
mass2d(a)= M^2_{Daa},	mass2e(a)= M^2_{Eaa}.		

The options are (see previous subsections for a description)
higloop choice of tree-level or radiatively corrected Higgs boson masses;
neuloop choice of tree-level or radiatively corrected neutralino masses;
msquarks,msleptons choice of squark and slepton masses.

To initialize DarkSUSY for a new model, you should call

subroutine **dsmodelsetup**(unphys,warning)

Purpose: To calculate the particle spectrum, widths and couplings.
Output:
- unphys i — non-zero if the model is unphysical
- warning i — non-zero if (typically the Higgs) code has issued a warning.

which calculates couplings, masses and some basic cross sections.

The following subroutines specify the values of the model parameters, and read/write them to a file. The user should create his own versions by editing a copy of them. Please call them with a different name.

subroutine **dsgive_model**(mu,m2,ma,tanbe,msq,atm,abm)

Purpose: Set the MSSM parameters as specified by the arguments.
Inputs:
- mu r8 — The μ parameter in GeV.
- m2 r8 — The M_2 parameter in GeV.
- ma r8 — The mass of the CP-odd Higgs boson, m_A in GeV.
- tanbe r8 — $\tan\beta$.
- msq r8 — Sets $\mathbf{M}^Q$, etc. to a common mass scale m_0 in GeV.
- atm r8 — Sets A_t in units of m_0 (range: -3 — 3).
- abm r8 — Sets A_b in units of m_0 (range: -3 — 3).

There are also routines to setup more general MSSM models, the currently most general being dsgive_model25.

The following subroutines are useful in the analysis.

subroutine **wspctm**(unit)

Purpose: Write the particle mass spectrum and mixing matrices to unit unit.

Inputs:

unit	i	Unit number to write to.

subroutine **wvertx**(unit)

Purpose: Write all non-vanishing three-particle vertices to unit unit.

Inputs:

unit	i	Unit number to write to.

subroutine **wunph**(unit)

Purpose: Write the reason for which the model is not physically acceptable (tachyons, etc.) to unit unit.

Inputs:

unit	i	Unit number to write to.

subroutine **wexcl**(unit)

Purpose: Write the reason(s) for which the model is experimentally excluded to unit $\hat{\text{u}}$nit.

Inputs:

unit	i	Unit number to write to.

subroutine **dswhwar**(unit)

Purpose: Write the reason(s) for which the Higgs calculation issued warnings to unit unit.

Inputs:

unit	i	Unit number to write to.

39.16.3 Routine headers – fortran files

dsb0loop.f

```
      complex*16 function dsb0loop(q,m1,m2)
c-----------------------------------------------------------------------
c  the two-point function b0(q,m1,m1).
c     uses two-point function b_0 from m. drees, k. hagiwara and a.
c     yamada, phys. rev. d45 (1992) 1725.
c  author: joakim edsjo (edsjo@fysik.su.se) 97-02-11
c=======================================================================
```

dschasct.f

```
      subroutine dschasct
c-----------------------------------------------------------------------
c  chargino masses and mixings.
c  called by susyin or mrkin.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995
c     940407 correction to chargino mixing
c     990724 drop v1,v2 (pg)
c=======================================================================
```

dsdmspin.f

```
************************************************************************
*** Function dsdmspin returns the dark matter spin (in hbar)         ***
***                                                                  ***
***  type : interface                                                ***
***                                                                  ***
***  desc : dark matter spin                                         ***
***                                                                  ***
*** author: Paolo Gondolo 2016-11-20                                 ***
************************************************************************

      real*8 function dsdmspin()
```

dsg0loop.f

```
      function dsg0loop(qsq,m1sq,m2sq)
c     a loop function
```

dsg4set.f

```
      subroutine dsg4set(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsg4set12.f

```
      subroutine dsg4set12(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
c  case of two neutral particles (1 and 2)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsg4set1234.f

```
      subroutine dsg4set1234(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
c  case of four neutral particles
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsg4set13.f

```
      subroutine dsg4set13(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
c  case of two neutral particles (1 and 3)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsg4set23.f

```
      subroutine dsg4set23(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
c  case of two neutral particles (2 and 3)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsg4set34.f

```
      subroutine dsg4set34(kp1,kp2,kp3,kp4,rvrtx,ivrtx)
c-----------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx=rcrtx+I*ivrtx
```

```
c  case of two neutral particles (3 and 4)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc.f

```
      subroutine dsg4setc(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc12.f

```
      subroutine dsg4setc12(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  case of two neutral particles (1 and 2)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc1234.f

```
      subroutine dsg4setc1234(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  case of four neutral particles
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc13.f

```
      subroutine dsg4setc13(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  case of two neutral particles (1 and 3)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc23.f

```
      subroutine dsg4setc23(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  case of two neutral particles (2 and 3)
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dsg4setc34.f

```
      subroutine dsg4setc34(kp1,kp2,kp3,kp4,vrtx)
c---------------------------------------------------------------------
c  auxiliary subroutine to dsvertx for quartic couplings
c  set the value of the 4-particle vertex to vrtx
c  case of two neutral particles (3 and 4)
```

```
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=====================================================================
```

dshgfu.f

```
      subroutine dshgfu(ma,tanb,mq,mur,md,mtop,at,ab,mu,vh,
     &     stop1,stop2,v,mz,alpha1,alpha2,alpha3z)
c
c carena, quiros, wagner gfun -- adapted to darksusy by gondolo
c
```

dshigsct.f

```
      subroutine dshigsct(unphys,hwarning)
c---------------------------------------------------------------------
c  higgs bosons masses and mixings
c  called by dssusy.
c  needs dssfesct.
c  higloop =  0  tree-level
c             1  brignole-ellis-ridolfi-zwirner eff. pot.
c             2  drees-nojiri eff. pot.
c             3  carena-espinosa-quiros-wagner rg-impr. eff. pot.
c                    (uses subh.f.)
c             4  carena-quiros-wagner impr. eff. pot.
c                    (uses subhpole2.f)
c             5  use FeynHiggs by Heinemeyer, Hollik and Weiglein
c                requires full FeynHiggs to be installed (see below)
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995
c  modified by: joakim edsjo, edsjo@fysik.su.se, 2000-09-01
c  modified by: paolo gondolo, 1999-2000
c=====================================================================
```

dshigwid.f

```
      subroutine dshigwid
c---------------------------------------------------------------------
c   Wrapper routine to choose which Higgs width routines to use
c   Author: Joakim Edsjo, edsjo@fysik.su.se
c   Date: 2009-03-13
c=====================================================================
```

dshigwid1.f

```
      subroutine dshigwid1
c---------------------------------------------------------------------
c  needs chasct, neusct, sfesct, higsct, vertx.
c  merging of dshwidths and dshigferqcd
c  author: Piero Ullio (ullio@sissa.it) 020917
c          partly based on dshwidth by P. Gondolo and J. Edsjo
c       formulas from higgs hunters guide,
c       Djouadi, Spira and Zerwas, hep-ph/9511344
c       and Spira, hep-ph/9705337
c       if higwid=-1, we come from an MSSM SLHA file, don't redefine couplings
c       in this case (for MSSM files)
c=====================================================================
```

dshlf2.f

```
      function dshlf2(x,y)
c paolo gondolo
```

dshlf3.f

```
      function dshlf3(p2,y1,y2)
c paolo gondolo
```

dsmass.f

```
**********************************************************************
*** This function returns the mass of (for now only standard model) particles,
*** identified by their PDG code
*** (http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf)
***
*** (NB this has to overload the corresponding SM function because the internal
***  particle code for th SM Higgs is different in the MSSM)
***
***      kPDG  | particle
***     --------+----------
***         1   | u quark
***         2   | d quark
***         3   | s quark
***         4   | c quark
***         5   | b quark
***         6   | t quark
***        11   | electron
***        12   | nu_e
***        13   | muon
***        14   | nu_mu
***        15   | tau
***        16   | nu_tau
***        21   | gluon
***        22   | photon
***        23   | Z
***        24   | W
***        25   | h
***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09
**********************************************************************

      real*8 function dsmass(kPDG)
```

dsmwimp.f

```
**********************************************************************
*** This function returns the WIMP mass
***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09
**********************************************************************

      real*8 function dsmwimp()
```

dsneusct.f

```
      subroutine dsneusct
c-----------------------------------------------------------------------
c  neutralino masses and mixings. base is b-ino, w3-ino, h1-ino, h2-ino.
c  uses quartic.
c  called by susyin or mrkin.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995
c  history:
c     940528 readability improvement (pg)
c     950316 order by increasing mass (pg)
c     951110 positive mass convention (pg)
c     970211 loop corrections via switch neuloop (joakim edsjo)
c     990724 drop v1,v2 and change mass scale to max(mz,m1,m2,mu) (pg)
```

```
c     080606 use T. Hahn Diag routines (pg)
c=======================================================================
```

dsorder_flavour.f

```
************************************************************************
*** This routine orders all sfermions according to flavour. For sleptons,   ***
*** e.g. this implies the following for the internal particle code ksl_flav ***
*** (while ksl itself is not necessarily ordered):                          ***
***                                                                         ***
***       ksl_flav(1,1) = lightest selectron-like sfermion                  ***
***       ksl_flav(2,1) = lightest smuon-like sfermion                      ***
***       ksl_flav(3,1) = lightest stau-like sfermion                       ***
***       ksl_flav(1,2) = heavier selectron-like sfermion                   ***
***       ksl_flav(2,2) = heavier smuon-like sfermion                       ***
***       ksl_flav(3,2) = heavier stau-like sfermion                        ***
***                                                                         ***
*** Here, the two "most selectron-like" particles are defined as those that ***
*** mix strongest with the light- and right-handed selectron, respectively. ***
*** For squarks, the analogous applies, while for sneutrinos only the first ***
*** 3 entries in the above table exist.                                     ***
***                                                                         ***
***       output: ksnu_flav, ksl_flav, ksqu_flav, ksqd_flav                 ***
***               (stored as common block variables in dsmssm.h)            ***
***                                                                         ***
*** NB: This routine assumes that the mixing and mass common blocks         ***
***     have already been set.                                              ***
***                                                                         ***
*** date:   2016-11-15                                                      ***
*** author: torsten.bringmannl@fys.uio.no,                                  ***
*** modified: tb 2018-12-07, added widths                                   ***
*** modified: tb 2018-12-07, introduced new particle codes ksl_flav etc.    ***
***                          rather than changing any masses or widths      ***
************************************************************************

      subroutine dsorder_flavour
```

dspole.f

```
      subroutine dspole(mchi,ma,tanb,mq,mur,mdr,
     &     mtop,at,ab,mu,mh,mhp,hm,hmp,amp,sa,ca,
     &     v,mz,alpha1,alpha2,alpha3z,sint,lambda,prtlevel,ierr)
c
c carena, quiros, wagner -- adapted to darksusy by gondolo
c
```

dsqindx.f

```
      function dsqindx(kp1,kp2,kp3,kp4)
c-----------------------------------------------------------------------
c  auxiliary function to dsvertx for quartic couplings
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

dsralph3.f

```
      real*8 function dsralph3(mscale)
No header found.
```

dsrghm.f

```
      subroutine dsrghm(mchi,ma,tanb,mq,mur,md,mtop,au,ad,mu,
     &     mhp,hmp,sa,ca,tanba,v,mz,alpha1,alpha2,alpha3z,
     &     lambda,prtlevel)
c
c carena, quiros, wagner -- adapted to darksusy by gondolo
c
```

dsrmq.f

```
      real*8 function dsrmq(mscale,kpart)
No header found.
```

dssfesct.f

```
      subroutine dssfesct(unphys)
c-----------------------------------------------------------------------
c  sfermion masses and mixings.
c  options:
c    msquarks=0  : squark masses and mixing from mass matrix
c    msquarks>0  : all squark masses equal to msquarks, no mixing
c    msquarks<0  : all squark masses = max(lsp,-msquarks), no mixing
c    msleptons=0  : squark masses and mixing from mass matrix
c    msleptons>0  : all squark masses equal to msleptons, no mixing
c    msleptons<0  : all squark masses = max(lsp,-msleptons), no mixing
c  called by susyin.
c  needs neusct.
c  author: paolo gondolo 1994-1999
c     981000 correction to diagonalization of mass matrices
c     941100 addition of generation-mixing for squarks
c     950100 correction to charged slepton mass matrix
c     990715 pg options msquarks and msleptons added
c     990724 pg drop chankowski constants
c     020219 pg better reporting of unphys
c     020405 pg matrix diagonalization rewritten to handle degenerate case
c     080611 pg 6x6 mass matrices
c=======================================================================
```

dsspectrum.f

```
      subroutine dsspectrum(unphys,hwarning)
c-----------------------------------------------------------------------
c  particle spectrum and mixing matrices
c  uses sconst, sfesct, chasct, higsct, neusct
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1999
c=======================================================================
```

dsspwid.f

```
      subroutine dsspwid
c-----------------------------------------------------------------------
c  Set or calcualte the widths of sparticles
c  Author: Joakim Edsjo, edsjo@fysik.su.se
c  Date: 2008-07-02
c=======================================================================
```

dssuconst_higgs.f

```
      subroutine dssuconst_higgs
c-----------------------------------------------------------------------
```

```
c  useful constants
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo 1994-1999
c  modified: 031105 neutrino's yukawa corrected (pg)
c=====================================================================
```

dssuconst_yukawa.f

```
      subroutine dssuconst_yukawa
c---------------------------------------------------------------------
c  useful constants
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo 1994-1999
c  modified: 031105 neutrino's yukawa corrected (pg)
c  modified: 081203 uncomment below to use masses run to the
c    electroweak scale (not to 2*m_lsp like
c    dssuconst_yukawa_running).  This requires
c    extra or external code to set the values of the
c    running masses. pat scott
c    WARNING: see comment in header of dssuconst_yukawa_running.
c             (Torsten Bringmann)
c=====================================================================
```

dssuconst_yukawa_running.f

```
      subroutine dssuconst_yukawa_running
c---------------------------------------------------------------------
c  Calculate Yukawas with running masses
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo 1994-1999
c  modified: 031105 neutrino's yukawa corrected (pg)
c  modified: 2015-10-29 switched of yukawa running for quarks if full NLO
c                       result for cross section is used (torsten bringmann)
c
c This routine replaces mass(kq) in the tree-level expressions with
c dsrmq(2m*mass(lsp),kq). This is always the correct scaling with
c energy, but the overall *normalization* / renormalization condition
c will in general depend on the process.
c For neutralino-neutralino annihilation, e.g., the result must be
c multiplied by mass(kq)/dsrmq(2m*mass(kq),kq), see 1510.02473.

c=====================================================================
```

dsvertx.f

```
      subroutine dsvertx
c---------------------------------------------------------------------
c  some couplings used in DarkSUSY
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994--
c  history:
c    951110 complex vertex constants
c    970213 joakim edsjo
c    990724 paolo gondolo trilinear higgs and goldstone couplings
c=====================================================================
c
c  vertices included:
c     see individual routines dsvertx1 and dsvertx3
c
```

dsvertx1.f

```
      subroutine dsvertx1
c-----------------------------------------------------------------------
c  some couplings used in neutralino-neutralino, neutralino-chargino
c  and chargino-chargino annihilation.
c  common:
c    'dsmssm.h' - file with susy common blocks
c  called by susyin.
c  needs neusct, chasct, sfesct, higsct.
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994-1999
c  history:
c    951110 complex vertex constants
c    970213 joakim edsjo
c    990724 paolo gondolo trilinear higgs and goldstone couplings
c    020710 Joakim Edsjo, chargino-(up-squark)-(down-quark) sign-change
c    020903 Mia Schelke, Higgs-squark-squark, A-terms sign-change
c=======================================================================
c
c  vertices included:
c     z-w-w, z-h-h, w-h-a, w-h-h, h-w-w, h-z-z, z-a-h, h-h-h, h-a-a,
c     h-h-h, a-f-f, h-f-f, z-f-f, a-n-n, h-n-n, z-n-n, w-n-c, h-n-c,
c     squark-gluino-quark, sf-n-f, h-c-c, a-c-c, squark-squark-higgs,
c     w-f-f', h-f-f', gamma-w-w, gamma-h-h, z-c-c, gamma-c-c
c     gamma-f-f
c     gld-h-h, gld-gld-h, gld-n-c
c     Z-f~-f~, gamma-f~-f~, gluon-f~-f~
c
```

dsvertx3.f

```
      subroutine dsvertx3
c-----------------------------------------------------------------------
c  some couplings used in sfermion coannihilations
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c  history:
c    0110XX-020618 paolo gondolo
c    020903 Mia Schelke, Higgs-sfermion-sfermion, A-terms sign-change
c=======================================================================
c
c  vertices included:
c     Z-f~-f~, gamma-f~-f~, gluon-f~-f~, W-f~-F~, h-f~-f~
c  quartic vertices included:
c     Z-Z-h-h, W-W-h-h, Z-W-h-h, W-W-f~-f~,
c     gamma-gamma-f~-f~, Z-Z-f~-f~, gamma-Z-f~-f~, gamma-W-f~-F~,
c     gluon-gluon-f~-f~, gluon-W-f~-F~, gluon-gamma-f~-f~, gluon-Z-f~-f~,
c     h-h-f~-f~, h-goldstone-f~-f~, goldstone-goldstone-f~-f~,
c     h-h-h-h, h-h-h-goldstone, h-h-goldstone-goldstone,
c     h-goldstone-goldstone-goldstone, 4 x goldstone
c
```

dswhwarn.f

```
      subroutine dswhwarn(unit,hwarning)
c-----------------------------------------------------------------------
c  write reasons for unphys<>0 to specified unit.
c  input:
c    unit - logical unit to write on (integer)
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dswspectrum.f

```
**********************************************************************
      subroutine dswspectrum(unit)
c-----------------------------------------------------------------------
c  write out a table of the mass spectrum.
c  input:
c    unit - logical unit to write on (integer)
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dswunph.f

```
      subroutine dswunph(unit,unphys)
c-----------------------------------------------------------------------
c  write reasons for unphys<>0 to specified unit.
c  input:
c    unit - logical unit to write on (integer)
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dswvertx.f

```
**********************************************************************
      subroutine dswvertx(unit)
c-----------------------------------------------------------------------
c  write out a table of the vertices constants.
c  input:
c    unit - logical unit to write on (integer)
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

dswwidth.f

```
**********************************************************************
      subroutine dswwidth(unit)
c-----------------------------------------------------------------------
c  write out a table of Higgs decay widths
c  input:
c    unit - logical unit to write on (integer)
c  common:
c    'dsmssm.h' - file with susy common blocks
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c=======================================================================
```

g4p.f

```
      function g4p(kp1,kp2,kp3,kp4)
c-----------------------------------------------------------------------
c  function returning the 4-particle vertex
c  author: paolo gondolo (pxg26@po.cwru.edu) 2001
c=======================================================================
```

39.17 mssm/ge_cmssm: cMSSM interface (Isasugra)

39.17.1 mSUGRA (ISASUGRA) interface to DarkSUSY

If Isasugra is available, DarkSUSY can use Isasugra to generate mSUGRA models. In **src_modles/mssm/ge_cmssm**, routines are avaible to transfer the mSUGRA parameters from DarkSUSY to Isasugra, call Isasugra and then transfer back the results to DarkSUSY. The philosophy of this interface is that whenever a user uses Isasugra, we should use all the results of Isasugra also in DarkSUSY. That means that instead of calculating the mass spectrum from the low-energy parameters obtained from Isasugra, we extract the masses and mixings from Isasugra.

39.17.2 Routine headers – fortran files

dsisasugra_darksusy.f

```
      subroutine dsisasugra_darksusy(valid)
c=====================================================================
c  interface between ISASUGRA and DarkSUSY common blocks
c  author: E.A.Baltz, 2001 eabaltz@alum.mit.edu
c  modified by J. Edsjo, 2002-03-19 to set alph3
c  updated to Isajet 7.74 by J. Edsjo and E.A. Baltz, 2006-02-20
c  modified by P. Ullio 02-07-10, 02-11-21
c  checked for isasugra 7.85 by J. Edsjo, 2016-04-13
c=====================================================================
```

dsrge_isasugra.f

```
      subroutine dsrge_isasugra(unphys,valid)
c=====================================================================
c  interface to ISASUGRA (ISAJET 7.78) routines for SUSY spectra
c  author: E.A.Baltz, 2001 eabaltz@alum.mit.edu
c
c  if valid is non-zero, the model is no good
c  the valid flag is equal to the isasugra nogood flag:
c  valid  reason for model being bad
c  -----  -------------------------
c      1  TACHYONIC PARTICLES
c      2  NO EW SYMMETRY BREAKING
c      3  M(H_P)^2<0
c      4  YUKAWA>10
c      5  Z1SS NOT LSP
c      7  XT EWSB IS BAD
c      8  MHL^2<0
c      9  if in our check any Higgs mass is NaN
c  The following are not set, but can be set by uncommenting
c  the appropriate lines in dsisasugra_check.f
c  10-14  dsisasugra_check has reported a possible error in the interface
c         while checking that chargino, neutralino and sfermion mass
c         matrices are diagonlized by isasugra
c Updated to ISAJET 7.74 by J. Edsjo and E.A. Baltz, 2006-02-20
c Updated to ISAJET 7.78 by J. Edsjo, 2008-06-05.
c Updated to ISAJET 7.79 by J. Edsjo, 2010-03-08
c Updated to ISAJET 7.81 by P. Gondolo, 2011-09-04
c Updated to ISAJET 7.85 by J. Edsjo, 2016-04-13
c=====================================================================
```

39.18 mssm/ge_slha: SUSY Les Houches Accord interface

39.18.1 SUSY Les Houches Accord

DarkSUSY includes routines to read and write SUSY Les Houches Accord [163, 164] files (SLHA files). This is done with the help of SLHALIB by T. Hahn [218].

DarkSUSY will write SLHA2 files and expect to get SLHA2 as well. The implementation right now takes a middle path between dumping everything or just a minimal set of inputs to the SLHA2 file. This choice was made to make most other SLHA2-aware programs able to exchange SLHA2 files with DarkSUSY.

However, we have not made a careful testing with lots of other codes, so if you try this out, please let us know if there are some things that don't work or could work better.

39.18.2 Routine headers – fortran files

dsmssmzero.f

```
      subroutine dsmssmzero
c--------------------------------------------------------------------
c
c     Routine to zero all EW SUSY parameters in DarkSUSY. This is
c     done to make sure we don't inherit old values from previous
c     models when reading e.g. SLHA files.
c     Author: Joakim Edsjo, edsjo@fysik.su.se
c     Date: 2008-07-01
c--------------------------------------------------------------------
```

dsfromslha.F

```
**********************************************************************
*** subroutine dsfromSLHA transfers SLHA data in the array slhadata
*** to DarkSUSY. The SLHA things are read with with the help of
*** SLHALIB by Thomas Hahn.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01
*** FIXME : Better error handling
***         More options on how to treat redundant information, e.g.
***         should Yukawas from SLHA file be used instead of letting
***         DarkSUSY run them (as is currently done).
*** Modified: 2016-11-11 Joakim Edsjo, fixed lsp setting
***           2018-12-07 Torsten Bringmann, added (s)particle widths
**********************************************************************
      subroutine dsfromSLHA
```

dsgive_model_SLHA.F

```
**********************************************************************
*** subroutine dsSLHAread reads SUSY Les Houches Accord files
*** with the help of SLHALIB by Thomas Hahn.
*** Input: file - filename
***        printlevel - 0 = print only errors
***                     1 = print errors and warnings
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01
*** Torsten Bringmann: streamlined to use with dsmodelsetup (2014-05-09)
**********************************************************************
      subroutine dsgive_model_SLHA(file,printlevel)
```

dssetfromslha.F

```
**********************************************************
```

```
*** dssetfromslha sets one parameter from the SLHA common
*** block slhadata, if the value is set. If not, it will take
*** the DarkSUSY default
*** Input: slhaitem - array item in SLHADATA
***        option - if 0, disregard missing item in the SLHA file
***                 if 1, print a warning that 'text' is missing
***                 if 2, print an error that 'text' is missing
***        text - explanatory text for what variable we are working with
*** Output: dsparam - changed to slhaitem if valid
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01
************************************************************

      subroutine dssetfromslha(slhaitem,dsparam,option,text)
```

dssetfromslha2.F

```
************************************************************
*** dssetfromslha sets one parameter from the SLHA common
*** block slhadata, checking two different entries (in the
*** order given). If neither of them is present,
*** it will take the DarkSUSY defualt
*** Input: slhaitem1 - first choice array item in SLHADATA
***        slhaitem2 - second choice array item in SLHADATA
***        option - if 0, disregard missing item in the SLHA file
***                 if 1, print a warning that 'text' is missing
***                 if 2, print an error that 'text' is missing
***        text - explanatory text for what variable we are working with
*** Output: dsparam - changed to slhaitem if valid
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01
************************************************************

      subroutine dssetfromslha2(slhaitem1,slhaitem2,dsparam,option,text)
```

dssetfromslhac.F

```
************************************************************
*** dssetfromslha sets one complex parameter from the SLHA commong
*** block slhadata, if the value is set. If not, it will take
*** the DarkSUSY defualt
*** Input: slhaitemr - real part of array item in SLHADATA
***        slhaitemi - imaginary part of array item in SLHADATA
***        option - if 0, disregard missing item in the SLHA file
***                 if 1, print a warning that 'text' is missing
***                 if 2, print an error that 'text' is missing
***        text - explanatory text for what variable we are working with
*** Output: dsparam - changed to slhaitem if valid
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-07-01
************************************************************

      subroutine dssetfromslhac(slhaitem,dsparam,
     &  option,text)
```

dsSLHAwrite.F

```
**********************************************************************
*** subroutine dsSLHAwrite writes SUSY Les Houches Accord files
*** with the help of SLHALIB by Thomas Hahn.
*** Input: file = filename of SLHA2 file to write to
***        opt = 1 - an SLHA2 file with full 6x6 squark and slepton
***                  mixings and 3x3 trilinear couplings is written.
***                  For minimal flavour violation
```

```
***                  this is a bit of an overkill, but more consistent
***                  with how things are stored internally in DarkSUSY
***              2 - a minimal flavour violation SLHA2 file is written
***                  with only third generation mixings and trilinear
***                  couplings. This option will in principle be an
***                  approximation to the internal setup in DarkSUSY,
***                  but most likely good enough for most scenarios.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-06-24
**********************************************************************
      subroutine dsSLHAwrite(file,opt)
```

dstoslha.F

```
**********************************************************************
*** subroutine dstoSLHA transfers DarkSUSY data to the SUSY Les
*** Houches Accord format (into the array slhadata)
*** with the help of SLHALIB by Thomas Hahn.
*** Input: opt = 1 - an SLHA2 file with full 6x6 squark and slepton
***                  mixings and 3x3 trilinear couplings is written.
***                  For minimal flavour violation
***                  this is a bit of an overkill, but more consistent
***                  with how things are stored internally in DarkSUSY
***              2 - a minimal flavour violation SLHA2 file is written
***                  with only third generation mixings and trilinear
***                  couplings. This option will in principle be an
***                  approximation to the internal setup in DarkSUSY,
***                  but most likely good enough for most scenarios.
***              3 - an mSUGRA input file (i.e. only with the SM and
***                  mSUGRA input parameters). This option could be
***                  of interest when the RGEs should be solved with
***                  an external program.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: 2008-06-24
*** mod:  2019-04-26 torsten.bringmann@fys.uio.no (added block DECAY)
**********************************************************************
      subroutine dstoSLHA(opt)
```

39.19 mssm/ini: Initialization routines

39.19.1 Initialization routines

Before DarkSUSY is used for some calculations, it needs to be initialized. This is done with a call to **dsinit**, which in turn calls the particle-specific initialization routine dsinit_module that resides in each module. This means that the call to **dsinit** should be the first call in any program using DarkSUSY. Any calls the user makes to other routines, either to calculte things or select a different model (e.g. a different halo model) should come after the call to **dsinit**.

In the mssm module, the routine dsinit_module that all standard parameters are defined, such as standard model parameters and particle codes. It also sets more particle-specific defaults for DarkSUSY routines where this is available, via calls to routines such as dsddset_mssm and dsanset_mssm.

39.19.2 Routine headers – fortran files

dsgive_model.f

```
      subroutine dsgive_model(amu,am2,ama,atanbe,amsq,atm,abm)
c--------------------------------------------------------------------
c
```

```
c     To specify the supersymmetric parameters of a model.
c     Inputs:
c        amu - mu parameter (GeV)
c        am2 - M2 parameter (GeV)
c        ama - Mass of the CP-odd Higgs boson A (or H3)
c        atanbe - ratio of Higgs vacuum expecation values, tan(beta)
c        amsq - common sfermion mass scale, M_sq_tilde (GeV)
c        atm - trilinear term in units of amsq, top sector
c        abm - trilinear term in units of amsq, bottom sector
c     Outputs:
c        The common blocks are set corresponding to the values above
c     Author: Paolo Gondolo, gondolo@mppmu.mpg.de
c     Date: 2000
c     Modified: Joakim Edsjo, edsjo@fysik.su.se
c        2001-02-13 - setting of idtag taken away
c-----------------------------------------------------------------------
```

dsgive_model13.f

```
      subroutine dsgive_model13(amu,am1,am2,ama,atanbe,
     &  amse,amsmu,amstau,amsq,
     &  atm,abm,ataum,aotherm)
c-----------------------------------------------------------------------
c
c     To specify the supersymmetric parameters of a model.
c     Inputs:
c        amu - mu parameter (GeV)
c        am1 - M1 parameter (GeV)
c        am2 - M2 parameter (GeV)
c        ama - Mass of the CP-odd Higgs boson A (or H3)
c        atanbe - ratio of Higgs vacuum expecation values, tan(beta)
c        amse - selectron mass scale (GeV)
c        amsmu - smuon mass scale (GeV)
c        amstau - stau mass scale (GeV)
c        amsq - common squark mass scale, M_sq_tilde (GeV)
c        atm - trilinear term in units of amsq, top sector
c        abm - trilinear term in units of amsq, bottom sector
c        ataum -  trilinear term in units of amstau, stau sector
c        aotherm - trilinear term for remaining squarks and sleptons
c                  in units of amsq, amse and amsmu respectively
c     Outputs:
c        The common blocks are set corresponding to the values above
c     Author: Joakim Edsjo, edsjo@fysik.su.se
c     Date:   2007-12-17
c-----------------------------------------------------------------------
```

dsgive_model15.f

```
      subroutine dsgive_model15(amu,am1,am2,ama,atanbe,
     &  amse,amsmu,amstau,amsq1,amsq2,amsq3,
     &  atm,abm,ataum,aotherm)
c-----------------------------------------------------------------------
c
c     To specify the supersymmetric parameters of a model.
c     Inputs:
c        amu - mu parameter (GeV)
c        am1 - M1 parameter (GeV)
c        am2 - M2 parameter (GeV)
c        ama - Mass of the CP-odd Higgs boson A (or H3)
c        atanbe - ratio of Higgs vacuum expecation values, tan(beta)
c        amse - selectron mass scale (GeV)
```

```
c        amsmu - smuon mass scale (GeV)
c        amstau - stau mass scale (GeV)
c        amsq1 - squark mass scale, 1st generation (GeV)
c        amsq2 - squark mass scale, 2nd generation (GeV)
c        amsq3 - squark mass scale, 3rd generation (GeV)
c        atm - trilinear term in units of amsq, top sector
c        abm - trilinear term in units of amsq, bottom sector
c        ataum -  trilinear term in units of amstau, stau sector
c        aotherm - trilinear term for remaining squarks and sleptons
c                  in units of amsq, amse and amsmu respectively
c     Outputs:
c        The common blocks are set corresponding to the values above
c     Author: Joakim Edsjo, edsjo@fysik.su.se
c     Date:   2007-12-17
c-----------------------------------------------------------------------
```

dsgive_model25.f

```
      subroutine dsgive_model25(am1,am2,am3,amu,ama,atanbe,
     &  amsqL1,amsql2,amsql3,amsqRu,amsqRc,amsqRt,amsqRd,
     &  amsqRs,amsqRb,amslL1,amslL2,amslL3,amslRe,
     &  amslRmu,amslRtau,atm,abm,ataum,aemum)
c-----------------------------------------------------------------------
c
c     To specify the supersymmetric parameters of a model.
c     Inputs:
c        Gaugino Sector
c        am1 - M1 parameter (GeV)
c        am2 - M2 parameter (GeV)
c        am3 - M3 parameter (GeV)
c
c        Higgs Sector
c        amu - mu parameter (GeV)
c        ama - Mass of the CP-odd Higgs boson A (or H3)
c        atanbe - ratio of Higgs vacuum expecation values, tan(beta)
c
c        Sfermion sector
c        amsqL1 - left type squark mass scale, 1st generation
c        amsqL2 - left type squark mass scale, 2nd generation
c        amsqL3 - left type squark mass scale, 3rd generation
c        amsqRu - right type sup squark mass scale
c        amsqRc - right type scharm squark mass scale
c        amsqRt - right type stop squark mass scale
c        amsqRd - right type sdown squark mass scale
c        amsqRs - right type sstrange squark mass scale
c        amsqRb - right type sbottom squark mass scale
c        amslL1 - left type slepton mass scale, 1st generation
c        amslL2 - left type slepton mass scale, 2nd generation
c        amslL3 - left type slepton mass scale, 3rd generation
c        amslRe - right type selectron mass scale
c        amslRmu - right type smuon mass scale
c        amslRtau - right type stau mass scale
c
c        Trilinear Couplings
c        atm - trilinear term, top sector, (in units of ?)
c        abm - trilinear term, bottom sector, (in units of ?)
c        ataum - trilinear term, stau sector
c        aemum - trilinear term, electron and muon sector
c
c     Outputs:
c        The common blocks are set corresponding to the values above
c
c     Author: Joakim Edsjo, edsjo@fysik.su.se
```

```
c     Date:   2007-12-17
c     Modified by Hamish Silverwood, moving from 15 to 25 parameters
c     Date:   2011-06-30
c-----------------------------------------------------------------------
```

dsgive_model_isasugra.f

```
      subroutine dsgive_model_isasugra(m0,mhf,a0,sgnmu,tgbeta)
c-----------------------------------------------------------------------
c
c     To specify the supersymmetric parameters of a model.
c     Inputs:
c        m0 - m0 parameter (GeV)
c        mhf - m_{1/2} parameter (GeV)
c        a0 - trilinear term (GeV)
c        sgnmu - sign of mu (+1.0d0 or -1.0d0)
c        tgbeta - ratio of Higgs vacuum expecation values, tan(beta)
c     Outputs:
c        The common blocks are set corresponding to the values above
c     Author: Joakim Edsjo, edsjo@fysik.su.se
c        2002-03-12
c-----------------------------------------------------------------------
```

dsmodelsetup.f

```
      subroutine dsmodelsetup(ierr,iwarn)
***_____________________________________________________________________
***  set up global variables for the supersymmetric model routines.
***  uses sconst, sfesct, chasct, higsct,
***    neusct, vertx, hwidths.
***                                                                     ***
***  type : interface                                                   ***
***                                                                     ***
***  desc : Sets up a new particle physics model                        ***
***                                                                     ***
***  output:                                                            ***
***    ierr  -  0 no errors                                             ***
***          - <0 theoretically inconsistent model parameters           ***
***          -  1 the neutralino is not the LSP                         ***
***    iwarn -  0 no warnings                                           ***
***           !=0 breakdown of approximations used in radiative corrections ***
***               in the Higgs sector                                   ***
***                                                                     ***
***  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994,1995
***  Torsten Bringmann: unified with SLHA, earlier dssusy(_isasugra)
***    (08/05/14)       and dsmodelsetup_isasugra
***    (24/05/19)       moved ivfam here to gurantee consistent flavour ordering
c=======================================================================
```

dsinit_module.F

```
*************************************************************************
***  This is the initialization subroutine for the "MSSM" module.       ***
***                                                                     ***
***  type : interface                                                   ***
***                                                                     ***
***  Last major edit by: Torsten Bringmann (torsten.bringmann@fys.uio.no)   ***
***  Date: 11/2016                                                      ***
*************************************************************************
      subroutine dsinit_module
```

39.20 mssm/kd: Kinetic decoupling

39.20.1 Routine headers – fortran files

dskdm2.f

```
************************************************************************
*** dskdm2 returns the full scattering amplitude for NEUTRALINO DM at zero
*** momentum transfer squared, SUMMED over both initial and final spin and
*** other internal states, and then divided by the DM internal degrees of
*** freedom.
***
***  type : interface
***  desc : Full scattering amplitude squared, at zero momentum transfer
***
***  input: omega -- CMS MOMENTUM of scattering partner
***  output: omega -- CMS ENERGY of scattering partner
***
***  input:   SMtype - SM scattering partners
***             "    = 7,8,9,10,11,12 - d,u,s,c,b,t quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2013-06-12 (made model-dependence explicit)
***          2016-12-21 (changed spin-average prescription)
***          2018-05-14 (moved momentum->energy conversion to src_models/)
************************************************************************

      real*8 function dskdm2(omega,SMtype)
```

dskdm2simp.f

```
************************************************************************
*** dskdm2simp returns the scattering amplitude for NEUTRALINO DM squared
*** at zero momentum transfer, SUMMED over both initial and final spin and
*** other internal states, and then divided by the DM internal degrees of
*** freedom. This is only valid in the limit of relativistic scattering
*** partners with small energies omega, where we can expand as
***
***    |M|**2 = cn*(omega/m0)**n + O( (omega/m0)**(n+1) )
***
***  type : INTERFACE
***  desc : Scattering amplitude squared, expanded in powers of energy
***
***  input: SMtype   - SM scattering partners:
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten bringmann (troms@physto.se), 2010-01-23
*** updates: 2013-06-12 (made model-dependence explicit)
***          2016-12-21 (changed spin-average prescription)
************************************************************************

      subroutine dskdm2simp(SMtype,cn,n)
```

dskdparticles.f

```
************************************************************************
*** Prepares integration of Boltzmann equation and has to be called    ***
*** before dskdboltz (called by dskdtkd)                              ***
***                                                                   ***
```

```
***  type : interface                                                      ***
***                                                                        ***
*** author: torsten bringmann (troms@physto.se), 2010-01-23               ***
*** updates: 2013-06-11 (made model-dependence explicit)                   ***
******************************************************************************

      subroutine dskdparticles()
```

39.21 mssm/rd: Relic density

39.21.1 Relic density of neutralinos

In the folder **mssm/rd**, we provide one of the two interface functions that are necessary for the core library to solve the Boltzmann equation for any cold dark matter particle and hence calculate its relic density:[‡] dsrdparticles determines which particles can coannihilate (based on their mass differences) and puts these particles into a common block for the annihilation rate routines (**dsanwx**); it also checks where we have resonances and thresholds and adds these to an array, which is passed to the relic density routines. The relic density routines then use this knowledge to make sure the tabulation of the cross section and the integrations are performed correctly at these difficult points.

For convenience, we include one further routine in this folder: dsrdwrate writes the invariant rate to a specified unit. This is mostly useful for debugging purposes.

39.21.2 Internal degrees of freedom

In Section 27.1.2, we have reviewed the standard way of calculating the relic density. Here, we add some comments which are specific to the implementation of the effective invariant rate in the mssm module

If we look at Eqs. (27.34) and (27.39) we see that we have a freedom on how to treat particles degenerate in mass, e.g. a chargino can be treated either

a) as two separate species χ_i^+ and χ_i^-, each with internal degrees of freedom $g_{\chi^+} = g_{\chi^-} = 2$, or,

b) as a single species $\chi_i^\pm$ with $g_{\chi_i^\pm} = 4$ internal degrees of freedom.

Of course the two views are equivalent, we just have to be careful including the g_i's consistently whichever view we take. In a), we have the advantage that all the W_{ij} that enter into Eq. (27.34) enter as they are, i.e. without any correction factors for the degrees of freedom. On the other hand we get many terms in the sum that are identical and we need some book-keeping machinery to avoid calculating identical terms more than once. On the other hand, with option b), the sum over W_{ij} in Eq. (27.34) is much simpler only containing terms that are not identical (except for the trivial identity $W_{ij} = W_{ji}$ which is easily taken care of). However, the individual W_{ij} will be some linear combinations of the more basic W_{ij} entering in option a), where the coefficients have to be calculated for each specific type of initial condition.

Below we will perform this calculation to show how the W_{ij} look like in option b) for different initial states. We will use a prime on the W_{ij} when they refer to these combined states to indicate the difference.

39.21.2.1 Neutralino-chargino annihilation

The starting point is Eq. (27.34) which we will use to define the W_{ij} in option b) such that $W_{\rm eff}$ is the same as in option a). Eq. (27.39) is then guaranteed to be the same in both cases since the sum in the denominator is linear in g_i.

[‡]The other, dsanwx, returns the effective invariant rate and resides in mssm/an/.

Now consider annihilation between χ_i^0 and χ_c^+ or χ_c^-. The corresponding terms in Eq. (27.34) does for option a) read

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \frac{p_{ic}}{p_{11}} \frac{2 \cdot 2}{2^2} \left[W_{\chi_i^0 \chi_c^+} + W_{\chi_i^0 \chi_c^-} + \underbrace{W_{\chi_c^+ \chi_i^0}}_{W_{\chi_i^0 \chi_c^+}} + \underbrace{W_{\chi_c^- \chi_i^0}}_{W_{\chi_i^0 \chi_c^-}} \right] \\ &= 2 \frac{p_{ic}}{p_{11}} \left[W_{\chi_i^0 \chi_c^+} + \underbrace{W_{\chi_i^0 \chi_c^-}}_{W_{\chi_i^0 \chi_c^+}} \right] = 4 \frac{p_{ic}}{p_{11}} W_{\chi_i^0 \chi_c^+} \end{aligned} \tag{39.60}$$

For option b), we instead get

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \frac{p_{ic}}{p_{11}} \frac{2 \cdot 4}{2^2} \left[W'_{\chi_i^0 \chi_c^\pm} + \underbrace{W'_{\chi_c^\pm \chi_i^0}}_{W'_{\chi_i^0 \chi_c^\pm}} \right] = 4 \frac{p_{ic}}{p_{11}} W'_{\chi_i^0 \chi_c^\pm} \tag{39.61}$$

Comparing Eq. (39.61) and Eq. (39.60) we see that they are indentical if we make the identification

$$W'_{\chi_i^0 \chi_c^\pm} \equiv W_{\chi_i^0 \chi_c^+} \tag{39.62}$$

39.21.2.2 Chargino-chargino annihilation

First consider the case where we include the terms in the sum for which we have annihilation between χ_c^+ or χ_c^- and χ_d^+ or χ_d^- with $c \neq d$.

In option a), the corresponding terms in Eq. (27.34) reads

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} \\ &= \frac{p_{cd}}{p_{11}} \frac{2 \cdot 2}{2^2} \Big[W_{\chi_c^+ \chi_d^+} + W_{\chi_c^+ \chi_d^-} + W_{\chi_c^- \chi_d^+} + W_{\chi_c^- \chi_d^-} \\ &\quad + \underbrace{W_{\chi_d^+ \chi_c^+}}_{W_{\chi_c^+ \chi_d^+}} + \underbrace{W_{\chi_d^+ \chi_c^-}}_{W_{\chi_c^- \chi_d^+}} + \underbrace{W_{\chi_d^- \chi_c^+}}_{W_{\chi_c^+ \chi_d^-}} + \underbrace{W_{\chi_d^- \chi_c^-}}_{W_{\chi_c^- \chi_d^-}} \Big] \\ &= 2 \frac{p_{cd}}{p_{11}} \left[W_{\chi_c^+ \chi_d^+} + W_{\chi_c^+ \chi_d^-} + \underbrace{W_{\chi_c^- \chi_d^+}}_{W_{\chi_c^+ \chi_d^-}} + \underbrace{W_{\chi_c^- \chi_d^-}}_{W_{\chi_c^+ \chi_d^+}} \right] \\ &= 4 \frac{p_{cd}}{p_{11}} \left[W_{\chi_c^+ \chi_d^+} + W_{\chi_c^+ \chi_d^-} \right] \end{aligned} \tag{39.63}$$

In option b), the corresponding terms would instead read

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} = \frac{p_{cd}}{p_{11}} \frac{4 \cdot 4}{2^2} \left[W'_{\chi_c^\pm \chi_d^\pm} + \underbrace{W'_{\chi_d^\pm \chi_c^\pm}}_{W'_{\chi_c^\pm \chi_d^\pm}} \right] = 8 \frac{p_{cd}}{p_{11}} W'_{\chi_c^\pm \chi_d^\pm} \tag{39.64}$$

Comparing Eq. (39.63) and Eq. (39.64) we see that they are identical if we make the following identifcation

$$W'_{\chi_c^\pm \chi_d^\pm} \equiv \frac{1}{2} \left[W_{\chi_c^+ \chi_d^+} + W_{\chi_c^+ \chi_d^-} \right] \tag{39.65}$$

For clarity, let's also consider the case where $c = d$. In option a), the terms in W_{eff} are

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \frac{p_{cc}}{p_{11}} \frac{2 \cdot 2}{2^2} \left[W_{\chi_c^+ \chi_c^+} + W_{\chi_c^+ \chi_c^-} + \underbrace{W_{\chi_c^- \chi_c^+}}_{W_{\chi_c^+ \chi_c^-}} + \underbrace{W_{\chi_c^- \chi_c^-}}_{W_{\chi_c^+ \chi_c^+}} \right] \\ &= 2 \frac{p_{cc}}{p_{11}} \left[W_{\chi_c^+ \chi_c^+} + W_{\chi_c^+ \chi_c^-} \right] \end{aligned} \tag{39.66}$$

In option b), the corresponding term would instead read

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} = \frac{p_{cc}}{p_{11}} \frac{4 \cdot 4}{2^2} W'_{\chi_c^\pm \chi_c^\pm} + = 4 \frac{p_{cc}}{p_{11}} W'_{\chi_c^\pm \chi_c^\pm} \tag{39.67}$$

Comparing Eq. (39.66) and Eq. (39.67) we see that they are identical if we make the following identifcation

$$W'_{\chi_c^\pm \chi_c^\pm} \equiv \frac{1}{2} \left[W_{\chi_c^+ \chi_c^+} + W_{\chi_c^+ \chi_c^-} \right] \tag{39.68}$$

i.e. the same identification as in the case $c \neq d$.

39.21.2.3 Neutralino-sfermion annihilation

For each sfermion we have in total four different states, $\tilde{f}_1$, $\tilde{f}_2$, $\tilde{f}_1^*$ and $\tilde{f}_2^*$. Of these, the $\tilde{f}_1$ and $\tilde{f}_2$ in general have different masses and have to be treated separately. Considering only one mass eigenstate $\tilde{f}_k$, option a) then means that we treat $\tilde{f}_k$ and $\tilde{f}_k^*$ as two separate species with $g_i = 1$ degree of freedom each, whereas option b) means that we treat them as one species $\tilde{f}'_k$ with $g_i = 2$ degrees of freedom. As before, the prime indicates that we mean both the particle and the antiparticle state.

Note, that for squarks we also have the number of colours $N_c = 3$ to take into account. In option a) we should choose to treat even colour state differently, i.e. $g_i = 1$, whereas $g_i = 6$ in case b). The expressions would be the same as above except that both the expression in a) and b) would be multiplied by the colour factor $N_c = 3$. The expression relating case a) and case b) is thus unaffected by this colour factor. Note however, that in option b) we take the average over the squark colours (or in this case calculate it only for one colour. See sections 39.21.2.6 and 39.21.2.7 below for more details.

For option a), Eq. (27.34) then reads

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \frac{p_{ik}}{p_{11}} \frac{2 \cdot 1}{2^2} \left[W_{\chi_i^0 \tilde{f}_k} + W_{\chi_i^0 \tilde{f}_k^*} + \underbrace{W_{\tilde{f}_k \chi_i^0}}_{W_{\chi_i^0 \tilde{f}_k}} + \underbrace{W_{\tilde{f}_k^* \chi_i^0}}_{W_{\chi_i^0 \tilde{f}_k^*}} \right] \\ &= \frac{p_{ik}}{p_{11}} \left[W_{\chi_i^0 \tilde{f}_k} + \underbrace{W_{\chi_i^0 \tilde{f}_k^*}}_{W_{\chi_i^0 \tilde{f}_k}} \right] = 2 \frac{p_{ik}}{p_{11}} W_{\chi_i^0 \tilde{f}_k} \end{aligned} \tag{39.69}$$

whereas for option b), Eq. (27.34) reads

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} = \frac{p_{ik}}{p_{11}} \frac{2 \cdot 2}{2^2} \left[W'_{\chi_i^0 \tilde{f}'_k} + \underbrace{W'_{\tilde{f}'_k \chi_i^0}}_{W'_{\chi_i^0 \tilde{f}'_k}} \right] = 2 \frac{p_{ik}}{p_{11}} W'_{\chi_i^0 \tilde{f}'_k} \tag{39.70}$$

Comparing Eq. (39.70) and Eq. (39.69) we see that they are indentical if we make the identification

$$W'_{\chi_i^0 \tilde{f}'_k} \equiv W_{\chi_i^0 \tilde{f}_k} \tag{39.71}$$

For clarity, for squarks the corresponding expression would be

$$W'_{\chi_i^0 \tilde{q'}_k} \equiv \frac{1}{3} \sum_{a=1}^{3} W_{\chi_i^0 \tilde{q}_k^a} \tag{39.72}$$

where a is a colour index.

39.21.2.4 Chargino-sfermion annihilation

In option a) the chargino has $g_i = 2$ and the sfermion has $g_i = 1$ degrees of freedom, whereas in option b), the chargino has $g_i = 4$ and the sfermion has $g_i = 2$ degrees of freedom

For option a), Eq. (27.34) then reads

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} \\ &= \frac{p_{ck}}{p_{11}} \frac{2 \cdot 1}{2^2} \Big[W_{\chi_c^+ \tilde{f}_k} + W_{\chi_c^+ \tilde{f}_k^*} + W_{\chi_c^- \tilde{f}_k} + W_{\chi_c^- \tilde{f}_k^*} \\ &\quad + \underbrace{W_{\tilde{f}_k \chi_c^+}}_{W_{\chi_c^+ \tilde{f}_k}} + \underbrace{W_{\tilde{f}_k^* \chi_c^+}}_{W_{\chi_c^+ \tilde{f}_k^*}} + \underbrace{W_{\tilde{f}_k \chi_c^-}}_{W_{\chi_c^- \tilde{f}_k}} + \underbrace{W_{\tilde{f}_k^* \chi_c^-}}_{W_{\chi_c^- \tilde{f}_k^*}} \Big] \\ &= \frac{p_{ck}}{p_{11}} \Big[W_{\chi_c^+ \tilde{f}_k} + W_{\chi_c^+ \tilde{f}_k^*} + \underbrace{W_{\chi_c^- \tilde{f}_k}}_{W_{\chi_c^+ \tilde{f}_k^*}} + \underbrace{W_{\chi_c^- \tilde{f}_k^*}}_{W_{\chi_c^+ \tilde{f}_k}} \Big] = 2 \frac{p_{ck}}{p_{11}} \Big[W_{\chi_c^+ \tilde{f}_k} + W_{\chi_c^+ \tilde{f}_k^*} \Big] \end{aligned} \tag{39.73}$$

In option b), Eq. (27.34) reads

$$W_{\text{eff}} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} = \frac{p_{ck}}{p_{11}} \frac{4 \cdot 2}{2^2} \Big[W'_{\chi_c^\pm \tilde{f'}_k} + \underbrace{W'_{\tilde{f'}_k \chi_c^\pm}}_{W'_{\chi_c^\pm \tilde{f'}_k}} \Big] = 4 \frac{p_{ck}}{p_{11}} W'_{\chi_c^\pm \tilde{f'}_k} \tag{39.74}$$

Comparing Eq. (39.74) and Eq. (39.73) we see that they are indentical if we make the identification

$$W'_{\chi_c^\pm \tilde{f'}_k} \equiv \frac{1}{2} \Big[W_{\chi_c^+ \tilde{f}_k} + W_{\chi_c^+ \tilde{f}_k^*} \Big] \tag{39.75}$$

For clarity, for squarks the corresponding expression would be

$$W'_{\chi_c^\pm \tilde{q'}_k} \equiv \frac{1}{2} \frac{1}{3} \sum_{a=1}^{3} \Big[W_{\chi_c^+ \tilde{q}_k^a} + W_{\chi_c^+ \tilde{q}_k^{a*}} \Big] \tag{39.76}$$

where a is a colour index.

39.21.2.5 Sfermion-sfermion annihilation

First consider the case where we have annihilation between sfmerions of different types, i.e. annihilation between $\tilde{f}_k$ or $\tilde{f}_k^*$ and $\tilde{f}_l$ or $\tilde{f}_l^*$.

For option a), Eq. (27.34) then reads

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} \\ &= \frac{p_{kl}}{p_{11}} \frac{1 \cdot 1}{2^2} \Big[W_{\tilde{f}_k \tilde{f}_l} + W_{\tilde{f}_k \tilde{f}_l^*} + W_{\tilde{f}_k^* \tilde{f}_l} + W_{\tilde{f}_k^* \tilde{f}_l^*} \\ &\quad + \underbrace{W_{\tilde{f}_l \tilde{f}_k}}_{W_{\tilde{f}_k \tilde{f}_l}} + \underbrace{W_{\tilde{f}_l \tilde{f}_k^*}}_{W_{\tilde{f}_k^* \tilde{f}_l}} + \underbrace{W_{\tilde{f}_l^* \tilde{f}_k}}_{W_{\tilde{f}_k \tilde{f}_l^*}} + \underbrace{W_{\tilde{f}_l^* \tilde{f}_k^*}}_{W_{\tilde{f}_k^* \tilde{f}_l^*}} \Big] \\ &= \frac{1}{2} \frac{p_{kl}}{p_{11}} \Big[W_{\tilde{f}_k \tilde{f}_l} + W_{\tilde{f}_k \tilde{f}_l^*} + \underbrace{W_{\tilde{f}_k^* \tilde{f}_l}}_{W_{\tilde{f}_k \tilde{f}_l^*}} + \underbrace{W_{\tilde{f}_k^* \tilde{f}_l^*}}_{W_{\tilde{f}_k \tilde{f}_l}} \Big] = \frac{p_{kl}}{p_{11}} \Big[W_{\tilde{f}_k \tilde{f}_l} + W_{\tilde{f}_k \tilde{f}_l^*} \Big] \end{aligned} \tag{39.77}$$

In option b) we would get

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} \\ &= \frac{p_{kl}}{p_{11}} \frac{2 \cdot 2}{2^2} \Big[W'_{\tilde{f'}_k \tilde{f'}_l} + \underbrace{W'_{\tilde{f'}_l \tilde{f'}_k}}_{W'_{\tilde{f'}_k \tilde{f'}_l}} \Big] = 2 \frac{p_{kl}}{p_{11}} \Big[W'_{\tilde{f'}_k \tilde{f'}_l} \Big] \end{aligned} \tag{39.78}$$

Comparing Eq. (39.78) and Eq. (39.77) we see that they are indentical if we make the identification

$$W'_{\tilde{f'}_k \tilde{f'}_l} \equiv \frac{1}{2} \Big[W_{\tilde{f}_k \tilde{f}_l} + W_{\tilde{f}_k \tilde{f}_l^*} \Big] \tag{39.79}$$

It is easy to show that this relation holds true even if $k = l$.

39.21.2.6 Squark-squark annihilation

Even though we treated sfermion-sfermion annihilation in the previous subsection, squarks have colour which can complicate things, so let's for clarity consider squarks separately.

Let's denote the squarks $\tilde{q}_k^a$ where a is now a colour index. In option a) we will let each colour be a seprate species, which means that $g_i = 1$ in this case. In option b) we will instead have $g_i = 6$.

In option a) we would have

$$\begin{aligned} W_{\text{eff}} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} \\ &= \frac{p_{kl}}{p_{11}} \frac{1 \cdot 1}{2^2} \sum_{a,b=1}^{3} \Big[W_{\tilde{q}_k^a \tilde{q}_l^b} + W_{\tilde{q}_k^a \tilde{q}_l^{b*}} + W_{\tilde{q}_k^{a*} \tilde{q}_l^b} + W_{\tilde{q}_k^{a*} \tilde{q}_l^{b*}} \\ &\quad + \underbrace{W_{\tilde{q}_l^a \tilde{q}_k^b}}_{W_{\tilde{q}_k^a \tilde{q}_l^b}} + \underbrace{W_{\tilde{q}_l^a \tilde{q}_k^{b*}}}_{W_{\tilde{q}_k^{a*} \tilde{q}_l^b}} + \underbrace{W_{\tilde{q}_l^{a*} \tilde{q}_k^b}}_{W_{\tilde{q}_k^a \tilde{q}_l^{b*}}} + \underbrace{W_{\tilde{q}_l^{a*} \tilde{q}_k^{b*}}}_{W_{\tilde{q}_k^{a*} \tilde{q}_l^{b*}}} \Big] \\ &= \frac{1}{2} \frac{p_{kl}}{p_{11}} \sum_{a,b=1}^{3} \Big[W_{\tilde{q}_k^a \tilde{f}_l^b} + W_{\tilde{q}_k^a \tilde{f}_l^{b*}} + \underbrace{W_{\tilde{q}_k^{a*} \tilde{q}_l^b}}_{W_{\tilde{q}_k^a \tilde{q}_l^{b*}}} + \underbrace{W_{\tilde{q}_k^{a*} \tilde{q}_l^{b*}}}_{W_{\tilde{q}_k^a \tilde{q}_l^b}} \Big] = \frac{p_{kl}}{p_{11}} \sum_{a,b=1}^{3} \Big[W_{\tilde{q}_k^a \tilde{q}_l^b} + W_{\tilde{q}_k^a \tilde{q}_l^{b*}} \end{aligned} \tag{39.80}$$

In option b) we would get

$$\begin{aligned} W_{\rm eff} &= \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W'_{ij} \\ &= \frac{p_{kl}}{p_{11}} \frac{6\cdot 6}{2^2} \left[W'_{\tilde{q'}_k \tilde{q'}_l} + \underbrace{W'_{\tilde{q'}_l \tilde{q'}_k}}_{W'_{\tilde{q'}_k \tilde{q'}_l}} \right] = 18 \frac{p_{kl}}{p_{11}} \left[W'_{\tilde{q'}_k \tilde{q'}_l} \right] \end{aligned} \tag{39.81}$$

Comparing Eq. (39.81) and Eq. (39.80) we see that they are indentical if we make the identification

$$W'_{\tilde{q'}_k \tilde{q'}_l} \equiv \frac{1}{2}\frac{1}{9} \sum_{a,b=1}^{3} \left[W_{\tilde{q}_k^a \tilde{q}_l^b} + W_{\tilde{q}_k^a \tilde{q}_l^{b*}} \right] \tag{39.82}$$

i.e. we get the same relation as for other sfermions, the only difference being that we in option b) should also take the average over the colour states.

39.21.2.7 Sfermion-squark annihilation

For clarity, if we have annihilation between a non-coloured sfermion and a squark, we would in the same way as in the previous subsection get

$$W'_{\tilde{f'}_k \tilde{q'}_l} \equiv \frac{1}{2}\frac{1}{3} \sum_{b=1}^{3} \left[W_{\tilde{f}_k \tilde{q}_l^b} + W_{\tilde{f}_k \tilde{q}_l^{b*}} \right] \tag{39.83}$$

39.21.2.8 Summary of degrees of freedom

We have found above the following relations between option b) and option a),

$$\begin{cases} W'_{\chi_i^0 \chi_j^\pm} \equiv W_{\chi_i^0 \chi_j^+} = W_{\chi_i^0 \chi_j^-} \quad , \quad \forall\ i = 1,\ldots,4,\ j = 1,2 \\ W'_{\chi_i^\pm \chi_j^\pm} \equiv \frac{1}{2}\left[W_{\chi_i^+ \chi_j^+} + W_{\chi_i^+ \chi_j^-} \right] = \frac{1}{2}\left[W_{\chi_i^- \chi_j^-} + W_{\chi_i^- \chi_j^+} \right] \quad , \quad \forall\ i = 1,2,\ j = 1,2 \\ W'_{\chi_i^0 \tilde{f'}_k} \equiv W_{\chi_i^0 \tilde{f}_k} \quad , \quad \forall i = 1,\ldots 4,\ k = 1,2 \\ W'_{\chi_c^\pm \tilde{f'}_k} \equiv \frac{1}{2}\left[W_{\chi_c^+ \tilde{f}_k} + W_{\chi_c^+ \tilde{f}_k^*} \right] \quad , \quad \forall c = 1,2,\ k = 1,2 \\ W'_{\tilde{f'}_k \tilde{f'}_l} \equiv \frac{1}{2}\left[W_{\tilde{f}_k \tilde{f}_l} + W_{\tilde{f}_k \tilde{f}_l^*} \right] \quad , \quad \forall k = 1,2,\ l = 1,2 \\ W'_{\tilde{q'}_k \tilde{q'}_l} \equiv \frac{1}{2}\frac{1}{9}\sum_{a,b=1}^{3}\left[W_{\tilde{q}_k^a \tilde{q}_l^b} + W_{\tilde{q}_k^a \tilde{q}_l^{b*}} \right] \quad , \quad \forall k = 1,2,\ l = 1,2 \end{cases} \tag{39.84}$$

We don't list all the possible cases with squarks explicitly, the principle being that we in option b) should take the *average* over the squark colour states (see the squark-squark entry in the list above).

We will choose option b) and the code (dsandwdcoscn, dsandwdcoscn, dsasdwdcossfsf and dsasdwdcossfchi) should thus return W' as defined above. Note again that squarks are assumed to have $g_i = 6$ degrees of freedom in this convention and the summing over colours should also be taken into account in the code.

39.22 Relic density – more details on routines

39.22.1 Global parameters

When the relic density has been calculated, the integer variable copart in dsandwcom.h is set to indicate which coannihilating particles that have been included in the calculation. In Table 39.4, the meaning of this variable is shown for the case of supersymmetric particles.

copart			PAW variables		
Bit set	Octal value	Decimal value	cop1 bit	cop2 bit	Particle
0	1	1	0	–	$\tilde{\chi}_1^0$
1	2	2	1	–	$\tilde{\chi}_2^0$
2	4	4	2	–	$\tilde{\chi}_3^0$
3	10	8	3	–	$\tilde{\chi}_4^0$
4	20	16	4	–	$\tilde{\chi}_1^\pm$
5	40	32	5	–	$\tilde{\chi}_2^\pm$
6	100	64	6	–	$\tilde{e}_1$
7	200	128	7	–	$\tilde{\mu}_1$
8	400	256	8	–	$\tilde{\tau}_1$
9	1 000	512	9	–	$\tilde{e}_2$
10	2 000	1 024	10	–	$\tilde{\mu}_2$
11	4 000	2 048	11	–	$\tilde{\tau}_2$
12	10 000	4 096	12	–	$\tilde{\nu}_e$
13	20 000	8 192	13	–	$\tilde{\nu}_\mu$
14	40 000	16 384	14	–	$\tilde{\nu}_\tau$
15	100 000	32 768	–	0	$\tilde{u}_1$
16	200 000	65 536	–	1	$\tilde{c}_1$
17	400 000	131 072	–	2	$\tilde{t}_1$
18	1 000 000	262 144	–	3	$\tilde{u}_2$
19	2 000 000	524 288	–	4	$\tilde{c}_2$
20	4 000 000	1 048 576	–	5	$\tilde{t}_2$
21	10 000 000	2 097 152	–	6	$\tilde{d}_1$
22	20 000 000	4 197 304	–	7	$\tilde{s}_1$
23	40 000 000	8 388 608	–	8	$\tilde{b}_1$
24	100 000 000	16 777 216	–	9	$\tilde{d}_2$
25	200 000 000	33 554 432	–	10	$\tilde{s}_2$
26	400 000 000	67 108 864	–	11	$\tilde{b}_2$

Table 39.4: The bits of copart are set to indicate which initial states that are included in the coannihilation calculation. In the output file *.omegaco, the value of copart is written in octal format. In PAW cop1 and cop2 are available. Check if a bit is set with btest(cop1,bit).

39.22.2 Routine headers – fortran files

dsrdparticles.f

```
      subroutine dsrdparticles(option,nsize,TSM,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
*********************************************************************
*** subroutine dsrdparticles returns which particles are included in
***   the calculation of the WIMP relic density (coannihilations,
***   resonances, thresholds)
***
***  type : interface
***
***  desc : Particles included in relic density calculation
***  desc : (coannihilations, resonances and thresholds)
***
*** input:
***   option = 0 - no coann
***            1 - include all relevant coannihilations (charginos,
***                neutralinos and sleptons; Delta m/m<1.5)
***            2 - include only coannihilations betweeen charginos
```

```
***                 and neutralinos (Delta m/m<1.5)
***             3 - include only coannihilations between sfermions
***                 and the lightest neutralino (Delta m/m<1.5)
***           101 - include all relevant coannihilations (charginos,
***                 neutralinos and sleptons; Delta m/m<2.1)
***           102 - include only coannihilations betweeen charginos
***                 and neutralinos (Delta m/m<2.1)
***           103 - include only coannihilations between sfermions
***                 and the lightest neutralino (Delta m/m<2.1)
***   nsize = size of arrays, do not fill larger than this
***   TSM   = finite temperature of the SM heat bath
***           (not supported in module MSSM)
***
*** output
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)
***    ncoann - number of particles coannihilating
***    mgev  - relic and coannihilating mass in gev
***    dof   - internal degrees of freedom of the particles
***    nrs   - number of resonances to take special care of
***    rm    - mass of resonances in gev
***    rw    - width of resonances in gev
***    nt    - number of thresholds to take special care of
***            do not include coannihilation thresholds (that's automatic)
***    tm    - sqrt(s) of the thresholds in gev
*** This output is returned (used by the RD routines in src/)
*** and also, together with kcoann, put in common block needed by dsandwdcos.
*** author: paolo gondolo
*** derived from dsrdomega
*** date: 13-10-04
*** Modified: Joakim Edsjo, edsjo@fysik.su.se, 2015-12-08
*** Modified: Torsten Bringmann 2018-02-28 (added selfcon)
*** mod 2021-12-20 [tb]: added TSM as argument
**********************************************************************
```

dsrdwdwdcos.f

```
      subroutine dsrdwdwdcos(p,n)

************************************************************
** write out a table of dsandwdcos as a function of costheta for  **
** the given p and with n number of steps (n+1 points)        **
************************************************************
```

dsrdwrate.f

```
**********************************************************************
      subroutine dsrdwrate(unit1,unit2,ich)
c----------------------------------------------------------------------
c  write out a table of
c    initial cm momentum p
c    invariant annihilation rate w
c  input:
c    unit1 - logical unit to write total rate to (integer)
c    unit2 - logical unit to write partial differ. rates to (integer)
c    ich   - what initial channel to look at:
c              ich=1 nn-ann.  ich=2 cn-ann.  ich=3  cc-ann
c  author: paolo gondolo (gondolo@lpthe.jussieu.fr) 1994
c  changes by je to include prtial in partials and coannihilation routines
c======================================================================
```

39.23 mssm/se_yield: Yields from annihilation in the Sun/Earth

39.23.1 Routine headers – fortran files

dsseyield.f

```
************************************************************************
*** Function dsseyield calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth summing
*** over all annihilation channels and Higgs decays as setup with dsanyieldset.
***
***  type : interface
***
*** Inputs:
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1.
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
*** Author: Joakim Edsjo (edsjo@fysik.su.se)
*** Date: April 9, 2008
```

```
************************************************************************

      real*8 function dsseyield(e,theta,wh,kind,type,istat)
```

dsseyield_ch.f

```
      real*8 function dsseyield_ch(mneu,e,theta,pdg1,pdg2,
     &  wh,kind,type,istat)
************************************************************************
*** Function dsseyield_ch calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth for a given
*** annihilation channel. Channels 1-xx are supported. Channel 10 (mu- mu+)
*** always gives zero (as muons are stopped in the Sun/Earth), but is
*** included to keep the channel numbering the same as for halo annihilation
*** routines.
*** Inputs:
***   mneu  - WIMP mass (GeV)
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   pdg1, pdg2 -- PDG codes of final state particles
***
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1 (kind 2) and GeV**-1 (kind 3)
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
```

```
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
*** Author: Joakim Edsjo (edsjo@fysik.su.se)
*** Date: 1995
*** Modified: April 3, 2008
*** Modified: May 6, 2011 Pat Scott (patscott@physics.mcgill.ca)
*** Modified: December, 2014 (edsjo)
**********************************************************************
*$      use omp_lib
```

39.24 mssm/se_yield_casc: Yields from annihilation in the Sun/Earth coming from cascade decays

39.24.1 Routine headers – fortran files

dssedydth.f

```
**********************************************************************
*** function dssedydth is the differential yield dyield/dcostheta which
*** should be integrated (by the routine gadap e.g.).
*** cth is cos(theta).
*** units: 1.0e-15 m**-2 (annihilation)**-1
**********************************************************************

      real*8 function dssedydth(cth,m0,m1,m2,e0,eth,
     &  thm,chi,wwh,fk,fv)
```

dsseemean.f

```
**********************************************************************
*** function dsseemean is used to calculate the mean energy of a decay product
*** when a moving particle decays. e0 and m0 are the energy and mass of
*** the moving particle and m1 and m2 are the masses of the decay products.
*** it is the mean energy of m1 that is returned. all energies and masses
*** should be given in gev.
**********************************************************************

      real*8 function dsseemean(e0,m0,m1,m2)
```

dsseyield_int.f

```
      real*8 function dsseyield_int(f,a,b,
     &     chi,wwh,m0,m1,m2,
     & e0,eth,thm,fk,fv,seerror)
c---------------------------------------------------------------------
c  integrate function f between a and b
c  Note, we give all arguments here instead of having them via common
c  blocks to allow for Open MP parallellization
c  input
c    integration limits a and b
c  called by dsseyieldfth
c  author: joakim edsjo (edsjo@fysik.su.se) 96-05-16
c  based on paolo gondolos wxint.f routine.
c=====================================================================
```

dsseyieldfth.f

```
**********************************************************************
```

```
*** function phiith integrates dssedydth over the angle theta.
*** it is the yield from particle 1 (which decays from m0)
*** that is calculated. particle one corresponds to WS channel chi
*** units: 1.0e-30 m**-2 (annihilation)**-1
************************************************************************

      real*8 function dsseyieldfth(e0,m0,mp1,mp2,emuthr,thmax,chi,wh,
     &  kind,type,istat,seerror)
```

dsseyields.f

```
************************************************************************
*** function dsseyields calculates the yield above threshold (kind=1) or the
*** differential yield (kind=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
************************************************************************

      real*8 function dsseyields(eh,emuth,thmax,hno,wh,kind,
     &  type,istat,seerror)
```

dsseyields2.f

```
************************************************************************
*** function dsseyields2 calculates the yield above threshold (kind=1) or the
*** differential yield (kind=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
************************************************************************

      real*8 function dsseyields2(eh,emuth,thmax,hno,wh,kind,
     &  type,istat,seerror)
```

dsseyields3.f

```
************************************************************************
*** function dsseyields3 calculates the yield above threshold (kind=1) or the
*** differential yield (kind=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
************************************************************************

      real*8 function dsseyields3(eh,emuth,thmax,hno,wh,kind,
     &  type,istat,seerror)
```

dsseyields4.f

```
************************************************************************
*** function dsseyields4 calculates the yield above threshold (kind=1) or the
*** differential yield (kind=2) from a given scalar
*** boson decaying in flight, the energy of the scalar boson should be given
*** in eh.
*** scalars hno = 1-4 are supported (S10, S20, S30 and S+/S-)
*** units: 1.0e-30 m**-2 (annihilation)**-1
************************************************************************
```

```
      real*8 function dsseyields4(eh,emuth,thmax,hno,wh,
     &  kind,type,istat,seerror)
```

39.25 mssm/xfeynhiggs: FeynHiggs interface

39.25.1 Routine headers – fortran files

dsfeynhiggs.F

```
       subroutine dsfeynhiggs(hwar)

c -------------------------------------------------------------------
c Implementation of FeynHiggs in DarkSUSY by
c Author: J. Edsjo, edsjo@fysik.su.se
c Date: 2008-06-24.
c Modified: 2008-07-02 by J. Edsjo, small bugs resolved.
c Based on implementation for FeynHiggs 2.4.1 by P. Gondolo
c Modified:2009-03-16 by E. Lundstrom, extraction of (partial)
c widhts and production cross section ratios (needed in Higgs
c Bounds) from FeynArts.
c 2010-09-24: Updated to work with FeynHiggs 2.7.2
c 2011-09-04: PG update to FeynHiggs 2.8.5
c 2013-02-09: PG update to FeynHiggs 2.9.4
c 2016-04-12: JE update to FeynHiggs 2.11.3 (here and in dshiggsbounds)
c    changed to new best looplevel and new best running_MT. Lightest
c    CP-even Higgs mass (H2) shift by a few GeV which affect mainly
c    scattering rates.
c 2017-12-19: JE update to FeynHiggs 2.13.0 (here and in dshiggsbounds)
c    set loglevel, extract deltar observable
c
c Output: mh is lighter scalar Higgs mass, HM is heavier Higgs mass
c -------------------------------------------------------------------

c -------------------------------------------------------------
c
c Warnings implemented by Joakim Edsjo
c
c                   Bit  Value
c  Bits of hwar:    0 -  1:   Error in FHSetFlags
c                   1 -  2:   Error in FHSetSMPara
c                   2 -  4:   Error in FHSetPara
c                   3 -  8:   Error in FHHiggsCorr
c                   4 -  16:  Error in FHConstraints
c                   5 -  32:  Error in FHSelectUZ
c                   6 -  64:  Error in FHCouplings
c                   7 -  128: Error in FHHiggsProd !Currently not used
c                   8 -  256: Other error (NaN returned)
c-------------------------------------------------------------
```

39.26 mssm/xhiggsbounds: HiggsBounds interface

39.26.1 Routine headers – fortran files

dshiggsbounds.F

```
c This is a modified version of the HBandHSwithFH.F file supplied with
```

```
c HiggsSignals 2.5.1.
c The routine runs FeynHiggs to evaluate the partonic input of
c HiggsBounds/HiggsSignals for MSSM.

c This file assumes that some FeyhHiggs subroutines have been
c called previously in dsfeynhiggs. They are marked below with
c 'done in dsfeynhiggs'.

* Updated for FeynHiggs 2.11.3 by JE 2016-04-11
* Updated for FeynHiggs 2.13.0 by JE 2017-12-19
* Updated for FeynHiggs 2.17.0, HB 5.9.0 and HS 2.5.1 2020-12-17
* Modified for DarkSUSY by JE 2016-06-28
* Modified to check whether HS&HB have been installed successfully by TB 2021-12-20


      subroutine dshiggsbounds(HBresult,pvalue)
c      use usefulbits_hs, only : HSres
c      use STXS, only : print_STXS
```

39.27 mssm/xsuperiso: SuperIso interface

Chapter 40

Silveira-Zee (Scalar Singlet)

The module silveira_zee implements the scalar singlet DM model of Silveira and Zee [4], also referred to as 'scalar phantoms' (which is the original name in [4]) and 'singlet Higgs DM'. This model adds a gauge-singlet real scalar field S to the standard model, with imposed Z_2 symmetry $S \to -S$. Its Lagrangian is

$$\mathcal{L}_{\rm SZ} = \mathcal{L}_{\rm SM} + \frac{1}{2}\partial_\mu S \partial^\mu S - \frac{1}{2}\mu^2 S^2 - \frac{1}{2}\lambda S^2 H^\dagger H, \tag{40.1}$$

where H is the Standard Model Higgs doublet. After electroweak symmetry breaking, the S boson acquires a tree-level mass

$$m_S = \sqrt{\mu^2 + \frac{1}{2}\lambda v_0^2}\,, \tag{40.2}$$

where $v_0 = (\sqrt{2}G_F)^{-1/2} = 246.2$ GeV is the Higgs vacuum expectation value. The module uses the S mass m_S and the S-Higgs coupling constant λ as model parameters. These parameters are set with a call to dsgivemodel_silveira_zee, followed as usual by a call to dsmodelsetup to initialize the model for use with DarkSUSY.

40.1 silveira_zee/an: Annihilation routines

40.1.1 Routine headers – fortran files

dsanunitaritycut.f

```
************************************************************************
***  Function dsanunitaritycut returns Ecut used in the default       ***
***  unitarization prescription. For E<Ecut, the unitarized version of ***
***  the Scalar singlet annihilation cross section is used (evaluating the ***
***  SM Higgs decay width at ECM rather than m_h in the propagator), while ***
***  for E>Ecut the tree level result for the (partial) Higgs width is used.***
***                                                                   ***
***  input:                                                           ***
***     TSM -- SM heat bath temperature [GeV]                         ***
***                                                                   ***
***  Return value: Ecut [GeV]                                         ***
***                                                                   ***
***  Author: Torsten Bringmann                                        ***
***  Date: 02/09/2021                                                 ***
************************************************************************
      real*8 function dsanunitaritycut(TSM)
```

dsanwx.f

```
************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.    ***
***                                                                      ***
***  type : INTERFACE                                                    ***
***  desc : Self-annihilation invariant rate                             ***
***                                                                      ***
***  Input:                                                              ***
***    p - initial cm momentum (real) for DM annihilations               ***
***  Output:                                                             ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  passed to dsrdens by dsrdomega.                                     ***
***
*** If v denotes the relative velocity of one particle in the frame of the
*** other, then the invariant rate W is
***  W = 4 p1.p2 sigma v = 4 (p^2+sqrt(p^2+m1^2)*sqrt(p^2+m2^2)) sigma v
*** and for equal mass m1=m2==mx
***  W = 4 (mx^2+2*p^2) sigma v
***                                                                      ***
***  author: Paolo Gondolo 2016                                          ***
***  mod TB 20-10-30: caught NAN, negative output                        ***
************************************************************************
      function dsanwx(p)
```

dsanwx_finiteT.f

```
************************************************************************
*** Function dsanwx_finiteT provides the effective, invariant rate for DM   ***
*** self-annihilation. Same as dsanwx, but including finite temperature     ***
*** effects in the cross section, as well as quantum statistics effects     ***
*** from the final state (and/or virtual heat bath) particles.              ***
***                                                                         ***
***  type : INTERFACE                                                       ***
***  desc : Self-annihilation invariant rate at finite temperature          ***
***                                                                         ***
***  Input:                                                                 ***
***    p   - cm momentum of each of the initial DM particles [GeV]          ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma          ***
***          temperature [both in GeV] at which sigma v is evaluated        ***
***                                                                         ***
***  Author: Torsten Bringmann                                              ***
***  Date: 21/09/2021                                                       ***
************************************************************************
      real*8 function dsanwx_finiteT(p,x)
```

dssigmav.f

```
************************************************************************
*** function dssigmav returns the *total* annihilation cross section        ***
*** sigma v for DM-DM annihilation. Here v is defined to be the relative    ***
*** velocity of one DM particle in the frame of the other.                  ***
***                                                                         ***
***  Input:                                                                 ***
***    p - cm momentum of either of the annihilating DM particles [GeV]     ***
***                                                                         ***
```

```
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                       ***
*** Notice that in 1306.4710, their sigma v_rel is related to the invariant ***
*** rate W via W = s sigma v_rel, with sigma v_rel being the relative      ***
*** velocity in the CMS frame. Hence,                                     ***
***    sigma v = s/4/(p^2+sqrt(p^2+m1^2)*sqrt(p^2+m2^2)) sigma v_rel      ***
***            = (mx^2+p^2)/(mx^2+2*p^2) sigma v_rel,                     ***
*** where the latter holds for identical initial state masses [m1=m2==mx]. ***
***                                                                       ***
*** author: Paolo Gondolo 2016                                            ***
*** mod: 2021-08-21 -- added hadron contribution [tb]                     ***
***      2021-09-01 -- added broken theory sv [tb]                        ***
*****************************************************************************
      function dssigmav(p)
```

dssigmav0tot.f

```
*****************************************************************************
*** function dssigmav0tot returns the *total* annihilation cross section   ***
*** sigma v at p=0 for WIMP-WIMP annihilation.                             ***
***                                                                       ***
***  type : INTERFACE                                                     ***
***                                                                       ***
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                       ***
*** The total cross section is obtained by summing over all implemented    ***
*** 2-body channels, plus contributions from final states with more        ***
*** particles (currently none in this particle module).                    ***
***                                                                       ***
*** author: Paolo Gondolo 2016                                            ***
*****************************************************************************
      real*8 function dssigmav0tot()
```

dssigmav_eff.f

```
*****************************************************************************
*** function dssigmav_eff returns the *effective*, total annihilation cross ***
*** section sigma v for DM-DM annihilation in the plasma (or COSMIC) frame  ***
*** and at finite temperature, as it appears on the right-hand side of the  ***
*** Boltzmann equation. Here v is defined to be the relative velocity of    ***
*** one particle in the frame of the other. For a full definition see       ***
*** (A.1,A.2) in 2111.14871                                                ***
***                                                                       ***
***  Input:                                                               ***
***    p   - cm momentum of each of the initial DM particles [GeV]        ***
***    x   - m_DM/T, where m_DM is the DM mass and T the SM/plasma         ***
***          temperature [both in GeV] at which sigma v is evaluated      ***
***                                                                       ***
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                       ***
*** Author: Torsten Bringmann                                             ***
*** Date: 21/09/2021                                                      ***
*****************************************************************************
      real*8 function dssigmav_eff(p,x)
```

dssigmav_finiteT.f

```
*****************************************************************************
*** function dssigmav_finiteT returns the *total* annihilation cross       ***
*** section sigma v at p=0 for WIMP-WIMP annihilation, evaluated in the    ***
*** CMS frame and at finite temperature. Here v is defined to be the       ***
*** relative velocity of one particle in the frame of the other.           ***
***                                                                       ***
```

```
***  Input:                                                                 ***
***    p     - cm momentum of each of the initial DM particles [GeV]       ***
***    gamma - Lorentz boost wrt the frame of the heat bath                ***
***    TSM   - SM/plasma temperature [in GeV] at which sigma v is evaluated ***
***                                                                         ***
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                         ***
*** Author: Torsten Bringmann (2021)                                      ***
***         (based on previous version by Kristian Vangsnes)              ***
*******************************************************************************
      real*8 function dssigmav_finiteT(p,gamma,TSM)
```

dssigmavpartial.f

```
*******************************************************************************
*** function dssigmavpartial returns the partial annihilation cross section ***
*** sigma v for DM-DM annihilation into channel 'ichannel'. Here v is       ***
*** defined to be the relative velocity of one DM particle in the frame of  ***
*** the other.                                                              ***
***                                                                         ***
***  Input:                                                                 ***
***    p - cm momentum of either of the annihilating DM particles [GeV]     ***
***    ichannel - final (SM) states as follows:                             ***
***            = 1  :  nue + anti-nue                                       ***
***            = 2  :  e+ + e-                                              ***
***            = 3  :  numu + anti-numu                                     ***
***            = 4  :  mu+ + mu-                                            ***
***            = 5  :  nutau + anti-nutau                                   ***
***            = 6  :  tau+ + tau-                                          ***
***            = 7  :  u + ubar                                             ***
***            = 8  :  d + dbar                                             ***
***            = 9  :  c + cbar                                             ***
***            = 10  :  s + sbar                                            ***
***            = 11  :  t + tbar                                            ***
***            = 12  :  b + bbar                                            ***
***            = 13  :  gamma + gamma                                       ***
***            = 14  :  W+ + W-                                             ***
***            = 15  :  Z + Z                                               ***
***            = 16  :  g + g                                               ***
***            = 17  :  H + H                                               ***
***            = 18  :  Z + gamma                                           ***
***            = 19  :  hadrons (below confinement scale)                   ***
***                                                                         ***
*** Units of returned cross section: cm^3 s^-1                            ***
***                                                                         ***
*** NB: Channels 7-12 & 16 assume free quarks and gluons, and hence return  ***
***     zero below the confinement scale. Conversely, ichannel=19 returns   ***
***     the total hadronic widths below the confinement scale, and zero     ***
***     otherwise. (linear interpolation to 0 between Ehadron_min and       ***
***     Ehadron_max, set in dsinit_module)                                  ***
***                                                                         ***
*** author: Paolo Gondolo 2016                                             ***
*** mod: 2021-08-21 -- added hadron contribution [tb]                      ***
***      2021-09-01 -- added broken theory sv [tb]                         ***
*******************************************************************************
      real*8 function dssigmavpartial(ichannel,p)
```

dssigmavpartial_finiteT.f

```
*******************************************************************************
*** function dssigmavpartial_finiteT returns the partial annihilation cross ***
*** section sigma v for DM-DM annihilation into channel 'ichannel',         ***
*** evaluated in the CMS frame and at finite temperature. Here v is defined ***
*** as the relative velocity of one DM particle in the frame of the other.  ***
```

```
***                                                                          ***
***  Input:                                                                  ***
***    p - cm momentum of either of the annihilating DM particles [GeV]      ***
***    gamma - Lorentz boost wrt the frame of the heat bath                  ***
***    TSM   - SM/plasma temperature [in GeV] at which sigma v is evaluated  ***
***    ichannel - final (SM) states as follows:                              ***
***            = 1  :  nue + anti-nue                                        ***
***            = 2  :  e+ + e-                                               ***
***            = 3  :  numu + anti-numu                                      ***
***            = 4  :  mu+ + mu-                                             ***
***            = 5  :  nutau + anti-nutau                                    ***
***            = 6  :  tau+ + tau-                                           ***
***            = 7  :  u + ubar                                              ***
***            = 8  :  d + dbar                                              ***
***            = 9  :  c + cbar                                              ***
***            = 10  :  s + sbar                                             ***
***            = 11  :  t + tbar                                             ***
***            = 12  :  b + bbar                                             ***
***            = 13  :  gamma + gamma                                        ***
***            = 14  :  W+ + W-                                              ***
***            = 15  :  Z + Z                                                ***
***            = 16  :  g + g                                                ***
***            = 17  :  H + H                                                ***
***            = 18  :  Z + gamma                                            ***
***            = 19  :  hadrons (below confinement scale)                    ***
***                                                                          ***
*** Units of returned cross section: cm^3 s^-1                               ***
***                                                                          ***
*** NB: Channels 7-12 & 16 assume free quarks and gluons, and hence return   ***
***     zero below the qcd phase transition and confinement scale.           ***
***     Conversely, ichannel=19 returns the total hadronic widths in this    ***
***     case, and zero otherwise.                                            ***
***     For CMS energies close to the confinement scale, and TSM around      ***
***     TQCD, linear interpolations between these regimes are adopted.       ***
***                                                                          ***
*** author: Paolo Gondolo 2016                                               ***
*** mod: 2021-08-21 -- added hadron contribution [tb]                        ***
***      2021-09-01 -- added broken theory sv [tb]                           ***
********************************************************************************
      real*8 function dssigmavpartial_finiteT(ichannel,p,gamma,TSM)
```

40.2 silveira_zee/cr: Cosmic rays

40.2.1 Routine headers – fortran files

dscrsource.f

```
********************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
```

```
***                  [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2016-06-28
**************************************************************************

      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
**************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2016-06-28
*** mod   : 2019-06-20 tb, added kinematics check
**************************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

40.3 silveira_zee/dd: Direct detection

40.3.1 Routine headers – fortran files

dsddgpgn.f

```
      subroutine dsddgpgn(gg,ierr)
**************************************************************************
***  subroutine dsddgpgn returns DM nucleon four-fermion couplings          ***
***                                                                         ***
***  type : interface                                                       ***
```

```
***         [NB: this routine will eventually be phased out as *interface*  ***
***              routine. Currently, neutrino telescope routines are the    ***
***              only routines in /src that access it. ]                    ***
***                                                                         ***
***   output:                                                               ***
***     gg     : complex*16 (ddng,2) : four-fermion couplings in GeV^-2       ***
***                                                                         ***
***  author: paolo gondolo (paolo.gondolo@utah.edu) 2016                    ***
*******************************************************************************
```

40.4 silveira_zee/fi: Freeze-in routines

A call to dsfiset_silveira_zee prior to dsfi2to2rhs allows to perform freeze-in calculations with modified default settings. For example, call dsfiset_silveira_zee('quantum_statistics','off') will lead to a freeze-in calculation without taking into account the effect of quantum statistics for the heat bath particles, and call dsfiset_silveira_zee('finiteT','off') will switch off other finite-temperature effects (thermal masses and further temperature-dependence due to phase transitions). Default settings are restored with corresponding calls to the same routine, replacing 'off' with 'default'.

40.4.1 Routine headers – fortran files

dsfiset_silveira_zee.f

```
      recursive subroutine dsfiset_silveira_zee(key,value)
c...  desc : Set parameters for freeze-in routines
c...  key,value - character strings specifying choice to be made
c...  author: torsten bringmann 2021-08-13
```

40.5 silveira_zee/ge: General routines

40.5.1 Routine headers – fortran files

dsdmspin.f

```
*******************************************************************************
*** Function dsdmspin returns the WIMP spin (in hbar)                       ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  desc : WIMP spin                                                       ***
***                                                                         ***
*** author: Paolo Gondolo 2016-11-20                                        ***
*******************************************************************************

      real*8 function dsdmspin()
```

dsgammah.f

```
*******************************************************************************
*** Function dsgammah returns the offshell total SM Higgs width in the      ***
*** Silveira-Zee module, at finite temperature and including both the SM    ***
*** contribution and that from invisible decays to scalar singlets.         ***
***                                                                         ***
***  Input:                                                                 ***
***    sqrts - CMS energy of (off-shell) Higgs [GeV]                        ***
```

```
***    gamma - Lorentz boost wrt the frame of the heat bath                ***
***    TSM   - SM/plasma temperature [in GeV]                              ***
***                                                                        ***
***  Return value is the decay width in GeV, evaluated in the CMS frame.   ***
***                                                                        ***
*** Based on Eqs.(2.24, 4.13) of 2111.14871, and the SM contribution at    ***
*** zero temperature as returned by dssmgammah(partial). For T<TQCD, the   ***
*** decay to hadronic final states is instead based on                      ***
*** dssmgammah_hadron_tab. See dsinit_sm to steer the default behaviour    ***
*** of these functions (flag 'gammahhow').                                 ***
***                                                                        ***
*** Author: torsten.bringmann@fys.uio.no, 2021-11-13                       ***
***         (original routine for T=0 by Paolo Gondolo)                    ***
******************************************************************************
      real*8 function dsgammah(sqrts,gamma,TSM)
```

dsmwimp.f

```
******************************************************************************
*** Function dsmwimp returns the WIMP mass                                 ***
***                                                                        ***
***  type : INTERFACE                                                      ***
***                                                                        ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                       ***
******************************************************************************

      real*8 function dsmwimp()
```

40.6 silveira_zee/ini: src_models/silveira_zee/ini

40.6.1 Routine headers – fortran files

dsgivemodel_silveira_zee.f

```
******************************************************************************
***  subroutine dsgivemodel_silveira_zee sets the model parameters         ***
***                                                                        ***
***  input:                                                                ***
***                                                                        ***
***    lambda - quartic coupling                                           ***
***    ms     - singlet Higgs mass in GeV                                  ***
***                                                                        ***
*** author: Paolo Gondolo                                                  ***
*** date 2016-05-19                                                        ***
******************************************************************************
      subroutine dsgivemodel_silveira_zee(mylambda,myms)
```

dsinit_module.f

```
******************************************************************************
***  This is the initialization subroutine for the "silveira_zee" module   ***
***                                                                        ***
***  Author: Paolo Gondolo, Torsten Bringmann                              ***
***  Date: 05/19/2016                                                      ***
***  mod 10/2021 (tb): added flags for hansling phases transition and      ***
***                    unitarization at very high CMS energies             ***
******************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
************************************************************************
***  subroutine dsmodelsetup computes various particle physics quantities   ***
***                                                                      ***
***  type : interface                                                    ***
***                                                                      ***
***  desc : Sets up a new particle physics model                         ***
***                                                                      ***
***  output:                                                             ***
***    ierr  -  0 no errors                                              ***
***          - <0 theoretically inconsistent model parameters            ***
***    iwarn -  0 no warnings                                            ***
***             1 invisible Higgs decay width too large                  ***
***             2 Singlet mass too small                                 ***
***             3 Singlet mass too large                                 ***
***                                                                      ***
*** author: Paolo Gondolo                                               ***
*** date 2016-05-19                                                     ***
*** Modified: Joakim Edsjo, 2019-12-13 (added SM width, fixed typo)     ***
************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

40.7 silveira_zee/kd: Kinetic decoupling

40.7.1 Routine headers – fortran files

dskdm2.f

```
*********************************************************************
*** dskdm2 returns the full scattering amplitude squared, SUMMED over
*** both initial and final spin and other internal states, and then
*** divided by the DM internal degrees of freedom.
*** The returned value is not evaluated at zero momentum transfer, but
*** averaged over t.
***
***  type : INTERFACE
***  desc : Full scattering amplitude squared, averaged over momentum transfer
***
***  input: omega -- CMS MOMENTUM of scattering partner
***  output: omega -- CMS ENERGY of scattering partner
***
***  input:   SMtype - SM scattering partners
***             "    = 7,8,9,10,11,12 - d,u,s,c,b,t quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten.bringmann@fys.uio.no, 2016-12-21
*** updated 2018-05-18 : moved momentum->energy conversion to src_models
*********************************************************************

      real*8 function dskdm2(omega,SMtype)
```

dskdm2simp.f

```
*********************************************************************
*** dskdm2simp returns the scattering amplitude squared, averaged over t.
*** Here, |M|^2 is SUMMED over both initial and final spin and
*** other internal states, and then divided by the DM internal degrees of
*** freedom. This is only valid in the limit of relativistic scattering
*** partners with small energies omega, where we can expand as
```

```
*** and averaged over the
***
***    <|M|**2>_t = cn*(omega/m0)**n + O( (omega/m0)**(n+1) )
***
***  type : INTERFACE
***  desc : Scattering amplitude squared, expanded in powers of energy
***
***  input: SMtype   - SM scattering partners:
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
*** author: torsten.bringmann@fys.uio.no, 2016-06-29
**********************************************************************

      subroutine dskdm2simp(SMtype,cn,n)
```

dskdparticles.f

```
**********************************************************************************
*** Prepares integration of Boltzmann equation and has to be called         ***
*** before dskdboltz (called by dskdtkd)                                    ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date  : 2016-06-29                                                      ***
**********************************************************************************
      subroutine dskdparticles()
```

40.8 silveira_zee/rd: Relic density

40.8.1 Routine headers – fortran files

dsrdparticles.f

```
**********************************************************************************
*** subroutine dsrdparticles returns kinematic information about the        ***
*** particles that are included in the calculation of the DM relic density  ***
*** (coannihilations, resonances  thresholds, thermal effects).             ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  desc : Particles included in relic density calculation                 ***
***  desc : (coannihilations, resonances and thresholds)                    ***
***                                                                         ***
*** Input:                                                                  ***
***    option - 0        = default                                          ***
***             1..99    = include various subsets of coannihilations       ***
***                        (not relevant in silveira_zee module)            ***
***    TSM   - finite temperature of the SM heat bath                       ***
***            (if TSM=0d0, finite-T effects will be ignored)               ***
***                                                                         ***
***    nsize - size of arrays, do not fill larger than this                 ***
***                                                                         ***
*** Output                                                                  ***
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)      ***
***    ncoann   - number of particles coannihilating                        ***
***    mcoann   - relic and coannihilating mass [GeV]                       ***
***    dof      - internal degrees of freedom of the particles              ***
***    nrs      - number of resonances to take special care of              ***
***    rm       - mass of resonances [GeV]                                  ***
```

```
***    rw       - width of resonances [GeV]                                ***
***    nt       - number of thresholds to take special care of             ***
***               (NB: coannihilation thresholds are automatic            ***
***                -- do not include here!)                               ***
***    tm       - sqrt(s) of the thresholds [GeV]                          ***
***                                                                        ***
*** This output is used by the RD routines in src/rd/ and src/fi           ***
***                                                                        ***
***                                                                        ***
*** author: paolo gondolo, derived from dsrdomega                          ***
*** date: 13-10-04                                                         ***
*** Modified: Joakim Edsjo, edsjo@fysik.su.se, 2015-12-08                  ***
*** Modified: Torsten Bringmann 2016-06-28                                 ***
*** Modified: Torsten Bringmann 2018-02-28 (added selfcon)                 ***
*** Modified: Torsten Bringmann 2018-02-28 (added further thresholds)      ***
*** Modified: Torsten Bringmann 2021-10-21 (added finite-T option)         ***
**************************************************************************
      subroutine dsrdparticles(option,nsize,TSM,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
```

40.9 silveira_zee/se_yield: Yields from annihilation in the Sun/Earth

40.9.1 Routine headers – fortran files

dsseyield.f

```
**************************************************************************
*** Function dsseyield calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth summing
*** over all annihilation channels.
***
***  type : INTERFACE
***
*** Inputs:
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
```

```
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1.
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
***
*** Author: Torsten.Bringmann@fys.uio.no
*** Date: 22/06/2015
**************************************************************************

      real*8 function dsseyield(e,theta,wh,kind,type,istat)
```

Chapter 41

Self-Interacting Dark Matter

The vdSIDM module implements a simplified model setup where DM is confined to a dark sector without (direct) interactions with the standard model. Besides the DM particle, which is assumed to be non-relativistic, there is also a dark radiation (DR) particle which in general will contribute to the relativistic degrees of freedom in the universe – a quantity which is conventionally stated in terms of the contribution from a single neutrino species, $\Delta N_{\rm eff}$, and which is tightly constrained by both BBN and CMB. Finally, the module implements a mediator particle. In principle, the code structure is set up such that arbitrary spin combinations of these three particles can be implemented, but presently only scalar and vector mediators (for Dirac DM and DR) are available. If the mass of the mediator particle is much lighter than the DM mass, the resulting Yukawa potential between DM particles results in a self-scattering rate that is strongly velocity-dependent – as indicated in the name of the module.

41.1 vdSIDM/an: Annihilation routines

For the spin combinations that are currently implemented in the vdSIDM module, DM annihilates via the t- and u-channel to a pair of mediators. In the limit of small CMS energies $\sqrt{s} \to 2m_\chi$, this gives the following contributions to the invariant rate:

$$W_{\chi\chi\to VV} \xrightarrow[p_{\rm CM}\ll m_\chi]{} 4\pi\alpha_\chi^2 \left(1-\frac{m_V^2}{m_\chi^2}\right)^{3/2} \left(1-\frac{m_V^2}{2m_\chi^2}\right)^{-2} \tag{41.1}$$

$$W_{\chi\chi\to\phi\phi} \xrightarrow[p_{\rm CM}\ll m_\chi]{} 6\pi\alpha_\chi^2 p_{\rm CM}^2 \left(1-\frac{m_\phi^2}{m_\chi^2}\right)^{1/2} \left(1-\frac{m_\phi^2}{2m_\chi^2}\right)^{-4} \left(1-\frac{8}{9}\frac{m_\phi^2}{m_\chi^2}+\frac{2}{9}\frac{m_\phi^4}{m_\chi^4}\right), \tag{41.2}$$

where $\alpha_\chi \equiv g_\chi/(4\pi)$ and $p_{\rm CM}$ is the initial CMS momentum of (each of) the DM particles. Furthermore, DM can annihilate to a pair of DR particles, by a mediator exchange in the s-channel. This gives

$$W_{\chi\chi\to V^*\to\tilde\gamma\tilde\gamma} = 8\pi\alpha_\chi s^2 \left(1+\frac{2m_\chi^2}{s}\right) \frac{\Gamma_{V\to\tilde\gamma\tilde\gamma}}{m_V} |D_V|^2 \tag{41.3}$$

$$W_{\chi\chi\to\phi^*\to\tilde\gamma\tilde\gamma} = 8\pi\alpha_\chi s^2 \left(1-\frac{4m_\chi^2}{s}\right) \frac{\Gamma_{\phi\to\tilde\gamma\tilde\gamma}}{m_\phi} |D_\phi|^2 , \tag{41.4}$$

where $D_{V,\phi}$ is defined as

$$|D_H(s)|^2 = \frac{1}{(s-m_H^2)^2 + m_H^2\Gamma_H^2}\,, \tag{41.5}$$

i.e. as the inverse of the denominator of the respective propagator.

The interface function dsanwx adds the full tree-level expressions for these processes, and then multiplies the leading-order expressions in the $v \to 0$ limit by an enhancement factor $S(v)$ due to the Sommerfeld effect. The factor $S(v)$ itself is provided by the auxiliary function dsansommerfeld residing in src_models/common/aux, which implements the analytic expressions from Ref. [219, 150] (resulting from approximating the Yukawa potential by a Hulthén potential). The Sommerfeld factor depends on whether the process is s-wave dominated (as in the vector mediator case) or p-wave dominated (as in the scalar mediator case), which hence is one of the input parameters for dsansommerfeld.

41.1.1 Routine headers – fortran files

dsanwx.f

```
************************************************************************
*** Function dsanwx provides the  WIMP self-annihilation invariant rate.    ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  Input:                                                                 ***
***    p - initial cm momentum (real) for DM annihilations                  ***
***  Output:                                                                ***
```

$$W_{\rm eff} = \sum_{ij} \frac{p_{ij}}{p_{11}} \frac{g_i g_j}{g_1^2} W_{ij} = \sum_{ij} \sqrt{\frac{[s-(m_i-m_j)^2][s-(m_i+m_j)^2]}{s(s-4m_1^2)}} \frac{g_i g_j}{g_1^2} W_{ij}.$$

where the p's are the momenta, the g's are the internal degrees of freedom, the m's are the masses and W_{ij} is the invariant annihilation rate for the included subprocess.

```
***  passed to dsrdens by dsrdomega.                                        ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                    ***
*** date 2015-06-11                                                         ***
************************************************************************
      real*8 function dsanwx(p)
```

dssigmav0tot.f

```
**********************************************************************
*** function dssigmav0tot returns the *total* annihilation cross section
*** sigma v at p=0 for WIMP-WIMP annihilation.
*** This is obtained by summing over all implemented 2-body channels
*** (as returned by dssigmav) plus contributions from final states with
*** more particles.
***
***  type : interface
***
*** Units of returned cross section: cm^3 s^-1
***
*** author: Torsten.Bringmann.fys.uio.no
*** date: 2018-02-18
**********************************************************************

      real*8 function dssigmav0tot()
```

41.2 vdSIDM/cr: Cosmic rays

The cosmic-ray source functions in vdSIDM simply return zero, as there are currently no couplings to SM particles implemented. In a future version of the code, the user will be able to add partial widths of the mediator particles to visible channels. The cosmic-ray source functions dscrsource and dscrsource_line will then be correspondingly updated.

41.2.1 Routine headers – fortran files

dscrsource.f

```
************************************************************************
***   function dscrsource returns the source term for DM-induced cosmic rays
***   (including neutral species), assuming that the corresponding flux scales
***   like a power of the DM density rho_DM. Note that monochromatic
***   contributions are not included here (see dscrsource_line for this).
***
***   type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of CR species
***          egev   - CR energy [in GeV]
***          diff   - dictates whether differential source term at egev (diff=1)
***                   or integrated source term above egev (diff=0) is returned
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: istat - will be zero in case of no errors.
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***         -> for diff=1, this is multiplied by 1/GeV
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2018-02-18
************************************************************************

      real*8 function dscrsource(egev,diff,pdg,power,v,istat)
```

dscrsource_line.f

```
************************************************************************
***   In analogy to dscrsource, subroutine dscrsource_line returns
***   *monochromatic* cosmic ray contributions to the source term, assuming
***   that the corresponding flux scales like a power of the DM density rho_DM.
***
***  type : interface
***
***   input: power  - determines scaling as flux ~ (rho_DM)^power
***          pdg    - PDG code of monochromatic CR species
***          n      - returns the nth monochromatic signal for this pdg code
***                   [if called with n=0, the first line will be returned,
***                    and n be set to the total number of existing lines]
***          v      - relative velocity of annihilating DM particles
***                   in units of c
***                   [only for power=2, otherwise ignored]
***
***   output: egev      - CR energy of particle pdg [in GeV]
***           widthline - signal width
***           pdg2      - pdgcode of associated 2nd final state particle
```

```
***           istat     - equals 0 if there are no errors, bit 1 is set if line
***                       n does not exist, higher non-zero bits specify model-
***                       specific errors
***
***   unit of return value: #particles / (cm^3 s) / (GeV / cm^3)^power
***         -> for power=2, this is multiplied by 1/c
***
*** author: Torsten.Bringmann.fys.uio.no
*** date  : 2018-02-18
**********************************************************************

      real*8 function dscrsource_line(pdg,n,power,v,egev,widthline,pdg2,istat)
```

41.3 vdSIDM/dd: Direct detection

41.3.1 Routine headers – fortran files

dsddgpgn.f

```
**********************************************************************
***  subroutine dsddgpgn returns DM nucleon four-fermion couplings  ***
***                                                                 ***
***  type : interface                                               ***
***         [NB: this routine will eventually be phased out as *interface*  ***
***              routine. Currently, neutrino telescope routines are the    ***
***              only routines in /src that access it. ]            ***
***                                                                 ***
***   output:                                                       ***
***     gg complex*16 array of couplings                             ***
***          first index is operator index (1:ddng), with ddng in dsddcom.h ***
***          second index is proton/neutron (1:2)                   ***
***     units: GeV^-2                                               ***
***                                                                 ***
*** author: Torsten.Bringmann.fys.uio.no                            ***
*** date  : 2018-02-18                                              ***
**********************************************************************
      subroutine dsddgpgn(gg,ierr)
```

41.4 vdSIDM/ge: General routines

41.4.1 Routine headers – fortran files

dsdmspin.f

```
**********************************************************************
*** Function dsdmspin returns the dark matter spin (in hbar)        ***
***                                                                 ***
***  type : interface                                               ***
***                                                                 ***
***  desc : dark matter spin                                        ***
***                                                                 ***
*** author: Torsten Bringmann 2018-02-18                            ***
**********************************************************************

      real*8 function dsdmspin()
```

dsmwimp.f

```
************************************************************************
*** Function dsmwimp returns the DM mass                             ***
***                                                                    ***
***  type : interface                                                  ***
***                                                                    ***
*** author: torsten.bringmann@fys.uio.no, 2014-05-09                   ***
************************************************************************

      real*8 function dsmwimp()
```

41.5 vdSIDM/ini: Model setup

Currently, the module has two concrete example models fully implemented, where DM is a massive Dirac fermion ψ_χ, DR is a massless fermion $\psi_{\tilde{\gamma}}$, and the mediator is either a vector V or a scalar ϕ. The interaction parts of the respective Lagrangians are thus given by

$$\Delta\mathcal{L}_{\rm vector} = g_\chi \bar{\psi}_\chi \not{V} \psi_\chi + g_{\tilde{\gamma}} \bar{\psi}_{\tilde{\gamma}} \not{V} \psi_{\tilde{\gamma}} \tag{41.6}$$

and

$$\Delta\mathcal{L}_{\rm scalar} = g_\chi \bar{\psi}_\chi \psi_\chi \phi + g_{\tilde{\gamma}} \bar{\psi}_{\tilde{\gamma}} \psi_{\tilde{\gamma}} \phi \,. \tag{41.7}$$

Calling dsgivemodel_vdSIDM_vector or dsgivemodel_vdSIDM_scalar sets the DM mass, mediator mass and couplings for these models after which, as usual, the model is set up by a call to dsmodelsetup. This routine calculates for example the total decay width of the mediator, for which we add the contributions from decay to DR and (for heavy mediators, if kinematically allowed) decay into DM particles.

41.5.1 Routine headers – fortran files

dsgivemodel_vdSIDM_scalar.f

```
************************************************************************
***  subroutine dsgivemodel_vdSIDM_scalar reads in parameters to descibe    ***
***  a simple model where a Dirac DM particle coupless to massive scalar    ***
***  mediators, which also couples (with the same strength) to massless     ***
***  dark radiation.                                                        ***
***  It then transfers these parameters to common blocks                    ***
***                                                                         ***
***  input:                                                                 ***
***                                                                         ***
***    mDM    - DM mass (in GeV)                                            ***
***    mmed   - scalar mediator mass (in GeV)                               ***
***    g      - DM-mediator coupling                                        ***
***    etaDM  - DM asymmetry = (n+ - n-)/s > 0                              ***
***                  (Note the conventional assumption about the sign!)     ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-05-22                                                        ***
***  mod 2022-10-19 (added etaDM as argument)                              ***
************************************************************************
      subroutine dsgivemodel_vdSIDM_scalar(mDM,mmed,g,etaDM)
```

dsgivemodel_vdSIDM_vector.f

```
************************************************************************
```

```
***  subroutine dsgivemodel_vdSIDM_vector reads in parameters to descibe    ***
***  a simple model where a Dirac DM particle coupless to massive vector    ***
***  mediators, which also couples (with the same strength) to massless     ***
***  dark radiation.                                                        ***
***  It then transfers these parameters to common blocks                    ***
***                                                                         ***
***  input:                                                                 ***
***                                                                         ***
***    mDM    - DM mass (in GeV)                                            ***
***    mmed   - vector mediator mass (in GeV)                               ***
***    g      - DM-mediator coupling                                        ***
***    etaDM  - DM asymmetry = (n+ - n-)/s > 0                              ***
***                  (Note the conventional assumption about the sign!)     ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2018-02-18                                                        ***
***  mod 2022-10-19 (added etaDM as argument)                               ***
*****************************************************************************
      subroutine dsgivemodel_vdSIDM_vector(mDM,mmed,g,etaDM)
```

dsinit_module.f

```
*****************************************************************************
***  This is the initialization subroutine for the "vdSIDM" module    ***
***  which just contains a simple WIMP model with a mass, annihilation      ***
***  branching fractions, cross sections etc.                               ***
***                                                                         ***
***  Author: Torsten Bringmann (torsten.bringmann@fys.uio.no)               ***
***  Date: 21/06/2016                                                       ***
*****************************************************************************
      subroutine dsinit_module
```

dsmodelsetup.f

```
*****************************************************************************
***  subroutine dsmodelsetup sets up a particle model                       ***
***                                                                         ***
***  type : interface                                                       ***
***                                                                         ***
***  desc : Sets up a new particle physics model                            ***
***                                                                         ***
***  output:                                                                ***
***    ierr  -  0 no errors                                                 ***
***          - <0 if an error is encountered (theoretically inconsistent    ***
***               model parameters)                                         ***
***    iwarn -  0 no warnings                                               ***
***                                                                         ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2016-05-09                                                        ***
***  mod 2022-10-19 (added aDM support)                                     ***
*****************************************************************************
      subroutine dsmodelsetup(ierr,iwarn)
```

41.6 vdSIDM/kd: Kinetic decoupling

DM particles are kept in local thermal equilibrium via scattering with the DR particles. This is dominated by t-channel exchange of mediator particles, for which we implement the momentum-

averaged scattering amplitude returned from Ref. [79]:

$$\left\langle |\mathcal{M}|^2 \right\rangle_t^{\text{vector}} = \frac{256}{3} g_\chi^2 g_{\tilde{\gamma}}^2 \left(\frac{m_\chi}{m_V}\right)^4 \left(\frac{\omega}{m_\chi}\right)^2 , \tag{41.8}$$

$$\left\langle |\mathcal{M}|^2 \right\rangle_t^{\text{scalar}} = \frac{512}{3} g_\chi^2 g_{\tilde{\gamma}}^2 \left(\frac{m_\chi}{m_\phi}\right)^4 \left(\frac{\omega}{m_\chi}\right)^2 , \tag{41.9}$$

where ω is the energy of the DR particle. We also add the amplitudes for DM scattering directly off heat bath vector mediators, $\left\langle |\mathcal{M}|^2 \right\rangle_t = 64 g_\chi^4/3$, and scalar mediators, $\left\langle |\mathcal{M}|^2 \right\rangle_t = 16 g_\chi^4/3$ [79]. The interface function dskdm2 returns the sum of these two contributions.

41.6.1 Routine headers – fortran files

dskdm2.f

```
************************************************************************
*** dskdm2 returns the full scattering amplitude squared, SUMMED over
*** both initial and final spin and other internal states, and then
*** divided by the DM internal degrees of freedom.
*** The returned value is not evaluated at zero momentum tranasfer, but
*** averaged over t.
***
***  type : INTERFACE
***  desc : Full scattering amplitude squared, averaged over momentum transfer
***
***  input: omega -- CMS MOMENTUM of scattering partner
***  output: omega -- CMS ENERGY of scattering partner
***
***  input:   Stype - SM scattering partners
***             "    = 10,11,12 - c,b,t quarks
***                    7,8,9 - d,u,s quarks
***                    4,5,6 - e,m,t leptons
***                    1,2,3 - e,m,t neutrinos
***
***           Stype  > 100 - non-SM scattering partners
***                         (NB: massless MUST be listed first!)
***             "    = 101 - DR particle
***             "    = 102 - mediator particle
***
*** author: torsten.bringmann@fys.uio.no, 2018-05-14
*** mod: 2022-12-05 bugfix (wrong paranthesis in converting k->omega)
************************************************************************

      real*8 function dskdm2(omega,Stype)
```

dskdm2simp.f

```
************************************************************************
*** dskdm2simp returns the scattering amplitude for DM at zero
*** momentum transfer squared in the vdSIDM model, SUMMED over both initial
*** and final spin and other internal states, and then divided by the DM
*** internal degrees of freedom. This is only valid in the limit of
*** relativistic scattering partners with small energies omega, where we
*** can expand as
***
***  type : INTERFACE
***  desc : Scattering amplitude squared, expanded in powers of energy
***
***  type : interface
***
***  input: Stype   - SM scattering partners:
```

```
***                   7,8,9 - u,d,s quarks
***                   4,5,6 - e,m,t leptons
***                   1,2,3 - e,m,t neutrinos
***
***         Stype    > 100 - non-SM scattering partners
***                        (NB: massless MUST be listed first!)
***             "    = 101 - DR particle
***             "    = 102 - mediator particle
***
*** author: torsten.bringmann@fys.uio.no, 2015-06-22
*** mod   : typo in amplitude [tb, 2020-06-22]
**********************************************************************

      subroutine dskdm2simp(Stype,cn,n)
```

dskdparticles.f

```
**************************************************************************
*** Prepares integration of Boltzmann equation and has to be called    ***
*** before dskdboltz (called by dskdtkd)                               ***
***                                                                    ***
***  type : interface                                                  ***
***                                                                    ***
*** author: Torsten.Bringmann.fys.uio.no                               ***
*** date  : 2015-06-11                                                 ***
**************************************************************************
      subroutine dskdparticles()
```

41.7 vdSIDM/rd: Relic density

In src_models/vdSIDM/rd we collect a set of routines to handle the temperature of the dark sector in this module. First of all, the interface function dsrdxi returns the temperature ratio $\xi = T_{\tilde{\gamma}}/T$, where T is the photon temperature. This is calculated by assuming that the two sectors were in thermal equilubrium ($\xi = 1$) at very early times when the DM particles were still relativistic, e.g. due to some portal that is effective only at very high energies and hence not relevant for the low-energy Lagrangian. After decoupling of the two sectors at a temperature $T_{\rm dc} \gg m_\chi$, entropy would be separately conserved, which implies that the temperature evolves as

$$\xi(T) = \frac{\left[g_*^{\rm SM}(T)/g_*^{\rm DS}(T)\right]^{\frac{1}{3}}}{\left[g_*^{\rm SM}(T_{\rm dc})/g_*^{\rm DS}(T_{\rm dc})\right]^{\frac{1}{3}}}\,, \tag{41.10}$$

where $g_*^{\rm SM,DS}$ are the entropy degrees of freedom in the visible and dark sector, respectively. Of course, dsrdxi could be replaced in the usual way by any user-supplied function in order to implement a different scenario for how the temperature of the dark sector evolves with time. Care must be taken, however, to ensure that such an implementation is consistent with the model assumptions. A constant value of ξ for example, as is often assumed for illustration, is *not* consistent with the assumption that there is no interaction between dark and visible sector.

Any additional relativistic energy density, as provided by the DR particles, is conventionally stated in terms of

$$\Delta N_{\rm eff} \equiv \frac{\rho_{\rm DS}}{\rho_{1\nu}} = \frac{4}{7} g_*^{\rm DS} \left(\frac{T_{\tilde{\gamma}}}{T_\nu}\right)^4\,, \tag{41.11}$$

where $\rho_{1\nu}$ is the energy density contributed by one massless neutrino species, and T_ν is the neutrino temperature (which differs from the photon temperature after $e^\pm$ annihilation). For convenience, DarkSUSY provides a function dsrddeltaneff to compute $\Delta N_{\rm eff}$ as a function of (photon) temperature.

41.7.1 Routine headers – fortran files

dsrddofDS.f

```
c----------------------------------------------------------------------
c  Function dsrddofDS returns the effective number of energy degrees of
c  freedom in the dark sector. For simplicity, it is assumed that all
c  dark sector particles are in thermal equilibirum with each other. In
c  this approximation, energy and entropy degrees of freedom are identical.
c
c  type : interface
c
c  desc : Relativistic BSM degrees of freedom
c
c  input:
c    Td - dark sector temperature [GeV]
c
c   NB: This specific implementation only applies to the vdSIDM module,
c       but it is of course straightforward to extend to other field
c       contents in the dark sector.
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2018-05-31
c======================================================================
      real*8 function dsrddofDS(Td)
```

dsrdparticles.f

```
      subroutine dsrdparticles(option,nsize,Tbath,
     &  selfcon,ncoann,mcoann,dof,nrs,rm,rw,nthr,tm)
**********************************************************************
*** subroutine dsrdparticles returns which particles are included in
***   the calculation of the WIMP relic density (coannihilations,
***   resonances, thresholds)
***
***  type : interface
***
***  desc : Particles included in relic density calculation
***  desc : (coannihilations, resonances and thresholds)
***
*** input:
***   option = 0 - default
***            !=0 - include various subsets of coannihilations
***   nsize = size of arrays, do not fill larger than this
***   Tbath = finite temperature of the heat bath
***            (not supported in module vdSIDM)
***
*** output
***    selfcon  - specifies whether DM is selfconjugate (1) or not (2)
***    ncoann   - number of particles coannihilating
***    mcoann   - relic and coannihilating mass in gev
***    dof      - internal degrees of freedom of the particles
***    nrs      - number of resonances to take special care of
***    rm       - mass of resonances in gev
***    rw       - width of resonances in gev
***    nt       - number of thresholds to take special care of
***               do not include coannihilation thresholds (that's automatic)
***    tm       - sqrt(s) of the thresholds in gev
*** This output is used by the RD routines in src/rd/
*** author: paolo gondolo
*** derived from dsrdomega
*** date: 13-10-04
*** Modified: Joakim Edsjo, edsjo@fysik.su.se, 2015-12-08
*** Modified: Torsten Bringmann 2016-06-28
*** Modified: Torsten Bringmann 2018-02-28 (added selfcon)
```

```
**********************************************************************
```

dsrdxi.f

```
c-----------------------------------------------------------------------
c  Function dsrdxi returns the temperature ratio for the temperature of
c  the heat bath that keeps DM in thermal equilibirum, Tdark, and the
c  photon temperature, Tphoton.
c
c  input:
c    x - mass/Tphoton
c
c  output:
c    xi = Tdark/Tphoton
c
c  NB: this particular version assumes that x=1 at very high temperatures,
c      when the DS decouples. In the vdSIDM module, this decoupling temperature
c      is set to 'infinity' in dsinit_module.
c      Further evolution after decoupling is only determined by how the
c      effective number of d.o.f. changes in the visible and dark sector,
c      respectively.
c
c  author: Torsten Bringmann (torsten.bringmann@fys.uio.no), 2018-06-01
c  mod: 2018-08-17 (TB, added decoupling temperature as free parameter)
c=======================================================================
      real*8 function dsrdxi(x)
```

41.8 vdSIDM/se_yield: Yields from annihilation in the Sun/Earth

41.8.1 Routine headers – fortran files

dsseyield.f

```
*************************************************************************
*** Function dsseyield calculates the yield of neutrinos, leptons or
*** hadronic showers from WIMP annihilations in the Sun/Earth summing
*** over all annihilation channels.
***
***  type : interface
***
*** Inputs:
***   emu   - energy of neutrino, lepton or hadronic shower (GeV)
***   theta - angle from the Sun / centre of the Earth (degrees)
***   wh    - 'su' for Sun, 'ea' for Earth
***   kind  - 1 = integrated
***           2 = differential
***           3 = mixed, integrated in theta, differential in energy
***   type  - Type of yield
***
***   type   Yield at detector
***   ----   -----------------
***   1      nu_e
***   2      nu_e-bar
***   3      nu_mu
***   4      nu_mu-bar
***   5      nu_tau
***   6      nu_tau-bar
***   7      e- at neutrino-nucleon vertex
***   8      e+ at neutrino-nucleon vertex
```

```
***   9      mu- at neutrino-nucleon vertex
***   10     mu+ at neutrino-nucleon vertex
***   11     tau- at neutrino-nucleon vertex
***   12     tau+ at neutrino-nucleon vertex
***   13     mu- at an imaginary plane in detector (i.e. after propagation)
***   14     mu+ at an imaginary plane in detector (i.e. after propagation)
***   15     hadronic shower from nu_e charged current (CC) interactions
***   16     hadronic shower from nu_e-bar charged current (CC) interactions
***   17     hadronic shower from nu_mu charged current (CC) interactions
***   18     hadronic shower from nu_mu-bar charged current (CC) interactions
***   19     hadronic shower from nu_tau charged current (CC) interactions
***   20     hadronic shower from nu_tau-bar charged current (CC) interactions
***   21     hadronic shower from nu_e neutral current (NC) interactions
***   22     hadronic shower from nu_e-bar neutral current (NC) interactions
***   23     hadronic shower from nu_mu neutral current (NC) interactions
***   24     hadronic shower from nu_mu-bar neutral current (NC) interactions
***   25     hadronic shower from nu_tau neutral current (NC) interactions
***   26     hadronic shower from nu_tau-bar neutral current (NC) interactions
***
*** Outputs:
***   yield in units of
***      1e-30 m**-2 (annihilation)**-1 for types 1-6 and 13-14
***      1e-30 m**-3 (annihilation)**-1 for types 7-12, 15-26.
***      For the differential yields, the units are the same plus
***      GeV**-1 degree**-1.
***   istat - status flag (non-zero if something went wrong)
*** NOTE: Compared to previous versions of this routine, particles and
*** antiparticles are no longer summed, you thus need to call it for both
*** types and add them up.
***
*** Author: Torsten.Bringmann@fys.uio.no
*** Date: 2018-02-18
**************************************************************************

      real*8 function dsseyield(e,theta,wh,kind,type,istat)
```

41.9 vdSIDM/si: Dark matter self-interactions

The interface function dssisigtm returns the transfer cross section σ_T for a Yukawa potential, by implementing the explicit expressions for the various scattering regimes discussed in detail in Section 32.1.

41.9.1 Routine headers – fortran files

dssisigtm.f

```
**************************************************************************
*** Function dssisigtm provides the momentum-transfer cross section per DM  ***
*** mass, in conventional units of cm**2/g.                                 ***
***                                                                        ***
***  type : interface                                                      ***
***  desc : momentum-transfer cross section                                ***
***                                                                        ***
***  Input:                                                                ***
***    vkms - relative velocity of scattering DM particles (in km/s)       ***
***                                                                        ***
***  This interfacte function is required by dssisigtmav in src/.          ***
***                                                                        ***
*** author: Torsten.Bringmann.fys.uio.no                                   ***
*** date 2015-05-18                                                        ***
```

```
*******************************************************************************
      real*8 function dssisigtm(vkms)
```

Part IV

Example programs

Chapter 42

examples: examples

42.1 Routine headers – fortran files

dsmain_decay.F

No header found.

dsmain_wimp.F

No header found.

Chapter 43

examples/test: examples/test

43.1 Routine headers – fortran files

dstest_genFIMP.f

No header found.

dstest_genWIMP.f

No header found.

dstest_mssm.f

No header found.

dstest_silveira_zee.f

No header found.

dstest_vdSIDM.f

No header found.

Chapter 44

examples/aux: examples/aux

44.1 Routine headers – fortran files

caprates.f

```
************************************************************************
*** Program to show how capture rates can be calculated.
*** Compared to caprates_ff.f, this program uses the tabulated capture
*** rates. It scans over a set of masses.
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: February, 2018
************************************************************************
```

caprates_ff.f

```
************************************************************************
*** Program to show how capture rates can be calculated.
*** This program is a bit more advanced than caprates.f as it goes into
*** interal workings of the routines to select exactly which elements to
*** include. It is provided to give an example on how to use routines
*** in more advanced way to get detailed information.
***
*** This program is intended to be used with the generic_wimp module
*** The program scans a few masses and calculates the total capture rate, and
*** capture rates on individual elements.
*** Compared to the default calculation in dsmain_wimp and dstest, this
*** program does not use tabulated capture rates, instead it always uses
*** the full numerical calculation (with numerical integration over the
*** velocity, radius and momentum transfer and sum over all the chosen
*** elements) to be able to choose different options
*** and choice of elements to include. Hence, the program takes quite a long
*** time to run (typically several hours in the default setup).
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: February, 2018
************************************************************************
```

DD_example.f

```
No header found.
```

DDCR_flux.f

```
No header found.
```

DDCR_limits.f

```
No header found.
```

DMhalo_bypass.f

```
**********************************************************************
*** program DMhalo_bypass shows how to provide your own hardcoded DM source,
*** bypassing the DM halo profile setting provided in DS.
***
*** WARNING: while simpler than DMhalo_new.f, this is still an example
*** for 'expert users'! In particular, the drawback of this approach
*** is that the system of automatic tabulation of quantities related
*** to DM rates cannot be easily exploited
***
*** Concretely, we demonstrate below how the DS subroutine dsdmddriver
*** must be overwritten. The new hardcoded DM source is in this
*** example provided by the function my_dm_sphsource further down
*** (and for this specific demonstration example chosen to coincide
*** with the NFW case). The example provided here assumes the hardcoded
*** halo is of temporary kind and hence no tabulations are loaded.
***
*** For testing purposes, this program reads in a file DMhalo_bypass_sav.dat
*** that contains the same set of cosmic ray fluxes computed with the
*** NFW profile as provided in the standard DS setup (this file is
*** compute with DMhalo5.f)
**********************************************************************
```

DMhalo_bypass_prep.f

```
**********************************************************************
*** program DMhalo_bypass_prep.f generates the test file needed for
*** DMhalo_bypass.f. Concretely, it computes the same DM rates as in
*** DMhalo_bypass.f -- but for the DS default halo model 'mwnfwdef'
*** provided with the DS release (rather than for the hardcoded DM
*** source as provided in DMhalo_bypass.f).
*** NOTE: we need to write a separate main
*** file since the constructions are alternative one to the other.
**********************************************************************
```

DMhalo_los.f

```
No header found.
```

DMhalo_new.f

```
**********************************************************************
*** program DMhalo_new shows how to a define a new parametric DM density
*** profile and make it consistently available to the halo driver routines
*** and hence all DS functionality connected to halo density profiles.
***
*** WARNING: unlike the example files DMhalo_predef.f and DMhalo_table.f,
*** this is an example for 'expert users'!
***
*** Concretely, we demonstrate how to still use the default dsdmddriver
*** and just add one extra option, for the example of the Zhao
*** ('alpha-beta-gamma') profile. For that purpose one needs to
```

```
*** define a new driver, namely dsdmddriver_abg specific for the Zhao
*** profile, and overwrite the subroutine dsdmddriver_choice
**********************************************************************
```

DMhalo_predef.f

```
**********************************************************************
*** The program DMhalo_predef demonstrates initialization and basic usage of
*** the DM density profiles already pre-defined in parametric forms in
*** the DS release. Quick reading guide:
***
*** line 55 (117, 137) ff : initialization of NFW (Burkert, Einasto) profile
*** line 153 ff : compute Jfactors and gamma-ray fluxes
*** line 217 ff : demonstrate how to scan over halo parameters
*** line 249 ff : reaidng out profile values for given halo
*** line 279 ff : example on how to re-define input parameters to
***               characterize a halo (here: local vs. scale density of NFW)
*** line 322 ff : auxiliary routines needed by main programme
***
**********************************************************************
```

DMhalo_table.f

```
**********************************************************************
*** Program DMhalo_table shows how to load a tabulated DM density profile
*** (radius vs density). In this example, the profile is spherical, and
*** interpolated with cubic splines or linear interpolation, in linear or
*** logarithmic scales. The program also contains examples about how to call
*** rates for Milky Way cosmic ray fluxes for (such) temporary profiles
*** Quick reading guide:
***
*** line 58 ff  : define (NFW-like) halo model based on table from file
***               + sample check against standard implemented (NFW) halo
*** line 122 ff : define (Burkert-like) halo model based on table
***               + sample check against standard implemented (Burkert) halo
*** line 197 ff : detailed examples for how to compute CR fluxes for first
***               numerical halo model
***               + comparison to standard implemented (NFW) halo
*** line 366 ff : Short version for how to compute CR fluxes for second
***               numerical halo model
***               + comparison to standard implemented (Burkert) halo
***
**********************************************************************
```

flxconv.f

```
No header found.
```

flxconvplot.f

```
No header found.
```

FreezeIn_generic_fimp.f

```
No header found.
```

FreezeIn_ScalarSinglet.f

```
No header found.
```

neutrinospectra.f

```
**********************************************************************
*** neutrinospectra. This program calculates yields in neutrino
*** telescopes for a generic WIMP and compares spectra with old
*** Pythia 6 and new Pythia 8 simulation results.
***
*** Output: nuspectrum.dat, file with energy and Pythia 6 and 8 yields
***
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: March 17, 2022
**********************************************************************
```

neutrinoyields.f

```
**********************************************************************
*** neutrinoyields. This program calculates yields in neutrino
*** telescopes for a generic WIMP.
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: June 24, 2021
**********************************************************************
```

oh2_aDM.f

```
**********************************************************************
*** program oh2_aDM loops over the mass of an asymmetric DM candidate
*** to find the annihilation cross section needed to get the correct
*** relic density (for DM + anti-DM).
*** Here we assume a constant annihilation cross section <sv> into
*** a given final states, and various values for the initial asymmetry
*** \eta = (n_+ - n-) / s.
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
***         (based on oh2_generic_wimp by J. Edsjö)
*** Date: 26/07/2022
**********************************************************************
```

oh2_cBE_ScalarSinglet.f

```
No header found.
```

oh2_dark_sector.f

```
**********************************************************************
*** program oh2_dark_sector performs relic density calculations in
*** a decoupled dark sector, as described in 2007.03696. The model is
*** thus defined by a Majorana or Dirac DM candidate, coupled to gS
*** relativistic scalar degrees of freedem.
*** The program loops over the dark matter mass and finds the 'thermal'
*** annihilation cross section that gives a relic density in accordance
*** with the Planck observations (2018 data, within 3 sigma limits).
***
*** Slightly modified version of program oh2_generic_wimp.
***
*** This program demonstrates how a specific dark sector module (vdSIDM)
*** can be effectively turned into a 'generic' dark sector model
*** by just replacing two functions of that module (the new versions for
*** dsanwx and dsgivemodel_vdSIDM_scalar are located in user_replaceables.
*** Note that the former is in the file dsanwx_dark_sector.f, because the
```

```
*** file dsanwx.f already contains a replaceable version for the program
*** oh2_generic_wimp -- see makefile target ).
***
***
*** Author: Torsten Bringmann, torsten.bringmann@fys.uio.no
*** Date: October, 2020
**********************************************************************
```

oh2_generic_wimp.f

```
**********************************************************************
*** program oh2_generic_wimp loops over WIMP mass and finds the
*** annihilation cross sections (for the chosen annihilation channel)
*** that give a good relic density (within the Planck 2015, 3 sigma
*** limits)
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: July 1, 2016
**********************************************************************
```

oh2_ScalarSinglet.f

```
No header found.
```

oh2_vdSIDM.f

```
No header found.
```

ScalarSinglet_thermal_averages.f

```
No header found.
```

ucmh_test.f

```
!DarkSUSY example program to calculate things about UCMHs
!
!Authors: Pat Scott
!          pscott@imperial.ac.uk
!         Madeleine Anthonisen
!          maddyanthonisen@hotmail.com
!Date: Oct 2014 - April 2015
```

wimpyields.f

```
**********************************************************************
*** wimpyields. This program calculates various stable particle yields
*** (positrons, neutrinos, gamma-rays,...) resulting from final states
*** of DM annihilation or decay. It is thus an example of how to use
*** DarkSUSY without relying on a specific SUSY model (instead linking
*** to the generic WIMP model).
*** Author: Joakim Edsjo, edsjo@fysik.su.se
*** Date: October 25, 2017
*** mod: May 27, 2020 -- added example for how to compare different
***                      versions of tabulated yields [TB]
**********************************************************************
```

Bibliography

[1] P. Gondolo, J. Edsjö, P. Ullio, L. Bergström, M. Schelke and E. A. Baltz, *DarkSUSY: Computing supersymmetric dark matter properties numerically*, *JCAP* **0407** (2004) 008, [astro-ph/0406204].

[2] T. Bringmann, J. Edsjö, P. Gondolo, P. Ullio and L. Bergström, *DarkSUSY 6: An Advanced Tool to Compute Dark Matter Properties Numerically*, .

[3] H. Zhao, *Analytical models for galactic nuclei*, *Mon. Not. Roy. Astron. Soc.* **278** (1996) 488–496, [astro-ph/9509122].

[4] V. Silveira and A. Zee, *SCALAR PHANTOMS*, *Phys. Lett.* **B161** (1985) 136–140.

[5] J. McDonald, *Gauge singlet scalars as cold dark matter*, *Phys. Rev.* **D50** (1994) 3637–3649, [hep-ph/0702143].

[6] C. P. Burgess, M. Pospelov and T. ter Veldhuis, *The Minimal model of nonbaryonic dark matter: A Singlet scalar*, *Nucl. Phys.* **B619** (2001) 709–728, [hep-ph/0011335].

[7] J. M. Cline, K. Kainulainen, P. Scott and C. Weniger, *Update on scalar singlet dark matter*, *Phys. Rev.* **D88** (2013) 055025, [1306.4710].

[8] J. Einasto, *On the Construction of a Composite Model for the Galaxy and on the Determination of the System of Galactic Parameters*, *Trudy Astrofizicheskogo Instituta Alma-Ata* **5** (1965) 87–100.

[9] J. F. Navarro, C. S. Frenk and S. D. M. White, *The Structure of cold dark matter halos*, *Astrophys. J.* **462** (1996) 563–575, [astro-ph/9508025].

[10] T. Sjöstrand, S. Mrenna and P. Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026, [hep-ph/0603175].

[11] M. Cirelli, G. Corcella, A. Hektor, G. Hutsi, M. Kadastik, P. Panci et al., *PPPC 4 DM ID: A Poor Particle Physicist Cookbook for Dark Matter Indirect Detection*, *JCAP* **1103** (2011) 051, [1012.4515].

[12] P. Gondolo and G. Gelmini, *Cosmic abundances of stable particles: Improved analysis*, *Nucl. Phys.* **B360** (1991) 145–179.

[13] J. Edsjö and P. Gondolo, *Neutralino relic density including coannihilations*, *Phys. Rev.* **D56** (1997) 1879–1894, [hep-ph/9704361].

[14] J. Edsjö, M. Schelke, P. Ullio and P. Gondolo, *Accurate relic densities with neutralino, chargino and sfermion coannihilations in mSUGRA*, *JCAP* **0304** (2003) 001, [hep-ph/0301106].

[15] T. Binder, T. Bringmann, M. Gustafsson and A. Hryczuk, *DRAKE: Dark matter Relic Abundance beyond Kinetic Equilibrium*, *Eur. Phys. J. C* **81** (2021) 577, [2103.01944].

[16] T. Bringmann, S. Heeba, F. Kahlhoefer and K. Vangsnes, *Freezing-in a hot bath: resonances, medium effects and phase transitions*, 2111.14871.

[17] T. Bringmann, *Particle Models and the Small-Scale Structure of Dark Matter*, *New J. Phys.* **11** (2009) 105027, [0903.0189].

[18] L. Bergström, J. Edsjö and P. Ullio, *Possible indications of a clumpy dark matter halo*, *Phys. Rev.* **D58** (1998) 083507, [astro-ph/9804050].

[19] L. Bergström, J. Edsjö and P. Gondolo, *Indirect detection of dark matter in km size neutrino telescopes*, *Phys. Rev.* **D58** (1998) 103519, [hep-ph/9806293].

[20] E. A. Baltz and J. Edsjo, *Positron propagation and fluxes from neutralino annihilation in the halo*, *Phys. Rev.* **D59** (1998) 023511, [astro-ph/9808243].

[21] L. Bergström, J. Edsjö and P. Ullio, *Cosmic anti-protons as a probe for supersymmetric dark matter?*, *Astrophys. J.* **526** (1999) 215–235, [astro-ph/9902012].

[22] T. Bringmann, A. J. Galea and P. Walia, *Leading QCD Corrections for Indirect Dark Matter Searches: a Fresh Look*, *Phys. Rev.* **D93** (2016) 043529, [1510.02473].

[23] T. Bringmann and M. Pospelov, *Novel direct detection constraints on light dark matter*, *Phys. Rev. Lett.* **122** (2019) 171801, [1810.10543].

[24] K. Bondarenko, A. Boyarsky, T. Bringmann, M. Hufnagel, K. Schmidt-Hoberg and A. Sokolenko, *Direct detection and complementary constraints for sub-GeV dark matter*, *JHEP* **03** (2020) 118, [1909.08632].

[25] L. Bergström and P. Gondolo, *Limits on direct detection of neutralino dark matter from b —> s gamma decays*, *Astropart. Phys.* **5** (1996) 263–278, [hep-ph/9510252].

[26] L. Bergström and P. Ullio, *Full one loop calculation of neutralino annihilation into two photons*, *Nucl. Phys.* **B504** (1997) 27–44, [hep-ph/9706232].

[27] P. Ullio and L. Bergström, *Neutralino annihilation into a photon and a Z boson*, *Phys. Rev.* **D57** (1998) 1962–1971, [hep-ph/9707333].

[28] T. Bringmann, L. Bergström and J. Edsjö, *New Gamma-Ray Contributions to Supersymmetric Dark Matter Annihilation*, *JHEP* **01** (2008) 049, [0710.3169].

[29] L. Bergström, T. Bringmann and J. Edsjö, *New Positron Spectral Features from Supersymmetric Dark Matter - a Way to Explain the PAMELA Data?*, *Phys. Rev.* **D78** (2008) 103520, [0808.3725].

[30] T. Bringmann and F. Calore, *Significant Enhancement of Neutralino Dark Matter Annihilation from Electroweak Bremsstrahlung*, *Phys. Rev. Lett.* **112** (2014) 071301, [1308.1089].

[31] T. Bringmann, F. Calore, A. Galea and M. Garny, *Electroweak and Higgs Boson Internal Bremsstrahlung: General considerations for Majorana dark matter annihilation and application to MSSM neutralinos*, *JHEP* **09** (2017) 041, [1705.03466].

[32] L. Bergström, T. Bringmann, I. Cholis, D. Hooper and C. Weniger, *New limits on dark matter annihilation from AMS cosmic ray positron data*, *Phys. Rev. Lett.* **111** (2013) 171101, [1306.3983].

[33] A. W. Strong, I. V. Moskalenko and V. S. Ptuskin, *Cosmic-ray propagation and interactions in the Galaxy*, *Ann. Rev. Nucl. Part. Sci.* **57** (2007) 285–327, [astro-ph/0701517].

[34] A. W. Strong and I. V. Moskalenko, *Propagation of cosmic-ray nucleons in the galaxy*, *Astrophys. J.* **509** (1998) 212–228, [astro-ph/9807150].

[35] C. Evoli, D. Gaggero, A. Vittino, G. Di Bernardo, M. Di Mauro, A. Ligorini et al., *Cosmic-ray propagation with DRAGON2: I. numerical solver and astrophysical ingredients*, *JCAP* **1702** (2017) 015, [1607.07886].

[36] R. Kissmann, *PICARD: A novel code for the Galactic Cosmic Ray propagation problem*, *Astropart. Phys.* **55** (2014) 37–50, [1401.4035].

[37] D. Maurin, F. Donato, R. Taillet and P. Salati, *Cosmic rays below z=30 in a diffusion model: new constraints on propagation parameters*, *Astrophys. J.* **555** (2001) 585–596, [astro-ph/0101231].

[38] C. Evoli, I. Cholis, D. Grasso, L. Maccione and P. Ullio, *Antiprotons from dark matter annihilation in the Galaxy: astrophysical uncertainties*, *Phys. Rev.* **D85** (2012) 123511, [1108.0664].

[39] S. Colafrancesco, S. Profumo and P. Ullio, *Multi-frequency analysis of neutralino dark matter annihilations in the Coma cluster*, *Astron. Astrophys.* **455** (2006) 21, [astro-ph/0507575].

[40] M. S. Turner, *Probing the Structure of the Galactic Halo with gamma Rays Produced by WIMP Annihilations*, *Phys. Rev.* **D34** (1986) 1921.

[41] J. R. Ipser and P. Sikivie, *Estimates of the Density of Dark Matter Near the Center of the Galaxy*, *Phys. Rev.* **D35** (1987) 3695.

[42] K. Freese and J. Silk, *Halo gamma-rays from cold dark matter annihilation*, *Phys. Rev.* **D40** (1989) 3828–3833.

[43] V. Berezinsky, A. Bottino and G. Mignola, *High-energy gamma radiation from the galactic center due to neutralino annihilation*, *Phys. Lett.* **B325** (1994) 136–142, [hep-ph/9402215].

[44] G. Lake, *Detectability of gamma-rays from clumps of dark matter*, *Nature* **346** (1990) 39–40.

[45] J. Silk and A. Stebbins, *Clumpy cold dark matter*, *Astrophys. J.* **411** (1993) 439–449.

[46] C. Calcaneo-Roldan and B. Moore, *The Surface brightness of dark matter: Unique signatures of neutralino annihilation in the galactic halo*, *Phys. Rev.* **D62** (2000) 123005, [astro-ph/0010056].

[47] L. Bergström, J. Edsjö, P. Gondolo and P. Ullio, *Clumpy neutralino dark matter*, *Phys. Rev.* **D59** (1999) 043506, [astro-ph/9806072].

[48] L. Bergström, J. Edsjö and C. Gunnarsson, *Neutralino gamma-ray signals from accreting halo dark matter*, *Phys. Rev.* **D63** (2001) 083515, [astro-ph/0012346].

[49] E. A. Baltz, C. Briot, P. Salati, R. Taillet and J. Silk, *Detection of neutralino annihilation photons from external galaxies*, *Phys. Rev.* **D61** (2000) 023514, [astro-ph/9909112].

[50] D. B. Cline and Y.-T. Gao, *Low-energy cosmic γ-rays from cosmological photino annihilation*, *Astron. Astrophys.* **231** (1990) L23–L26.

[51] Y.-T. Gao, F. W. Stecker and D. B. Cline, *The Lightest supersymmetric particle and the extragalactic gamma-ray background*, *Astron. Astrophys.* **249** (1991) 1–4.

[52] L. Bergström, J. Edsjö and P. Ullio, *Spectral gamma-ray signatures of cosmological dark matter annihilation*, *Phys. Rev. Lett.* **87** (2001) 251301, [astro-ph/0105048].

[53] G. Belanger, K. Kannike, A. Pukhov and M. Raidal, *Z_3 Scalar Singlet Dark Matter*, *JCAP* **1301** (2013) 022, [1211.1014].

[54] G. Bélanger, K. Kannike, A. Pukhov and M. Raidal, *Minimal semi-annihilating $\mathbb{Z}_N$ scalar dark matter*, *JCAP* **1406** (2014) 021, [1403.4960].

[55] P. Ko and Y. Tang, *Self-interacting scalar dark matter with local Z_3 symmetry*, *JCAP* **1405** (2014) 047, [1402.6449].

[56] S.-M. Choi and H. M. Lee, *SIMP dark matter with gauged Z_3 symmetry*, *JHEP* **09** (2015) 063, [1505.00960].

[57] C. Arina, T. Bringmann, J. Silk and M. Vollmann, *Enhanced Line Signals from Annihilating Kaluza-Klein Dark Matter*, *Phys. Rev.* **D90** (2014) 083506, [1409.0007].

[58] S. Campbell, B. Dutta and E. Komatsu, *Effects of Velocity-Dependent Dark Matter Annihilation on the Energy Spectrum of the Extragalactic Gamma-ray Background*, *Phys. Rev.* **D82** (2010) 095007, [1009.3530].

[59] M. Lattanzi and J. I. Silk, *Can the WIMP annihilation boost factor be boosted by the Sommerfeld enhancement?*, *Phys. Rev.* **D79** (2009) 083523, [0812.0360].

[60] A. L. Fitzpatrick, W. Haxton, E. Katz, N. Lubbers and Y. Xu, *The Effective Field Theory of Dark Matter Direct Detection*, *JCAP* **1302** (2013) 004, [1203.3542].

[61] P. Gondolo and S. Scopel, , *in prep.* (2018) .

[62] J. Alvey, T. Bringmann and H. Kolesova, *No room to hide: implications of cosmic-ray upscattering for GeV-scale dark matter*, 2209.03360.

[63] XENON collaboration, E. Aprile et al., *Dark Matter Search Results from a One Ton-Year Exposure of XENON1T*, *Phys. Rev. Lett.* **121** (2018) 111302, [1805.12562].

[64] Borexino collaboration, G. Alimonti et al., *Science and technology of BOREXINO: A Real time detector for low-energy solar neutrinos*, *Astropart. Phys.* **16** (2002) 205–234, [hep-ex/0012030].

[65] B. Dasgupta and J. F. Beacom, *Reconstruction of supernova ν_μ, ν_τ, anti-ν_μ, and anti-ν_τ neutrino spectra at scintillator detectors*, *Phys. Rev.* **D83** (2011) 113006, [1103.2768].

[66] A. V. Maccio', A. A. Dutton and F. C. v. d. Bosch, *Concentration, Spin and Shape of Dark Matter Haloes as a Function of the Cosmological Model: WMAP1, WMAP3 and WMAP5 results*, *Mon. Not. Roy. Astron. Soc.* **391** (2008) 1940–1954, [0805.1926].

[67] J. Schaye et al., *The EAGLE project: Simulating the evolution and assembly of galaxies and their environments*, *Mon. Not. Roy. Astron. Soc.* **446** (2015) 521–554, [1407.7040].

[68] M. Schaller, C. S. Frenk, R. G. Bower, T. Theuns, A. Jenkins, J. Schaye et al., *Baryon effects on the internal structure of ΛCDM haloes in the EAGLE simulations*, *Mon. Not. Roy. Astron. Soc.* **451** (2015) 1247–1267, [1409.8617].

[69] L. Wang, A. A. Dutton, G. S. Stinson, A. V. Macciò, C. Penzo, X. Kang et al., *NIHAO project – I. Reproducing the inefficiency of galaxy formation across cosmic time with a large sample of cosmological hydrodynamical simulations*, *Mon. Not. Roy. Astron. Soc.* **454** (2015) 83–94, [1503.04818].

[70] E. Tollet et al., *NIHAO – IV: core creation and destruction in dark matter density profiles across cosmic time*, *Mon. Not. Roy. Astron. Soc.* **456** (2016) 3542–3552, [1507.03590].

[71] A. Burkert, *The Structure of dark matter halos in dwarf galaxies*, *IAU Symp.* **171** (1996) 175, [astro-ph/9504041].

[72] A. S. Eddington, *The dynamics of a stellar system. Third paper: oblate and other distributions*, *MNRAS* **76** (Nov., 1915) 37.

[73] R. Catena and P. Ullio, *The local dark matter phase-space density and impact on WIMP direct detection*, *JCAP* **05** (2012) 005, [1111.3556].

[74] L. J. Hall, K. Jedamzik, J. March-Russell and S. M. West, *Freeze-In Production of FIMP Dark Matter*, *JHEP* **03** (2010) 080, [0911.1120].

[75] X. Chu, T. Hambye and M. H. G. Tytgat, *The Four Basic Ways of Creating Dark Matter Through a Portal*, *JCAP* **05** (2012) 034, [1112.0493].

[76] X. Chu, Y. Mambrini, J. Quevillon and B. Zaldivar, *Thermal and non-thermal production of dark matter via Z'-portal(s)*, *JCAP* **01** (2014) 034, [1306.4677].

[77] T. Bringmann and S. Hofmann, *Thermal decoupling of WIMPs from first principles*, *JCAP* **0704** (2007) 016, [hep-ph/0612238].

[78] T. Binder, T. Bringmann, M. Gustafsson and A. Hryczuk, *Early kinetic decoupling of dark matter: when the standard way of calculating the thermal relic density fails*, *Phys. Rev.* **D96** (2017) 115010, [1706.07433].

[79] T. Bringmann, H. T. Ihle, J. Kersten and P. Walia, *Suppressing structure formation at dwarf galaxy scales and below: late kinetic decoupling as a compelling alternative to warm dark matter*, *Phys. Rev.* **D94** (2016) 103529, [1603.04884].

[80] J. Kasahara, *Neutralino Dark Matter: The Mass of the smallest Halo and the golden Region*, Ph.D. thesis, University of Utah, 2009.

[81] P. Gondolo, J. Hisano and K. Kadota, *The Effect of quark interactions on dark matter kinetic decoupling and the mass of the smallest dark halos*, *Phys. Rev.* **D86** (2012) 083523, [1205.1914].

[82] A. M. Green, S. Hofmann and D. J. Schwarz, *The First wimpy halos*, *JCAP* **0508** (2005) 003, [astro-ph/0503387].

[83] A. Loeb and M. Zaldarriaga, *The Small-scale power spectrum of cold dark matter*, *Phys. Rev.* **D71** (2005) 103520, [astro-ph/0504112].

[84] E. Bertschinger, *The Effects of Cold Dark Matter Decoupling and Pair Annihilation on Cosmological Perturbations*, *Phys. Rev.* **D74** (2006) 063509, [astro-ph/0607319].

[85] J. Diemand, B. Moore and J. Stadel, *Earth-mass dark-matter haloes as the first structures in the early Universe*, *Nature* **433** (2005) 389–391, [astro-ph/0501589].

[86] K. Griest and D. Seckel, *Three exceptions in the calculation of relic abundances*, *Phys. Rev.* **D43** (1991) 3191–3203.

[87] M. Srednicki, R. Watkins and K. A. Olive, *Calculations of Relic Densities in the Early Universe*, *Nucl. Phys.* **B310** (1988) 693.

[88] M. Drees, F. Hajkarim and E. R. Schmitz, *The Effects of QCD Equation of State on the Relic Density of WIMP Dark Matter*, *JCAP* **1506** (2015) 025, [1503.03513].

[89] T. Bringmann, P. F. Depta, M. Hufnagel and K. Schmidt-Hoberg, *Precise dark matter relic abundance in decoupled sectors*, *Phys. Lett. B* **817** (2021) 136341, [2007.03696].

[90] T. Bringmann and J. Edsjö, *DarkSUSY 6.3 – Freeze-in, out-of-equilibrium freeze-out, cosmic-ray upscattering and further new features*, *PoS* **CompTools2021** (2022) 038, [2203.07439].

[91] J. Bernstein, *KINETIC THEORY IN THE EXPANDING UNIVERSE*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, Cambridge, U.K., 1988, 10.1017/CBO9780511564185.

[92] E. W. Kolb and M. S. Turner, *The Early Universe*, *Front. Phys.* **69** (1990) 1–547.

[93] L. G. van den Aarssen, T. Bringmann and Y. C. Goedecke, *Thermal decoupling and the smallest subhalo mass in dark matter models with Sommerfeld-enhanced annihilation rates*, *Phys. Rev.* **D85** (2012) 123512, [1202.5456].

[94] L. M. Krauss.

[95] W. H. Press and D. N. Spergel, *Capture by the sun of a galactic population of weakly interacting massive particles*, *Astrophys. J.* **296** (1985) 679–684.

[96] J. Silk, K. A. Olive and M. Srednicki, *The Photino, the Sun and High-Energy Neutrinos*, *Phys. Rev. Lett.* **55** (1985) 257–259.

[97] L. M. Krauss, K. Freese, W. Press and D. Spergel, *Cold dark matter candidates and the solar neutrino problem*, *Astrophys. J.* **299** (1985) 1001.

[98] L. M. Krauss, M. Srednicki and F. Wilczek, *Solar System Constraints and Signatures for Dark Matter Candidates*, *Phys. Rev.* **D33** (1986) 2079–2083.

[99] T. K. Gaisser, G. Steigman and S. Tilav, *Limits on Cold Dark Matter Candidates from Deep Underground Detectors*, *Phys. Rev.* **D34** (1986) 2206.

[100] K. Griest and D. Seckel, *Cosmic Asymmetry, Neutrinos and the Sun*, *Nucl. Phys.* **B283** (1987) 681–705.

[101] J. S. Hagelin, K. W. Ng and K. A. Olive, *A High-energy Neutrino Signature From Supersymmetric Relics*, *Phys. Lett.* **B180** (1986) 375–380.

[102] K. Freese, *Can Scalar Neutrinos Or Massive Dirac Neutrinos Be the Missing Mass?*, *Phys. Lett.* **167B** (1986) 295–300.

[103] M. Kamionkowski, *Energetic neutrinos from heavy neutralino annihilation in the sun*, *Phys. Rev.* **D44** (1991) 3021–3042.

[104] F. Halzen, T. Stelzer and M. Kamionkowski, *Signatures of dark matter in underground detectors*, *Phys. Rev.* **D45** (1992) 4439–4442.

[105] A. Bottino, V. de Alfaro, N. Fornengo, G. Mignola and M. Pignone, *Indirect search for neutralinos at neutrino telescopes*, *Phys. Lett.* **B265** (1991) 57–63.

[106] A. Bottino, N. Fornengo, G. Mignola and L. Moscoso, *Signals of neutralino dark matter from earth and sun*, *Astropart. Phys.* **3** (1995) 65–76, [hep-ph/9408391].

[107] R. Gandhi, J. L. Lopez, D. V. Nanopoulos, K.-j. Yuan and A. Zichichi, *Scrutinizing supergravity models through neutrino telescopes*, *Phys. Rev.* **D49** (1994) 3691–3703, [astro-ph/9309048].

[108] L. Bergström, J. Edsjö and P. Gondolo, *Indirect neutralino detection rates in neutrino telescopes*, *Phys. Rev.* **D55** (1997) 1765–1770, [hep-ph/9607237].

[109] F. Halzen, *Status of neutrino astronomy: The Quest for kilometer scale instruments*, *Comments Nucl. Part. Phys.* **22** (1997) 155–170, [astro-ph/9701029].

[110] S. Ritz and D. Seckel, *Detailed Neutrino Spectra From Cold Dark Matter Annihilations in the Sun*, *Nucl. Phys.* **B304** (1988) 877–908.

[111] G. F. Giudice and E. Roulet, *Energetic Neutrinos From Supersymmetric Dark Matter*, *Nucl. Phys.* **B316** (1989) 429–442.

[112] M. Drees, G. Jungman, M. Kamionkowski and M. M. Nojiri, *Neutralino annihilation into gluons*, *Phys. Rev.* **D49** (1994) 636–647, [hep-ph/9306325].

[113] G. Jungman and M. Kamionkowski, *Neutrinos from particle decay in the sun and earth*, *Phys. Rev.* **D51** (1995) 328–340, [hep-ph/9407351].

[114] V. Berezinsky, A. Bottino, J. R. Ellis, N. Fornengo, G. Mignola and S. Scopel, *Searching for relic neutralinos using neutrino telescopes*, *Astropart. Phys.* **5** (1996) 333–352, [hep-ph/9603342].

[115] M. Kamionkowski, K. Griest, G. Jungman and B. Sadoulet, *Model independent comparison of direct versus indirect detection of supersymmetric dark matter*, *Phys. Rev. Lett.* **74** (1995) 5174–5177, [hep-ph/9412213].

[116] J. Edsjö and P. Gondolo, *WIMP mass determination with neutrino telescopes*, *Phys. Lett.* **B357** (1995) 595–601, [hep-ph/9504283].

[117] L. Bergström, J. Edsjö and M. Kamionkowski, *Astrophysical neutrino detection with angular and energy resolution*, *Astropart. Phys.* **7** (1997) 147–160, [astro-ph/9702037].

[118] G. Jungman, M. Kamionkowski and K. Griest, *Supersymmetric dark matter*, *Phys. Rept.* **267** (1996) 195–373, [hep-ph/9506380].

[119] A. Gould, *Resonant Enhancements in WIMP Capture by the Earth*, *Astrophys. J.* **321** (1987) 571.

[120] J. N. Bahcall, M. H. Pinsonneault and S. Basu, *Solar models: Current epoch and time dependences, neutrinos, and helioseismological properties*, *Astrophys. J.* **555** (2001) 990–1012, [astro-ph/0010346].

[121] W. F. McDonough, *Compositional Model for the Earth's Core*, *Treatise on Geochemistry* **2** (Dec., 2003) 568.

[122] *The Earth: its properties, composition, and structure.* Encyclopædia Britannica, Inc., 1999.

[123] A. Gould, *Gravitational diffusion of solar system WIMPs*, *Astrophys. J.* **368** (Feb., 1991) 610–615.

[124] P. Farinella, C. Froeschlé, C. Froeschlé, R. Gonczi, G. Hahn, A. Morbidelli et al., *Asteroids falling into the Sun*, *Nature* **371** (Sept., 1994) 314–317.

[125] A. Gould and S. M. Khairul Alam, *Can heavy WIMPs be captured by the earth?*, *Astrophys. J.* **549** (2001) 72–75, [astro-ph/9911288].

[126] J. Lundberg and J. Edsjö, *WIMP diffusion in the solar system including solar depletion and its effect on earth capture rates*, *Phys. Rev.* **D69** (2004) 123505, [astro-ph/0401113].

[127] J. Edsjö, "WimpSim Neutrino Monte Carlo." `http://www.fysik.su.se/~edsjo/wimpsim/`, 2007.

[128] J. Edsjö, *Aspects of neutrino detection of neutralino dark matter*, Ph.D. thesis, Uppsala U., 1997. `hep-ph/9704384`.

[129] J. Edsjö, "NuSigma Neutrino Interaction Monte Carlo." `http://www.fysik.su.se/~edsjo/wimpsim/`, 2007.

[130] Particle Data Group collaboration, D. E. Groom et al., *Review of particle physics. Particle Data Group*, *Eur. Phys. J.* **C15** (2000) 1–878.

[131] M. Blennow, J. Edsjö and T. Ohlsson, *Neutrinos from WIMP annihilations using a full three-flavor Monte Carlo*, *JCAP* **0801** (2008) 021, [0709.3898].

[132] D. N. Spergel and P. J. Steinhardt, *Observational evidence for selfinteracting cold dark matter*, *Phys. Rev. Lett.* **84** (2000) 3760–3763, [astro-ph/9909386].

[133] A. Loeb and N. Weiner, *Cores in Dwarf Galaxies from Dark Matter with a Yukawa Potential*, *Phys. Rev. Lett.* **106** (2011) 171302, [1011.6374].

[134] M. Vogelsberger, J. Zavala and A. Loeb, *Subhaloes in Self-Interacting Galactic Dark Matter Haloes*, *Mon. Not. Roy. Astron. Soc.* **423** (2012) 3740, [1201.5892].

[135] A. H. G. Peter, M. Rocha, J. S. Bullock and M. Kaplinghat, *Cosmological Simulations with Self-Interacting Dark Matter II: Halo Shapes vs. Observations*, *Mon. Not. Roy. Astron. Soc.* **430** (2013) 105, [1208.3026].

[136] J. Zavala, M. Vogelsberger and M. G. Walker, *Constraining Self-Interacting Dark Matter with the Milky Way's dwarf spheroidals*, *Mon. Not. Roy. Astron. Soc.* **431** (2013) L20–L24, [1211.6426].

[137] L. G. van den Aarssen, T. Bringmann and C. Pfrommer, *Is dark matter with long-range interactions a solution to all small-scale problems of ΛCDM cosmology?*, *Phys. Rev. Lett.* **109** (2012) 231301, [1205.5809].

[138] O. D. Elbert, J. S. Bullock, S. Garrison-Kimmel, M. Rocha, J. Onorbe and A. H. G. Peter, *Core formation in dwarf haloes with self-interacting dark matter: no fine-tuning necessary*, *Mon. Not. Roy. Astron. Soc.* **453** (2015) 29–37, [1412.1477].

[139] A. Kamada, M. Kaplinghat, A. B. Pace and H.-B. Yu, *How the Self-Interacting Dark Matter Model Explains the Diverse Galactic Rotation Curves*, *Phys. Rev. Lett.* **119** (2017) 111102, [1611.02716].

[140] A. Robertson et al., *The diverse density profiles of galaxy clusters with self-interacting dark matter plus baryons*, *Mon. Not. Roy. Astron. Soc.* **476** (2018) L20–L24, [1711.09096].

[141] J. S. Bullock and M. Boylan-Kolchin, *Small-Scale Challenges to the ΛCDM Paradigm*, *Ann. Rev. Astron. Astrophys.* **55** (2017) 343–387, [1707.04256].

[142] R. A. Flores and J. R. Primack, *Observational and theoretical constraints on singular dark matter halos*, *Astrophys. J.* **427** (1994) L1–4, [astro-ph/9402004].

[143] B. Moore, *Evidence against dissipationless dark matter from observations of galaxy haloes*, *Nature* **370** (1994) 629.

[144] M. Boylan-Kolchin, J. S. Bullock and M. Kaplinghat, *Too big to fail? The puzzling darkness of massive Milky Way subhaloes*, *Mon. Not. Roy. Astron. Soc.* **415** (2011) L40, [1103.0007].

[145] M. Boylan-Kolchin, J. S. Bullock and M. Kaplinghat, *The Milky Way's bright satellites as an apparent failure of LCDM*, *Mon. Not. Roy. Astron. Soc.* **422** (2012) 1203–1218, [1111.2048].

[146] K. A. Oman et al., *The unexpected diversity of dwarf galaxy rotation curves*, *Mon. Not. Roy. Astron. Soc.* **452** (2015) 3650–3665, [1504.01437].

[147] K. A. Oman, J. F. Navarro, L. V. Sales, A. Fattahi, C. S. Frenk, T. Sawala et al., *Missing dark matter in dwarf galaxies?*, *Mon. Not. Roy. Astron. Soc.* **460** (2016) 3610–3623, [1601.01026].

[148] B. Moore, S. Ghigna, F. Governato, G. Lake, T. R. Quinn, J. Stadel et al., *Dark matter substructure within galactic halos*, *Astrophys. J.* **524** (1999) L19–L22, [astro-ph/9907411].

[149] A. A. Klypin, A. V. Kravtsov, O. Valenzuela and F. Prada, *Where are the missing Galactic satellites?*, *Astrophys. J.* **522** (1999) 82–92, [astro-ph/9901240].

[150] S. Tulin, H.-B. Yu and K. M. Zurek, *Beyond Collisionless Dark Matter: Particle Physics Dynamics for Dark Matter Halo Structure*, *Phys. Rev.* **D87** (2013) 115007, [1302.3898].

[151] F. Kahlhoefer, K. Schmidt-Hoberg and S. Wild, *Dark matter self-interactions from a general spin-0 mediator*, *JCAP* **1708** (2017) 003, [1704.02149].

[152] S. Tulin and H.-B. Yu, *Dark Matter Self-interactions and Small Scale Structure*, *Phys. Rept.* **730** (2018) 1–57, [1705.02358].

[153] J. L. Feng, M. Kaplinghat and H.-B. Yu, *Halo Shape and Relic Density Exclusions of Sommerfeld-Enhanced Dark Matter Explanations of Cosmic Ray Excesses*, *Phys. Rev. Lett.* **104** (2010) 151301, [0911.0422].

[154] F.-Y. Cyr-Racine, K. Sigurdson, J. Zavala, T. Bringmann, M. Vogelsberger and C. Pfrommer, *ETHOS,Äîan effective theory of structure formation: From dark particle physics to the matter distribution of the Universe*, *Phys. Rev.* **D93** (2016) 123527, [1512.05344].

[155] S. A. Khrapak, A. V. Ivlev, G. E. Morfill and S. K. Zhdanov, *Scattering in the Attractive Yukawa Potential in the Limit of Strong Interaction*, *Phys. Rev. Lett.* **90** (2003) 225002.

[156] S. A. Khrapak, A. V. Ivlev and G. E. Morfill, *Momentum transfer in complex plasmas*, *Phys. Rev. E* **70** (Nov, 2004) 056405.

[157] Y. L. Luke, *The Special Functions and Their Approximations*, vol. II of *Mathematics in Science and Engineering, Volume 53-II, Editor: Richard Bellman.* 1969.

[158] K. G. Chetyrkin, J. H. Kuhn and M. Steinhauser, *RunDec: A Mathematica package for running and decoupling of the strong coupling and quark masses*, *Comput. Phys. Commun.* **133** (2000) 43–65, [hep-ph/0004189].

[159] H. E. Haber and G. L. Kane, *The Search for Supersymmetry: Probing Physics Beyond the Standard Model*, *Phys. Rept.* **117** (1985) 75–263.

[160] J. F. Gunion and H. E. Haber, *Higgs Bosons in Supersymmetric Models. 1.*, *Nucl. Phys.* **B272** (1986) 1.

[161] F. E. Paige, S. D. Protopopescu, H. Baer and X. Tata, *ISAJET 7.69: A Monte Carlo event generator for pp, anti-p p, and e+e- reactions*, hep-ph/0312045.

[162] http://www.phy.bnl.gov/~isajet.

[163] P. Z. Skands et al., *SUSY Les Houches accord: Interfacing SUSY spectrum calculators, decay packages, and event generators*, *JHEP* **07** (2004) 036, [hep-ph/0311123].

[164] B. C. Allanach et al., *SUSY Les Houches Accord 2*, *Comput. Phys. Commun.* **180** (2009) 8–25, [0801.0045].

[165] M. Drees, M. M. Nojiri, D. P. Roy and Y. Yamada, *Light Higgsino dark matter*, *Phys. Rev.* **D56** (1997) 276–290, [hep-ph/9701219].

[166] D. Pierce and A. Papadopoulos, *Radiative corrections to neutralino and chargino masses in the minimal supersymmetric model*, *Phys. Rev.* **D50** (1994) 565–570, [hep-ph/9312248].

[167] A. B. Lahanas, K. Tamvakis and N. D. Tracas, *One loop corrections to the neutralino sector and radiative electroweak breaking in the MSSM*, *Phys. Lett.* **B324** (1994) 387–396, [hep-ph/9312251].

[168] S. Heinemeyer, W. Hollik and G. Weiglein, *FeynHiggs: A Program for the calculation of the masses of the neutral CP even Higgs bosons in the MSSM*, *Comput. Phys. Commun.* **124** (2000) 76–89, [hep-ph/9812320].

[169] http://www.feynhiggs.de.

[170] S. Heinemeyer, W. Hollik and G. Weiglein, *QCD corrections to the masses of the neutral CP - even Higgs bosons in the MSSM*, *Phys. Rev.* **D58** (1998) 091701, [hep-ph/9803277].

[171] S. Heinemeyer, W. Hollik and G. Weiglein, *The Masses of the neutral CP - even Higgs bosons in the MSSM: Accurate analysis at the two loop level*, *Eur. Phys. J.* **C9** (1999) 343–366, [hep-ph/9812472].

[172] S. Heinemeyer, W. Hollik and G. Weiglein, *The Mass of the lightest MSSM Higgs boson: A Compact analytical expression at the two loop level*, *Phys. Lett.* **B455** (1999) 179–191, [hep-ph/9903404].

[173] T. Bringmann et al., *DarkBit: A GAMBIT module for computing dark matter observables and likelihoods*, *Eur. Phys. J.* **C77** (2017) 831, [1705.07920].

[174] GAMBIT collaboration, C. Balázs et al., *ColliderBit: a GAMBIT module for the calculation of high-energy collider observables and likelihoods*, *Eur. Phys. J.* **C77** (2017) 795, [1705.07919].

[175] P. Bechtle, O. Brein, S. Heinemeyer, G. Weiglein and K. E. Williams, *HiggsBounds: Confronting Arbitrary Higgs Sectors with Exclusion Bounds from LEP and the Tevatron*, *Comput. Phys. Commun.* **181** (2010) 138–167, [0811.4169].

[176] ATLAS collaboration, G. Aad et al., *Search for squarks and gluinos with the ATLAS detector in final states with jets and missing transverse momentum using* $\sqrt{s} = 8$ *TeV proton–proton collision data*, *JHEP* **09** (2014) 176, [1405.7875].

[177] F. Mahmoudi, *SuperIso: A Program for calculating the isospin asymmetry of B —> K* gamma in the MSSM*, *Comput. Phys. Commun.* **178** (2008) 745–754, [0710.2067].

[178] The GAMBIT Flavour Workgroup collaboration, F. U. Bernlochner et al., *FlavBit: A GAMBIT module for computing flavour observables and likelihoods*, *Eur. Phys. J.* **C77** (2017) 786, [1705.07933].

[179] BaBar collaboration, J. P. Lees et al., *Exclusive Measurements of* $b \to s\gamma$ *Transition Rate and Photon Energy Spectrum*, *Phys. Rev.* **D86** (2012) 052012, [1207.2520].

[180] BaBar collaboration, J. P. Lees et al., *Precision Measurement of the $B \to X_s\gamma$ Photon Energy Spectrum, Branching Fraction, and Direct CP Asymmetry $A_{CP}(B \to X_{s+d}\gamma)$*, *Phys. Rev. Lett.* **109** (2012) 191801, [1207.2690].

[181] Belle collaboration, A. Abdesselam et al., *Measurement of the inclusive $B \to X_{s+d}\gamma$ branching fraction, photon energy spectrum and HQE parameters*, in *Proceedings, 38th International Conference on High Energy Physics (ICHEP 2016): Chicago, IL, USA, August 3-10, 2016*, 2016, 1608.02344, https://inspirehep.net/record/1479946/files/arXiv:1608.02344.pdf.

[182] LHCb collaboration, R. Aaij et al., *Measurement of the $B_s^0 \to \mu^+\mu^-$ branching fraction and effective lifetime and search for $B^0 \to \mu^+\mu^-$ decays*, *Phys. Rev. Lett.* **118** (2017) 191801, [1703.05747].

[183] T. Moroi, *The Muon anomalous magnetic dipole moment in the minimal supersymmetric standard model*, *Phys. Rev.* **D53** (1996) 6565–6575, [hep-ph/9512396].

[184] Muon g-2 collaboration, G. W. Bennett et al., *Final Report of the Muon E821 Anomalous Magnetic Moment Measurement at BNL*, *Phys. Rev.* **D73** (2006) 072003, [hep-ex/0602035].

[185] GAMBIT collaboration, P. Athron et al., *SpecBit, DecayBit and PrecisionBit: GAMBIT modules for computing mass spectra, particle decay rates and precision observables*, 1705.07936.

[186] A. C. Hearn, *Reduce 3.5*, 1993.

[187] J. Edsjö, "form2f, perl script to convert form output to fortran."

[188] J. Hisano, S. Matsumoto and M. M. Nojiri, *Explosive dark matter annihilation*, *Phys. Rev. Lett.* **92** (2004) 031303, [hep-ph/0307216].

[189] J. Hisano, S. Matsumoto, M. M. Nojiri and O. Saito, *Non-perturbative effect on dark matter annihilation and gamma ray signature from galactic center*, *Phys. Rev.* **D71** (2005) 063528, [hep-ph/0412403].

[190] A. Birkedal, K. T. Matchev, M. Perelstein and A. Spray, *Robust gamma ray signature of WIMP dark matter*, hep-ph/0507194.

[191] L. Bergström, T. Bringmann, M. Eriksson and M. Gustafsson, *Gamma rays from Kaluza-Klein dark matter*, *Phys. Rev. Lett.* **94** (2005) 131301, [astro-ph/0410359].

[192] L. Bergström, *Radiative Processes in Dark Matter Photino Annihilation*, *Phys. Lett.* **B225** (1989) 372–380.

[193] L. Bergström, T. Bringmann, M. Eriksson and M. Gustafsson, *Gamma rays from heavy neutralino dark matter*, *Phys. Rev. Lett.* **95** (2005) 241301, [hep-ph/0507229].

[194] M. W. Goodman and E. Witten, *Detectability of Certain Dark Matter Candidates*, *Phys. Rev.* **D31** (1985) 3059.

[195] A. Bottino, F. Donato, G. Mignola, S. Scopel, P. Belli and A. Incicchitti, *Exploring the supersymmetric parameter space by direct search for WIMPs*, *Phys. Lett.* **B402** (1997) 113–121, [hep-ph/9612451].

[196] J. R. Ellis and R. A. Flores, *Realistic Predictions for the Detection of Supersymmetric Dark Matter*, *Nucl. Phys.* **B307** (1988) 883–908.

[197] J. R. Ellis and R. A. Flores, *Elastic supersymmetric relic - nucleus scattering revisited*, *Phys. Lett.* **B263** (1991) 259–266.

[198] J. Engel, *Nuclear form-factors for the scattering of weakly interacting massive particles*, *Phys. Lett.* **B264** (1991) 114–119.

[199] K. Griest, *Cross-Sections, Relic Abundance and Detection Rates for Neutralino Dark Matter*, *Phys. Rev.* **D38** (1988) 2357.

[200] R. Barbieri, M. Frigeni and G. F. Giudice, *Dark Matter Neutralinos in Supergravity Theories*, *Nucl. Phys.* **B313** (1989) 725–735.

[201] G. B. Gelmini, P. Gondolo and E. Roulet, *Neutralino dark matter searches*, *Nucl. Phys.* **B351** (1991) 623–644.

[202] A. Bottino, V. de Alfaro, N. Fornengo, S. Mignola and S. Scopel, *On the neutralino as dark matter candidate. 2. Direct detection.*, *Astropart. Phys.* **2** (1994) 77–90, [hep-ph/9309219].

[203] J. Gasser, H. Leutwyler and M. E. Sainio, *Sigma term update*, *Phys. Lett.* **B253** (1991) 252–259.

[204] Spin Muon collaboration, D. Adams et al., *A New measurement of the spin dependent structure function g1(x) of the deuteron*, *Phys. Lett.* **B357** (1995) 248–254.

[205] R. L. Jaffe and A. Manohar, *The G(1) Problem: Fact and Fantasy on the Spin of the Proton*, *Nucl. Phys.* **B337** (1990) 509–546.

[206] J. Engel and P. Vogel, *Spin dependent cross-sections of weakly interacting massive particles on nuclei*, *Phys. Rev.* **D40** (1989) 3132–3135.

[207] H. E. Haber and D. Wyler, *RADIATIVE NEUTRALINO DECAY*, *Nucl. Phys.* **B323** (1989) 267–310.

[208] S. Dimopoulos and S. D. Thomas, *Dynamical relaxation of the supersymmetric CP violating phases*, *Nucl. Phys.* **B465** (1996) 23–33, [hep-ph/9510220].

[209] J. R. Ellis, G. Ridolfi and F. Zwirner, *Radiative corrections to the masses of supersymmetric Higgs bosons*, *Phys. Lett.* **B257** (1991) 83–91.

[210] J. R. Ellis, G. Ridolfi and F. Zwirner, *On radiative corrections to supersymmetric Higgs boson masses and their implications for LEP searches*, *Phys. Lett.* **B262** (1991) 477–484.

[211] A. Brignole, J. R. Ellis, G. Ridolfi and F. Zwirner, *The Supersymmetric charged Higgs boson mass and LEP phenomenology*, *Phys. Lett.* **B271** (1991) 123–132.

[212] M. Drees and M. M. Nojiri, *One loop corrections to the Higgs sector in minimal supergravity models*, *Phys. Rev.* **D45** (1992) 2482–2492.

[213] M. Carena, J. R. Espinosa, M. Quiros and C. E. M. Wagner, *Analytical expressions for radiatively corrected Higgs masses and couplings in the MSSM*, *Phys. Lett.* **B355** (1995) 209–221, [hep-ph/9504316].

[214] M. Carena, M. Quiros and C. E. M. Wagner, *Effective potential methods and the Higgs mass spectrum in the MSSM*, *Nucl. Phys.* **B461** (1996) 407–436, [hep-ph/9508343].

[215] T. Hahn, S. Heinemeyer, W. Hollik, H. Rzehak and G. Weiglein, *FeynHiggs 2.7*, *Nucl. Phys. Proc. Suppl.* **205-206** (2010) 152–157, [1007.0956].

[216] Particle Data Group collaboration, C. Patrignani et al., *Review of Particle Physics*, *Chin. Phys.* **C40** (2016) 100001.

[217] H. E. Haber, *Perspectives in Higgs Physics II*. World Scientific, 1997.

[218] T. Hahn, *SUSY Les Houches Accord 2 I/O made easy*, *Comput. Phys. Commun.* **180** (2009) 1681–1693, [hep-ph/0605049].

[219] S. Cassel, *Sommerfeld factor for arbitrary partial wave processes*, *J. Phys.* **G37** (2010) 105009, [0903.5307].