NPointFunctions — an extension to the FlexibleSUSY program, overview and applications

Uladzimir Khasianevich

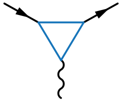
from Institute of Nuclear and Particle Physics @ TU Dresden

Workshop on Automatic Phenomenology @ Institut Henri Poincaré

INSTITUT FÜR KERN- UND TEILCHENPHYSIK

TECHNISCHE UNIVERSITÄT DRESDEN

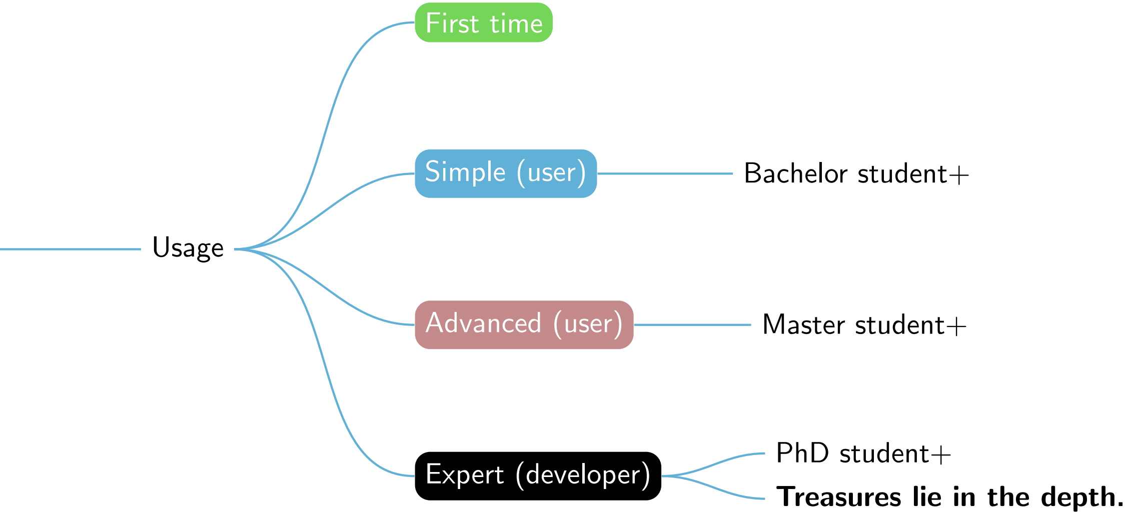
Physics motivations

- Initial study — COMETs: 3,4 — $\mu \to e$ conversion
- Connection — 
  - MEG-II: 1 — $\mu \to e\gamma$
  - Mu3es: 2,4 — $\mu \to 3e$
  - BNL + FNAL: $(25.1 \pm 5.9) \cdot 10^{-10}$ — $(g-2)_\mu$
  - $l_i \to l_j \gamma$
  - $l_i \to l_j l_k l_k^c$
  - $\cdots$ — **So complicated!**
- Models
  - MRSSM
  - $\cdots$ — **How to do?**

Pragmatic motivation

- Challenges
  - Workflow of a phenomenologist (me)
    1: Define $\mathcal{L}_i$
    2: Get vertices, masses, RGE
    3: Calculate observable$_i$
    4: Make parameter scans
- Solutions
  - `SARAH`
  - ↰ `FlexibleSUSY` `v.2.?`
    - ↰ `AMuon`, `EDM`
    - ↰ `BtoS-`, `LToLGamma`
    - ↰ `HiggsDecays`
    - ↰ `NPointFunctions` — **For me and you.**

Implementation



First time

Possible questions — What is the value of quantity Q for parameters P in model M?

- `SARAH`
  - Install
    - With `FlexibleSUSY` — `./install-sarah`
    - Manually — `hepforge`
  - Details ●●
- `FlexibleSUSY`
  - `github` — `README.rst`
    - `conan`
    - Manually — Requirements
  - Choose model M — `model_files/` — M
  - Configure
    - So important?
    - Literature
      - `README.rst`
      - `doc/`
      - articles[1406.2319], [1710.03760]
  - Make
  - Run `models/`M`/run_`M`.x`
    - `LesHouches`[hep-ph/0311123], [0801.0045]
      - P
      - Q
    - `ls-scan`
      - `python`
      - `github`
  - Details ●●●
- `[NPointFunctions]`
  - Install `FeynArts`+`FormCalc` — `FeynInstall` — extra P and Q!
  - Details ●●●
  - **Not aware? Nothing happens.**

Simple changes (user)

Possible questions
- Change model M^* conventions?
- Setup spectrum generator M?
- Choose beloved observable $\in$ quantities?

`SARAH` — Modify model
- Where?
  - Default `Mathematica` path
  - `sarah/`M^*`/`
- What?
  - M^*`.m`
  - `particles.m`
  - `parameters.m`

`FlexibleSUSY` — Modify generator
- `model/`M`/FlexibleSUSY.m`

`NPointFunctions`
- Select observables
  - Output
    - `FlexibleSUSYLowEnergy`
    - `FLHA`[1008.0762]
    - `WCxf`*[1712.05298]
  - Issues
    - Energy above `LowScale`
    - No documentation
- **Simple. When documented.**

FlexibleSUSY?[1406.2319], [1710.03760] The end-user side.

- Define $\mathcal{L}_i$ $\xrightarrow{\text{1) create}}$ #1 #2 #3 $\xrightarrow{\text{2) SARAH}}$ Mathematica code:

#1	#2	#3
M^*.m	parameters.m	particles.m
model definition	additional information	additional information

vertices
masses
2-loop RGE
EWSB

- Setup model$_i$ $\xrightarrow{\text{3) create}}$ #1 #2 $\xrightarrow{\text{4) FlexibleSUSY}}$ C++* code, which:

#1	#2
FlexibleSUSY.m	LesHouches.in.MODEL
restrictions*, extra*, settings	numerical values for the input parameters

solves BVP*
finds mixings
finds masses
finds observables*

A nice place for something new! $\xrightarrow{\text{5) ...}}$

```
FlexibleSUSYObservable`BrLTo3L[Fe@2 -> {Fe@1, Fe@1, bar@Fe@1}, Scalars, 1]
```

Advanced changes (user)

Possible questions
- Add M^* model?
- Add M generator?
- Configure observable Q?

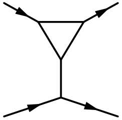
`SARAH`
- Add model as `sarah/`M^*`/`
  - M^*`.m`
  - `particles.m`
  - `parameters.m`

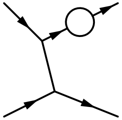
`FlexibleSUSY`
- Add generator as `model_files/`M`/`
  - `FlexibleSUSY.m`
  - `LesHouches.in.`M

`NPointFunctions`
- Change settings in `meta/NPointFunctions/`Q`/`
  - `main.m` — Simple settings
  - `settings.m` — $\rightarrow$More settings?
  - Issues
    - 0 and 1 loop now*
    - No documentation
- **The main level. Refactoring!**

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$





settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

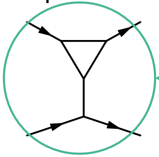
1) Mathematica* code,
2) C++ code, which:
 adds*, new input blocks
 evaluates observable$_i$
 evaluates Wilson coefficients

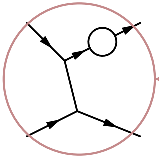
Example: PRELIMINARY

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$

1) Mathematica* code,
2) C++ code, which:
adds*, new input blocks
evaluates observable$_i$
evaluates Wilson coefficients





settings.m*

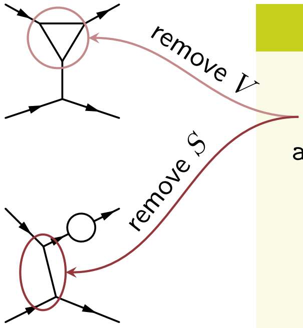
```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

Example: PRELIMINARY

```
topologies[1] = {
 Scalars -> triangleT,
 Vectors -> outSelfT,..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$ 1) Mathematica* code,
  2) C++ code, which:
  adds*, new input blocks
  evaluates observable$_i$
  evaluates Wilson coefficients



settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

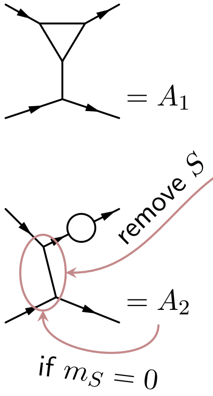
Example: PRELIMINARY

```
diagrams[1, Plus] = {
 Scalars -> {
  triangleT -> {"No V",
   FreeQ[LoopFields@##,
    FeynArts`V]&},..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$ 1) Mathematica* code,

2) C++ code, which:
adds*, new input blocks
evaluates observable$_i$
evaluates Wilson coefficients



settings.m*

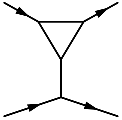
```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

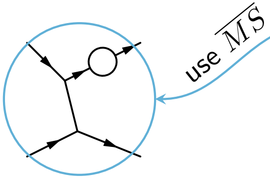
Example: PRELIMINARY

```
diagrams[1, Minus] = {
 Vectors -> {
  outSelfT -> {"No S",
   FreeQ[#, InternalMass[
    FeynArts`S, 5] -> 0]&},..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$ 1) Mathematica* code,
  2) C++ code, which:
  adds*, new input blocks
  evaluates observable$_i$
  evaluates Wilson coefficients





settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

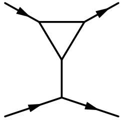
Example: PRELIMINARY

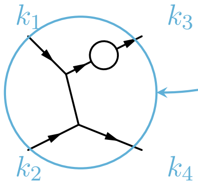
```
regularization[1] = {
  triangleT -> 4,
  outSelfT -> D,..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$

1) Mathematica* code,
2) C++ code, which:
adds*, new input blocks
evaluates observable$_i$
evaluates Wilson coefficients





replace k_2

settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

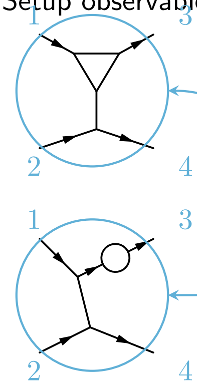
Example: PRELIMINARY

```
momenta[1] = {
 triangleT -> 4,
 outSelfT -> 2,..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$

1) Mathematica* code,
2) C++ code, which:
   adds*, new input blocks
   evaluates observable$_i$
   evaluates Wilson coefficients



settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

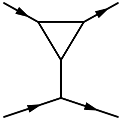
Example: PRELIMINARY

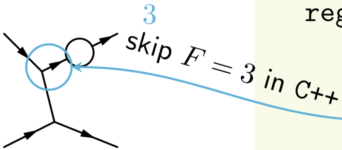
```
order[] = {1, 2, 4, 3};
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$

1) Mathematica* code,
2) C++ code, which:
 adds*, new input blocks
 evaluates observable$_i$
 evaluates Wilson coefficients





settings.m*

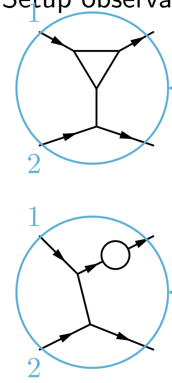
```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

Example: PRELIMINARY

```
sum[1] = {
 outSelfT -> {"Unsame"
  {6, Field[#3, 3]&}},..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$



settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

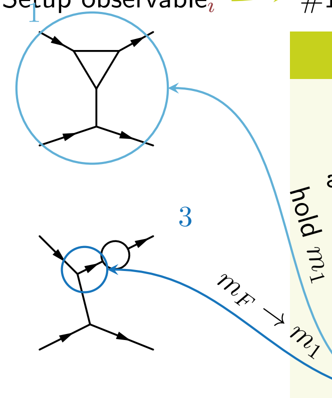
1) Mathematica* code,
2) C++ code, which:
 adds*, new input blocks
 evaluates observable$_i$
 evaluates Wilson coefficients

Example: PRELIMINARY

```
chains[1] = {
 ExceptLoops -> {
  1[k[4|2], ___] -> 0,..
```

NPointFunctions?[2206.00745] The end-user side.

- Setup observable$_i$ $\xrightarrow{\text{5) configure}}$ #1 $\xrightarrow{\text{6) NPointFunctions}}$ 1) Mathematica* code,
  2) C++ code, which:
  adds*, new input blocks
  evaluates observable$_i$
  evaluates Wilson coefficients



settings.m*

```
topologies[LOOPS]
diagrams[LOOPS, TYPE]
amplitudes[LOOPS, TYPE]
regularization[LOOPS]
momenta[LOOPS]
order[]
sum[LOOPS]
chains[LOOPS]
mass[LOOPS]
```

Example: PRELIMINARY

```
mass[1] = {
 triangleT -> {"Hold it"
  {Hold, ExternalMass[1]}},
 ..
```

Expert changes (developer)

- **Possible questions**
  - Better loop libraries? — sh, fortran, C++ — Fine
  - Create tests? — sh, make, Mathematica, C++ — Challenging
  - Add new input LesHouches blocks? — C++ — Complex
  - Add more observables? — sh, Mathematica, C++ — Doable$^{(1\text{-}2d)}$
  - Modify existing Q? — sh, Mathematica, C++ — Easy
  - Need more settings for Q?

`SARAH`

- `FlexibleSUSY`
  - Add libraries
    - Documentation
      - `doc/add_loop_library.rst`
      - `loop_library_interface.hpp`
    - Modify in places
      - `src/loop_libraries/`
      - `configure` — Automatize
  - Create tests
    - github

- `NPointFunctions`
  - Package structure in `meta/`
    - `NPointFunctions/`Q`/`
    - `NPointFunctions.m` — `npftex` — github
    - `NPointFunctions/`

Heart of `NPointFunctions` — observable Q

- `/`Q`/`
  - `[type.m]`
    - Defines
      - What to put in `model/`M`/FlexibleSUSY.m`
      - `C++` namespace/filename for Q
      - `Mathematica` Q arguments
  - `observable.m`
    - `FlexibleSUSY C++` defaults for Q
    - Loaded by — `meta/Observables.m`
  - `write.m`
    - Outputs to `SLHA` and `FLHA`
    - Gently screams at you
    - Loaded by — `meta/WriteOut.m`
  - `class.m`
    - Fills `C++` templates — `templates/npointfunctions/` { `.hpp` `.cpp`
    - Moves them to `models/`M`/`
    - Loaded by — `meta/FlexibleSUSY.m`
  - `[main.m]`
    - How `NPointFunctions.m` should be used — Add n loops
    - Extracts Wilson coefficients
  - `[settings.m]`
    - Contains observable settings
    - Loaded by — `meta/NPointFunctions/internal.m`
  - `[librarylink.m]`
    - How to call observable via `Mathematica` call
    - Loaded by — `meta/FSMathLink.m`
  - **Modular and lazy.**

Heart of `NPointFunctions` — C++ converter

- `.m`
  - Regimes
    - Simple settings
      - Tests in `test/`
        - SM: $h \to h$, $d_i \to d_i$
        - $MSSM$: $h_i \to h_j$, $d_i \to d_i$
        - $MRSSM$: $l_i(Al) \to l_j(Al)$
    - Observable settings
      - Q
        - $l_i \to l_j$ conversion
        - $l_i \to l_j l_k l_k^c$
        - $d_i \to d_j l_k l_k^c$
  - Purpose in life
    - Convert `FeynArts`/`FormCalc`*to C++
      - In two steps
    - Stores output
      - `npftex`
  - **One to rule them all.**

Heart of `NPointFunctions` — settings parser

- `/`
  - `internal.m`
    - Loads*
      - `FeynArts`/`FormCalc`
      - `$Q$/settings.m`
    - Runs usual `FeynArts`/`FormCalc` sequence
  - `type.m` and `tools.m` — Contain `Mathematica` patterns and some tools
  - `settings.m`
    - Defines which settings exist
    - Defines how each setting should be applied
  - `tree.m`
    - How topologies/amplitudes/diagrams are combined
    - What to *remove*
  - `topologies.m` — Connects nice names with keys
  - `chains.m`
    - Chains extractor
    - Embedded language definition
  - `mass.m` — How mass in chains*/loop integral/amplitudes is treated
  - `rules.m` — From `FeynArts`/`FormCalc` to `FlexibleSUSY` notation
  - **Flexible settings.**

Applications

- GNM
  - Grimus-Neufeld model
    - THDM+1ν+$\approx Z_2$
    - **cLFV restricts scalars.**
  - Authors
    - V.Dūdėnas, Th.Gajdosik
    - **U.Kh.**, W.Kotlarski, D.Stöckinger
  - Preprint
    - 2206.00661
- MRSSM
  - Minimal R-Symmetric Supersymmetric model
    - Distinct from MSSM
    - **BSM enhancements.**
  - Authors
    - **U.Kh.**, W.Kotlarski
    - D.Stöckinger, H.Stöckinger-Kim
  - Preprint
    - [soon]
- LQ
  - Scalar Leproquarks S_1 and R_2
    - Simplest extensions
    - **Couplings interplay.**
  - Authors
    - **U.Kh.**, D.Stöckinger
    - H.Stöckinger-Kim, J.Wünsche
  - Preprint
    - [soon]