From Lagrangians to events

Benjamin Fuks

LPTHE / Sorbonne Université

12th MCNet Summer School

Prato (Italy), 23 July 2018
Outline

1. Introduction: overview of an LHC collision
2. Implementing models into Monte Carlo event generators
3. From models to hard scattering events
4. Summary
The quest for physics beyond the Standard Model at the LHC is on-going

- How to get hints of new physics?
  - Confront data to the Standard Model expectation in search channels
  - Observe unexplained deviations at a good confidence level

- Theory ingredient 1: predictions for the Standard Model background
- Theory ingredient 2: predictions for the new physics signals
Monte Carlo tools and new physics

Path towards the characterization of (potentially observed) new physics

- Getting information on the nature of an observation
  - Fitting (and interpreting) deviations by some new physics signals
  - Leading order Monte Carlo tools and techniques can do a proper-enough job

- Final words on the nature of any potential new physics
  - Accurate measurements of the model parameters
  - More precise predictions are mandatory
Monte Carlo tools and new physics

Path towards the characterization of (potentially observed) new physics

✦ Getting information on the nature of an observation
  ★ Fitting (and interpreting) deviations by some new physics signals
  ★ Leading order Monte Carlo tools and techniques can do a proper-enough job

✦ Final words on the nature of any potential new physics
  ★ Accurate measurements of the model parameters
  ★ More precise predictions are mandatory

Monte Carlo tools play a key role at each step of this new physics story

✦ Prospective collider studies (a priori preparation)
  ★ Study of various new physics signal

✦ Reinterpretation of existing results (a posteriori reactions to announcements)
  ★ LHC recasting in new theoretical context

New physics simulations for the LHC are crucial
Deciphering a proton-proton collision

[ASHERPA artist]

Proton

Proton

From Lagrangians to events
Deciphering a proton-proton collision

- Hard process
  - Depends on the physics model (SM, BSM, ...)
  - Perturbative QCD
Deciphering a proton-proton collision

- **Hard process**
  - Depends on the physics model (SM, BSM,...)
  - Perturbative QCD

- **Parton showering**
  - Universal (QCD)
Deciphering a proton-proton collision

- **Hard process**
  - Depends on the physics model (SM, BSM, …)
  - Perturbative QCD

- **Parton showering**
  - Universal (QCD)

- **Hadronization**
  - Models-based, universal

- **Underlying event**
  - Model-based, non universal
Deciphering a proton-proton collision

- **Hard process**
  - Depends on the physics model (SM, BSM, ...)
  - Perturbative QCD

- **Parton showering**
  - Universal (QCD)

- **Hadronization**
  - Models-based, universal

- **Underlying event**
  - Model-based, non-universal

- **Detector simulation**

[Image: Proton-Proton Collision Diagram]
We have a multi-scale problem where each ingredient factorizes:

- Hard scattering process: high-scale, potential new physics
- Parton showering and hadronization: driven by QCD
- Detector simulation: driven by the detector choice

We have tools and methods to handle each step.
Monte Carlo simulations of proton collisions

We have a multi-scale problem where each ingredient factorizes

- Hard scattering process: high-scale, potential new physics
- Parton showering and hadronization: driven by QCD
- Detector simulation: driven by the detector choice

The need for better simulation tools has spurred a very intense activity

- Matrix-element generation (CALCHEP, HERWIG, MG5_AMC, SHERPA, WHIZARD, etc.)
- Higher-order computations (MC@NLO, POWHEG, NNLO)
- Parton showering and hadronization (PYTHIA, HERWIG, SHERPA)
- Matrix element - parton showering matching
- Merging techniques (MLM, CKKW, FxFx, UNLOPS, etc.)
SM and BSM simulations: the status

✦ Standard Model simulations
  ✤ All processes relevant for the LHC can be simulated with a very good precision
  ✤ The precision will improve in the next few years (e.g. electroweak corrections)

SM simulations under control
What about new physics?
SM and BSM simulations: the status

✦ Standard Model simulations
  ✤ All processes relevant for the LHC can be simulated with a very good precision
  ✤ The precision will improve in the next few years (e.g. electroweak corrections)

**SM simulations under control**
What about new physics?

✦ The challenge with respect to new physics is different
  ✤ Theoretically, we are still in the dark
    ★ No sign of new physics, all measurements are Standard-Model-like
  ✤ There is not any leading new physics candidate theory
    ★ Plethora of models to implement in the tools
SM and BSM simulations: the status

✦ **Standard Model simulations**
  ✤ All processes relevant for the LHC can be simulated with a very good precision
  ✤ The precision will improve in the next few years (e.g. electroweak corrections)
  
  **SM simulations under control**
  **What about new physics?**

✦ **The challenge with respect to new physics is different**
  ✤ Theoretically, we are still in the dark
    ★ No sign of new physics, all measurements are Standard-Model-like
  ✤ There is not any leading new physics candidate theory
    ★ Plethora of models to implement in the tools

✦ **New physics is a standard in many tools today**
  ✤ Result of 20 years of developments
  ✤ NLO and loop-induced processes can now be simulated automatically
  ✤ BSM precision simulations are standard (in particular in MG5_aMC@NLO)
  
  **What are the ingredients of this success?**
The connection of a physics models to event analysis is streamlined

- This relies on a **framework**:
  - Any new physics model can be **implemented**
  - Any new physics model can be **tested** against data
  - Easy to **validate**, to **maintain**
  - Easily **integrable in a software chain**

**Idea Lagrangian** \(\rightarrow\) **Simulated collisions** \(\rightarrow\) **Event analysis**

**Chain of tools**

**Other tools**
A comprehensive approach to MC simulations

[Christensen, de Aquino, Degrande, Duhr, BF, Herquet, Maltoni & Schumann (EPJC'11)]

✨ Tools connecting an idea to simulated collisions

- Idea / Lagrangian
- Model building
- Feynman diagram and amplitude generation
- Monte Carlo integration
- Event generation
- QCD environment
- Parton showering
- Hadronization
- Underlying event
- Detector simulation
- Simulation of the detector response
- Object reconstruction
- Event analysis
  - Signal/background analysis
  - LHC recasting

FEYNRULES / SARAH / LANHEP/ UFO

Matrix Element Generator

Parton showers

Hadronization

Detector

Simulated collisions

Analysis

From Lagrangians to events
A comprehensive approach to MC simulations

[Christensen, de Aquino, Degrande, Duhr, BF, Herquet, Maltoni & Schumann (EPJC`11)]

Tools connecting an idea to simulated collisions

- **Model building**
  - FEYNRULES, SARAH, LANHEP
  - UFO

- **Matrix element generation**
  - CALCHEP, HERWIG++, MG5_AMC, SHERPA, WHIZARD, …

- **QCD environment**
  - HERWIG, PYTHIA, SHERPA

- **Detector simulation**
  - DELPHES / PGS
  - RIVET / MADANALYSIS 5

- **Event analysis**
  - RIVET / MADANALYSIS 5
A comprehensive approach to MC simulations

[Christensen, de Aquino, Degrande, Duhr, BF, Herquet, Maltoni & Schumann (EPJC`11)]

✨ Tools connecting an idea to simulated collisions

- **Model building**
  - FEYNRULES, SARAH, LANHEP
  - UFO

- **Matrix element generation**
  - CALCHEP, HERWIG++, MG5_AMC, SHERPA, WHIZARD, …

- **QCD environment**
  - HERWIG, PYTHIA, SHERPA

- **Detector simulation**
  - DELPHES / PGS
  - RIVET / MADANALYSIS 5

- **Event analysis**
  - RIVET / MADANALYSIS 5

From Lagrangians to events
Outline

1. Introduction: overview of an LHC collision

2. Implementing models into Monte Carlo event generators

3. From models to hard scattering events

4. Summary
New physics simulations: the ‘how-to’

✦ How to implement a new physics model in a Monte Carlo program?

★ Model definition: particles, parameters & vertices (≡ Lagrangian)

★ To be translated in a programming language, following some conventions, etc.

★ Tedious, time-consuming, error prone

★ Iterations for all considered tools and models

★ Beware of the restrictions of each tool (Lorentz structures, color structures)
How to implement a new physics model in a Monte Carlo program?

- Model definition: particles, parameters & vertices (≡ Lagrangian)
- To be translated in a programming language, following some conventions, etc.
- Tedious, time-consuming, error prone
- Iterations for all considered tools and models
- Beware of the restrictions of each tool (Lorentz structures, color structures)

- Highly redundant (each tool, each model)
- No-brainer tasks (from Feynman rules to codes)
How to implement a new physics model in a Monte Carlo program?

- Model definition: particles, parameters & vertices (≡ Lagrangian)
- To be translated in a programming language, following some conventions, etc.
- Tedious, time-consuming, error prone
- Iterations for all considered tools and models
- Beware of the restrictions of each tool (Lorentz structures, color structures)

Highly redundant (each tool, each model)

No-brainer tasks (from Feynman rules to codes)
A Monte Carlo tool framework for new physics

✧ Specifications

✧ Inputs / Outputs
  ★ A physics object: the Lagrangian (unique and non ambiguous, no MC dependence)
  ★ Flexible (a change in the model = a change in the Lagrangian)
  ★ Automatic derivation of the Feynman rules and generate MC model files

✧ Validation
  ★ Automatic and systematical

✧ Distribution
  ★ Public, open source, transparent
  ★ No private tools

[ Christensen, de Aquino, Degrande, Duhr, BF, Herquet, Maltoni & Schumann (EPJC'11) ]
The **FeynRules** platform (since 2009)

- Automatic linking of Lagrangians to files in a given programming language
  - With all model particles, interactions, etc.

- Working environment: **Mathematica**
  - Flexibility, symbolic manipulations, easy implementation of new methods, etc.
  - Shipped with many computation platforms (superspace, spectrum, decays, NLO, etc.)

- Interfaced to many Monte Carlo tools
  - Dedicated interfaces (**CALCHEP**, **FeynArts**, and more in the past)
  - Interfaced to more tools via the UFO (**HERWIG**, **MG5_AMC**, **Sherpa**, **Whizard**, …)

- Very few limitations on models
  - Higher-dimensional operators
  - Spins (up to 2); color structures (1, 3, 6, 8)

[Christensen & Duhr (CPC '09)]
[Alloul, Christensen, Degrande, Duhr & BF (CPC'14)]
Existing programs

✦ The FEYNRULES platform (since 2009)
  ✷ Automatic linking of Lagrangians to files in a given programming language
    ★ With all model particles, interactions, etc.
  ✷ Working environment: MATHEMATICA
    ★ Flexibility, symbolic manipulations, easy implementation of new methods, etc.
    ★ Shipped with many computation platforms (superspace, spectrum, decays, NLO, etc.)
  ✷ Interfaced to many Monte Carlo tools
    ★ Dedicated interfaces (CALCHEP, FEYNARTS, and more in the past)
    ★ Interfaced to more tools via the UFO (HERWIG, MG5_AMC, SHERPA, WHIZARD, …)
  ✷ Very few limitations on models
    ★ Higher-dimensional operators
    ★ Spins (up to 2); color structures (1, 3, 6, 8)

[ Christensen & Duhr (CPC '09) ]
[ Alloul, Christensen, Degrande, Duhr & BF (CPC'14) ]

✦ LANHEP (since 1997)
  ✷ Automatic linking of Lagrangians to files in a given programming language
  ✷ Working environment: C
  ✷ Initially restricted to CALCHEP/COMPHEP
  ✷ Now: FEYNARTS and UFO outputs

[ Semenov (NIMA'97; CPC'98; CPC'09; CPC'16) ]
### Existing programs

#### The **FEYNRULES** platform (since 2009)
- Automatic linking of Lagrangians to files in a given programming language
  - With all model particles, interactions, etc.
- Working environment: **MATHEMATICA**
  - Flexibility, symbolic manipulations, easy implementation of new methods, etc.
  - Shipped with many computation platforms (superspace, spectrum, decays, NLO, etc.)
- Interfaced to many Monte Carlo tools
  - Dedicated interfaces (**CALCHEP**, **FEYNARTS**, and more in the past)
  - Interfaced to more tools via the UFO (**HERWIG**, **MG5_AMC**, **SHERPA**, **WHIZARD**, …)
- Very few limitations on models
  - Higher-dimensional operators
  - Spins (up to 2); color structures (1, 3, 6, 8)

#### **LANHEP** (since 1997)
- Automatic linking of Lagrangians to files in a given programming language
- Working environment: **C**
- Initially restricted to **CALCHEP/COMPHEP**
- Now: **FEYNARTS** and UFO outputs

#### The **SARAH** package (since 2013)
- Automatic linking of Lagrangians to files in a given programming language
- Working environment: **MATHEMATICA**
- Spectrum generator, indirect constraints
- Interfaced to many tools (**CALCHEP**, **COMPHEP**, **FEYNARTS**, **UFO**, **WHIZARD**)

---

[Semionov (NIMA’97; CPC’98; CPC’09; CPC’16)]

[Christensen & Duhr (CPC ’09)]

[Alloul, Christensen, Degrande, Duhr & BF (CPC’14)]

[Staub (CPC’13; CPC’14)]
The implementation of any new physics theory in a MC tool is straightforward.

Many interfaces to specific tools:
- Removal of non compliant vertices
- Translation to a specific format/language
- To be maintained

Not efficient
Why do we need UFOs?

- **Color structures: not supported in full generality by MC generators**
  - The treatment of the color information is hard-coded
  - The interfaces to a specific tool discard all non-supported vertices
  - Representations usually handled: 1, 3, 8 (sometimes limited), sometimes 6
Why do we need UFOs?

✧ Color structures: not supported in full generality by MC generators
  ✧ The treatment of the color information is hard-coded
  ✧ The interfaces to a specific tool discard all non-supported vertices
  ✧ **Representations usually handled:** 1, 3, 8 (sometimes limited), sometimes 6

✧ Lorentz and spin structures not supported in full generality by MC programs
  ✧ The treatment of the Lorentz structures of the different vertices is hard-coded
  ✧ The possible spins for the particles are restricted
  ✧ The interfaces discard all non-supported vertices
  ✧ **Spin representations usually handled:** 0, 1/2, 1; sometimes 3/2, 2
  ✧ **Lorentz structures usually handled:** usually anything MSSM-like
Why do we need UFOs?

✧ Color structures: not supported in full generality by MC generators
  ✧ The treatment of the color information is hard-coded
  ✧ The interfaces to a specific tool discard all non-supported vertices
  ✧ Representations usually handled: 1, 3, 8 (sometimes limited), sometimes 6

✧ Lorentz and spin structures not supported in full generality by MC programs
  ✧ The treatment of the Lorentz structures of the different vertices is hard-coded
  ✧ The possible spins for the particles are restricted
  ✧ The interfaces discard all non-supported vertices
  ✧ Spin representations usually handled: 0, 1/2, 1; sometimes 3/2, 2
  ✧ Lorentz structures usually handled: usually anything MSSM-like

Each interface dedicated to a given tool is specific
  ★ Removal of vertices not compliant with the tool
  ★ Translation to a specific format and programming language
  ⇒ not efficient
  ⇒ better: one translation and the tools parse it
A step further: the Universal FEYNRULES Output

✦ The UFO in a nutshell

✧ UFO $\equiv$ Universal FEYNRULES output
  ★ Universal as not tied to any specific Monte Carlo program
  ✧ Consists of a PYTHON module to be linked to any code
  ✧ This module contains all the model information
    ★ Allows the models to contain generic color and Lorentz structures
  ✧ Can be employed for next-to-leading order calculations

★ Universal as not tied to any specific Monte Carlo program

✧ The UFO in a nutshell

✦ The UFO is now a standard used by many programs (on top of FEYNRULES)

ALOHA  GOSAM  HERWIG ++  MadAnalysis 5  SHERPA

MADGRAPH5_aMC@NLO  Whizard  LanHEP  Sarah
The UFO in practice

✦ The UFO is a set of PYTHON files
  ✷ Factorization of the information: particles, interactions, propagation, parameters, NLO, etc.

✦ Example

```python
[fuks@Benjamins-MacBook-Pro-3 ~/Work/tools/FeynRules/trunk/models/SUSYQCD_UFO]$ ls
CT_couplings.py  SUSYQCD_UFO.log  couplings.py  object_library.py
CT_parameters.py  __init__.py    function_library.py  parameters.py
CT_vertices.py   coupling_orders.py  lorentz.py  particles.py
[fuks@Benjamins-MacBook-Pro-3 ~/Work/tools/FeynRules/trunk/models/SUSYQCD_UFO]$ ls
```

- **Propagators**
- **Parameters**
- **Particles**
- **Interactions**
- **NLO**
Particles

- Particles are stored in the `particles.py` file
  - Instances of the particle class
  - Attributes: particle spin, color representation, mass, width, PDG code, etc.
  - Antiparticles automatically derived

```python
G = Particle(pdg_code = 21, name = 'G', antiname = 'G', spin = 3, color = 8, mass = Param.ZERO, width = Param.ZERO, texname = 'G', antitexname = 'G', charge = 0)
go = Particle(pdg_code = 1000021, name = 'go', antiname = 'go', spin = 2, color = 8, mass = Param.Mgo, width = Param.Wgo, texname = 'go', antitexname = 'go', charge = 0)

sq1 = Particle(pdg_code = 1000006, name = 'sq1', antiname = 'sq1~', spin = 1, color = 3, mass = Param.Msq1, width = Param.Wsq1, texname = 'sq1', antitexname = 'sq1~', charge = 0)
q = Particle(pdg_code = 6, name = 'q', antiname = 'q~', spin = 2, color = 3, mass = Param.Mq, width = Param.Wq, texname = 'q', antitexname = 'q~', charge = 0)

q_tilde__ = q.anti()
sq1_tilde__ = sq1.anti()
sq2 = Particle(pdg_code = 2000006, name = 'sq2', antiname = 'sq2~', spin = 1, color = 3, mass = Param.Msq2, width = Param.Wsq2, texname = 'sq2', antitexname = 'sq2~', charge = 0)
sq2_tilde__ = sq2.anti()
```
Parameters

Parameters are stored in the parameters.py file

- Instances of the parameter class
- External parameters are organized following a Les Houches-like structure (blocks and counters)
- PYTHON-compliant formula for the internal parameters

```python
aS = Parameter(name = 'aS',
    nature = 'external',
    type = 'real',
    value = 0.1184,
    texname = '\alpha_s',
    lhablock = 'SMINPUTS',
    lhacode = [ 3 ])

G = Parameter(name = 'G',
    nature = 'internal',
    type = 'real',
    value = 2*cmath.sqrt(aS)*cmath.sqrt(cmath.pi),
    texname = 'G')

Mgo = Parameter(name = 'Mgo',
    nature = 'external',
    type = 'real',
    value = 500,
    texname = '\text{Mgo}',
    lhablock = 'MASS',
    lhacode = [ 1000021 ])

Wq = Parameter(name = 'Wq',
    nature = 'external',
    type = 'real',
    value = 1.50833649,
    texname = '\text{Wq}',
    lhablock = 'DECAY',
    lhacode = [ 6 ])
```
Vertices decomposed in a spin x color basis (coupling strengths \(\equiv\) coordinates)

- Example: the quartic gluon vertex can be written as

\[

\begin{align*}
ig_s^2 f^{a_1a_2b} f^{ba_3a_4} & \left( \eta^{\mu_1\mu_4} \eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3} \eta^{\mu_2\mu_4} \right) \\
+ ig_s^2 f^{a_1a_3b} f^{ba_2a_4} & \left( \eta^{\mu_1\mu_4} \eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2} \eta^{\mu_3\mu_4} \right) \\
+ ig_s^2 f^{a_1a_4b} f^{ba_2a_3} & \left( \eta^{\mu_1\mu_3} \eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2} \eta^{\mu_3\mu_4} \right)
\end{align*}
\]

\[\Rightarrow \begin{pmatrix} f^{a_1a_2b} f^{ba_3a_4} & f^{a_1a_3b} f^{ba_2a_4} & f^{a_1a_4b} f^{ba_2a_3} \end{pmatrix} \times \begin{pmatrix} ig_s^2 & 0 & 0 \\
0 & ig_s^2 & 0 \\
0 & 0 & ig_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4} \eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3} \eta^{\mu_2\mu_4} \\
\eta^{\mu_1\mu_4} \eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2} \eta^{\mu_3\mu_4} \\
\eta^{\mu_1\mu_3} \eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2} \eta^{\mu_3\mu_4} \end{pmatrix}
\]

- ★ 3 elements for the color basis
- ★ 3 elements for the spin (Lorentz structure) basis
- ★ 9 coordinates (6 are zero)

- Several files are used for the storage of the information
Example: the quartic gluon vertex

★ General information in vertex.py

```python
V_2 = Vertex(name = 'V_2',
              color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
              lorentz = [ L.VVVV1, L.VVVV2, L.VVVV3 ],
              couplings = {(1,1):C.GC_4,(0,0):C.GC_4,(2,2):C.GC_4})
```

★ lorentz ≡ spin basis
   (in lorentz.py; common to all vertices)

★ color ≡ color basis

★ couplings ≡ coordinates
   (in couplings.py; common to all vertices)
Example: the quartic gluon vertex

**General information in vertex.py**

```python
V_2 = Vertex(name = 'V_2',
              color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1)' ],
              lorentz = [ L.VVVV1, L.VVVV2, L.VVVV3 ],
              couplings = {(1,1):C.GC_4,(0,0):C.GC_4,(2,2):C.GC_4})
```

- **lorentz** ≡ spin basis
  (in lorentz.py; common to all vertices)
- **color** ≡ color basis
- **couplings** ≡ coordinates
  (in couplings.py; common to all vertices)

\[
\left( f_{a_1a_2b} f_{ba_3a_4}, f_{a_1a_3b} f_{ba_2a_4}, f_{a_1a_4b} f_{ba_2a_3} \right) \\
\times \\
\left( \begin{array}{ccc}
ig_s^2 & 0 & 0 \\
0 & ig_s^2 & 0 \\
0 & 0 & ig_s^2 \\
\end{array} \right) \\
\left( \begin{array}{c}
\eta_{\mu_1\mu_4}\eta_{\mu_2\mu_3} - \eta_{\mu_1\mu_3}\eta_{\mu_2\mu_4} \\
\eta_{\mu_1\mu_4}\eta_{\mu_2\mu_3} - \eta_{\mu_1\mu_2}\eta_{\mu_3\mu_4} \\
\eta_{\mu_1\mu_3}\eta_{\mu_2\mu_4} - \eta_{\mu_1\mu_2}\eta_{\mu_3\mu_4} \\
\end{array} \right)
\]

**Lorentz structures: straightforward implementations in lorentz.py**

```python
VVVV1 = Lorentz(name = 'VVVV1',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) - Metric(1,3)*Metric(2,4)')
```

From Lagrangians to events
Example: the quartic gluon vertex

General information in `vertex.py`

```python
V_2 = Vertex(name = 'V_2',
             color = [ 'f(-1,1,2)*f(3,4,-1)', 'f(-1,1,3)*f(2,4,-1)', 'f(-1,1,4)*f(2,3,-1) ],
             lorentz = [ L.VVVV1, L.VVVV2, L.VVVV3 ],
             couplings = {(1,1):C.GC_4,(0,0):C.GC_4,(2,2):C.GC_4})
```

- **lorentz** ≡ spin basis
  (in lorentz.py; common to all vertices)
- **color** ≡ color basis
- **couplings** ≡ coordinates
  (in couplings.py; common to all vertices)

$$
\begin{align*}
(f^{a_1 a_2 b} f^{b a_3 a_4}, & f^{a_1 a_3 b} f^{b a_2 a_4}, f^{a_1 a_4 b} f^{b a_2 a_3}) \\
\times & \begin{pmatrix}
    i g_s^2 & 0 & 0 \\
    0 & i g_s^2 & 0 \\
    0 & 0 & i g_s^2
\end{pmatrix}
\begin{pmatrix}
    \eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4} \\
    \eta^{\mu_1 \mu_4} \eta^{\mu_2 \mu_3} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4} \\
    \eta^{\mu_1 \mu_3} \eta^{\mu_2 \mu_4} - \eta^{\mu_1 \mu_2} \eta^{\mu_3 \mu_4}
\end{pmatrix}
\end{align*}
$$

Lorentz structures: straightforward implementations in `lorentz.py`

```python
VVVV1 = Lorentz(name = 'VVVV1',
                spins = [ 3, 3, 3, 3 ],
                structure = 'Metric(1,4)*Metric(2,3) - Metric(1,3)*Metric(2,4)')
```

Couplings: straightforward implementations in `couplings.py`

```python
GC_4 = Coupling(name = 'GC_4',
                 value = 'complex(0,1)*G**2',
                 order = {'QCD':2})
```

Coupling orders: for selecting diagrams
Recap’ on NLO calculations

Contributions to an NLO result in QCD

Three ingredients: the Born, virtual loop and real emission contributions

\[ \sigma_{NLO} = \int d^4\Phi_n B + \int d^4\Phi_n \int_{\text{loop}} d^d\ell V + \int d^4\Phi_{n+1} R \]

- **Born**
- **Virtuals**: one extra power of \( \alpha_s \) and divergent
- **Reals**: one extra power of \( \alpha_s \) and divergent

Extra information is needed
Virtual contributions

✧ **Loop diagram calculations**

✦ Calculations to be done in $d=4-2\varepsilon$ dimensions
  ★ Divergences made explicit ($1/\varepsilon^2$, $1/\varepsilon$)

✦ Rewriting loop integrals with **scalar integrals**

\[
\int d^d\ell \frac{N(\ell)}{D_0 D_1 \cdots D_{m-1}} = \sum a_i \int d^d\ell \frac{1}{D_{i_0} D_{i_1} \cdots}
\]

★ Involves integrals with **up to four denominators**
  ➢ The decomposition basis is finite
  ➢ Can be computed once and for all
★ The reduction is the process-dependent part

$m$-point diagram with $n$ external momenta

From Lagrangians to events
The loop momentum lives in a $d$-dimensional space

- Reduction to be done in $d$ dimensions

\[ \int \frac{d^d \ell}{D_0 D_1 \cdots D_{m-1}} \frac{N(\ell, \tilde{\ell})}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}} \quad \text{with} \quad \tilde{\ell} = \ell + \tilde{\ell} \]

- Numerical methods works in 4 dimensions: need to be compensated!
The rational terms ($R_1$ and $R_2$)

- The loop momentum lives in a $d$-dimensional space
  - Reduction to be done in $d$ dimensions

\[ \int d^d \ell \frac{N(\ell, \tilde{\ell})}{D_0 D_1 \cdots D_{m-1}} \quad \text{with} \quad \tilde{\ell} = \ell + \tilde{\ell} \]

- Numerical methods works in 4 dimensions: need to be compensated!

- The $R_1$ terms originates from the denominators
  - Connected to the internal propagators

- The $R_2$ terms originates from the numerator
  - Seen as extra diagrams with special Feynman rules (model dependent)
  - Connected to the ultraviolet structure of the integrals
    - Like the ultraviolet counterterms
The rational terms ($R_1$ and $R_2$)

- The loop momentum lives in a $d$-dimensional space
  - Reduction to be done in $d$ dimensions
    \[
    \int d^d \ell \frac{N(\ell, \tilde{\ell})}{D_0 D_1 \cdots D_{m-1}} \quad \text{with} \quad \tilde{\ell} = \ell + \tilde{\ell}
    \]
  - Numerical methods works in 4 dimensions: need to be compensated!

- The $R_1$ terms originates from the denominators
  - Connected to the internal propagators

- The $R_2$ terms originates from the numerator
  - Seen as extra diagrams with special Feynman rules (model dependent)
  - Connected to the ultraviolet structure of the integrals
    - Like the ultraviolet counterterms

UFO @ NLO
**R\textsubscript{1} terms**

✦ The \( R_{1} \) terms originates from the denominators

\[
\frac{1}{D} = \frac{1}{D} \left( 1 - \frac{\vec{\ell}^2}{D} \right)
\]

✦ These extra pieces can be calculated **generically** (3 integrals in total)

\[
\int d^d\ell \frac{\vec{\ell}^2}{D_i D_j} = -\frac{i\pi^2}{2} \left[ m_i^2 + m_j^2 - \frac{p_i - p_j}{2} \right] + \mathcal{O}(\varepsilon)
\]

\[
\int d^d\ell \frac{\vec{\ell}^2}{D_i D_j D_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon)
\]

\[
\int d^d\ell \frac{\vec{\ell}^2}{D_i D_j D_k D_l} = -\frac{i\pi^2}{6} + \mathcal{O}(\varepsilon)
\]

✦ The denominator structure is already known at the reduction time

✦ The \( R_{1} \) coefficients are extracted during the reduction
**The \( R_2 \) terms originates from the numerator**

\[
\bar{N}(\ell) = N(\ell) + \tilde{N}(\tilde{\ell}, \ell, \varepsilon) \\
\begin{array}{c}
\downarrow \text{D-dim} \quad \downarrow \text{4-dim} \quad \downarrow \text{(-2\varepsilon)-dim}
\end{array}
\Rightarrow \quad R_2 \equiv \lim_{\varepsilon \to 0} \frac{1}{(2\pi)^4} \int d^{d-\varepsilon} \ell \frac{\bar{N}(\tilde{\ell}, \ell, \varepsilon)}{D_0 D_1 \cdots D_{m-1}}
\]

✦ Practically, we isolate the epsilon part

✦ There is only a finite set of loops for which it does not vanish
R₂ terms

✦ The R₂ terms originates from the numerator

\[
\bar{N}(\ell) = N(\ell) + \tilde{N}(\ell, \ell, \varepsilon) \quad \Rightarrow \quad R_2 \equiv \lim_{\varepsilon \to 0} \frac{1}{(2\pi)^4} \int d^d \ell \frac{\tilde{N}(\ell, \ell, \varepsilon)}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}}
\]

✦ Practically, we isolate the epsilon part

✦ There is only a finite set of loops for which it does not vanish

✧ They can be re-expressed in terms of R₂ Feynman rules

\[
\propto \int d^d \ell \frac{\tilde{\ell}^2}{D_i D_j D_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon) \quad \Rightarrow \quad -\frac{i}{2\pi} \alpha e^\gamma \mu
\]
The $R_2$ are process dependent and model-dependent (like Feynman rules)

- In a renormalizable theory, there is a finite number of them
- They can be derived from the sole knowledge of the bare Lagrangian

[Ossala, Papadopoulos, Pittau (JHEP'08)]
The $R^2$ are process dependent and model-dependent (like Feynman rules)

- In a renormalizable theory, there is a finite number of them
- They can be derived from the sole knowledge of the bare Lagrangian

[Ossala, Papadopoulos, Pittau (JHEP'08)]

The $R^2$ calculation can be automated and performed once and for all

- Development of the NLOCT package (extension of FEYNRULES)
- Computation, for any model, of all $R^2$ and UV counterterms
  - In the on-shell and MSbar schemes
- Inclusion of the output in the UFO

[Degrande (CPC'15)]
Automated NLO simulations with MG5_aMC

The implementation of any new physics theory in a MC tool is straightforward.

**Idea / Lagrangian**

FeynRules + NLOCT ➔ UFO @ NLO

MG5_aMC ➔ Parton showers ➔ Hadronization ➔ Detector ➔ Simulated collisions ➔ Analysis

Same chain with NLO-QCD corrections
Outline

1. Introduction: overview of an LHC collision
2. Implementing models into Monte Carlo event generators
3. From models to hard scattering events
4. Summary
Back to the simulation chain

Tools connecting an idea to simulated collisions

- Idea / Lagrangian
- FeynRules / SARAH / LanHEP / UFO
- Matrix Element Generator
- Parton showers
- Hadronization
- Detector
- Simulated collisions
- Analysis

Hard scattering process
- Feynman diagram and amplitude generation
- Monte Carlo integration
- Event generation

Talk from S. Platzer
Predictions at the LHC (using QCD)

✴ Distribution of an observable $\omega$: the QCD factorization theorem

$$\frac{d\sigma}{d\omega} = \sum_{a,b} \int dx_a dx_b d\Phi_n f_{a/p_1}(x_a, \mu_F) f_{b/p_2}(x_b, \mu_F) |\mathcal{M}|^2 O_\omega(\Phi_n)$$

✴ Long distance physics: the parton densities
  (fitted from experimental data; evolution driven by QCD)

✴ Short distance physics: the differential parton cross section $d\sigma_{ab}$

✴ Separation of both regimes through the factorization scale $\mu_F$ (theory errors)
Predictions at the LHC (using QCD)

- Distribution of an observable \( \omega \): the QCD factorization theorem
  \[
  \frac{d\sigma}{d\omega} = \sum_{a,b} \int dx_a dx_b d\Phi_n f_{a/p_1}(x_a, \mu_F) f_{b/p_2}(x_b, \mu_F) |\mathcal{M}|^2 \omega(\Phi_n)
  \]
- Long distance physics: the parton densities
  (fitted from experimental data; evolution driven by QCD)
- Short distance physics: the differential parton cross section \( d\sigma_{ab} \)
- Separation of both regimes through the factorization scale \( \mu_F \) (theory errors)

- Short distance physics: the partonic cross section
  Calculated order by order in perturbative QCD:
  \[
  d\sigma = d\sigma^{(0)} + \alpha_s d\sigma^{(1)} + \ldots
  \]
  ★ The more orders included, the more precise the predictions
  ★ Truncation of the series and \( \alpha_s \gg \) theoretical uncertainties

Built on Feynman diagrams, from Feynman rules encoded in UFOs
Feynman diagram calculations

✦ Direct squared matrix element computations

◆ Extraction of the amplitude from the Feynman rules

\[ iM = ig_s^2 \left( \bar{v}_2 \gamma^\mu u_1 \right) \left( \eta_{\mu\nu} \left( \bar{u}_3 \gamma^\nu v_4 \right) \right) T_{c_2c_1}^a T_{c_3c_4}^a \]
Direct squared matrix element computations

- Extraction of the amplitude from the Feynman rules
  \[ i\mathcal{M} = ig_s^2 \left[ \bar{v}_2 \gamma^\mu u_1 \right] \frac{\eta_{\mu\nu}}{s} \left[ \bar{u}_3 \gamma^\nu v_4 \right] T^a_{c_2c_1} T^a_{c_3c_4} \]

- Squaring with the conjugate amplitude
- Algebraic calculation of the color and Lorentz structures
- Sum/average over the external states

\[
|\mathcal{M}|^2 = \frac{1}{36} \frac{2g_s^4}{s^2} \text{Tr} \left[ \bar{\phi}_1 \gamma^\mu \phi_2 \gamma^\nu \right] \left[ \bar{\phi}_3 \gamma_\mu \phi_4 \gamma_\nu \right]
= \frac{16g_s^4}{9s^2} \left[ (p_1 \cdot p_3)(p_2 \cdot p_4) + (p_1 \cdot p_4)(p_2 \cdot p_3) \right]
\]
**Direct squared matrix element computations**

- Extraction of the amplitude from the Feynman rules

\[
i \mathcal{M} = ig_s^2 \bar{v}_2 \gamma^\mu u_1 \bar{u}_3 \gamma^\nu v_4 \frac{\eta_{\mu\nu}}{s} T^a_{c_2c_1} T^a_{c_3c_4}
\]

- Squaring with the conjugate amplitude
- Algebraic calculation of the color and Lorentz structures
- Sum/average over the external states

\[
|\mathcal{M}|^2 = \frac{1}{36} \frac{2g_s^4}{s^2} \text{Tr} \left[ \gamma^\mu \gamma^\nu \gamma^\mu \gamma^\nu \right] \left[ \gamma^\mu \gamma^\nu \right] = \frac{16g_s^4}{9s^2} \left[ (p_1 \cdot p_3)(p_2 \cdot p_4) + (p_1 \cdot p_4)(p_2 \cdot p_3) \right]
\]

- The squared matrix element needs to be integrated

---

**Talk from S. Platzer**
Feynman diagram calculations

Direct squared matrix element computations

- Extraction of the amplitude from the Feynman rules

\[ iM = ig_s^2 \bar{v}_2 \gamma^\mu u_1 \eta_{\mu\nu} \left( \bar{u}_3 \gamma^\nu v_4 \right) T_{c_2 c_1} T_{c_3 c_4} \]

- Squaring with the conjugate amplitude
- Algebraic calculation of the color and Lorentz structures
- Sum/average over the external states

\[ |M|^2 = \frac{1}{36} \frac{2g_s^4}{s^2} \text{Tr} \left[ \gamma^\mu \gamma^\nu \right] \left[ \gamma^\mu \gamma^\nu \right] \]

\[ = \frac{16g_s^4}{9s^2} \left[ (p_1 \cdot p_3)(p_2 \cdot p_4) + (p_1 \cdot p_4)(p_2 \cdot p_3) \right] \]

- The squared matrix element needs to be integrated

The number of diagrams increases with the number of final-state particles

- The complexity rises as \( N^2 \)
- Any calculation beyond 2-to-3 becomes a problem

Talk from S. Platzer
**Principle**

- Evaluation of the amplitude for fixed external helicities
- Add all amplitudes (we get complex numbers)
- Squaring
- Sum/average over the external states
Helicity amplitudes

✦ Principle
- Evaluation of the amplitude for fixed external helicities
- Add all amplitudes (we get complex numbers)
- Squaring
- Sum/average over the external states

✦ Practical example
Practical example

1. External incoming particles (numbers)
   ★ For fixed helicity and momentum

\[ u_1 = f(p_1, m_1) \]

\[ \bar{v}_2 = f(p_2, m_2) \]
Helicity amplitudes

✦ Principle
- Evaluation of the amplitude for fixed external helicities
- Add all amplitudes (we get complex numbers)
- Squaring
- Sum/average over the external states

✦ Practical example

\[ W = f(\bar{v}_2, u_1, \Gamma_1) \propto \bar{v}_2 \gamma^\mu u_1 \frac{\eta_{\mu \nu}}{s} \]

1. External incoming particles (numbers)
   ★ For fixed helicity and momentum
2. Wave function of the gluon propagator
Ideas

Principle
- Evaluation of the amplitude for fixed external helicities
- Add all amplitudes (we get complex numbers)
- Squaring
- Sum/average over the external states

Practical example

\[ W = f(\bar{v}_2, u_1, \Gamma_1) \propto \bar{v}_2 \gamma^{\mu} u_1 \frac{\eta_{\mu\nu}}{s} \]

1. External incoming particles (numbers)
   - For fixed helicity and momentum
2. Wave function of the gluon propagator
3. External outgoing particles
**Helicity amplitudes**

**Principle**
- Evaluation of the amplitude for fixed external helicities
- Add all amplitudes (we get complex numbers)
- Squaring
- Sum/average over the external states

**Practical example**

1. **External incoming particles (numbers)**
   - For fixed helicity and momentum
2. Wave function of the gluon propagator
3. External outgoing particles
4. Full amplitude (complex number)

\[ W = f(\bar{v}_2, u_1, \Gamma_1) \propto \bar{v}_2 \gamma^\mu u_1 \frac{\eta_{\mu\nu}}{s} \]

\[ \bar{u}_3 = f(p_3, m_3) \]

\[ u_1 = f(p_1, m_1) \]

\[ \bar{v}_2 = f(p_2, m_2) \]

\[ v_4 = f(p_4, m_4) \]

\[ iM = f(\bar{u}_3, v_4, W, \Gamma_2) \]
The building blocks of the amplitude are the so-called HELAS functions

- HELAS = HELicity Amplitude Subroutine
- One specific routine for each Lorentz structure ($\mathcal{I}_i$)
- Not generic for any model

- **SM**  
  [ Murayama, Watanabe & Hagiwara (KEK-91-11) ]
- **MSSM**  
  [ Cho, Hagiwara, Kanzaki, Plehn, Rainwater & Stelzer (PRD'06) ]
- **HEFT**  
  [ Frederix (2007) ]
- **Spin 2**  
  [ Hagiwara, Kanzaki, Li & Mawatari (EPJC'08) ]
- **Spin 3/2**  
  [ Mawatari & Takaesu (EPJC'11) ]

Sufficient for many models
The building blocks of the amplitude are the so-called HELAS functions

- **HELAS** ≡ HELicity Amplitude Subroutine
- One specific routine for each Lorentz structure \((\Gamma_i)\)
- Not generic for any model
  - ★ SM  [ Murayama, Watanabe & Hagiwara (KEK-91-11) ]
  - ★ MSSM  [ Cho, Hagiwara, Kanzaki, Plehn, Rainwater & Stelzer (PRD’06) ]
  - ★ HEFT  [ Frederix (2007) ]
  - ★ Spin 2  [ Hagiwara, Kanzaki, Li & Mawatari (EPJC’08) ]
  - ★ Spin 3/2  [ Mawatari & Takaesu (EPJC’11) ]

- Sufficient for many models

Generalization: ALOHA  [ de Aquino, Link, Maltoni, Mattelaer & Stelzer (CPC’12) ]

- Translation of any vertex present in a UFO into a HELAS subroutine
- Any model can hence be coped with by MG5_aMC@NLO

Other MC generators are today connected directly to the UFO
- **HERWIG, SHERPA, WHIZARD**
Heavy particle decays

✧ Concrete BSM models
  ✤ Many additional new states
    ★ Usually pair-produced
    ★ Cascade-decaying into each other
  ✤ The lightest new state can be stable (and a dark matter candidate)

Is the simulation of 2-to-N processes (with a large $N$) a problem?
Heavy particle decays

Concrete BSM models
- Many additional new states
  - Usually pair-produced
  - Cascade-decaying into each other
- The lightest new state can be stable
  (and a dark matter candidate)

Is the simulation of 2-to-N processes
(with a large $N$) a problem?

2-to-N matrix-element generation is possible
- Nothing really new or fancy
- Computationally challenging for event generation

The issue is the computing time
- Connected to the final-state multiplicity
- Practically useless: diagrams with intermediate resonances dominate

Factorization of the production from the decay
Making decays easy: the key principle

✦ Production and decay processes are factorized
  ✤ Propagators can be seen as sums of products of external wave functions
  ✤ Example for a vector resonance

\[
\mathcal{M} = j_1^\mu \left[ \eta_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda j_1^\mu \varepsilon^*_\mu(\lambda) j_2^\nu \varepsilon_\nu(\lambda)
\]

Propagation

Production of the resonance

Decay of the resonance
Making decays easy: the key principle

✦ Production and decay processes are factorized
  ✷ Propagators can be seen as sums of products of external wave functions
  ✷ Example for a vector resonance

\[ M = j_1^\mu \left[ \eta_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda j_1^\mu \varepsilon^*_\mu(\lambda) j_2^\nu \varepsilon_\nu(\lambda) \]

✧ Off-shell effects are lost (as a result of the factorization)
  ★ Resonance mass smearing: partial recovery  
  [Frixione, Laenen, Motylinksi, Webber (JHEP '07)]
Practical implementations of decays

✦ Case 1: loss of spin correlations

✦ Helicity sums performed independently at the production and decay levels

\[ M = j_1^\mu \left[ \eta_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda j_1^\mu \varepsilon_\mu^*(\lambda) \ j_2^\nu \varepsilon_\nu(\lambda) \approx \sum_\lambda j_1^\mu \varepsilon_\mu^*(\lambda) \sum_{\lambda'} j_2^\nu \varepsilon_\nu(\lambda') \]

★ Used in PYTHIA
Practical implementations of decays

Case 1: loss of spin correlations

- Helicity sums performed independently at the production and decay levels

\[ M = j_1^\mu \left[ \eta_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda j_1^\mu \varepsilon^*_\mu(\lambda) j_2^\nu \varepsilon_\nu(\lambda) \approx \sum_\lambda j_1^\mu \varepsilon^*_\mu(\lambda) \sum_{\lambda'} j_2^\nu \varepsilon_\nu(\lambda') \]

- Used in PYTHIA

Case 2: including spin correlations

- Helicity sums performed after accounting for production and decays

\[ M = \sum_\lambda j_1^\mu \varepsilon^*_\mu(\lambda) j_2^\nu \varepsilon_\nu(\lambda) \]

- Reweighting according to decay matrix element (e.g. MADSPIN) [Artoisenet et al. (JHEP ’13)]

- Using spin density matrices (e.g. HERWIG, SHERPA) [Richardson (JHEP ’01); Höche et al. (EPJ C ’15)]
Importance of correctly handling decays

✦ Is a correct decay handling important: this depends on the observable

Angle between the leptons in the respective mother top rest frames

Invariant mass between decay products originating from different cascade steps

MADSPIN

$\bar{t}tH$ production @ (N)LOQCD
[ LHC8, dileptonic $\bar{t}t$ decay]

[ Artoisenet, Frederix, Mattelaer & Rietkerk (JHEP'13) ]

SHERPA @ LO[ LHC8 ]

$pp \rightarrow \tilde{u}\tilde{u}^\dagger$

$\tilde{u} \rightarrow d\tilde{\chi}_1^+ \rightarrow d\chi_1^0 W^+ \rightarrow d\chi_1^0 \mu^+ \nu_\mu$

$\tilde{u}^\dagger \rightarrow ... \rightarrow \bar{u}e^+ e^- \tilde{\chi}_1^0$

[ Höche, Kuttimalai, Schumann & Siegert (EPJ'C'15) ]
Outline

1. Introduction: overview of an LHC collision
2. Implementing models into Monte Carlo event generators
3. From models to hard scattering events
4. Summary
Event simulation is a complex process
- Nature allows us to factorize it into pieces
- Event simulation is performed step-by-step

This talk: 1st parts of the simulation chain
- Connecting models (Lagrangians) to tools
- Generating squared matrix elements
- Including the decays of heavy particles