FlexibleSUSY – A spectrum generator generator for supersymmetric models

Alexander Voigt

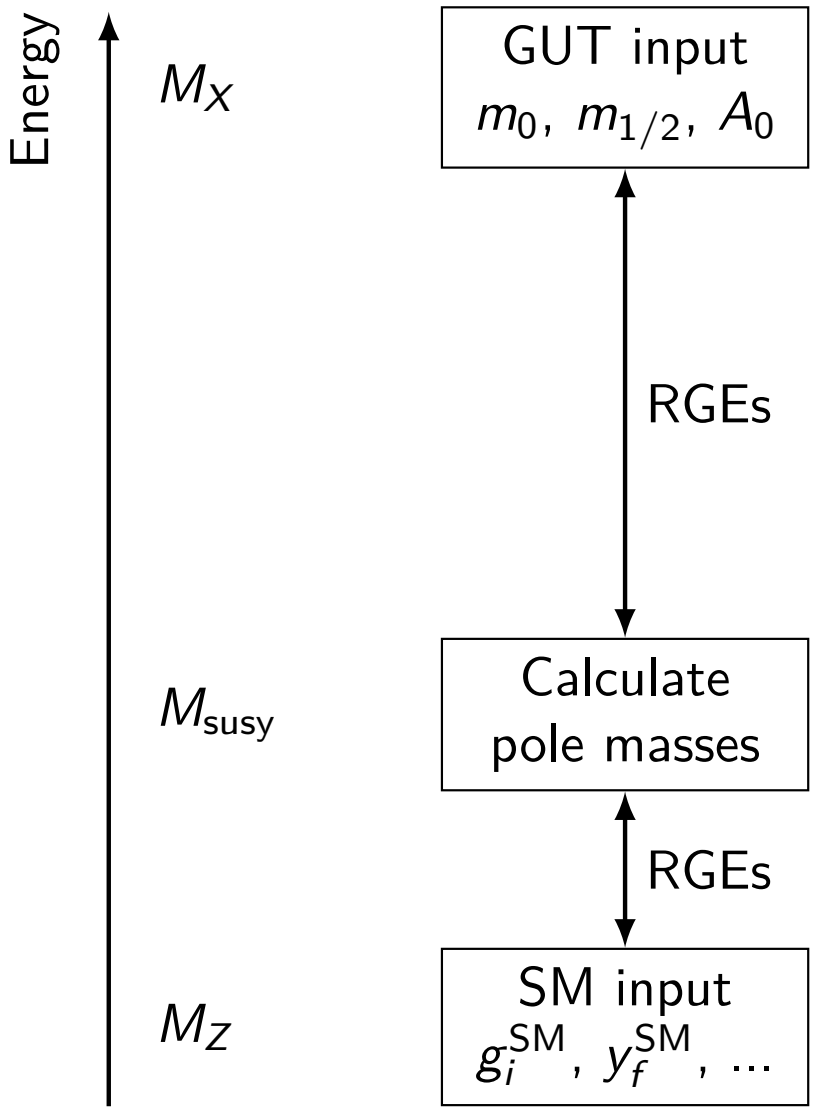
Technische Universität Dresden
Institut für Kern- und Teilchenphysik

Linear Collider Froum DESY 2013
11 October 2013

In collaboration with Peter Athron, Jae-hyeon Park and Dominik Stöckinger

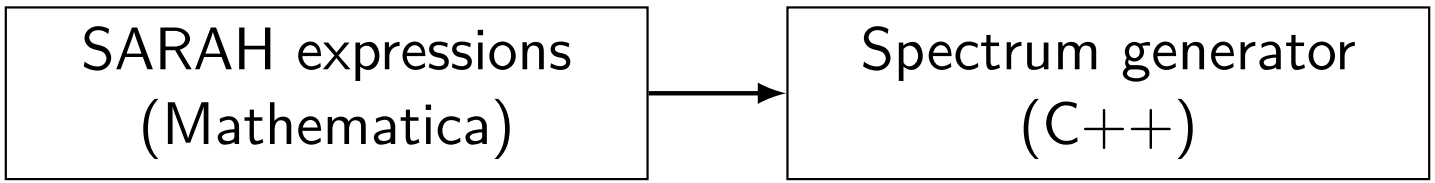
Content

Motivation – What is a spectrum generator?



Motivation – Why creating a new one?

FlexibleSUSY provides Mathematica meta code which creates a spectrum generator



Motivation:

- large variety of supersymmetric models

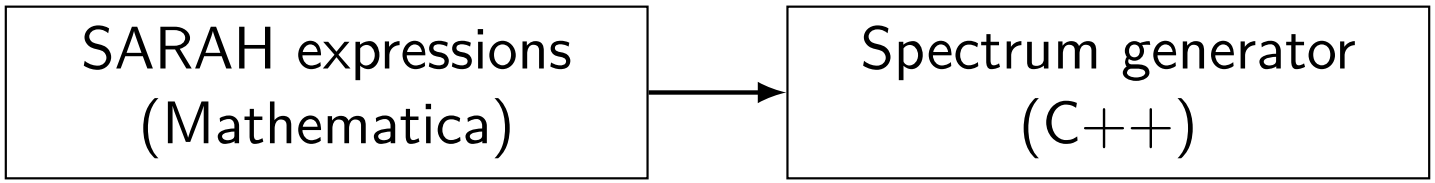
$\rightarrow$ user customization desired

- convergence problems in certain parameter regions

$\rightarrow$ provide alternative RG solvers

Content

Design goals



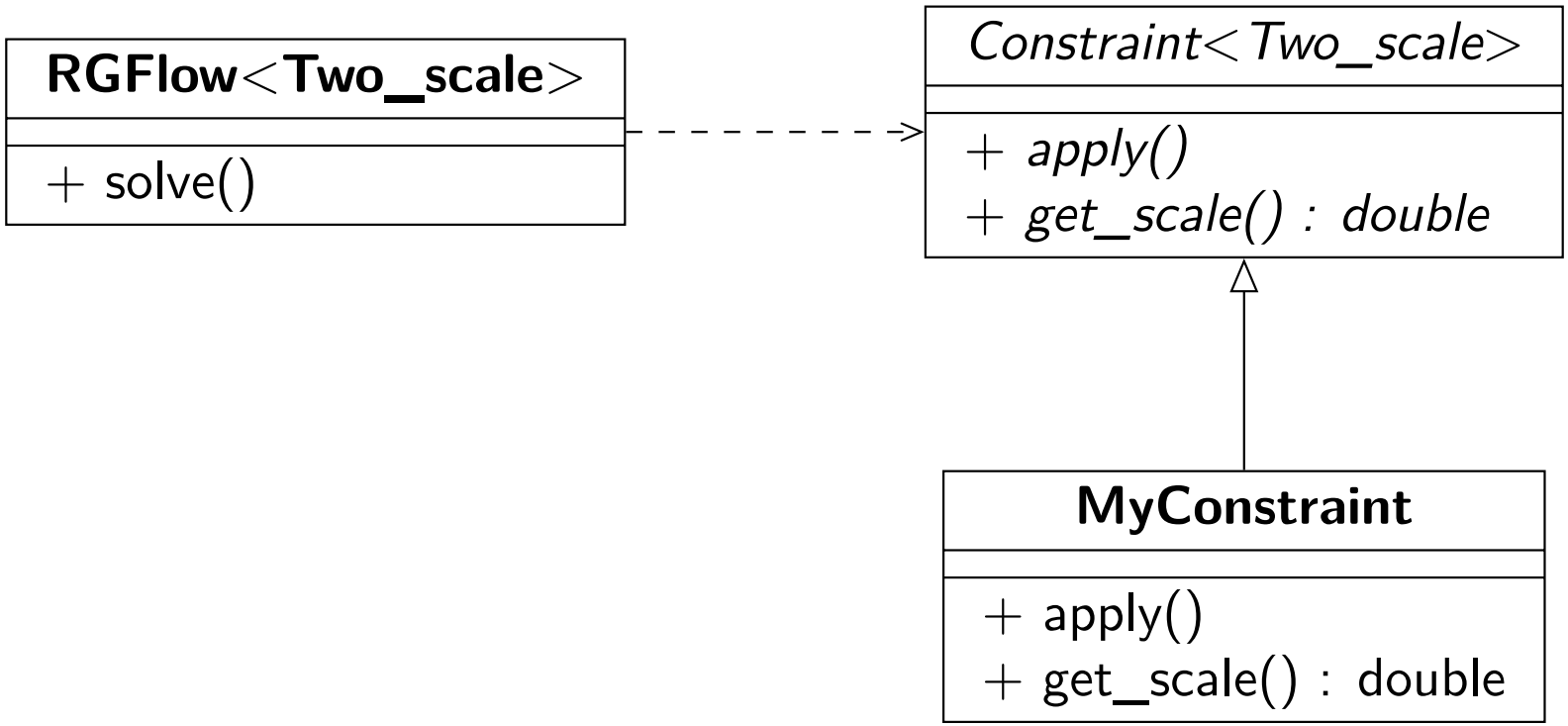
Design goals:

- modular, object oriented C++ code $\Rightarrow$ easy to hack! ✓
- fast (smart linear algebra, multithreading) ✓
- multiple RGE solvers:
  - two-scale running (adaptive Runge-Kutta) ✓
  - lattice method + variants (Jae-hyeon Park) ✓
- SARAH-like user interface ✓
- tower of effective field theories ✗

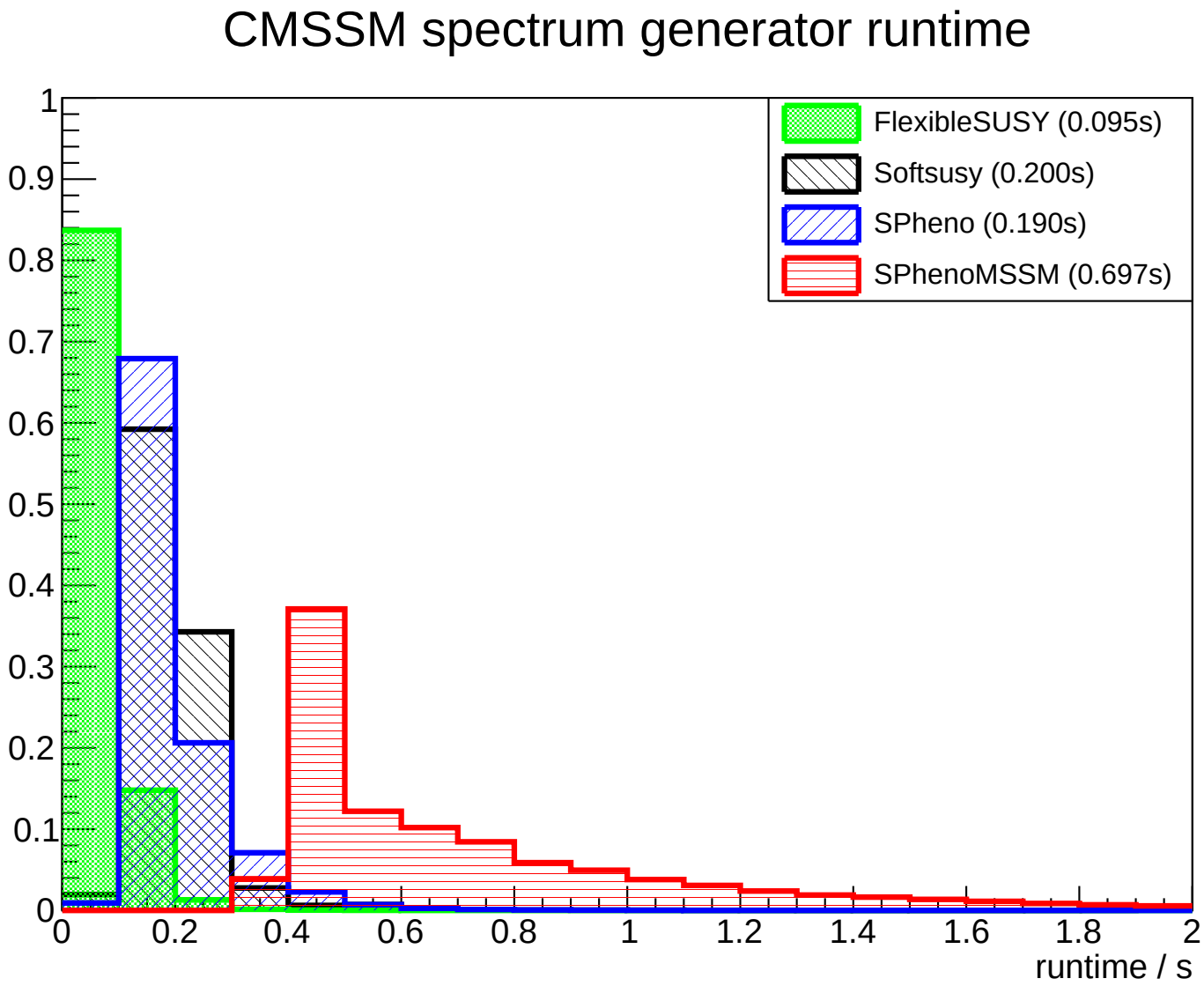
Modularity – generated C++ code

```
typedef Two_scale T; // or Lattice
MSSM<T> mssm;
MSSM_input_parameters input;

std::vector<Constraint<T>*> constraints = {
  new MSSM_low_scale_constraint<T>(input),
  new MSSM_susy_scale_constraint<T>(input),
  new MSSM_high_scale_constraint<T>(input)
};

// solve RG eqs. with the above constraints
RGFlow<T> solver;
solver.add_model(&mssm, constraints);
solver.solve();

mssm.calculate_spectrum();
```

Modularity – Constraint interface

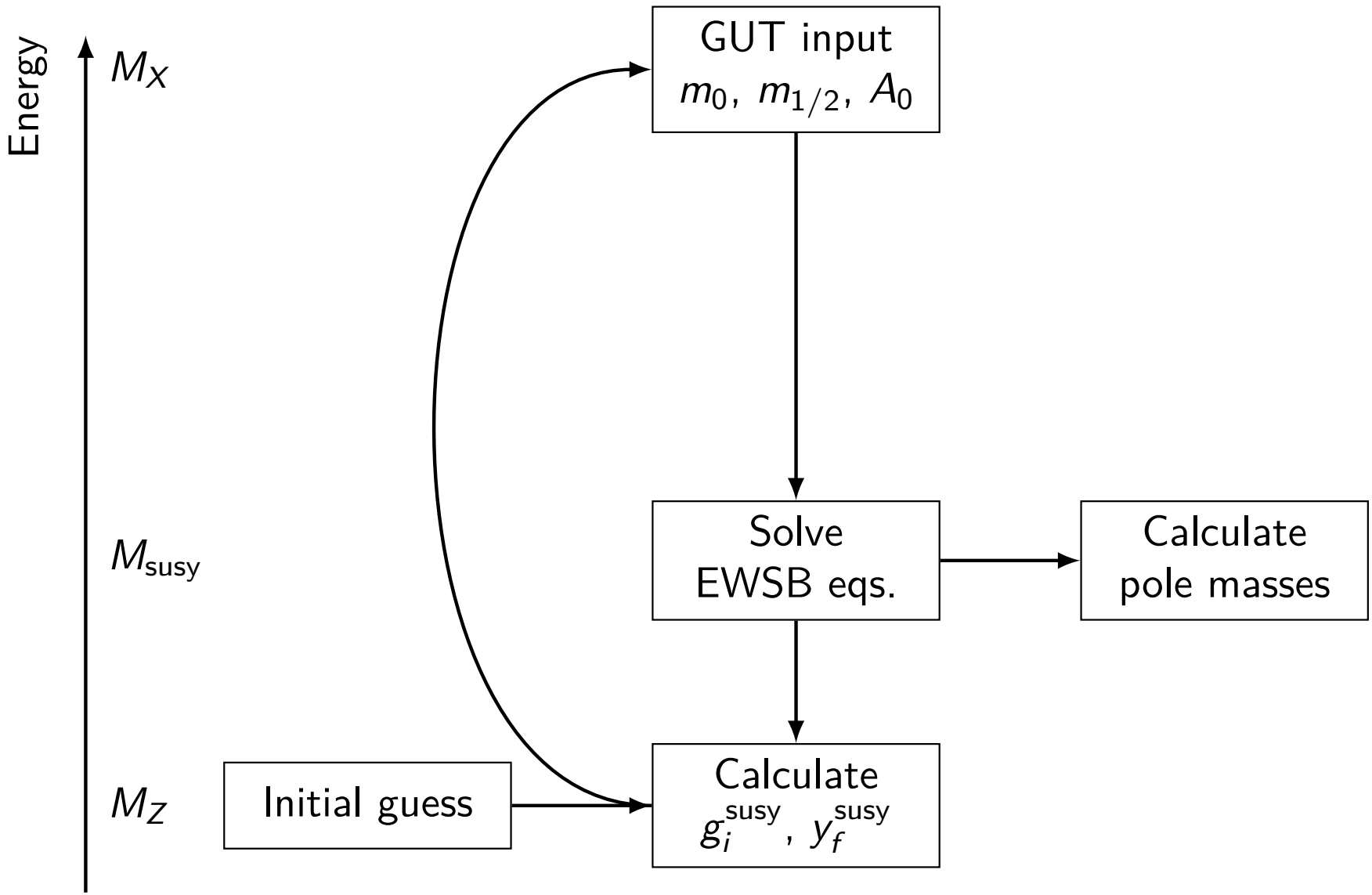


Speed – runtime comparison



g++ 4.8.0, gfortran 4.8.0

RGE solvers – two-scale algorithm



Content

Usage example

Get the source code:

```
$ git clone https://github.com/Expander/FlexibleSUSY
$ cd FlexibleSUSY
```

Create a MSSM spectrum generator:

```
$ ./createmodel --models=MSSM
$ ./configure --with-models=MSSM
$ make
```

Run it:

```
$ ./models/MSSM/run_MSSM.x
```

Run it with the SLHA interface:

```
$ ./models/MSSM/run_MSSM.x \
   --slha-input-file=input.slha \
   --slha-output-file=output.slha
```

FlexibleSUSY model file

```
FSModelName = "MSSM";

MINPAR = { {1, m0},
           {2, m12},
           {3, TanBeta},
           {4, Sign[Mu]},
           {5, Azero} };

EWSBOutputParameters = { B[Mu], Mu };

SUSYScale = Sqrt[M[Su[1]]*M[Su[6]]];

HighScale = g1 == g2;

HighScaleInput = {
   {mHd2, m0^2}, {mHu2, m0^2}, {mq2, UNITMATRIX[3] m0^2},
   ...
};

LowScale = SM[MZ];

LowScaleInput = { ... };
```

FlexibleSUSY model file – special features

Minimizer (iterative):

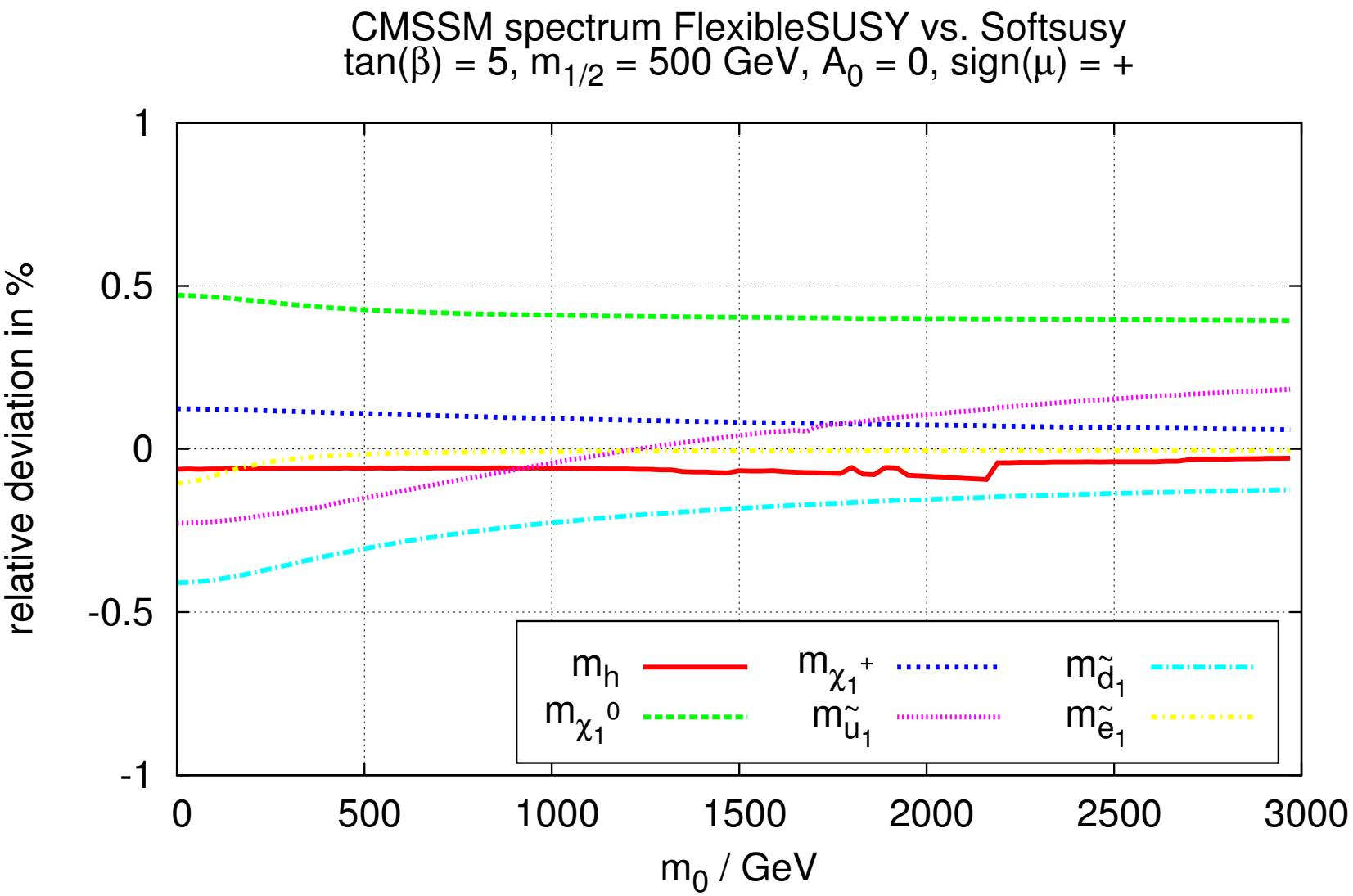
```
LowScaleInput = {
   FSMinimize[{vd,vu},
              (SM[MZ] - Pole[M[VZ]])^2 /
                 STANDARDDEVIATION[MZ]^2
            + (SM[MH] - Pole[M[hh[1]]])^2 /
                 STANDARDDEVIATION[MH]^2 ]
};
```

Root finder (iterative):

```
LowScaleInput = {
   FSFindRoot[{vd,vu}, {SM[MZ] - Pole[M[VZ]],
                        SM[MH] - Pole[M[hh[1]]]} ]
};
```

Content

Spectrum comparison FlexibleSUSY vs. Softsusy



Conclusions

FlexibleSUSY

- is **Modular** (C++ classes, easy to modify and extend)
- is **Fast** (CMSSM runtime $\approx 0.1s$)
- Provides **different RGE solvers**
  - two-scale running (adaptive Runge-Kutta)
  - lattice method + variants

Currently supported models:

- MSSM, NMSSM, SMSSM, UMSSM, E6SSM, MRSSM

Future plans:

- determination of $\sin\theta_W$ from G_μ
- two-loop leading log Higgs mass corrections
- tower of effective field theories