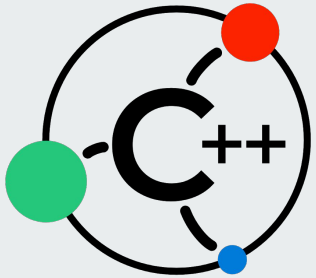


MARTY

Modern **AR**tificial **T**heoretical ph**Y**sicist

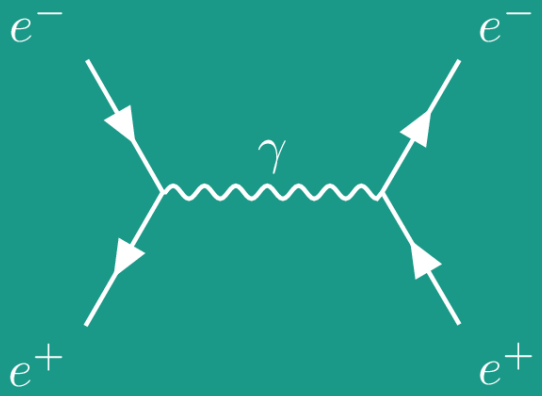
A C++ framework for easy Beyond the Standard Model (BSM) phenomenology.

https://marty.in2p3.fr

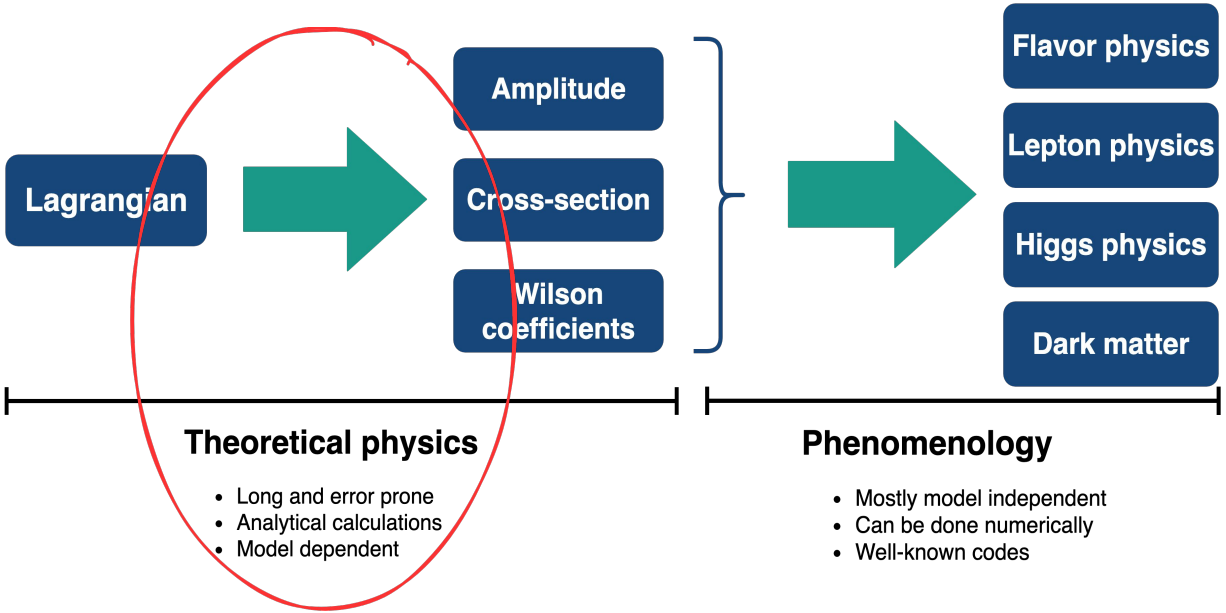
arxiv: 2011.02478 [hep-ph]

Grégoire Uhlrich (PhD student)
Nazila Mahmoudi (Advisor)
Alexandre Arbey

Introduction

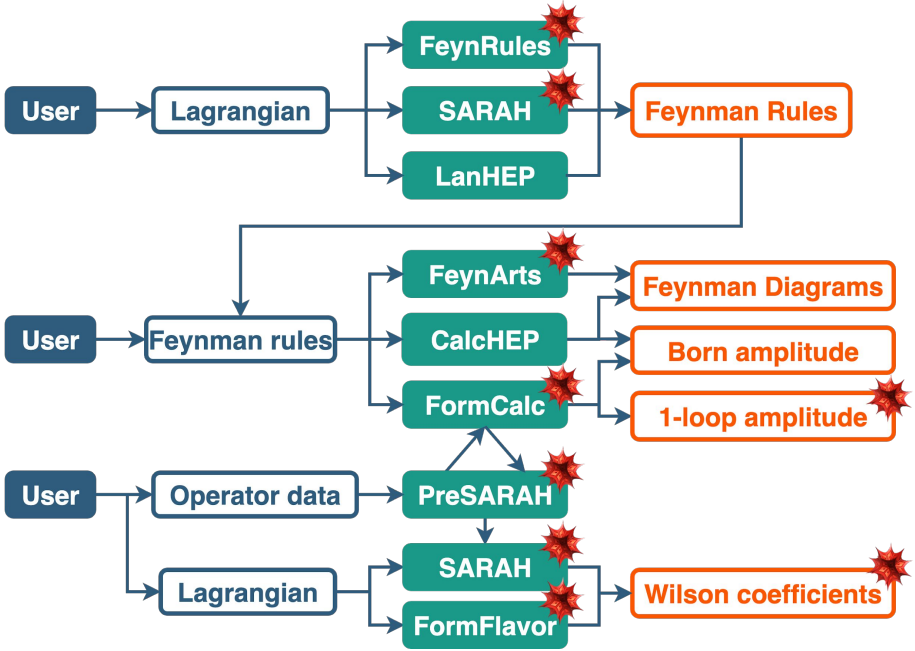


Phenomenology for BSM models

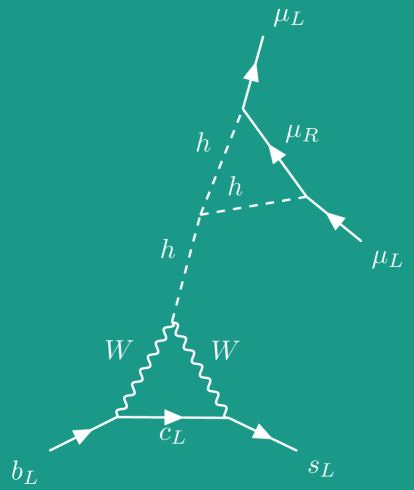


Present Ecosystem

- Many distinct codes
- **Several user inputs necessary**
- Depending on **Mathematica**
- Hard to automate calculation of Wilson coefficients at NLO



The challenge MARTY



Philosophy

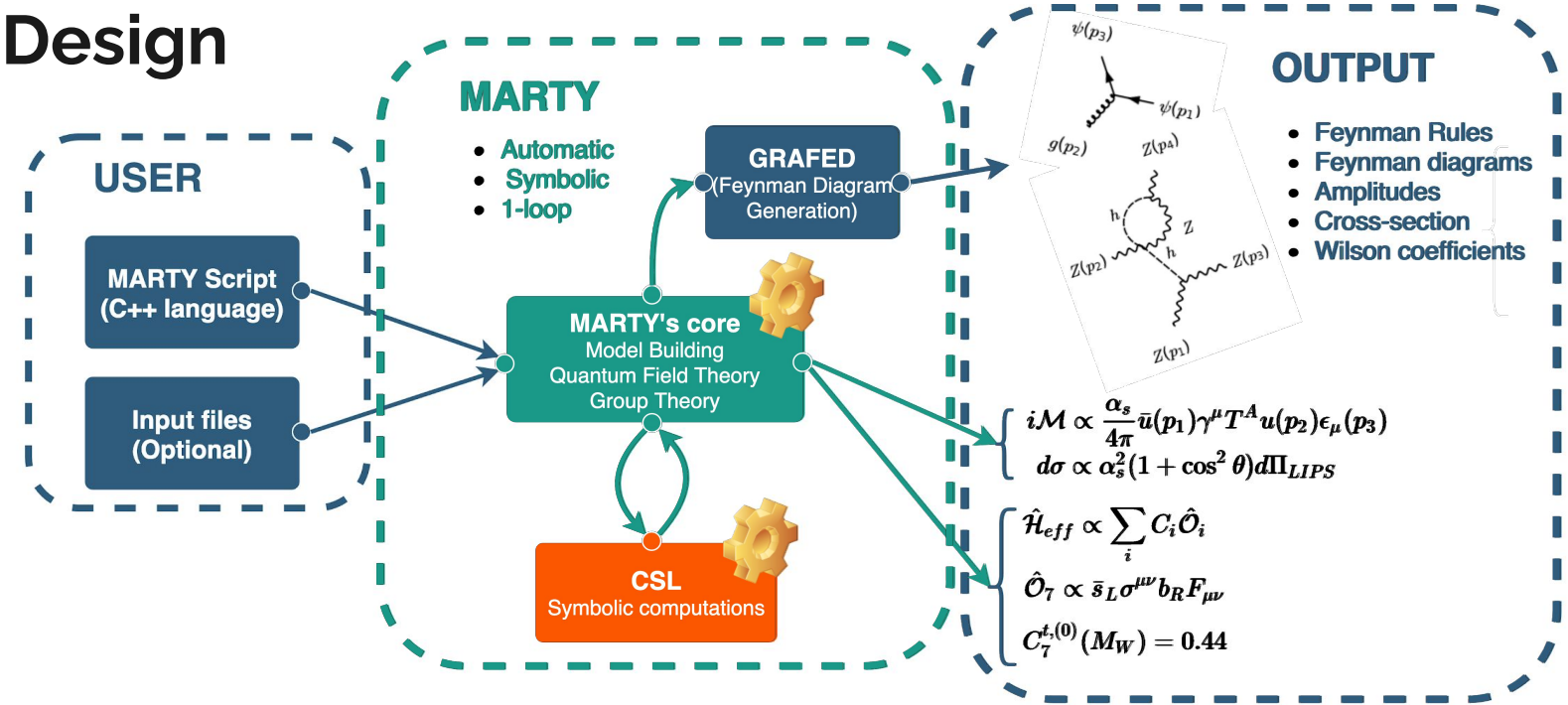
Physicist point of view

- Generality
  - BSM model variety
  - Calculation capabilities
- 1-loop calculations including NLO.
- Simple User interface
- No need to have deep C++ knowledge

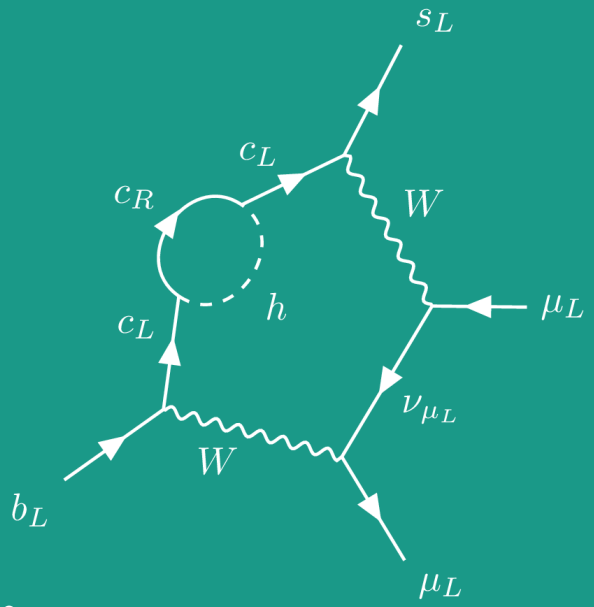
Developer point of view

- Modularity (all modules belong to MARTY)
  - **MARTY** (physics part)
  - **CSL** (C++ Symbolic computation Library)
  - **GRAFED** (Generating and Rendering Application for FEynman Diagrams)
- **Independence: no Mathematica !**
- Readability - modern C++ (2017 standard)

Design


USER
MARTY Script
(C++ language)
Input files
(Optional)
MARTY
Automatic
Symbolic
1-loop
GRAFED
(Feynman Diagram
Generation)
MARTY's core
Model Building
Quantum Field Theory
Group Theory
CSL
Symbolic computations
OUTPUT
Feynman Rules
Feynman diagrams
Amplitudes
Cross-section
Wilson coefficients
$i\mathcal{M} \propto \frac{\alpha_s}{4\pi}\bar{u}(p_1)\gamma^\mu T^A u(p_2)\epsilon_\mu(p_3)$
$d\sigma \propto \alpha_s^2(1+\cos^2\theta)d\Pi_{LIPS}$
$\hat{\mathcal{H}}_{eff} \propto \sum_i C_i\hat{\mathcal{O}}_i$
$\hat{\mathcal{O}}_7 \propto \bar{s}_L\sigma^{\mu\nu}b_R F_{\mu\nu}$
$C_7^{t,(0)}(M_W) = 0.44$

What MARTY-1.0 can do ?



MARTY's current BSM capabilities

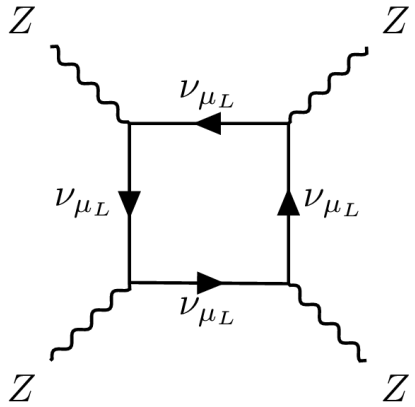
- **Quantum Field Theory**
  - 4-dimensional Minkowski space-time
  - Spin 0, ½, 1
  - Model building utilities
- **Group theory**
  - Semi-simple groups (SU(N), SO(N), Sp(N), E_6, E_7, E_8, F_4, G_2)
  - Representation theory
  - Algebra generators
  - Simplifications, traces

$$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu} + \left(D^{\mu} H\right)^{\dagger} D_{\mu} H + \mu H^{\dagger} H - \lambda \left(H^{\dagger} H\right)^{2},$$

$$H \to \begin{pmatrix} 0 \\ \\ \frac{h+v}{\sqrt{2}} \end{pmatrix}$$

MARTY's calculation capabilities

- Fully **symbolic and automatic (CSL)**
- Up to 5 external particles, at **1-loop**
- **Amplitudes, partonic cross-sections, Wilson coefficients**
- **Simplifications**
  - Dirac algebra
  - Group algebra
  - Tensor reduction for momentum integrals
  - Dimensional regularization
  - **LoopTools [hep-ph/9807565]** numerical integrals
  - Equations of motion (Dirac equation)
  - Definition of abbreviations



Library generation

- Allows to get actual **numbers**
- **Fully automated**
- **Spectrum generation** automated
  - Mass matrix diagonalization
  - Mixing calculations
  - Bi-diagonalization support
- **Kinematics to be defined by the user**

```
Library SMWil("SMWil", "marty_libs");
SMWil.addFunction("C7", C7);
SMWil.build();
```

⇩ Automatic generation

```
std::complex<double> C7(
      const double M_W,
      const double m_b,
      const double m_s,
      const double m_t,
      const double s_12
      );
```

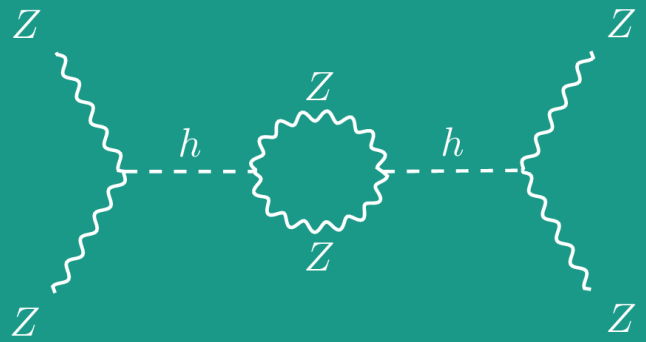
⇩ Simple to use

```
#include <SMWil>   C++

SMWil::C7(...);
```

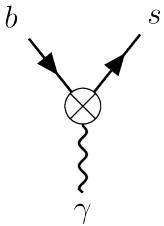
⇩ Upcoming ...

```
import SMWil  Python

SMWil.C7(...)
```

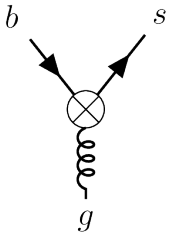
A 1-loop Wilson example



C^t_7 - C^t_8 at LO with MARTY

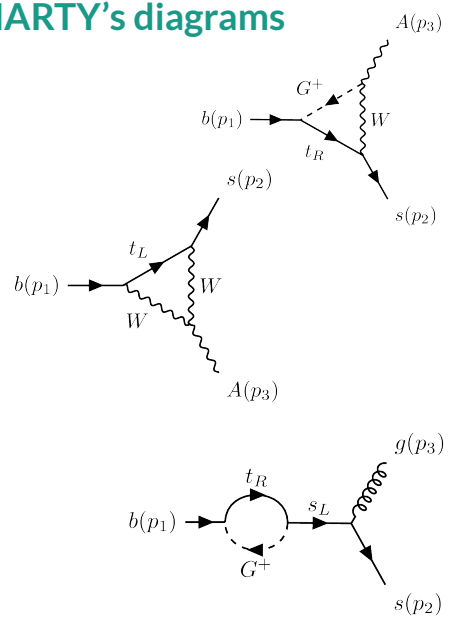


$$\propto C_7\langle\hat{O}_7\rangle + C'_7\langle\hat{O}'_7\rangle = C_7 \cdot \bar{s}_L\sigma^{\mu\nu}b_R F_{\mu\nu} + C'_7 \cdot \bar{s}_R\sigma^{\mu\nu}b_L F_{\mu\nu}.$$



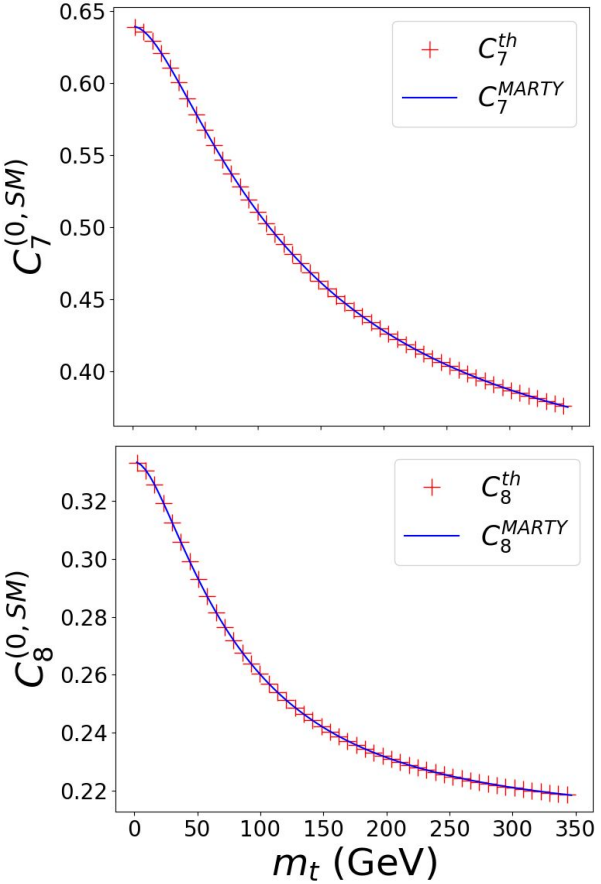
$$\propto C_8\langle\hat{O}_8\rangle + C'_8\langle\hat{O}'_8\rangle = C_8 \cdot \bar{s}_L\sigma^{\mu\nu}G_{\mu\nu}b_R + C'_8 \cdot \bar{s}_R\sigma^{\mu\nu}G_{\mu\nu}b_L.$$

MARTY's diagrams



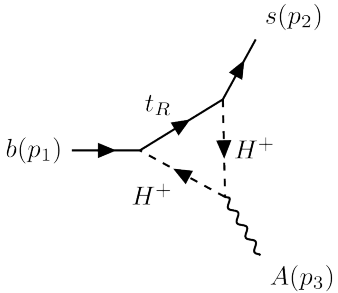
Results in the SM

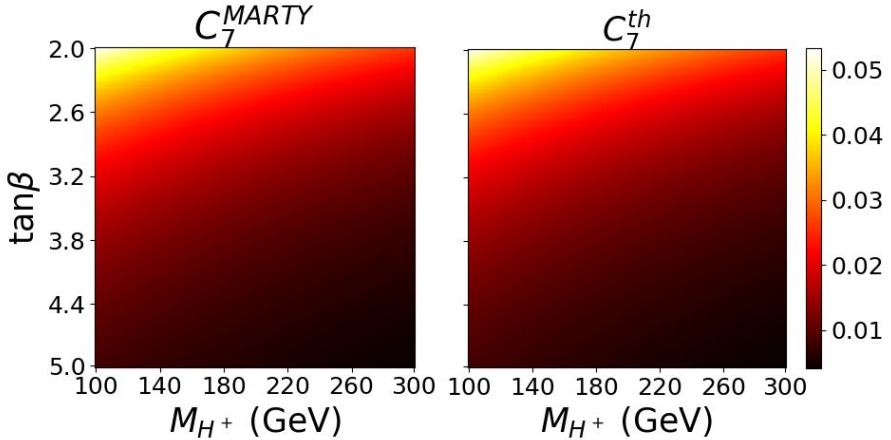
- **Multiple gauges**
  - Unitary (W contributions)
  - Feynman (W and Goldstone contributions)
- ~ 1 s in Feynman gauge
- Automatic C++ code generation
- Plot with python
- **Source:** SuperIso manual (arXiv: 0808.3144)



Results in the 2HDM (type IV - Lepton specific)

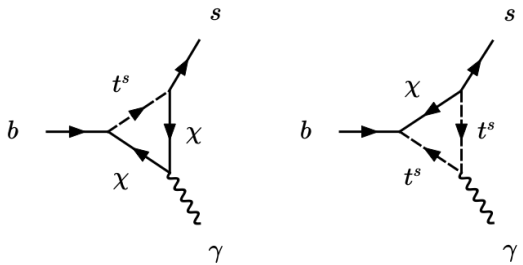
- **Charged Higgs** contributions
- Depend on
  - M_{H^+}
  - $\tan\beta$
- **Source:** SuperIso manual [arXiv: 0808.3144]

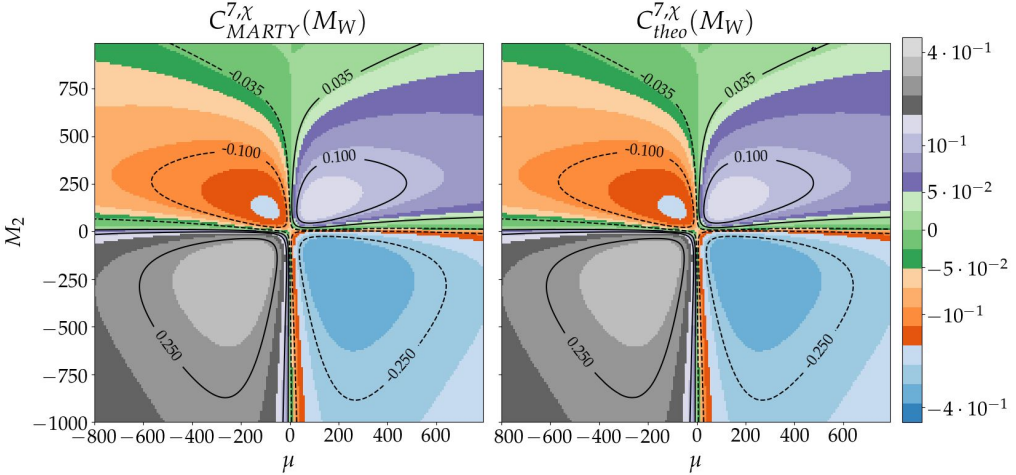




Results in the pMSSM

- Chargino stop contributions
- Spectrum generation with MARTY
- Depend on
  - μ : Higgs potential parameter
  - M_2 : Wino mass
- **Source:** hep-ph/9806308

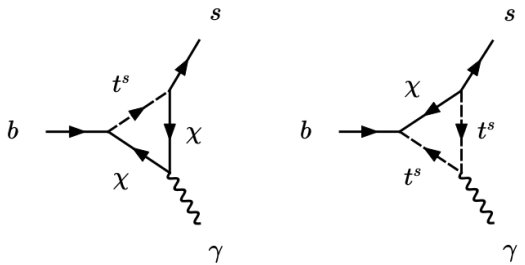


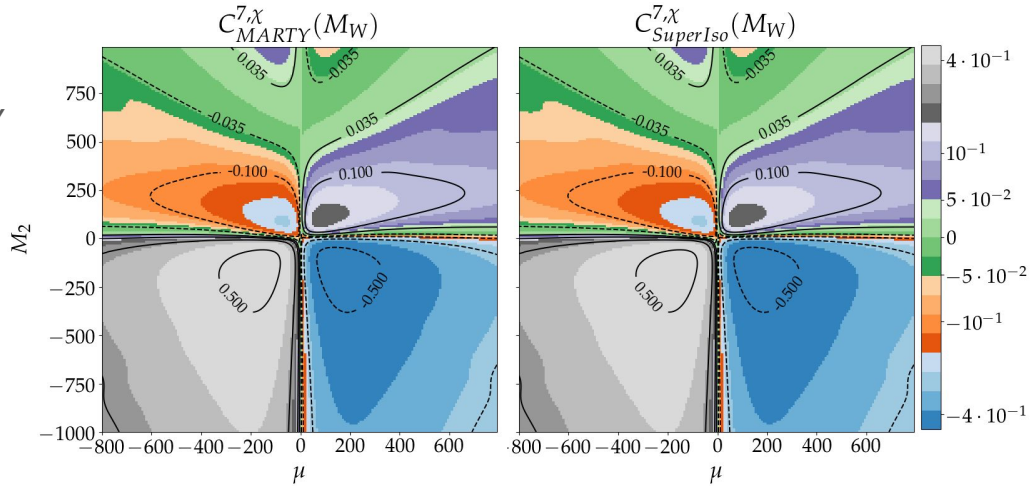


$$\tan\beta = 50$$

Results in the pMSSM

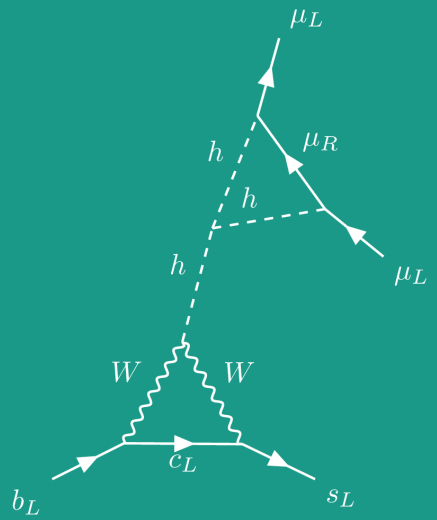
- Chargino stop contributions
- Spectrum generation with SOFTSUSY
- Depend on
  - μ : Higgs potential parameter
  - M_2 : Wino mass
- **Source:** SuperIso [arXiv:0808.3144]





$$\tan\beta = 50$$

MARTY's users



Manual & Documentation

- Public code [arxiv: **2011.02478**]
- About **150 000 lines** of C++
- HTML **documentation** (doxygen)
- Very detailed **manuals**
- Many examples
  - Scalar theory
  - sQED
  - QED, QCD
  - Electro-Weak
  - 2HDM

◆ MinkowskiVector()

csl::Tensor MinkowskiVector (std::string const & name)

Returns a **csl::Tensor**, vector in **csl::Minkowski** space.

The vector is by definition a tensor with one index. For example, a Minkowski vector may be created by:

```
csl::Tensor X = MinkowskiVector("X");
```

Then, assuming that 'mu' is an object of type **csl::Index**,

```
X(mu)
```

means the tensor with a lowered index X_μ, and

```
X(+mu)
```

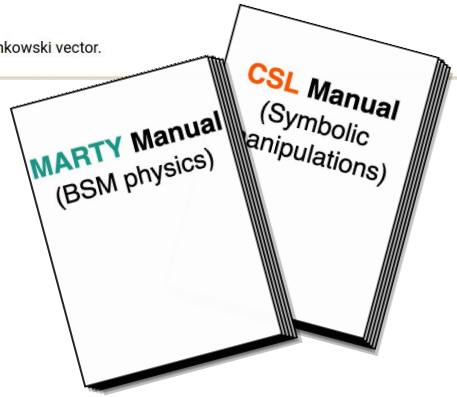
means the tensor with index up X^μ. For more information about tensors and indices see **csl::TensorElement** and **csl::Index**.

Parameters

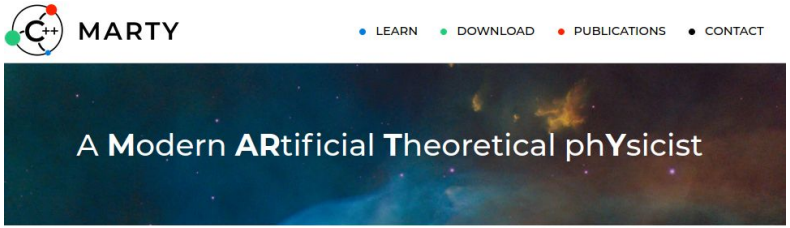
name Name of the vector.

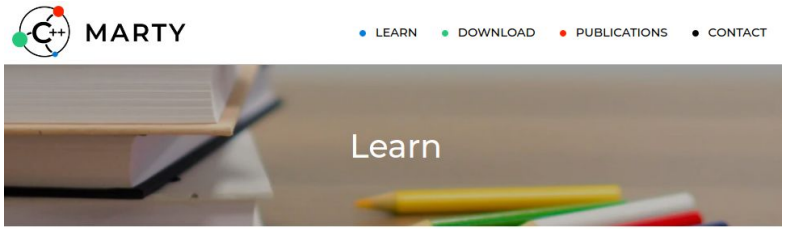
Returns

An indicial Minkowski vector.



Website: *https://marty.in2p3.fr*



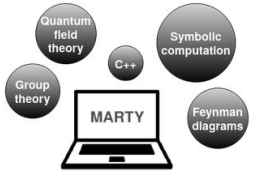


MARTY in short

MARTY is a symbolic computation program specialized for high-energy physics computations: amplitudes, cross-sections, and Wilson coefficients in a large variety of Beyond the Standard Model (BSM) models. All computations are automated and symbolic.

MARTY is composed of three modules. Its core, containing all the physics ; *CSL* (C++ Symbolic computation Library) that allows to manipulate mathematical expressions symbolically ; and *GRAFED* (Generating and Rendering Application for FEynman diagrams) that generates and displays Feynman diagrams.

Discover its features more in detail in the tutorials and the documentation.



The physics part

For starters, I recommend to see first the "Get started". You will learn the very basics of *MARTY*, starting from zero. Then, tutorials will show you how to build your own models, with examples going from a scalar theory up to the Minimal Supersymmetric Standard Model (MSSM).

Then, then manual provides comprehensive explanations about *MARTY*'s features, and more importantly about the physics implemented in it: formulas, conventions, computation methods ... The documentation is more detailed and interactive. In particular, all main objects, functions and variables of *MARTY* are discussed in it whereas the manual presents more general features.

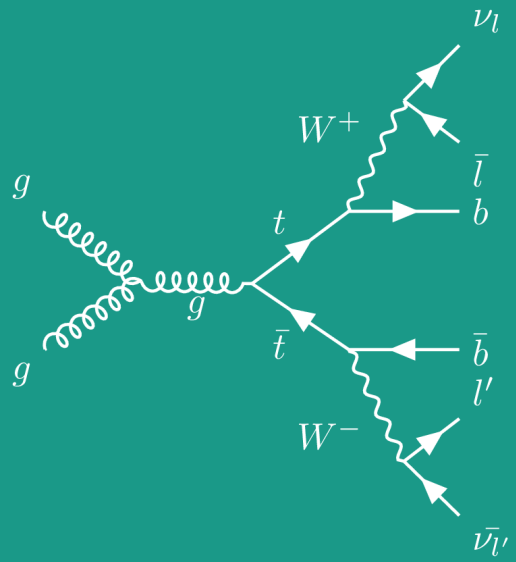
Get started
Learn the basics about MARTY

Tutorials
See how to use MARTY

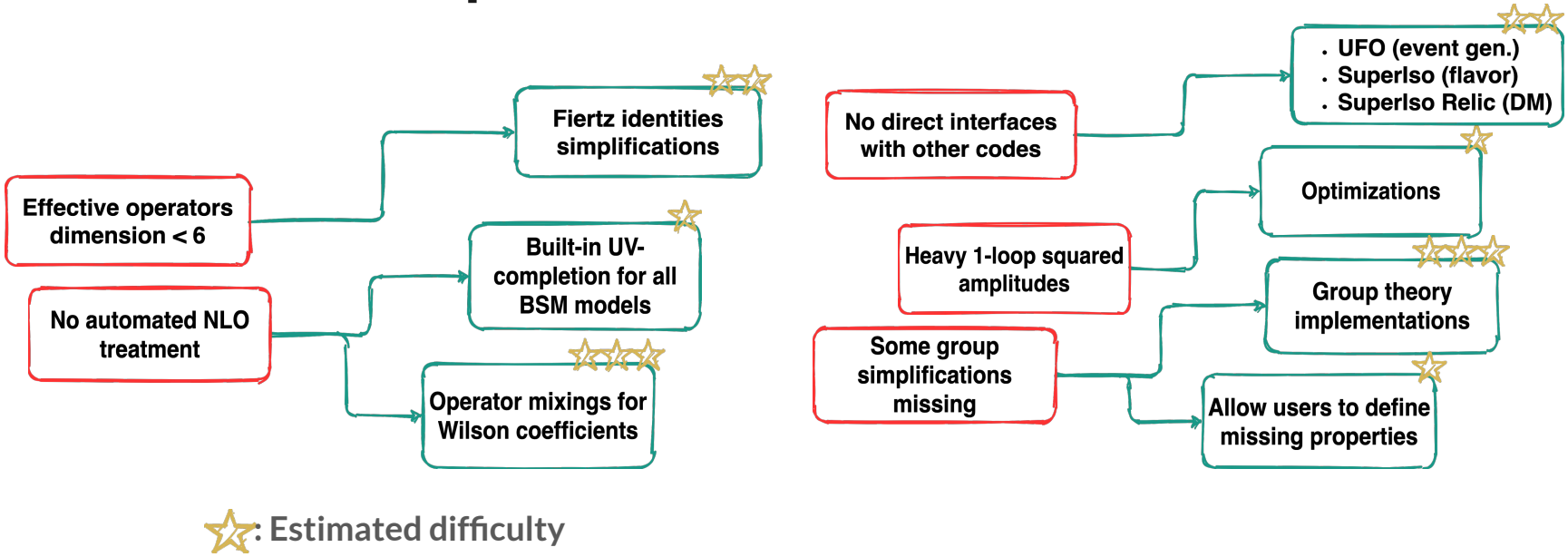
Manual
Detailed features and methods

The documentation
Interactive reference

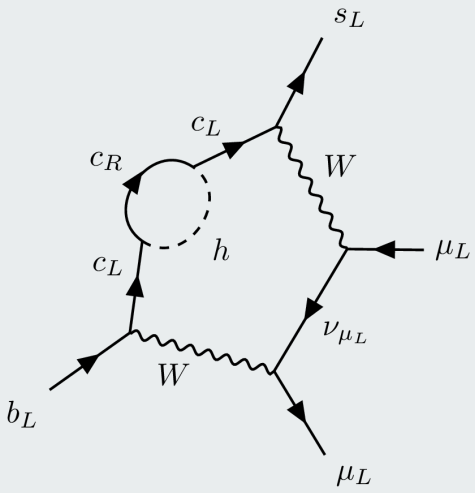
Future prospects



Future developments



Prospects

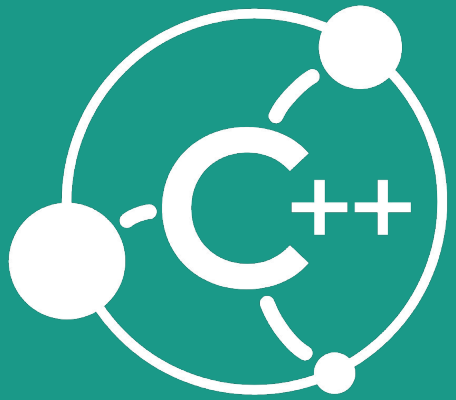


What makes MARTY unique

- All theoretical tools in one
- Very general implementation
- Independent of Mathematica
- Direct interfaces with phenomenology
  - SuperIso (+ Relic) upcoming

What you can do now

- Nothing (always an option)
- Watch ‘Back to the Future’
- **Use** MARTY for BSM
- Help **develop** it !

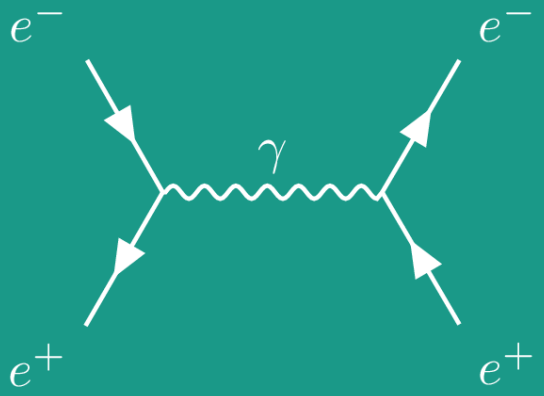


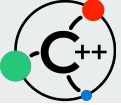
MARTY

Demonstration at 4 pm (Room B)

- CSL : 5 min
- GRAFED : 5 min
- MARTY : 35 min
  - Model building : 20 min
  - Calculation : 5 min
  - Library generation : 10 min
- Questions : 15 min

Backup





Dependencies

- **C++ Standard Library (2017 version)**
- Numerical matrix diagonalization: **GNU Scientific Library (GSL)**
- Numerical integrals: **LoopTools**
- Feynman diagram drawing / edition: **Qt (open-source and free version)**

MARTY's user interface

$h \to \gamma\gamma$ (SM, 1-loop)

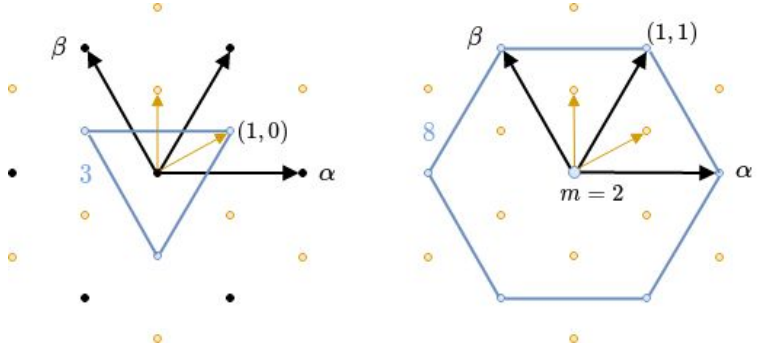
```
SM_Model SM;

auto res = ComputeAmplitude(
      Order::OneLoop,
      {
       Incoming("h"),
       Outgoing("A"),
       Outgoing("A")
      }
);

Display(res); // expressions
Show(res);    // diagrams
```

$b \to s\gamma$ (2HDM, 1-loop)

```
THDM_Model THDM;

DisableParticle("H^+"); // SM-like
ForceParticle("t");     // Only top quark

auto res = ComputeAmplitude(
      Order::OneLoop,
      {
       Incoming("b"),
       Outgoing("s"),
       Outgoing("A")
      }
);
```

Group theory

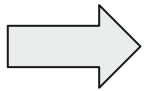
- **All semi-simple algebras**
- SU(3) is **A_2: 2 dinkin labels**
- Tensor product decomposition

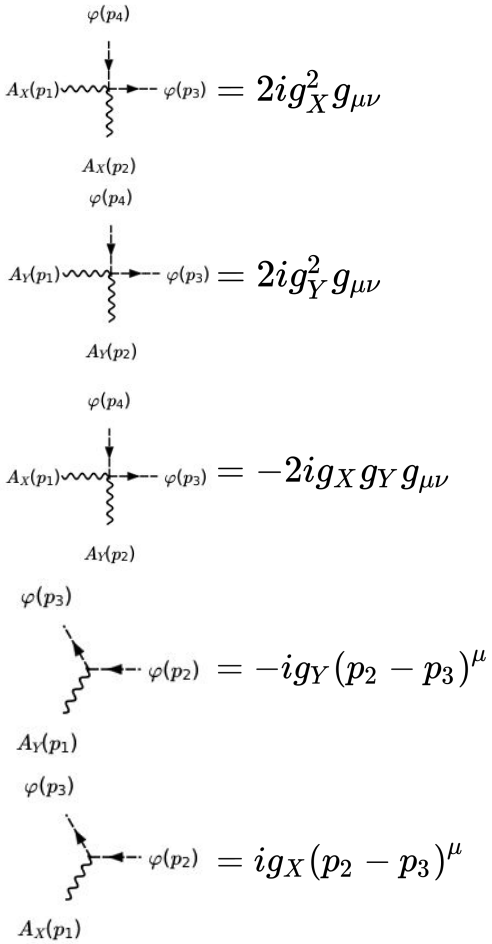



```
auto SU3 = CreateAlgebra(AlgebraType::A, 2);

auto q     = GetIrrep(SU3, {1, 0});
auto q_bar = GetIrrep(SU3, {0, 1});
auto gluon = GetIrrep(SU3, {1, 1});

cout << "3 x 3 x 3 = " << q * q * q << endl;
// >> 3 x 3 x 3 = 1 + 8 + 8 + 10  (Total dim = 27)

cout << "8 x 8     = " << gluon * gluon << endl;
// >> 8 x 8     = 1 + 8 + 8 + 10 + 10 + 27  (Total dim = 64)
```

Scalar $U(1)_X$ x $U(1)_Y$

```
Model sQED;
AddGaugedGroup(sQED, GroupType::U1, "X");
AddGaugedGroup(sQED, GroupType::U1, "Y");
Init(sQED);

Particle phi = scalarboson_s("\\phi", sQED);
SetMass(phi, "m");
SetGroupRep(phi, "X", 1);
SetGroupRep(phi, "Y", -1);
AddField(sQED, phi);

cout << sQED << endl;

auto rules = ComputeFeynmanRules(sQED);
Display(rules);
Show(rules);
```

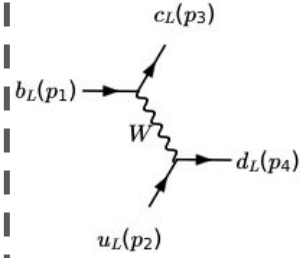




Amplitude (SM)

```
SM_model SM;

Particle bL = GetParticle(SM, "b_L");
Particle cL = GetParticle(SM, "c_L");
Particle uL = GetParticle(SM, "u_L");
Particle dL = GetParticle(SM, "d_L");

auto bu_to_cd = ComputeAmplitudesWithDiagrams(
        Order::TreeLevel,
        SM,
        {Incoming(bL), Incoming(uL),
        Outgoing(cL), Outgoing(dL)}
        );

Display(bu_to_cd);
Show(bu_to_cd);
```

Bare result



$$= -iAb_{0045}\,Prop(p1-p3, M_W^2)$$
$$\cdot P_{L\beta_1\gamma_1} P_{L\beta_2\gamma_2} \gamma^{\mu}_{\alpha_1\beta_1} \gamma_{\mu\alpha_2\beta_2} EXT^{ijkl}_{\alpha_1\alpha_2\gamma_1\gamma_2}$$

$$Ab_{0045} = Ab_{0031} \cdot Ab_{0039}$$
$$= \sqrt{\frac{2\pi\alpha_{em}}{\sin^2\theta_W}}|V_{ud}| \cdot \sqrt{\frac{2\pi\alpha_{em}}{\sin^2\theta_W}}|V_{cb}| = \frac{2\pi\alpha_{em}}{\sin^2\theta_W} V_{ud} V_{cd}$$

$$EXT^{ijkl}_{\alpha_1\alpha_2\gamma_1\gamma_2} = \bar{c}^k_{\alpha_1}(p_3) b^i_{\gamma_1}(p_1) \bar{d}^l_{\alpha_2}(p_4) u^j_{\gamma_2}(p_2)$$

Translated result

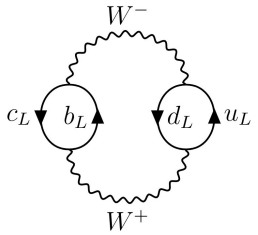
$$i\mathcal{M}^{ijkl} = -i\frac{2\pi\alpha_{em}}{\sin^2\theta_W (t-M_W^2)} V_{ud} V_{cb} \left(\bar{c}^k(p_3)\gamma^{\mu} P_L b^i(p_1)\right) \left(\bar{d}^l(p_4)\gamma_{\mu} P_L u^j(p_2)\right)$$

Cross-section (SM)

- Polarization sums
  - $\sum_i v^i(p)\bar{v}^i(p) = \gamma^\mu p_\mu - m$
  - $\sum_i u^i(p)\bar{u}^i(p) = \gamma^\mu p_\mu + m$
- Dirac traces
  - Non chiral $Tr\left(\gamma^{\mu_1} \dots \gamma^{\mu_n}\right)$
  - Chiral $Tr\left(\gamma^5 \gamma^{\mu_1} \dots \gamma^{\mu_n}\right)$
- Color traces $Tr\left(T_R^{A_1} \dots T_R^{A_N}\right)$

$$d\sigma = \frac{1}{2^2} \sum_{ijkl=\pm 1/2} (M^{ijkl})^\dagger M^{ijkl} d\Pi_{LIPS}$$

$$= \frac{1}{4}$$



$$d\Pi_{LIPS}$$

```
Expr xSection = SM.computeCrossSection(bu_to_cd.expressions);
std::cout << xSection << std::endl;
```

$$m_q = 0$$

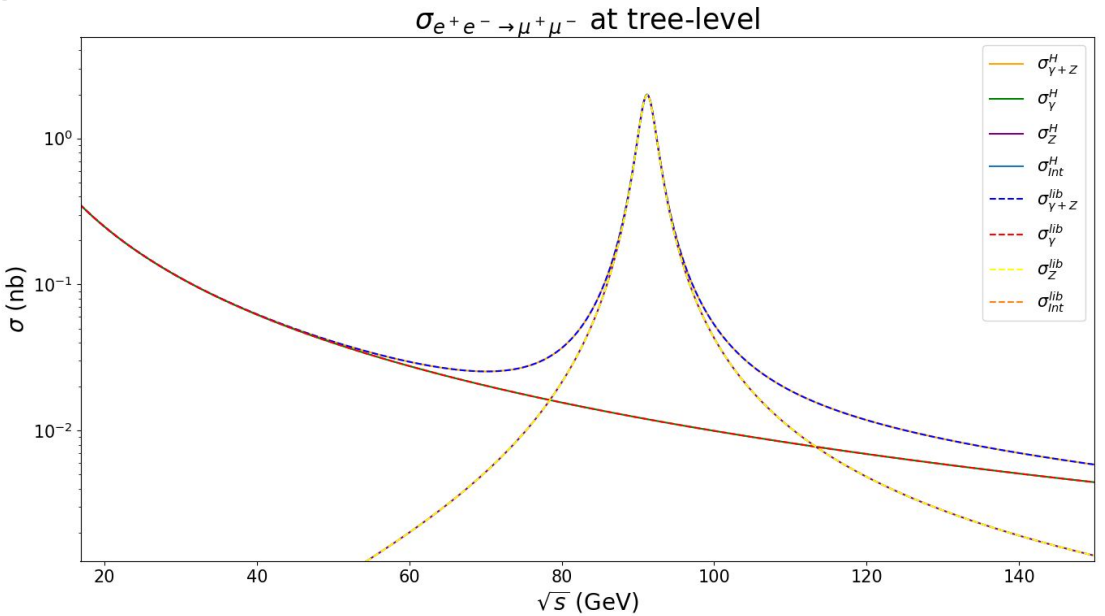
$$\epsilon_{\mu\nu\rho\sigma} p_1^\mu p_2^\nu p_3^\rho p_4^\sigma = 0$$

$$\frac{d\sigma}{d\Pi_{LIPS}} = \frac{144\pi^2 \alpha_{em}^2}{\sin^4 \theta_W} V_{ud}^2 V_{cb}^2 \frac{p_1^\mu p_{2\mu} p_3^\nu p_{4\nu}}{(M_W^2 - 2p_1^\mu p_{3\mu})^2}$$

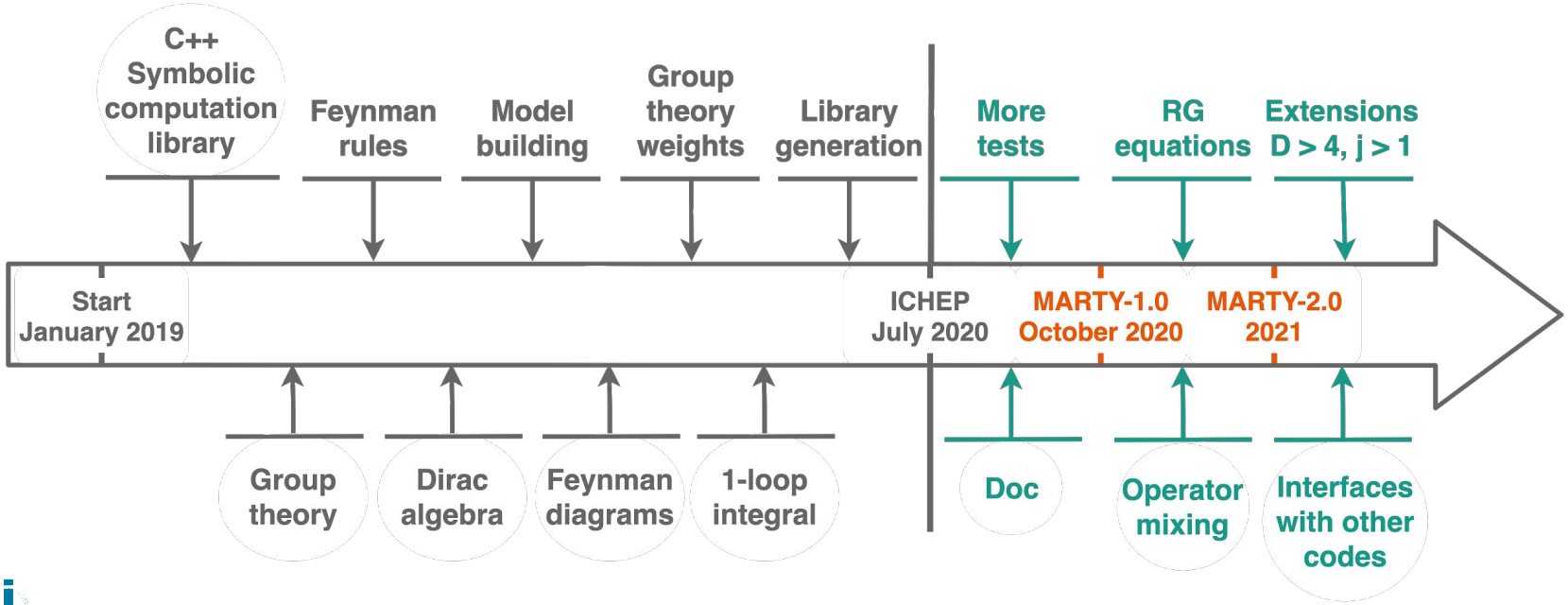
$$= \frac{144\pi^2 \alpha_{em}^2}{\sin^4 \theta_W} V_{ud}^2 V_{cb}^2 \frac{s^2}{(M_W^2 - t)^2}$$

Library example

- **Z resonance** ~ 91 GeV
- $\sigma_{SM}(e^+e^- \to \mu^+\mu^-)$
- Exact match with known results
- $\sigma_{peak} \approx 2$ nb
- $\Gamma_Z \approx 2.5$ GeV
- **May vary any parameter**
- **LEP** experiments



Road map


C++ Symbolic computation library
Feynman rules
Model building
Group theory weights
Library generation
More tests
RG equations
Extensions D > 4, j > 1
Start January 2019
ICHEP July 2020
MARTY-1.0 October 2020
MARTY-2.0 2021
Group theory
Dirac algebra
Feynman diagrams
1-loop integral
Doc
Operator mixing
Interfaces with other codes