Automation of NLO Calculations for BSM

A report on recent progress in 1-Loop BSM automation

Jean-Nicolas Lang

Universität Zürich

Probing Physics Beyond the Standard Model with Precision
Mainz
8.3.2018

Era after the Higgs discovery

- SM is very well consistent with LHC
- Recent progress in higher precision calculations (α_s^2, α)
  EW corrections are becoming state of the art.
- EW BSM extension of SM appealing.
  Differences to SM expected of the order of EW corrections.
  EW corrections in BSM required (?)
- How difficult is it to generalize this progress to BSM NLO?
  Why do we need automation?

From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation

$\sigma, \mathrm{d}\sigma$

$\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

$\mathcal{L}$

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

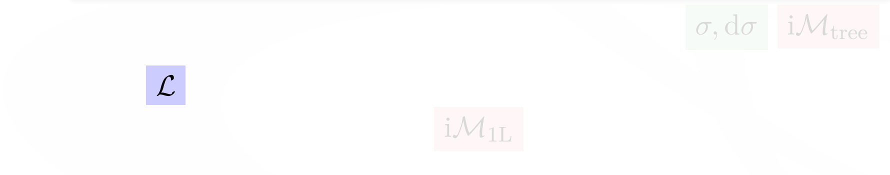
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



$\sigma, \mathrm{d}\sigma$ $\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

$\mathcal{L}$

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation

$\sigma, \mathrm{d}\sigma$ $\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

$\mathcal{L}$

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation

$\sigma, \mathrm{d}\sigma$

$\mathrm{i}\mathcal{M}_{\text{tree}}$

$\mathcal{L}$

$\mathrm{i}\mathcal{M}_{\text{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation

Model generator

$\mathcal{L}$

$\sigma, \mathrm{d}\sigma$

$\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

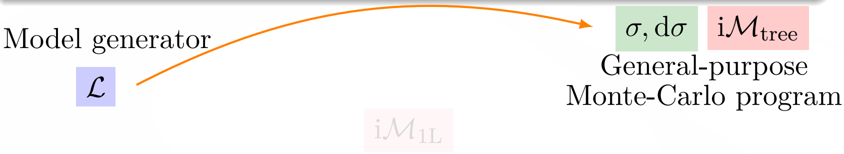
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



Model generator
$\mathcal{L}$
$\sigma, \mathrm{d}\sigma$
$\mathrm{i}\mathcal{M}_{\mathrm{tree}}$
General-purpose
Monte-Carlo program
$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

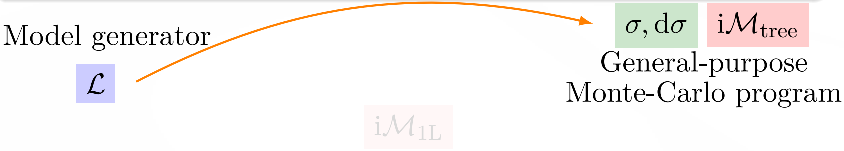
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



Model generator

$\mathcal{L}$

$\sigma, \mathrm{d}\sigma$ $\quad$ $\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

General-purpose Monte-Carlo program

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

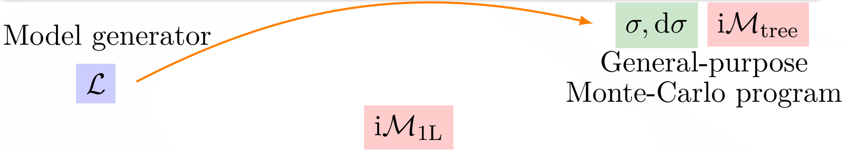
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



Model generator

$\mathcal{L}$

$\sigma, \mathrm{d}\sigma$ $\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

General-purpose Monte-Carlo program

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

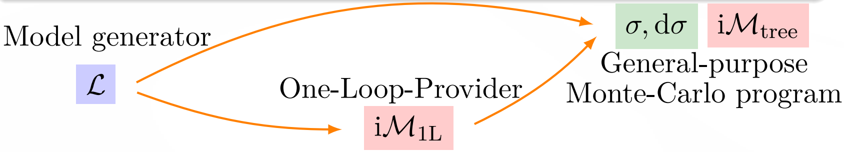
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



Model generator

$\mathcal{L}$

One-Loop-Provider

$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$

$\sigma, \mathrm{d}\sigma$ $\mathrm{i}\mathcal{M}_{\mathrm{tree}}$

General-purpose Monte-Carlo program

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

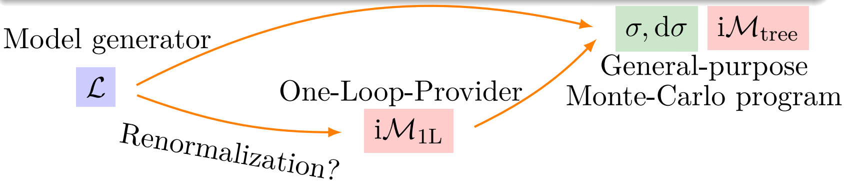
From theory to collider predictions: Big picture

Typical problem:

- Interested in a particular **theory/Lagrangian** defined by (gauge-)symmetry and field content
- A (class of) **process(es)** with (potentially) interesting **signatures** for this extension

Goal of automation:

- Avoid model/process by model/process implementation
- Automatize model generation and process computation



Model generator
$\mathcal{L}$
One-Loop-Provider
$\mathrm{i}\mathcal{M}_{\mathrm{1L}}$
Renormalization?
$\sigma, \mathrm{d}\sigma$
$\mathrm{i}\mathcal{M}_{\mathrm{tree}}$
General-purpose
Monte-Carlo program

- Tree-level automation through model generator + GPMC
- One-loop building blocks through specialized OLP

Content of this talk

From theory to Feynman rules

Automation of amplitude generation and computation

1-Loop automation: First steps in BSM automation

Beyond pure QCD: Challenges at NLO EW

From theory to Feynman rules

Model generators

Feynrules:

A Mathematica package to calculate Feynman rules.

Written in: Mathematica

Input: Model file informations (Mathematica syntax)

Ref: `feynrules.irmp.ucl.ac.be`

SARAH:

A Mathematica package for building and analyzing models.

Written in: Mathematica

Input: Model file informations (Mathematica syntax)

Ref: `sarah.hepforge.org`

LanHEP:

A program for Feynman rules generation.

Written in: C

Input: Model file informations (custom script syntax)

Ref: `theory.sinp.msu.ru/~semenov/lanhep.html`

Essential ingredients for automation

Tools are characterized by high-level/compact language input:

- user-friendly
- reuseability/efficiency
  - allows to e.g. take over most parts of the SM
  - reduces the risk of mistakes

E.g. LanHEP input(QED)

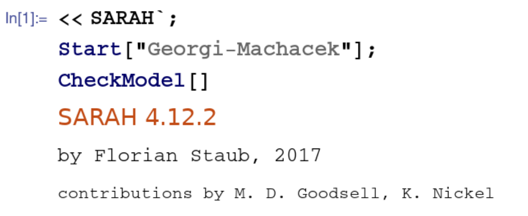
```
model QED/1.
parameter ee=0.31333:'elementary electric charge'.
spinor e1/E1:(electron, mass me=0.000511).
vector A/A:(photon).
let F^mu^nu=deriv^mu*A^nu-deriv^nu*A^mu.
lterm -1/4*(F^mu^nu)**2 - 1/2*(deriv^mu*A^mu)**2.
lterm E1*(i*gamma*deriv+me)*e1.
lterm ee*E1*gamma*A*e1.
```

Essential ingredients for automation

The user is assisted by automated model checks:

- Normalization of Lagrangian
- Mass-spectrum
- Hermiticity
- Local & global symmetries
- Presence/absence of anomalies

E.g. SARAH: CheckModel call

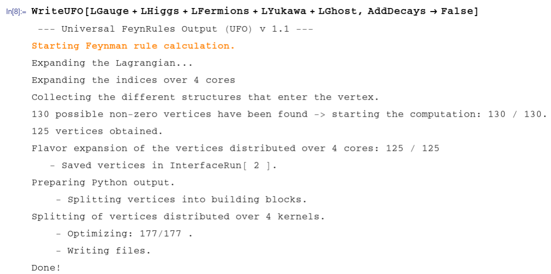


```
In[1]:= << SARAH`;
        Start["Georgi-Machacek"];
        CheckModel[]
        SARAH 4.12.2
        by Florian Staub, 2017
        contributions by M. D. Goodsell, K. Nickel
```

Essential ingredients for automation

The UFO format eliminates the need for different interfaces:

E.g. in FeynRules: WriteUFO call



```
In[8]:= WriteUFO[LGauge + LHiggs + LFermions + LYukawa + LGhost, AddDecays → False]
         --- Universal FeynRules Output (UFO) v 1.1 ---
        Starting Feynman rule calculation.
        Expanding the Lagrangian...
        Expanding the indices over 4 cores
        Collecting the different structures that enter the vertex.
        130 possible non-zero vertices have been found -> starting the computation: 130 / 130.
        125 vertices obtained.
        Flavor expansion of the vertices distributed over 4 cores: 125 / 125
           - Saved vertices in InterfaceRun[ 2 ].
        Preparing Python output.
            - Splitting vertices into building blocks.
        Splitting of vertices distributed over 4 kernels.
            - Optimizing: 177/177 .
            - Writing files.
        Done!
```

Major GPMC tools (MadGraph, Sherpa, Whizard, ...) support the UFO format and can thus compute hard processes/perform spectrum calculations for a given model file at LO.

Essential ingredients for automation

The UFO format, arXiv:1108.2040

UFO:

Modular model file format.

Written in: Python

Ref: `feynrules.irmp.ucl.ac.be`

- Flexible and modular
- No assumptions on vertex structure
- Almost no parsing necessary

Essential ingredients for automation

The UFO format, arXiv:1108.2040

UFO:

Modular model file format.

Written in: Python

Ref: `feynrules.irmp.ucl.ac.be`

- Flexible and modular
- No assumptions on vertex structure
- Almost no parsing necessary

```
Vertex(name='V_149',
          particles=[P.H, P.H, P.H],
          color=['1'],
          couplings={(0,0):C.GC_102},
          lorentz=[L.SSS1])
```

Essential ingredients for automation

The UFO format, arXiv:1108.2040

UFO:

Modular model file format.

Written in: Python

Ref: `feynrules.irmp.ucl.ac.be`

- Flexible and modular
- No assumptions on vertex structure
- Almost no parsing necessary

```
Vertex(name='V_149',
       particles=[P.H, P.H, P.H],
       color=['1'],
       couplings={(0,0):C.GC_102},
       lorentz=[L.SSS1])
```

```
Particle(name='H', antiname='H', spin=1,
         color=1, mass=Param.MH,
         width=Param.WH, charge=0,
         GhostNumber=0, LeptonNumber=0, Y=0)
```

Essential ingredients for automation

The UFO format, arXiv:1108.2040

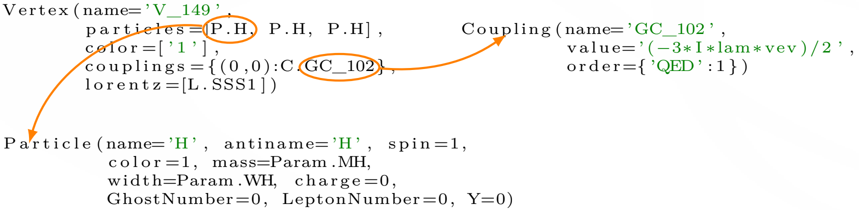
UFO:

Modular model file format.

Written in: Python

Ref: `feynrules.irmp.ucl.ac.be`

- Flexible and modular
- No assumptions on vertex structure
- Almost no parsing necessary



```
Vertex(name='V_149',
       particles=[P.H, P.H, P.H],
       color=['1'],
       couplings={(0,0):C.GC_102},
       lorentz=[L.SSS1])
```

```
Coupling(name='GC_102',
         value='(−3*I*lam*vev)/2',
         order={'QED':1})
```

```
Particle(name='H', antiname='H', spin=1,
         color=1, mass=Param.MH,
         width=Param.WH, charge=0,
         GhostNumber=0, LeptonNumber=0, Y=0)
```

Essential ingredients for automation

The UFO format, arXiv:1108.2040

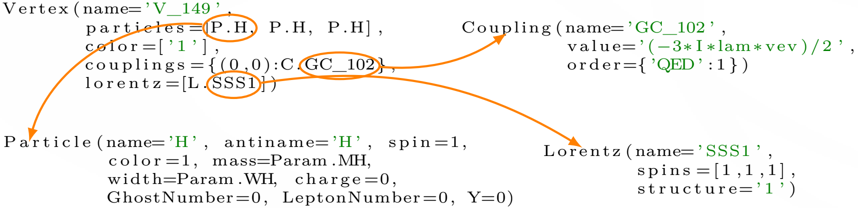
UFO:

Modular model file format.

Written in: Python

Ref: `feynrules.irmp.ucl.ac.be`

- Flexible and modular
- No assumptions on vertex structure
- Almost no parsing necessary



```
Vertex(name='V_149',
       particles=[P.H, P.H, P.H],
       color=['1'],
       couplings={(0,0):C.GC_102},
       lorentz=[L.SSS1])
```

```
Coupling(name='GC_102',
         value='(-3*I*lam*vev)/2',
         order={'QED':1})
```

```
Particle(name='H', antiname='H', spin=1,
         color=1, mass=Param.MH,
         width=Param.WH, charge=0,
         GhostNumber=0, LeptonNumber=0, Y=0)
```

```
Lorentz(name='SSS1',
        spins=[1,1,1],
        structure='1')
```

Automation of amplitude generation and computation

One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

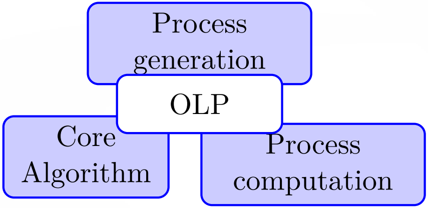
Internal machinery:

One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

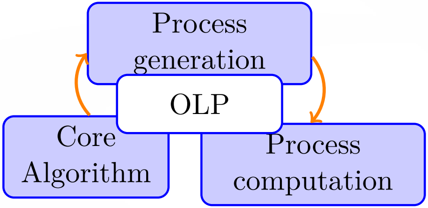


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

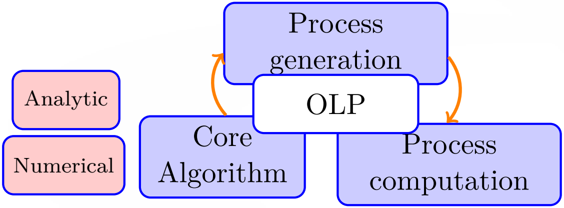


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

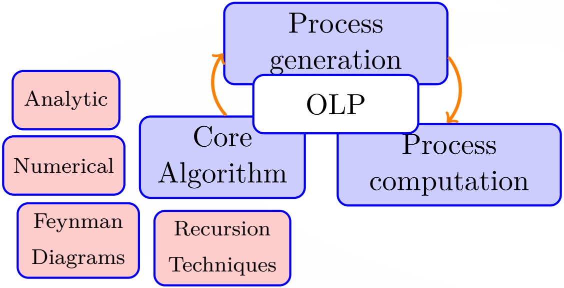


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

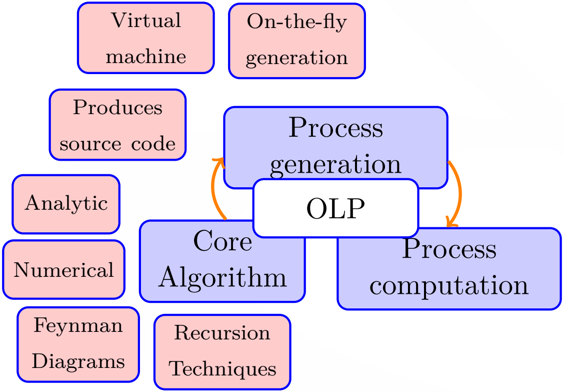


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

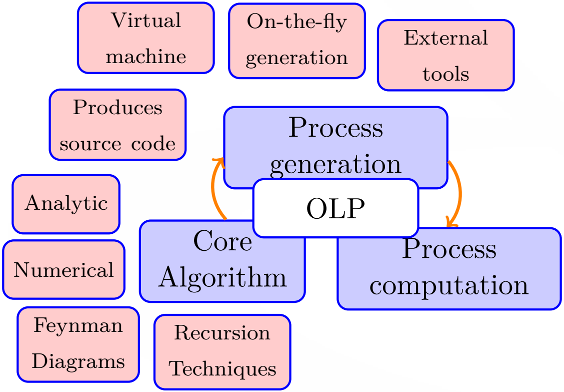


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

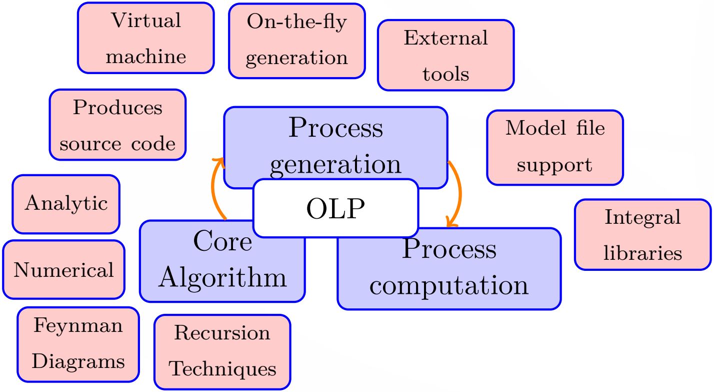


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

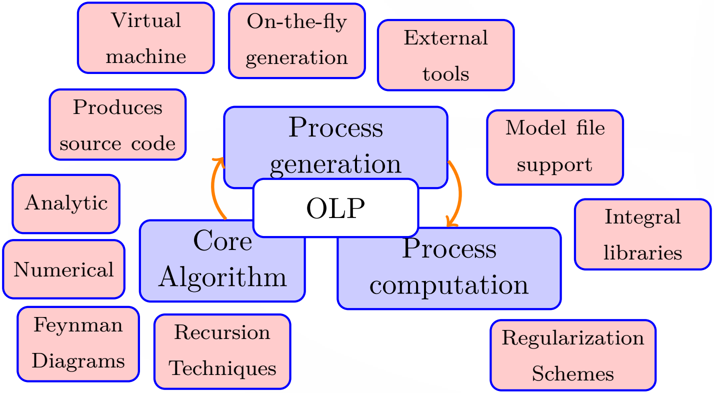


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:

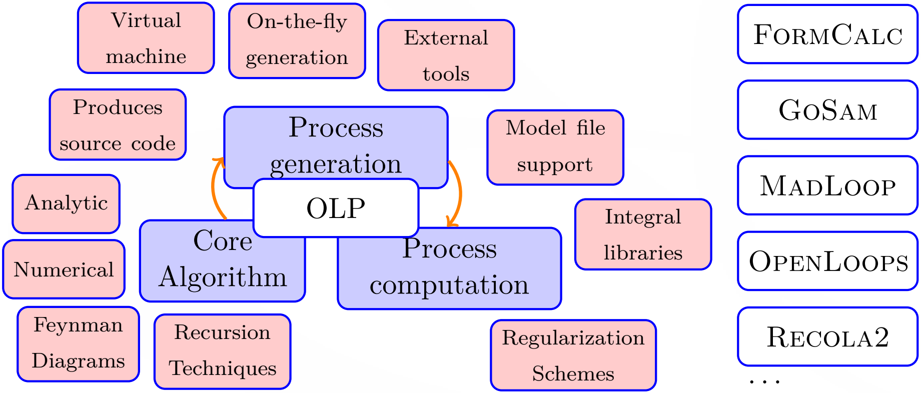


One-Loop Technology Landscape

External view of OLP

- Computes tree/real/virtual/squared/correlated amplitudes
- Interface to MC tools

Internal machinery:



Virtual machine

On-the-fly generation

External tools

Produces source code

Process generation

Model file support

Analytic

OLP

Integral libraries

Numerical

Core Algorithm

Process computation

Feynman Diagrams

Recursion Techniques

Regularization Schemes

FormCalc

GoSam

MadLoop

OpenLoops

Recola2

. . .

FormCalc

arXiv:1604.04611

FormCalc:

Calculates and simplifies tree-level and one-loop Feynman diagrams.

Written in: Mathematica

Model file input: Uses FeynArts model files

Amplitude output as: Fortran95, C source code

External dependencies:

- FeynArts, Form: Amplitude generation and analytic simplifications
- LoopTools: Tensor integral library

- Amplitude output is modular and can be used in other (automated) tool chains.
- Implements the SM (+BFM), MSSM at NLO and the 2HDM (at NLO QCD?).

FormCalc

arXiv:1604.04611

FormCalc:

Calculates and simplifies tree-level and one-loop Feynman diagrams.

Written in: Mathematica

Model file input: Uses FeynArts model files

Amplitude output as: Fortran95, C source code

External dependencies:

- FeynArts, Form: Amplitude generation and analytic simplifications
- LoopTools: Tensor integral library

- Amplitude output is modular and can be used in other (automated) tool chains.
- Implements the SM (+BFM), MSSM at NLO and the 2HDM (at NLO QCD?).

FormCalc

arXiv:1604.04611

FormCalc:

Calculates and simplifies tree-level and one-loop Feynman diagrams.

Written in: Mathematica
Model file input: Uses FeynArts model files
Amplitude output as: Fortran95, C source code
External dependencies:

- FeynArts, Form: Amplitude generation and analytic simplifications
- LoopTools: Tensor integral library

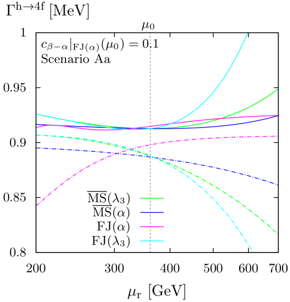
- Amplitude output is modular and can be used in other (automated) tool chains.
- Implements the SM (+BFM), MSSM at NLO and the 2HDM (at NLO QCD?).

FormCalc

Selection of BSM applications in automated frameworks

arXiv:1710.07598,arXiv:1801.0729:
Higgs decays in the 2HDM and HSESM using Prophecy.

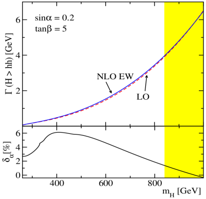
$$p_0 = p^{(1)} + \delta p^{(1)}(p^{(1)}) = p^{(2)} + \delta p^{(2)}(p^{(2)})$$



arXiv:1602.05495,arXiv:1705.02209:
Renormalization of the NMSSM and application to Higgs decays in SLoopS.

	tree-level	One-loop			
	(MeV)	$t_{134A_1A_2}$	$OS_{34h_2A_1A_2H^+}$	$\overline{\mathrm{DR}}\,Q_M$	$\overline{\mathrm{DR}}\,Q_{\mathrm{susy}}$
$h_2^0 \to A_1^0 A_1^0$	47.9	128%	-12%	0.4%	-0.4%
$h_3^0 \to h_1^0 h_2^0$	22.1	116%	79%	52%	-1.7%
$h_3^0 \to \tilde{\chi}_1^0 \tilde{\chi}_3^0$	35.2	122%	-3%	2%	0.3%
$h_3^0 \to \tilde{\chi}_2^0 \tilde{\chi}_3^0$	33.8	126%	-35%	3%	1.1%
$h_3^0 \to \tilde{\chi}_1^+ \tilde{\chi}_1^-$	45.5	1%	-11%	-9%	-7.4%
$A_2^0 \to Z^0 h_2^0$	18.6	120%	80%	-56%	-14.5%
$A_2^0 \to \tilde{\chi}_1^0 \tilde{\chi}_1^0$	33.0	28%	13%	0.3%	-1.6%
$A_2^0 \to \tilde{\chi}_1^0 \tilde{\chi}_3^0$	24.4	130%	-31%	8%	6.2%
$A_2^0 \to \tilde{\chi}_2^0 \tilde{\chi}_3^0$	30.2	122%	-5%	-0.4%	-1.9%
$A_2^0 \to \tilde{\chi}_1^+ \tilde{\chi}_1^-$	55.1	-10%	-1.5%	-6%	-8%
$H^+ \to W^+ h_2^0$	20.1	119%	79%	-56%	-16%
$H^+ \to \tilde{\chi}_1^+ \tilde{\chi}_3^0$	64.0	125%	-18%	3%	1.1%

arXiv:1511.08120:
Renormalization of the HSESM and application to Higgs-to Higgs decays



in SLoopS.

GoSam
arXiv:1404.7096

GoSam:

Scattering Amplitudes from Unitarity based Reduction At Integrand level

Written in: Python 2.7
Model file input: Supports the UFO format
Amplitude output as: Fortran95 source code
External dependencies:

- QGRAF, Form: Amplitude generation and analytic simplifications
- OneLoop, QCDLoop, Ninja, Samurai, Golem95: Scalar integral/Integrand reduction/Tensor integral libraries

- Implements the full SM NLO QCD + EW

GoSam
arXiv:1404.7096

GoSam:

Scattering Amplitudes from Unitarity based Reduction At Integrand level

Written in: Python 2.7
Model file input: Supports the UFO format
Amplitude output as: Fortran95 source code
External dependencies:

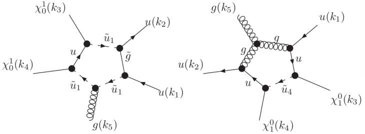
- QGRAF, Form: Amplitude generation and analytic simplifications
- OneLoop, QCDLoop, Ninja, Samurai, Golem95: Scalar integral/Integrand reduction/Tensor integral libraries

- Implements the full SM NLO QCD + EW

GoSam

BSM applications in automated frameworks

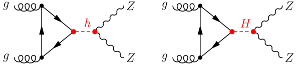
arXiv:1212.5154:
Neutralino Pair production plus jet in the MSSM at NLO QCD (UFO used)



arXiv:1308.2194:
Diphoton plus jet through graviton exchange (UFO used)

$$\begin{aligned} B_{\mu\nu,\rho\sigma}(k,m) = &\left(\eta_{\mu\rho} - \frac{k_\mu k_\rho}{m^2}\right)\left(\eta_{\nu\sigma} - \frac{k_\nu k_\sigma}{m^2}\right) \\ &+ \left(\eta_{\mu\sigma} - \frac{k_\mu k_\sigma}{m^2}\right)\left(\eta_{\nu\rho} - \frac{k_\nu k_\rho}{m^2}\right) \\ &- \frac{2}{3}\left(\eta_{\mu\nu} - \frac{k_\mu k_\nu}{m^2}\right)\left(\eta_{\rho\sigma} - \frac{k_\rho k_\sigma}{m^2}\right) \end{aligned}$$

arXiv:1512.07232:
Vector-boson pair production in the THDM at NLO QCD



arXiv:1602.05141:
Dim 8 anomalous couplings (UFO used)

$$\begin{aligned} \mathcal{O}_1 &= \frac{c_1}{\Lambda^4} G^a_{\mu\nu} G^{a,\mu\nu} W^I_{\rho\sigma} W^{I,\rho\sigma} &= \frac{c_1}{\Lambda^4}\tilde{\mathcal{O}}_1 \\ \mathcal{O}_2 &= \frac{c_2}{\Lambda^4} \tilde{G}^a_{\mu\nu} G^{a,\mu\nu} W^I_{\rho\sigma} W^{I,\rho\sigma} &= \frac{c_2}{\Lambda^4}\tilde{\mathcal{O}}_2 \\ \mathcal{O}_3 &= \frac{c_3}{\Lambda^4} G^a_{\mu\nu} G^{a,\mu\nu} \tilde{W}^I_{\rho\sigma} W^{I,\rho\sigma} &= \frac{c_3}{\Lambda^4}\tilde{\mathcal{O}}_3 \end{aligned}$$

MadLoop5

arXiv:1512.07232

MadLoop:

MadLoop5 is part of MadGraph5_aMC@NLO and responsible for the code generation and (numerical) computation of one-loop matrix elements.

Written in: Python 2.7

Model file input: Supports the NLO UFO format

Amplitude output as: Fortran95 source code

External dependencies:

- OneLoop,QCDLoop,Ninja,CutTools,Collier,. . . : Scalar integral/Integrand reduction/Tensor integral libraries

- Shipped with the full SM NLO QCD + EW and various BSM NLO QCD as UFO model files

MadLoop5

arXiv:1512.07232

MadLoop:

MadLoop5 is part of MadGraph5_aMC@NLO and responsible for the code generation and (numerical) computation of one-loop matrix elements.

Written in: Python 2.7
Model file input: Supports the NLO UFO format
Amplitude output as: Fortran95 source code
External dependencies:

- OneLoop,QCDLoop,Ninja,CutTools,Collier,...: Scalar integral/Integrand reduction/Tensor integral libraries

- Shipped with the full SM NLO QCD + EW and various BSM NLO QCD as UFO model files

MadLoop5

Selection of BSM applications in automated frameworks

arXiv:1412.5589 (UFO used):

Stop and sgluon dynamics in simplified model

$$\mathcal{L}_3 = D_\mu \sigma_3^\dagger D^\mu \sigma_3 - m_3^2 \sigma_3^\dagger \sigma_3 + \frac{i}{2}\bar{\chi}\partial\!\!\!/\chi - \frac{1}{2} m_\chi \bar{\chi}\chi$$
$$+ \left[\sigma_3 \bar{t}\left(\tilde{g}_L P_L + \tilde{g}_R P_R\right)\chi + \text{h.c.}\right]$$

First application using a fully automated renormalization procedure (NLOCT)

$$\delta Z_g = \delta Z_g^{(SM)} - \frac{g_s^2}{96\pi^2}\left[\frac{1}{\bar{\epsilon}} - \log\frac{m_3^2}{\mu_R^2}\right]$$

$$\delta Z_{\sigma_3} = 0 \quad \text{and} \quad \delta m_3^2 = -\frac{g_s^2 m_3^2}{12\pi^2}\left[\frac{3}{\bar{\epsilon}} + 7 - 3\log\frac{m_3^2}{\mu_R^2}\right]$$

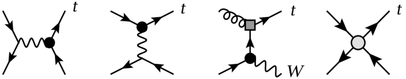
$$R_2^{\sigma_3^\dagger \sigma_3} = \frac{i g_s^2}{72\pi^2}\delta_{c_1 c_2}\left[3m_3^2 - p^2\right]$$

$$R_2^{g\sigma_3^\dagger \sigma_3} = \frac{53 i g_s^3}{576\pi^2} T^{a_1}_{c_2 c_3}\left(p_2 - p_3\right)^{\mu_1}$$

$$R_2^{gg\sigma_3^\dagger \sigma_3} = \frac{i g_s^4}{1152\pi^2}\eta^{\mu_1\mu_2}\left[3\delta^{a_1 a_2} - 187\{T^{a_1}, T^{a_2}\}\right]_{c_3 c_4}$$

arXiv:1508.00564 (UFO used):

Top production at NLO QCD EFT

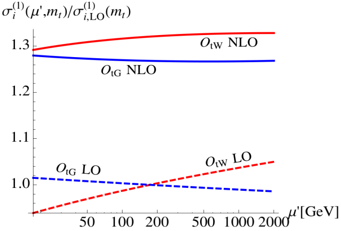


$$O^{(3)}_{\varphi Q} = i\frac{1}{2}y_t^2\left(\varphi^\dagger \overleftrightarrow{D}^I_\mu \varphi\right)\left(\bar{Q}\gamma^\mu \tau^I Q\right)$$

$$O_{tW} = y_t g_W (\bar{Q}\sigma^{\mu\nu}\tau^I t)\tilde{\varphi} W^I_{\mu\nu}$$

$$O_{tG} = y_t g_s (\bar{Q}\sigma^{\mu\nu} T^A t)\tilde{\varphi} G^A_{\mu\nu}$$

$$O^{(3)}_{qQ,rs} = (\bar{q}_r \gamma_\mu \tau^I q_s)(\bar{Q}\gamma^\mu \tau^I Q)$$



OpenLoops

arXiv:1111.5206

OpenLoops:

OpenLoops is an algorithm for the fast numerical evaluation of tree and one-loop matrix elements for any Standard Model process.

Written in: Fortran95

Model file input: OpenLoops model files (Fortran95)
Working on supporting the UFO model file

Amplitude output as: Pre-generated process libraries.

External dependencies:

- FeynArts, OpenLoops: Amplitude generation.
- OneLoop, QCDLoop, CutTools, Samurai, Collier: Scalar integral/Integrand reduction/Tensor integral libraries

- Supports the SM (NLO QCD + EW) and 2HDM NLO QCD

OpenLoops

arXiv:1111.5206

OpenLoops:

OpenLoops is an algorithm for the fast numerical evaluation of tree and one-loop matrix elements for any Standard Model process.

Written in: Fortran95
Model file input: OpenLoops model files (Fortran95)
Working on supporting the UFO model file
Amplitude output as: Pre-generated process libraries.
External dependencies:

- FeynArts, OpenLoops: Amplitude generation.
- OneLoop, QCDLoop, CutTools, Samurai, Collier: Scalar integral/Integrand reduction/Tensor integral libraries

- Supports the SM (NLO QCD + EW) and 2HDM NLO QCD

RECOLA2

arXiv:1711.07388

RECOLA2:

RECOLA2 implements a fully recursive algorithm to compute tree and one-loop amplitudes allowing for extreme calculations with many final particle states.

Written in:	FORTRAN95
Model file input:	RECOLA2 model files (FORTRAN95) Supports the LO UFO model file
Amplitude output as:	Generated in memory, no source code produced
External dependencies:	

- COLLIER: Tensor integral library

- SM, Higgs singlet and 2HDM NLO QCD + EW + BFM

RECOLA2

arXiv:1711.07388

RECOLA2:

RECOLA2 implements a fully recursive algorithm to compute tree and one-loop amplitudes allowing for extreme calculations with many final particle states.

Written in: FORTRAN95
Model file input: RECOLA2 model files (FORTRAN95)
Supports the LO UFO model file
Amplitude output as: Generated in memory, no source code produced
External dependencies:

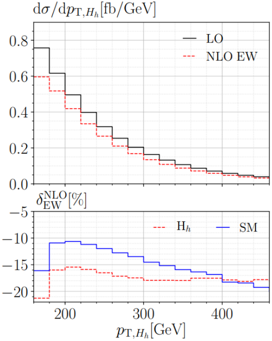
- COLLIER: Tensor integral library

- SM, Higgs singlet and 2HDM NLO QCD + EW + BFM

Recola2

BSM applications in automated frameworks

arXiv:1705.06053:
Higgs production at NLO EW in the 2HDM and HSESM (UFO used)



arXiv:1803:?:
Diboson production including Dim 6 and Dim 8 anomalous couplings (UFO used)

$$
\begin{aligned}
\mathcal{O}_{WWW} &= \frac{g_w^3}{4}\epsilon_{ijk}[W^i_{\mu\nu}W^{\nu\rho\,j}W_\rho^{\mu\,k}],\\
\mathcal{O}_W &= \mathrm{i}g_w(D_\mu\Phi)^\dagger\frac{\tau_k}{2}W^{\mu\nu\,k}(D_\nu\Phi),\\
\mathcal{O}_B &= -\mathrm{i}\frac{g_1}{2}(D_\mu\Phi)^\dagger B^{\mu\nu}(D_\nu\Phi),\\
\mathcal{O}_{\tilde{W}WW} &= -\frac{g_w^3}{4}\epsilon_{ijk}[\tilde{W}^i_{\mu\nu}W^{\nu\rho\,j}W_\rho^{\mu\,k}],\\
\mathcal{O}_{\tilde{W}} &= -\mathrm{i}g_w(D_\mu\Phi)^\dagger\frac{\tau_k}{2}\tilde{W}^{\mu\nu\,k}(D_\nu\Phi),
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{O}_{BW} &= -\mathrm{i}\,\Phi^\dagger B_{\mu\nu}\frac{\tau_i}{2}W^{\mu\rho\,i}\,\{D_\rho, D^\nu\}\,\Phi + \mathrm{h.c.},\\
\mathcal{O}_{WW} &= \mathrm{i}\,\Phi^\dagger\frac{\tau_i}{2}\frac{\tau_j}{2}W^i_{\mu\nu}W^{\mu\rho\,j}\,\{D_\rho, D^\nu\}\,\Phi + \mathrm{h.c.},\\
\mathcal{O}_{BB} &= \mathrm{i}\,\Phi^\dagger B_{\mu\nu}B^{\mu\rho}\,\{D_\rho, D^\nu\}\,\Phi + \mathrm{h.c.},\\
\mathcal{O}_{\widetilde{B}W} &= -\mathrm{i}\,\Phi^\dagger\widetilde{B}_{\mu\nu}\frac{\tau_i}{2}W^{\mu\rho\,i}\,\{D_\rho, D^\nu\}\,\Phi + \mathrm{h.c.},
\end{aligned}
$$

1-Loop automation: First steps in BSM automation

Ingredients for 1-loop BSM amplitudes

Renormalized 1-loop amplitude:

$$\mathcal{M}_1 = \sum_k \underbrace{c_{k,\mu_1,\mu_2,\dots}}_{\text{I}} T_k^{\mu_1,\mu_2,\dots} + \underbrace{\mathcal{M}_{\mathbf{CT}}}_{\text{II}} + \underbrace{\mathcal{M}_{\mathbf{R_2}}}_{\text{III}}$$

I Computation of the tensor coefficients

- Analytic approach, typically performed in high-level language, then exported to number-crunching language (with a lot of CSE and other optimizations)
- Numerical approaches require recursion kernels

II Counterterms terms

- Can be treated on equal footing with tree-amplitudes
- requires solution for counterterm parameters provided by external tools (later in this talk)

III Rational terms

- For free in analytic approach.
- In numerical approach typically included as additional Feynman rules.

Ingredients for 1-loop BSM amplitudes

Renormalized 1-loop amplitude:

$$\mathcal{M}_1 = \sum_k \underbrace{c_{k,\mu_1,\mu_2,\dots}}_{\text{I}} \; T_k^{\mu_1,\mu_2,\dots} + \underbrace{\boldsymbol{\mathcal{M}}_{\mathbf{CT}}}_{\text{II}} + \underbrace{\mathcal{M}_{\mathbf{R_2}}}_{\text{III}}$$

I Computation of the tensor coefficients

- Analytic approach, typically performed in high-level language, then exported to number-crunching language (with a lot of CSE and other optimizations)
- Numerical approaches require recursion kernels

II Counterterms terms

- Can be treated on equal footing with tree-amplitudes
- requires solution for counterterm parameters provided by external tools (later in this talk)

III Rational terms

- For free in analytic approach.
- In numerical approach typically included as additional Feynman rules.

Ingredients for 1-loop BSM amplitudes

Renormalized 1-loop amplitude:

$$\mathcal{M}_1 = \sum_k \underbrace{c_{k,\mu_1,\mu_2,\dots}}_{\mathrm{I}} \; T_k^{\mu_1,\mu_2,\dots} + \underbrace{\boldsymbol{\mathcal{M}}_{\mathbf{CT}}}_{\mathrm{II}} + \underbrace{\boldsymbol{\mathcal{M}}_{\mathbf{R_2}}}_{\mathrm{III}}$$

I Computation of the tensor coefficients

- Analytic approach, typically performed in high-level language, then exported to number-crunching language (with a lot of CSE and other optimizations)
- Numerical approaches require recursion kernels

II Counterterms terms

- Can be treated on equal footing with tree-amplitudes
- requires solution for counterterm parameters provided by external tools (later in this talk)

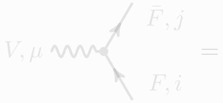
III Rational terms

- For free in analytic approach.
- In numerical approach typically included as additional Feynman rules.

I Computation of recursion kernels

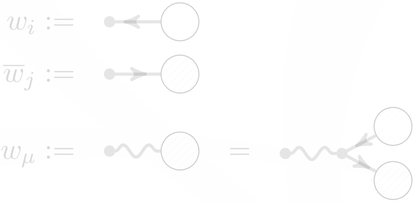
The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

$\mathrm{VFF} = (\gamma \mathrm{e}^+\mathrm{e}^-, \mathrm{W}^+\bar{\nu}_\mathrm{e}\mathrm{e}^-, \ldots)$



$$\gamma^\mu_{jk}\left(c^+w^+ + c^-w^-\right)_{ki}$$

- $c^\pm$ couplings
- $w^\pm = (1 \pm \gamma_5)/2$



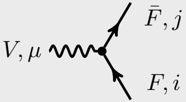
$$= D^V_{\mu,\nu}\, \gamma^\nu_{jk}\left(c^+w^+ + c^-w^-\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but …

I Computation of recursion kernels

The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

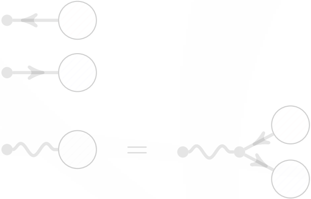
VFF = ($\gamma e^+ e^-$, $W^+ \bar{\nu}_e e^-$, ...)



$$= \gamma^{\mu}_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki}$$

- $c^{\pm}$ couplings
- $w^{\pm} = (1 \pm \gamma_5)/2$

$w_i :=$, $\overline{w}_j :=$, $w_\mu :=$ $=$



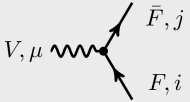
$$= D^V_{\mu,\nu}\, \gamma^{\nu}_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but ...

I Computation of recursion kernels

The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

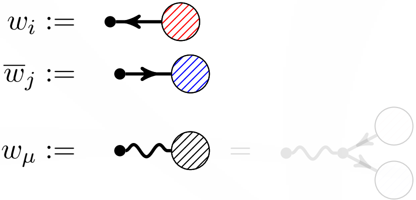
VFF = $(\gamma \mathrm{e}^+\mathrm{e}^-,\ \mathrm{W}^+\bar{\nu}_\mathrm{e}\mathrm{e}^-,\ \ldots)$



$=$

$$\gamma^\mu_{jk}\left(c^+w^+ + c^-w^-\right)_{ki}$$

- $c^\pm$ couplings
- $w^\pm = (1 \pm \gamma_5)/2$



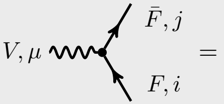
$$= D^V_{\mu,\nu}\,\gamma^\nu_{jk}\left(c^+w^+ + c^-w^-\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but . . .

I Computation of recursion kernels

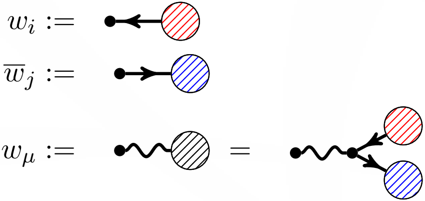
The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

VFF = ($\gamma e^+ e^-$, $W^+ \bar{\nu}_e e^-$, ...)



$$\gamma^{\mu}_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki}$$

- $c^{\pm}$ couplings
- $w^{\pm} = (1 \pm \gamma_5)/2$



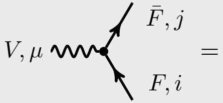
$$= D^{V}_{\mu,\nu}\, \gamma^{\nu}_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but ...

I Computation of recursion kernels

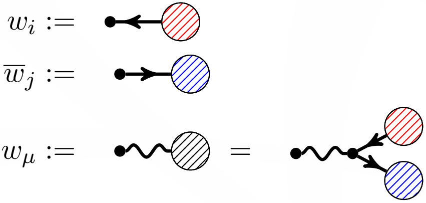
The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

$\mathrm{VFF} = (\gamma \mathrm{e}^{+}\mathrm{e}^{-},\ \mathrm{W}^{+}\bar{\nu}_{\mathrm{e}}\mathrm{e}^{-},\ \ldots)$



$$\gamma^{\mu}_{jk}\left(c^{+}w^{+} + c^{-}w^{-}\right)_{ki}$$

- $c^{\pm}$ couplings
- $w^{\pm} = (1 \pm \gamma_5)/2$



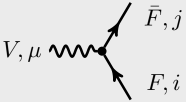
$$= D^{V}_{\mu,\nu}\ \gamma^{\nu}_{jk}\left(c^{+}w^{+} + c^{-}w^{-}\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but …

I Computation of recursion kernels

The OpenLoops and the Recola algorithm require highly optimized recursion kernels which depend on the Lorentz structures appearing in the theory

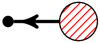
VFF $= (\gamma \mathrm{e}^+\mathrm{e}^-, \mathrm{W}^+\bar{\nu}_\mathrm{e}\mathrm{e}^-, \ldots)$

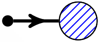


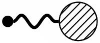
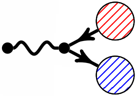
$=$

$$\gamma^\mu_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki}$$

- $c^\pm$ couplings
- $w^\pm = (1 \pm \gamma_5)/2$

$w_i :=$ 

$\overline{w}_j :=$ 

$w_\mu :=$  $=$ 

$$= D^V_{\mu,\nu}\, \gamma^\nu_{jk} \left(c^+ w^+ + c^- w^-\right)_{ki} \times w_i \times \bar{w}_j$$

⇒ In principle simple contraction of wave functions with lorentz structure, but ...

I Computation of recursion kernels

- Recursion kernels are a key ingredient in numerical approaches. Can be a bottleneck especially for EFT/new gauges.
- MadGraph5_aMC@NLO computes the recursion kernels on-the-fly when generating processes with the help of Aloha (arXiv:1108.2041).
- Recola2 model files include all necessary recursion kernels which are generated with the help of Rept1l (arXiv:1705.06053).
- OpenLoops is working on automatizing the generation of recursion kernels.

I Computation of recursion kernels

- Recursion kernels are a key ingredient in numerical approaches. Can be a bottleneck especially for EFT/new gauges.
- MadGraph5_aMC@NLO computes the recursion kernels on-the-fly when generating processes with the help of Aloha (arXiv:1108.2041).
- Recola2 model files include all necessary recursion kernels which are generated with the help of Rept1l (arXiv:1705.06053).
- OpenLoops is working on automatizing the generation of recursion kernels.

I Computation of recursion kernels

- Recursion kernels are a key ingredient in numerical approaches. Can be a bottleneck especially for EFT/new gauges.
- MadGraph5_aMC@NLO computes the recursion kernels on-the-fly when generating processes with the help of Aloha (arXiv:1108.2041).
- Recola2 model files include all necessary recursion kernels which are generated with the help of Rept1l (arXiv:1705.06053).
- OpenLoops is working on automatizing the generation of recursion kernels.

I Computation of recursion kernels

- Recursion kernels are a key ingredient in numerical approaches. Can be a bottleneck especially for EFT/new gauges.

- MadGraph5_aMC@NLO computes the recursion kernels on-the-fly when generating processes with the help of Aloha (arXiv:1108.2041).

- Recola2 model files include all necessary recursion kernels which are generated with the help of Rept1l (arXiv:1705.06053).

- OpenLoops is working on automatizing the generation of recursion kernels.

II/III Renormalization, Counterterms, R2

Currently two automated frameworks exist:

NLOCT: arXiv:1406:3030

Toolchain: FeynRules + FeynArts+ NLOCT

Results in: NLO UFO model file, can be used by MadLoop

Rept1l: arXiv:1705.06053

Toolchain: LO UFO + Recola2 + Rept1l

Results in: NLO Recola2 model file

Differences to NLOCT:

- Derives counterterms from Feynman rules (LO UFO)
- Obtains Vertex/Green's functions via Recola2

II/III Renormalization, Counterterms, R2

Currently two automated frameworks exist:

NLOCT: arXiv:1406:3030

Toolchain: FeynRules + FeynArts+ NLOCT

Results in: NLO UFO model file, can be used by MadLoop

Rept1l: arXiv:1705.06053

Toolchain: LO UFO + Recola2 + Rept1l

Results in: NLO Recola2 model file

Differences to NLOCT:

- Derives counterterms from Feynman rules (LO UFO)
- Obtains Vertex/Green's functions via Recola2

II/III Renormalization, Counterterms, R2

Currently two automated frameworks exist:

NLOCT: arXiv:1406:3030

Toolchain: FeynRules + FeynArts+ NLOCT

Results in: NLO UFO model file, can be used by MadLoop

Rept1l: arXiv:1705.06053

Toolchain: LO UFO + Recola2 + Rept1l

Results in: NLO Recola2 model file

Differences to NLOCT:

- Derives counterterms from Feynman rules (LO UFO)
- Obtains Vertex/Green's functions via Recola2

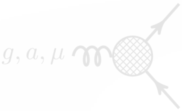
II/III Renormalization, Counterterms, R2

NLO QCD BSM is much alike NLO QCD SM when it comes to renormalization. Standard recipe can be applied:

- Renormalize all colour-charged particles on-shell (or $\overline{\text{MS}}$?) (SM + new ones)

$$\Sigma\left(M^2\right) = 0, \quad \Sigma'\left(M^2\right) = 0$$

- Renormalize α_s in $\overline{\text{MS}}$/MOM scheme.



$$\Bigg|_{\text{P.P.}} + \mathrm{i} g_s T^a_{ij}\gamma_\mu\left(\delta Z_{g_s} + \frac{\delta Z_g^{\overline{\text{MS}}/\text{MOM}}}{2} + \delta Z_q^{\overline{\text{MS}}}\right) \overset{!}{=} 0$$

- Not necessary to renormalize other parameters. ✓

How is the renormalization steered in NLOCT/REPT1L?

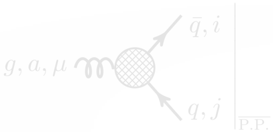
II/III Renormalization, Counterterms, R2

NLO QCD BSM is much alike NLO QCD SM when it comes to renormalization. Standard recipe can be applied:

- Renormalize all colour-charged particles on-shell (or $\overline{\text{MS}}$?) (SM + new ones)

$$\Sigma\left(M^2\right) = 0, \quad \Sigma'\left(M^2\right) = 0$$

- Renormalize α_s in $\overline{\text{MS}}$/MOM scheme.



$$+ \mathrm{i} g_s T^a_{ij} \gamma_\mu \left(\delta Z_{g_s} + \frac{\delta Z_g^{\overline{\text{MS}}/\text{MOM}}}{2} + \delta Z_q^{\overline{\text{MS}}} \right) \stackrel{!}{=} 0$$

- Not necessary to renormalize other parameters. ✓

How is the renormalization steered in NLOCT/REPT1L?

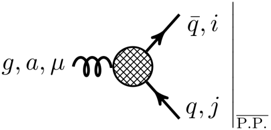
II/III Renormalization, Counterterms, R2

NLO QCD BSM is much alike NLO QCD SM when it comes to renormalization. Standard recipe can be applied:

- Renormalize all colour-charged particles on-shell (or $\overline{\mathrm{MS}}$?) (SM + new ones)

$$\Sigma\left(M^2\right) = 0, \quad \Sigma'\left(M^2\right) = 0$$

- Renormalize α_s in $\overline{\mathrm{MS}}$/MOM scheme.



$$+ \mathrm{i} g_s T^a_{ij} \gamma_\mu \left(\delta Z_{g_s} + \frac{\delta Z_g^{\overline{\mathrm{MS}}/\mathrm{MOM}}}{2} + \delta Z_q^{\overline{\mathrm{MS}}} \right) \overset{!}{=} 0$$

- Not necessary to renormalize other parameters. ✓

How is the renormalization steered in NLOCT/REPT1L?

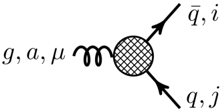
II/III Renormalization, Counterterms, R2

NLO QCD BSM is much alike NLO QCD SM when it comes to renormalization. Standard recipe can be applied:

- Renormalize all colour-charged particles on-shell (or $\overline{\text{MS}}$?) (SM + new ones)

$$\Sigma\left(M^2\right) = 0, \quad \Sigma'\left(M^2\right) = 0$$

- Renormalize α_{s} in $\overline{\text{MS}}$/MOM scheme.



$$\Bigg|_{\text{P.P.}} + \mathrm{i} g_s T^a_{ij} \gamma_\mu \left(\delta Z_{g_s} + \frac{\delta Z_g^{\overline{\text{MS}}/\text{MOM}}}{2} + \delta Z_q^{\overline{\text{MS}}} \right) \overset{!}{=} 0$$

- Not necessary to renormalize other parameters. ✓

How is the renormalization steered in NLOCT/Rept1l?

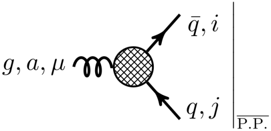
II/III Renormalization, Counterterms, R2

NLO QCD BSM is much alike NLO QCD SM when it comes to renormalization. Standard recipe can be applied:

- Renormalize all colour-charged particles on-shell (or $\overline{\mathrm{MS}}$?) (SM + new ones)

$$\Sigma\left(M^2\right) = 0, \quad \Sigma'\left(M^2\right) = 0$$

- Renormalize α_s in $\overline{\mathrm{MS}}$/MOM scheme.



$$+\mathrm{i}g_s T^a_{ij}\gamma_\mu\left(\delta Z_{g_s} + \frac{\delta Z_g^{\overline{\mathrm{MS}}/\mathrm{MOM}}}{2} + \delta Z_q^{\overline{\mathrm{MS}}}\right) \overset{!}{=} 0$$

- Not necessary to renormalize other parameters. ✓

How is the renormalization steered in NLOCT/REPT1L?

NLOCT

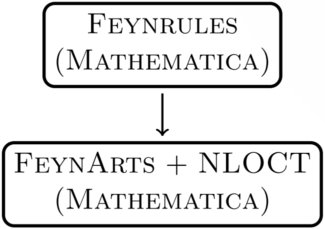
Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians[arXiv:1406:3030]

FEYNRULES
(MATHEMATICA)

```
Quit[]
<<FeynRules`
LoadModel["SM.fr"]
Lren = OnShellRenormalization[LSM],
        QCDOnly->True, FlavorMixing->False];
WriteFeynArtsOutput[Lren, "SMNLO"]
```

NLOCT

Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians[arXiv:1406:3030]


FeynRules (Mathematica)

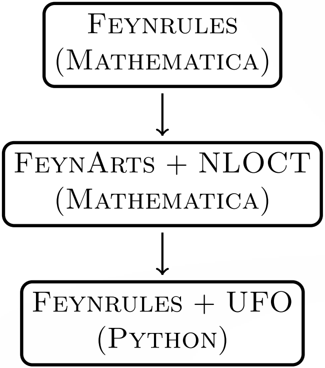
FeynArts + NLOCT (Mathematica)

```
Quit[]
<<FeynRules‘
LoadModel["SM.fr"]
Lren = OnShellRenormalization[LSM],
        QCDOnly->True,FlavorMixing->False];
WriteFeynArtsOutput[Lren,"SMNLO"]
```

```
Quit[]
<<FeynArts‘
<<NLOCT‘
WriteCT["SMNLO/SMNLO",
        "SMNLO/SMNLO",
         QCDOnly -> True]
```

NLOCT

Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians[arXiv:1406:3030]



FeynRules (Mathematica)

↓

FeynArts + NLOCT (Mathematica)

↓

FeynRules + UFO (Python)

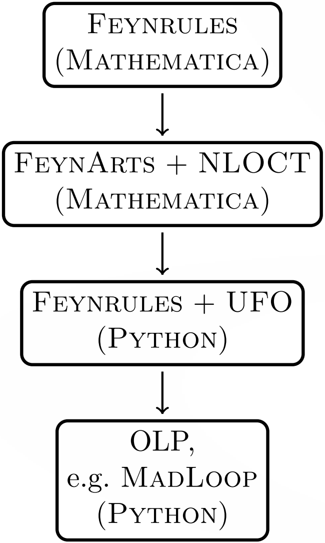
```
Quit[]
<<FeynRules`
LoadModel["SM.fr"]
Lren = OnShellRenormalization[LSM],
        QCDOnly->True,FlavorMixing->False];
WriteFeynArtsOutput[Lren,"SMNLO"]
```

```
Quit[]
<<FeynArts`
<<NLOCT`
WriteCT["SMNLO/SMNLO",
        "SMNLO/SMNLO",
        QCDOnly -> True]
```

```
Quit[]
<<Feynrules`
LoadModel["SM.fr"]
Get["SMNLO.nlo"]
WriteUFO[LSM,
          UVCounterterms -> UV$vertlist,
          R2Counterterms -> R2$vertlist]
```

NLOCT

Automatic evaluation of UV and R2 terms for beyond the Standard Model Lagrangians[arXiv:1406:3030]



FeynRules (Mathematica)

FeynArts + NLOCT (Mathematica)

FeynRules + UFO (Python)

OLP, e.g. MadLoop (Python)

```
Quit[]
<<FeynRules`
LoadModel["SM.fr"]
Lren = OnShellRenormalization[LSM],
        QCDOnly->True,FlavorMixing->False];
WriteFeynArtsOutput[Lren,"SMNLO"]
```

```
Quit[]
<<FeynArts`
<<NLOCT`
WriteCT["SMNLO/SMNLO",
        "SMNLO/SMNLO",
        QCDOnly -> True]
```

```
Quit[]
<<Feynrules`
LoadModel["SM.fr"]
Get["SMNLO.nlo"]
WriteUFO[LSM,
         UVCounterterms -> UV$vertlist,
         R2Counterterms -> R2$vertlist]
```

```
import SM
generate u u~ > u u~ h [QCD]
output
launch
```

Rept1L

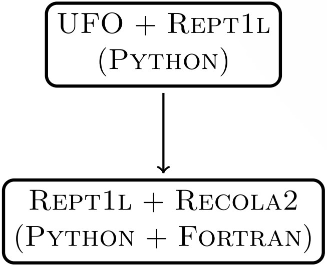
REnormalization in Python at 1-Loop [arXiv:1705.06053]

UFO + Rept1L
(Python)

```
MODEL=PATH/TO/SM_UFO
./run_model -cct $HOME/model/SM_tmp
```

Rept1l

REnormalization in Python at 1-Loop [arXiv:1705.06053]

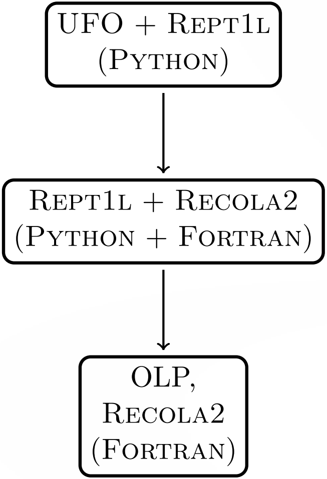

UFO + Rept1l (Python)

Rept1l + Recola2 (Python + Fortran)

```
MODEL=PATH/TO/SM_UFO
./run_model -cct $HOME/model/SM_tmp





./renormalize_qcd -o particles
./renormalize_qcd -o alphaS -nf 4
./renormalize_qcd -o alphaS -nf 5
./renormalize_qcd -o alphaS -nf 6
./run_r2 -np 2 3 4 -j 8

./run_model -cct -cr2 -src
```

REPT1L

REnormalization in PYTHON at 1-Loop [arXiv:1705.06053]



```
MODEL=PATH/TO/SM_UFO
./run_model -cct $HOME/model/SM_tmp
```

```
./renormalize_qcd -o particles
./renormalize_qcd -o alphaS -nf 4
./renormalize_qcd -o alphaS -nf 5
./renormalize_qcd -o alphaS -nf 6
./run_r2 -np 2 3 4 -j 8

./run_model -cct -cr2 -src
```

```python
from pyrecola import *
define_process_rcl(1,'u u -> u u H','NLO')
compute_process(...)
```

Challenges beyond pure QCD

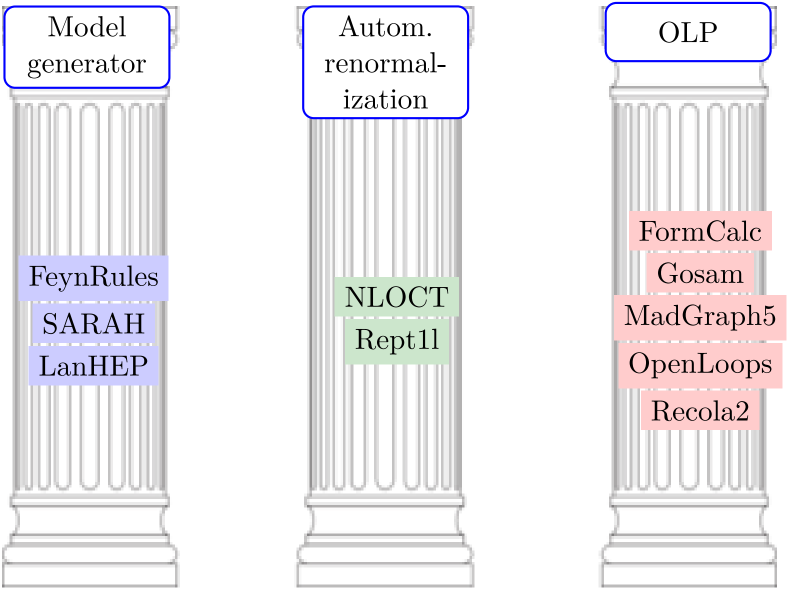
- Renormalization of new parameters requires original ideas, may be showstopper for automation. See renormalization of t_β or mixing angles ... ?
- EW corrections in the presence of unstable particles. ✓
- New Lorentz structures ✓
- Mixed coupling orders
- Tadpoles ✓

Conclusion:

Modern OLP are (getting) ready for BSM theories.
Missing tasks: construction of new one-loop renormalized model files.

- Unified framework? UFO NLO?

The three pillars of 1-Loop BSM automation



Model generator

FeynRules

SARAH

LanHEP

Autom. renormalization

NLOCT

Rept1l

OLP

FormCalc

Gosam

MadGraph5

OpenLoops

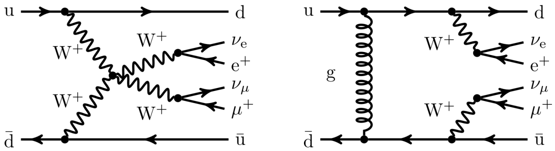
Recola2

Backup slides

Challenges beyond pure QCD

- Mixed coupling orders/interferences

Example: VBS: $pp \to \mu^+\nu_\mu e^+\nu_e$jj (arXiv:1708.00268.)



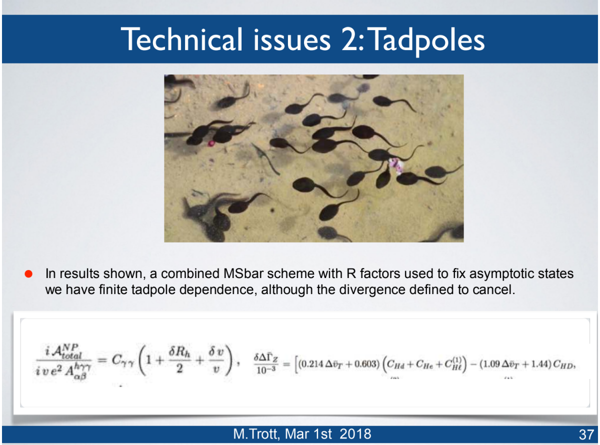
$$\begin{tikzcd}
\textbf{LO} & & \mathcal{O}(\alpha^6) \arrow[dl, "\text{EW}"'] \arrow[dr, "\text{QCD}"] & & \mathcal{O}(\alpha_s\alpha^5) \arrow[dl, "\text{EW}"'] \arrow[dr, "\text{QCD}"] & & \mathcal{O}(\alpha_s^2\alpha^4) \arrow[dl, "\text{EW}"'] \arrow[dr, "\text{QCD}"] & \\
\textbf{NLO} & \mathcal{O}(\alpha^7) & & \mathcal{O}(\alpha_s\alpha^6) & & \mathcal{O}(\alpha_s^2\alpha^5) & & \mathcal{O}(\alpha_s^3\alpha^4)
\end{tikzcd}$$

- Mixing enters already in the renormalization

$$\delta m_t = \delta m_t^{\text{QED}} + \delta m_t^{\text{QCD}}$$

Challenges beyond pure QCD

Tadpoles can be confusing.



Technical issues 2: Tadpoles

- In results shown, a combined MSbar scheme with R factors used to fix asymptotic states we have finite tadpole dependence, although the divergence defined to cancel.

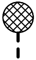
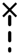
$$\frac{i\,\mathcal{A}^{NP}_{total}}{i\,v\,e^2\,A^{h\gamma\gamma}_{\alpha\beta}} = C_{\gamma\gamma}\left(1 + \frac{\delta R_h}{2} + \frac{\delta v}{v}\right), \quad \frac{\delta\Delta\bar{\Gamma}_Z}{10^{-3}} = \Big[(0.214\,\Delta\bar{v}_T + 0.603)\left(C_{Hd} + C_{He} + C^{(1)}_{H\ell}\right) - (1.09\,\Delta\bar{v}_T + 1.44)\,C_{HD},$$

M.Trott, Mar 1st 2018 37

- ► How can we tell our results are "correct"?

Challenges beyond pure QCD

What is the tadpole renormalization after all?

 +  $= 0 \quad \Leftrightarrow \quad \langle \phi \rangle = 0$ (♠)

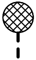
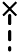
- The prescription ♠ is not unique. At least three different *tadpole counterterm schemes* encountered in the literature. For reference:

  MSSM hep-ph/0205281
  SM arXiv:0709.1075
  FJ Phys.Rev. D23 (1981) 2001-2026

- In the SM, or in general for theories where all renormalization conditions are based on *momentum subtraction* or $\overline{\text{MS}}$ applied to gauge-independent parameters, all tadpole counterterm schemes yield the same S-matrix.

Challenges beyond pure QCD

What is the tadpole renormalization after all?

 $+$  $= 0 \quad \Leftrightarrow \quad \langle \phi \rangle = 0$ (♠)

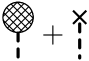
- The prescription ♠ is not unique. At least three different *tadpole counterterm schemes* encountered in the literature. For reference:

  MSSM hep-ph/0205281
  SM arXiv:0709.1075
  FJ Phys.Rev. D23 (1981) 2001-2026

- In the SM, or in general for theories where all renormalization conditions are based on *momentum subtraction* or $\overline{\mathrm{MS}}$ applied to gauge-independent parameters, all tadpole counterterm schemes yield the same S-matrix.

Challenges beyond pure QCD

What is the tadpole renormalization after all?



$$= 0 \quad \Leftrightarrow \quad \langle \phi \rangle = 0 \qquad (\spadesuit)$$

- The prescription ♠ is not unique. At least three different *tadpole counterterm schemes* encountered in the literature. For reference:

  MSSM hep-ph/0205281
  SM arXiv:0709.1075
  FJ Phys.Rev. D23 (1981) 2001-2026

- In the SM, or in general for theories where all renormalization conditions are based on *momentum subtraction* or $\overline{\mathrm{MS}}$ applied to gauge-independent parameters, all tadpole counterterm schemes yield the same S-matrix.

Challenges beyond pure QCD

GSW-theory ($SU(2) \times U(1)$) w/o fermions:

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

Parameter choice:

Before SSB	After SSB
g_1, g_2, λ, μ	M_H, M_W, e, s_w , t

- The FJ tadpole counterterm scheme is based on pure reparametrization invariance of QFT.
- In other tadpole counterterm schemes the renormalized vev enters the definition of bare masses and bare mixing angles, rendering them gauge-dependent.
- This may result in a gauge-dependent S-matrix when employing $\overline{\text{MS}}$ schemes. (arXiv:1607.07352)

Challenges beyond pure QCD

GSW-theory ($SU(2) \times U(1)$) w/o fermions:

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

Parameter choice:

Before SSB	After SSB
g_1, g_2, λ, μ	M_H, M_W, e, s_w , t

- The FJ tadpole counterterm scheme is based on pure reparametrization invariance of QFT.
- In other tadpole counterterm schemes the renormalized vev enters the definition of bare masses and bare mixing angles, rendering them gauge-dependent.
- This may result in a gauge-dependent S-matrix when employing $\overline{\text{MS}}$ schemes. (arXiv:1607.07352)

Challenges beyond pure QCD

GSW-theory $(SU(2) \times U(1))$ w/o fermions:

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

Parameter choice:

Before SSB	After SSB
g_1, g_2, λ, μ	M_H, M_W, e, s_w , t

- The FJ tadpole counterterm scheme is based on pure reparametrization invariance of QFT.
- In other tadpole counterterm schemes the renormalized vev enters the definition of bare masses and bare mixing angles, rendering them gauge-dependent.
- This may result in a gauge-dependent S-matrix when employing $\overline{\text{MS}}$ schemes. (arXiv:1607.07352)

Challenges beyond pure QCD

GSW-theory ($SU(2) \times U(1)$) w/o fermions:

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

Parameter choice:

Before SSB	After SSB
g_1, g_2, λ, μ	M_H, M_W, e, s_w , t

- The FJ tadpole counterterm scheme is based on pure reparametrization invariance of QFT.
- In other tadpole counterterm schemes the renormalized vev enters the definition of bare masses and bare mixing angles, rendering them gauge-dependent.
- This may result in a gauge-dependent S-matrix when employing $\overline{\text{MS}}$ schemes. (arXiv:1607.07352)

How do we compute scattering amplitudes?

$$\mathcal{M}_0 = \sum \mathcal{M}_{0,i}, \quad \mathcal{M}_1 = \sum_k c_{k,\mu_1,\mu_2,\dots} T_k^{\mu_1,\mu_2,\dots}$$

$$\mathcal{M}_{0,i},\ c_k \sim \prod (\bar{u}\Gamma u)$$

Compute $|\mathcal{M}|^2$ using trace techniques?

Compute helicity amplitudes $\mathcal{M}$ directly!

- Spinor-Helicity formalism.
  State of the art in FormCalc and GoSam.
- Numerical helicity methods.
  Used by MadLoop, OpenLoops and Recola

How do we compute scattering amplitudes?

$$\mathcal{M}_0 = \sum \mathcal{M}_{0,i}, \quad \mathcal{M}_1 = \sum_k c_{k,\mu_1,\mu_2,\dots} T_k^{\mu_1,\mu_2,\dots}$$

$$\mathcal{M}_{0,i},\ c_k \sim \prod (\bar{u}\Gamma u)$$

Compute $|\mathcal{M}|^2$ using trace techniques?

Compute helicity amplitudes $\mathcal{M}$ directly!

- Spinor-Helicity formalism.
  State of the art in FormCalc and GoSam.
- Numerical helicity methods.
  Used by MadLoop, OpenLoops and Recola

How do we compute scattering amplitudes?

$$\mathcal{M}_0 = \sum \mathcal{M}_{0,i}, \quad \mathcal{M}_1 = \sum_k c_{k,\mu_1,\mu_2,\ldots} T_k^{\mu_1,\mu_2,\ldots}$$

$$\mathcal{M}_{0,i},\ c_k \sim \prod (\bar{u}\Gamma u)$$

Compute $|\mathcal{M}|^2$ using trace techniques? NO!

Compute helicity amplitudes $\mathcal{M}$ directly!

- Spinor-Helicity formalism.
  State of the art in FormCalc and GoSam.
- Numerical helicity methods.
  Used by MadLoop, OpenLoops and Recola

How do we compute scattering amplitudes?

$$\mathcal{M}_0 = \sum \mathcal{M}_{0,i}, \quad \mathcal{M}_1 = \sum_k c_{k,\mu_1,\mu_2,\ldots} T_k^{\mu_1,\mu_2,\ldots}$$

$$\mathcal{M}_{0,i},\ c_k \sim \prod (\bar{u}\Gamma u)$$

Compute $|\mathcal{M}|^2$ using trace techniques? NO!

Compute helicity amplitudes $\mathcal{M}$ directly!

- Spinor-Helicity formalism.
  State of the art in FormCalc and GoSam.
- Numerical helicity methods.
  Used by MadLoop, OpenLoops and Recola

How do we compute scattering amplitudes?

$$\mathcal{M}_0 = \sum \mathcal{M}_{0,i}, \quad \mathcal{M}_1 = \sum_k c_{k,\mu_1,\mu_2,\ldots} T_k^{\mu_1,\mu_2,\ldots}$$

$$\mathcal{M}_{0,i},\ c_k \sim \prod (\bar{u}\Gamma u)$$

Compute $|\mathcal{M}|^2$ using trace techniques? NO!

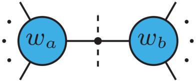
Compute helicity amplitudes $\mathcal{M}$ directly!

- Spinor-Helicity formalism.
  State of the art in FormCalc and GoSam.
- Numerical helicity methods.
  Used by MadLoop, OpenLoops and Recola

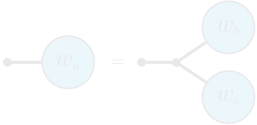
I Computation of tensor coefficients

OpenLoops algorithm, arXiv:1111.5206

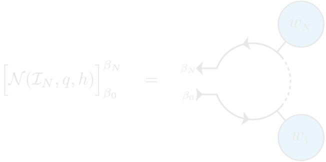
- Reconstruct Tree-diagrams from cut diagrams



- Compute cut diagrams by recursion relations



- Cut loop-diagrams and reconstruct numerator, attaching tree-amplitudes

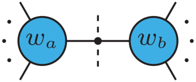

$\Big[\mathcal{N}(\mathcal{I}_N, q, h)\Big]^{\beta_N}_{\beta_0}$
=
β_N
β_0
w_N
w_1

  - A modified OpenLoops algorithm is used by MadLoop.
  - OpenLoops 2 (arXiv:1710.11452) allows for an on-the-fly tensor reduction, allowing to directly extract scalar integral coefficients.

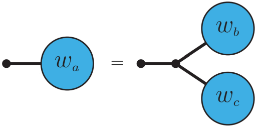
I Computation of tensor coefficients

OpenLoops algorithm, arXiv:1111.5206

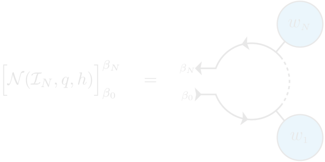
- Reconstruct Tree-diagrams from cut diagrams



- Compute cut diagrams by recursion relations



- Cut loop-diagrams and reconstruct numerator, attaching tree-amplitudes

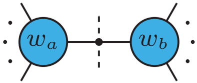

$\left[\mathcal{N}(\mathcal{I}_N, q, h)\right]^{\beta_N}_{\beta_0}$
=
β_N
β_0
w_N
w_1

  - A modified OpenLoops algorithm is used by MadLoop.
  - OpenLoops 2 (arXiv:1710.11452) allows for an on-the-fly tensor reduction, allowing to directly extract scalar integral coefficients.

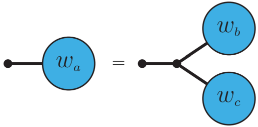
I Computation of tensor coefficients

OpenLoops algorithm, arXiv:1111.5206

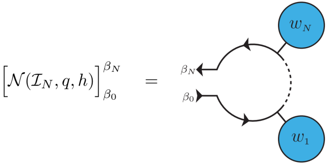
- Reconstruct Tree-diagrams from cut diagrams



- Compute cut diagrams by recursion relations



- Cut loop-diagrams and reconstruct numerator, attaching tree-amplitudes

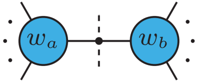

$\left[\mathcal{N}(\mathcal{I}_N, q, h)\right]^{\beta_N}_{\beta_0}$
=
β_N
β_0
w_N
w_1

- A modified OpenLoops algorithm is used by MadLoop.
- OpenLoops 2 (arXiv:1710.11452) allows for an on-the-fly tensor reduction, allowing to directly extract scalar integral coefficients.

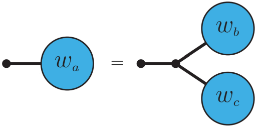
I Computation of tensor coefficients

OpenLoops algorithm, arXiv:1111.5206

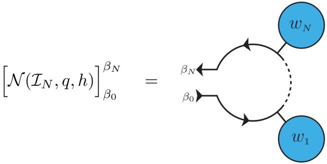
- Reconstruct Tree-diagrams from cut diagrams



- Compute cut diagrams by recursion relations



- Cut loop-diagrams and reconstruct numerator, attaching tree-amplitudes



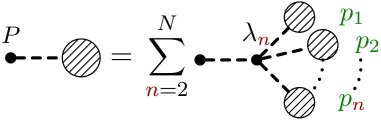
$\left[\mathcal{N}(\mathcal{I}_N, q, h)\right]^{\beta_N}_{\beta_0}$
$=$
β_N
β_0
w_N
w_1

- A modified OpenLoops algorithm is used by MadLoop.
- OpenLoops 2 (arXiv:1710.11452) allows for an on-the-fly tensor reduction, allowing to directly extract scalar integral coefficients.

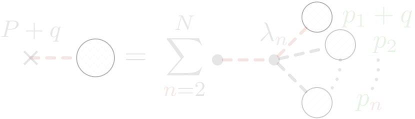
I Computation of tensor coefficients

RECOLA algorithm, arXiv:1211.6316

- Reconstruct Tree-amplitude via Berends-Giele recursion (BGR) [Berends Giele '88]



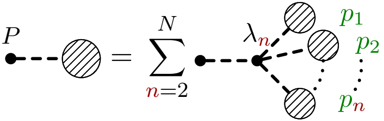
- Reconstruct tensor coefficient via modified BGR [van hameren '09]



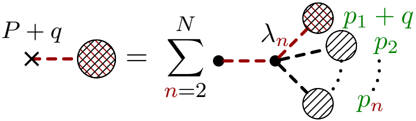
I Computation of tensor coefficients

RECOLA algorithm, arXiv:1211.6316

- Reconstruct Tree-amplitude via Berends-Giele recursion (BGR) [Berends Giele '88]



- Reconstruct tensor coefficient via modified BGR [van hameren '09]


$P+q$
$=$
$\sum_{n=2}^{N}$
λ_n
p_1+q
p_2
p_n