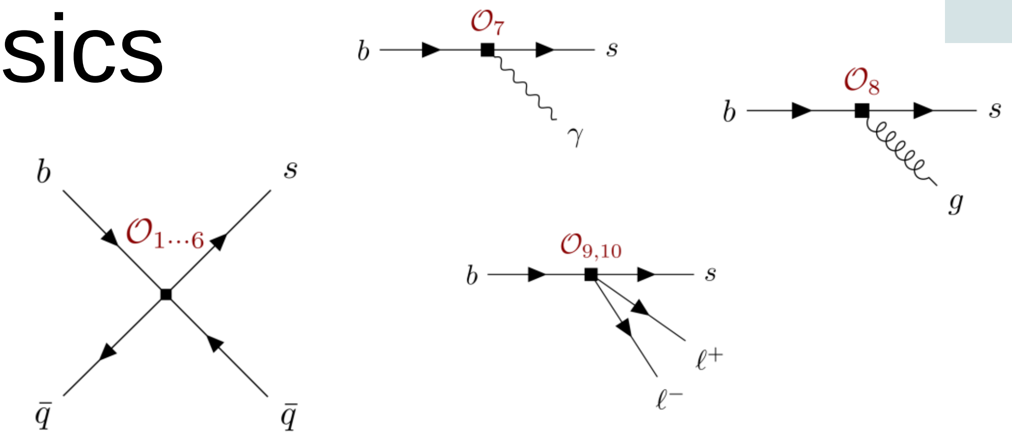
A general BSM calculator for flavour observables

Niels Fardeau, Théo Reymermier

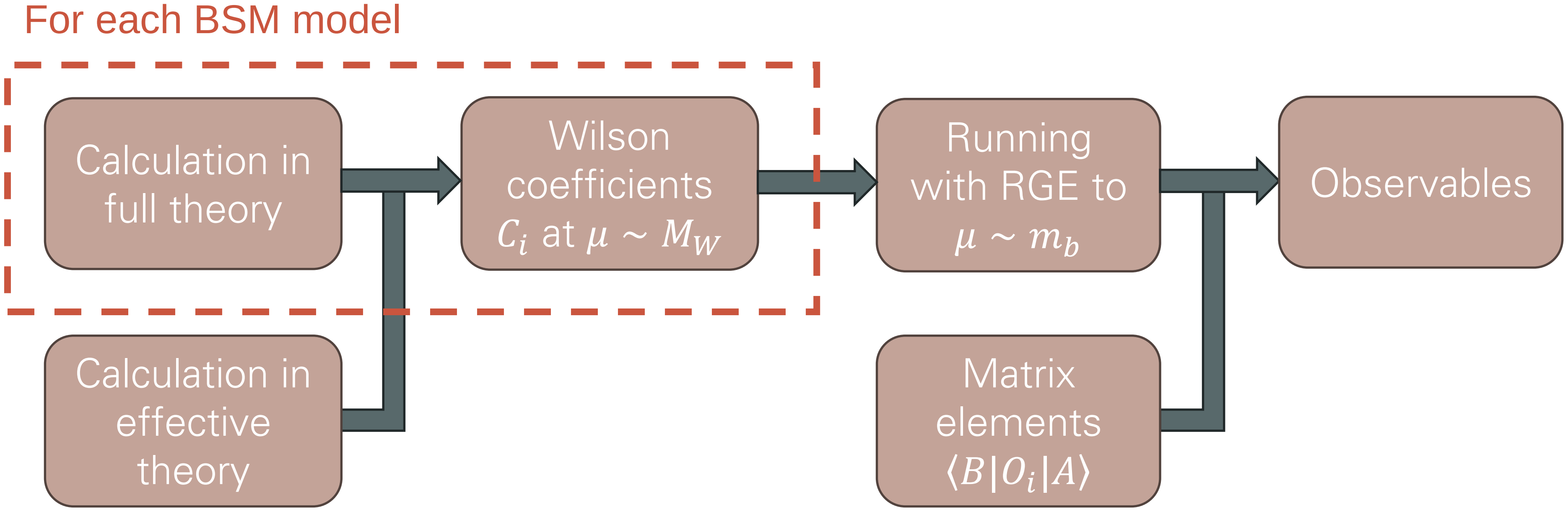
February 5 2025

RPP 2025

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

Calculation in full theory

Calculation in effective theory

Wilson coefficients C_i at $\mu \sim M_W$

Running with RGE to $\mu \sim m_b$

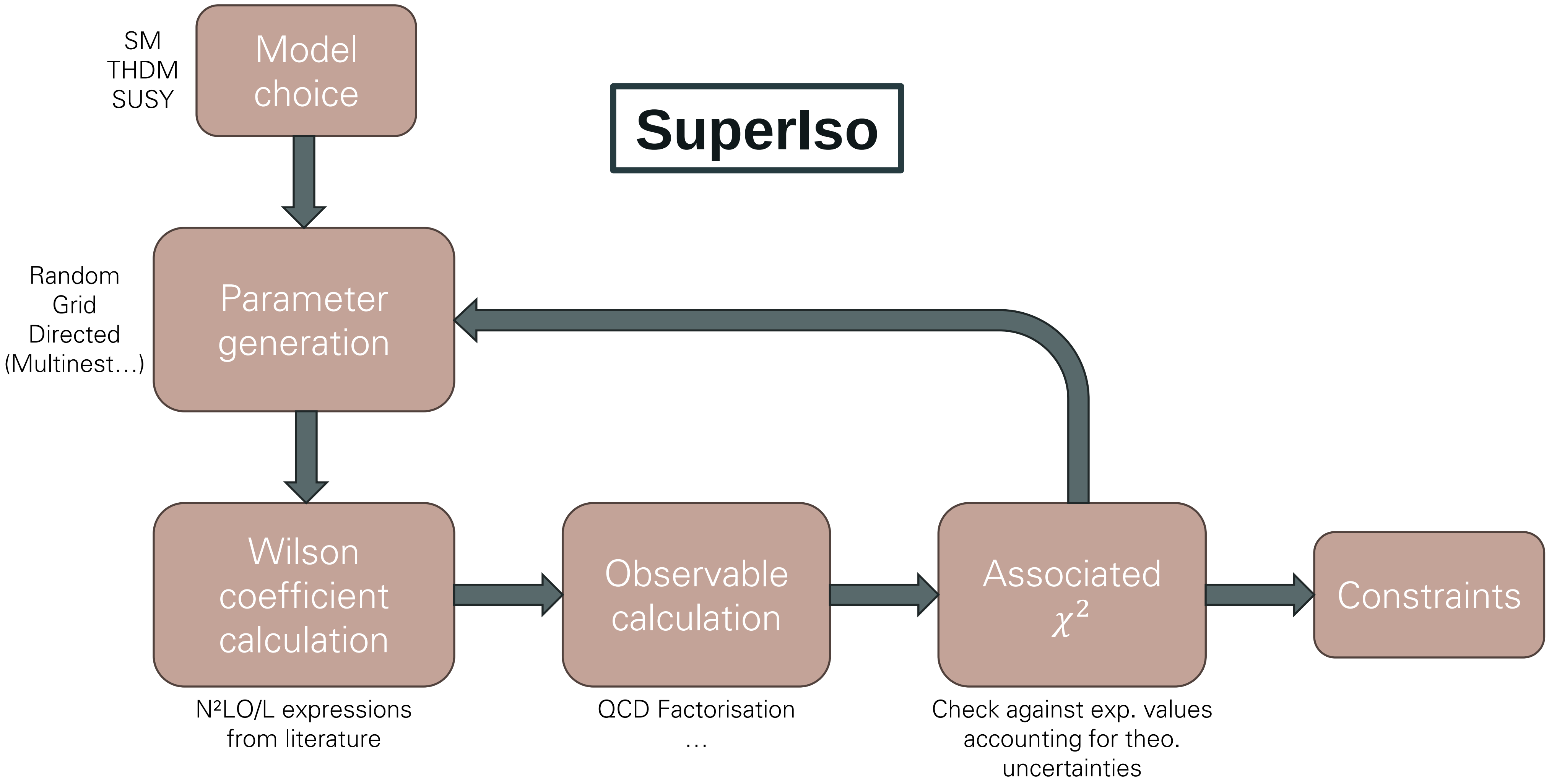
Matrix elements $\langle B|O_i|A\rangle$

Observables

Need for automated calculations !

SuperIso

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



SuperIso

SM
THDM
SUSY

Model choice

Random
Grid
Directed
(Multinest…)

Parameter generation

Wilson coefficient calculation

N²LO/L expressions from literature

Observable calculation

QCD Factorisation
…

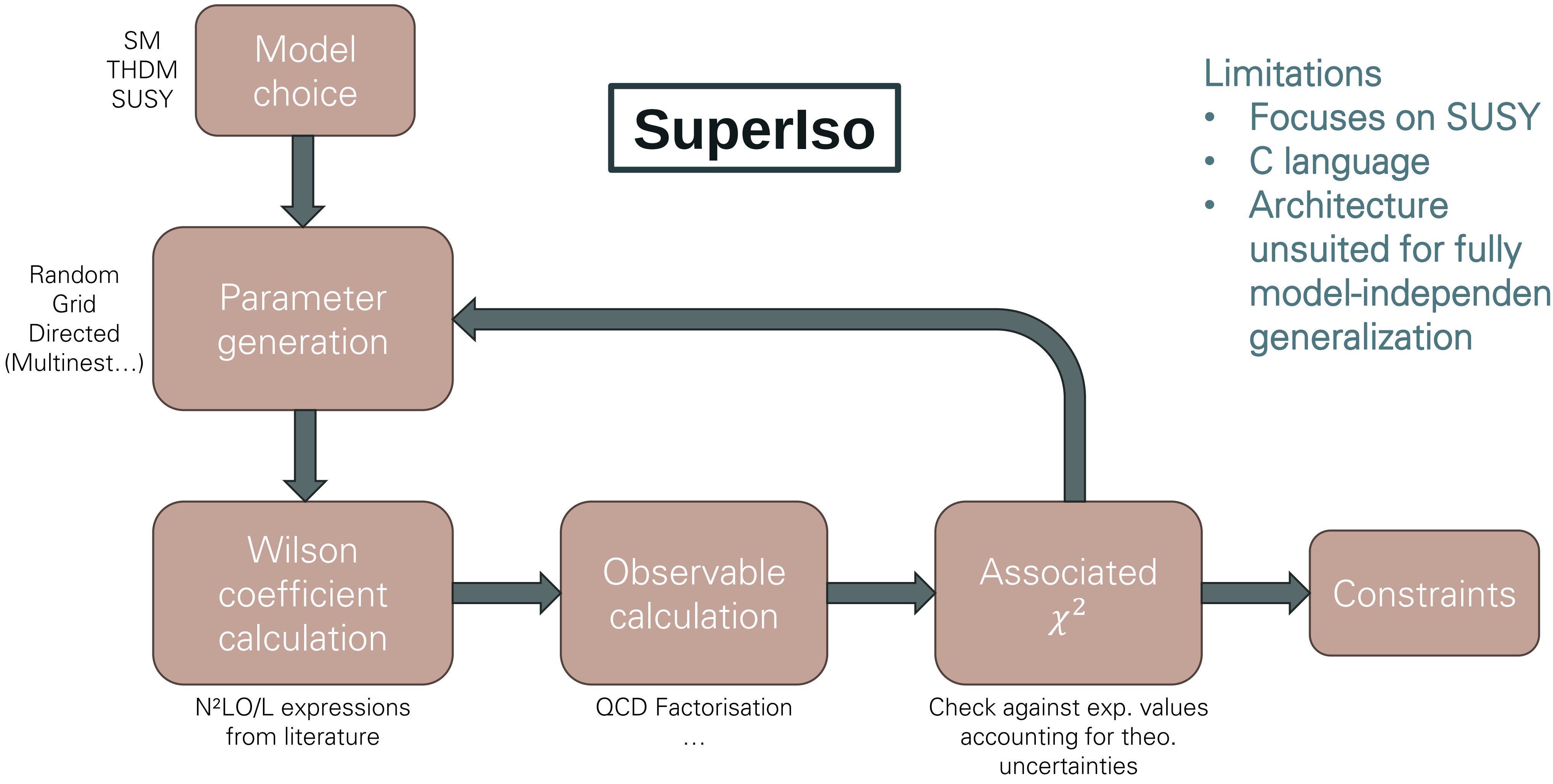
Associated χ^2

Check against exp. values accounting for theo. uncertainties

Constraints

SuperIso

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

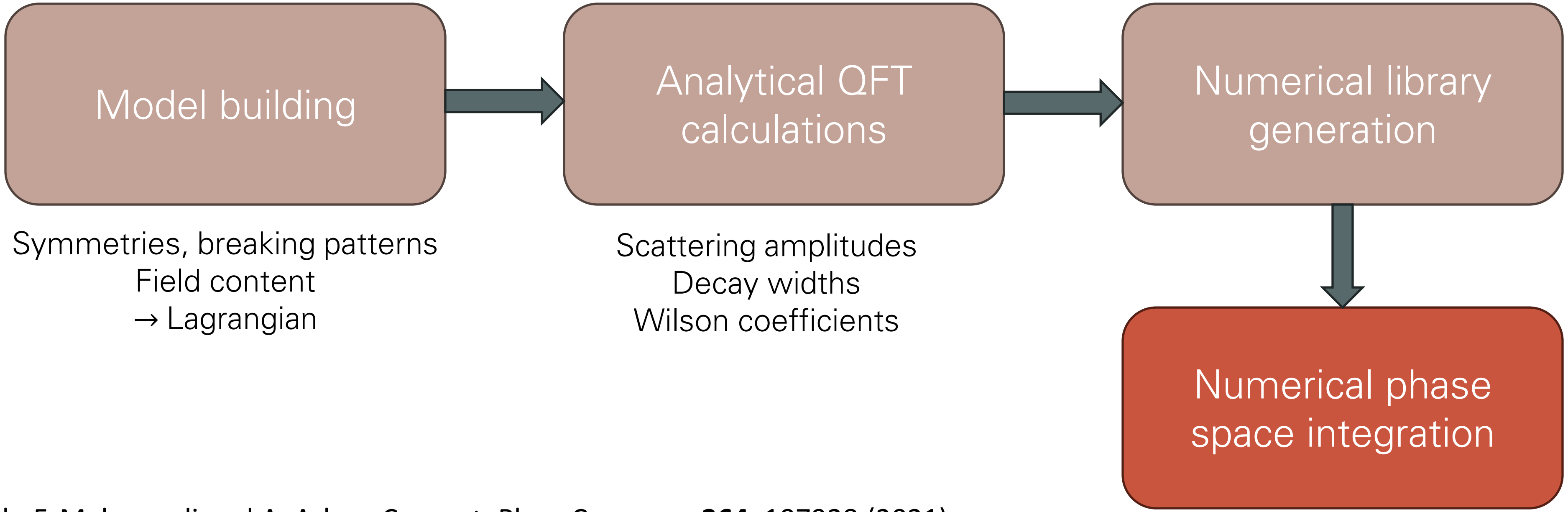


Limitations

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

MARTY (1.5)

MARTY



Model building

Symmetries, breaking patterns
Field content
→ Lagrangian

Analytical QFT calculations

Scattering amplitudes
Decay widths
Wilson coefficients

Numerical library generation

Numerical phase space integration

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

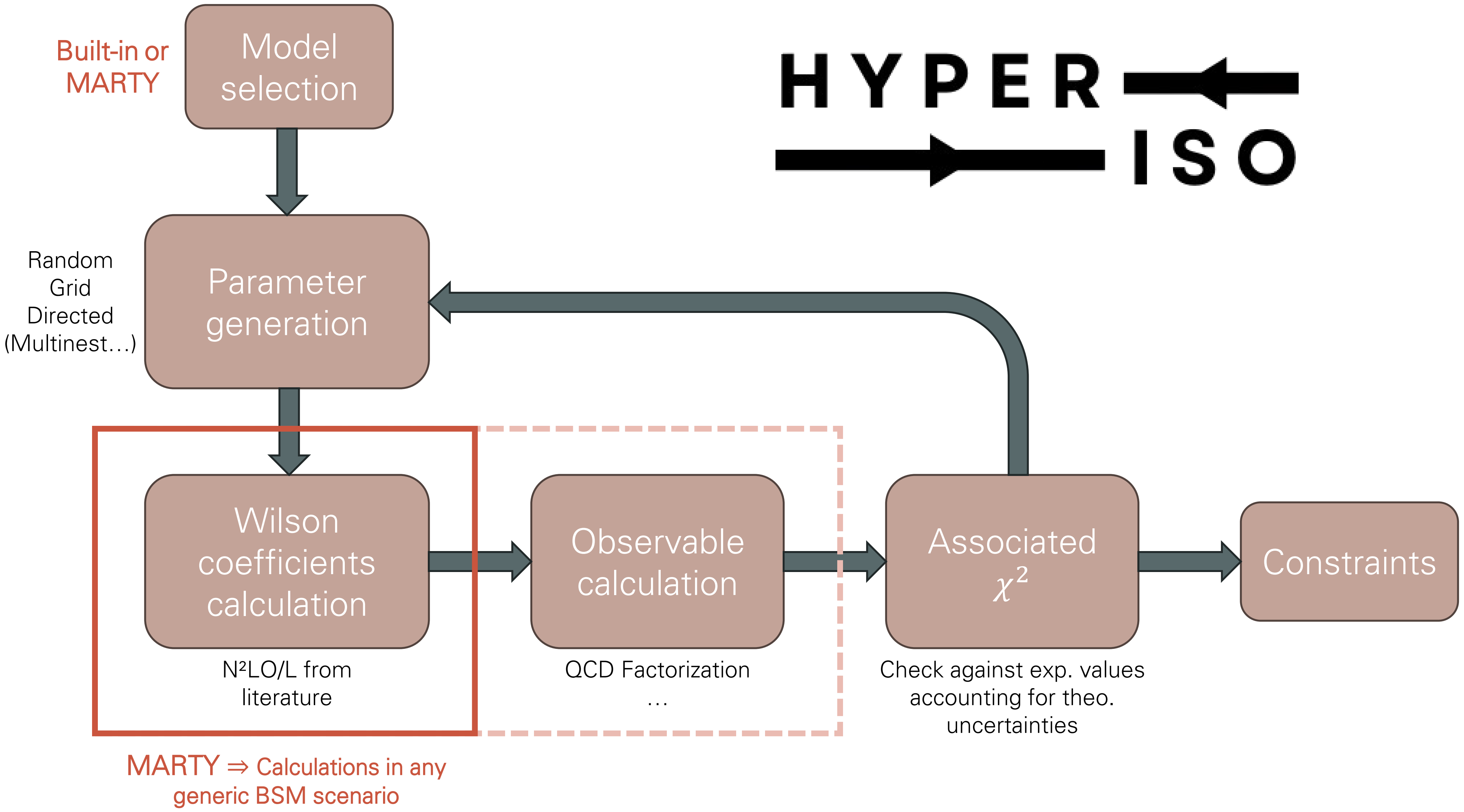
HyperIso

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



Built-in or MARTY

Model selection

Random
Grid
Directed
(Multinest…)

Parameter generation

Wilson coefficients calculation

N^2LO/L from literature

Observable calculation

QCD Factorization
…

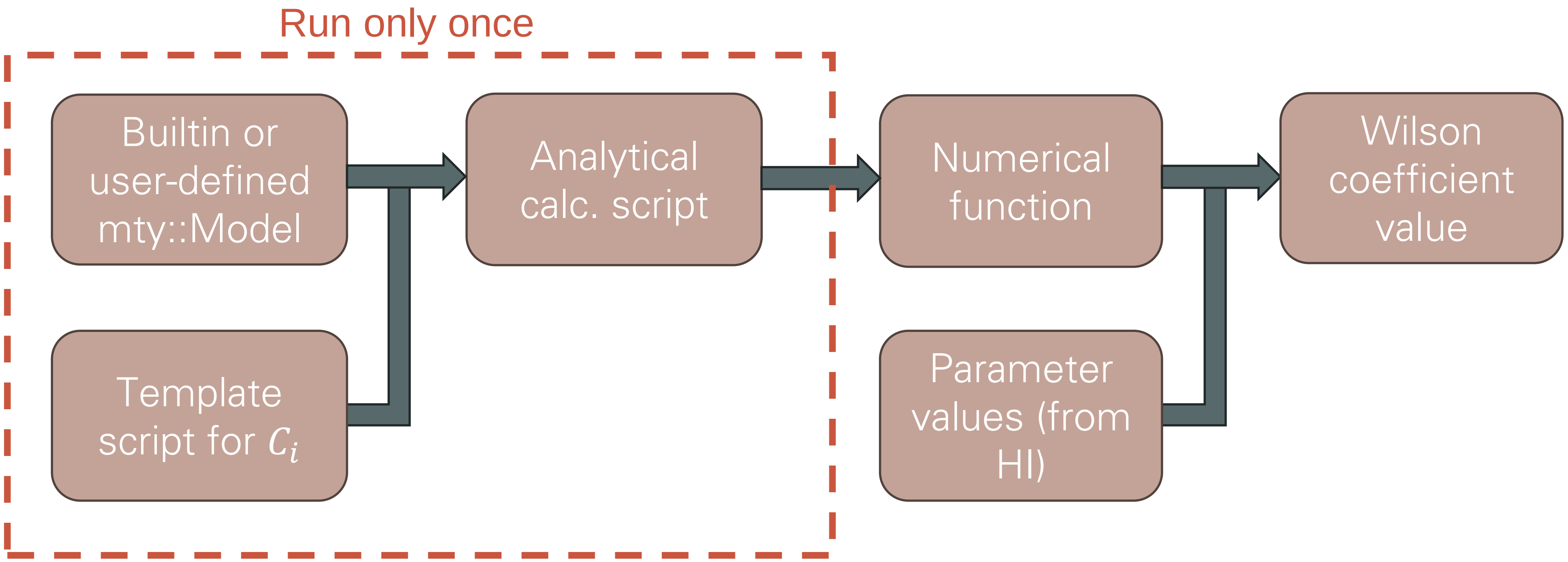
Associated χ^2

Check against exp. values accounting for theo. uncertainties

Constraints

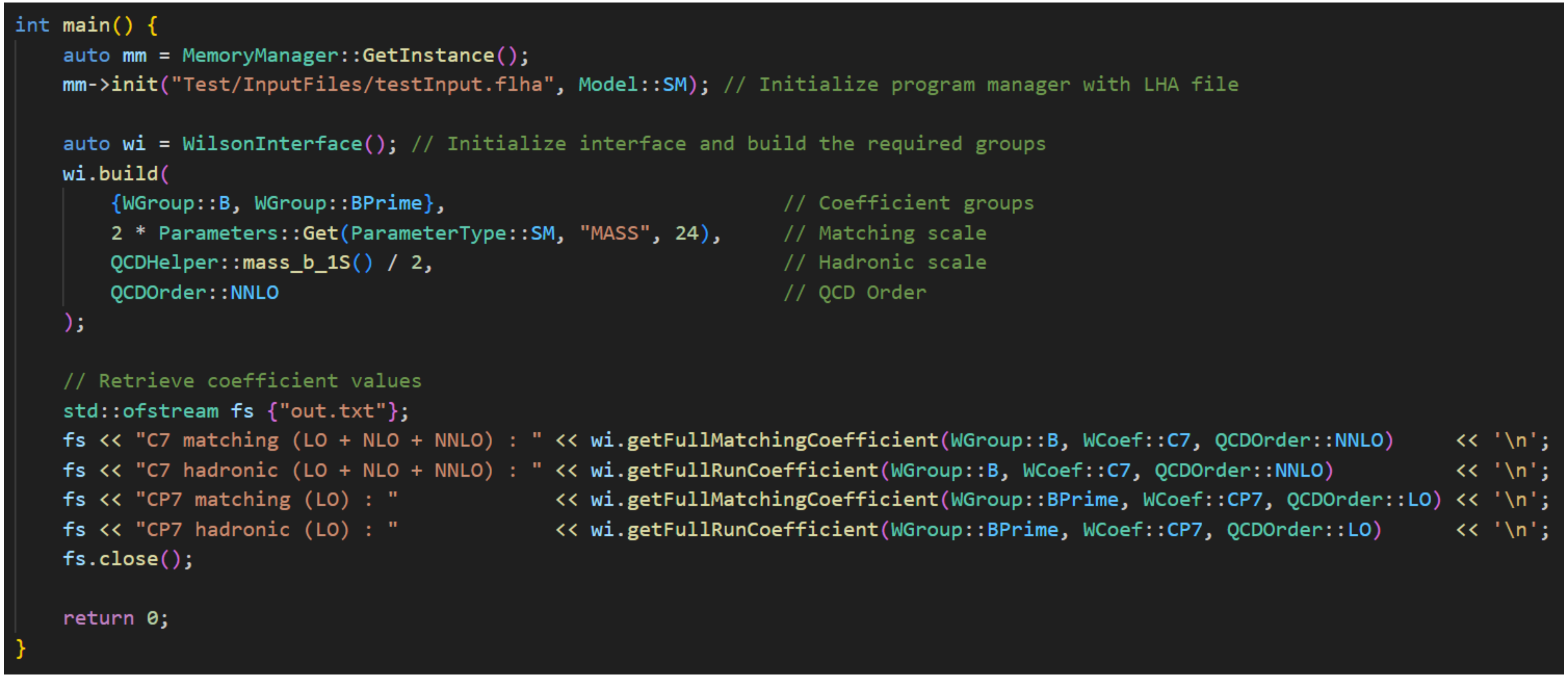
MARTY ⇒ Calculations in any generic BSM scenario

MARTY Interfacing


Run only once
Builtin or user-defined mty::Model
Template script for C_i
Analytical calc. script
Numerical function
Parameter values (from HI)
Wilson coefficient value

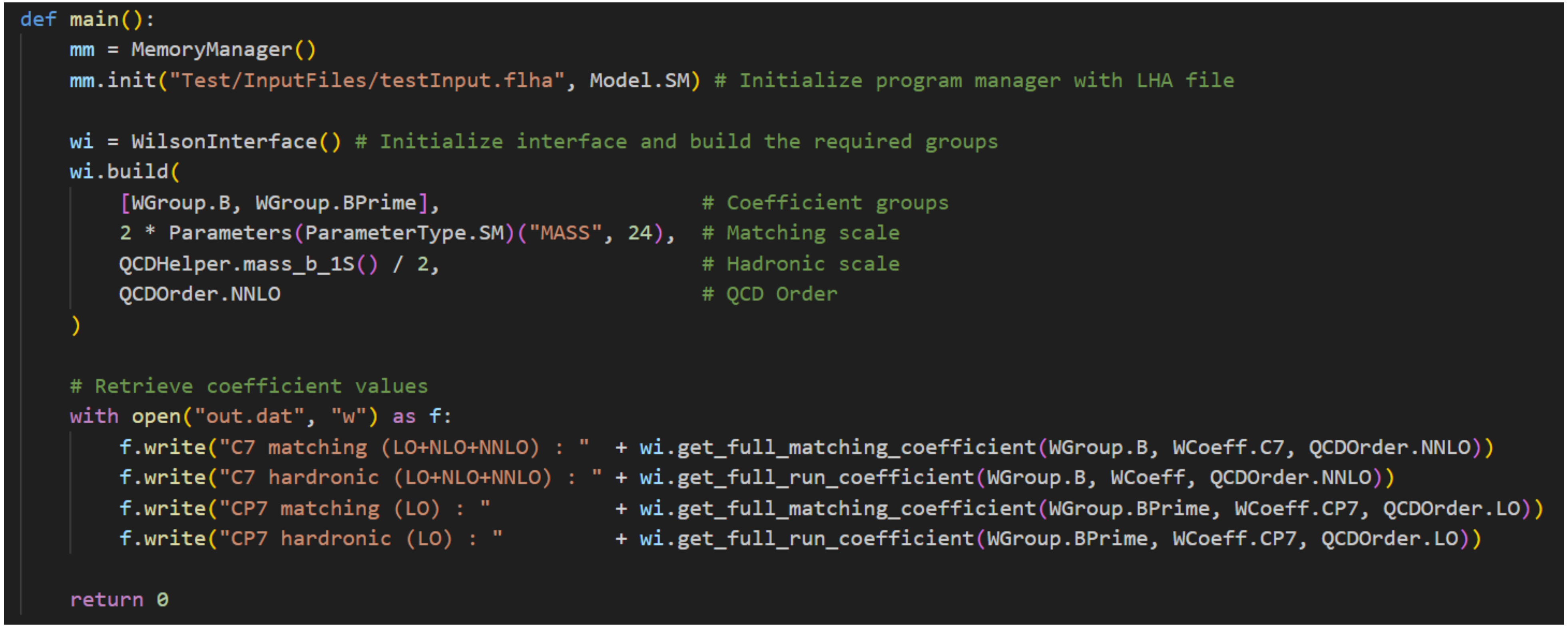
C++ Interface

Supports multithreading !



```
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;
}
```

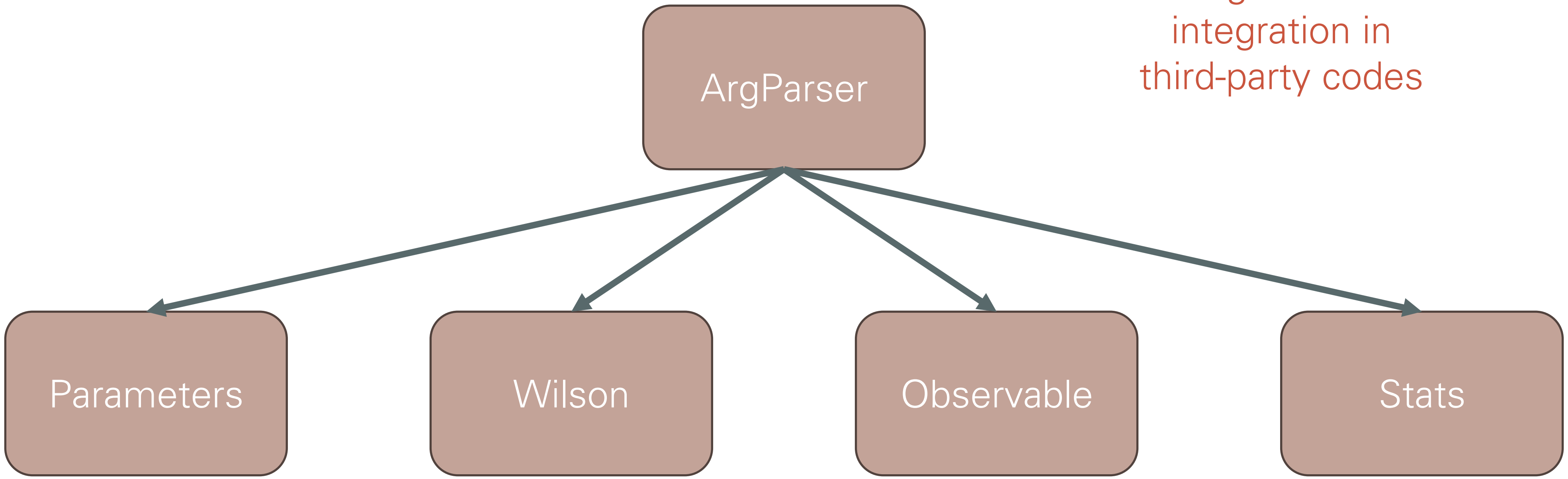
Python Interface

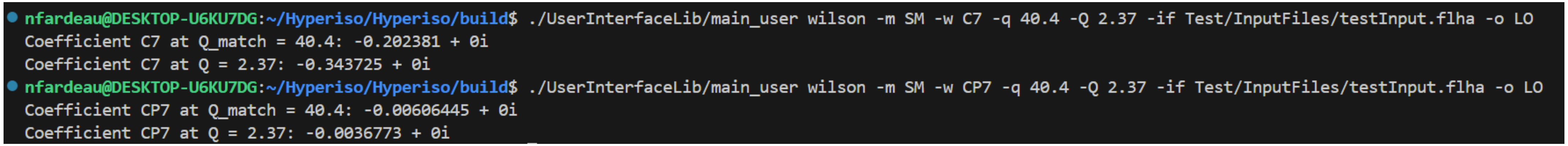


```python
def main():
    mm = MemoryManager()
    mm.init("Test/InputFiles/testInput.flha", Model.SM) # Initialize program manager with LHA file

    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

Time for a live demo !

Future improvements

Statistical side

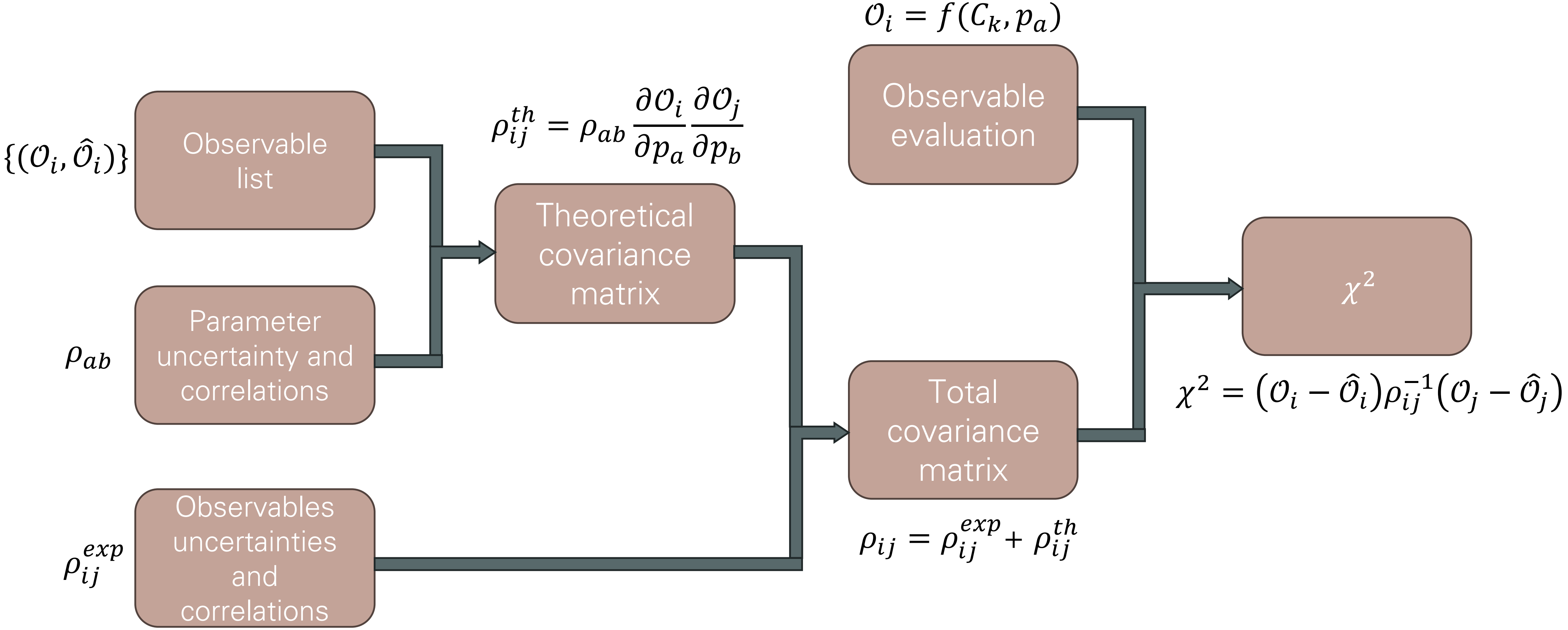
- Extend statistical calculations to generic nuisance distributions / likelihoods
- Builtin interface with scanning softwares (e.g. BSMArt)

Practical side

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

Thanks

Uncertainty estimation and model validation



$\{(\mathcal{O}_i, \hat{\mathcal{O}}_i)\}$ Observable list

ρ_{ab} Parameter uncertainty and correlations

ρ_{ij}^{exp} Observables uncertainties and correlations

$$\rho_{ij}^{th} = \rho_{ab}\frac{\partial \mathcal{O}_i}{\partial p_a}\frac{\partial \mathcal{O}_j}{\partial p_b}$$

Theoretical covariance matrix

$$\mathcal{O}_i = f(C_k, p_a)$$

Observable evaluation

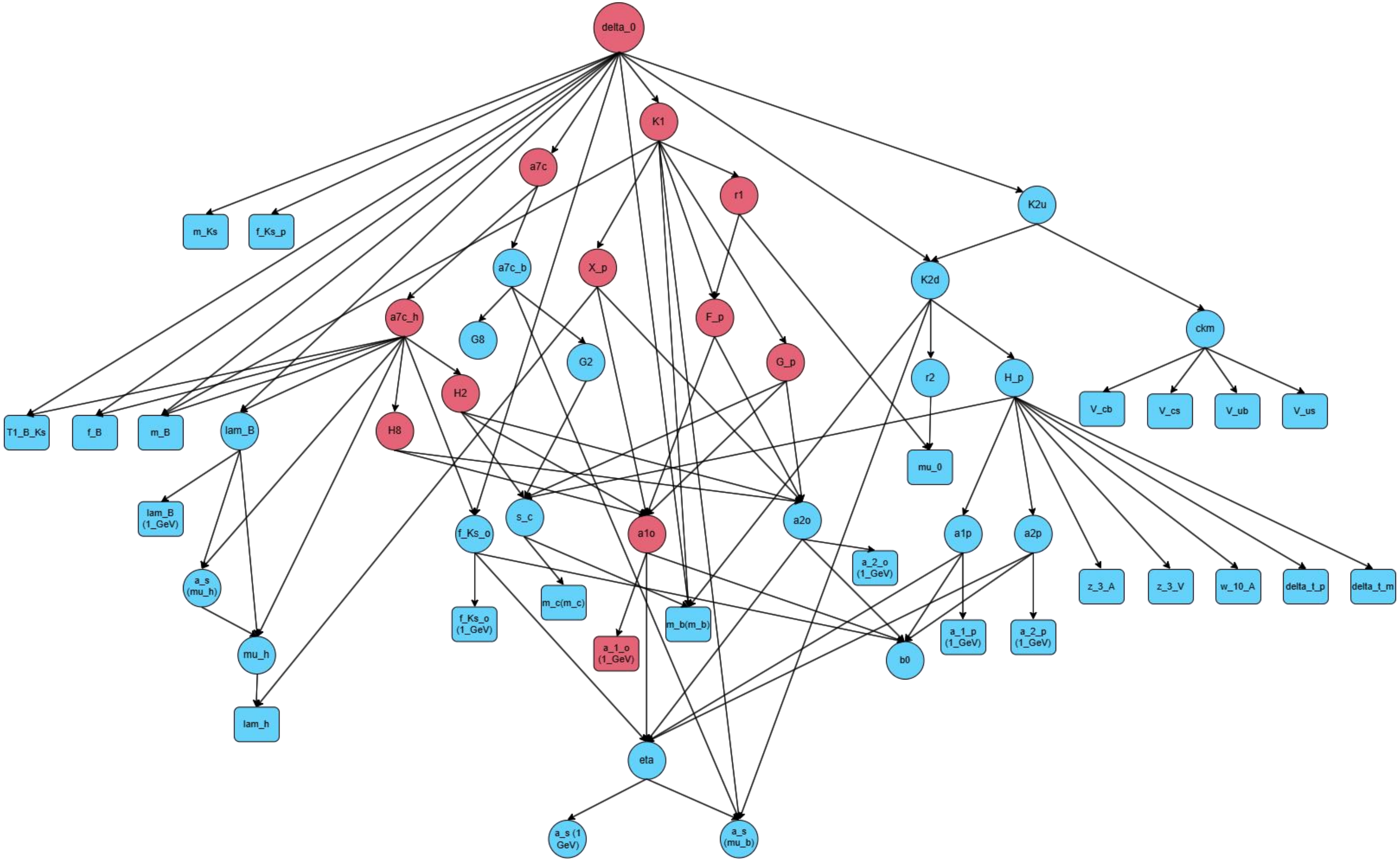
Total covariance matrix

$$\rho_{ij} = \rho_{ij}^{exp} + \rho_{ij}^{th}$$

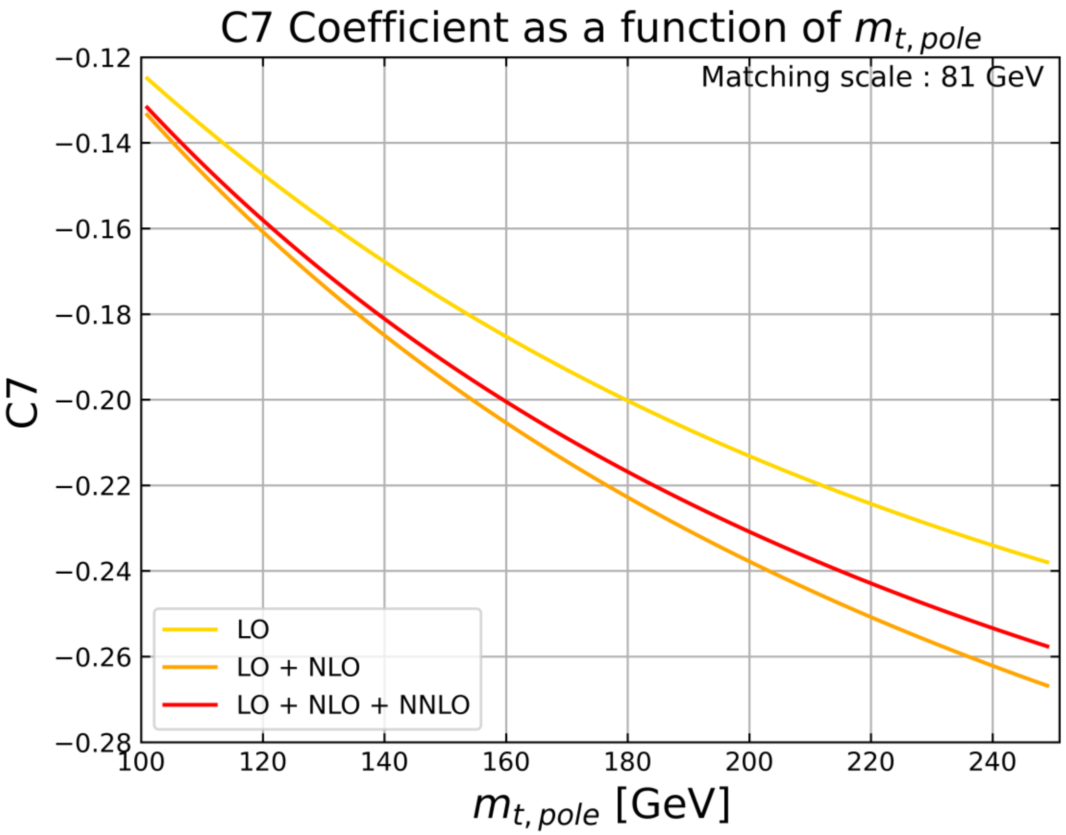
χ^2

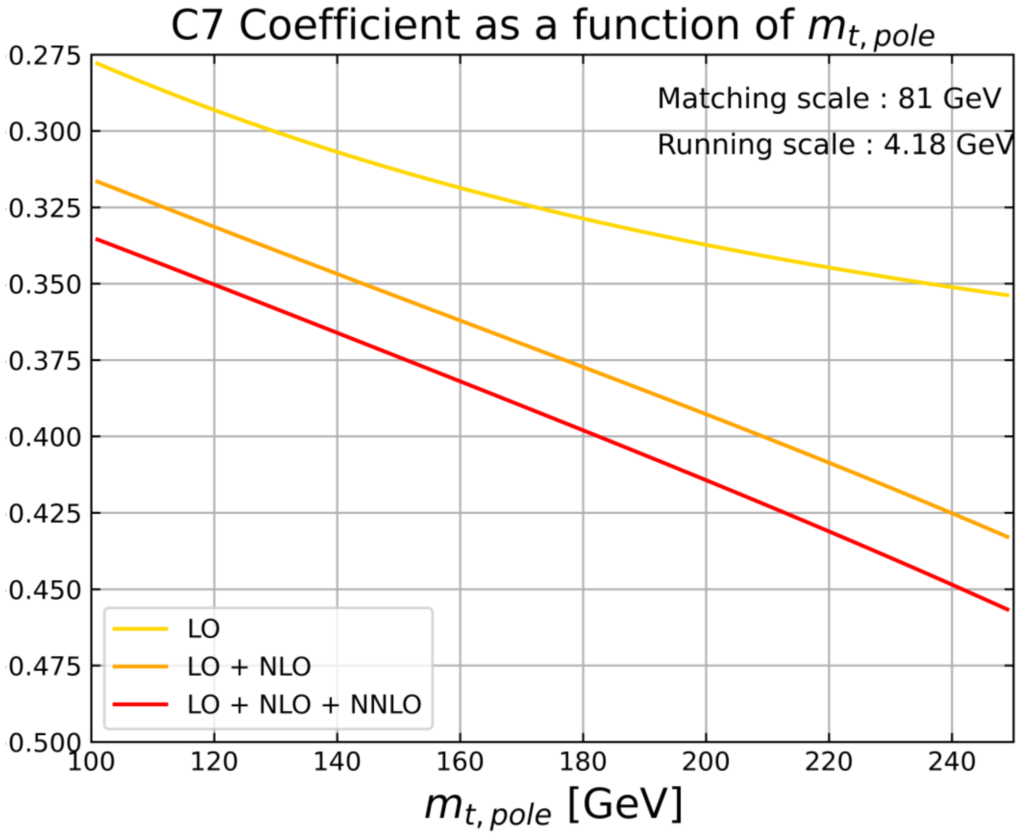
$$\chi^2 = (\mathcal{O}_i - \hat{\mathcal{O}}_i)\rho_{ij}^{-1}(\mathcal{O}_j - \hat{\mathcal{O}}_j)$$

Graph-based observable evaluation



Example plots


C7 Coefficient as a function of $m_{t,pole}$
Matching scale : 81 GeV
C7
$m_{t,pole}$ [GeV]
LO
LO + NLO
LO + NLO + NNLO


C7 Coefficient as a function of $m_{t,pole}$
Matching scale : 81 GeV
Running scale : 4.18 GeV
$m_{t,pole}$ [GeV]
LO
LO + NLO
LO + NLO + NNLO