HYPER ISO

A general BSM calculator for flavour observables

Théo Reymermier

April 17 2025

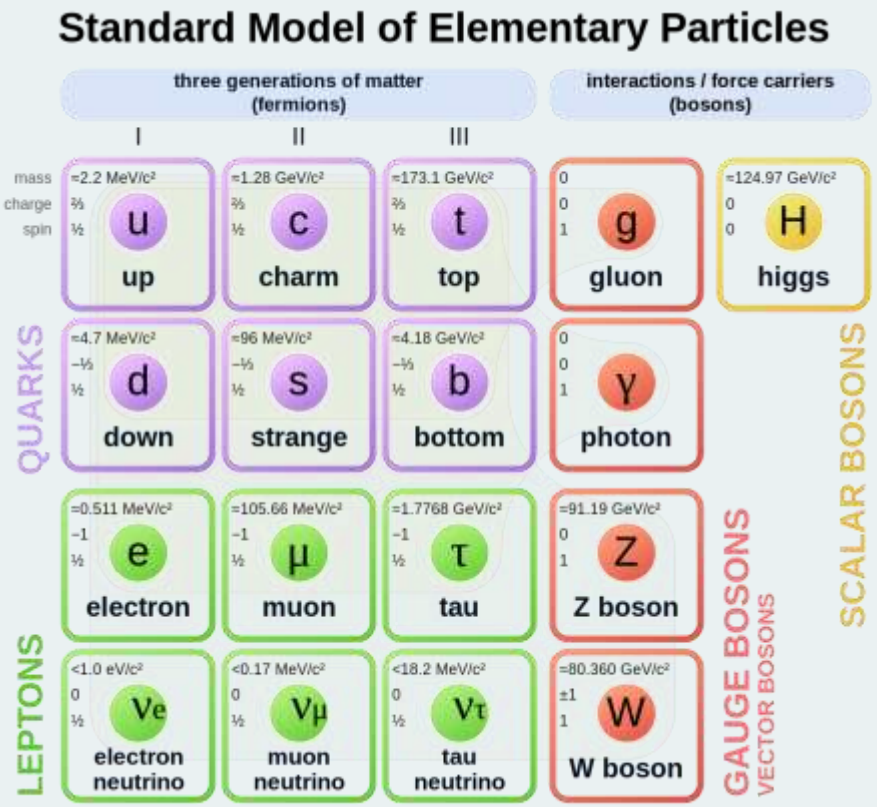
PhD days 2025

Hyperiso team

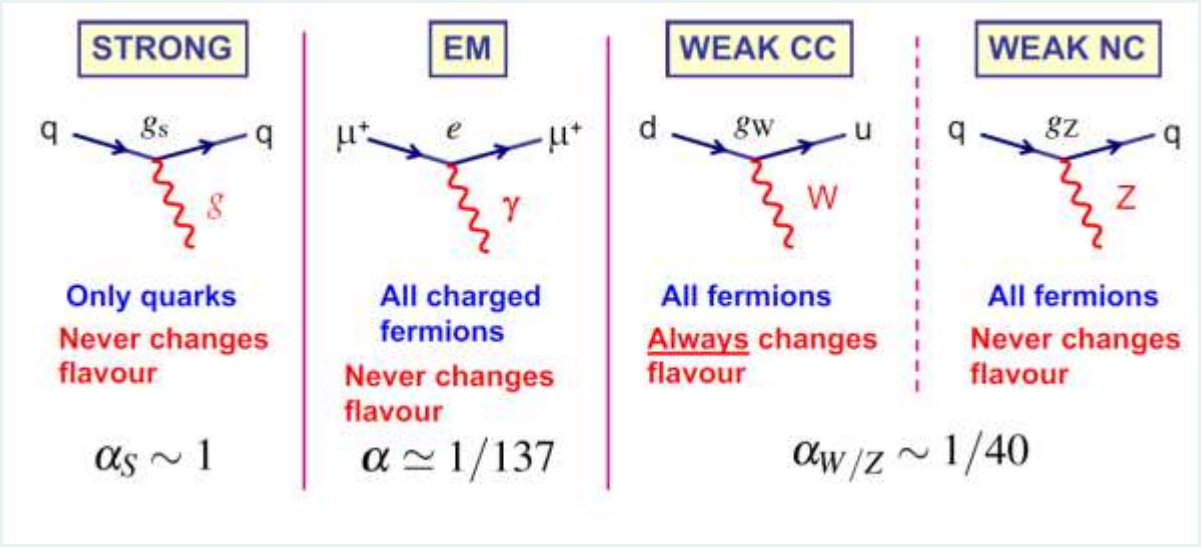
Laboratory : Institut de Physique des Deux Infinis de Lyon (IP2I)
Equipe Théorie

- Nazila Mahmoudi (PhD supervisor)
- Niels Fardeau (PhD student) ⟸ Developer
- Siavash Neshatpour (Post-doctoral researcher)
- Théo Reymermier (PhD student) ⟸ Developer

The Standard Model



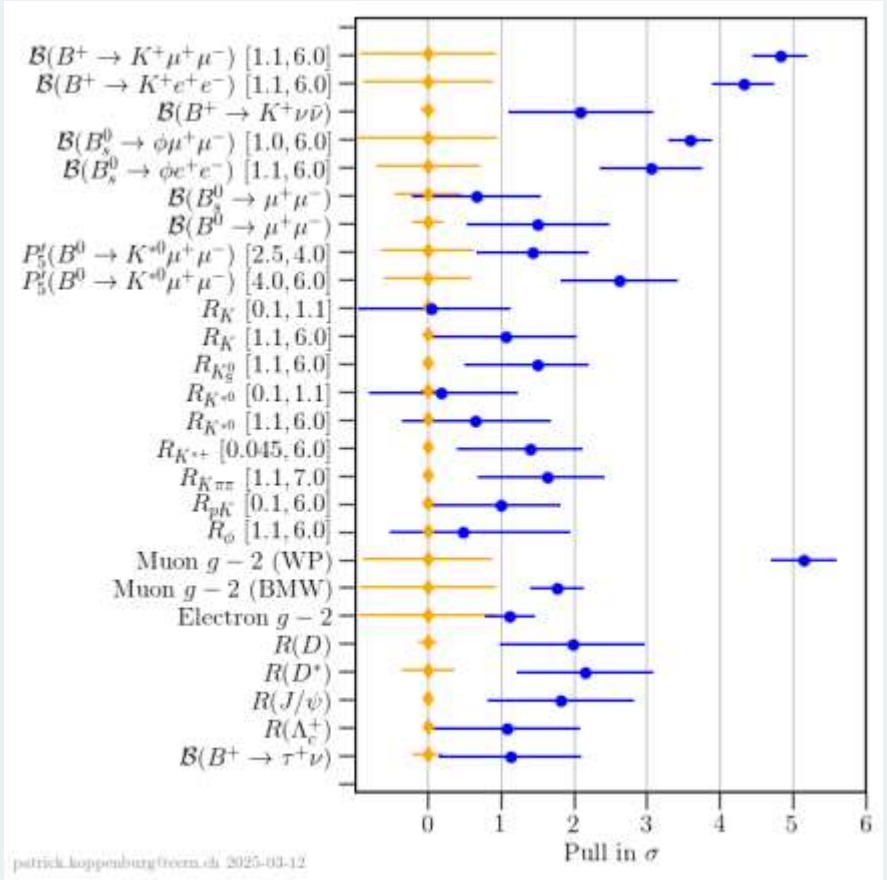
Particles of the Standard Model (Physique au-delà du modèle standard, Wikipedia).



Fondamental interactions in the Standard Model (Part III Particle Physics, Dr Lester, Cambridge University).

Limitation of the Standard Model

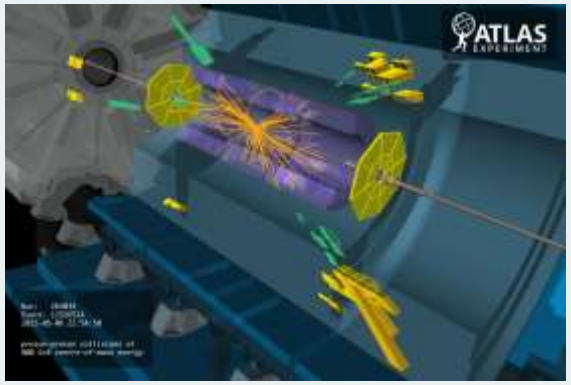
- Number and values of SM parameters ?
- Dark Matter ?
- Flavor anomalies ?
- Why three generations ?
- Is there new particles ? New interactions ?
- Some theory weakness ?



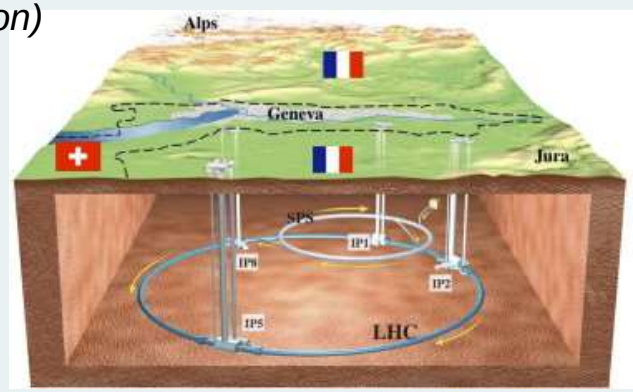
Patrick Koppenburg (2025) – Flavor anomalies

PhD objective

High energy physics



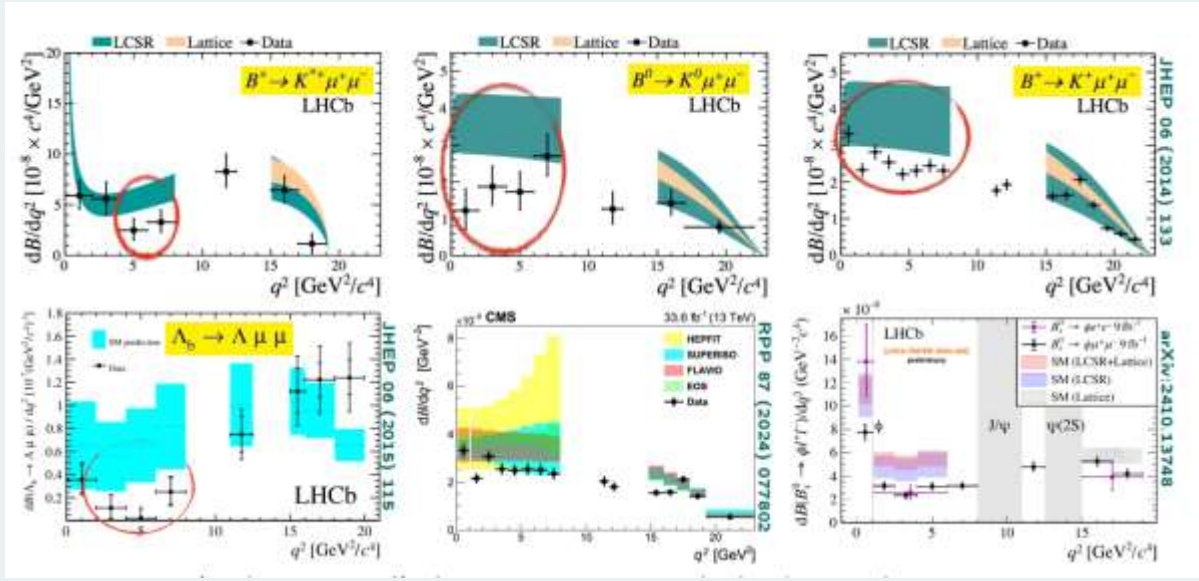
Visualization of ATLAS data in a pp collision at 9 TeV (ATLAS collaboration)



Scheme of the operation of LHC at the French-Swiss frontier (The large hadron collider, H. Burkhardt)

PhD project !

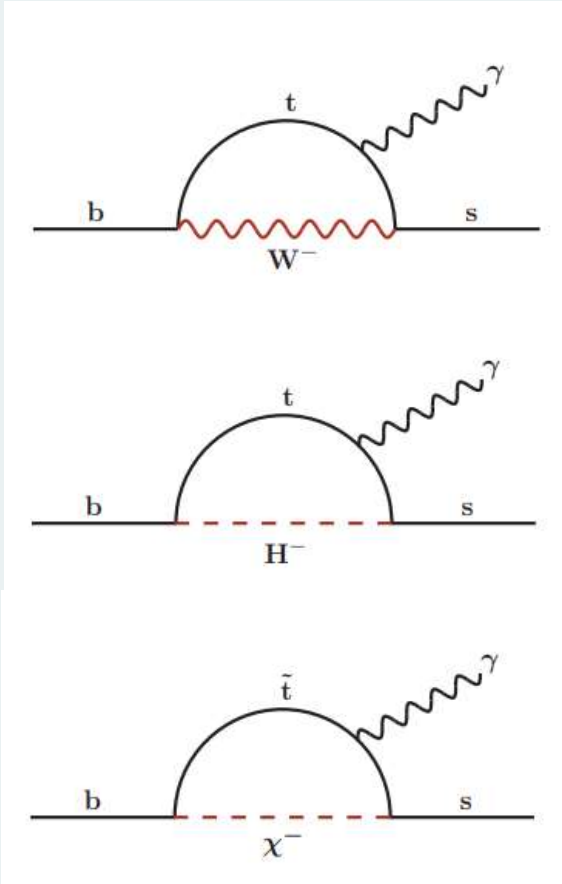
Low energy physics (High intensity)



Tension between experimental results and SM predictions in the $b \to s\mu^+\mu^-$ decay.

Effective Theory

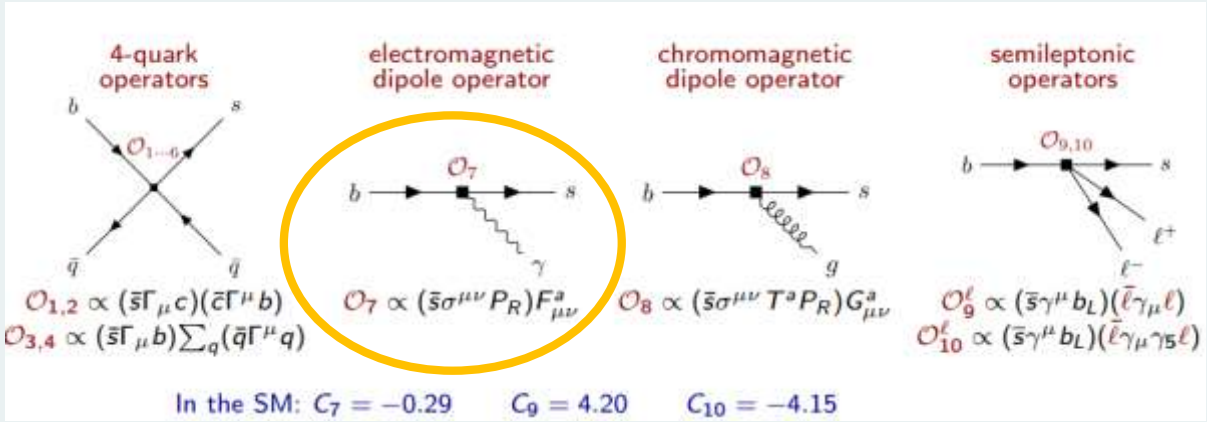
High energy physics



Feynman diagrams contributing in the decays of the B meson, in SUSY models (Update on SuperIso and interpretation of the flavour anomalies, Nazila Mahmoudi)

PhD project !

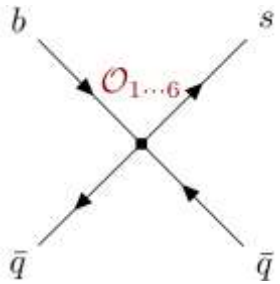
Low energy physics (High intensity)

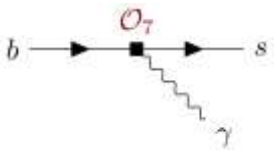


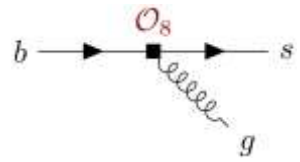
Feynman diagrams in effective theory for B meson decays (Model independant analysis of bsll data, Nazila Mahmoudi)

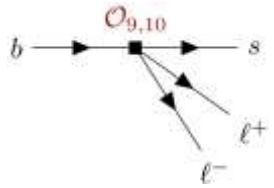
Calculations in flavour physics

$$L_{(B)SM} \to H_{\text{eff}}(b \to sX) = -\frac{4G_F}{\sqrt{2}} V_{tb} V_{ts}^* \sum_{i=1}^{10} C_i(\mu) O_i(\mu)$$

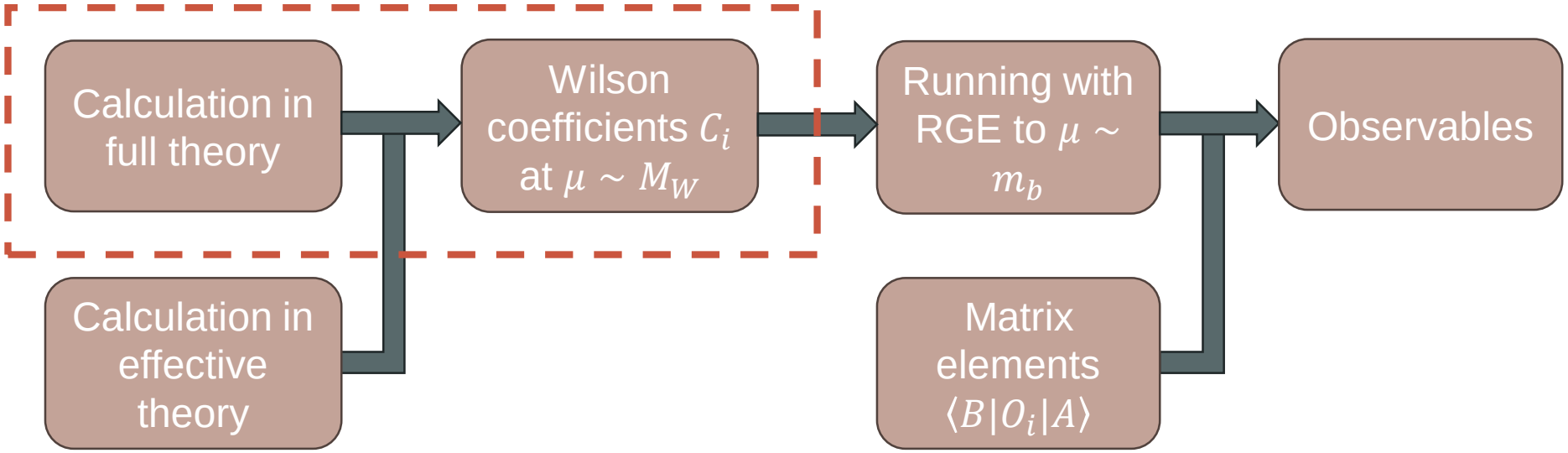








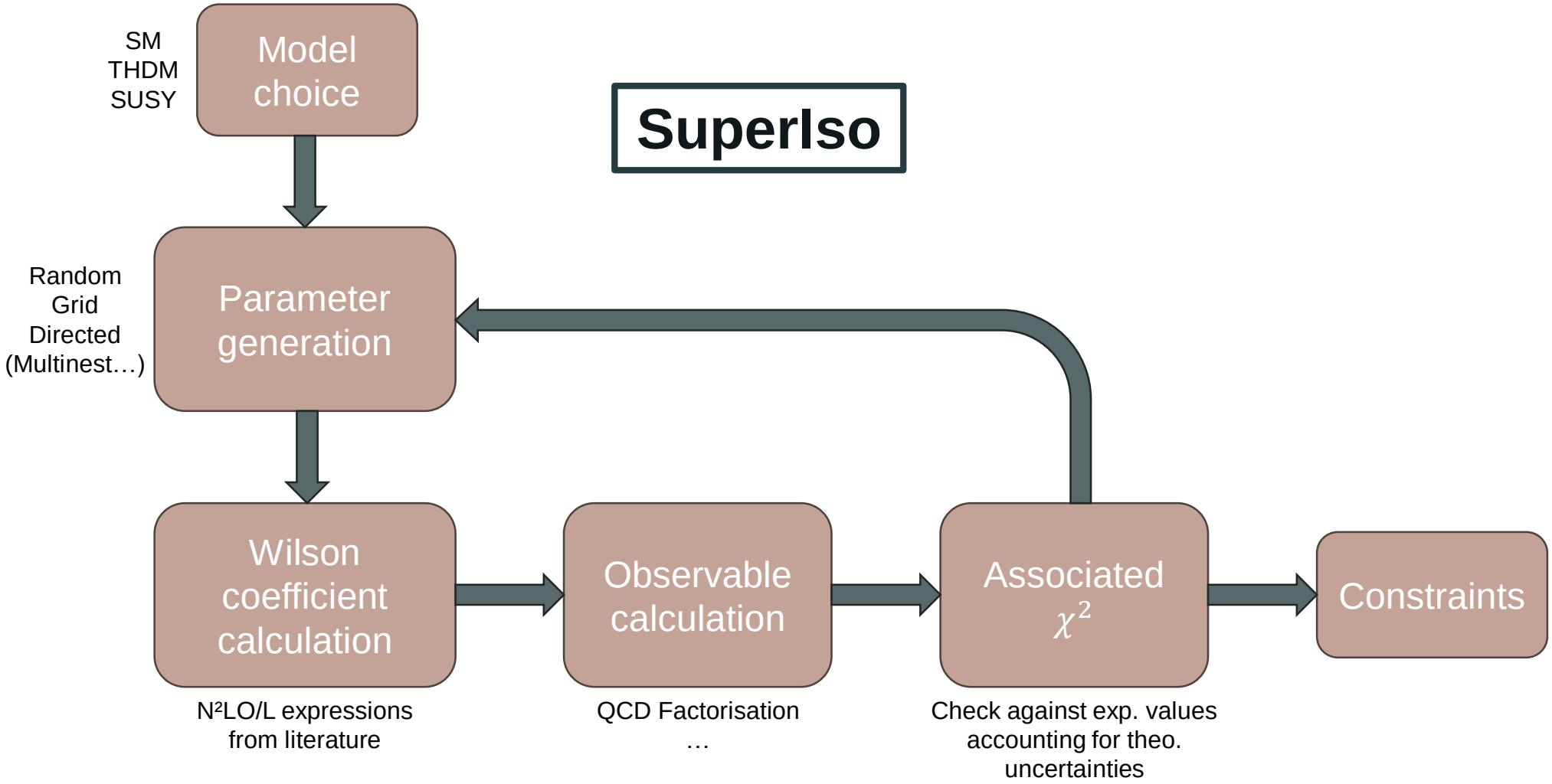
For each BSM model



Need for automated calculations !

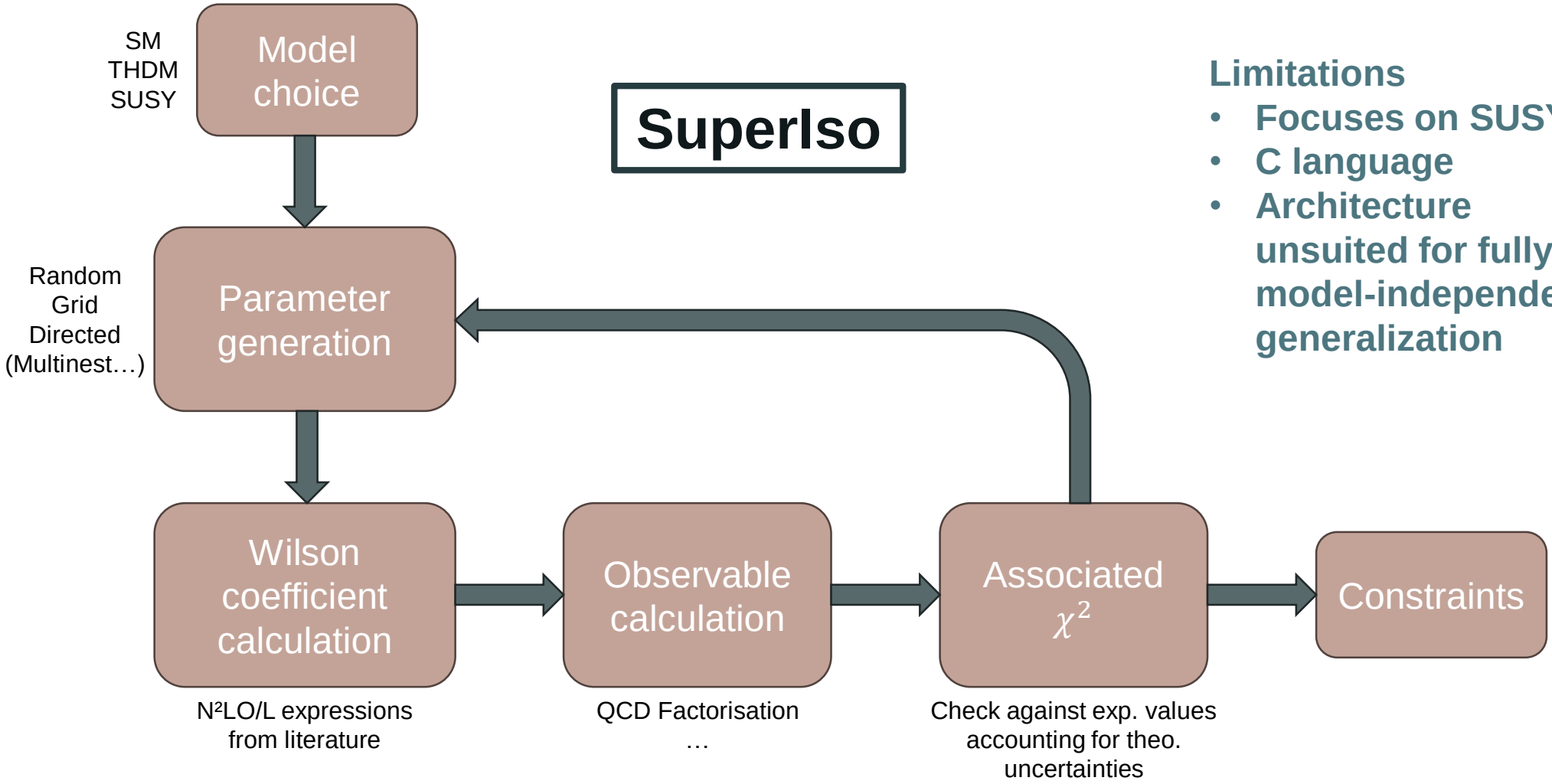
SuperIso

F. Mahmoudi, Comput. Phys. Commun. **178**, 745 (2008) [0710.2067]



SuperIso

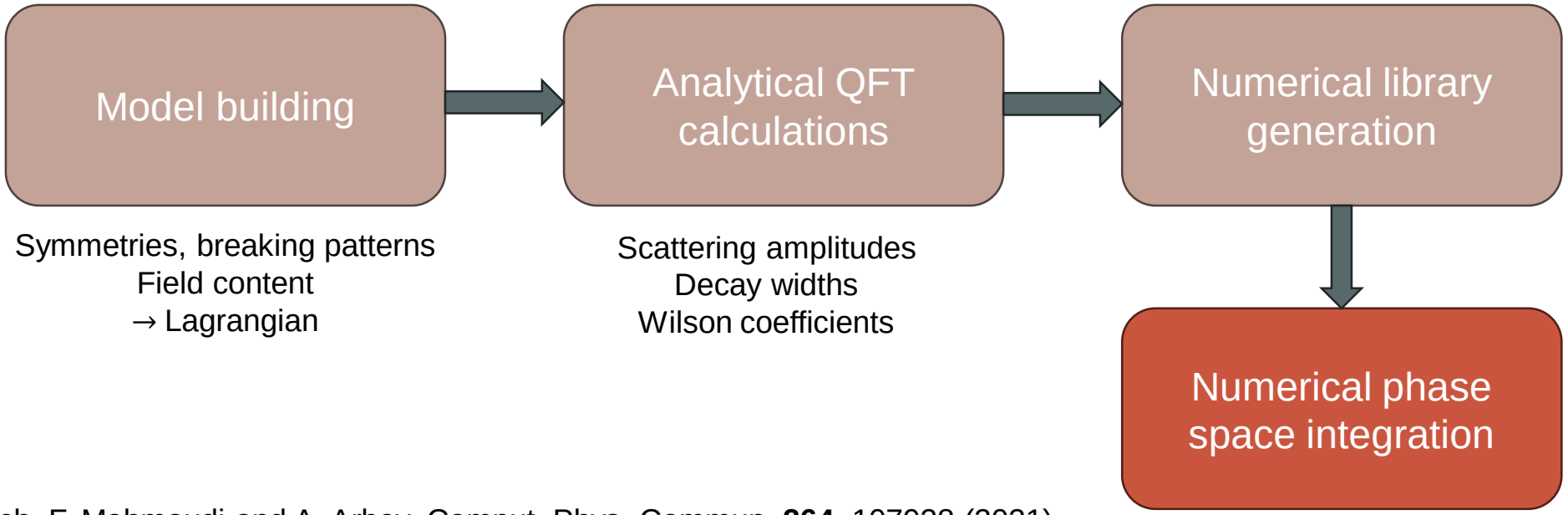
F. Mahmoudi, Comput. Phys. Commun. **178**, 745 (2008) [0710.2067]



Limitations

- **Focuses on SUSY**
- **C language**
- **Architecture unsuited for fully model-independent generalization**

MARTY (1.5)



G. Uhlrich, F. Mahmoudi and A. Arbey, Comput. Phys. Commun. **264**, 107928 (2021)

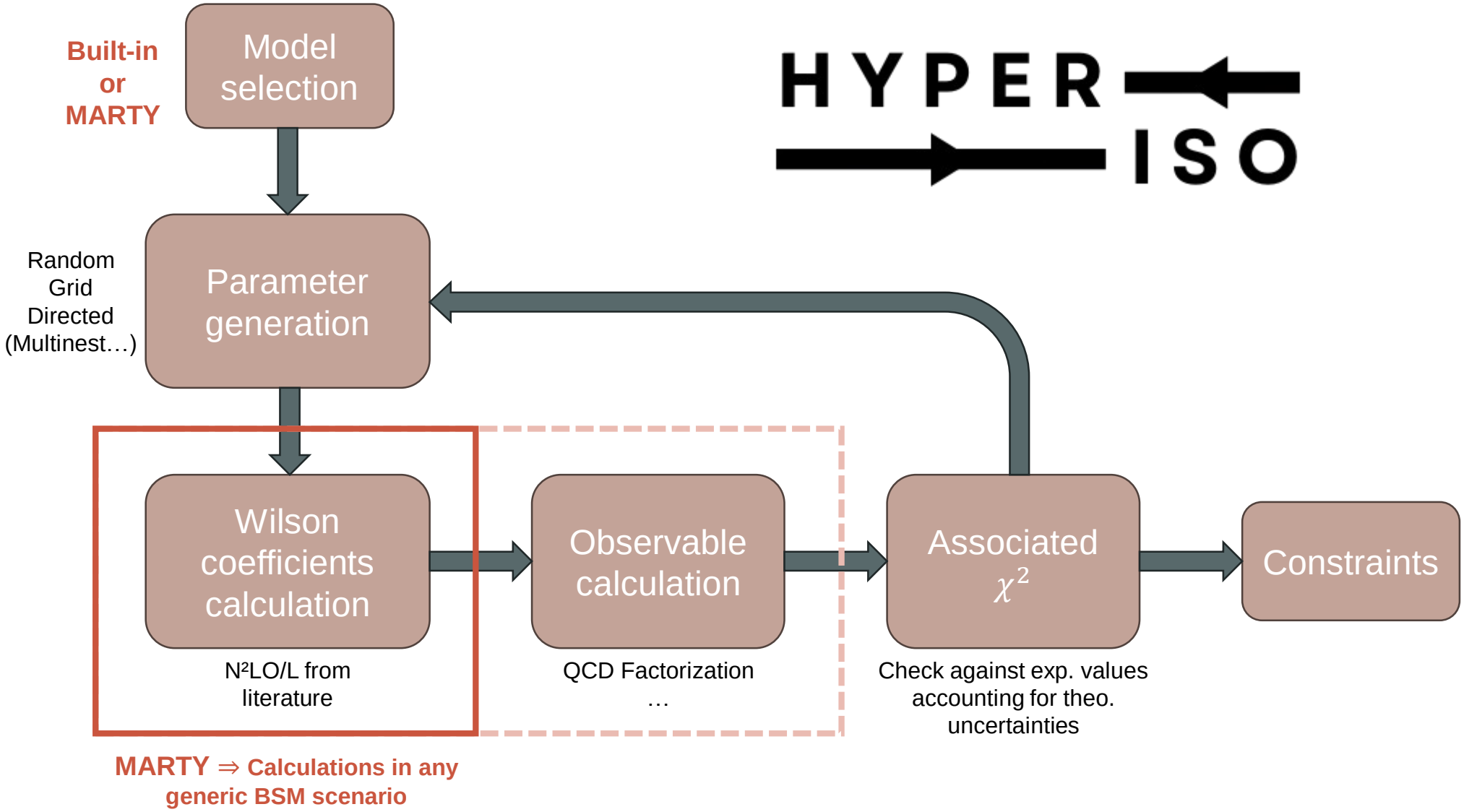
HyperIso

C++ overhaul of SuperIso (C99). Workflow diagram unchanged.

Upgrades :

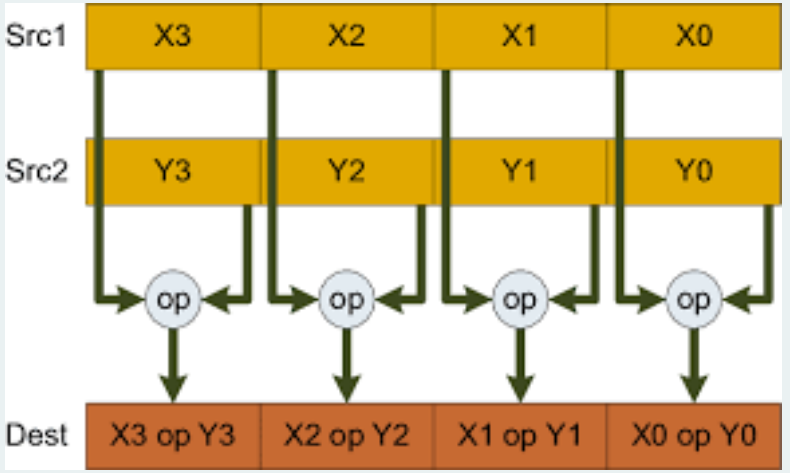
- Modern C++ features
- Clear software architecture
- Various optimizations
- Reproduces SuperIso's behavior for the calculation of Wilson Coefficients (in SM, THDM and SUSY) and observables (WIP)
- Greater flexibility and model-independence
- Several UIs to fit all needs (C++ / Python / GUI)

HyperIso – MARTY



Optimizations

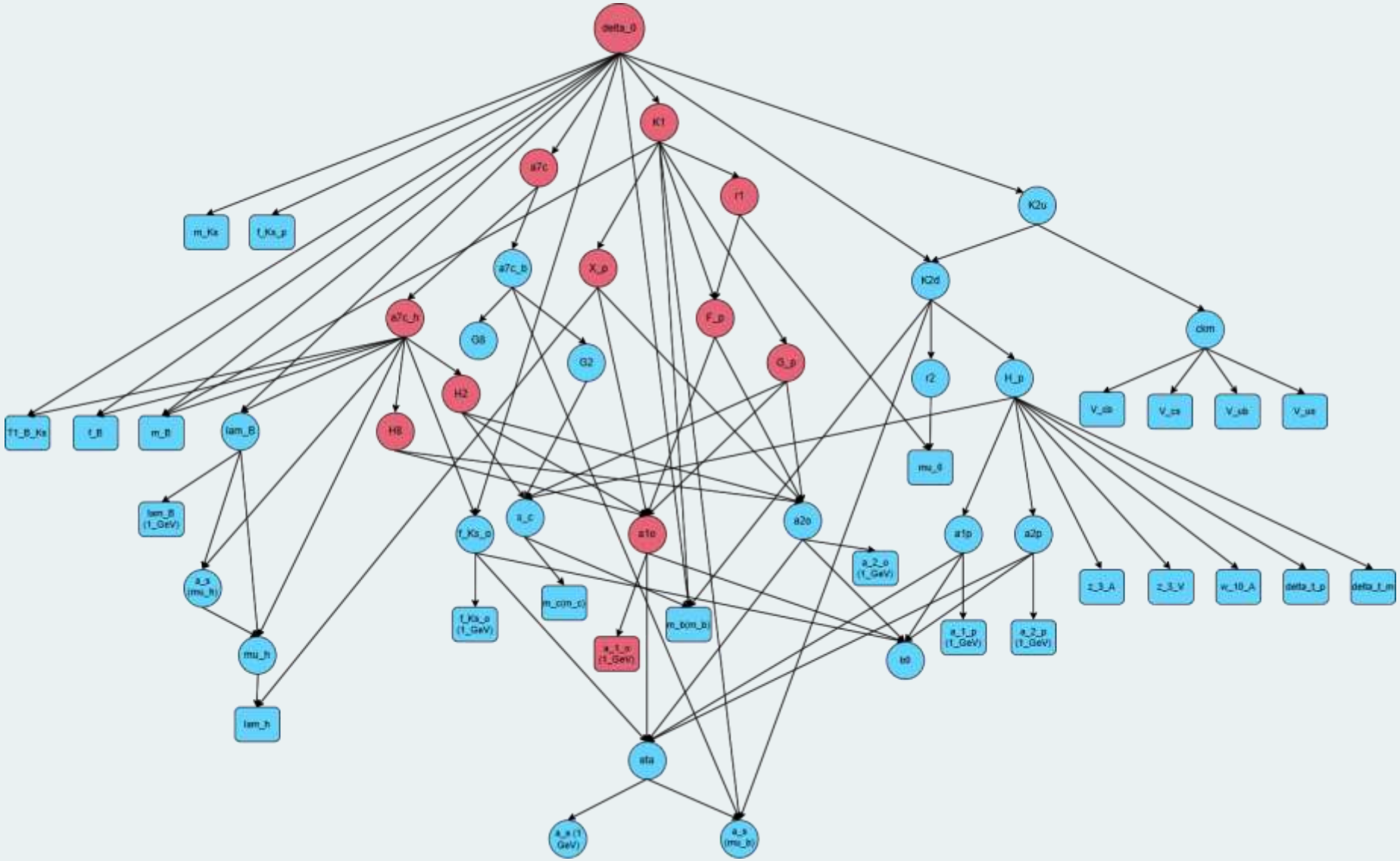
- Sparse matrix optimization
- Eigen
- SIMD optimization (AVX, MMX, etc.)
- Parallelization



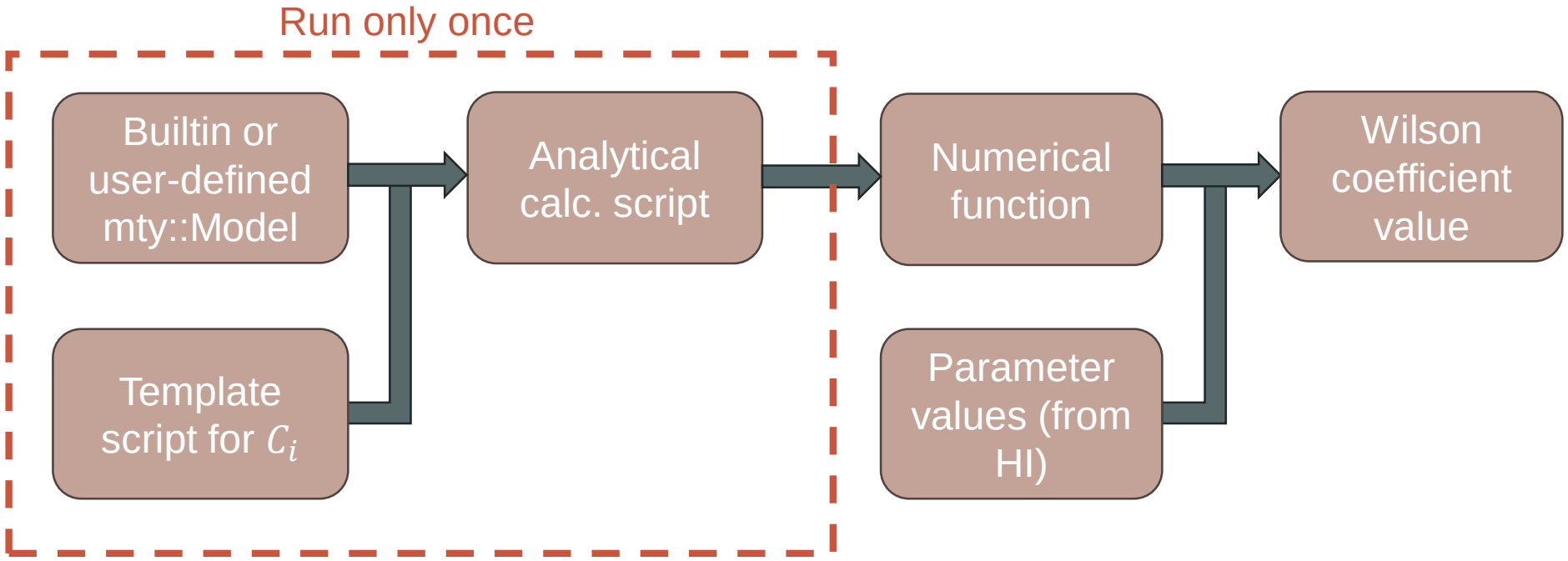
$$
\begin{aligned}
\vec{C}^{(0)\mathrm{eff}}(\mu_b) &= U^{(0)}\vec{C}^{(0)\mathrm{eff}}(\mu_W)\,, \\
\vec{C}^{(1)\mathrm{eff}}(\mu_b) &= \eta\left[U^{(0)}\vec{C}^{(1)\mathrm{eff}}(\mu_W) + U^{(1)}\vec{C}^{(0)\mathrm{eff}}(\mu_W)\right]\,, \\
\vec{C}^{(2)\mathrm{eff}}(\mu_b) &= \eta^2\left[U^{(0)}\vec{C}^{(2)\mathrm{eff}}(\mu_W) + U^{(1)}\vec{C}^{(1)\mathrm{eff}}(\mu_W) + U^{(2)}\vec{C}^{(0)\mathrm{eff}}(\mu_W)\right]
\end{aligned}
$$

$$
U_{kl}^{(n)} = \sum_{j=0}^{n}\sum_{i=1}^{9} m_{kli}^{(nj)}\eta^{a_i-j}\,.
$$

Graph-based observable evaluation

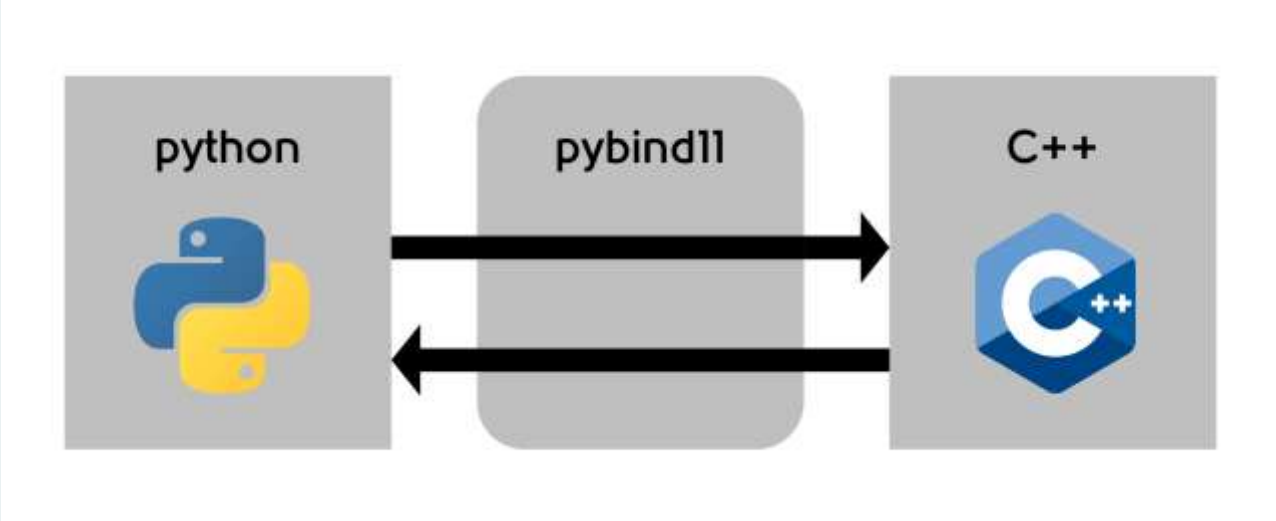


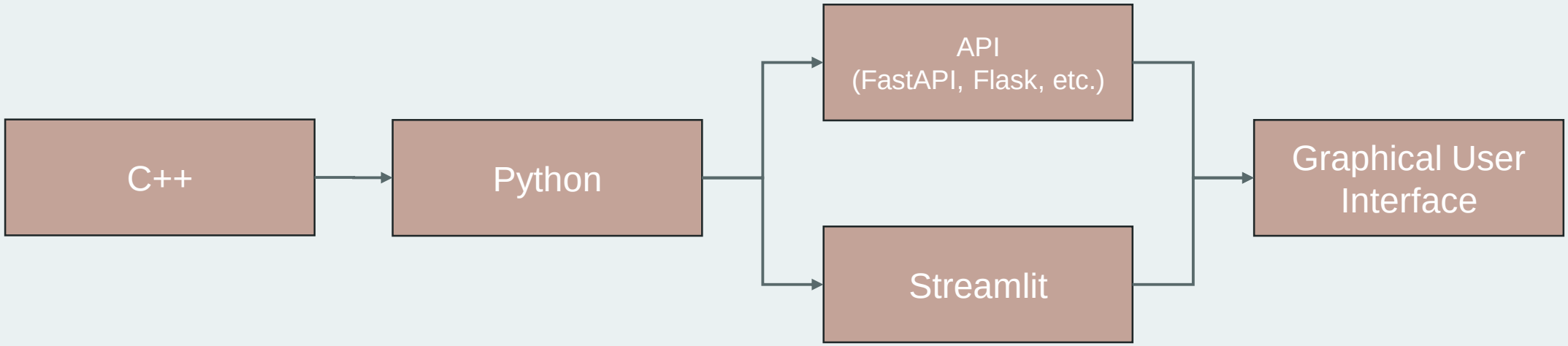
MARTY Interfacing



Python binding

- Parameters management
- Wilson calculation
- Observable calculation
- χ^2 calculation





C++ Interface

Supports multithreading !

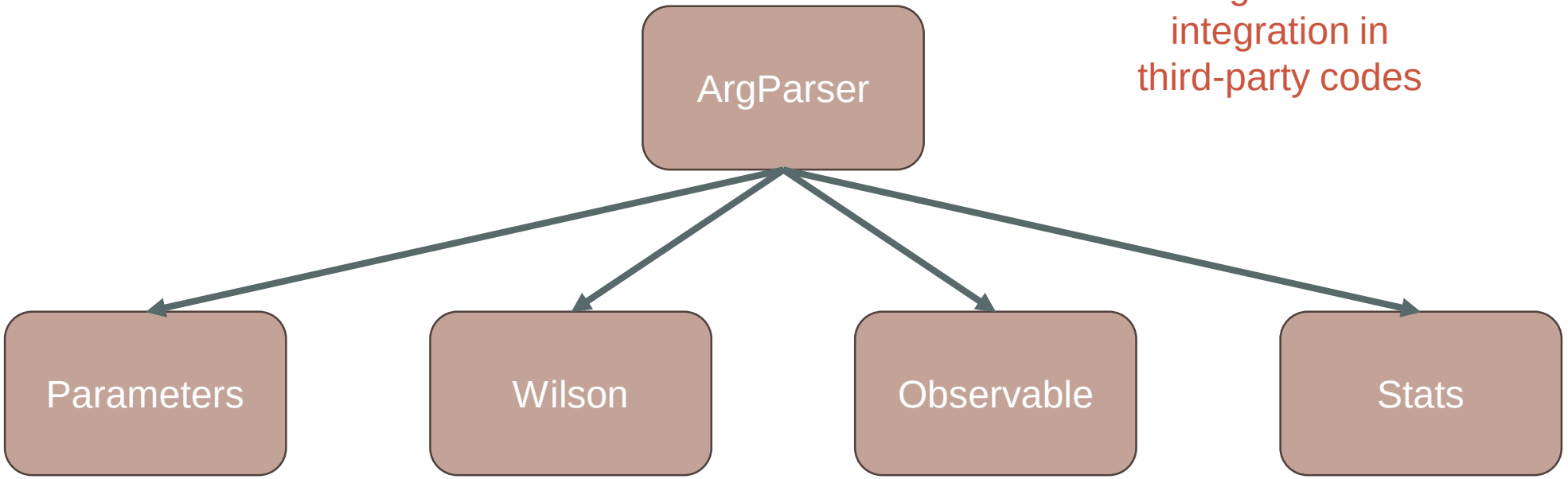
```cpp
int main() {
    auto mm = MemoryManager::GetInstance();
    mm->init("Test/InputFiles/testInput.flha", Model::SM); // Initialize program manager with LHA file

    auto wi = WilsonInterface(); // Initialize interface and build the required groups
    wi.build(
        {WGroup::B, WGroup::BPrime},                          // Coefficient groups
        2 * Parameters::Get(ParameterType::SM, "MASS", 24),   // Matching scale
        QCDHelper::mass_b_1S() / 2,                           // Hadronic scale
        QCDOrder::NNLO                                        // QCD Order
    );

    // Retrieve coefficient values
    std::ofstream fs {"out.txt"};
    fs << "C7 matching (LO + NLO + NNLO) : " << wi.getFullMatchingCoefficient(WGroup::B, WCoef::C7, QCDOrder::NNLO)     << '\n';
    fs << "C7 hadronic (LO + NLO + NNLO) : " << wi.getFullRunCoefficient(WGroup::B, WCoef::C7, QCDOrder::NNLO)          << '\n';
    fs << "CP7 matching (LO) : "             << wi.getFullMatchingCoefficient(WGroup::BPrime, WCoef::CP7, QCDOrder::LO) << '\n';
    fs << "CP7 hadronic (LO) : "             << wi.getFullRunCoefficient(WGroup::BPrime, WCoef::CP7, QCDOrder::LO)      << '\n';
    fs.close();

    return 0;
}
```

Python Interface

```python
def main():
    mm = MemoryManager()
    mm.init("Test/InputFiles/testInput.flha", Model.SM) # Initialize program manager with LHA file

    wi = WilsonInterface() # Initialize interface and build the required groups
    wi.build(
        [WGroup.B, WGroup.BPrime],                    # Coefficient groups
        2 * Parameters(ParameterType.SM)("MASS", 24),  # Matching scale
        QCDHelper.mass_b_1S() / 2,                    # Hadronic scale
        QCDOrder.NNLO                                 # QCD Order
    )

    # Retrieve coefficient values
    with open("out.dat", "w") as f:
        f.write("C7 matching (LO+NLO+NNLO) : "  + wi.get_full_matching_coefficient(WGroup.B, WCoeff.C7, QCDOrder.NNLO))
        f.write("C7 hardronic (LO+NLO+NNLO) : " + wi.get_full_run_coefficient(WGroup.B, WCoeff, QCDOrder.NNLO))
        f.write("CP7 matching (LO) : "          + wi.get_full_matching_coefficient(WGroup.BPrime, WCoeff.CP7, QCDOrder.LO))
        f.write("CP7 hardronic (LO) : "         + wi.get_full_run_coefficient(WGroup.BPrime, WCoeff.CP7, QCDOrder.LO))

    return 0
```

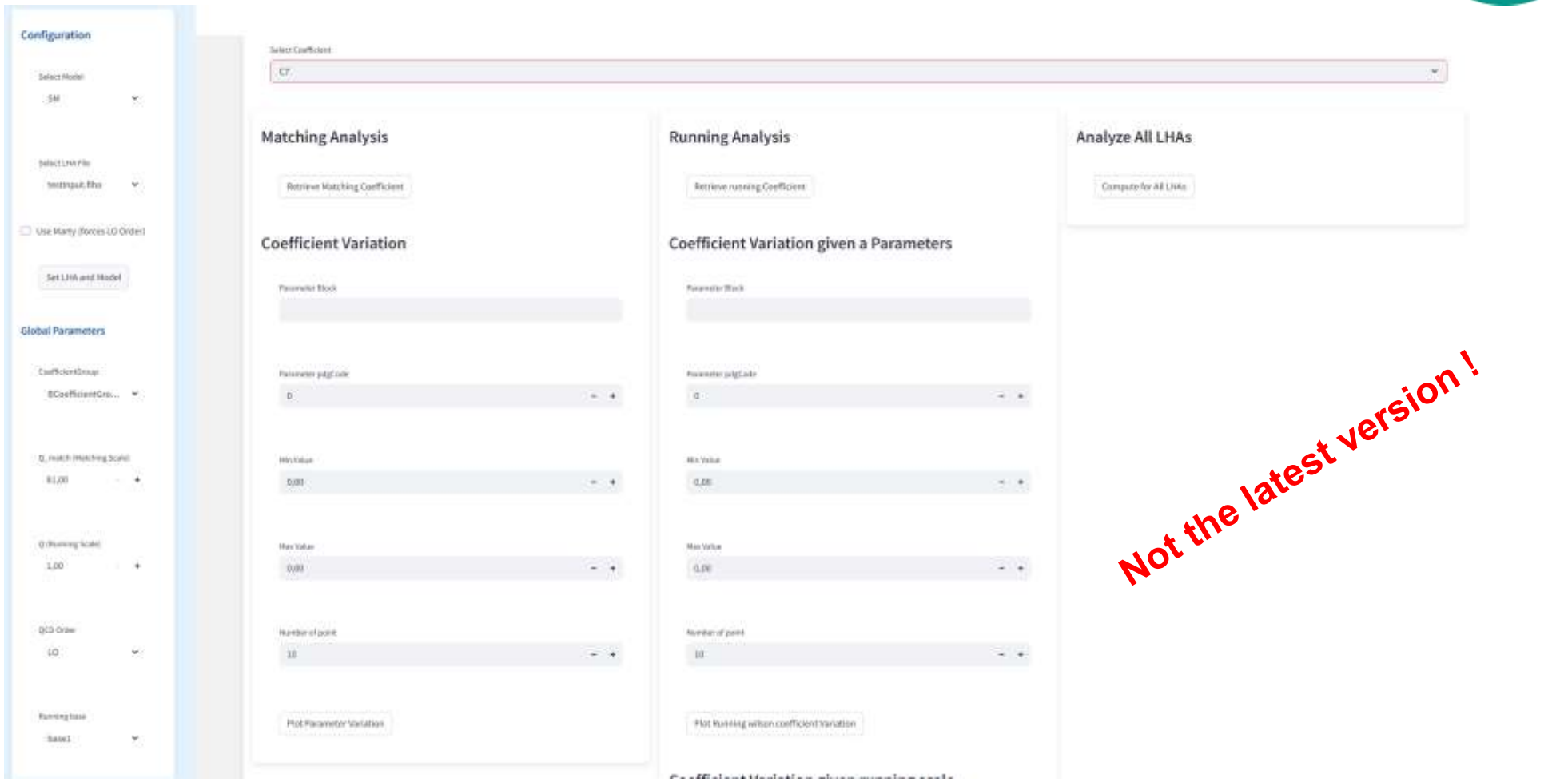
Terminal Interface

Straightforward integration in third-party codes




```
nfardeau@DESKTOP-U6KU7DG:~/Hyperiso/Hyperiso/build$ ./UserInterfaceLib/main_user wilson -m SM -w C7 -q 40.4 -Q 2.37 -if Test/InputFiles/testInput.flha -o LO
Coefficient C7 at Q_match = 40.4: -0.202381 + 0i
Coefficient C7 at Q = 2.37: -0.343725 + 0i
nfardeau@DESKTOP-U6KU7DG:~/Hyperiso/Hyperiso/build$ ./UserInterfaceLib/main_user wilson -m SM -w CP7 -q 40.4 -Q 2.37 -if Test/InputFiles/testInput.flha -o LO
Coefficient CP7 at Q_match = 40.4: -0.00606445 + 0i
Coefficient CP7 at Q = 2.37: -0.0036773 + 0i
```

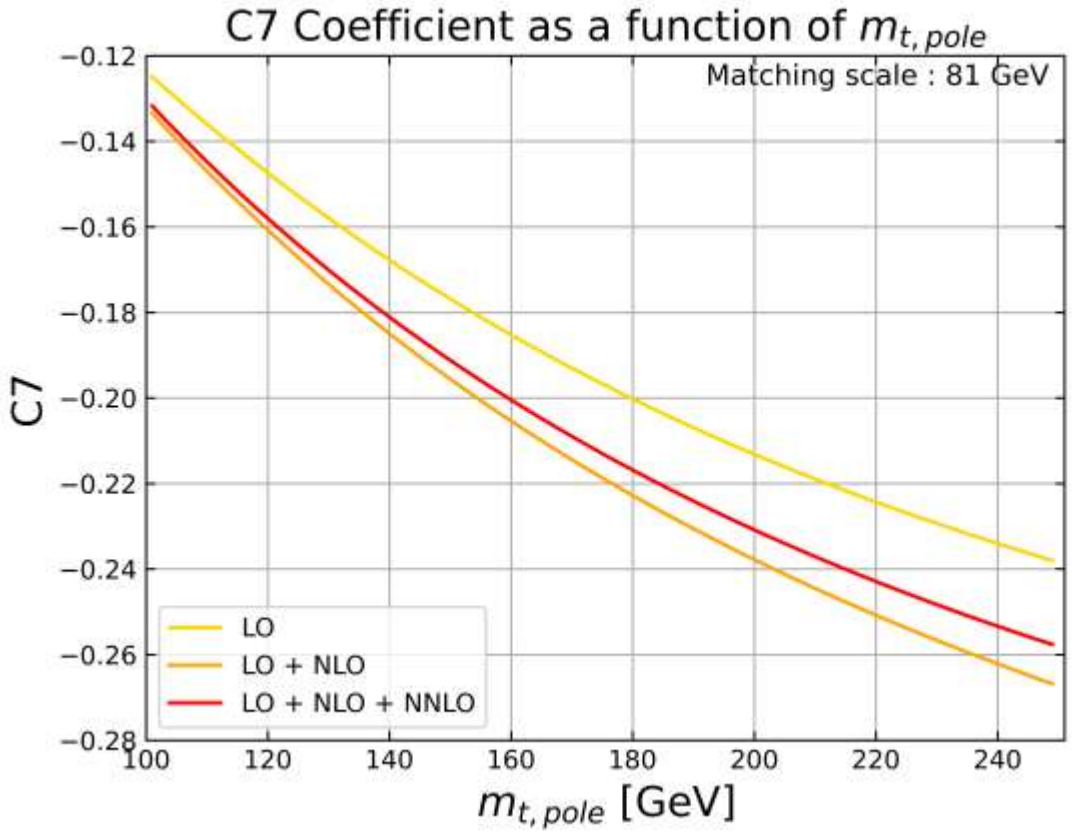
Graphical Interface

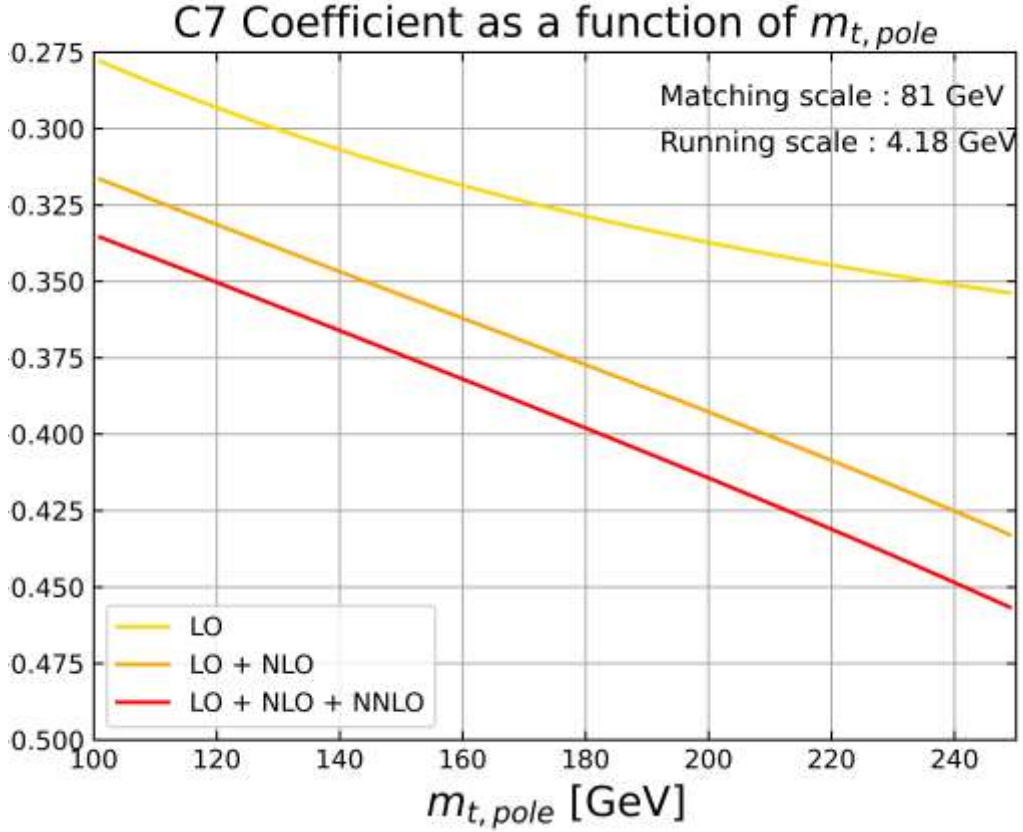


Not the latest version !

Example plots

Unphysical





Future improvements

Statistical side

- Extend statistical calculations to generic nuisance distributions / likelihoods
- Builtin interface with scanning softwares (e.g. BSMArt)

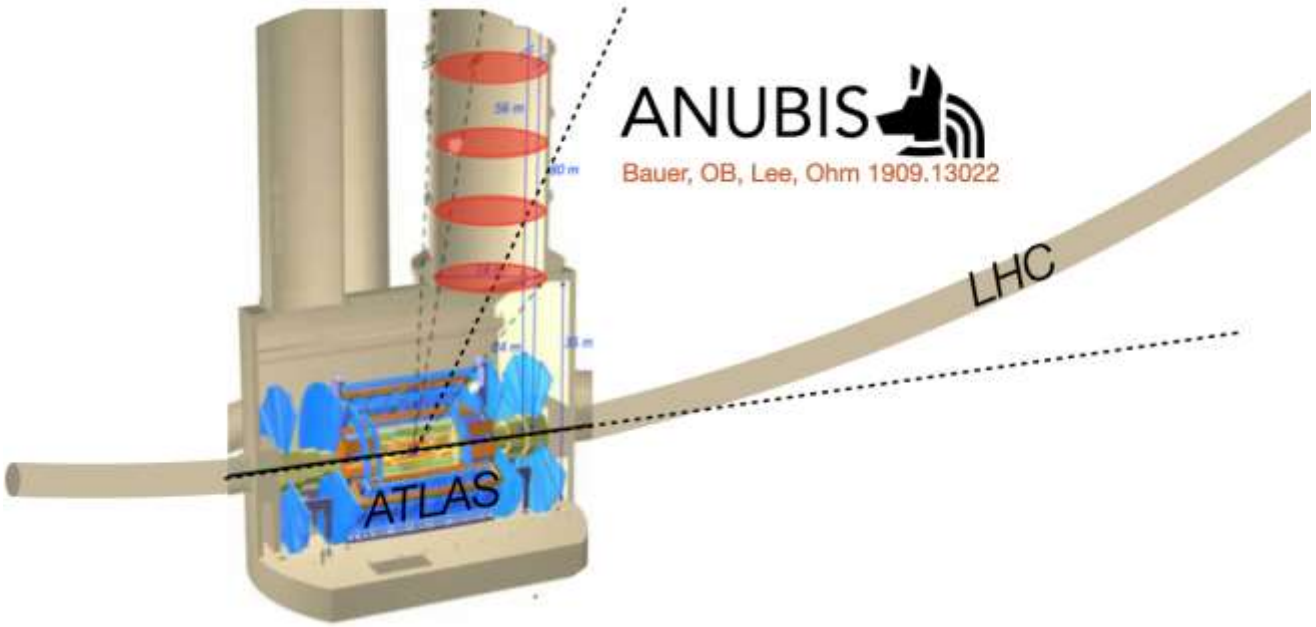
Practical side

- Improve front-end features and user-friendliness
- Online accessibility of the GUI with server backend
- UFO compliant input

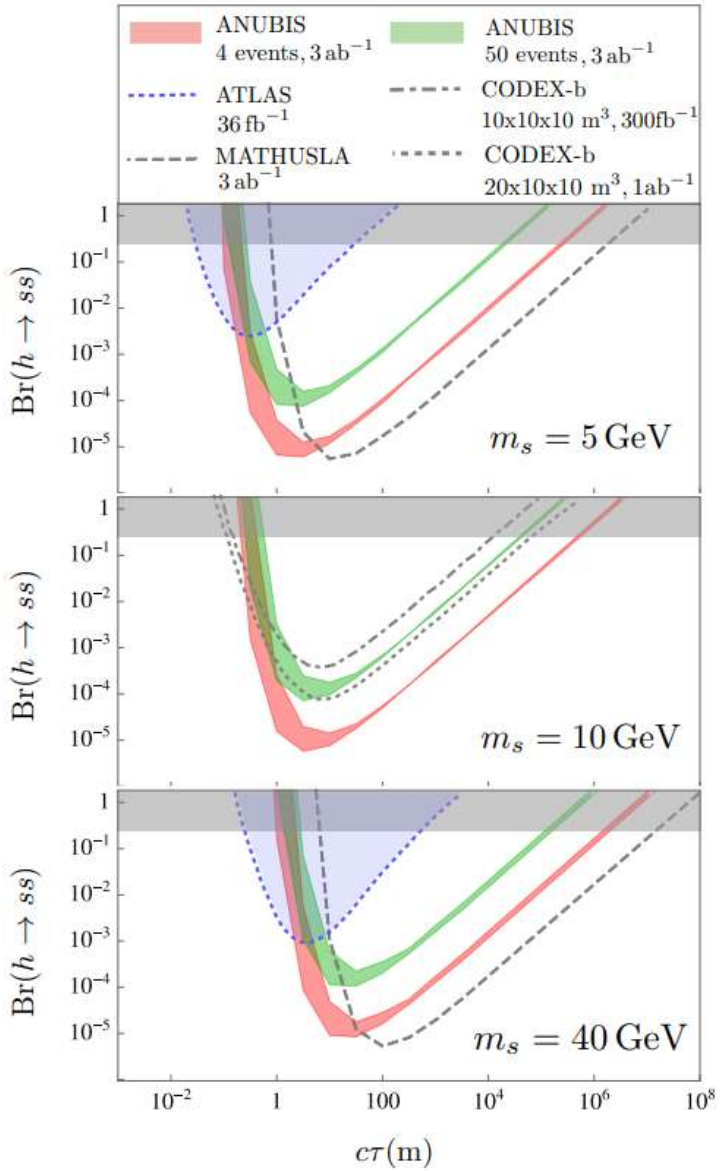
Even further

- Web coding on GUI
- Docker and kubernetes implementation, GPU optimization
- New version, HyperHyperiso ? UltraIso ? MegaIso ? GigaIso ? TurboIso 2000 ? Who knows

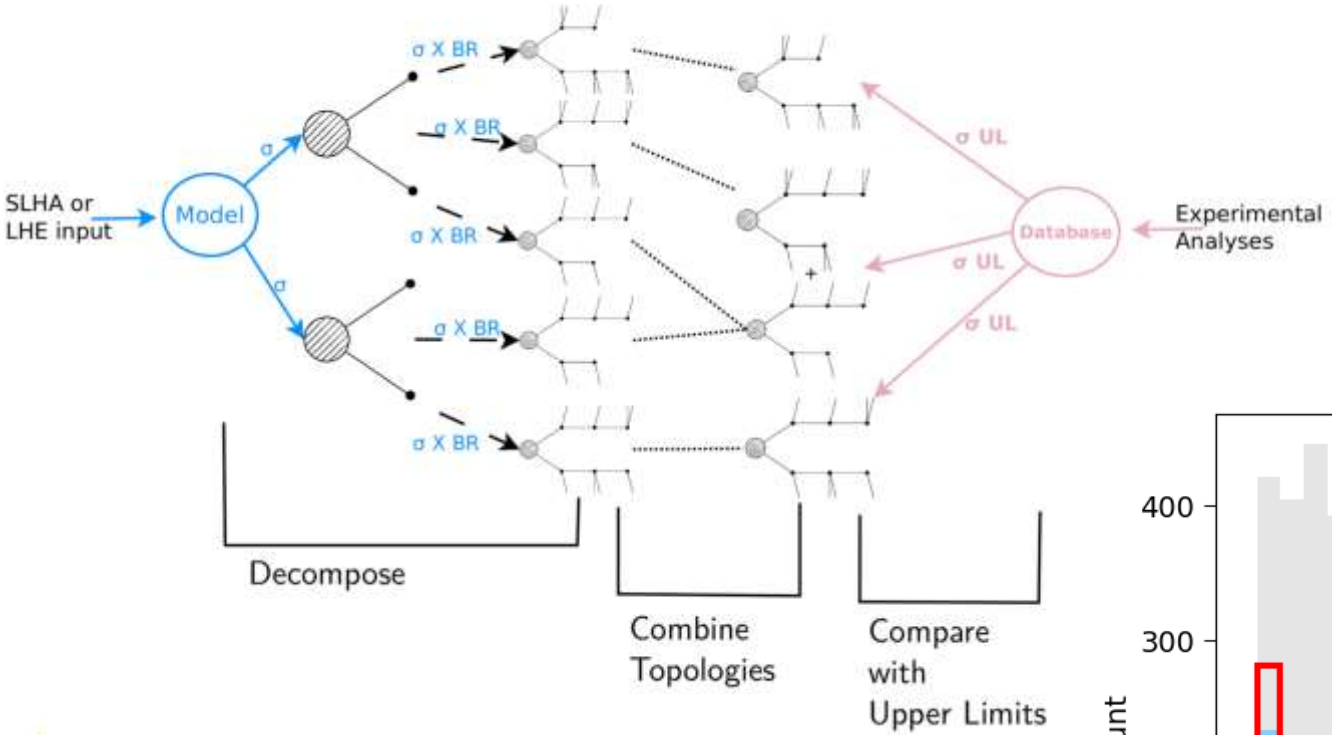
Others projects (Cambridge)

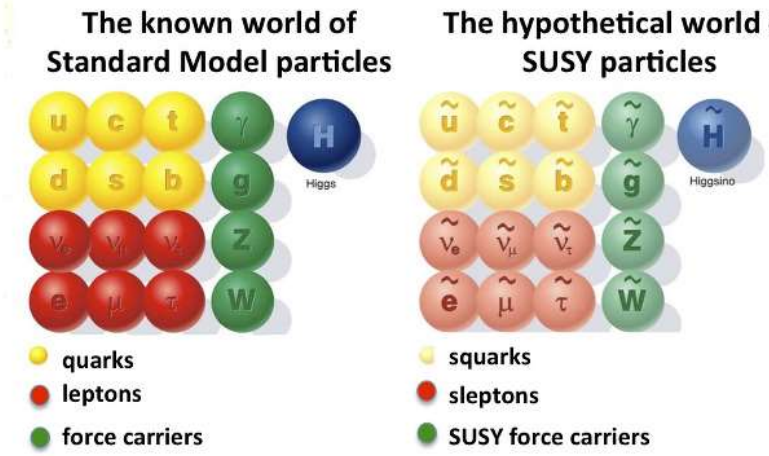


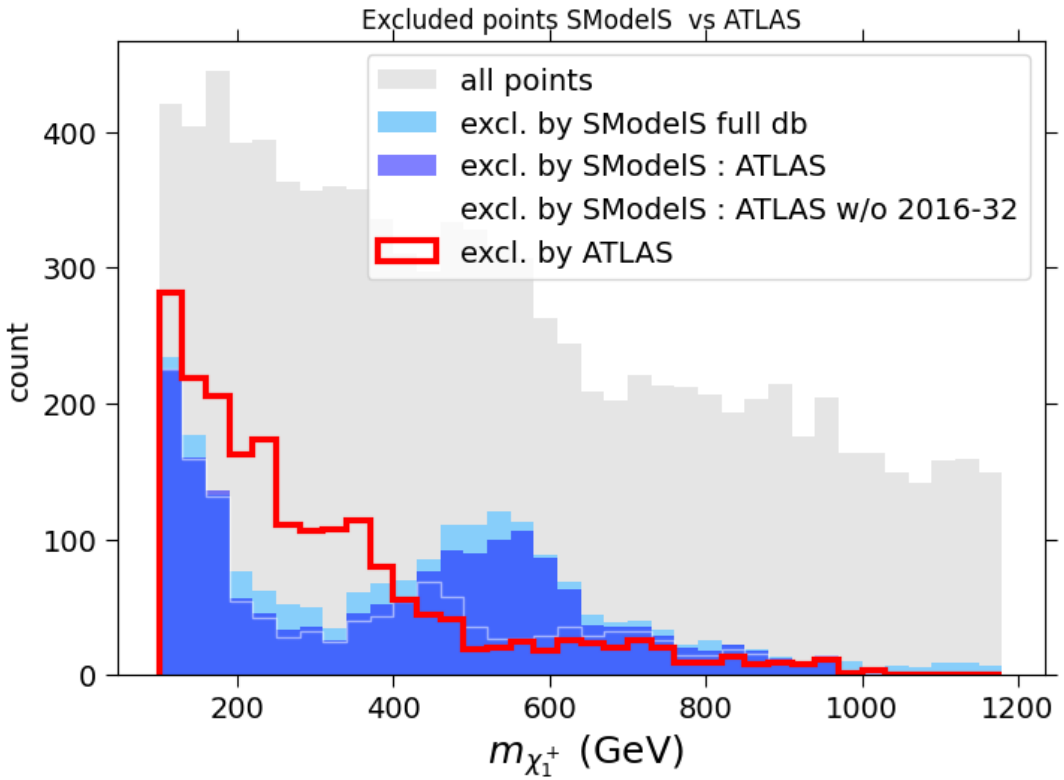
UNIVERSITY OF CAMBRIDGE



Others projects (Grenoble, LPSC)







Thanks