FlexibleSUSY – A spectrum generator generator for supersymmetric models

Peter Athron, Jae-hyeon Park,
Dominik Stöckinger, Alexander Voigt

University of Adelaide, Universitat de València, TU Dresden

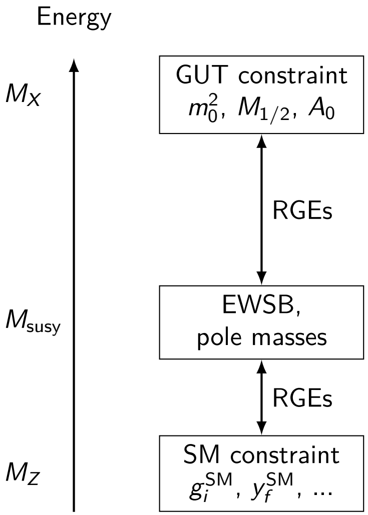
SUSY 2014
21-26 July, Manchester

TECHNISCHE UNIVERSITÄT DRESDEN

Content

Content

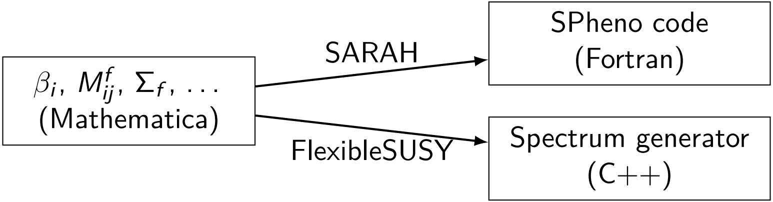
Motivation – What is a spectrum generator?



Motivation – Available SUSY spectrum generators

Model	Spectrum generator
MSSM	ISASUSY, SOFTSUSY, SPheno, SuSeFlav, SuSpect
NMSSM	NMSPEC, SOFTSUSY, (SPheno)
CE_6SSM	CE6SSMSpecGen
any SUSY model	SARAH, **FlexibleSUSY**

Motivation – Why creating a new one?



Motivation:

- large variety of supersymmetric models
  $\rightarrow$ user customization desired
- convergence problems in certain parameter regions
  $\rightarrow$ provide alternative RG solvers
- High-dimensional parameter space
  $\rightarrow$ short run-time desired

Design goals

- **modular**, object oriented, well readable C++ code
  ⇒ easy to customize, reuse and extend! ✓
- **multiple RGE+BC solvers**:
  - two-scale running (adaptive Runge-Kutta) ✓
  - lattice method + variants (Jae-hyeon Park) ✓
- **speed** (smart linear algebra, multithreading) ✓
- **high precision** (2-loop β_i, 1-loop Σ_f, 1-loop EWSB, leading 2-loop Higgs for MSSM + NMSSM) ✓
- SARAH-like user interface ✓
- tower of effective field theories ✓

Content

Usage examples – NMSSM spectrum generator

1. Get the source code from https://flexiblesusy.hepforge.org
2. Create a NMSSM spectrum generator:

```
$ ./install-sarah # if not already installed
$ ./createmodel --name=NMSSM
$ ./configure --with-models=NMSSM
$ make
```

3. Calculate spectrum for given parameter point (SLHA format):

```
$ ./models/NMSSM/run_NMSSM.x \
   --slha-input-file=models/NMSSM/LesHouches.in.NMSSM

Block MASS
   1000021     5.05906233E+02   # Glu
   1000024     1.46609728E+02   # Cha_1
   1000037     3.99399367E+02   # Cha_2
        37     4.33363816E+02   # Hpm_2
...
```

Usage examples – boundary conditions

```
$ cat models/NMSSM/FlexibleSUSY.m
```

```
FSModelName = "NMSSM";

MINPAR = { {1, m0}, {2, m12}, {3, TanBeta}, {5, Azero} };

EXTPAR = { {61, LambdaInput} };

EWSBOutputParameters = { \[Kappa], vS, ms2 };

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 = { ... };
```

Usage examples – NUH-NMSSM spectrum generator

Create NUH-NMSSM model file:

```
$ mkdir model_files/NUHNMSSM
$ cp model_files/NMSSM/* model_files/NUHNMSSM/
```

Edit the model file:

```
EXTPAR = {{21, mHd2In}, {22, mHu2In}, ... };

HighScaleInput = {
  {mHd2, mHd2In}, {mHu2, mHu2In}, ... };
```

Create the NUH-NMSSM spectrum generator:

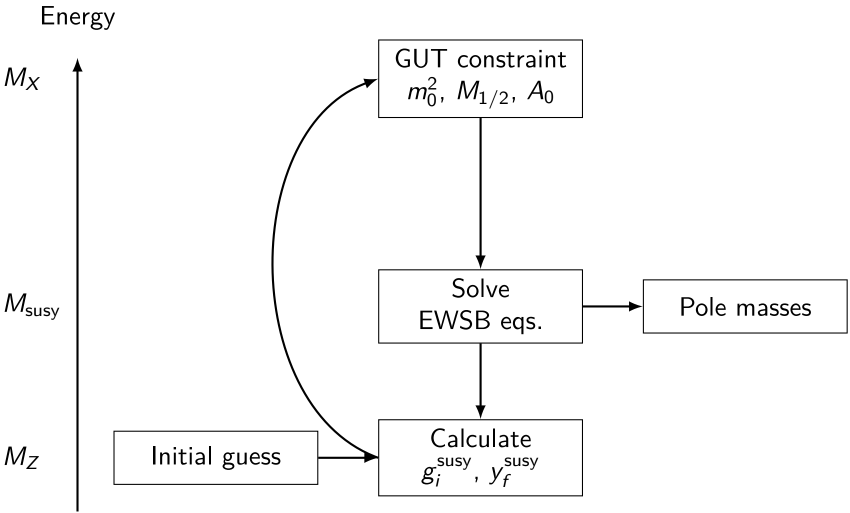
```
$ ./createmodel --name=NUHNMSSM --sarah-model=NMSSM
$ ./configure --with-models=NUHNMSSM
$ make
$ ./models/NUHNMSSM/run_NUHNMSSM.x \
   --slha-input-file=[...]
```

Content

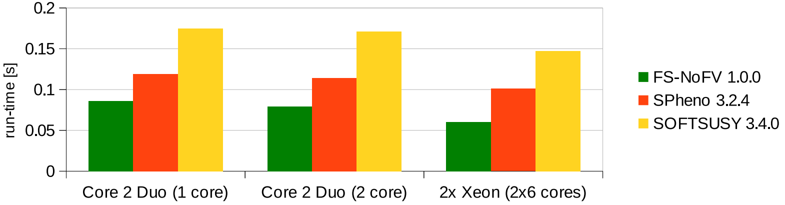
Generated spectrum generator C++ code

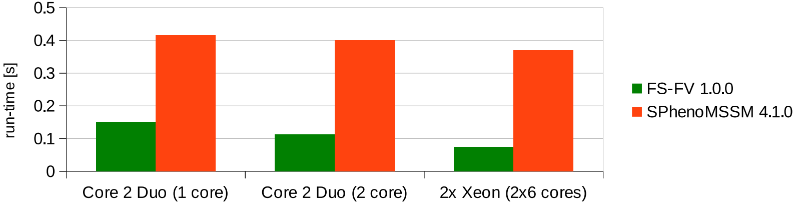
```
typedef Two_scale T; // or Lattice
NMSSM<T> nmssm;
NMSSM_input_parameters input;
QedQcd qedqcd;

// create BCs
std::vector<Constraint<T>*> constraints = {
   new NMSSM_low_scale_constraint<T>(input, qedqcd),
   new NMSSM_susy_scale_constraint<T>(input),
   new NMSSM_high_scale_constraint<T>(input)
};

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

nmssm.calculate_spectrum();
```

RGE+BC solver



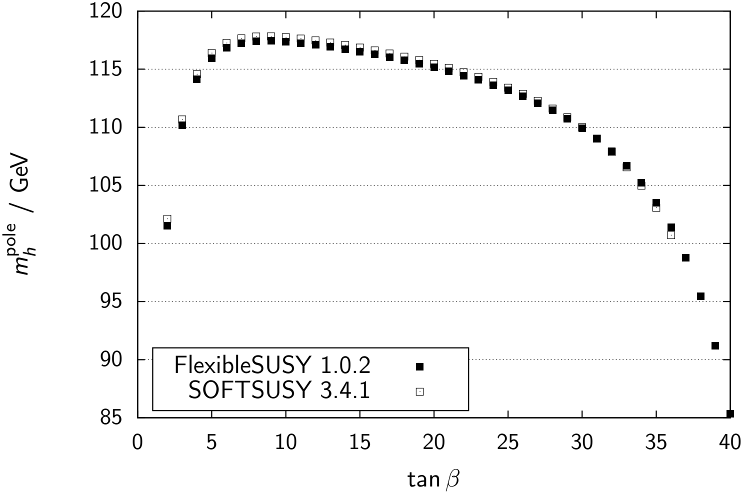
CMSSM run-time comparison





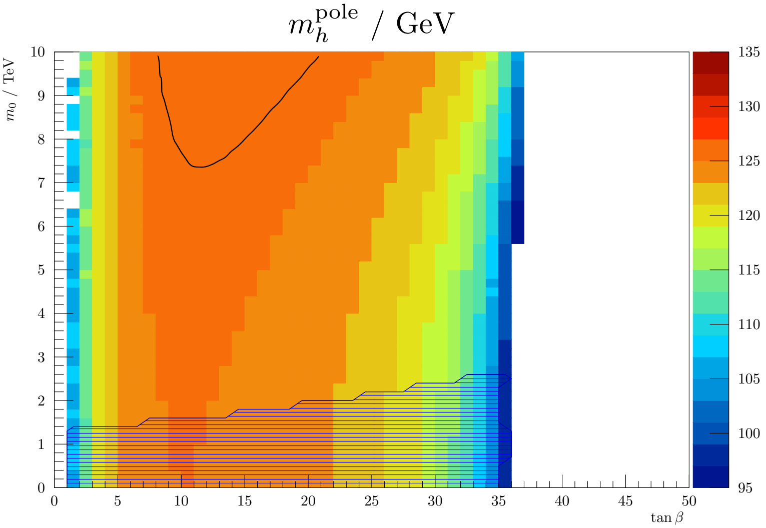
g++ 4.8.0, ifort 13.1.3 20130607

NMSSM Higgs mass comparison



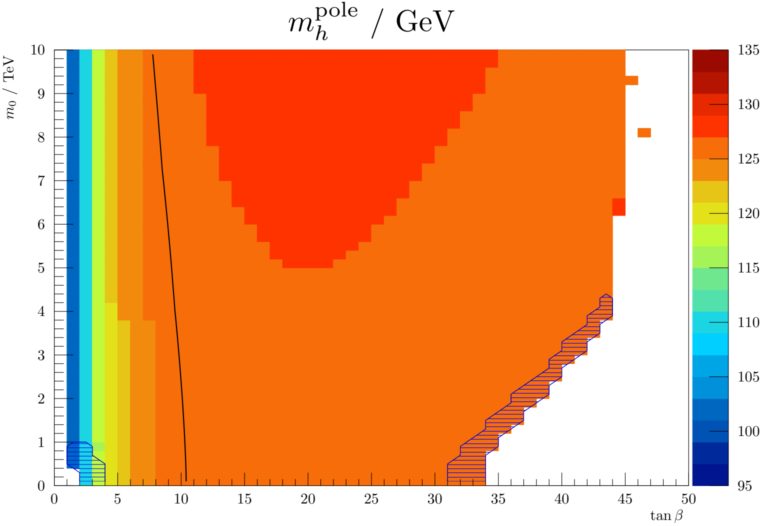
$m_0 = M_{1/2} = -A_0 = 1\,\text{TeV}, \lambda(M_X) = 0.1, \text{sign}\, v_s = +1.$

NMSSM parameter scan



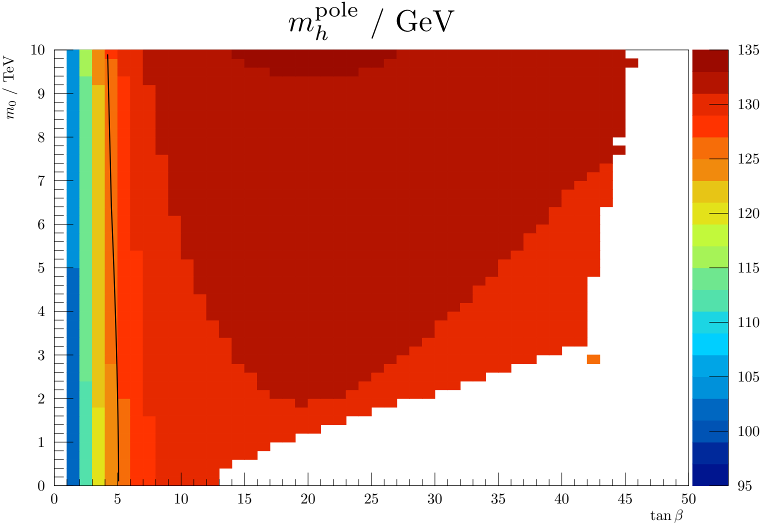
$M_{1/2} = -A_0 = 5\,\text{TeV}, \lambda(M_X) = 0.1, \text{sign}\, v_s = +1.$

USSM parameter scan



$M_{1/2} = A_0 = 5\,\text{TeV}, \lambda(M_X) = 0.1, v_s = 10\,\text{TeV}.$

E_6SSM parameter scan



$M_{1/2} = A_0 = 5\,\mathrm{TeV}, \lambda(M_X) = \kappa(M_X) = 0.1, v_s = 10\,\mathrm{TeV}.$

Conclusions

FlexibleSUSY

- is **modular** (C++ classes, easy to modify, extend and reuse)
- is **fast** (CMSSM run-time ≈ 0.1 s)
- is **precise** (2-loop β_i, 1-loop Σ_f, 1-loop EWSB, leading 2-loop Higgs for MSSM + NMSSM)
- provides **different RGE solvers**
  - two-scale running (adaptive Runge-Kutta)
  - lattice method + variants (prototype status)

Currently supported models:

- CMSSM, NUH-MSSM, NMSSM, SMSSM, USSM, NUHM-E_6SSM, MRSSM, TMSSM, ...

https://flexiblesusy.hepforge.org

Future plans

- FlexibleDecay ($h \to \gamma\gamma, \ldots$)
- some observables ($(g-2)_\mu, \ldots$)
- interface to HiggsBounds
- leading two-loop Higgs mass corrections $\forall$ SUSY models
- support non-SUSY models (FlexibleBSM)
- *automatic* tower of effective field theories
- complex parameters (to study CP violation, etc.)
- ...

Thank you!

NMSSM-SOFTSUSY vs. NMSSM-FlexibleSUSY

NMSSM-SOFTSUSY	NMSSM-FlexibleSUSY
Decay interface for NMHDECAY	FlexibleDecay
optimized couplings	automatically generated couplings
2 EWSB variants	user-defined
BCs via C++	BCS via Mathematica
fast pole masses	fast RGE running
stable code basis	automatically generated
few dependencies	requires Mathematica, SARAH, Boost, etc.
G_μ input	m_W input

NMSSM

$$SU(3)_c \times SU(2)_L \times U(1)_Y$$

$$\mathcal{W}_{\text{NMSSM}} = \lambda S(H_1 H_2) - y^e_{ij}(H_1 L_i)\bar{E}_j - y^d_{ij}(H_1 Q_i)\bar{D}_j - y^u_{ij}(Q_i H_2)\bar{U}_j + \frac{\kappa}{3} S^3$$

$$h_1^0 \to \frac{v_1}{\sqrt{2}} + h_1^0, \qquad h_2^0 \to \frac{v_2}{\sqrt{2}} + h_2^0, \qquad s \to \frac{v_s}{\sqrt{2}} + s$$

mSUGRA-inspired GUT constraint:

$$\begin{aligned} (m_f^2)_{ij}(M_X) &= m_0^2 \delta_{ij} && (f = q, \ell, u, d, e, h_1, h_2), \\ A^f_{ij}(M_X) &= A_0, && (f = u, d, e, \lambda, \kappa), \\ M_i(M_X) &= M_{1/2} && (i = 1, 2, 3). \end{aligned}$$

EWSB output: $\kappa(M_S), v_s(M_S), m_s^2(M_S)$

USSM

$$SU(3)_{\mathrm{c}} \times SU(2)_{\mathrm{L}} \times U(1)_Y \times U(1)'$$

$$\mathcal{W}_{\mathrm{USSM}} = \lambda S(H_1 H_2) - y^e_{ij}(H_1 L_i)\bar{E}_j - y^d_{ij}(H_1 Q_i)\bar{D}_j - y^u_{ij}(Q_i H_2)\bar{U}_j$$

$$h_1^0 \to \frac{v_1}{\sqrt{2}} + h_1^0, \qquad h_2^0 \to \frac{v_2}{\sqrt{2}} + h_2^0, \qquad s \to \frac{v_s}{\sqrt{2}} + s$$

mSUGRA-inspired GUT constraint:

$$\begin{aligned} (m_f^2)_{ij}(M_X) &= m_0^2 \delta_{ij} & \quad (f = q, \ell, u, d, e), \\ A^f_{ij}(M_X) &= A_0, & \quad (f = u, d, e, \lambda), \\ M_i(M_X) &= M_{1/2} & \quad (i = 1, 2, 3, 4). \end{aligned}$$

EWSB output: $m^2_{h_1}(M_S), m^2_{h_2}(M_S), m^2_s(M_S)$

E_6SSM

$$SU(3)_c \times SU(2)_L \times U(1)_Y \times U(1)_N$$

$$\mathcal{W}_{E_6SSM} = \lambda_3 S_3 (H_{13} H_{23}) - y^e_{ij}(H_{13} L_i)\bar{E}_j - y^d_{ij}(H_{13} Q_i)\bar{D}_j - y^u_{ij}(Q_i H_{23})\bar{U}_j \\ + \kappa_{ij} S_3 (X_i \bar{X}_j) + \lambda_{\alpha\beta} S_3 (H_{1\alpha} H_{2\beta}) + \mu'(H'\bar{H}')$$

$$h_1^0 \to \frac{v_1}{\sqrt{2}} + h_1^0, \qquad h_2^0 \to \frac{v_2}{\sqrt{2}} + h_2^0, \qquad s \to \frac{v_s}{\sqrt{2}} + s$$

mSUGRA-inspired GUT constraint:

$$\begin{aligned} (m_f^2)_{ij}(M_X) &= m_0^2 \delta_{ij} && (\forall \text{ scalars, except } h_1, h_2, s), \\ A^f_{ij}(M_X) &= A_0, && (f = u, d, e, \lambda, \kappa), \\ M_i(M_X) &= M_{1/2} && (i = 1, 2, 3, 4). \end{aligned}$$

EWSB output: $m^2_{h_1}(M_S), m^2_{h_2}(M_S), m^2_s(M_S)$