

DarkSUSY 6

Tutorial – Part II–III

Joakim Edsjö
edsjo@fysik.su.se

With Torsten Bringmann, Paolo Gondolo, Piero Ullio and
Lars Bergström



Stockholm
University

KMI School, Nagoya
2018-02-28 – 2018-03-02



Osaka Klein
centre

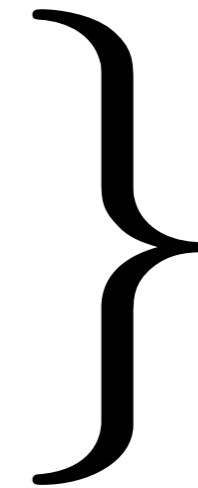
Follow-up of yesterday's question on what you want to learn

- Using DarkSUSY
 - *will cover*
- Non-thermal production
 - *the production is not included in DS, but you can still use it to calculate various rates*
- Calculations for direct detection
 - *I have added an example on this for today's tutorial*
- Good parameter choices
 - *best option is to look at paper discussing benchmarks, providing e.g. SLHA files*
- Rates from 100 TeV WIMP
 - *you cannot currently do this as we cannot run Pythia for such massive WIMPs*
- How do you apply astrophysics constraints?
 - *DarkSUSY focuses on calculating rates, you have to compare with data on your own. It is a good idea to have an example of this though (not in this tutorial though)*
- Can we modify the basic equations, like the Boltzmann equation?
 - *you can e.g. change the degrees of freedom, more advanced changes would require you to change internal routines*
- Calling DarkSUSY from C++?
 - *you can call DarkSUSY from C++, I will look into providing an example on how to do this (not in the tutorial yet)*
- $\overline{\nu}_e(\nu_e)$
 - *OK, I guess...*

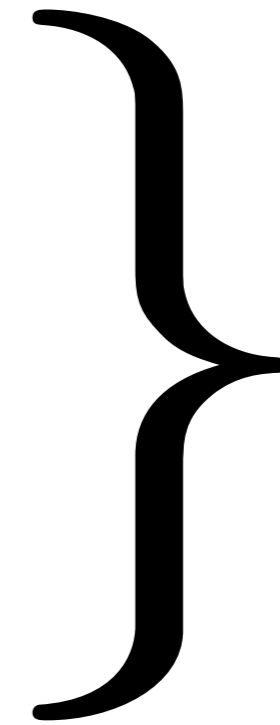
The tutorial (and later examples/tutorials) will be made available on the DarkSUSY web page darksusy.org

Outline of hands-on

1. dstest program
2. dsmain_wimp program
 - MSSM
 - generic WIMP
3. Writing your own programs and using makefiles in DarkSUSY 6
4. Using dsmain_wimp with SLHA files
5. Other example programs
6. Replaceable functions
7. Direct detection example
8. Creating a new particle physics module



Part I



Part II+III



3. Makefiles and writing your own code

Makefiles

- The way we choose which particle physics module to use is when we build our main program, e.g.

```
gfortran -o dsmain_wimp dsmain_wimp.F -lds_core.a -lds_mssm.a
```

- This can be made more flexible with makefiles,

```
dscheckmod :
    test `ls ../lib/ | grep libds_${DS_MODULE}.a` || { echo ERROR: Module $
{DS_MODULE} does not exist, or is not compiled; exit 1;}

dsmain_wimp : DS_MODULE = $(shell sed -n '1p' dsmain_wimp.driver)

dsmain_wimp : dscheckmod makefile dsmain_wimp.F
    printf "#define MODULE_CONFIG MODULE_"$(DS_MODULE)"\n" > module_compile.F
    printf "$(LIB)/libds_core_user.a\n"$(LIB)" /libds_core.a\n"$(LIB)" /libds_"$
(DS_MODULE)"_user.a\n"$(LIB)" /libds_"$(DS_MODULE)".a" > module_link.txt
    $(ADD_SCR) libds_tmp.a module_link.txt
    $(FF) $(FOPT) $(INC) $(INC_MSSM) -L$(LIB) -o dsmain_wimp dsmain_wimp.F \
libds_tmp.a $(shell if [ "x$(DS_MODULE)" = "xmssm" ]; then printf "%s" " $
(AUX_LIB_MSSM)"; fi)
    rm -f module_compile.F
    rm -f module_link.txt
    rm -f libds_tmp.a
```

Some details of dsmain_wimp.F

- In dsmain_wimp we have code blocks of this type

```
#if MODULE_CONFIG == MODULE_generic_wimp
  subroutine dspmterparameters
    [more code for this module]
#endif
```

- This is how dsmain_wimp.F performs model-specific setup.
- We could as well have prepared one separate main program for each particle physics module if we preferred (the makefile is then a bit simpler as well, see e.g. examples/aux/makefile)

Starting points for your own programs

- `dsmain_wimp.F` is a good starting point for your own program. Other good starting points are the example programs in `examples/aux`. If you want to use any of these as a starting point, either

Option A (preferred)

- make a copy out of it
- put it in your own private folder
- copy `examples/makefile` to your private folder and modify it to your liking
- make in your private folder and run your code

A) keeps your own routines separate from the DarkSUSY routines, makes updates easier

Option B

- make a copy out of it
- modify `examples/makefile.in` to copy-paste the lines about `dsmain_wimp.F` and modify to your liking
- run `./configure` in the DS root
- make and run

B) is not the best way as it will make it harder for you to upgrade to new DarkSUSY versions

Task: Make your own program

- Make your own program to setup an MSSM model
- Let it calculate e.g. the relic density
- Make sure it links to the correct libraries and compiles
- Run it!

Task: Make your own program

- Make your own program to setup an MSSM model
- Let it calculate e.g. the relic density
- Make sure it links to the correct libraries and compiles
- Run it!

```
program myprogram
implicit none

include 'dsver.h'
real*8 oh2,xf
integer unphys,war,ierr,iwar,nfc
real*8 dsrdomega

call dsinit          mu      M2      MA      tan(beta)
call dsgive_model(500.0d0,1000.d0,300.d0,10.d0,
& 3000.d0,0.d0,0.d0)
call dsmodelsetup(unphys,war)
oh2=dsrdomega(1,1,xf,ierr,iwar,nfc)
Ab/m0 write(*,*) 'Relic density, omega h^2 = ',oh2
At/m0
end
```

Hints:

- Make your own folder, e.g. examples/own where you put your program
- Write your main program (e.g. by following the example on Part I, slide 7, see above)
- Copy examples/aux/makefile to own/makefile and setup build instructions for your code, e.g. by copying the build lines for flxconv and replace flxconv with your program name
- Make your program

Dropbox link: <https://www.dropbox.com/s/3m8bmlky7eyzs00/myprog.f?dl=0>

Output

```
Initialization of particle physics module MSSM complete.  
Initialization of DarkSUSY complete.
```

```
-----  
FeynHiggs 2.13.0  
built on Feb 27, 2018  
H. Bahl, T. Hahn, S. Heinemeyer, W. Hollik, S. Passehr, H. Rzehak, G. Weiglein  
http://feynhiggs.de  
-----
```

```
FHHiggsCorr contains code by:  
P. Slavich et al. (2L rMSSM Higgs self-energies)  
H. Rzehak et al. (2L cMSSM asat Higgs self-energies)  
S. Passehr et al. (2L cMSSM atat Higgs self-energies)  
Relic density,  $\omega h^2 = 4.7486651906502690E-002$ 
```

Task: generic WIMP model

- Do the same exercise, i.e. write your own main program to setup a generic WIMP model and calculate e.g. the relic density
- You can e.g. choose a generic WIMP with the following parameters
 - mass: 100 GeV
 - selfconjugate=.true. (is its own antiparticle)
 - annihilation cross section, $3 \times 10^{-26} \text{ cm}^3 \text{ s}^{-1}$
 - annihilation channel, b b-bar, pdg code 5
 - scattering cross section (SI): 10^{-7} pb

Hints:

- set up your makefile as in the previous example, changing module to generic_wimp
- look in src_models/generic_wimp/ini/dsgivemodel_generic_wimp how to set up the model (a logical parameter is expressed as .true. or .false. in Fortran)

Dropbox link: <https://www.dropbox.com/s/ovpm1v61j3qnf76/mygenprog.f?dl=0>

Output

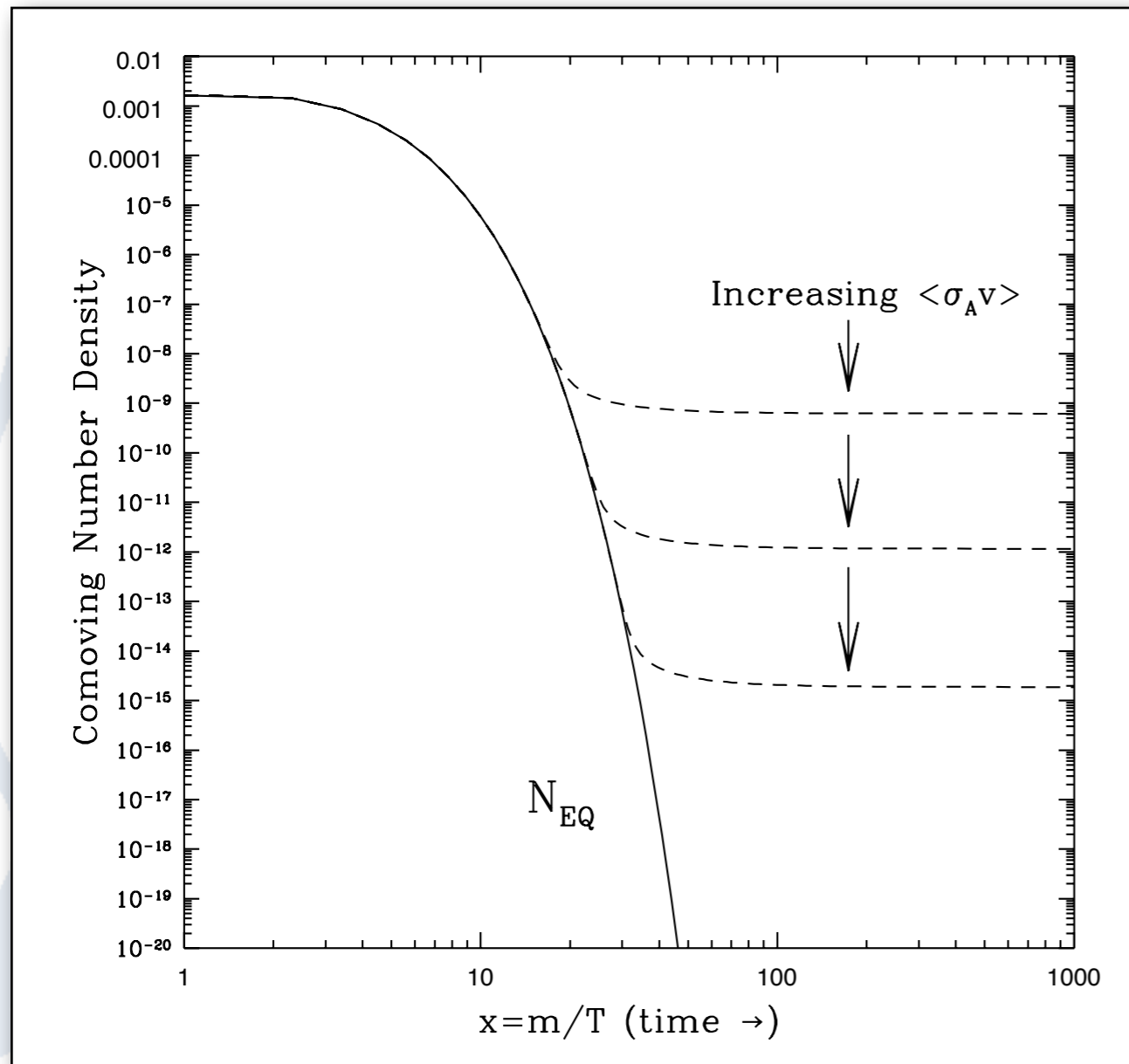
```
*****  
*** Welcome to DarkSUSY version  
*** darksusy-6.0.0  
*****  
  
Initializing native DarkSUSY SM routines...  
Initialization of particle physics module generic_wimp complete.  
Initialization of DarkSUSY complete.  
  
Relic density,  $\omega h^2 = 8.3169607208332816E-002$ 
```

This relic density was obtained for an annihilation cross section of $3 \times 10^{-26} \text{ cm}^3 \text{ s}^{-1}$. It is a bit too low compared to the Planck measurement of 0.1193 ± 0.0028 (2σ). [arXiv:1502.01589]

Question: Should we increase or decrease the annihilation cross section to get closer to the Planck measurement?

Relic density

simple approach (more advanced in real life)



- We got a *too low* relic density
- To get closer to the Planck measurement, we need freeze-out to occur *earlier*
- \Rightarrow We need a *lower* annihilation cross section

Comment on WIMP models

- There are in principle two distinct classes of WIMP models:
 - **Concrete models** based on a theoretical framework where the WIMP is embedded in a bigger theory, e.g. SUSY – **predictive, theoretically motivated**
 - **Ad hoc models** where one invents a WIMP with properties that could explain some data, e.g. inelastic dark matter, exciting DM, generic WIMP, etc – **simpler, phenomenological framework**

4. Using dsmain_wimp.F with SLHA files

- For SUSY, you can use either the built in spectrum calculator in DS (pMSSM) or isasugra (cMSSM), interfaced in DS. These are accessible in dsmain_wimp
- You can also use a stand-alone spectrum calculator that outputs SLHA files and feed these into DS

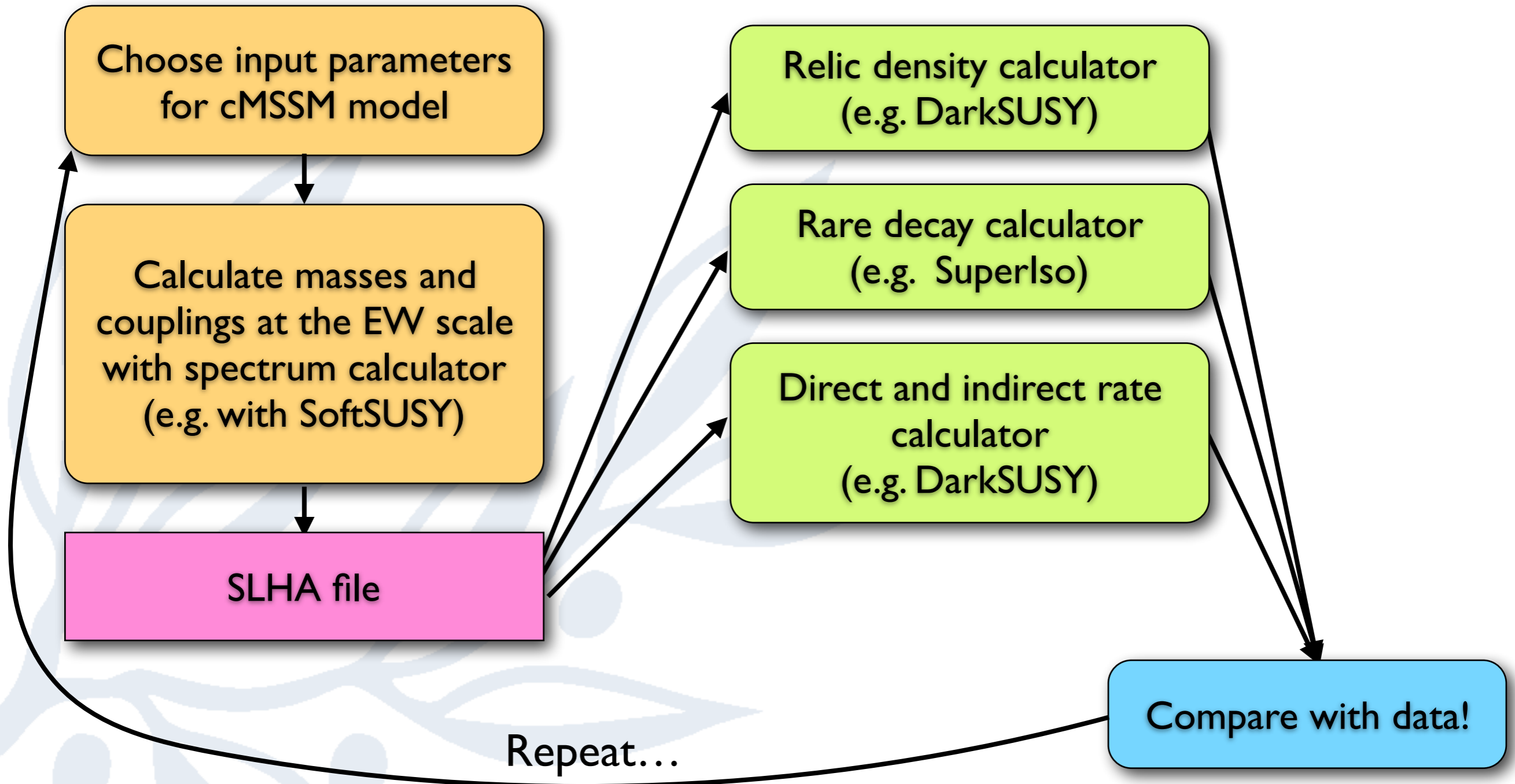
SLHA files

- SLHA files contain blocks
- Typically a spectrum calculator reads input blocks (e.g. parameters at the GUT scale) and returns a new SLHA file with output blocks filled in (e.g. masses and couplings at the EW scale).
- Other codes can read the files and add new blocks, e.g. particle decay tables
- DM codes typically read SLHA files, but does not output new SLHA files (except for the particle physics part of the calculation)

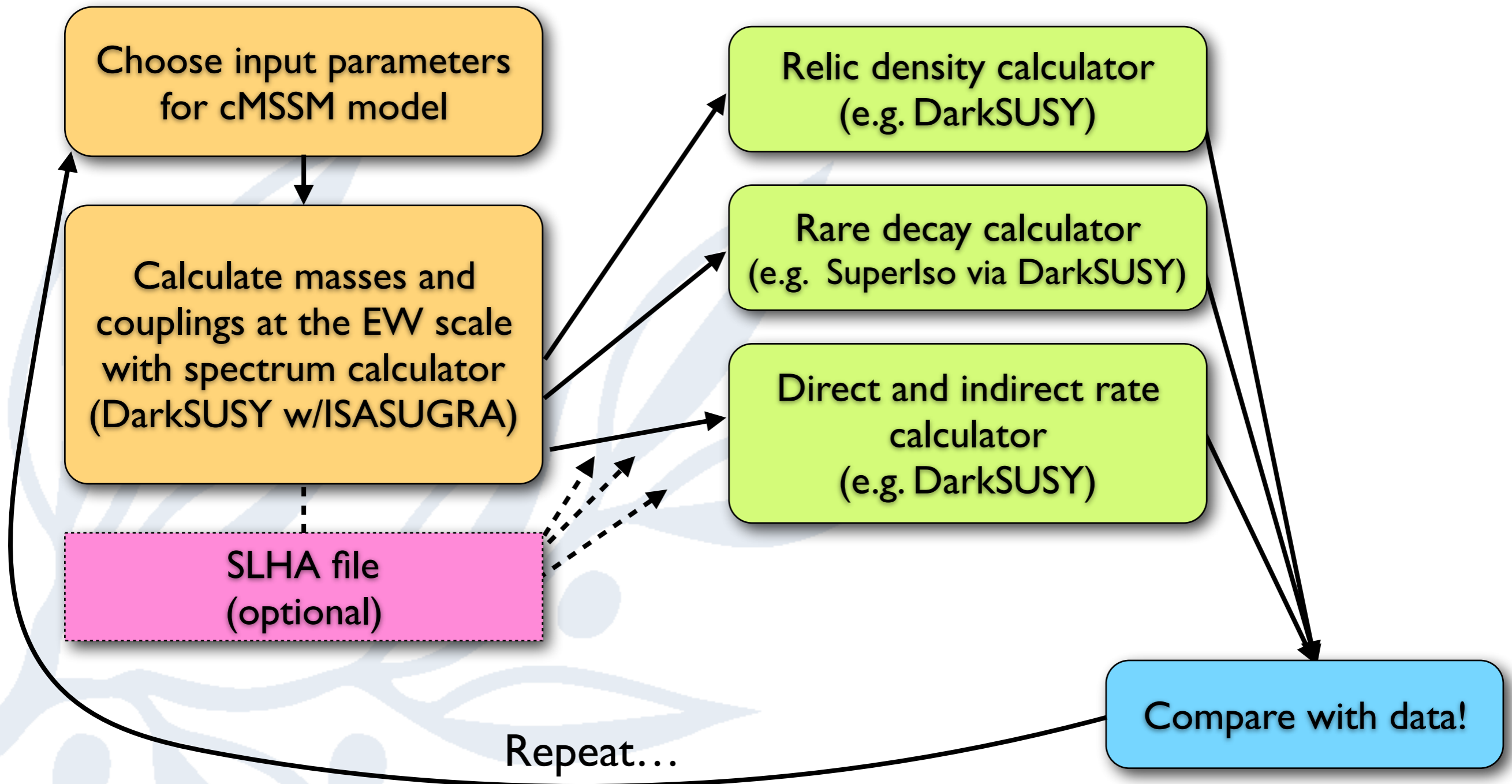
Example of an SLHA file

```
# SUSY Les Houches Accord 2 – MSSM spectrum + Decays
# SPheno v4.0.3
# W. Porod, Comput. Phys. Commun. 153 (2003) 275–315, hep-ph/0301101;
# W. Porod, F.~Staub, Comput. Phys. Commun. 183 (2012) 2458
#       arXiv:1104.1573 [hep-ph]
# in case of problems send email to porod@physik.uni-wuerzburg.de
# Created: 27.02.2018, 12:20
Block SPINFO      # Program information
  1  SPheno      # spectrum calculator
  2  v4.0.3      # version number
#
Block SPhenoINFO  # SPheno specific information
  1  2           # using 2-loop RGEs
Block MODSEL      # Model selection
  1  1          # mSUGRA model
Block MINPAR      # Input parameters
  1  7.00000000E+01 # m0
  2  2.50000000E+02 # m12
  3  1.01113110E+01 # tanb at m_Z
  4  1.00000000E+00 # cos(phase_mu)
  5  -3.00000000E+02 # A0
#
```

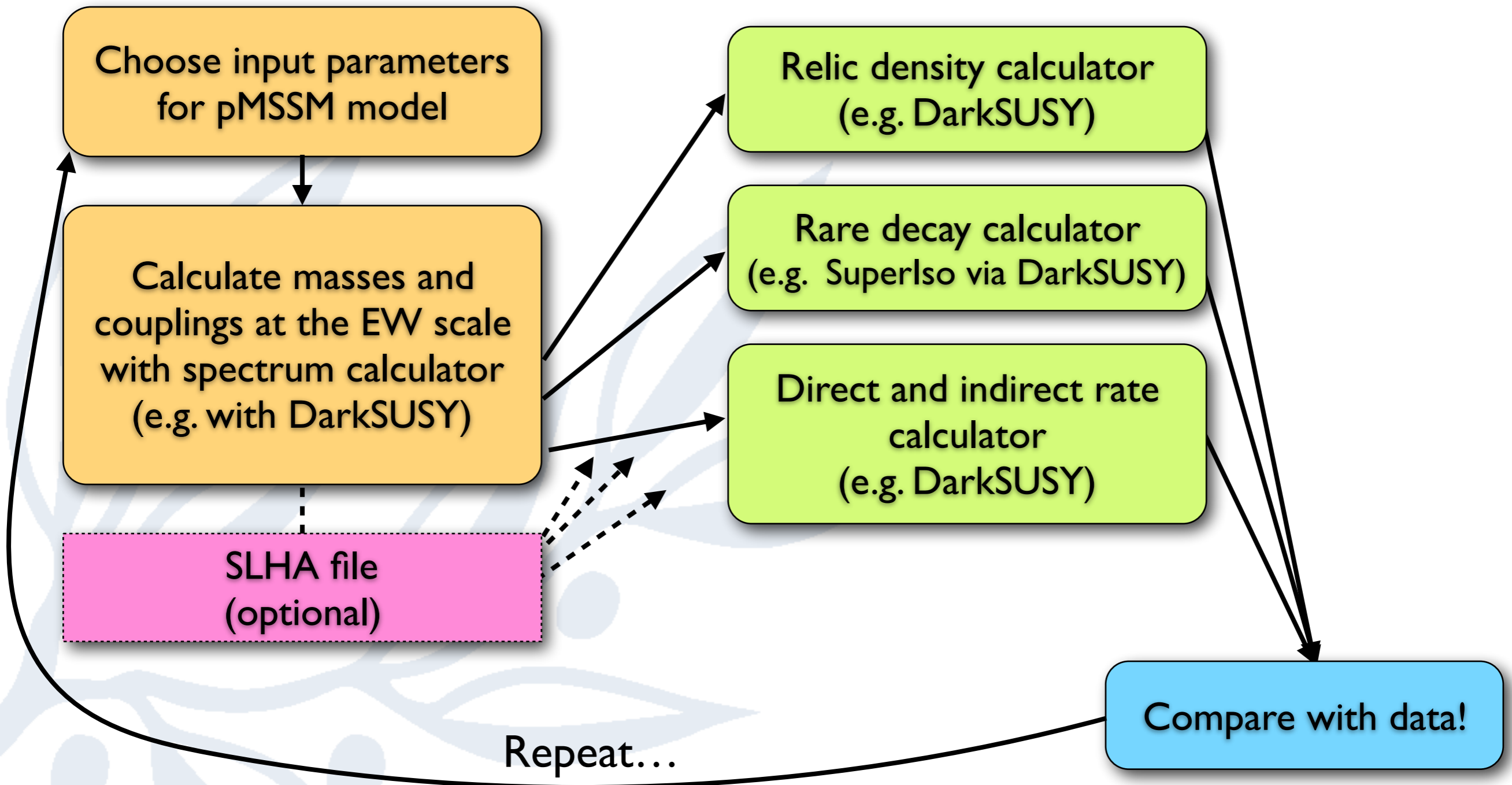
Typical calculation flowchart CMSSM with generic spectrum calculator



Typical calculation flowchart CMSSM with DarkSUSY w/ ISASUGA



Typical calculation flowchart pMSSM (all parameters at EW scale)



Task: use SPheno with DS

- Download SPheno from <https://sphenno.hepforge.org>
 - Unpack it (tar zxvf SPheno-4.0.3.tar.gz)
 - Modify Makefile so that F90 is set to your compiler (typically gfortran)
 - make
 - ./bin/SPheno input/LesHouches.in ⇒ SPheno.spc
 - Run dsmain_wimp and read in this SLHA file
- This is an mSUGRA model specified at the GUT scale
- This is an SLHA file with masses etc specified at the EW scale

If you have trouble running SPheno, the SLHA file can be downloaded here:
<https://www.dropbox.com/s/zptsfn3ictlkfhs/SPheno.spc?dl=0>

Example, SPheno

- The output should look something like

```
No ModSel_FV option found, will try to determine file type from content.  
Found 2x2 stau mixing but no 6x6 mixing, assumes ModSel_FV=2  
SLHA mSUGRA file regarded as output file, will read low-energy values from file.
```

⋮

```
info =          2176  
WIMP mass =     98.091678500000000  
  
Calculating relic density without coannihilations, please be patient...  
  Oh2 =    0.14139676870838372          0          0  
Calculating omega h^2 with coannihilations, please be patient...  
with coannihilations Oh2 =    0.11662741351469266          0          0  
Chemical decoupling (freeze-out) occurred at  
T_f =    4.2152773169914024          GeV.
```

etc

5. Other main programs

- In examples/aux we have a few example programs for other typical calculations, e.g.
 - the program to calculate the relic density in the Silveira-Zee model
 - the program to calculate the relic density in the generic wimp model

Programs in examples/aux I

- **generic_wimp_oh2.f** – to calculate the cross section that gives a correct relic density for different masses
- **ScalarSinglet_RD.f** – to calculate the couplings required in the scalar singlet model to give a correct relic density
- **ucmh_test.f** – example of how the ultra-compact mini halo routines can be used
- **wimpyields.f** – to calculate yields from a generic WIMP annihilating into different particles

Programs in examples/aux II

- **flxconv.f** – to convert between different fluxes and rates from searches for neutrinos from the Sun/Earth
- **caprates.f** – to calculate capture rates in the Sun from spin-independent and spin-dependent scattering
- **caprates_ff.f** – to calculate capture rates in a more advanced setting using full numerical routines and keeping track of capture on individual elements

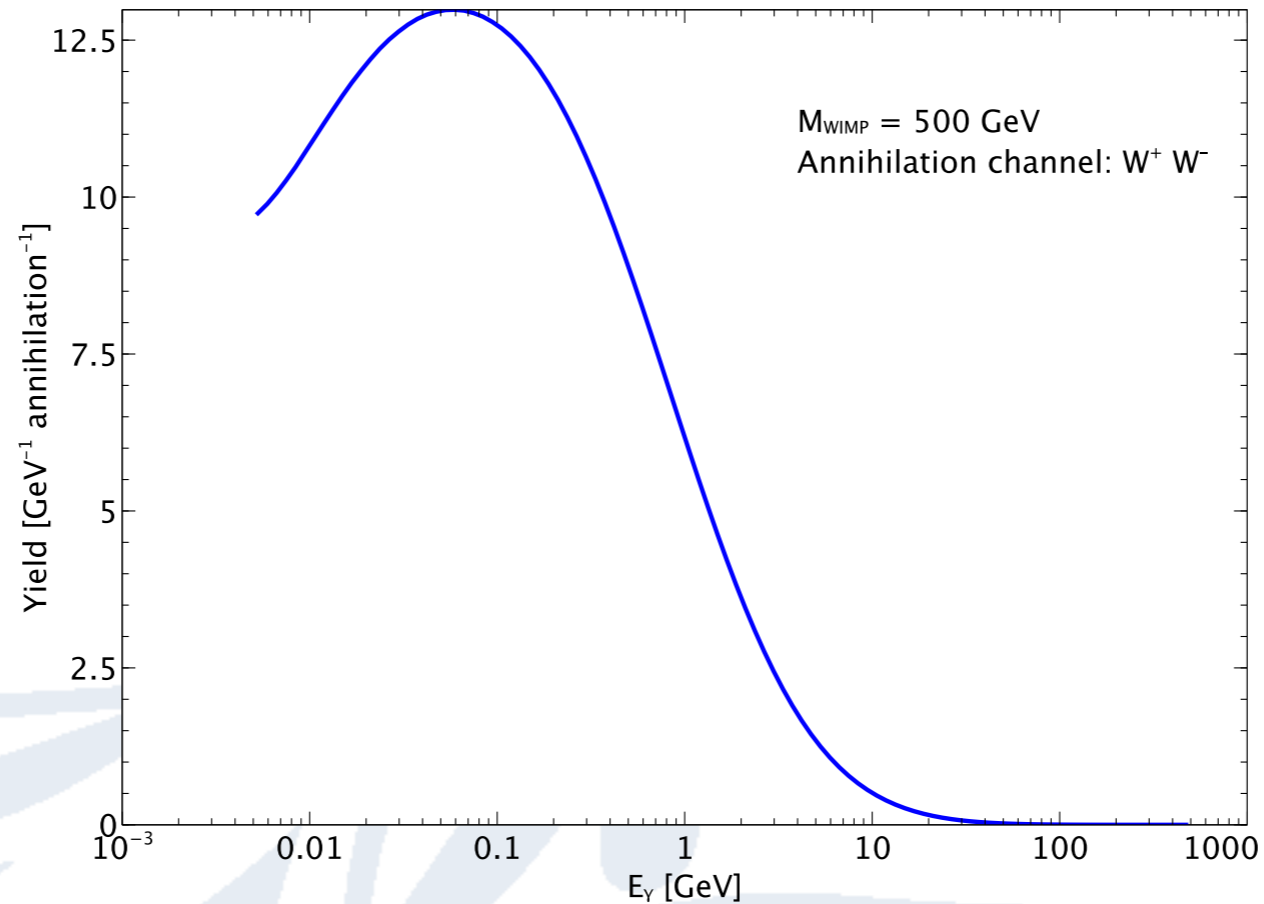
Programs in examples/aux III

- **DMhalo_predef.f** – example on how to use dsdmsdriver to load additional profiles into the halo database
- **DMhalo_table.f** – example on how to load a halo profile from a data file
- **DMhalo_new.f** – example on how to add a new halo parameterization
- **DMhalo_bypass.f** – an example of using a new dsdmsdriver to load a halo profile (i.e. not adding to the existing setup)

wimpyields.f

- wimpyields.f is a simple program that shows how to access the Pythia tables of yields
- These tables can give yields of different particles for different annihilation final states, interpolating in both mass and energy
- **Task:** Calculate the differential gamma ray flux for a WIMP of
 - mass 500 GeV
 - annihilating to $W+W-$
- The PDG particle numbering scheme can be found here: <http://pdg.lbl.gov/2017/reviews/rpp2017-rev-monte-carlo-numbering.pdf>

Results

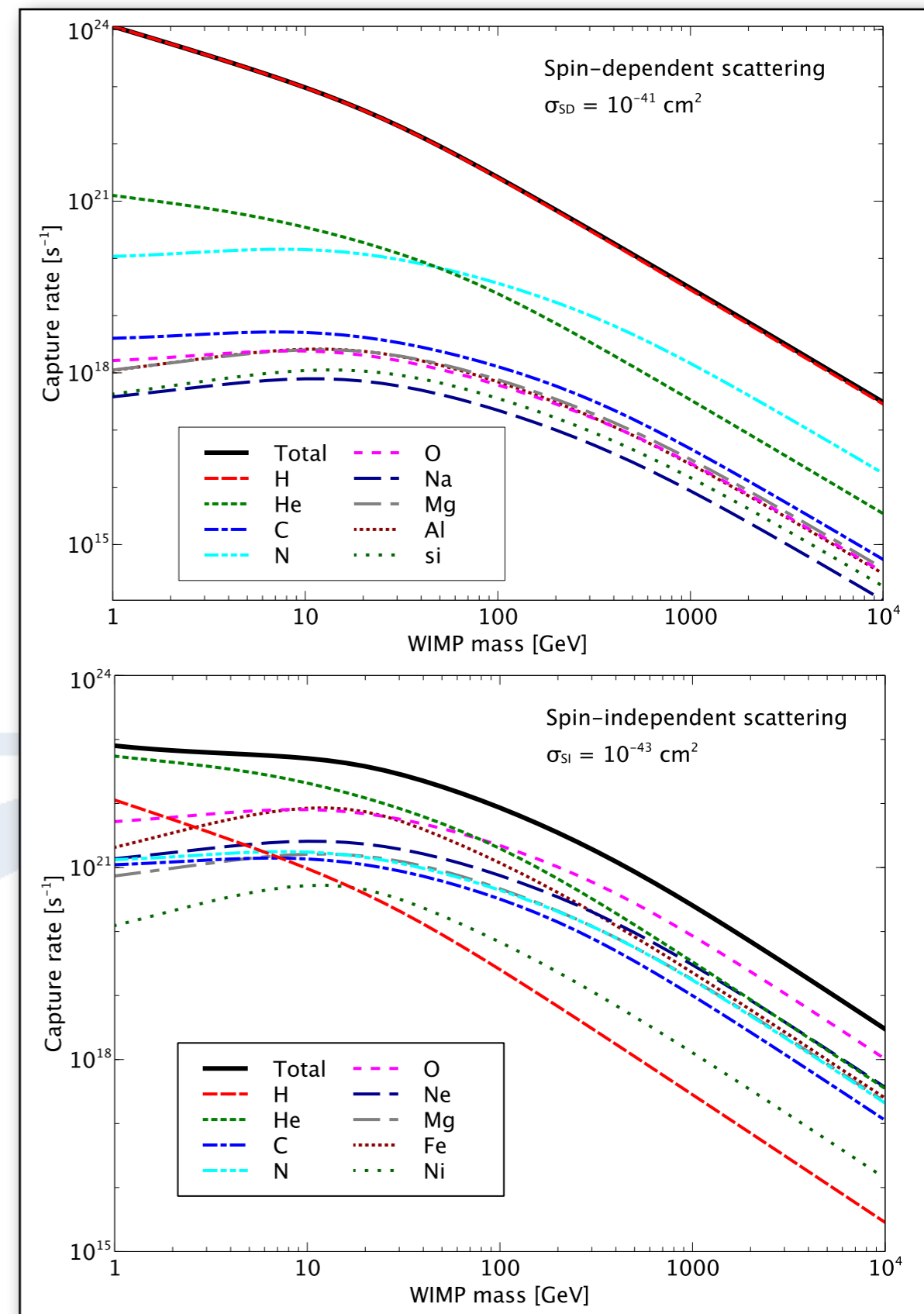


- If you want to explore this further, you could e.g.
 - look at integrated yields
 - look at other channels
 - add astrophysical part, the J-factor, see `dsmain_wimp.F` and/or `DMhalo*.f` for examples on how to do that

caprates.f – capture in the Sun

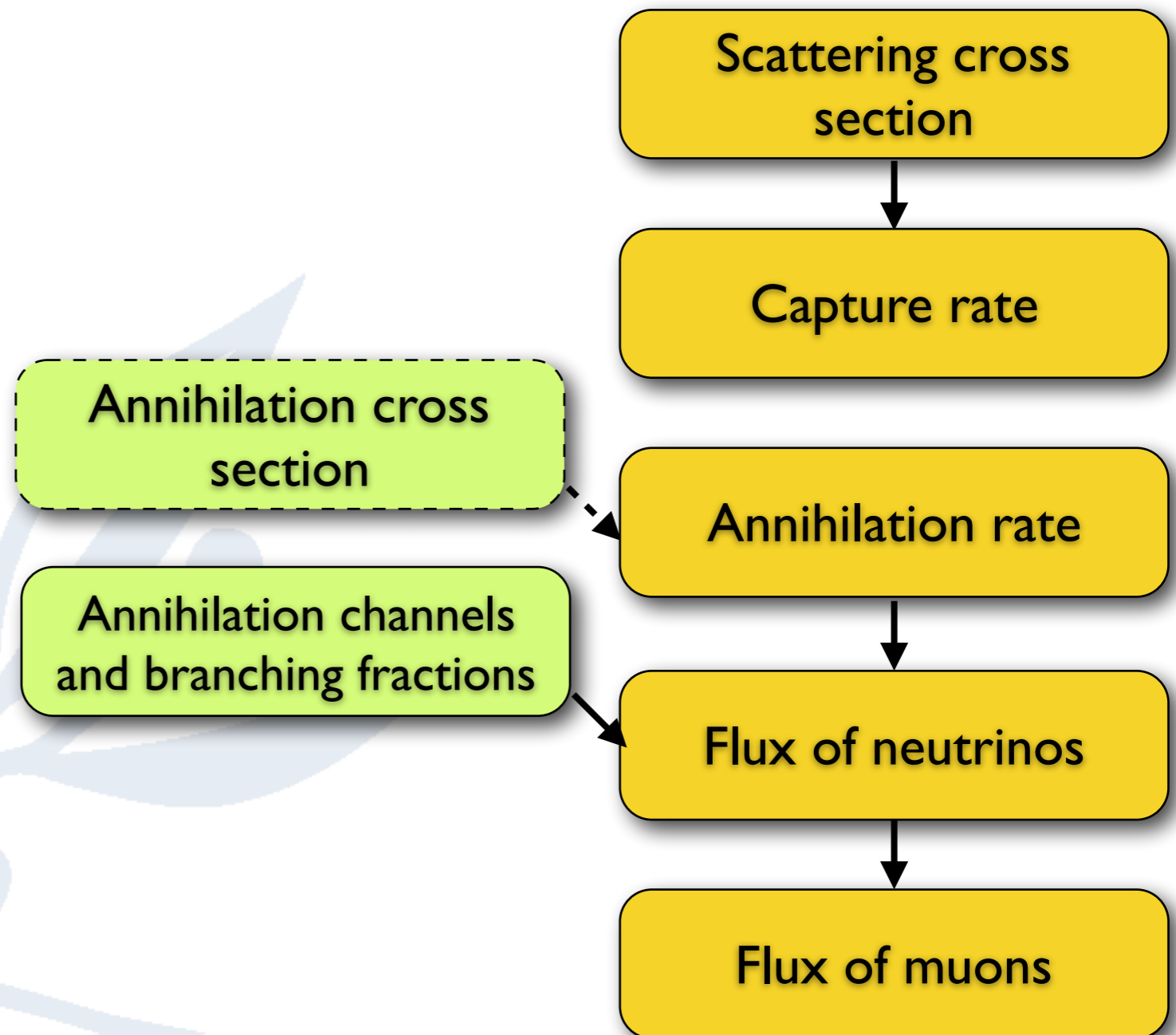
- **Task:** Compile and run caprates.f to calculate the capture rate in the Sun
- You should be able to reproduce the upper solid curves in these figures

Note: To calculate the capture rates on individual elements you need to run the more advanced caprates_ff.f. It takes a bit too long to run here though.



flxconv.f – converting neutrino fluxes from the Sun/Earth

- A given scattering cross section and annihilation channel gives a relation between all quantities
- Can calculate conversion factors to go between them
- For the Sun, the dependence on the annihilation cross section is usually suppressed (equilibrium between capture and annihilation)



flxconv.f

- Imagine you have an experimental limit on the neutrino-induced muon flux from the Sun from Super-Kamiokande and want to convert it to a limit on the neutrino flux, flxconv.f can help you out
- **Task:** Super-Kamiokande (arXiv:1108.3384) gives a limit on the flux of muons (μ^+ and μ^-) of $4.1 \times 10^{-15} \text{ cm}^{-2} \text{ s}^{-1}$ for a WIMP of 100 GeV (within 7 degrees, above 1 GeV and assuming a hard spectrum, W^+W^-). What is the corresponding limit on the
 - spin-dependent scattering cross section σ_p^{SD} (assuming $\sigma_p^{\text{SI}}=0$)?
 - Compare with their figure 8. Does your number agree?
 - Extra task: can you instead give the limit on the neutrino flux (muon neutrinos and anti-neutrinos)?

DMhalo_predef.f

- This example sets up a halo of an object at a given distance from us
- It then shows how you can define multiple halos and use them to calculate J values and gamma ray fluxes

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

- Run the example program, figuring out what it calculates

DMhalo_table.f

- Example on how to read a halo profile from a file instead of defining it parametrically

Note: DMhalo_new is a more advanced example, which shows how to create a completely new parametric profile, just like the default ones already included

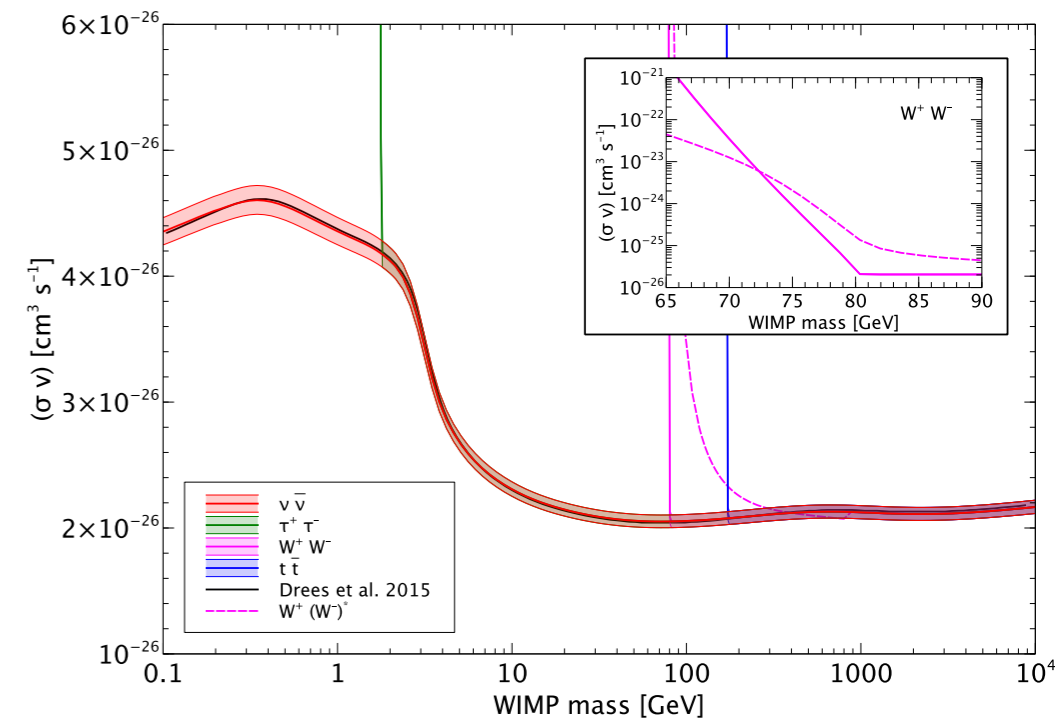
generic_wimp_oh2

- This is the example program that creates the figure of relic density in the generic wimp model

```
cd examples/aux  
make generic_wimp_oh2  
./generic_wimp_oh2
```

Creates an output file `generic_wimp_oh2-planck-sigmav.dat` that can e.g. be plotted

- It scans through the mass range, and for each mass makes a binary search in σv to find the Planck measurement ± 2 sigma
- The default setup takes about 11 min to run, change 'f=1.1' to 'f=1.3' in line 40 and 'fth=1.02' to 'fth=1.1' on line 41 to speed it up for the tutorial (takes 3m07s on my laptop)



Using a replaceable function

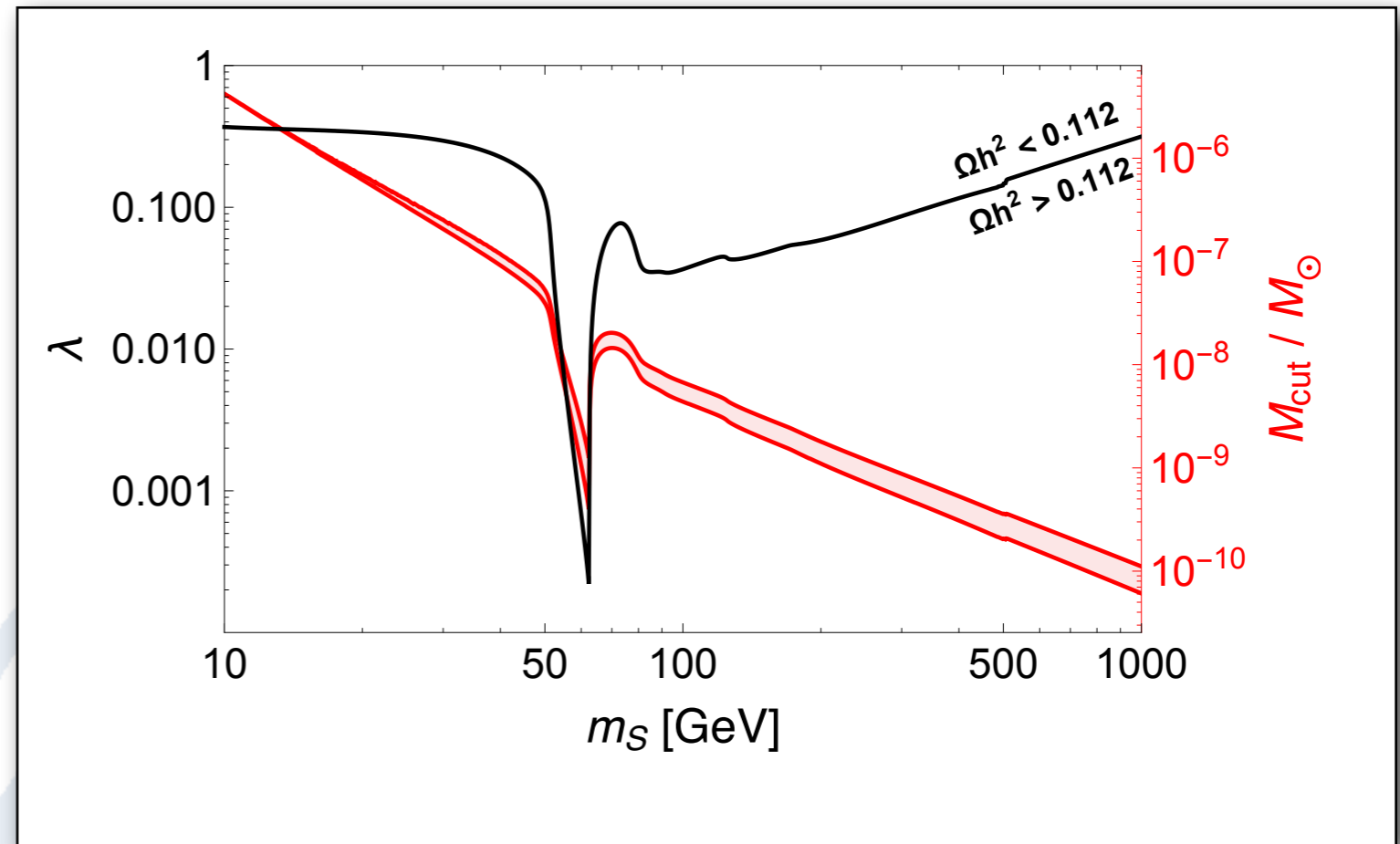
- The default in `generic_wimp` is to use a sharp cut-off in W_{eff} when $m_X < m_{\text{final}}$
- We can use an effective model with an off-shell final state particle, i.e. $XX \rightarrow W^+ W^{*-}$
- An implementation of this is in `examples/aux/user_replaceables/dsanwx.f`
- Just compile replacing the regular `dsanwx.f` with this new one to test it:

```
make generic_wimp_oh2_threshold
```

(takes about 3m32s to run)

ScalarSinglet_RD.f

- This is an example of using the scalar singlet particle physics module
- The model has two parameters, m_S and λ



- **Task:** Run ScalarSinglet_RD and see if you can reproduce the curves in the figure above, i.e.
 - which parameter values that give the correct relic density (black curve)
 - the smallest halo sizes, i.e. the cut-off scale (red curve)
- Note: takes about 18 min to run, we don't do it here...**

6. Replaceable functions

- If you want to modify an existing DarkSUSY function or subroutine, **DON'T!**
- Instead create your own version of the routine and link to that one instead.
- You can either just create your own version and link to it (before the DS library is linked to), or
- Use the script `scr/make_replaceable.f` to make a `user_replaceable` function for you, for which the makefiles are already set up to work

Replaceable function example

- As an example, we will look at the source term for DM annihilation in the galactic halo

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

This code is in `src_models/generic_wimp/cr/dscrsource.f`

- Let's add a boost factor from substructures

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

Replaceable function (cont)

- In the root directory, type
`scr/make_replaceable.pl src_models/generic_wimp/
cr/dscrsource.f`
- This will give you a new file
`src_models/generic_wimp/user_replaceables/
dscrsource.f`
- Modify it, configure and make again (in the root), then
`make -B dsmain_wimp DS_MODULE=generic_wimp`
in examples and run `dsmain_wimp`
- **Task:** perform this change and run `dsmain_wimp` to
see if you get the intended change on

Replaceable function (cont.)

- At the end of `scr/make_replaceable.pl` we got

```
*****  
File created: src_models/generic_wimp/user_replaceables/dscrsources.f  
The file is also added to the list in  
src_models/generic_wimp/user_replaceables/files-to-include.txt  
You now need to run configure, modify your file and compile.  
*****
```

- To stop using our user-replaced function, we just delete the file from `src_models/generic_wimp/user_replaceables/files-to-include.txt` (i.e. the file should just read `'SRC = '`), `configure` and `make` again

7. Direct detection example

- The main routine for cross section calculations is `dsddsigma(v,e,a,z,sigij,ierr)` (resides in `src_models`)
- It returns the unpolarized equivalent WIMP nucleus cross section including form factors
- It relies on the couplings defined by the model and depends on velocity and recoil energy
- It returns the partial cross section array `sigij` in an effective operator framework.
 - `sigij(1,1)` is the usual spin-independent cross section
 - `sigij(4,4)` is the usual spin-dependent cross section

`sigij` is a 27×27 real*8 array

Direct detection routines

- In src/ we have the two routines
 - dsddg2sigma that take couplings g as input and calculates the sig_{ij} array for given velocities and recoil energies including form factors
 - dsdddrde($t, e, n, a, z, \text{stoich}, \text{rsi}, \text{rsd}, \text{modulation}$) calculates the differential rate

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

Direct detection example

- **Task:** Create a main program that
 - defines an MSSM model
 - calculates and prints the cross sections in the zero-momentum limit and on some nucleus, e.g. Na-23 including form factors
 - scans over recoil energies and calculates the differential rate dR/dE as a function of recoil energy E , e.g. on the target NaI

Direct detection example

- **Task:** Create a main program that
 - defines an MSSM model
 - calculates and prints the cross sections in the zero-momentum limit and on some nucleus, e.g. Na-23 including form factors
 - scans over recoil energies and calculates the differential rate dR/dE as a function of recoil energy E , e.g. on the target NaI

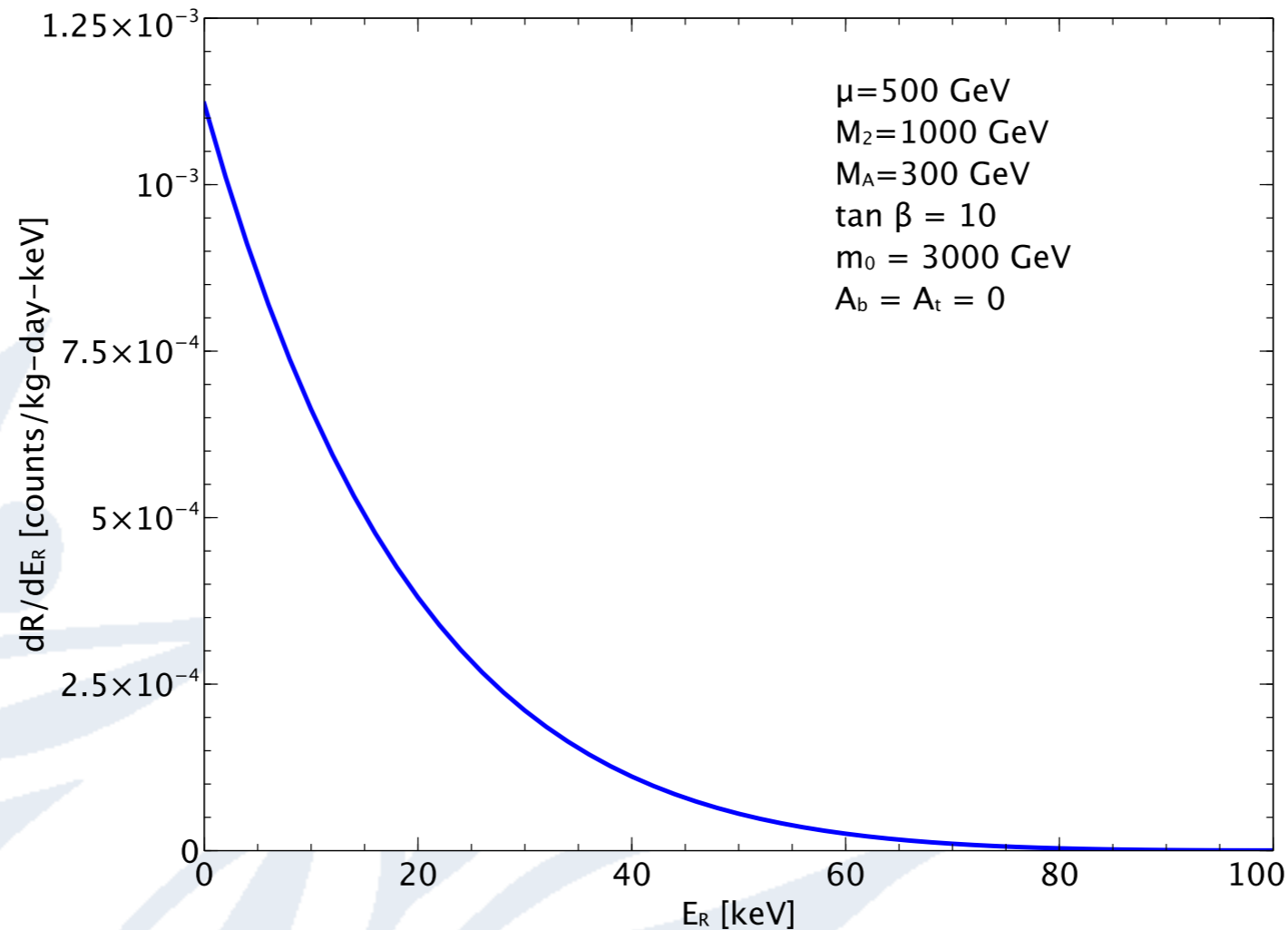
Hints!

- You can use the main program for MSSM you made earlier as a starting point
- Look in `dsmain_wimp.F` on how to get the cross sections and the differential rates. The routines you need to call are `dsddsigma` and `dsddrde`. Look inside those routines to see what the arguments are what they mean

An example main program can be found here:

<https://www.dropbox.com/s/re7vj19287q4d83/myddprog.f?dl=0>

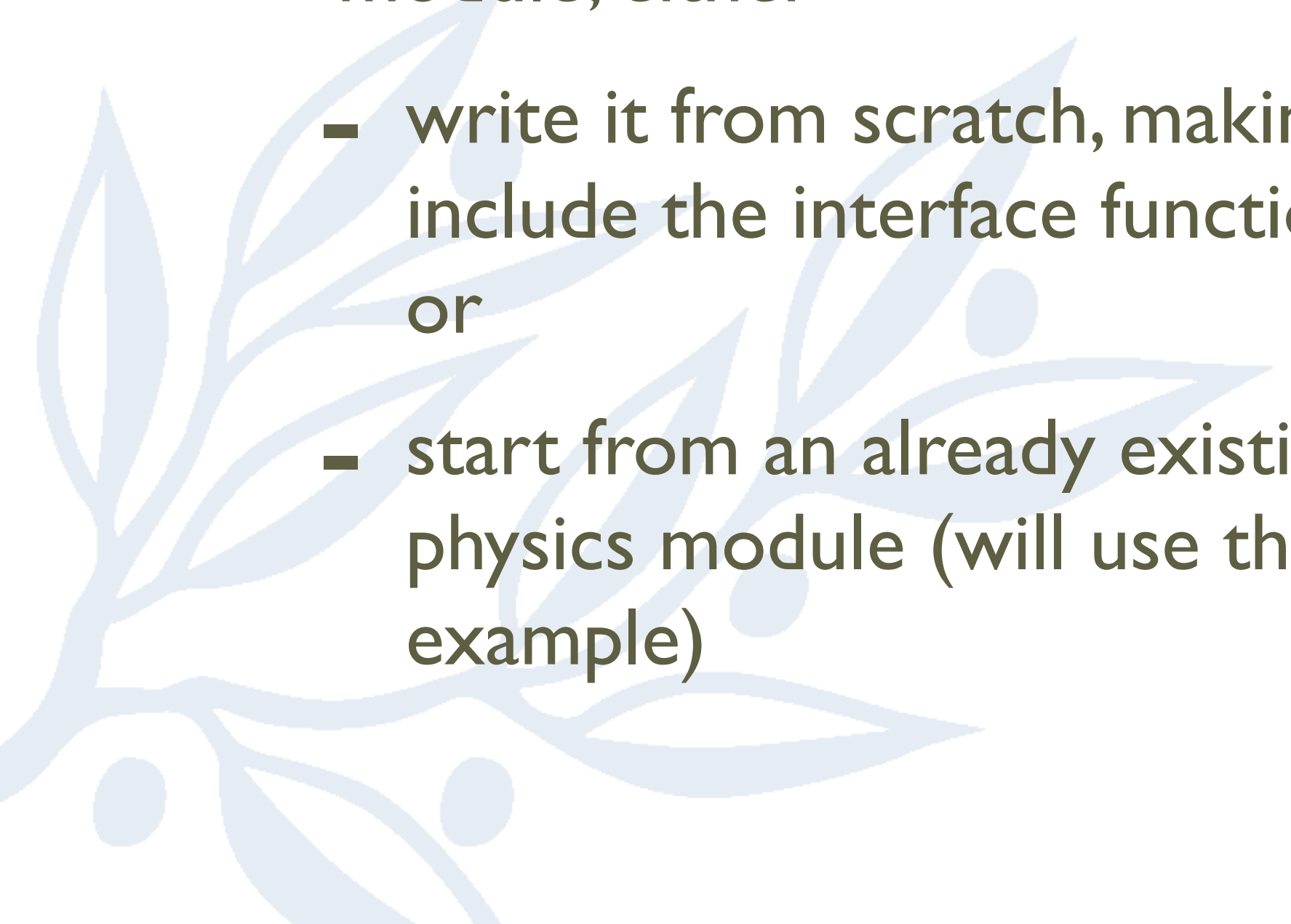
Example output



To explore this further, you could e.g.

- include modulations (see dsdddrde on how to do this)
- look at other model parameters, e.g. by reading an SLHA file
- change halo properties
- change form factors
- change target material

8. Creating a new particle physics module

- To create a completely new particle physics module, either
 - write it from scratch, making sure to include the interface functions you need, or
 - start from an already existing particle physics module (will use this as an example)
- 

Particle physics modules

- In `src_models` we currently have
 - `mssm` - Minimal Supersymmetric Standard Model
 - `silveira_zee` - Scalar singlet model
 - `generic_wimp` - a generic annihilating WIMP model
 - `generic_decayingDM` - a generic decaying dark matter model
 - `empty` - an empty model with just the basic set of interface functions for a 'fresh' start
- If you add one and want others to use it, please let us know and we can add it to the distribution (or point to your preferred download page)

Example, extend generic wimp

- The generic WIMP model has the following parameters
 - m_{wimp} = mass of WIMP
 - selfconjugated DM or not
 - annihilation cross section σv
 - pdg code of annihilation final state
 - spin-independent scattering cross section

You need to have autoconf installed for this to work

Extended generic wimp

- Now assume that we want to make the following change: Instead of having the annihilation cross section constant, we want to add a term proportional to v^2 , i.e.

$$\sigma v = a \quad \rightarrow \quad \sigma v = a + bv^2$$

- **Task:** Create a new module that includes this additional b-parameter

Extended generic wimp

- Create a new module by typing (in the root directory)

```
scr/make_module.pl generic_wimp extended_wimp
```

- Then type
./configure
make
- You then have a new module extended_wimp in src_models
- It is right now identical to generic_wimp (apart from name changes), but you can modify it to your liking, by adding the b-term
- Which functions do you need to modify?

You need to have autoconf installed for this to work

Extended generic wimp

- Create a new module by typing (in the root directory)

```
scr/make_module.pl generic_wimp extended_wimp
```

- Then type
./configure
make

- You then have a new module extended_wimp in src_models
- It is right now identical to generic_wimp (apart from name changes), but you can modify it to your liking, by adding the b-term
- Which functions do you need to modify?

-
- You need to modify

- src_models/extended_wimp/ini/dsgivemodel_extended_wimp.f
- src_models/extended_wimp/an/dsanwx.f (annihilation cross section), however, this one already contains the b-term, but it is not used in the default setup
- src_models/extended_wimp/include/dsextended_wimp.h – if you need to add more global variables
- a main program of your choice to use your new

You need to have autoconf installed for this to work

Helpful tools

- The `extended_wimp` is automatically included in the build system, but when/if you start adding files you need to tell the build system. To help you, we have a few scripts
 - `scr/makemf.pl <directory>` - adds all source files in the given `<directory>` to the relevant makefiles, or rather makefile.in's (without argument it adds source files in all directories in `src/` and `src_models/`)
 - `scr/preconfig.pl` - adds source files **AND** new directories to the build system and updates both the configure script and makefiles

You need to have `autoconf` installed for this to work

Main program

- You can e.g. use your new module with `dsmain_wimp` (or any other main program you choose)
- For `dsmain_wimp`, you need it to be aware of your new module by adding lines of this type:

```
#if MODULE_CONFIG == MODULE_extended_wimp  
[add your code here]  
#endif
```

This can be done by e.g. copy-pasting the corresponding `generic_wimp` lines and replace `generic_wimp` with `extended_wimp`

Feedback

- I would like to get feedback on this tutorial
- Please fill out this (short) survey

<https://goo.gl/forms/t0AkBk8l7Vupb8UpI>

Conclusions

- DarkSUSY 6 publically available (finally!)
- DarkSUSY 6 is much more modular and include other improvements.
- When comparing different signals, it is crucial to perform these calculations in a consistent framework, with e.g. a tool like DarkSUSY

ありがとう

Joakim Edsjö
edsjo@fysik.su.se



Stockholm
University



Osaka Klein
centre