# Simulating new physics at colliders

**Fuks Benjamin**

**LPTHE / UPMC**

**1st MadAnalysis 5 workshop on LHC recasting**
**@ High 1, Gangwon Province, Korea**

**August 20 - 28, 2017**

# Outline

# Standard Model simulations: the status

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

- ✤ Automated matrix element generation (MADGRAPH5, 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.*)

✦ 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

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

**What are the ingredients behind this success?**

# Outline

# A Monte Carlo tool framework for new physics

## ✦ Streamlining the chain from physics models to analyzed simulated collisions

- ✤ Need for a framework…
  - ★ … where any new physics model can be implemented
  - ★ … where any new physics model can be tested against data
  - ★ … easy to validate, to maintain
  - ★ … easily integrable in a software chain

## ✦ 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) ]

# Automating new physics simulations

✦ First steps towards a new physics simulation framework: LANHEP

✤ Restricted to the CALCHEP / COMPHEP environment

✤ Working environment: C

[ Semenov (NIMA'97; CPC'98; CPC'09; CPC'16) ]
[ Boos *et al.* (IJMPC'94; NIMA'04) ]
[ Belyaev, Christensen & Pukhov (CPC'13) ]

✦ FEYNRULES: a platform for new physics model implementations in MC tools

✤ Working environment: MATHEMATICA

★ Flexibility, symbolic manipulations, easy implementation of new methods, *etc.*

✤ Interfaced to many Monte Carlo tools

★ Dedicated translators to several tools (obsolete today thanks to the UFO)

✤ Automatic linking of Lagrangians to files in a given programming language

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

✦ The SARAH package

✤ Working environment: MATHEMATICA

✤ Interfaced to many Monte Carlo tools

✤ Spectrum generator features

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

# Example: FeynRules

## ✦ What is FeynRules?

- ✤ A framework to develop new physics models
- ✤ Automatic export to several Monte Carlo event generators

➡ Facilitate phenomenological investigations of BSM models
➡ Facilitate the confrontation of BSM models to data

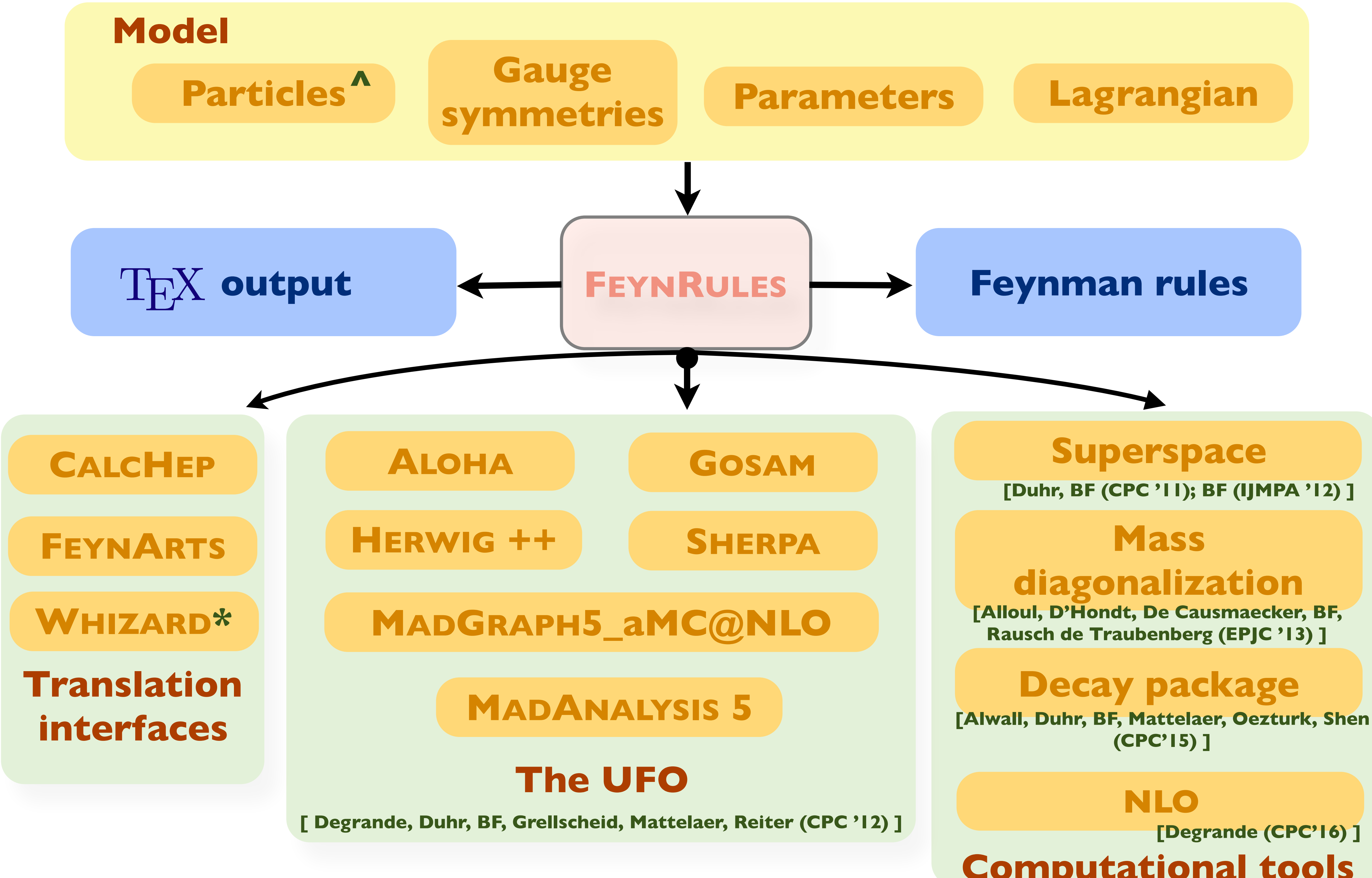
- ✤ Validation of an implementation using several of the linked MC programs

## ✦ Main features

- ✤ Mathematica package
- ✤ Core function: derives Feynman rules from a Lagrangian
- ✤ Requirements: locality, Lorentz and gauge invariance
- ✤ Supported fields: scalar, fermion, vector (+ ghost), spin-3/2, tensor, superfield

# From FEYNRULES to Monte Carlo tools...

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




* Whizard interface: Christensen, Duhr, BF, Reuter, Speckner (EPJC '12)

^ Support for spin 3/2: Christensen, de Aquino, Deutschmann, Duhr, BF, Garcia-Cely, Mattelaer, Mawatari, Oexl, Takaesu (EPJC '13)

# Example: supersymmetric QCD

## ✦ Particle content

### ✤ Matter sector

- ★ One massive Dirac fermion: a quark
- ★ Two massive scalar fields: a left-handed and a right-handed squark

### ✤ Gauge sector

- ★ One massive Majorana fermion: a gluino
- ★ One massless gauge boson: the gluon

## ✦ Lagrangian

$$\begin{aligned}\mathcal{L} = & -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2} \bar{\tilde{g}} \not{D} \tilde{g} + D_\mu \tilde{q}_L^\dagger D^\mu \tilde{q}_L + D_\mu \tilde{q}_R^\dagger D^\mu \tilde{q}_R + i \bar{q} \not{D} q \\ & - m^2_{\tilde{q}_L} \tilde{q}_L^\dagger \tilde{q}_L - m^{2R}_{\tilde{q}_L} \tilde{q}_R^\dagger \tilde{q}_R - m_q \bar{q} q - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g} \\ & - \frac{g_s^2}{2} \Big[ - \tilde{q}_L^\dagger T^a \tilde{q}_L \pm \tilde{q}_R^\dagger T^a \tilde{q}_R \Big] \Big[ - \tilde{q}_L^\dagger T^a \tilde{q}_L + \tilde{q}_R^\dagger T^a \tilde{q}_R \Big] \\ & + \sqrt{2} g_s \Big[ - \tilde{q}_L^\dagger T^a \big( \bar{\tilde{g}}^a P_L q \big) + \big( \bar{q} P_L \tilde{g}^a \big) T^a \tilde{q}_R + \text{h.c.} \Big]\end{aligned}$$

- ✤ Kinetic and mass terms for all fields (first two lines)
- ✤ Supersymmetric gauge interactions (last two lines)

# How to write a FEYNRULES model file?

- ✦ A FEYNRULES model file is compliant with the MATHEMATICA syntax
- ✦ It contains:

✤ A preamble
- ★ Author information
- ★ Model information
- ★ Index definitions

✤ The declaration of the gauge group
- ★ Abelian or not
- ★ Representation matrices
- ★ Structure constants
- ★ Coupling constant
- ★ Gauge boson or vector superfield

✤ The declaration of the fields
- ★ Names, spins, PDG codes
- ★ Indices, quantum numbers
- ★ Masses, widths
- ★ Classes and class members

✤ The declaration of the parameters
- ★ External and internal
- ★ Scalar and tensor

✤ A Lagrangian

**See the manual for more details**

# Gauge groups

## ✦ Declaration in the *M$GaugeGroups list*

- ✤ A declaration ≡ a set of MATHEMATICA replacement rules
  - ★ SUSY-QCD: $SU(3)_C$
- ✤ Each rule represents one group property
  - ★ *Abelian*: abelian or non-abelian gauge group
  - ★ *GaugeBoson*: the associated gauge boson
  - ★ *CouplingConstant*: the coupling constants
  - ★ *StructureConstant*: the structure constants
  - ★ *Representation*: list of 2-tuples linking an index (*Colour*) to the related representation (*T*)

```
M$GaugeGroups = {
  SU3C == {
    Abelian            -> False,
    GaugeBoson         -> G,
    CouplingConstant   -> gs,
    StructureConstant  -> f,
    Representations    -> { {T,Colour} }
  }
};
```

## ✦ Advantages of a proper gauge group declaration

- ✤ Render the writing of the Lagrangian easier:
  - ★ Covariant derivatives (*DC[sq, mu]*)
  - ★ Field strength tensors (*FS[G, mu, nu, a]*)

# Fields

- Declaration in the *M$ClassesDescription* list
  - A declaration ≡ a set of MATHEMATICA replacement rules
    - The gluon example: *G*
  - Each rule ≡ a property of the field
    - Vector field ➢ the label is V[1] (with V)
    - Classname: symbol to use in the Lagrangian (*G*)
    - SelfConjugate: own antiparticle (*True*)
    - Indices: the gluon lies in the adjoint representation of SU(3)c
      - ➢ The gluon has been previously set as the gauge boson of $SU(3)_c$
      - ➢ Its index (*Gluon*) is internally linked to the adjoint representation of the group
    - Other properties: vanishing mass and widths, PDG code set to 21

```
M$ClassesDescription = {
  V[1] == {
    ClassName       -> G,
    SelfConjugate   -> True,
    Indices         -> {Index[Gluon]},
    Mass            -> 0,
    Width           -> 0,
    PDG             -> 21
  },
  …
```

# Declaring mixing squark fields

✦ Extra elements in the *M$ClassesDescription* list

```
S[1] == {
  ClassName        -> sqL,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Unphysical       -> True,
  Definitions      -> {sqL[c_] -> Cos[theta] sq1[c] - Sin[theta] sq2[c]}
},
S[2] == {
  ClassName        -> sqR,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Unphysical       -> True,
  Definitions      -> {sqR[c_] -> Sin[theta] sq1[c] + Cos[theta] sq2[c]}
},
```

```
S[3] == {
  ClassName        -> sq1,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Mass             -> {Msq1,300},
  Width            -> {Wsq1,10},
  PDG              -> 1000006
},
S[4] == {
  ClassName        -> sq2,
  SelfConjugate    -> False,
  Indices          -> {Index[Colour]},
  Mass             -> {Msq2,800},
  Width            -> {Wsq2,2},
  PDG              -> 2000006
}
```

✤ Squark fields mix:

$$\begin{pmatrix} \tilde{q}_1 \\ \tilde{q}_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \tilde{q}_L \\ \tilde{q}_R \end{pmatrix}$$

★ Left and right-handed squarks are declared as unphysical

★ Definitions linking gauge- and mass-eigenstates are provided

➢ The rotations will be performed automatically by FeynRules

➢ The Lagrangian can be written in the gauge basis (easier)

# Parameter declaration

- **The model contains three parameters:**
  - The strong coupling constants: $g_s$, $\alpha_s$ are both needed (required by the MC tools)
  - The squark mixing angle $\theta$
  - Masses and widths are handled automatically

- **Declaration in the list *M$Parameters***
  - A declaration ≡ a set of replacement rules
  - Each rule ≡ a property of the parameters
    - ★ Internal and External parameters
      - ➢ External ≡ free parameter ⇒ numerical value
      - ➢ Internal ≡ dependent parameter ⇒ formula
    - ★ InteractionOrder: specific to MG5_AMC (more efficient diagram generation)
    - ★ ParameterName: specific to Monte Carlo tools
    - ★ Other options exist (but unused)

```
M$Parameters = {
  aS     == {
    ParameterType    -> External,
    Value            -> 0.1184,
    InteractionOrder -> {QCD, 2}
  },
  gs == {
    ParameterType    -> Internal,
    Value            -> Sqrt[4 Pi aS],
    InteractionOrder -> {QCD, 1},
    ParameterName    -> G
  },
  theta == {
    ParameterType    -> External,
    Value            -> Pi/4.
  }
};
```

# Implementing the vector Lagrangian

- For the sake of the example: restriction to the gauge content of the theory
- The dynamics of the model is embedded in the vector Lagrangian

$$\mathcal{L}_{\rm vector} = -\frac{1}{4} g_{\mu\nu} g^{\mu\nu} + \frac{i}{2} \bar{\tilde{g}} \not{D} \tilde{g} - \frac{1}{2} m_{\tilde{g}} \bar{\tilde{g}} \tilde{g}$$

- The implementation in FEYNRULES is easy

```
LVector1 := -1/4 FS[G,mu,nu,a] FS[G,mu,nu,a] +
            I/2 gobar.Ga[mu].DC[go,mu] -
            1/2 Mgo gobar.go;
```

# Feynman rules and FeynRules

✦ Extract all N-point interactions from the Lagrangian (with N>2)

```
In[14]:= FeynmanRules[LVector1, ScreenOutput → True];

Starting Feynman rule calculation.
Expanding the Lagrangian...
Collecting the different structures that enter the vertex.
3 possible non-zero vertices have been found -> starting the computation: 3 / 3.
3 vertices obtained.
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
Vertex 1
Particle 1 : Vector , G
Particle 2 : Vector , G
Particle 3 : Vector , G
Vertex:
```

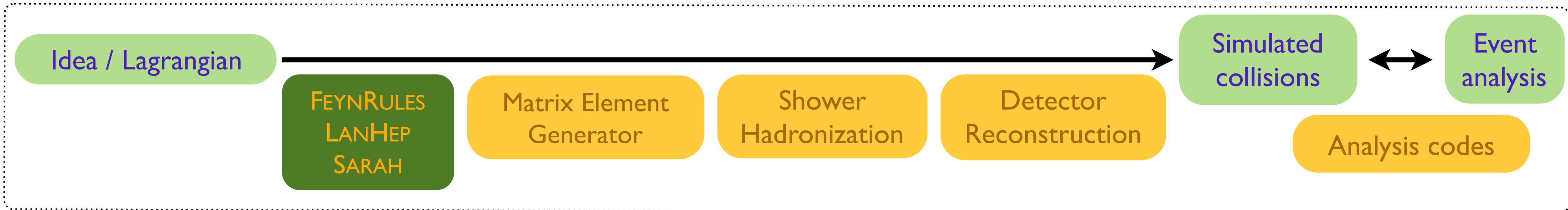
$$gs\, f_{a_1,a_2,a_3}\, p_1^{\mu_3}\, \eta_{\mu_1,\mu_2} - gs\, f_{a_1,a_2,a_3}\, p_2^{\mu_3}\, \eta_{\mu_1,\mu_2} - gs\, f_{a_1,a_2,a_3}\, p_1^{\mu_2}\, \eta_{\mu_1,\mu_3} + gs\, f_{a_1,a_2,a_3}\, p_3^{\mu_2}\, \eta_{\mu_1,\mu_3} + gs\, f_{a_1,a_2,a_3}\, p_2^{\mu_1}\, \eta_{\mu_2,\mu_3} - gs\, f_{a_1,a_2,a_3}\, p_3^{\mu_1}\, \eta_{\mu_2,\mu_3}$$

- ✤ Three vertices are found
- ✤ The expression above consists of the triple gluon vertex
- ✤ The index $a_i$ is related to the $i^{th}$ particle
- ✤ The index $\mu_i$ is the Lorentz index of the $i^{th}$ (vector) particle

✦ This can then be exported automatically to MC generators

# New physics simulations: other challenges

✦ A comprehensive approach to Monte Carlo simulations




✦ 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**

# A step further: the Universal FEYNRULES Output

✦ Improving the maintenance: the UFO one format to rule them all

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

✦ The UFO in a nutshell

- ✤ 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

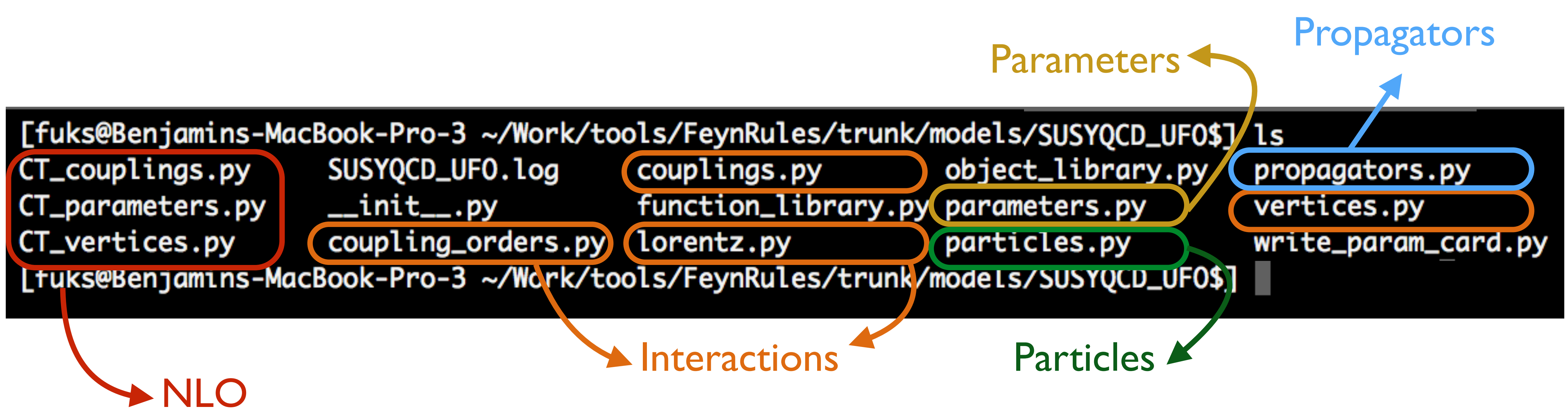
✦ The UFO is now a standard and used by many other programs

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: a UFO for supersymmetric QCD



# 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

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

★ 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)

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

## ✦ 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

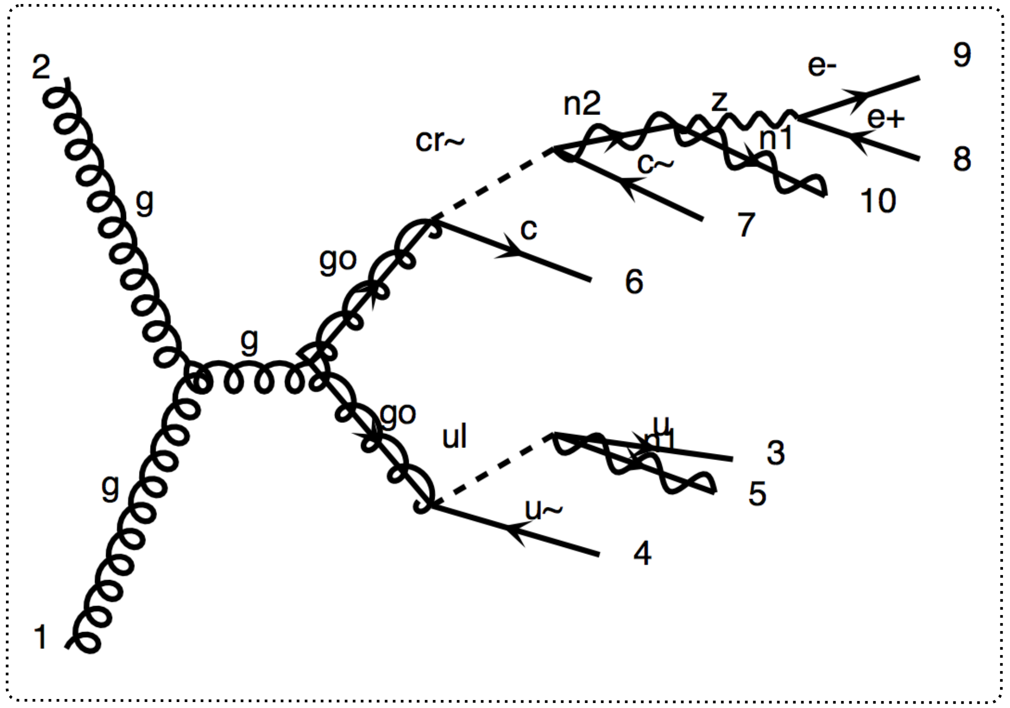
# Outline

1. When new physics meets Monte Carlo simulations
2. FeynRules and the UFO
3. **Cascade decays**
4. Merging and next-to-leading order
5. Conclusions - summary

# Do we need cascade decays?

✦ Concrete models

- Many new states to be 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 N large) a problem?

✦ The issue is the computing time

- Matrix element generation is possible
- Computationally challenging
- Practically useless: diagrams with intermediate resonances dominate

# Making decays easy: the key principle

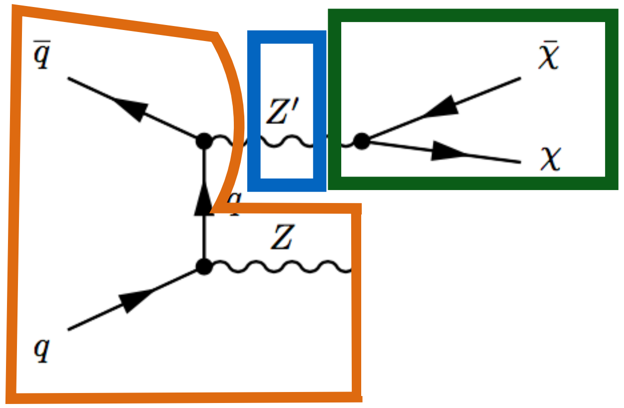
- Production and decay processes are factorized
  - Propagators can be seen as sums of products of external wave functions
  - Example for a vector resonance

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

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

## ✦ Case 2: including spin correlations

✤ Helicity sums performed after accounting for production and decays

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

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

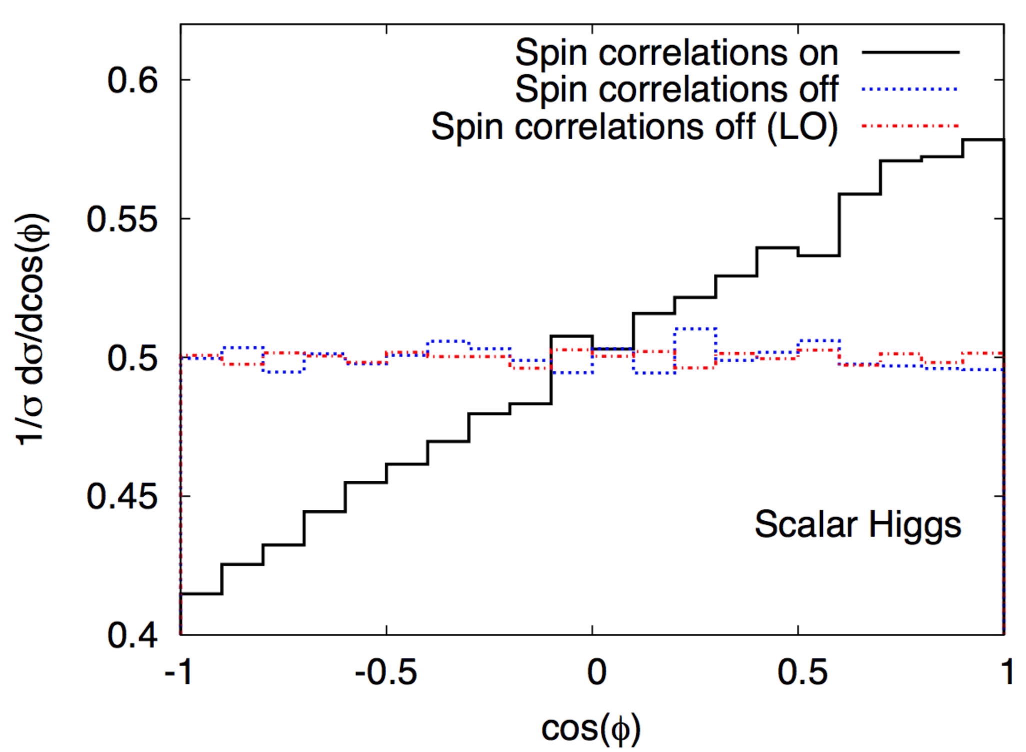
**HERWIG** [ Richardson (JHEP '01) ]

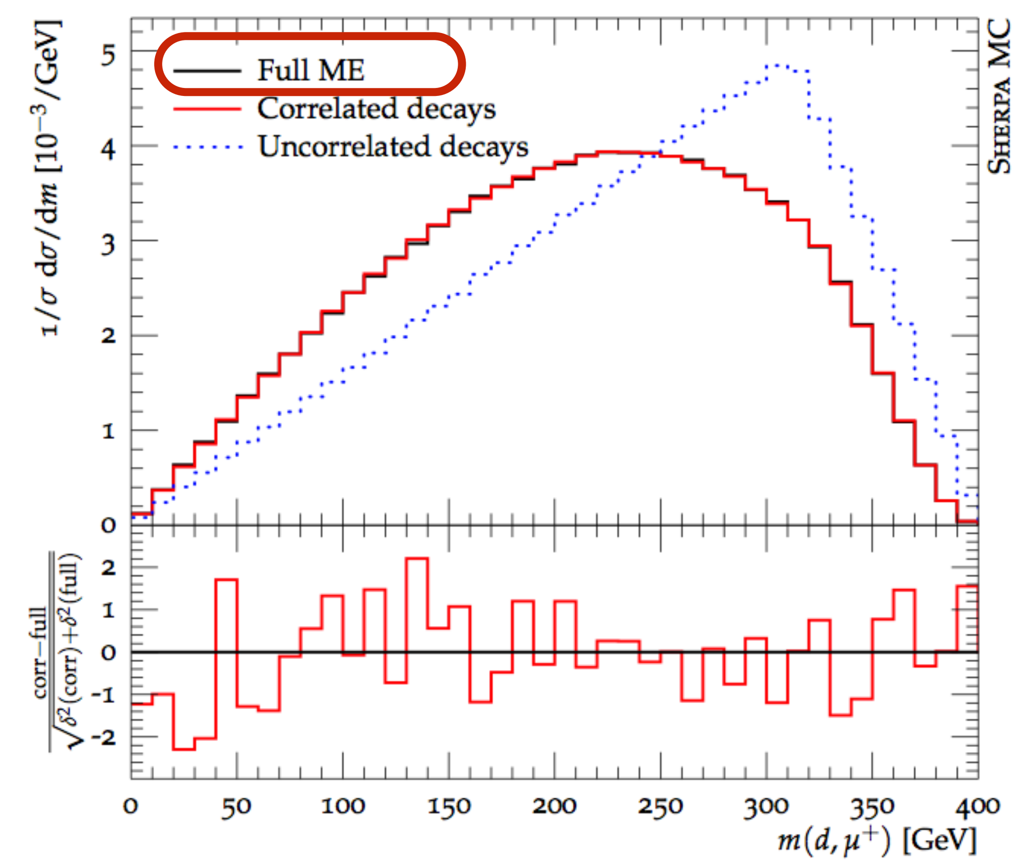
**PYTHIA 8** [Sjostrand, *et al.* (CPC '08) ]

**SHERPA** [Höche *et al.* (EPJC '15) ]

# Importance of correctly handling decays

✦ Is a correct decay handling important: yes!



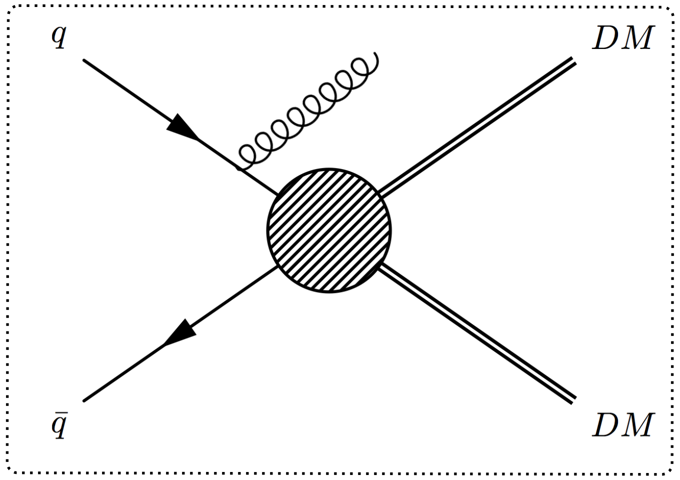

MADSPIN
$t\bar{t}H$ production @ (N)LOQCD
[ LHC8, dileptonic $t\bar{t}$ decay]

SHERPA
squark pair production @ LO
[ LHC8, $\tilde{u} \to d\,\chi_1^+ \,[\to \chi_1^0\, W^+\, [\to \mu^+\, \nu_\mu]]$ $\tilde{u}^* \to \bar{u}\,\chi_2^0\,[\to e^+\,\tilde{e}_R^-\,[\to e^-\,\chi_1^0]]$ ]

# Outline

# Modeling of extra QCD emissions

- ✦ Initial (and final) state radiation modeling is crucial
  - ✤ Monojet-based dark matter searches
  - ✤ Compressed spectra searches
  - ✤ *etc.*
- ✦ Radiation can be predicted in different ways




- ✦ Matrix-element and parton-shower predictions are complementary
  - ✤ Both can be combined
- ✦ Other option: NLO calculations
  - ✤ Correct modeling of the first emission
  - ✤ Merging of samples with different jet multiplicities also possible

# NLO calculations in a nutshell

## ✦ Contributions to an NLO result in QCD

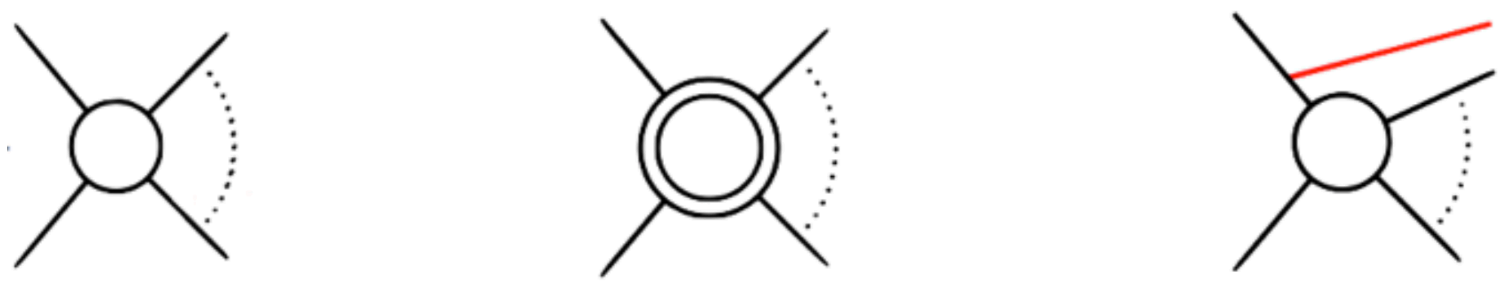
✤ Three ingredients: the Born, virtual loop and real emission contributions

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

Born

Virtuals: one extra power of $\alpha_s$ and divergent

Reals: one extra power of $\alpha_s$ and divergent



✤ Challenge: computing predictions numerically (and in four dimensions)

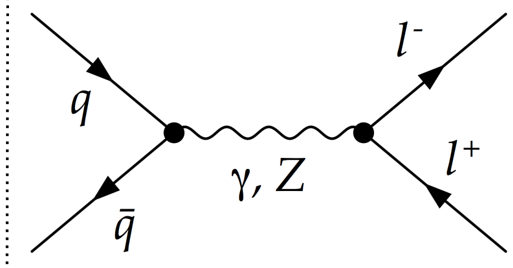
★ The MADGRAPH5_aMC@NLO solution

# Fixed-order predictions

✦ Leading-order (LO): $d\sigma \approx d\sigma^{(0)}$

- Very naive
  - ★ Rough estimate for many observables (large uncertainties)
  - ★ Cannot be used for any observable (e.g., dilepton $p_T$)

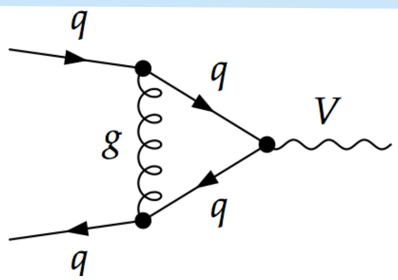
The Drell-Yan example



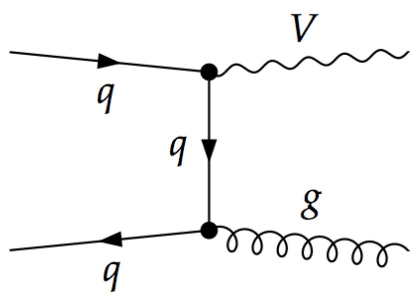

✦ Next-to-leading-order (NLO): $d\sigma \approx d\sigma^{(0)} + \alpha_s\, d\sigma^{(1)}$

- Two divergent components: virtuals and reals
  - ★ Their sum is finite (KLN theorem)
- Reduction of the theoretical uncertainties
  - ★ Loops compensate trees
- Better description of the process
  - ★ Impact of extra radiation, more initial states
  - ★ Sometimes not enough

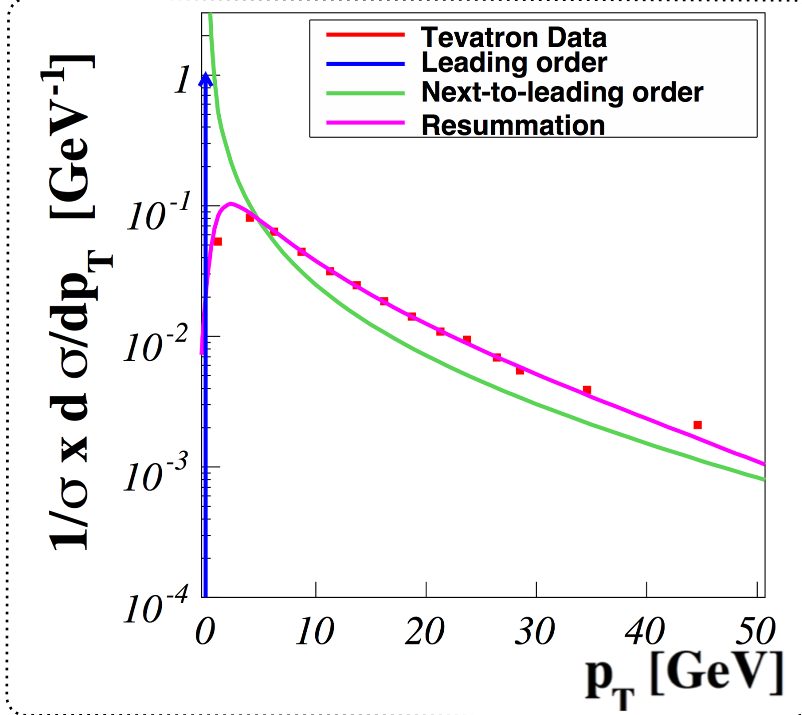
The Drell-Yan example:
Representative virtual




Representative real

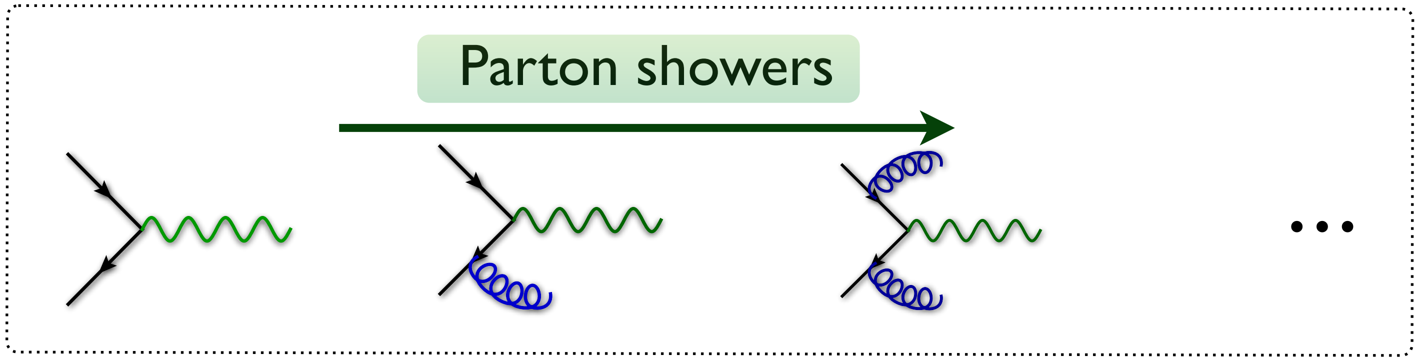


# Matrix-element / parton shower matching




- ✦ Problems with NLO (fixed-order) calculations
  - ✤ Soft and collinear radiation ➢ large logarithms
  - ✤ Spoil the convergence of the perturbative series

- ✦ Matching with parton showers




  - ✤ Resummation of the soft and collinear radiation
  - ✤ Predictions for a fully exclusive description of the collisions
  - ✤ Suitable for going beyond the parton level (hadronization, detector simulation)

# Virtual contributions

## ✦ Loop diagram calculations

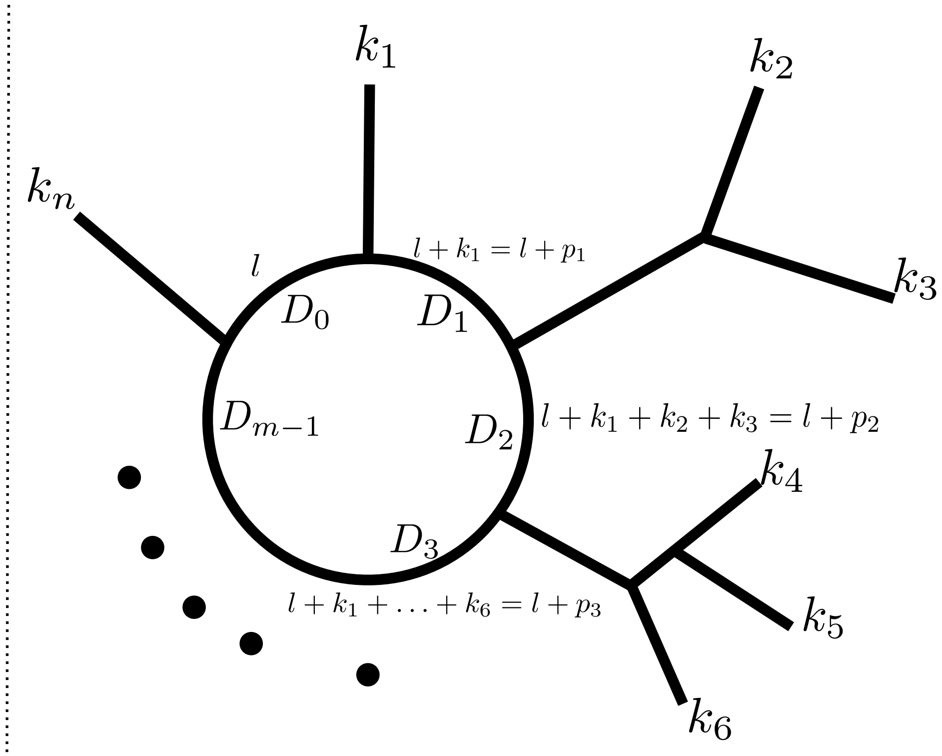
- ✤ Calculations to be done in $d=4-2\varepsilon$ dimensions
  - ★ Divergences made explicit ($1/\varepsilon^2$, $1/\varepsilon$)
  - ★ Numerical challenge
- ✤ Rewriting loop integrals with scalar integrals

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

  - ★ Involves integrals with up to four denominators
  - ★ The decomposition basis is finite

The basis integrals can be calculated once and for all

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



# The rational terms

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

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

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

- **The $R_1$ terms originate 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)

- **The $R_2$ terms originate from the numerator**
  - Process-dependent contributions proportional to $\tilde{\ell}^2$
  - In a renormalizable theory, there is a finite number of such $R_2$ pieces
    - ★ They can be calculated once and for all for a specific model (➢ NLOCT) [ Degrande (CPC'15) ]
      - ➢ *$R_2$ counterterm Feynman rules*

# Matching fixed order with parton showers

## ✦ Subtracting the poles

✤ The structure of the poles appearing at NLO is known ➢ subtraction methods

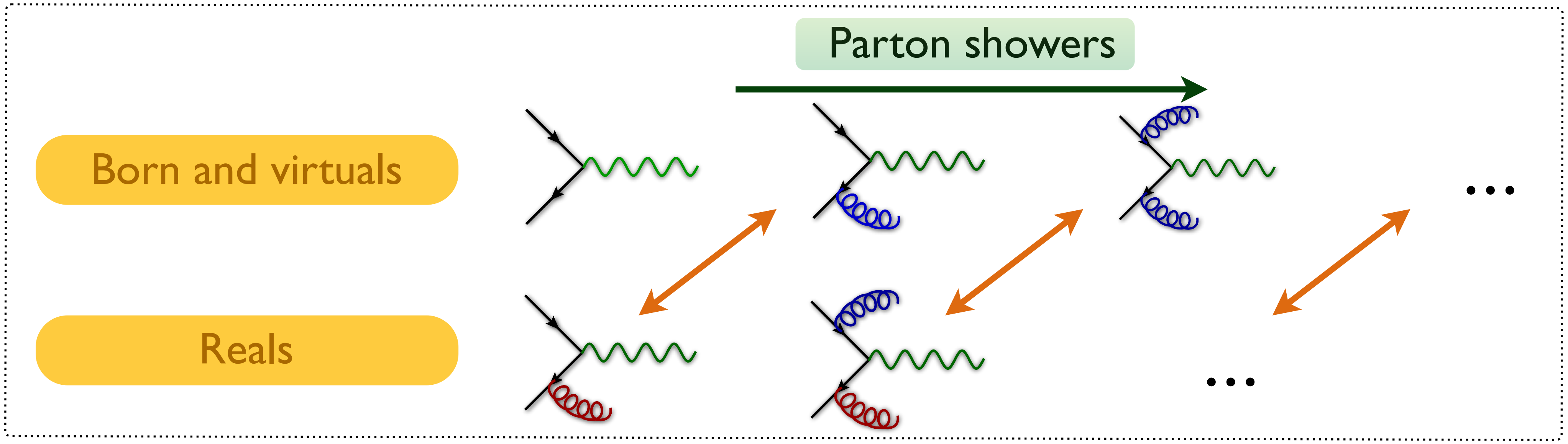
★ $\mathcal{C}$ subtracted from the reals ➢ makes them finite

★ $\mathcal{C}$ integrated and added back to the virtuals ➢ makes them finite

★ Integrals can be made numerically (in four dimensions)

$$\sigma_{NLO} = \int \mathrm{d}^4\Phi_n \, \mathcal{B} + \int \mathrm{d}^4\Phi_{n+1} \left[\mathcal{R} - \mathcal{C}\right] + \int \mathrm{d}^4\Phi_n \left[\int_{\mathrm{loop}} \mathrm{d}^d\ell \, \mathcal{V} + \int \mathrm{d}^d\Phi_1 \mathcal{C}\right]$$

## ✦ Double counting when matching with parton showers: another subtraction



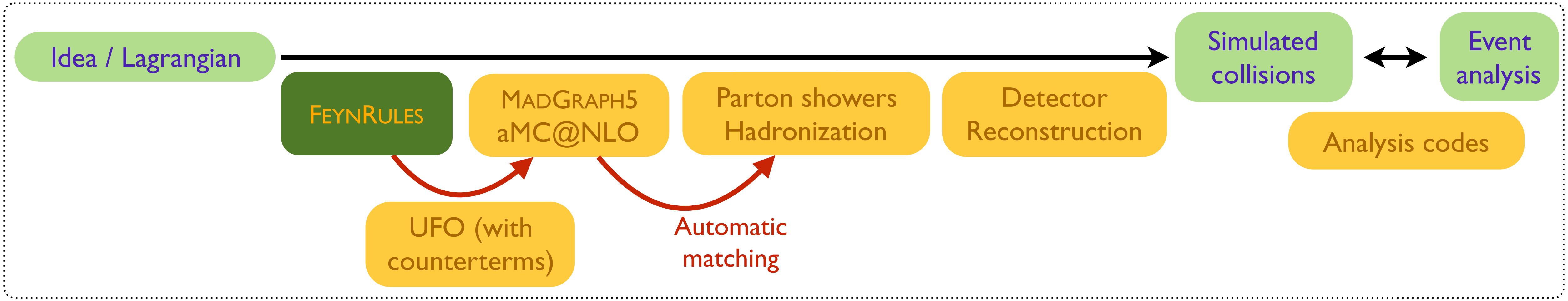

✤ Two sources of double counting that compensate each other (shower unitarity)

★ Radiation: both at the level of the reals and of the shower

★ No radiation: both in the virtuals and in the no-emission probability

# Automated NLO simulations with MG5_aMC

## ✦ A comprehensive approach to Monte Carlo simulations




## ✦ From Lagrangians to analyzed NLO simulated collisions

✤ FeynRules is linked to the NLOCT module [ Alloul, Christensen, Degrande, Duhr & BF (CPC'14) ] [ Degrande (CPC'15) ] [ Degrande, Duhr, BF, Mattelaer & Reither (CPC'12) ] [ Degrande, Duhr, BF, Hirschi, Mattelaer, Shao *et al.* (*in prep.*) ]

- ★ Calculation of UV and $R_2$ counterterms
- ★ Export of the information to the UFO

## ✦ Matching with parton showers within MG5_aMC@NLO

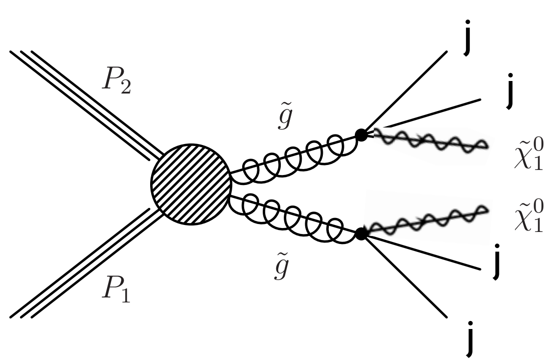
✤ Automatically handled [ Alwall, Frederix, Frixione, Hirschi, Mattelaer, Shao, Stelzer, Torrielli & Zaro (JHEP'14) ]

# 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

Gluino - multijet + MET




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

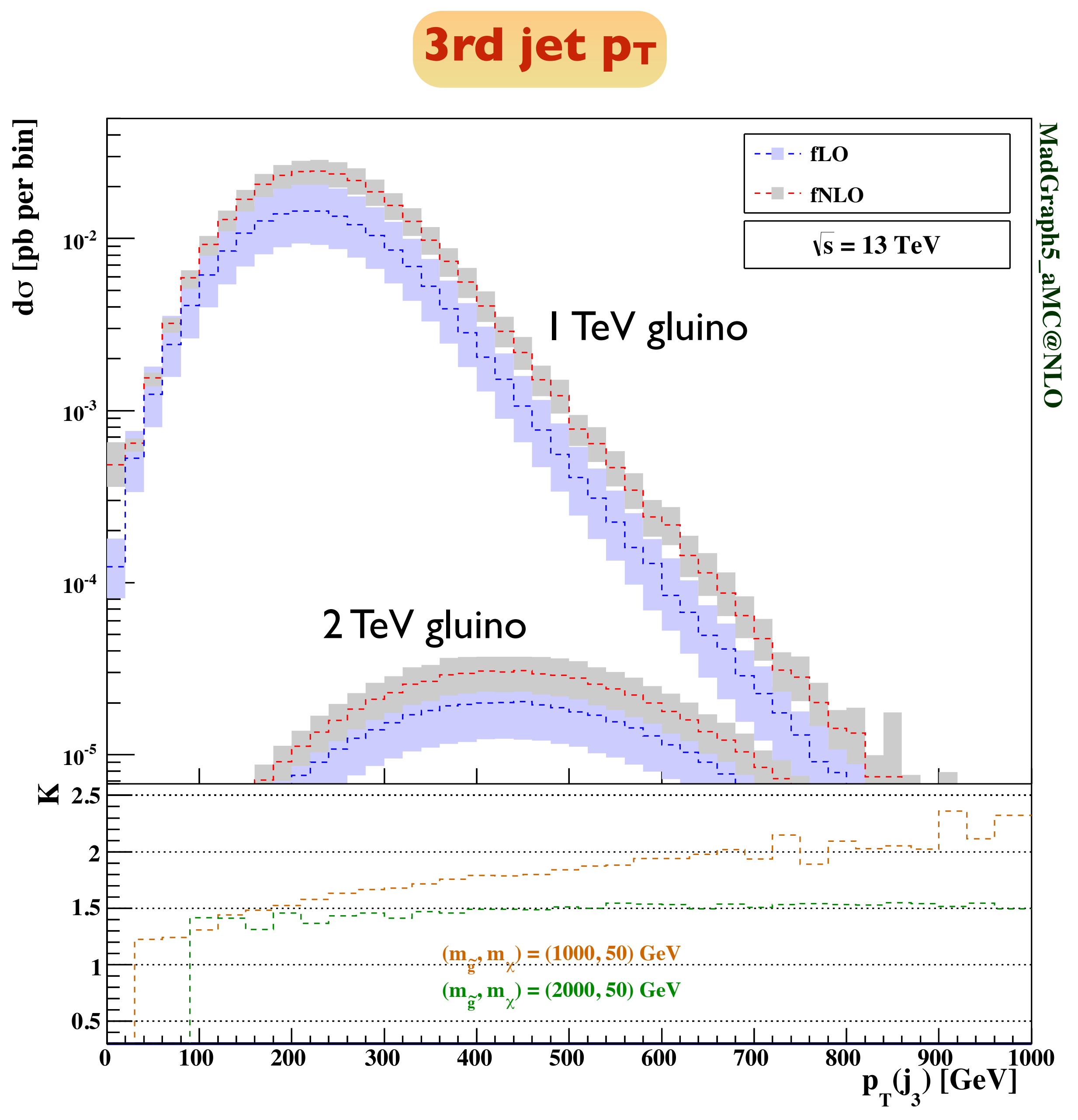
✦ Two types of jets

- ✤ 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) ]



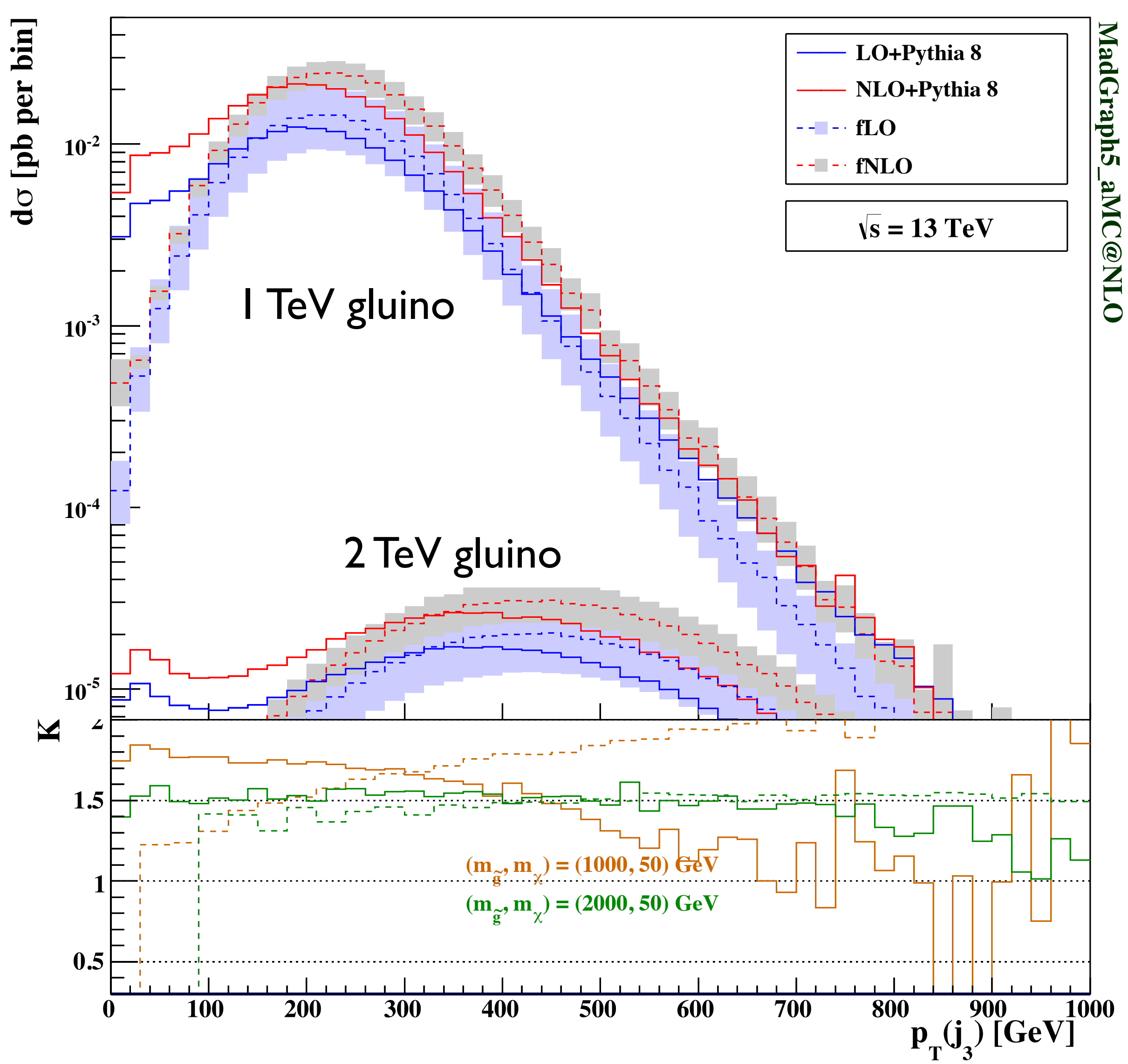

- Mixed effects: origin of the third jet
  - ★ Sometimes a decay jet
  - ★ Sometimes a radiation jet
  - ★ Some activity in the low-$p_T$ region
- Constant *K*-factors not accurate
  - ★ At all for 1 TeV gluinos
  - ★ In the small $p_T$ region for 2 TeV gluinos
- NLO effects
  - ★ Crucial for a precise signal description
    - ➢ Normalization enhancement
    - ➢ Distortion of the shapes
  - ★ Reduction of the theoretical uncertainties

LO description + constant *K*-factor:
➢ very inaccurate signal modelling
➢ in particular in the low-$p_T$ region

# Differential distributions (ME+PS)

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

3rd jet $p_T$



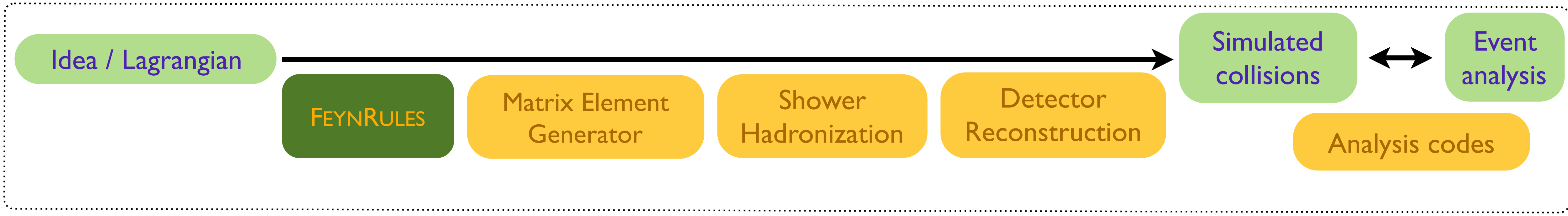

- ✤ Parton showers populate the low-$p_T$ region
  - ★ Emitted partons often not reclustered back
  - ★ Extra softer jets
  - ★ Distortion of the spectrum
  - ★ Effects milder for hard $p_T$ (the matrix element drives the shape)
- ✤ Mixed effects: origin of the third jet
  - ★ Sometimes a decay jet
  - ★ Sometimes a radiation jet
  - ★ Entanglement of the two effects: two peaks
- ✤ *K*-factor behavior (fixed-order *vs.* ME+PS)
  - ★ Changes more pronounced for 1 TeV gluinos
  - ★ Effect at larger $p_T$ for 2 TeV gluinos

# Outline

1. When new physics meets Monte Carlo simulations
2. FEYNRULES and the UFO
3. Cascade decays
4. Merging and next-to-leading order
5. **Conclusions - summary**

# Summary

✦ Streamlining the links between models and simulations




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

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

- Handling the heavy particle decays
- Next-to-leading order corrections (or multipartonic matrix element merging)

✦ Techniques used both by theorists and experimentalists