Monte Carlo's for the LHC

Fabio Maltoni

Centre for Cosmology, Particle Physics and Phenomenology (CP3), BelgiuM

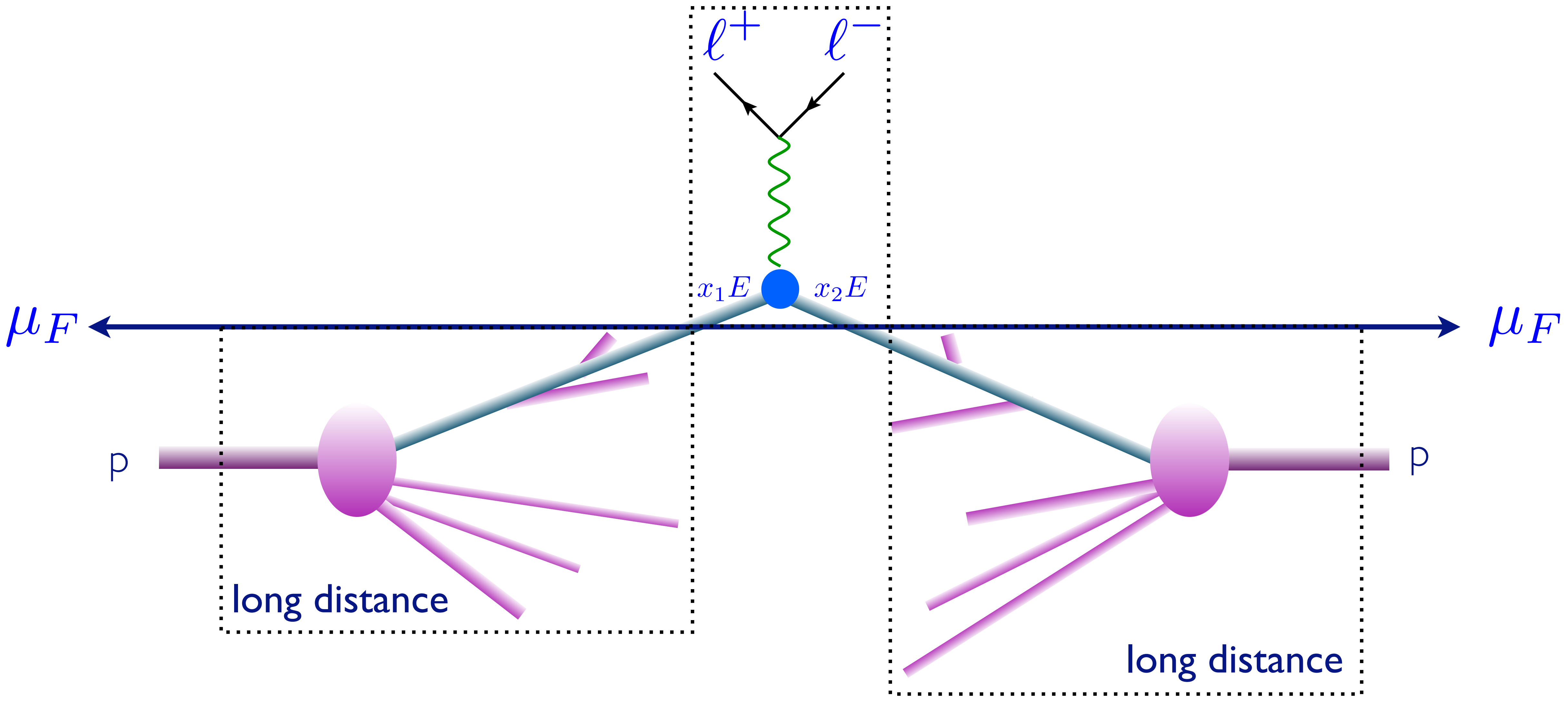
Lecture I

PLAN

- Basics : LO predictions and event generation
- Fixed-order calculations : from NLO to NNLO
- Exclusive predictions : Parton Shower
- Merging ME+PS
- Matching NLO with PS

Today

Master formula for the LHC



$$\sum_{a,b}\int dx_1 dx_2 d\Phi_{\rm FS}\, f_a(x_1,\mu_F) f_b(x_2,\mu_F)\, \hat\sigma_{ab\to X}(\hat s,\mu_F,\mu_R)$$

Phase-space integral

Parton density functions

Parton-level cross section

Master formula for the LHC

$$\sum_{a,b} \int dx_1 dx_2 d\Phi_{\rm FS}\, f_a(x_1,\mu_F) f_b(x_2,\mu_F)\, \hat{\sigma}_{ab\to X}(\hat{s},\mu_F,\mu_R)$$

Phase-space integral

Parton density functions

Parton-level cross section

Two ingredients necessary:

1. Parton distribution functions : non perturbative (fit from experiments, but evolution from theory)

2. Parton-level cross section: short distance coefficients as an expansion in α_S (from theory)

Perturbative expansion

$\hat{\sigma}_{ab\to X}(\hat{s},\mu_F,\mu_R)$ **Parton-level cross section**

- The parton-level cross section can be computed as a series in perturbation theory, using the coupling constant as an expansion parameter

$$\hat{\sigma} = \underbrace{\sigma^{\rm Born}}_{\text{LO predictions}}\left(1 + \underbrace{\frac{\alpha_s}{2\pi}\sigma^{(1)}}_{\text{NLO corrections}} + \underbrace{\left(\frac{\alpha_s}{2\pi}\right)^2\sigma^{(2)}}_{\text{NNLO corrections}} + \underbrace{\left(\frac{\alpha_s}{2\pi}\right)^3\sigma^{(3)}}_{\text{NNNLO corrections}} + \ldots\right)$$

- Including higher corrections improves predictions and reduces theoretical uncertainties

Predictions at LO

How do we calculate a LO cross section for 3 jets at the LHC?

I. Identify all subprocesses (gg→ggg, qg→qgg....) in:

$$\sigma(pp \to 3j) = \sum_{ijk} \int f_i(x_1) f_j(x_2) \hat{\sigma}(ij \to k_1 k_2 k_3)$$

easy

II. For each one, calculate the amplitude:

$$\mathcal{A}(\{p\},\{h\},\{c\}) = \sum_i D_i$$

difficult

III. Square the amplitude, sum over spins & color, integrate over the phase space (D ~ 3n)

$$\hat{\sigma} = \frac{1}{2\hat{s}} \int d\Phi_p \sum_{h,c} |\mathcal{A}|^2$$

quite hard

Phase-space integral

- Calculations of cross section or decay widths involve integrations over phase space of very complex functions

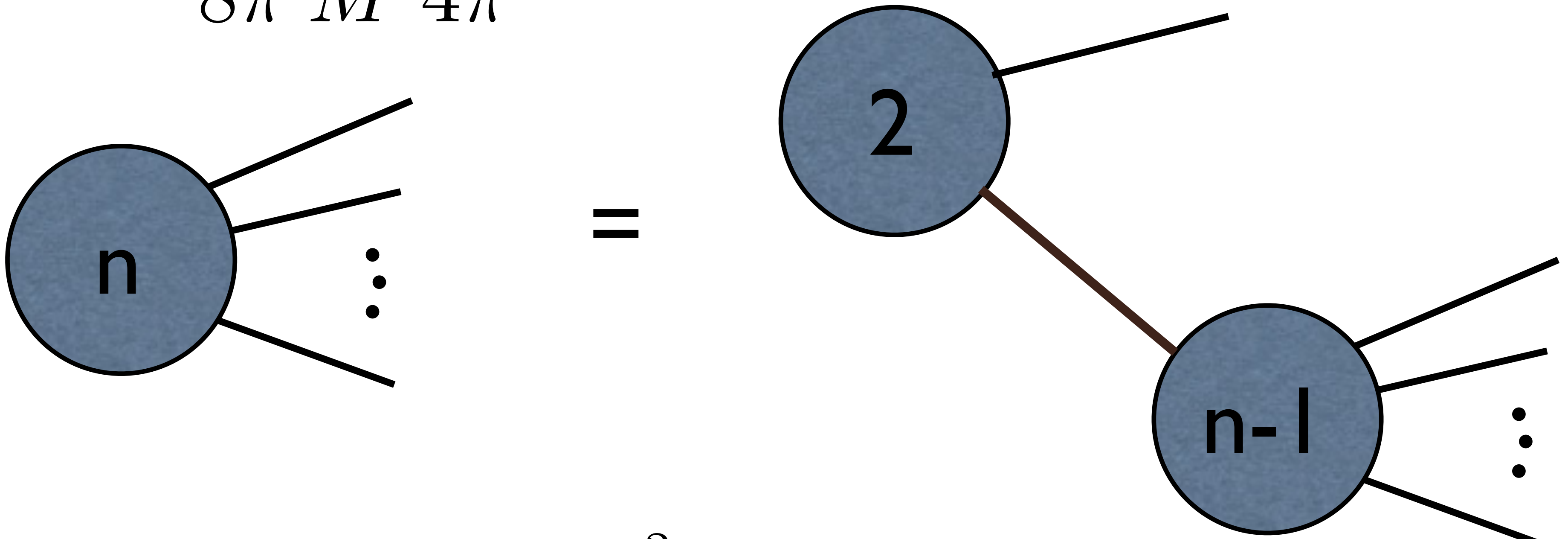
$$\sigma = \frac{1}{2s} \int |\mathcal{M}|^2 d\Phi(n) \quad \longleftarrow \quad Dim[\Phi(n)] \sim 3n$$

General and flexible method is needed:
Numerical (Monte Carlo) integration

Phase-space

$$d\Phi_n = \left[\Pi_{i=1}^n \frac{d^3 p_i}{(2\pi)^3(2E_i)}\right](2\pi)^4\delta^{(4)}(p_0 - \sum_{i=1}^n p_i)$$

$$d\Phi_2(M) = \frac{1}{8\pi}\frac{2p}{M}\frac{d\Omega}{4\pi}$$



$$d\Phi_n(M) = \frac{1}{2\pi}\int_0^{(M-\mu)^2} d\mu^2 d\Phi_2(M) d\Phi_{n-1}(\mu)$$

INTEGRALS AS AVERAGES

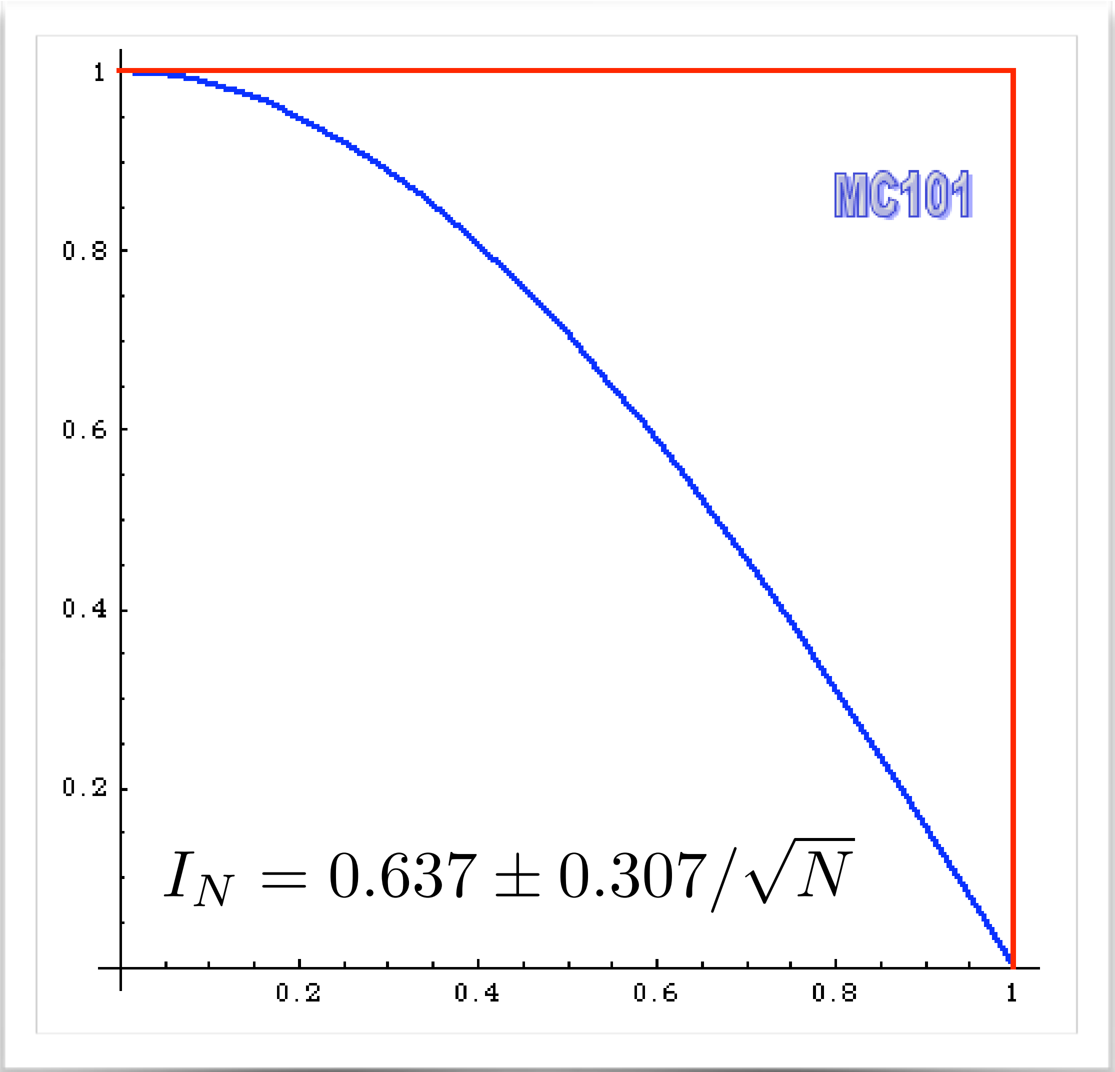
$$I = \int_{x_1}^{x_2} f(x)dx \quad \Rightarrow \quad I_N = (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f(x)$$

$$V = (x_2 - x_1)\int_{x_1}^{x_2} [f(x)]^2 dx - I^2 \quad \Rightarrow \quad V_N = (x_2 - x_1)^2 \frac{1}{N}\sum_{i=1}^{N}[f(x)]^2 - I_N^2$$

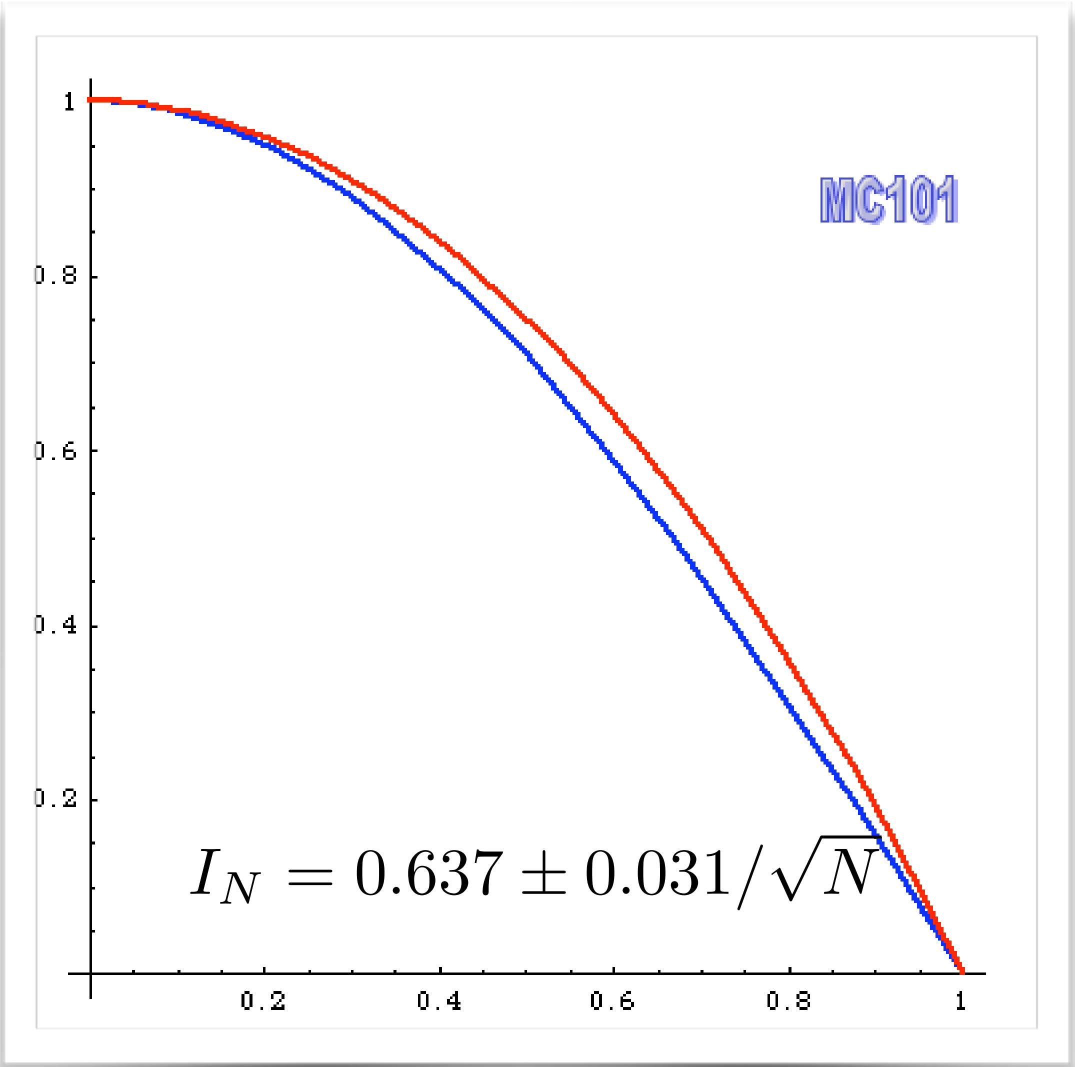
$$I = I_N \pm \sqrt{V_N / N}$$

- ☞ Convergence is slow but it can be estimated easily
- ☞ Error does not depend on # of dimensions!
- ☞ Improvement by minimizing V_N
- ☞ Optimal/Ideal case: f(x) = Constant $\Rightarrow$ $V_N = 0$

Importance sampling



$I_N = 0.637 \pm 0.307/\sqrt{N}$



$I_N = 0.637 \pm 0.031/\sqrt{N}$

$$I = \int_0^1 dx \cos\frac{\pi}{2}x$$

$$I = \int_0^1 dx(1-x^2)\frac{\cos\frac{\pi}{2}x}{1-x^2}$$

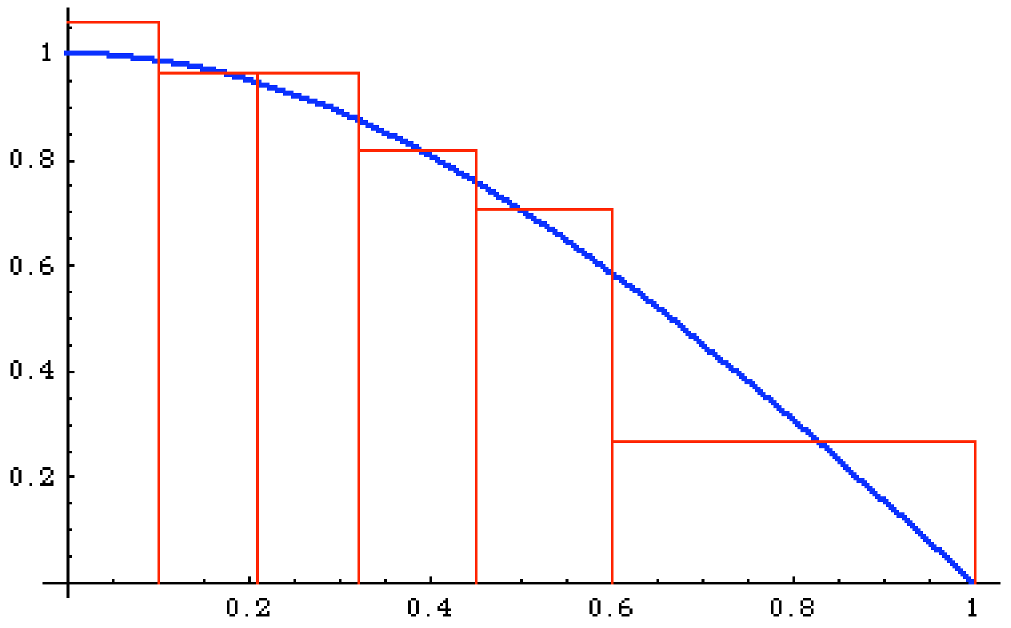
$$= \int_{\xi_1}^{\xi_2} d\xi \frac{\cos\frac{\pi}{2}x[\xi]}{1-x[\xi]^2} \Rightarrow \simeq 1$$

Importance sampling

But... you need to know too much about f(x)!

Idea: learn during the run and build a step-function approximation p(x) of f(x) $\Rightarrow$ VEGAS

MC101



more bins where f(x) is large

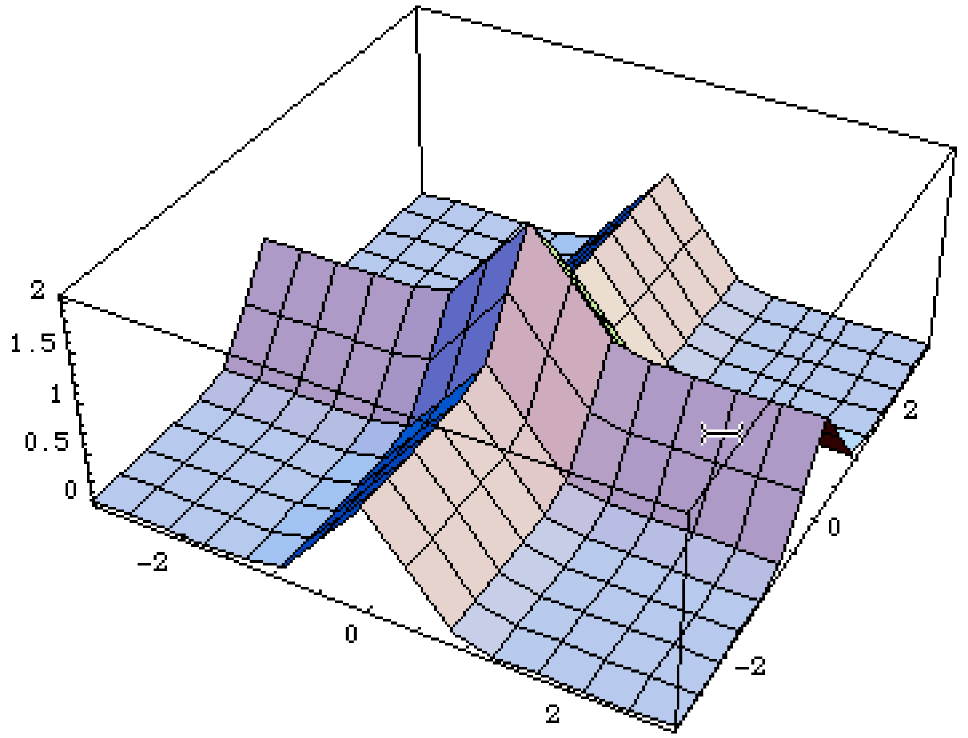
$$p(x) = \frac{1}{N_b \Delta x_i}, \quad x_i - \Delta x_i < x < x_i$$

Importance Sampling

can be generalized to n dimensions:

$$p(\vec{x}) = p(x) \cdot p(y) \cdot p(z) \ldots$$

but the peaks of $f(\vec{x})$ need to be "aligned" to the axis!



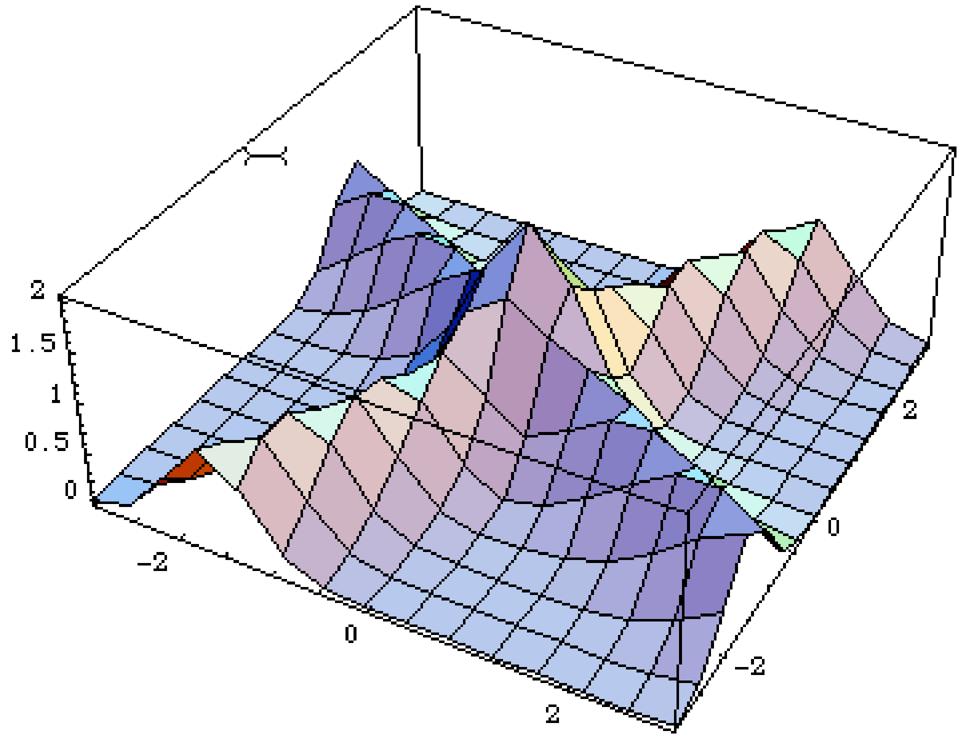
This is ok...

Importance Sampling

can be generalized to n dimensions:

$$p(\vec{x}) = p(x) \cdot p(y) \cdot p(z) \ldots$$

but the peaks of $f(\vec{x})$ need to be "aligned" to the axis!



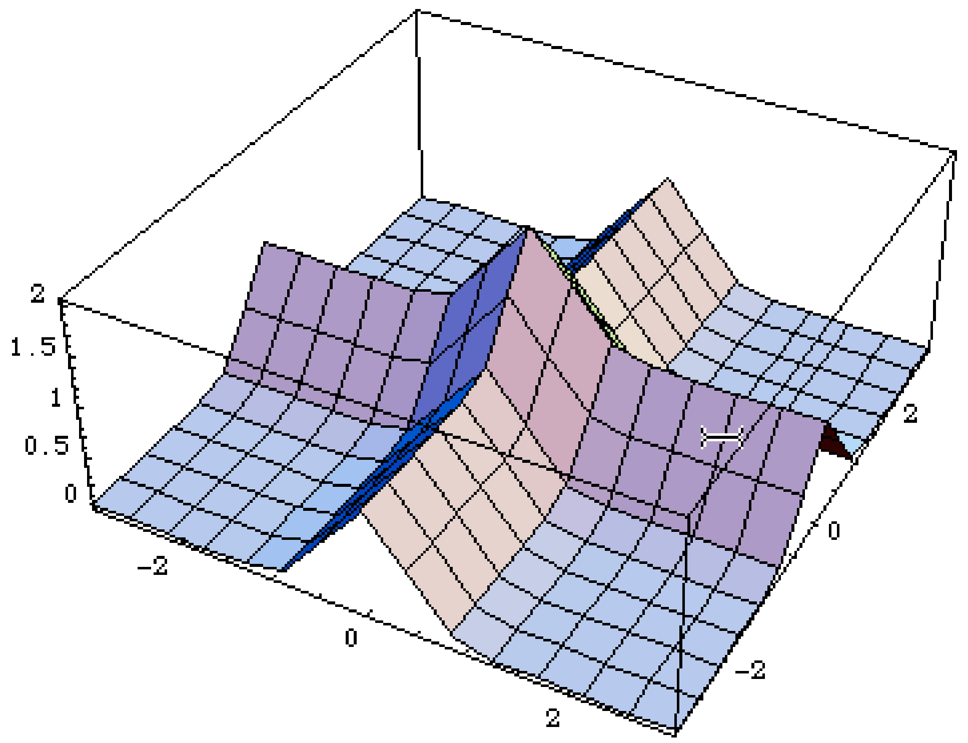
This is not ok...

Importance Sampling

can be generalized to n dimensions:

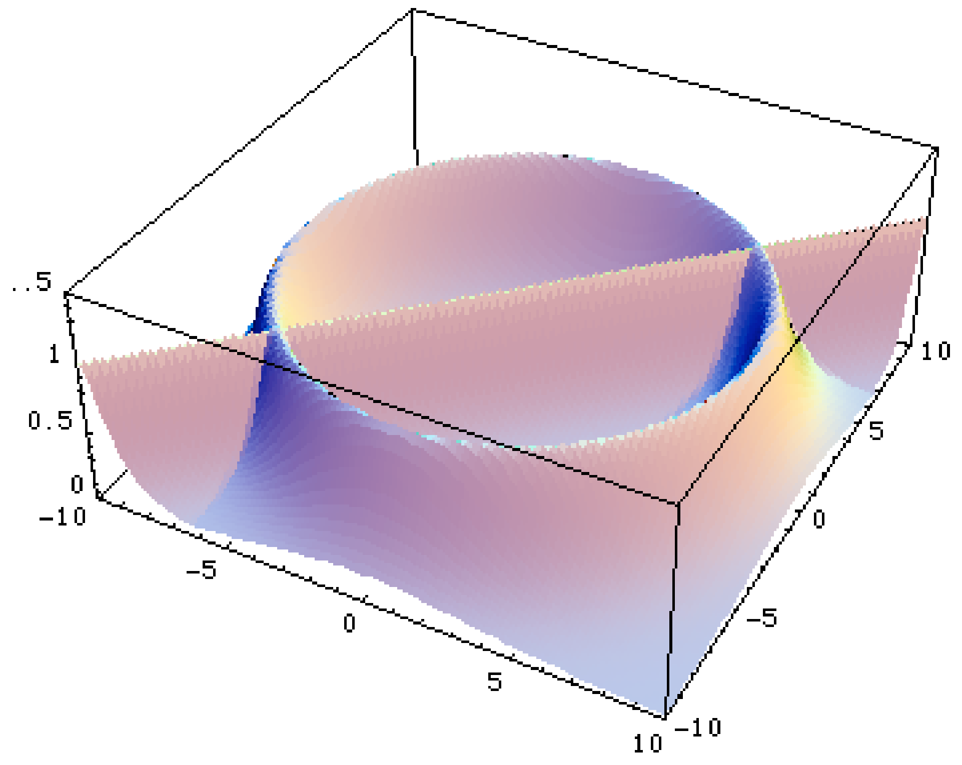
$$p(\vec{x}) = p(x) \bullet p(y) \bullet p(z) \ldots$$

but the peaks of $f(\vec{x})$ need to be "aligned" to the axis!



but it is sufficient to make a change of variables!

MULTI-CHANNEL



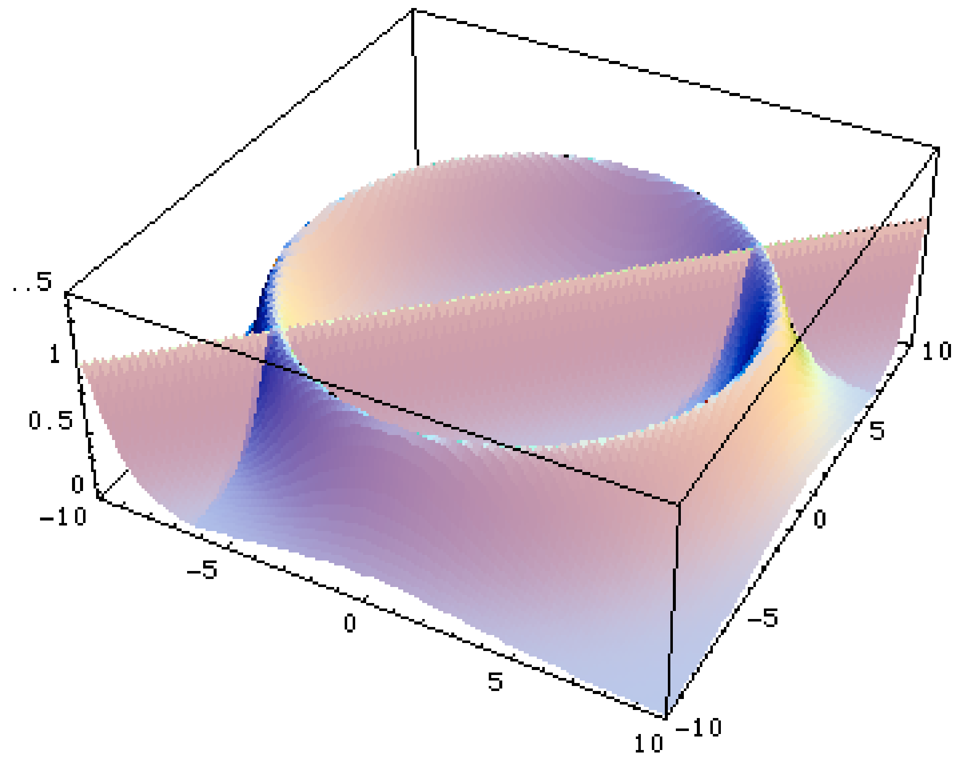
In this case there is no unique tranformation: Vegas is bound to fail!

Solution: use different transformations= channels

$$p(x) = \sum_{i=1}^{n} \alpha_i p_i(x) \qquad \text{with} \qquad \sum_{i=1}^{n} \alpha_i = 1$$

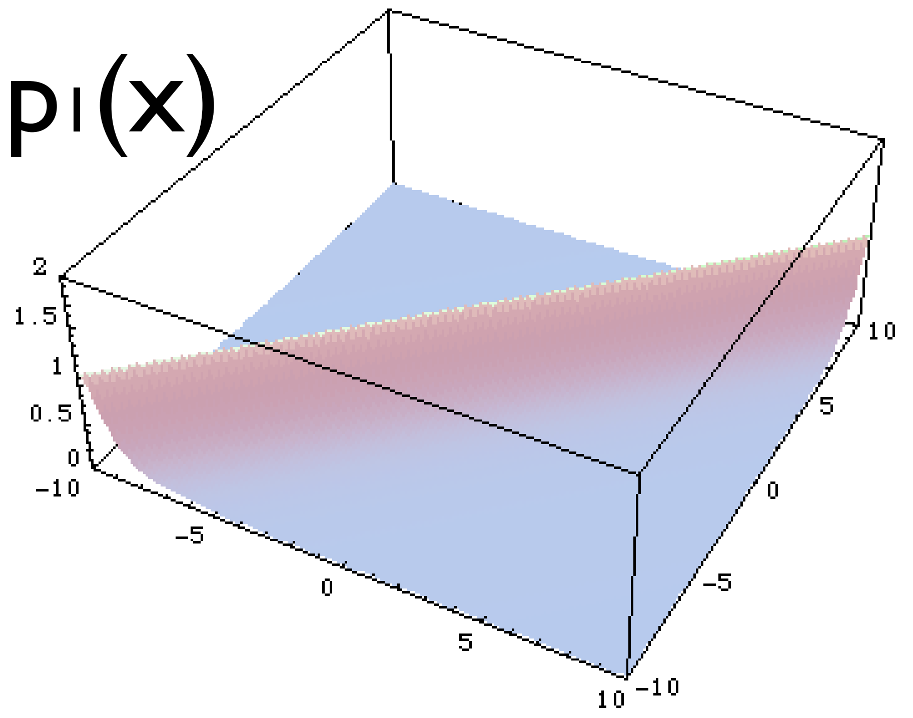
with each $p_i(x)$ taking care of one "peak" at the time

MULTI-CHANNEL

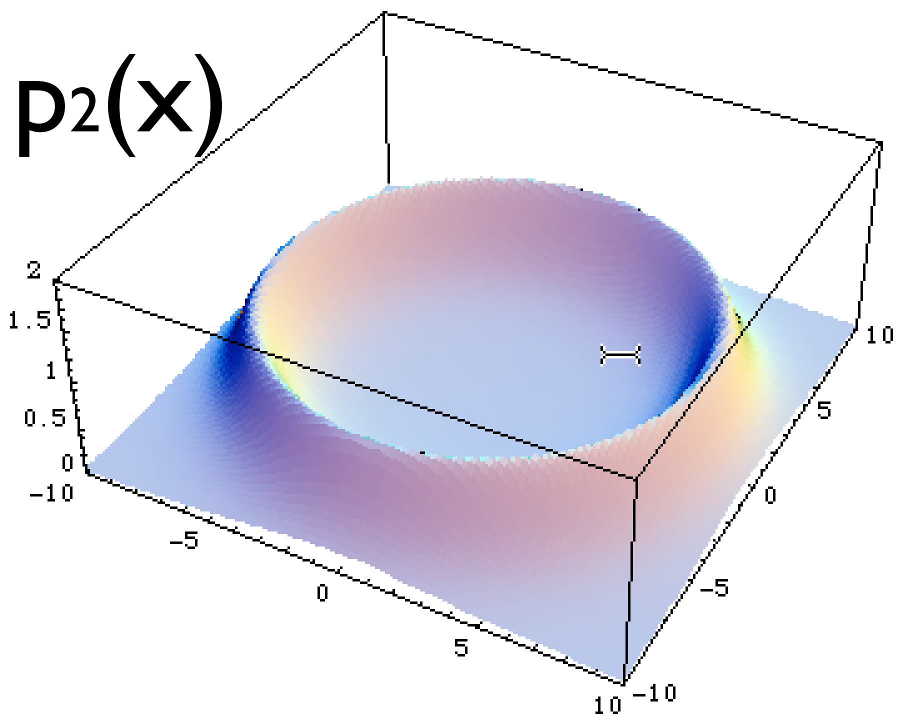


In this case there is no unique tranformation: Vegas is bound to fail!

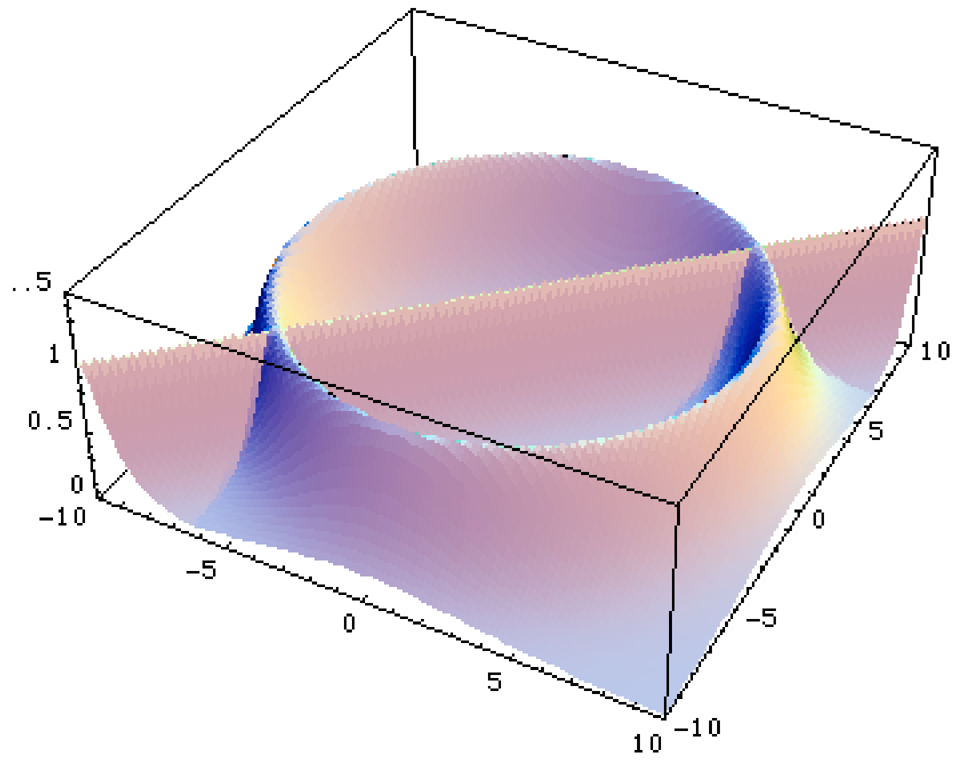
$p_1(x)$



$p_2(x)$



Multi-channel



In this case there is no unique tranformation: Vegas is bound to fail!

But if you know where the peaks are (=in which variables) we can use different transformations= channels:

$$p(x) = \sum_{i=1}^{n} \alpha_i p_i(x) \qquad \textbf{with} \qquad \sum_{i=1}^{n} \alpha_i = 1$$

$$I = \int f(x)dx = \sum_{i=1}^{n} \alpha_i \int \frac{f(x)}{p(x)} p_i(x) dx$$

MULTI-CHANNEL

- Advantages
  - The integral does not depend on the α_i but the variance does and can be minimised by a careful choice
- Drawbacks
  - Need to calculate all g_i values for each point
  - Each phase space channel must be invertible
  - N coupled equations for α_i so it might only work for small number of channels

Very popular method!

Multi-channel based on single diagrams

Consider the integration of an amplitude |M|^2 at treel level which lots of diagrams contribute to. If there were a basis of functions,

$$f = \sum_{i=1}^{n} f_i \quad \text{with} \quad f_i \geq 0\,, \quad \forall \;\; i\,,$$

such that:

1. we know how to integrate each one of them,
2. they describe all possible peaks,

then the problem would be solved:

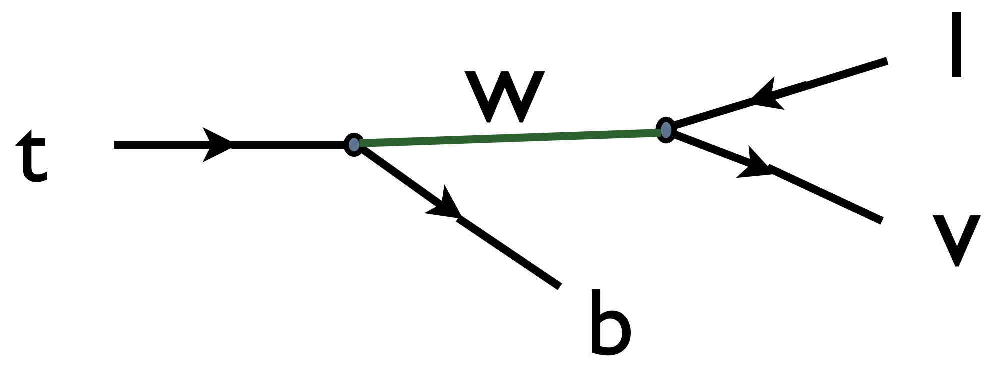
$$I = \int d\vec{\Phi} f(\vec{\Phi}) = \sum_{i=1}^{n} \int d\vec{\Phi}\, g_i(\vec{\Phi})\, \frac{f_i(\vec{\Phi})}{g_i(\vec{\Phi})} = \sum_{i=1}^{n} I_i\,,$$

Does such a basis exist? YES! $f_i = \dfrac{|A_i|^2}{\sum_i |A_i|^2} |A_{\rm tot}|^2$

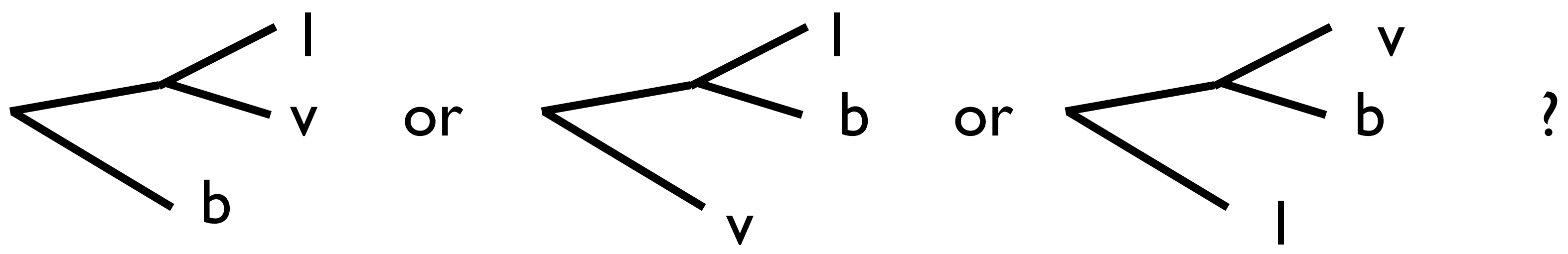
Multi-channel : MadGraph

- Key Idea
  - Any single diagram is "easy" to integrate
  - Divide integration into pieces, based on diagrams
- Get N independent integrals
  - Errors add in quadrature so no extra cost
  - No need to calculate "weight" function from other channels.
  - Can optimize # of points for each one independently
  - Parallel in nature
- What about interference?
  - Never creates "new" peaks, so we're OK!

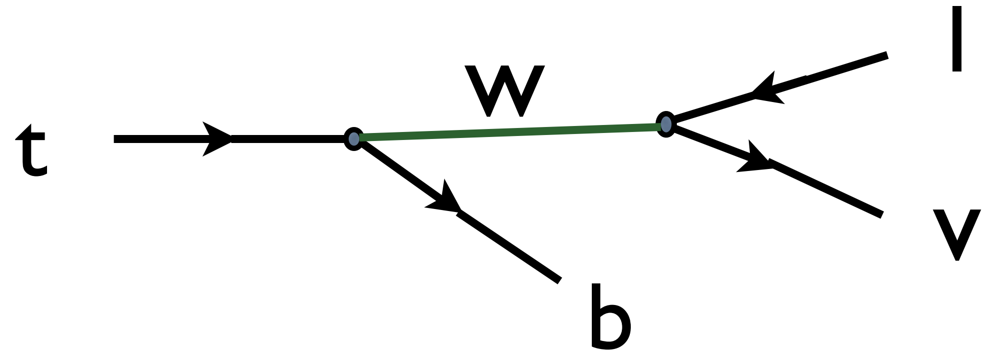
Exercise: Top decay

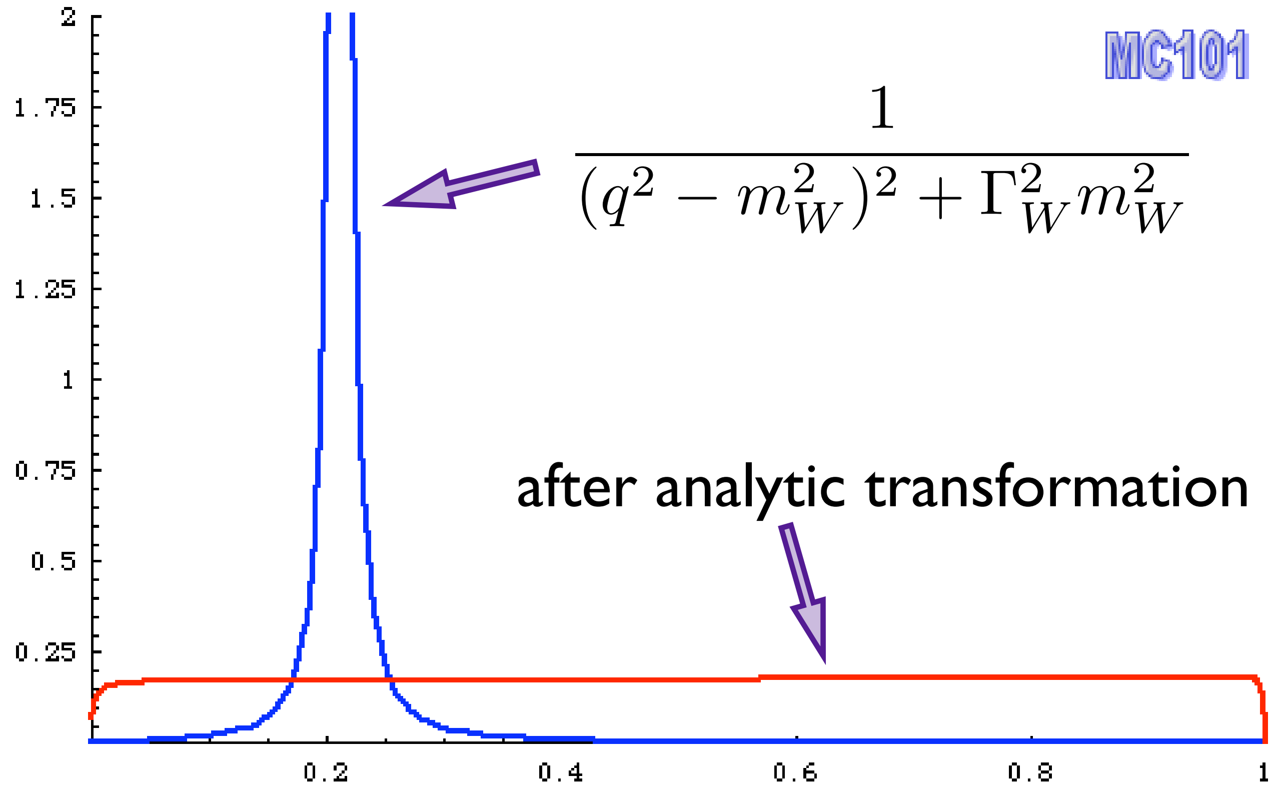


- Easy but non-trivial
- Breit-Wigner peak $$\frac{1}{(q^2 - m_W^2)^2 + \Gamma_W^2 m_W^2}$$ to be "flattened":
- Choose the right "channel" for the phase space:



Exercise: Top decay





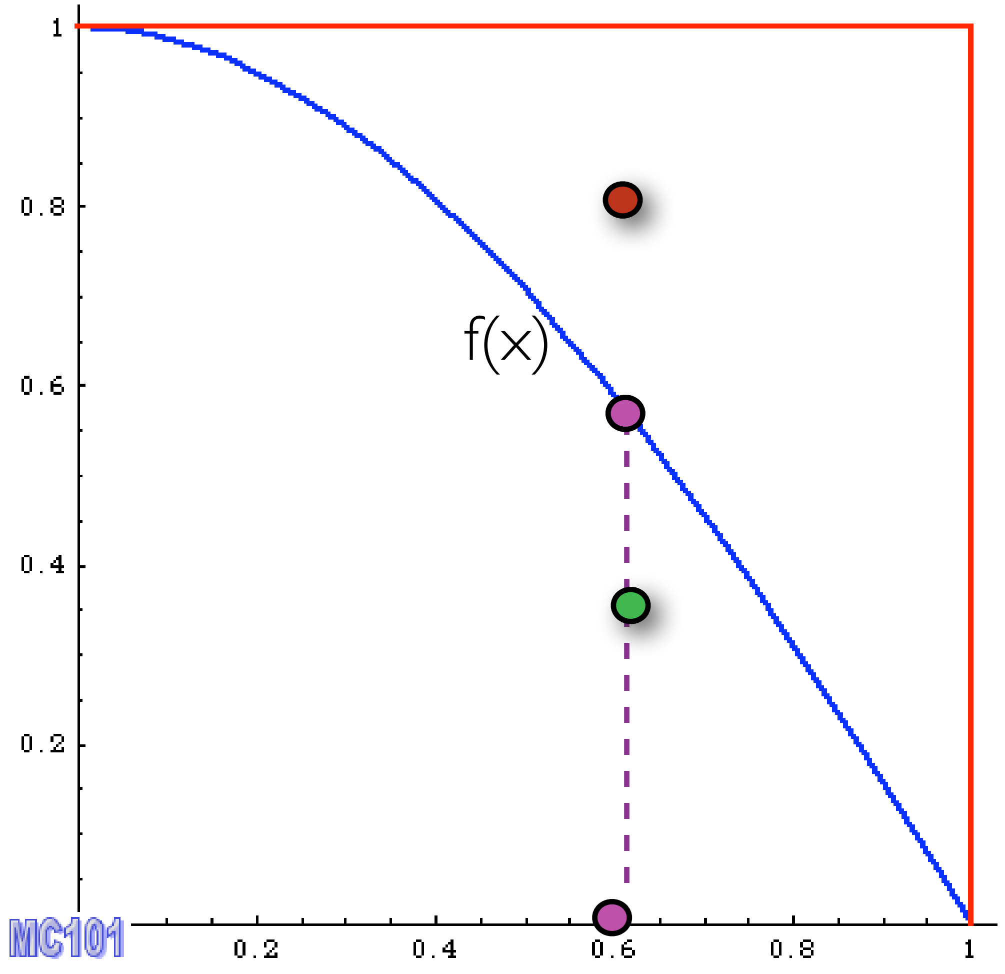
$$\frac{1}{(q^2 - m_W^2)^2 + \Gamma_W^2 m_W^2}$$

after analytic transformation

Event generation

- Every phase-space point computed in this way, can be seen as an event (=collision) in a detector
- However, they still carry the "weight" of the matrix elements:
  - ▻ events with large weights where the cross section is large
  - ▻ events with small weights where the cross section is small
- In nature, the events don't carry a weight:
  - ▻ more events where the cross section is large
  - ▻ less events where the cross section is small
- How to go from weighted events to unweighted events?

Event generation



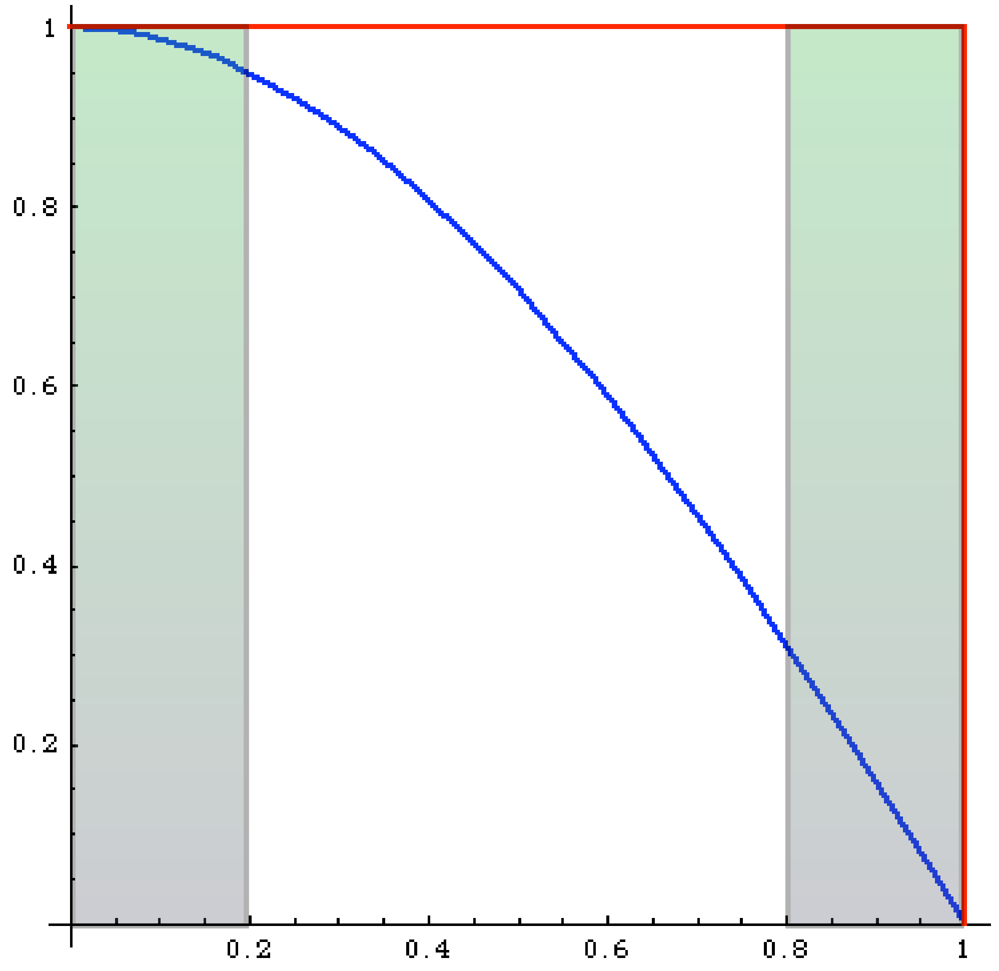
Alternative way

1. (randomly) pick x
2. calculate f(x)
3. (randomly) pick 0<y<fmax
4. Compare:
   if f(x)>y accept event,

   else reject it.

$$\text{Integral} = \frac{\text{accepted}}{\text{total tries}} = \text{efficiency}$$

Event generation



What's the difference?

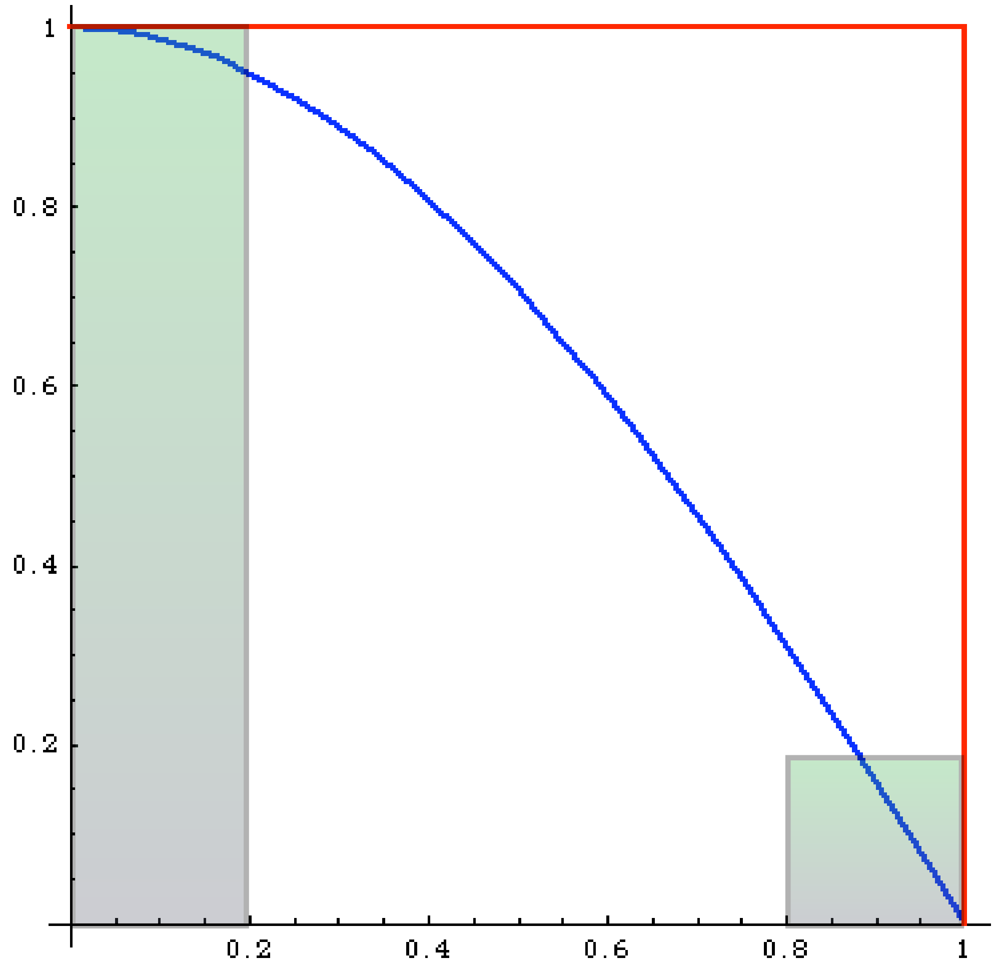
before:

Same # of events in areas of phase space with very different probabilities:

Events must have different weights:

$$w_i = p(x_i)$$

Event generation



What's the difference?

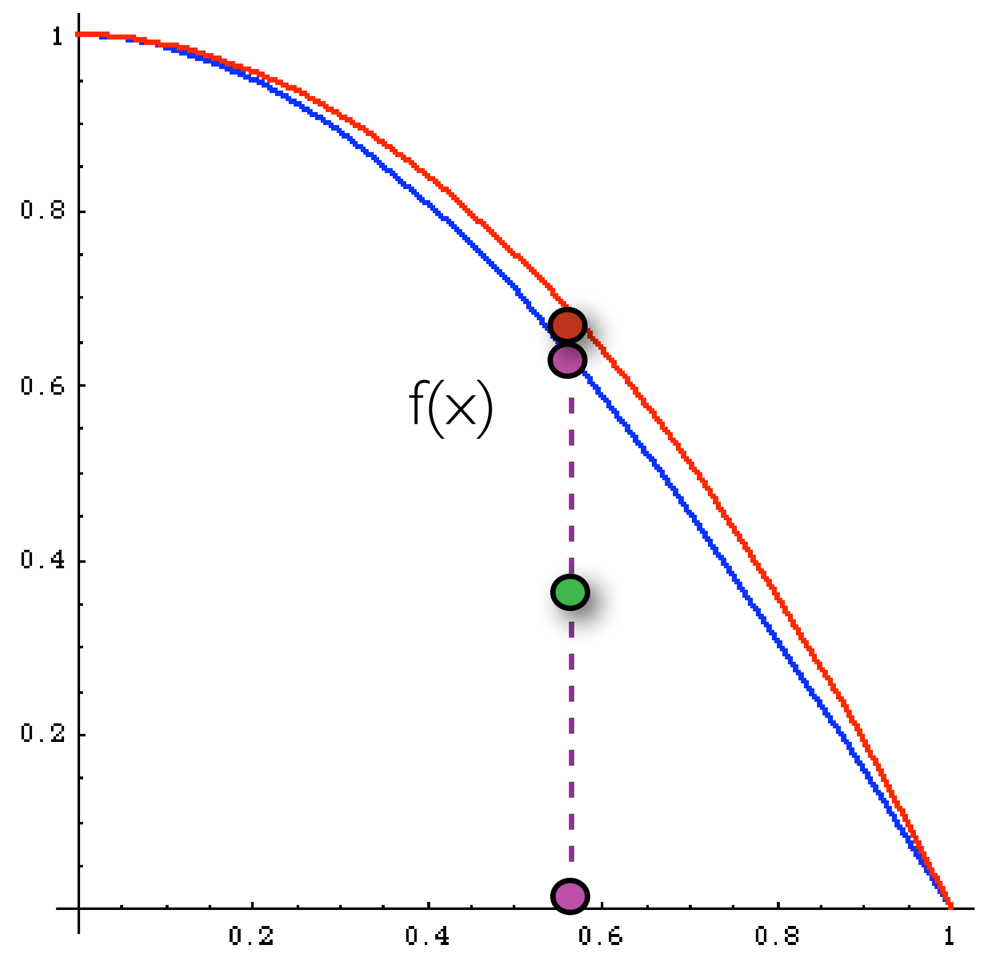
after:

# events is proportional to the probability of areas of phase space:

Events have all the same weight ("unweighted")

Events distributed as in Nature

Event Generation



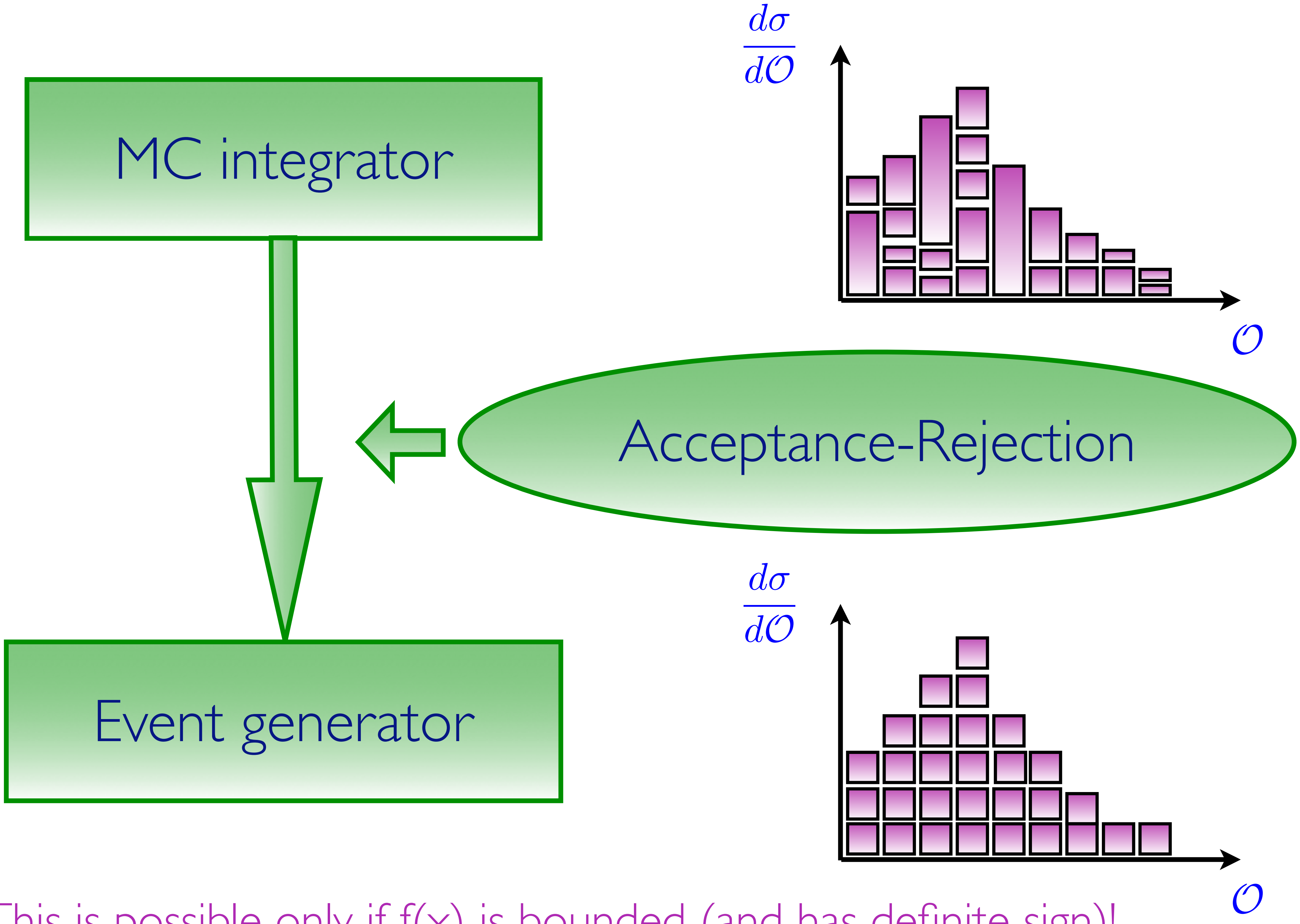
Improved

1. pick x distributed as p(x)
2. calculate f(x) and p(x)
3. pick 0<y<1
4. Compare:
   if f(x)>y p(x) accept event,

else reject it.

much better efficiency!!!

Event generation



☞ This is possible only if f(x) is bounded (and has definite sign)!

MC Event generator: definition

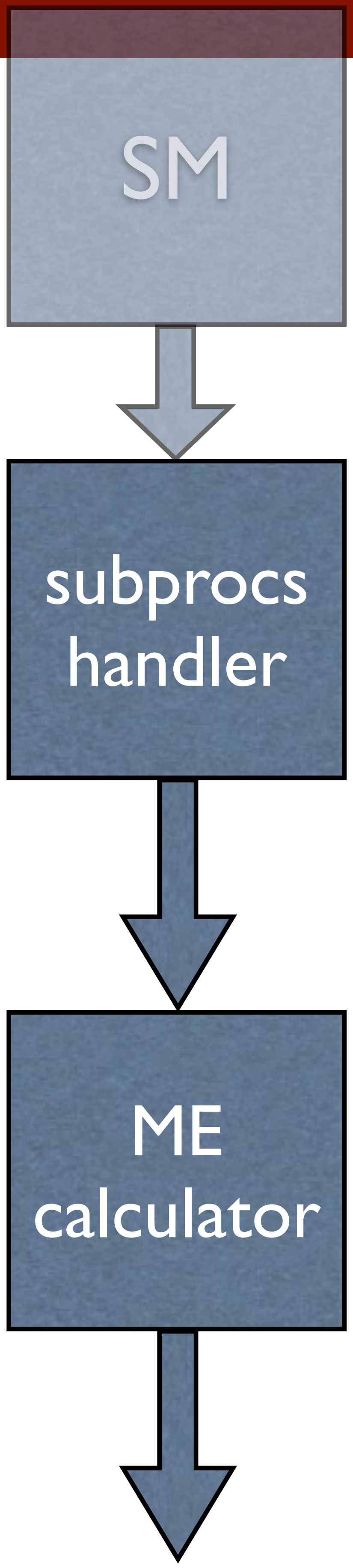
At the most basic level a Monte Carlo event generator is a program which produces particle physics events with the same probability as they occur in nature (virtual collider).

In practice it performs (a possibly large) number of (sometimes very difficult) integrals and then unweights to give the four momenta of the particles that interact with the detector (simulation).

Note that, at least among theorists, the definition of a "Monte Carlo program" also includes codes which don't provide a fully exclusive information on the final state but only cross sections or distributions at the parton level, even when no unweighting can be performed (typically at NLO).

I will refer to these kind of codes as "MC integrators".

General structure



Includes all possible subprocess leading to a given multi-jet final state automatically or manually (done once for all)

```
d~ d -> a d d~ u u~ g
d~ d -> a d d~ c c~ g
s~ s -> a d d~ u u~ g
s~ s -> a d d~ c c~ g
...
```

"Automatically" generates a code to calculate |M|2 for arbitrary processes with many partons in the final state. Use Feynman diagrams with tricks to reduce the factorial growth, others have recursive relations to reduce the complexity to exponential. ☺

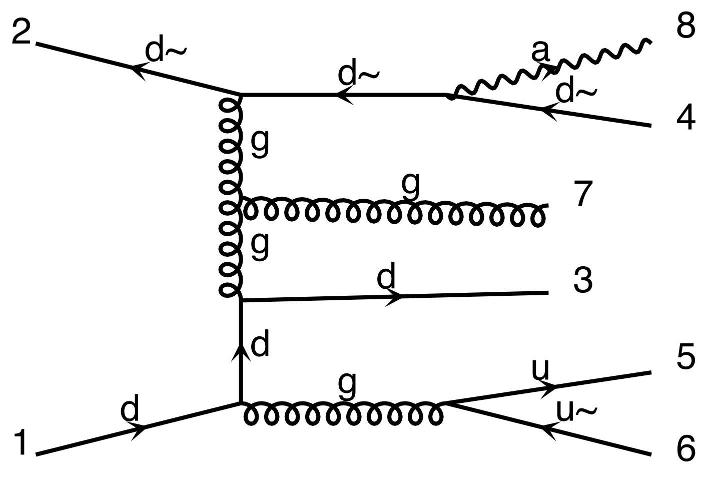
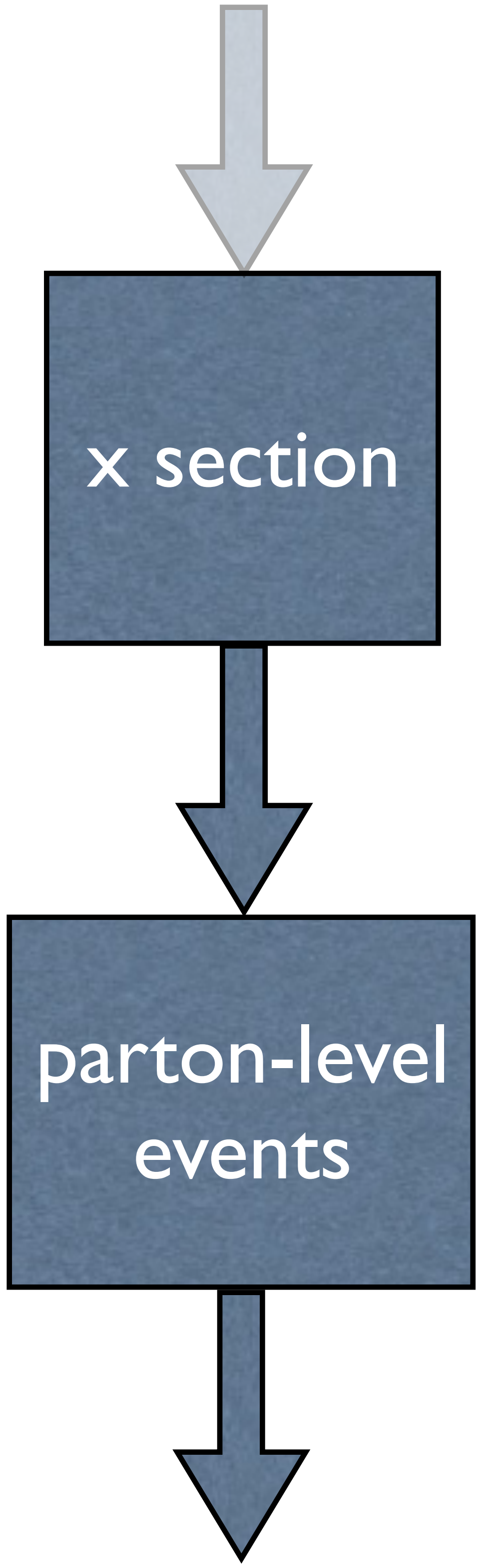
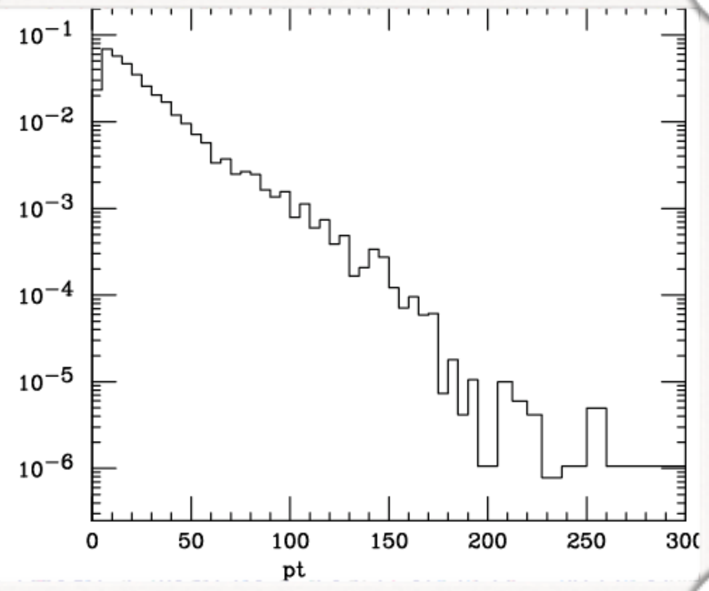


diagram 949 QCD=5, QED=1

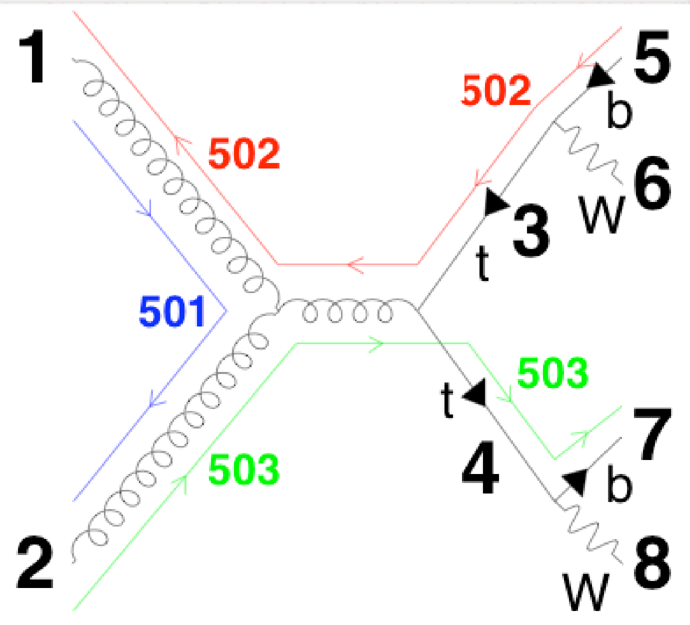
General structure



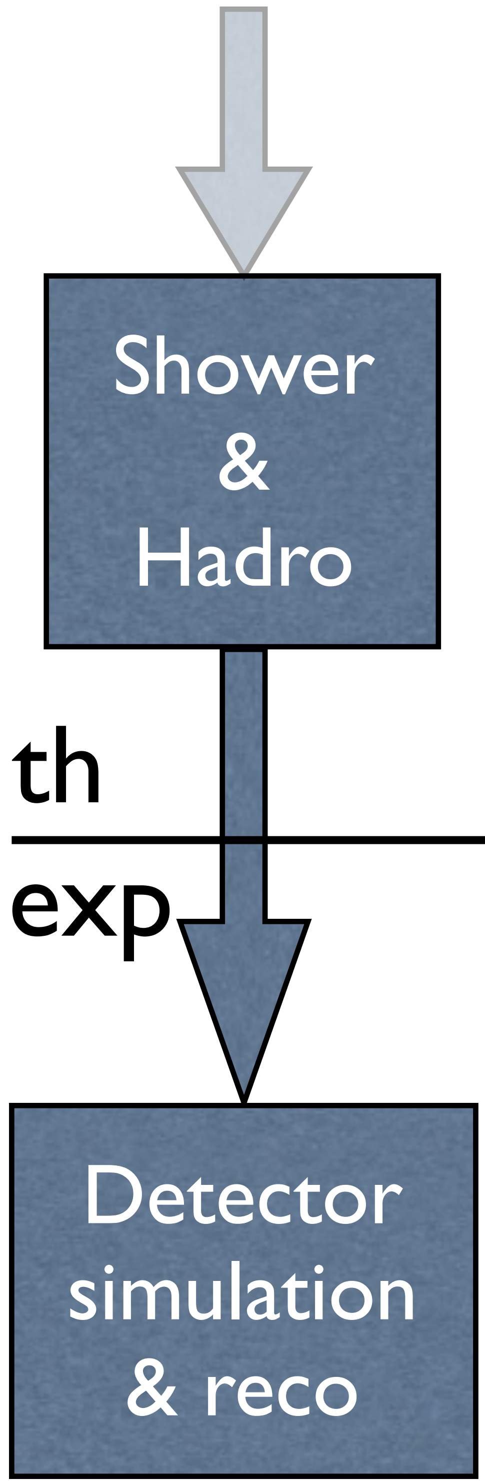
Integrate the matrix element over the phase space using a multi-channel technique and using parton-level cuts.



Events are obtained by unweighting. These are at the parton-level. Information on particle id, momenta, spin, color is given in the Les Houches format.

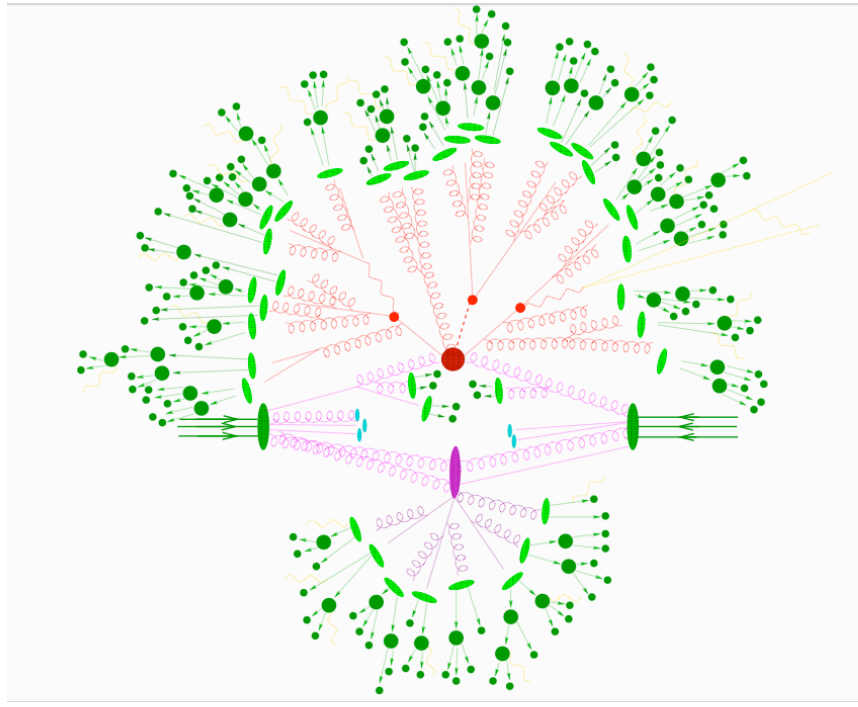


General structure

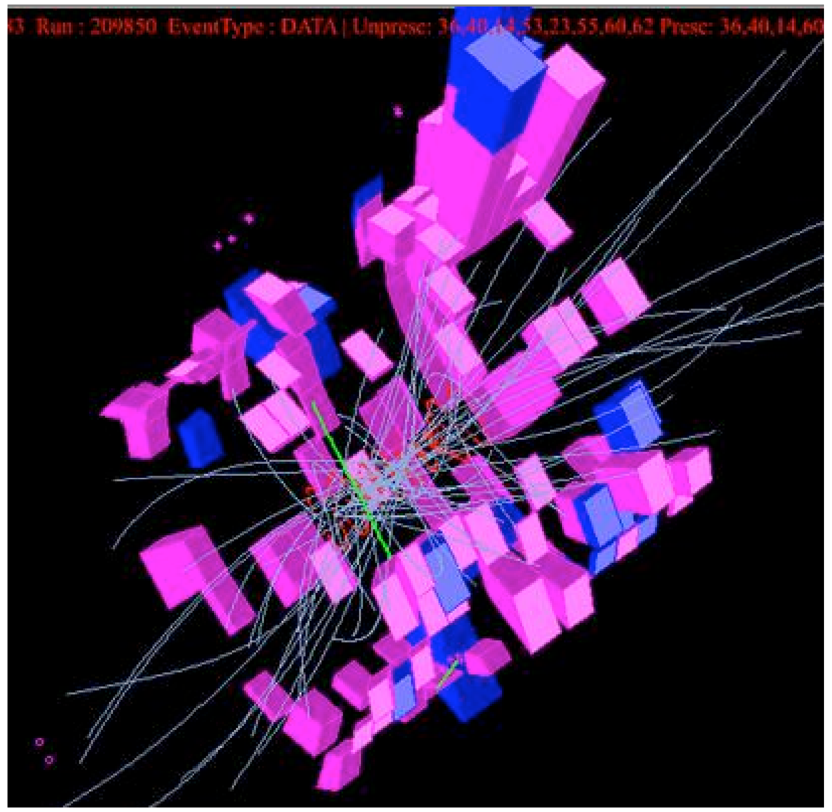


Events in the LH format are passed to the showering and hadronization⇒

high multiplicity hadron-level events



Events in HepMC format are passed through fast or full simulation, and physical objects (leptons, photons, jet, b-jets, taus) are reconstructed.



Codes

- Example of tree-level Monte Carlo codes:
    - Alpgen: fast matrix elements due to use of recursion relations. SM only.
    - Comix (Sherpa): fast matrix elements due to use of recursion relations. Some BSM models implemented (however, e.g. no Majorana particles).
    - MadGraph: Feynman diagrams to generate matrix elements which results in high unweighting efficiency. Virtually all BSM models are (or can be) implemented.
- and more: CalcHEP/CompHEP, Whizard...

Skip FeynRules

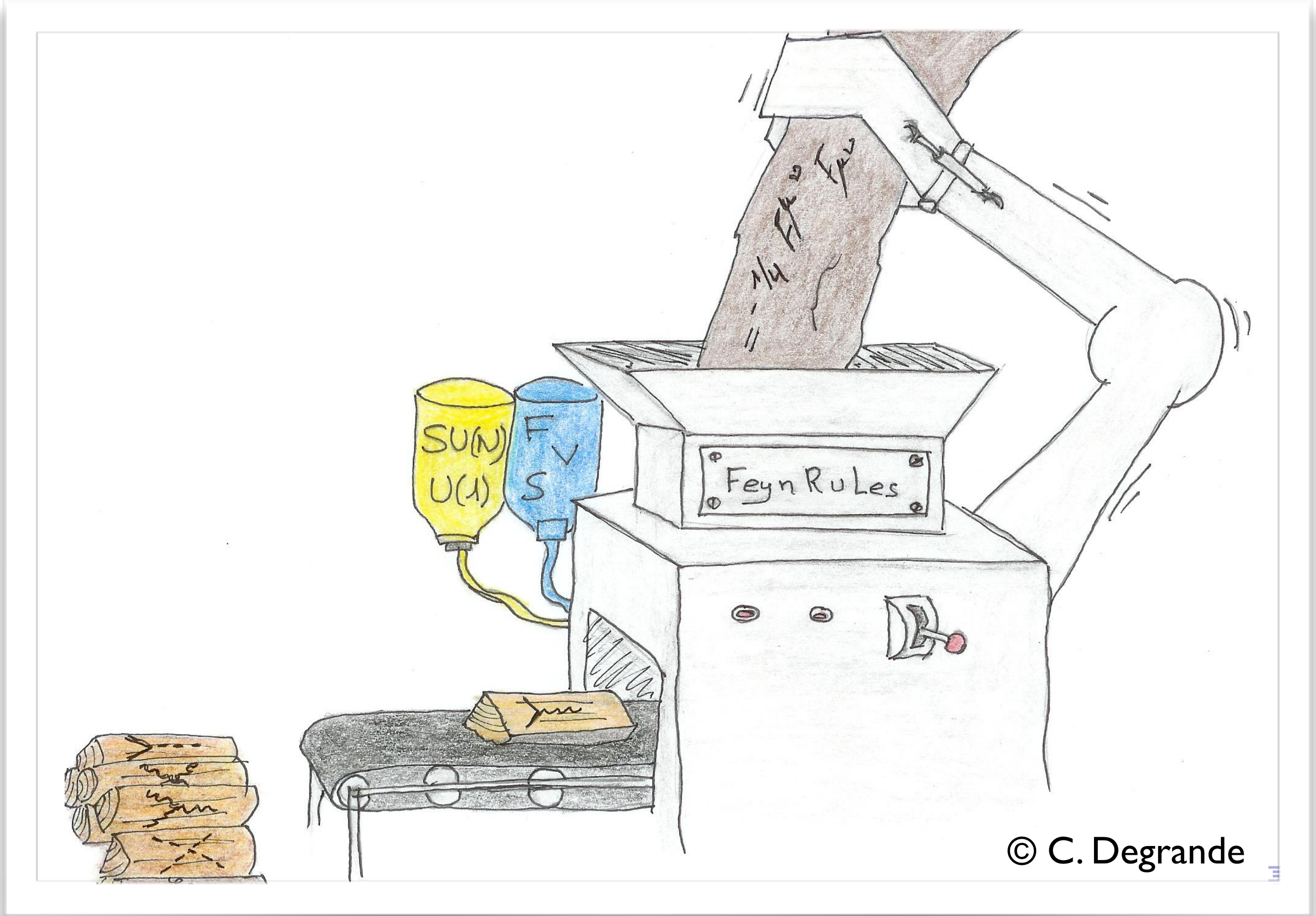
FeynRules

- FeynRules is a Mathematica package that allows to derive Feynman rules from a Lagrangian.
- Current public version: 1.6.x.
- The only requirements on the Lagrangian are:
  - ➡ All indices need to be contracted (i.e. Lorentz and gauge invariance)
  - ➡ Locality
  - ➡ Supported field types:

    spin 0, 1/2, 1, 2 & ghosts (3/2 are coming)

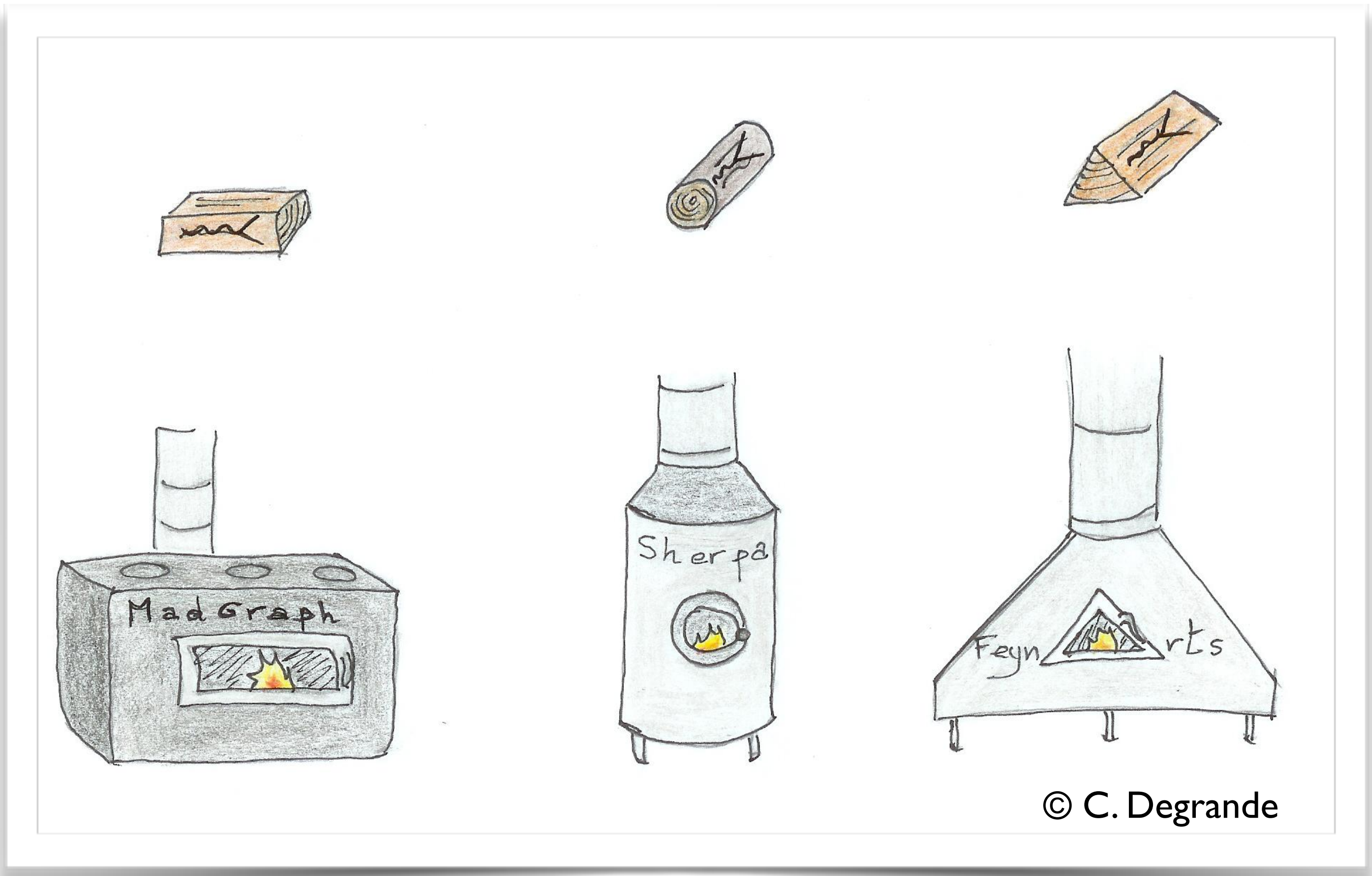
FeynRules

- FeynRules comes with a set of interfaces, that allow to export the Feynman rules to various matrix element generators.
- Interfaces coming with current public version
    - ➡ CalcHep / CompHep
    - ➡ FeynArts / FormCalc
    - ➡ MadGraph
    - ➡ Sherpa
    - ➡ Whizard / Omega
    - ➡ Universal FeynRules Output



FEYNRULES

- FeynRules comes with a set of interfaces, that allow to export the Feynman rules to various matrix element generators.
- Interfaces coming with current public version
  - ➡ CalcHep / CompHep
  - ➡ FeynArts / FormCalc
  - ➡ MadGraph
  - ➡ Sherpa
  - ➡ Whizard / Omega
  - ➡ Universal FeynRules Output



FeynRules

- The input requested form the user is twofold.

- The Model File:
  Definitions of particles and parameters (e.g., a quark)

```
F[1]  ==
   {ClassName      ->   q,
    SelfConjugate ->  False,
    Indices          ->  {Index[Colour]},
    Mass              ->  {MQ,  200},
    Width            ->  {WQ, 5}   }
```

- The Lagrangian:

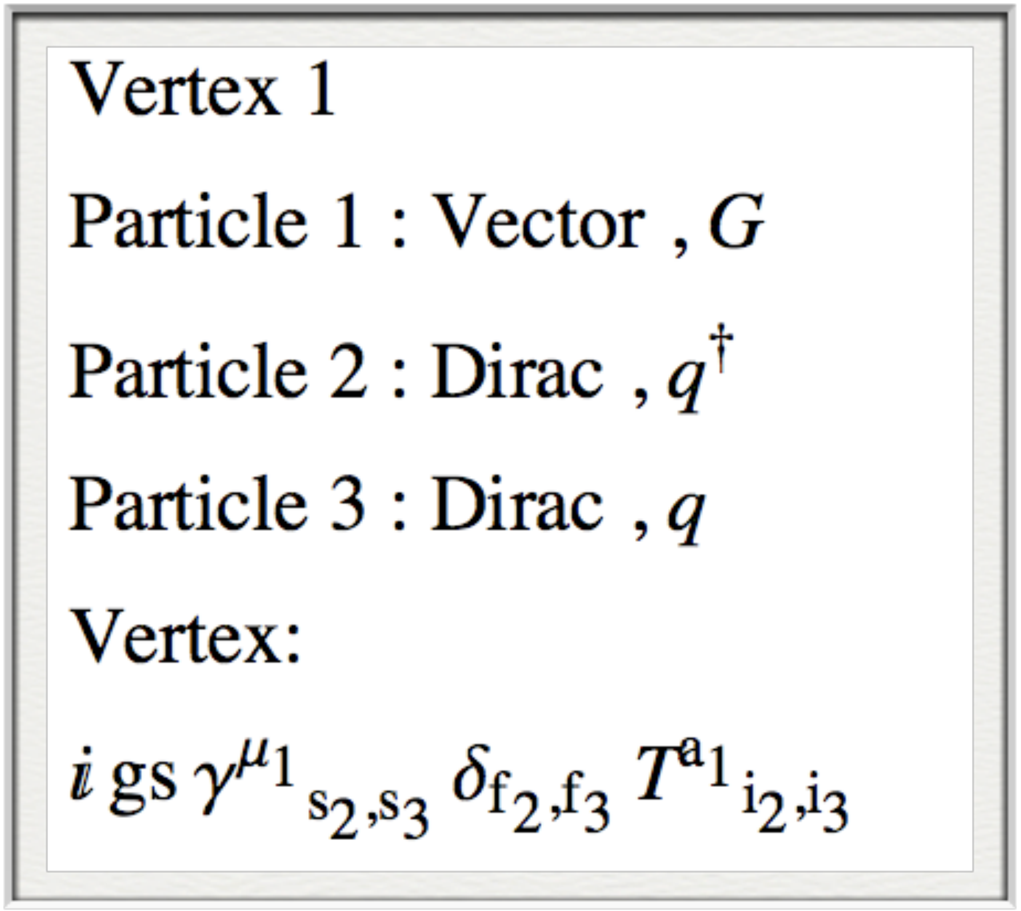
$$\mathcal{L} = -\frac{1}{4} G^a_{\mu\nu} \, G_a^{\mu\nu} + i\bar{q}\,\gamma^\mu \, D_\mu q - M_q \, \bar{q} \, q$$

```
L =
-1/4 FS[G,mu,nu,a] FS[G,mu,nu,a]
+ I qbar.Ga[mu].del[q,mu]
- MQ qbar.q
```

FeynRules

- Once this information has been provided, FeynRules can be used to compute the Feynman rules for the model:

FeynmanRules[L]



Vertex 1

Particle 1 : Vector , G

Particle 2 : Dirac , $q^\dagger$

Particle 3 : Dirac , q

Vertex:

$i\,\mathrm{gs}\,\gamma^{\mu_1}{}_{s_2,s_3}\,\delta_{f_2,f_3}\,T^{a_1}{}_{i_2,i_3}$

FEYNRULES

- Once we have the Feynman rules, we can export them to a MC event generator via the UFO:

WriteUFOutput[L]

- This produces a set of files that can be directly used in the matrix element generator ("plug 'n' play").

interactions.dat

```
  q  q G      GG        QCD
   G G G      MGVX1  QCD
G G G G      MGVX2  QCD QCD
```

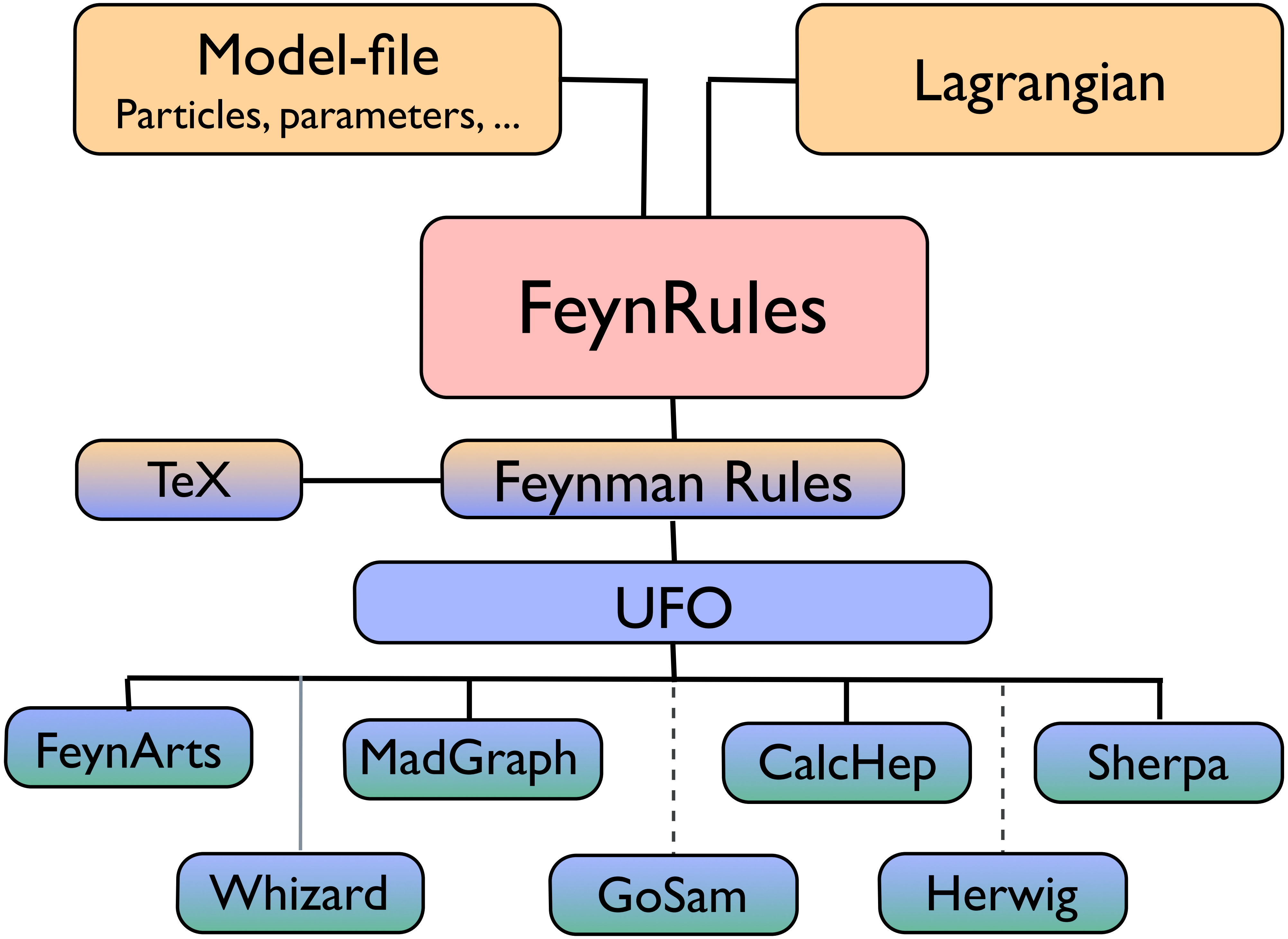
couplings.dat

```
GG(1)    =  -G
GG(1)    =  -G
MGVX1 = G
MGVX2 = G^2
```

particles.dat

```
q   q~  F   S   ZERO  ZERO  T    d    1
G   G    V  C   ZERO  ZERO  O   G  21
```

FeynRules



Model-file
Particles, parameters, ...
Lagrangian
FeynRules
TeX
Feynman Rules
UFO
FeynArts
MadGraph
CalcHep
Sherpa
Whizard
GoSam
Herwig

FEYNRULES

- Already available models:
  - Standard Model
  - Simple extensions of the SM (4th generation, 2HDM, ...)
  - SUSY models ((N)MSSM, RPV-MSSM, ...)
  - Extra-dimensional models (minimal UED, Large Extra Dimensions, ...)
  - Strongly coupled and effective field theories (Minimal Walking Technicolor, Chiral Perturbation theory, ...)
- Straight-forward to start from a given model and to add extra particles/ interactions
- All available models, restrictions, syntax and more information can be found on the FeynRules website:

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

LO PREDICTIONS : REMARKS

$$\sigma_X = \sum_{a,b} \int_0^1 dx_1 dx_2\, f_a(x_1,\mu_F^2) f_b(x_2,\mu_F^2) \times \hat{\sigma}_{ab\to X}(x_1,x_2,\alpha_S(\mu_R^2),\frac{Q^2}{\mu_F^2},\frac{Q^2}{\mu_R^2})$$

- By calculating the short distance coefficient at tree-level we obtain the first estimate of rates for inclusive final states.

- **Even at LO extra radiation is included:** it is described by the PDF's in the initial state and by the definition of a final state parton, which at LO represents all possible final state evolutions.

- Due to the above approximations a cross section at LO can strongly depend on the factorization and renormalization scales.

- Predictions can be systematically improved, at NLO and NNLO, by including higher order corrections in the short distance and in the evolution of the PDF's.

SUMMARY

- Having accurate and flexible simulations tools available for the LHC is a necessity (even more now!!)
- At LO event generation is technically challenging, yet conceptually straightforward.

Credits

To organize this presentation I have benefited from lectures (and actual slides), talks and discussions with many people.
In particular:

- Mike Seymour (MC basics)
- Claude Duhr (FeynRules)
- Johan Alwall (ME+PS merging)
- Rikkert Frederix, Paolo Torrielli (NLO+PS)
- Stefano Frixione, Michelangelo Mangano, Paolo Nason (for QCD, PS, LO, NLO, and more...)
- ....

Whom I all warmly thank!!