New physics at colliders
A tools vision

Fuks Benjamin

LPTHE / UPMC

Tools 2017: Tools for the SM and the New Physics
@ Corfu, Greece

September 9 - 14, 2017

Outline

1. New physics & Monte Carlo simulations
2. Model implementations
3. Cascade decays
4. Towards precision: merging and next-to-leading order corrections
5. Conclusions - summary

Standard Model simulations: the status

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

- Matrix-element generation (MADGRAPH5, CALCHEP, FEYNARTS, 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.*)

See talk by Peter Richardson

See talk by Gionata Luisoni

See talk by Marek Schoenherr

Standard Model 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)

Standard Model simulations under control
What about new physics?

New physics simulations: the challenges

- ✦ The challenges with respect to new physics simulations are 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

However...

New physics simulations: the challenges

- New physics is a standard in many tools today
  - Result of 20 years of developments
  - Simulations were usually mostly achieved at the leading-order accuracy in QCD
  - This has started to change a couple of years ago (NLO-QCD is available)

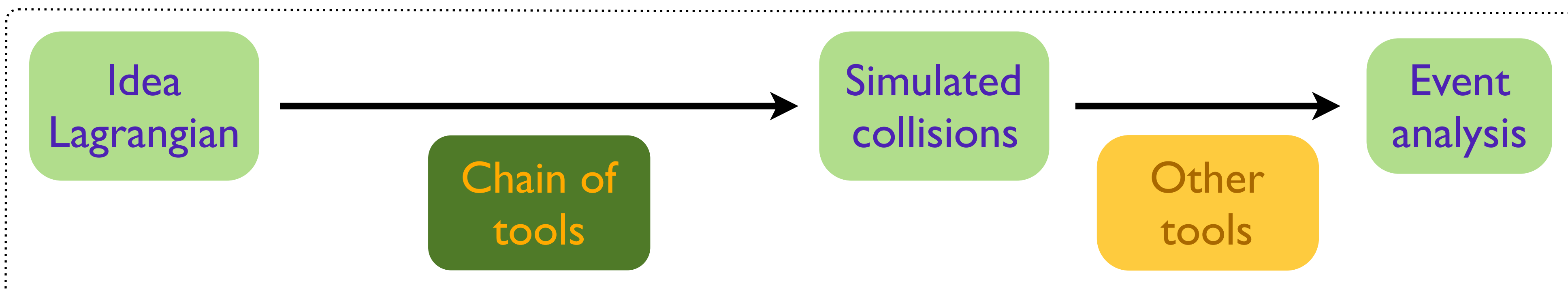
What are the ingredients behind this success?

Outline

1. New physics & Monte Carlo simulations
2. **Model implementations**
3. Cascade decays
4. Towards precision: merging and next-to-leading order corrections
5. Conclusions - summary

New physics @ colliders: keys for a success

- The links of a physics models to analyzed simulated collisions are 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



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, transparent
    - ★ No private tools

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

Towards an MC framework for BSM - step I

✦ The first steps: LanHEP

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

- ✤ Automatic linking of Lagrangians to files in a given programming language
- ✤ Working environment: C
- ✤ Initially restricted to CalcHEP / CompHEP
- ✤ Can now generate FeynArts and UFO outputs (≡ interface to many tools)

A. V. Semenov

2002

LanHEP -- a package for automatic generation of Feynman rules in field theory. Version 2.0

Abstract. The LanHEP program for Feynman rules generation in momentum representation is presented. It reads the Lagrangian written in a compact form, close to the one used in publications. It means that Lagrangian terms can be written with summation over indices of broken symmetries and using special symbols for complicated expressions, such as covariant derivative and strength tensor for gauge fields. The output is Feynman rules in terms of physical fields and independent parameters. This output can be written in LaTeX format and in the form of CompHEP model files, which allows one to start calculations of processes in the new physical model. Although this job is rather straightforward and can be done manually, it requires careful calculations and in modern theories with many particles and vertices, such as supersymmetric models, can lead to errors and misprints. The program allows one to introduce into CompHEP new gauge theories as well as various anomalous terms.

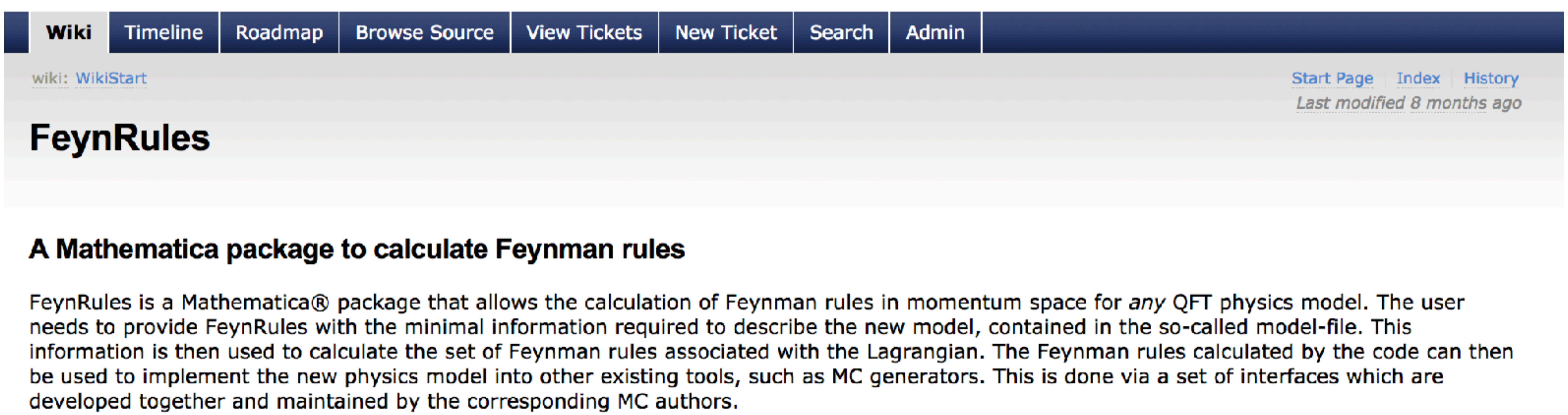
http://theory.sinp.msu.ru/~semenov/lanhep.html

The FeynRules package

✦ The FeynRules platform

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

- ✤ Automatic linking of Lagrangians to files in a given programming language
- ✤ Working environment: Mathematica
  - ★ Flexibility, symbolic manipulations, easy implementation of new methods, *etc.*
  - ★ Shipped with many computation tools (superspace, spectrum, decays, NLO, *etc.*)
- ✤ Interfaced to many Monte Carlo tools
  - ★ Dedicated translators to several tools (CalcHep, FeynArts, and more in the past)
  - ★ Interfaced to more tools via the UFO (Herwig, MG5_aMC, Sherpa, Whizard, *etc.*)



Wiki | Timeline | Roadmap | Browse Source | View Tickets | New Ticket | Search | Admin

wiki: WikiStart

Start Page | Index | History

Last modified 8 months ago

FeynRules

A Mathematica package to calculate Feynman rules

FeynRules is a Mathematica® package that allows the calculation of Feynman rules in momentum space for *any* QFT physics model. The user needs to provide FeynRules with the minimal information required to describe the new model, contained in the so-called model-file. This information is then used to calculate the set of Feynman rules associated with the Lagrangian. The Feynman rules calculated by the code can then be used to implement the new physics model into other existing tools, such as MC generators. This is done via a set of interfaces which are developed together and maintained by the corresponding MC authors.

http://feynrules.irmp.ucl.ac.be/

The SARAH program

✦ The SARAH package

[Staub (CPC'13; CPC'14)]

- ✤ Automatic linking of Lagrangians to files in a given programming language
- ✤ Working environment: MATHEMATICA
- ✤ Spectrum generator features

SARAH https://sarah.hepforge.org/

Current version

The current version is **4.12.2** (Download)
Last update: 01.09.2017 (Changelog)

Description

SARAH is a Mathematica package for **building and analyzing SUSY and non-SUSY models**. It calculates all vertices, mass matrices, tadpoles equations, one-loop corrections for tadpoles and self-energies, and two-loop RGEs for a given model. SARAH writes **model files** for FeynArts, CalcHep/CompHep, which can also be used for dark matter studies using MicrOmegas, the UFO format which is supported by MadGraph 5 and for WHIZARD and OMEGA.
SARAH is also the first available **spectrum-generator-generator**: based on the derived, analytical expression it creates source code for SPheno. In that way, it is possible to implement new models in SPheno without the need to write any Fortran code by hand. The output for Vevacious can be used to check for the global minimum for a given model and parameter point.

The SARAH program

✦ The SARAH package

[Staub (CPC'13; CPC'14)]

- Automatic linking of Lagrangians to files in a given programming language
- Working environment: MATHEMATICA
- Spectrum generator features

SARAH https://sarah.hepforge.org/

Current version

The current version is **4.12.2** (Download)
Last update: 01.09.2017 (Changelog)

Description

SARAH is a Mathematica package for **building and analyzing SUSY and non-SUSY models**. It calculates all vertices, mass matrices, tadpoles equations, one-loop corrections for tadpoles and self-energies, and two-loop RGEs for a given model. SARAH writes **model files** for FeynArts, CalcHep/CompHep, which can also be used for dark matter studies using MicrOmegas, the UFO format which is supported by MadGraph 5 and for WHIZARD and OMEGA.
SARAH is also the first available **spectrum-generator-generator**: based on the derived, analytical expression it creates source code for SPheno. In that way, it is possible to implement new models in SPheno without the need to write any Fortran code by hand. The output for Vevacious can be used to check for the global minimum for a given model and parameter point.

New physics simulations in details

✦ A comprehensive approach to Monte Carlo simulations

Idea / Lagrangian → Simulated collisions → Event analysis

FEYNRULES LANHEP SARAH

Matrix Element Generator

Shower Hadronization

Detector Reconstruction

Analysis codes

✦ Implementation of any new physics theory in a MC tool is straightforward

Many interfaces dedicated to specific tools

★ Removal of non compliant vertices

★ Translation to a specific format/language

⚠ **Not efficient**

and more not represented here

FEYNARTS: talk by Thomas Hahn

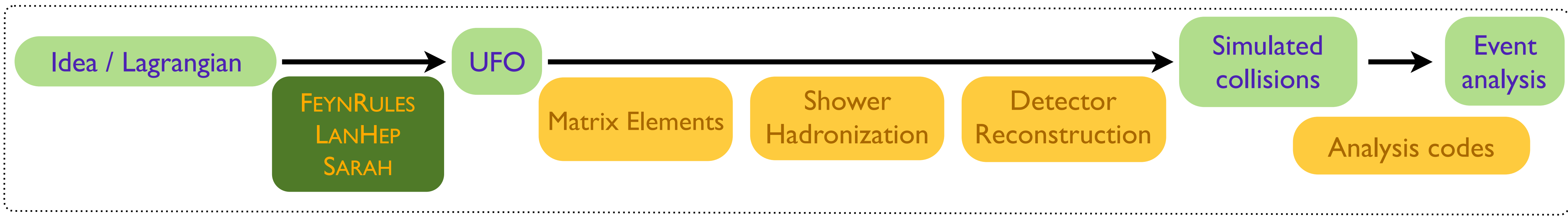
MG5_AMC: talk by Olivier Mattelaer

WHIZARD: talk by Juergen Reuter

HERWIG: talk by Peter Richardson

New physics simulations in details

✦ A comprehensive approach to Monte Carlo simulations



One format to rule them all

- ★ Easier to maintain
- ★ The MC generator decides what is needed

A step further: the Universal FEYNRULES Output

✦ The UFO in a nutshell

[Degrande, Duhr, BF, Grellscheid, Mattelaer, Reiter (CPC '12)]
[Degrande, Duhr, BF, Hirschi, Mattelaer, Shao *et al.* (*in prep.*)]

- ✤ UFO ≡ 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

✦ The UFO is now a standard and used by many other programs

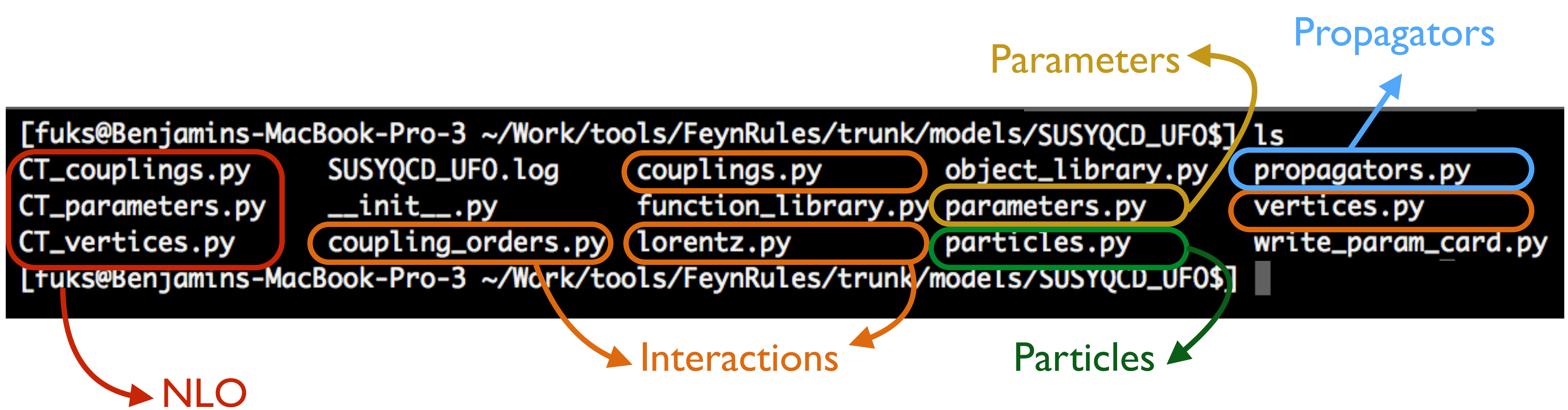
ALOHA | GOSAM | HERWIG ++ | MADANALYSIS 5 | SHERPA

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**



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

```
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)

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()
```

```
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()
```

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

```
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 ])
```

Interactions: generalities

✦ Vertices decomposed in a spin x color basis (coupling strengths ≡ coordinates)

✤ Example: the quartic gluon vertex can be written as

$$\begin{aligned} & i g_s^2\, f^{a_1 a_2 b} f^{b a_3 a_4}\, \left(\eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4}\right) \\ & \quad + i g_s^2\, f^{a_1 a_3 b} f^{b a_2 a_4}\, \left(\eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4}\right) \\ & \quad + i g_s^2\, f^{a_1 a_4 b} f^{b a_2 a_3}\, \left(\eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4}\right) \end{aligned} \quad \Rightarrow \quad \begin{aligned} & \left(f^{a_1 a_2 b} f^{b a_3 a_4},\ f^{a_1 a_3 b} f^{b a_2 a_4}, f^{a_1 a_4 b} f^{b a_2 a_3}\right) \\ & \times \begin{pmatrix} i g_s^2 & 0 & 0 \\ 0 & i g_s^2 & 0 \\ 0 & 0 & i g_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix} \end{aligned}$$

- ★ 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

```
V_2 = Vertex(name = 'V_2',
             particles = [ P.G, P.G, P.G, P.G ],
             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)

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ f^{a_1a_3b}f^{ba_2a_4}, f^{a_1a_4b}f^{ba_2a_3}\right)$$
$$\times\begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix}\begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

Example: the quartic gluon vertex

✦ General information in vertex.py

```
V_2 = Vertex(name = 'V_2',
             particles = [ P.G, P.G, P.G, P.G ],
             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)

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ f^{a_1a_3b}f^{ba_2a_4},\ f^{a_1a_4b}f^{ba_2a_3}\right) \times \begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

✦ Lorentz structures: straightforward implementations in lorentz.py

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

Example: the quartic gluon vertex

✦ General information in vertex.py

```
V_2 = Vertex(name = 'V_2',
             particles = [ P.G, P.G, P.G, P.G ],
             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)

$$\left(f^{a_1a_2b}f^{ba_3a_4},\ f^{a_1a_3b}f^{ba_2a_4},\ f^{a_1a_4b}f^{ba_2a_3}\right) \times \begin{pmatrix} ig_s^2 & 0 & 0 \\ 0 & ig_s^2 & 0 \\ 0 & 0 & ig_s^2 \end{pmatrix} \begin{pmatrix} \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} \\ \eta^{\mu_1\mu_4}\eta^{\mu_2\mu_3} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \\ \eta^{\mu_1\mu_3}\eta^{\mu_2\mu_4} - \eta^{\mu_1\mu_2}\eta^{\mu_3\mu_4} \end{pmatrix}$$

✦ Lorentz structures: straightforward implementations in lorentz.py

```
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

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

Coupling orders: for selecting diagrams

Outline

1. New physics & Monte Carlo simulations
2. Model implementations
3. **Cascade decays**
4. Towards precision: merging and next-to-leading order corrections
5. Conclusions - summary

Cascade decays

✦ Concrete models

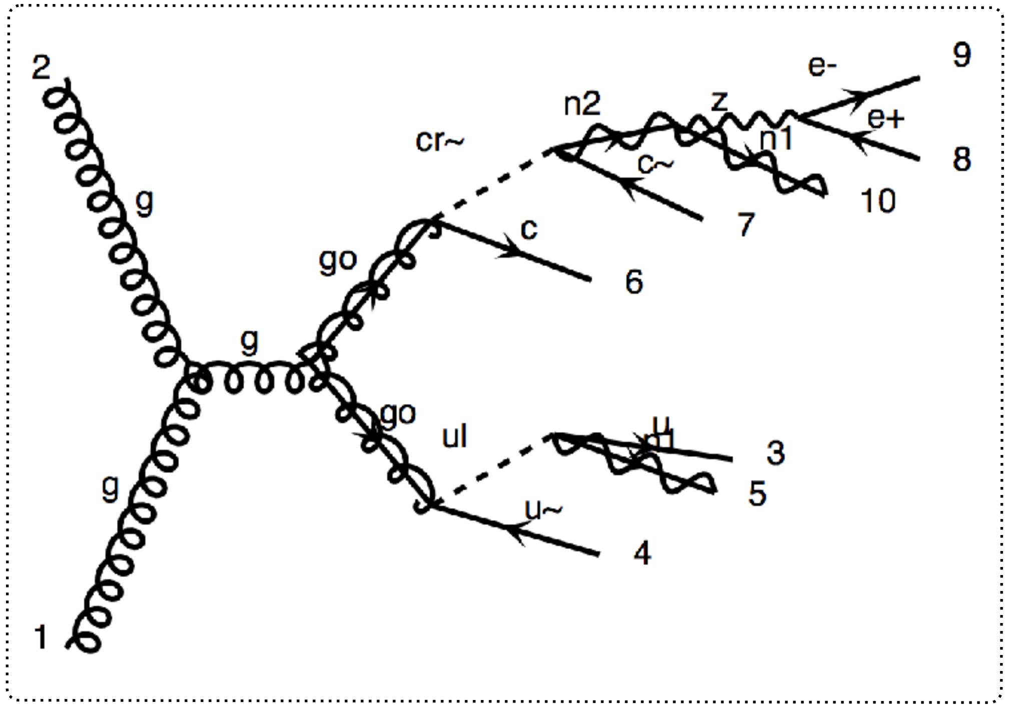
✤ Many new states are supplemented to the Standard Model

★ Usually pair-produced

★ Cascade-decaying into each other

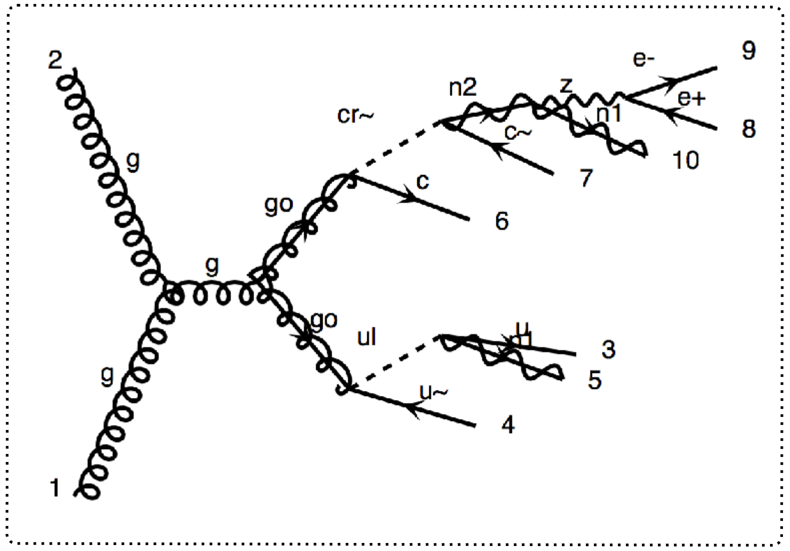
✤ The lightest new state can be stable (and a dark matter candidate)

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



Simulating cascade decays

- 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

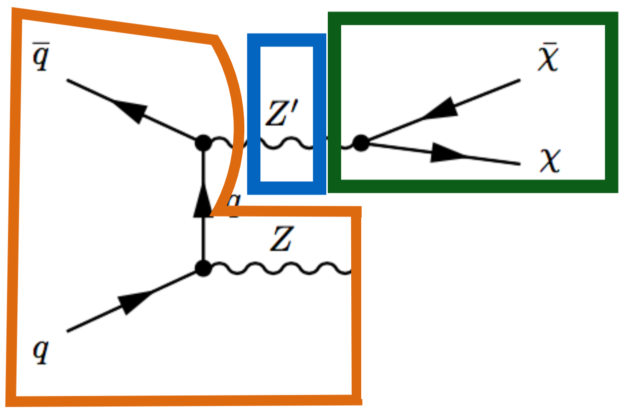
- Production and decay processes are factorized
  - Propagators can be seen as sums of products of external wave functions
  - Example for a vector resonance

$$\mathcal{M} \sim j_1^\mu \left[g_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda \underbrace{j_1^\mu \varepsilon^*_\mu(\lambda)}_{\mathcal{M}_{\rm prod}(\lambda)} \underbrace{\varepsilon_\nu(\lambda) j_2^\nu}_{\mathcal{M}_{\rm dec}(\lambda)}$$

Propagation

Production of the resonance

Decay of the resonance



- 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

$$\mathcal{M} \sim j_1^\mu \left[g_{\mu\nu} - \frac{p_\mu p_\nu}{p^2} \right] j_2^\nu = \sum_\lambda \underbrace{j_1^\mu \varepsilon_\mu^*(\lambda)}_{\mathcal{M}_{\rm prod}(\lambda)} \underbrace{\varepsilon_\nu(\lambda) j_2^\nu}_{\mathcal{M}_{\rm dec}(\lambda)} \simeq \sum_\lambda \underbrace{j_1^\mu \varepsilon_\mu^*(\lambda)}_{\mathcal{M}_{\rm prod}(\lambda)} \sum_{\lambda'} \underbrace{\varepsilon_\nu(\lambda') j_2^\nu}_{\mathcal{M}_{\rm dec}(\lambda')}$$

PYTHIA 8 [Sjostrand, *et al.* (CPC '08)]

PYTHIA 6 [Sjostrand, Mrenna, Skands (JHEP '06)]

Practical implementations of decays

- ✦ Case 2: including spin correlations
  - ✤ Helicity sums performed after accounting for production and decays

$$\sum_\lambda \underbrace{j_1^\mu \varepsilon_\mu^*(\lambda)}_{\mathcal{M}_{\rm prod}(\lambda)} \underbrace{\varepsilon_\nu(\lambda) j_2^\nu}_{\mathcal{M}_{\rm dec}(\lambda)}$$

HERWIG [Richardson (JHEP '01)]

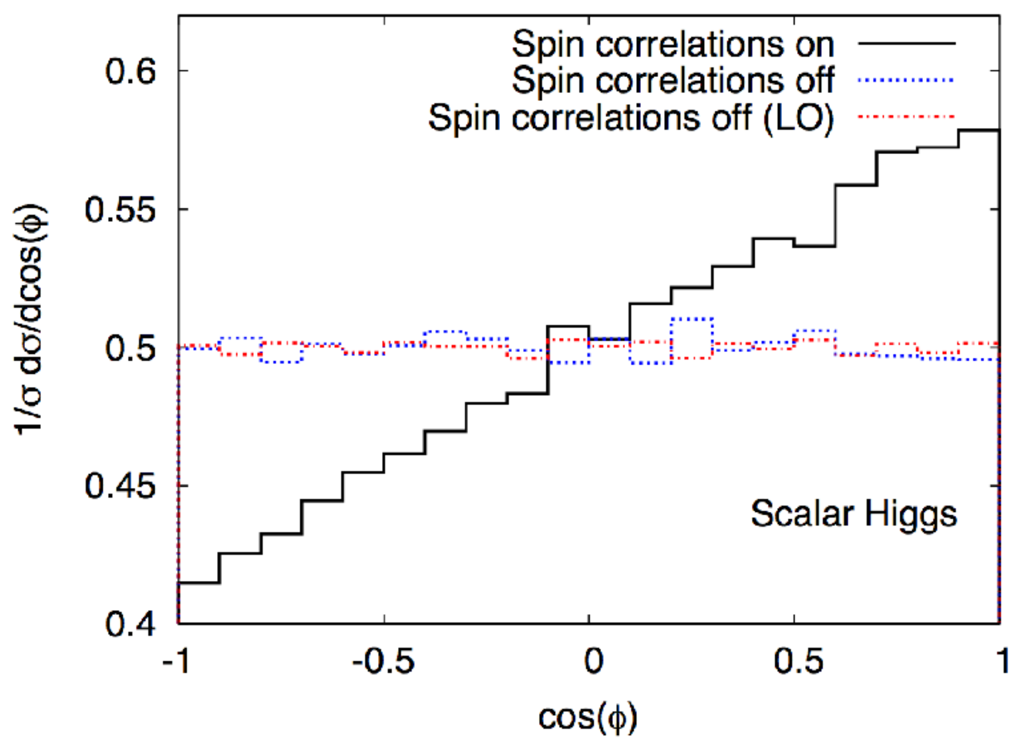
MADSPIN [Artoisenet *et al.* (JHEP '13)]

SHERPA [Höche *et al.* (EPJC '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



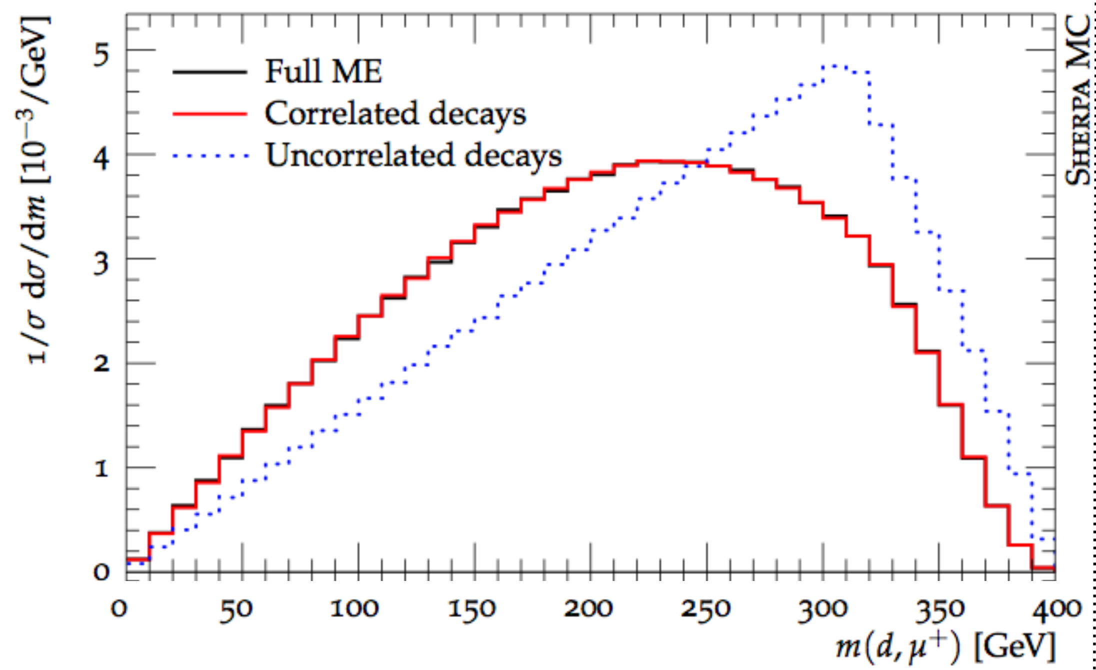
MADSPIN

$t\bar{t}H$ production @ (N)LOQCD

[LHC8, dileptonic $t\bar{t}$ decay]

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

Invariant mass between decay products originating from different cascade steps



SHERPA @ LO[LHC8]

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

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

$\tilde{u}^\dagger \to \ldots \to \bar{u}e^+e^-\tilde{\chi}_1^0$

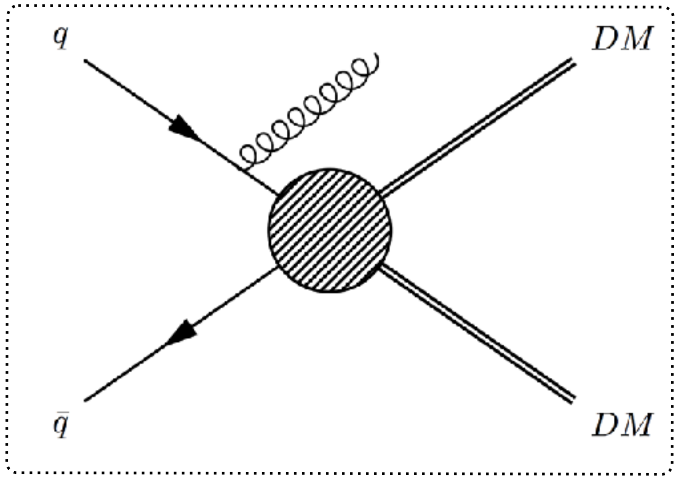
[Höche, Kuttimalai, Schumann & Siegert (EPJC'15)]

[Höche, Kuttimalai, Schumann & Siegert (EPJC'15)]

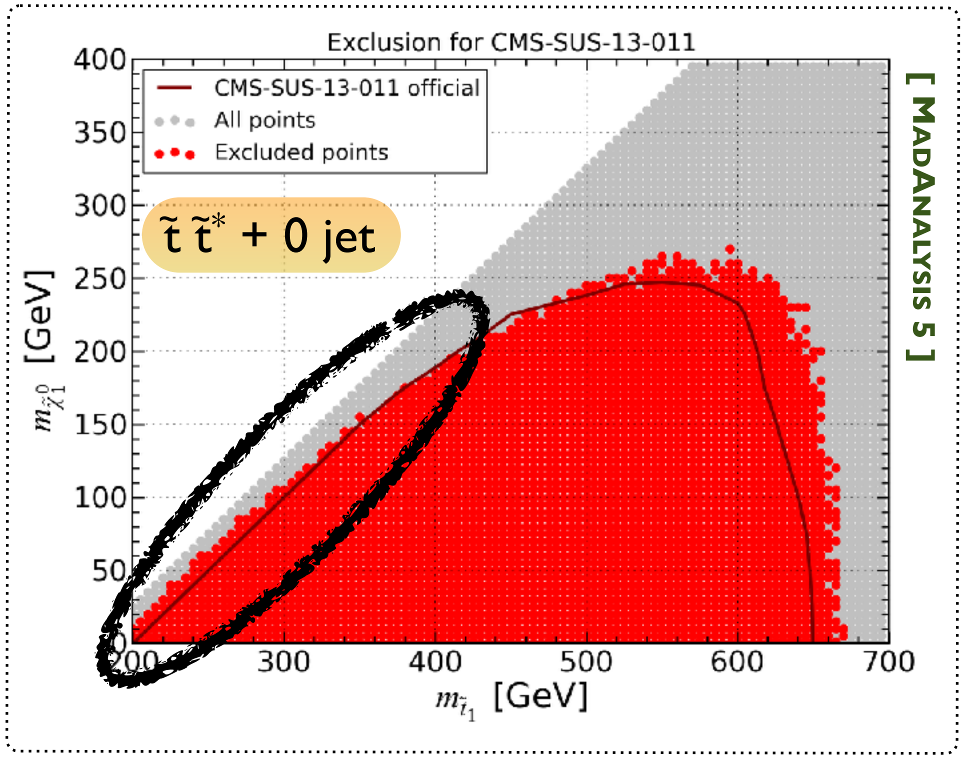
Outline

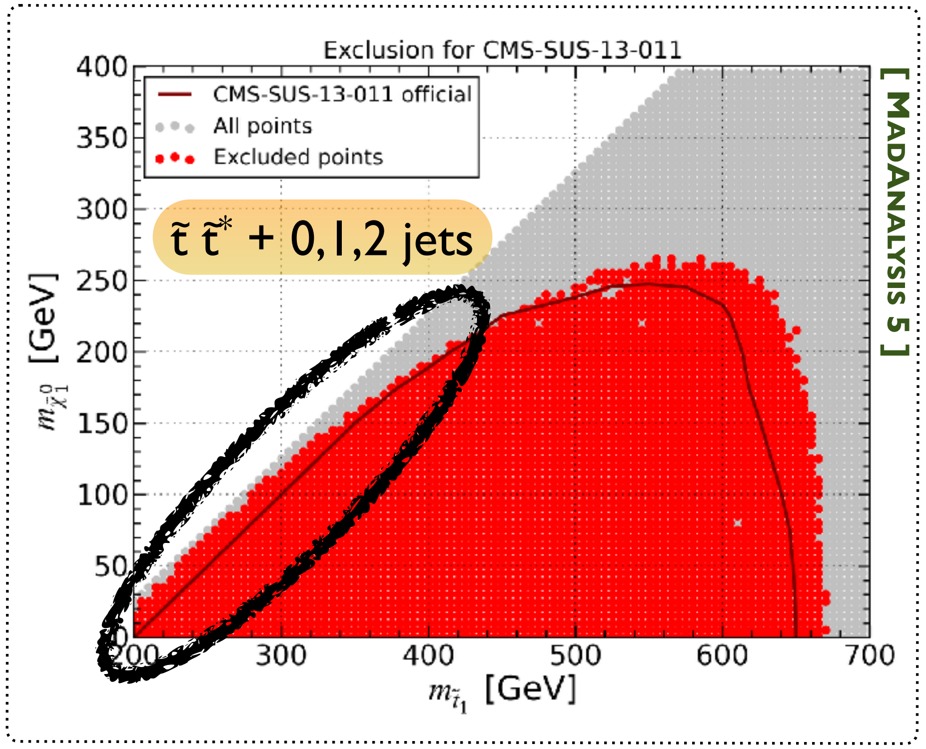
Importance of the extra QCD emissions

- Initial (and final) state radiation modeling is crucial
  - Monojet-based dark matter searches
  - Compressed spectra searches
  - Electroweak new physics
  - *etc.*



- Effects on stop pair production





Matrix elements and parton showers

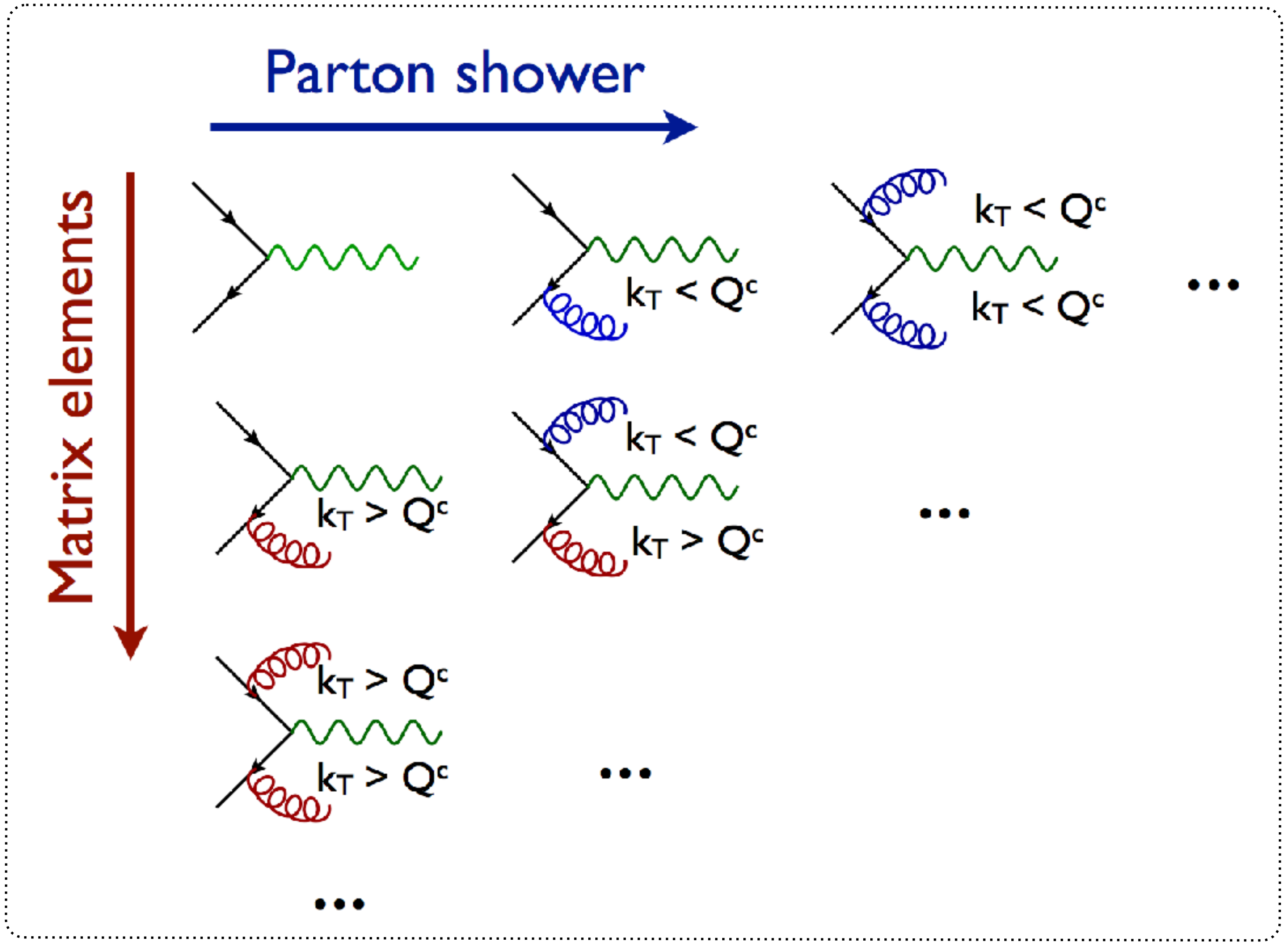
- ✦ Matrix-element-based predictions
  - ✤ Relies on the fixed-order theory
  - ✤ Technical limit on the number of final-state particles
  - ✤ Valid for hard and well-separated partons
  - ✤ Correct handling of color and spin information, and of interferences
- ✦ Parton-shower-based predictions
  - ✤ Resumation of large soft-collinear logarithms
  - ✤ Technically easy and no limit on the final-state multiplicity
  - ✤ Valid for soft and/or collinear partons
  - ✤ Approximate handling of color and spin information, and of interferences

Multipartonic matrix-element merging

- ✦ Matrix-element and parton-showers are complementary
  - ✤ Both can be combined

See talk by Marek Schoenherr

- ✦ The double-counting of specific radiation must be prevented



- ✤ Matrix elements:
  - ⇒ only hard radiation
- ✤ Parton showers:
  - ⇒ only soft-collinear
- ✤ Cut in phase space (Q^c)
- ✤ Check of the procedure
  - ★ Matrix elements mimic parton showers near Q^c
  - ★ Verification with Q^c independent observables

Higher-order corrections (in QCD)

✦ Other option: NLO calculations

- ✤ Correct modeling of the first emission
- ✤ Merging of samples with different jet multiplicities also possible

✦ NLO calculations matched to parton shower (for BSM) are automated

- ✤ Model-dependent parts of calculations (on top of the tree-level information)
  - ★ Counterterms
  - ★ Finite pieces of the loop-integrals
- ✤ Model independent contributions
  - ★ Subtraction of the divergences
  - ★ Matching to the parton showers

UFO @ NLO

[Degrande, Duhr, BF, Hirschi, Mattelaer, Shao *et al.* (*in prep.*)]

See talk by
Gionata Luisoni

Recap' on NLO calculations

✦ Contributions to an NLO result in QCD

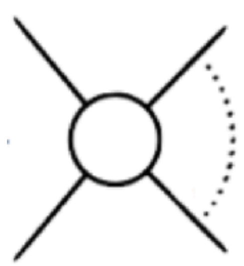
✤ Three ingredients: the Born, virtual loop and real emission contributions

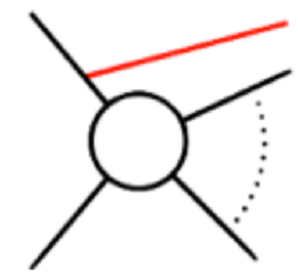
$$\sigma_{NLO} = \int \mathrm{d}^4\Phi_n \mathcal{B} \quad + \boxed{\int \mathrm{d}^4\Phi_n \int_{\mathrm{loop}} \mathrm{d}^d\ell \; \mathcal{V}} + \int \mathrm{d}^4\Phi_{n+1} \; \mathcal{R}$$

Born

Virtuals: one extra power of α_s and divergent

Reals: one extra power of α_s and divergent





Extra information is needed

Virtual contributions

✦ Loop diagram calculations

✤ Calculations to be done in d=4-2ε dimensions

★ Divergences made explicit (1/ε^2, 1/ε)

✤ Rewriting loop integrals with scalar integrals

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

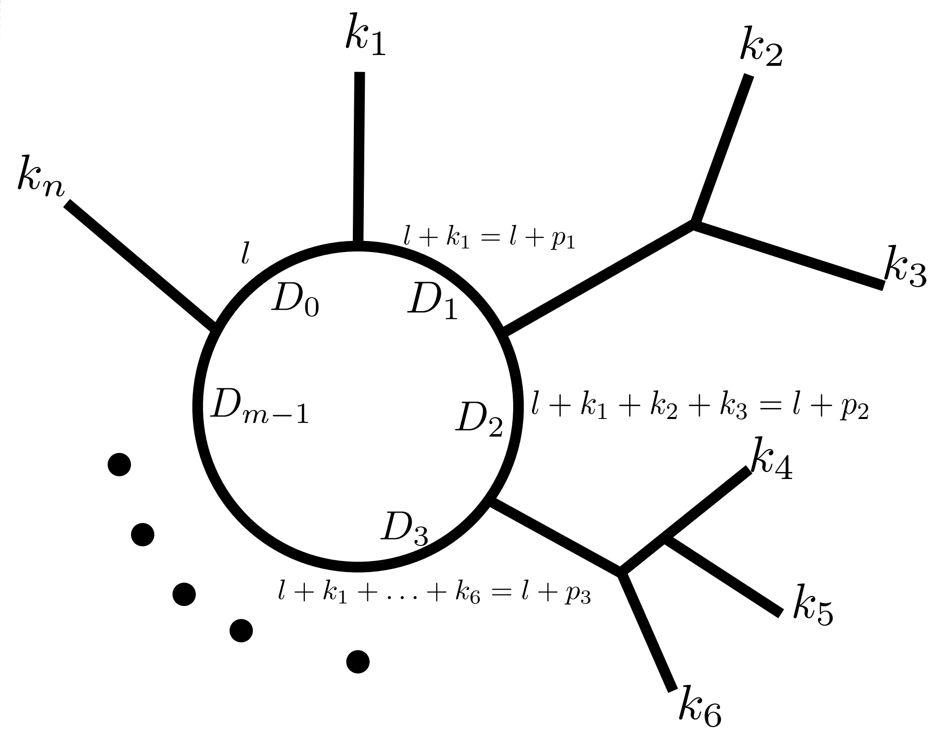
★ Involves integrals with up to four denominators

➢ 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



The rational terms (R_1 and R_2)

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

$$\int \mathrm{d}^d\ell \frac{N(\ell,\tilde{\ell})}{\bar{D}_0\bar{D}_1\cdots\bar{D}_{m-1}} \quad \text{with } \underbrace{\bar{\ell}}_{\text{D-dim}} = \underbrace{\ell}_{\text{4-dim}} + \underbrace{\tilde{\ell}}_{(-2\varepsilon)\text{-dim}}$$

  - 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
  - Can be seen as extra diagrams with special Feynman rules

R_I terms

- The R_I terms originates from the denominators

$$\frac{1}{\bar{D}} = \frac{1}{D}\left(1 - \frac{\tilde{\ell}^2}{\bar{D}}\right)$$

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

$$\int \mathrm{d}^d\bar{\ell}\,\frac{\tilde{\ell}^2}{\bar{D}_i\bar{D}_j} = -\frac{i\pi^2}{2}\left[m_i^2 + m_j^2 - \frac{p_i - p_j)^2}{2}\right] + \mathcal{O}(\varepsilon)$$

$$\int \mathrm{d}^d\bar{\ell}\,\frac{\tilde{\ell}^2}{\bar{D}_i\bar{D}_j\bar{D}_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon)$$

$$\int \mathrm{d}^d\bar{\ell}\,\frac{\tilde{\ell}^2}{\bar{D}_i\bar{D}_j\bar{D}_k\bar{D}_l} = -\frac{i\pi^2}{6} + \mathcal{O}(\varepsilon)$$

- The denominator structure is already known at the reduction time
- The R_I coefficients are extracted during the reduction

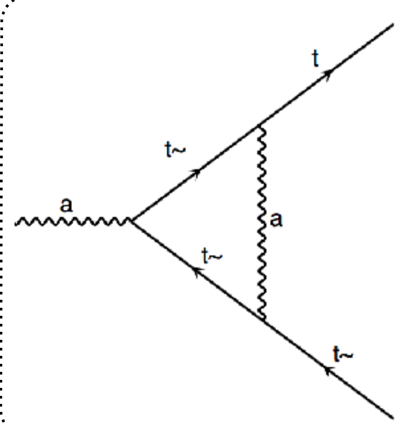
R_2 terms

- The R_2 terms originates from the numerator

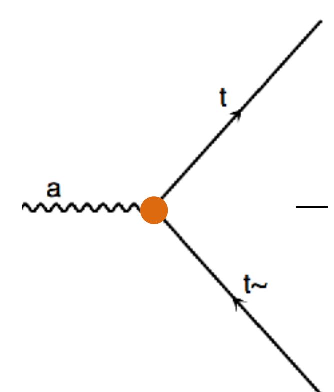
$$\underset{\text{D-dim}}{\bar{N}(\bar{\ell})} = \underset{\text{4-dim}}{N(\ell)} + \underset{\text{(-2}\varepsilon\text{)-dim}}{\tilde{N}(\tilde{\ell}, \ell, \varepsilon)} \quad \Rightarrow \quad R_2 \equiv \lim_{\varepsilon \to 0} \frac{1}{(2\pi)^4} \int \mathrm{d}^d \bar{\ell} \frac{\tilde{N}(\tilde{\ell},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

  - 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 R_2 Feynman rules



$$\propto \int \mathrm{d}^d \bar{\ell} \frac{\tilde{\ell}^2}{\bar{D}_i \bar{D}_j \bar{D}_k} = -\frac{i\pi^2}{2} + \mathcal{O}(\varepsilon) \quad \Rightarrow$$



$$-i \frac{\alpha}{2\pi} e \gamma^\mu$$

R_2 Feynman rules

- ✦ 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

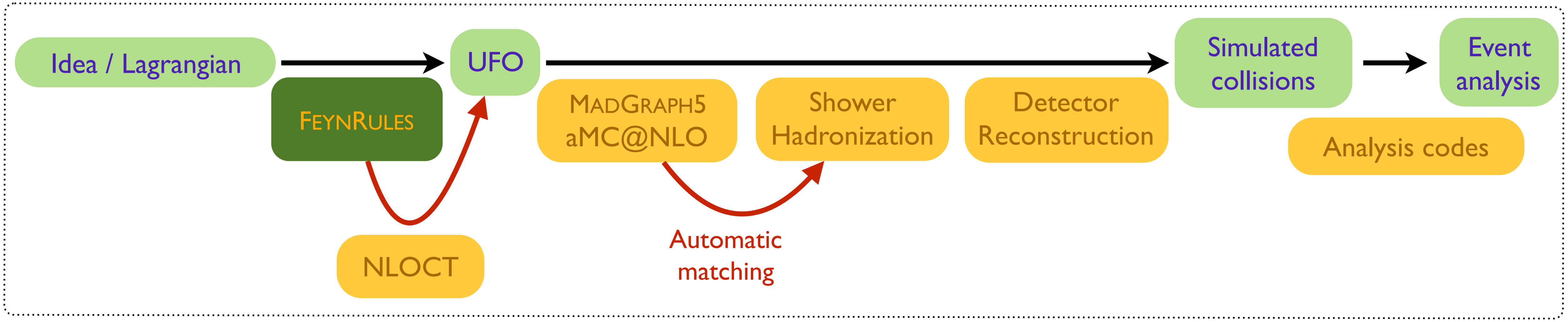
[Ossala, Papadopoulos, Pittau (JHEP'08)]

- ✦ 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

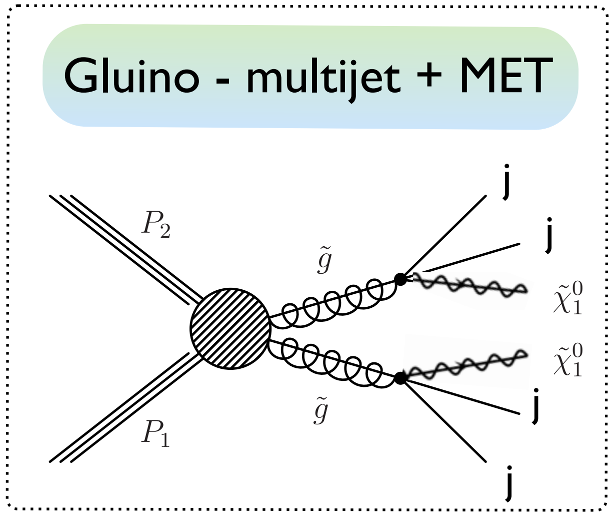
✦ A comprehensive approach to Monte Carlo simulations at the NLO in QCD



Importance of NLO: gluino pair production

[Degrande, BF, Hirschi, Proudom & Shao (PRD'15; PLB'16)]

✦ We produce two gluinos that each decays into 2 jets and missing energy



- Decays via decoupled virtual squarks
- Topology: 4 jets (2 for each gluinos) and missing energy
- Important jet activity (massive colored particle production)

✦ Two types of jets

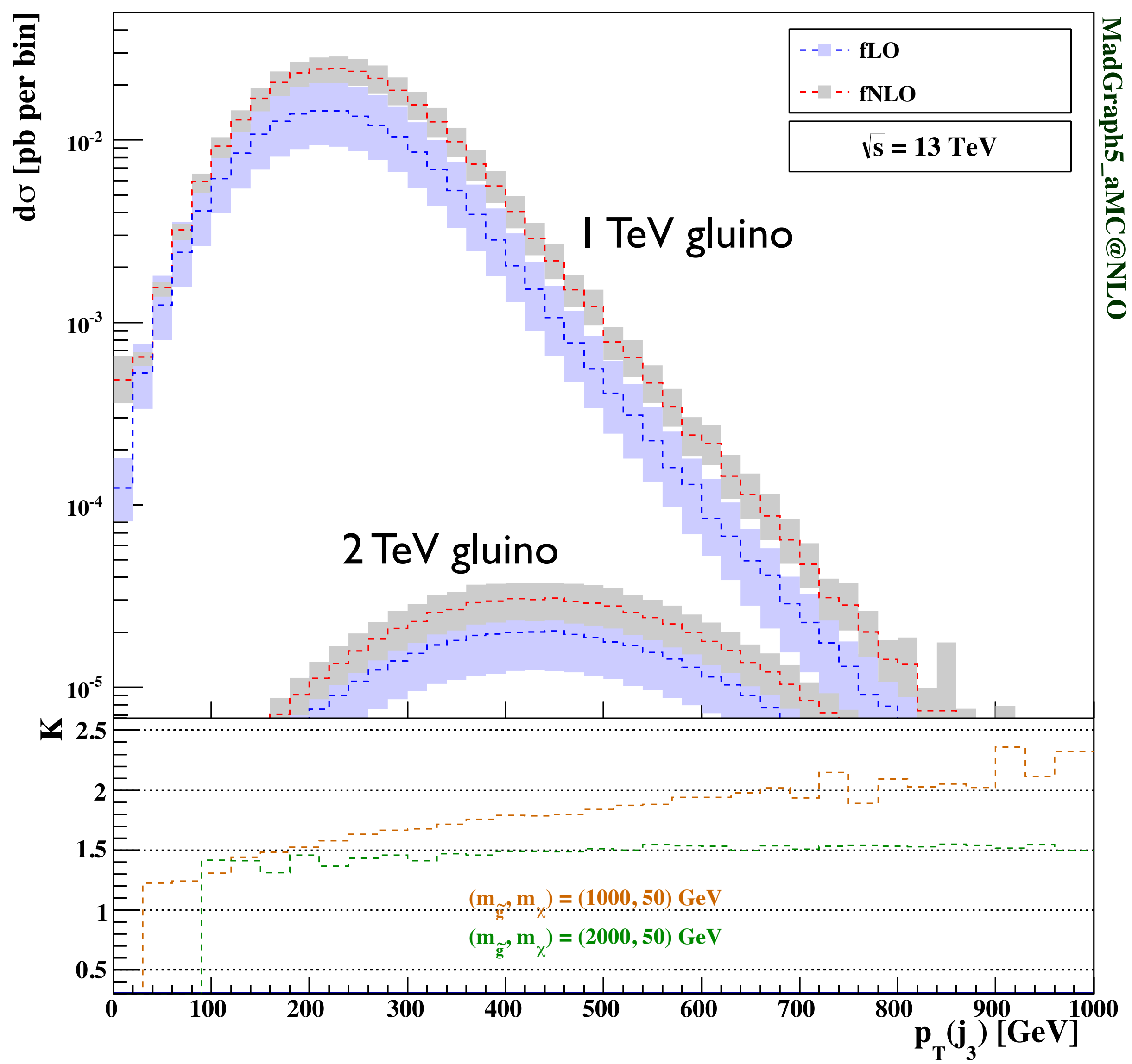
- Decay jets arising from the massive gluino decays: hard
- Radiation jets: rather soft

Behavior of the 3rd jet

Differential distributions at the fixed order

[Degrande, BF, Hirschi, Proudom & Shao (PRD'15; PLB'16)]

3rd jet p_T

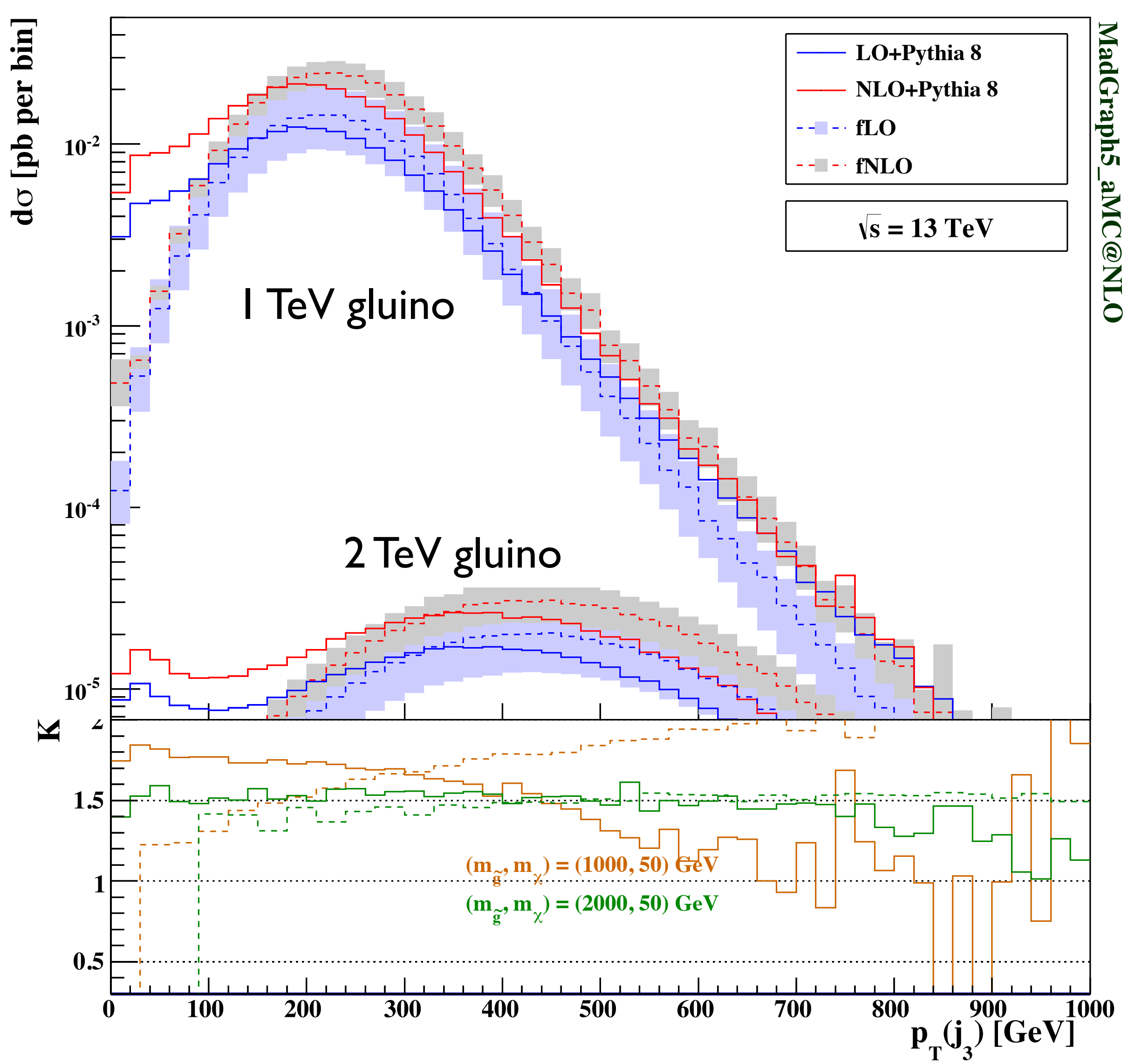


- Origin of the third jet
  - ★ Sometimes a decay jet (hard)
  - ★ Sometimes a radiation jet (soft)
    - ➢ Activity in the low-p_T region
- Constant *K*-factors not accurate
  - ★ In particular in the small p_T region
- NLO effects
  - ★ Crucial for a precise signal description
    - ➢ Normalization enhancement
    - ➢ Distortion of the shapes
  - ★ Reduction of the theoretical uncertainties

Differential distributions (ME+PS)

[Degrande, BF, Hirschi, Proudom & Shao (PRD'15; PLB'16)]

3rd jet p_T



- Parton showers populate the low-p_T region
  - ★ Emitted partons often not reclustered back
    - ➢ Extra softer jets
  - ★ Distortion of the spectrum
  - ★ Effects milder for hard p_T (the matrix element drives the shape)
- Mixed effects: origin of the third jet
  - ★ Two peaks

Outline

1. New physics & Monte Carlo simulations

2. Model implementations

3. Cascade decays

4. Towards precision: merging and NLO corrections

5. **Conclusions - summary**

Summary

✦ Streamlining the links between models and simulations

Idea / Lagrangian → UFO → Simulated collisions → Event analysis

FeynRules
LanHEP
SARAH

Matrix Elements

Shower
Hadronization

Detector
Reconstruction

Analysis codes

✦ Implementation of any theory in MC tools is straightforward (LO and NLO)

✦ Many efforts have been invested in the simulations for new physics

- Model implementations
- Handling the heavy particle decays
- Description of the jet activity

✦ Can we reverse the chain (LHC recasting)?

See the review of
Eric Conte

See talk by Wolfgang
Waltenberger

See talk by
Pat Scott