Automated computation of scattering amplitudes

Tiziano Peraro

Max-Planck-Institut für Physik
Föhringer Ring 6, D-80805 München, Germany

Particle Physics School Munich Colloquium
Friday, March 14th, 2014

Outline

Outline

Motivation

- The Large Hadron Collider (LHC) proved to be capable of
  - validating SM in unexplored regions of the phase space
  - making new discoveries, e.g. a Higgs(-like?) boson
- LHC high-energy events are characterized by
  - large SM background (could hide new physics)
  - multi-particle final states

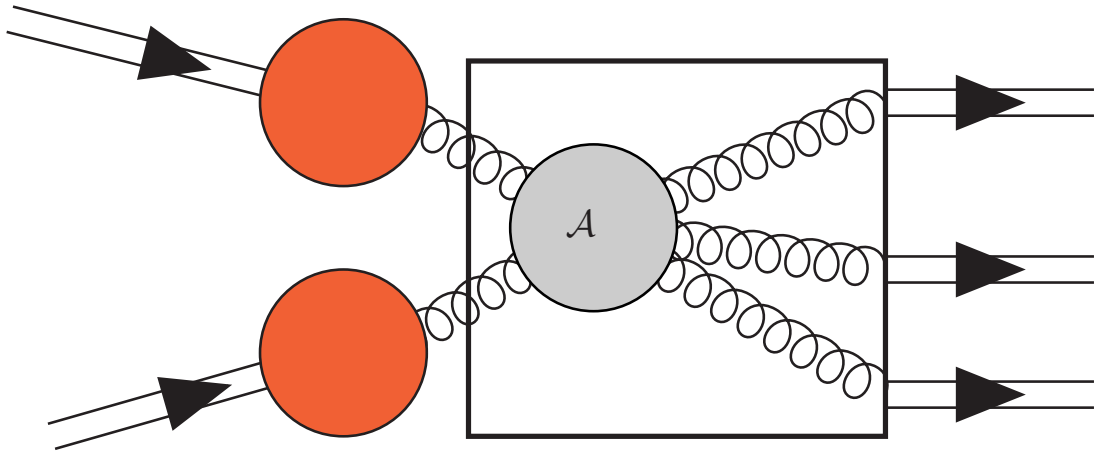
How to make the best use of LHC data?

We need theoretical predictions with

- high accuracy
- multi-particle interactions

Scattering Amplitudes

- Hadron collider interactions



  - Scattering amplitudes represent
    - the process-dependent part of a physical event
    - the main point of contact btw. theoretical models and phenomenology
  - They can be computed in perturbation theory

$$\mathcal{A} \sim \mathcal{A}_{\text{LO}} + \alpha\, \mathcal{A}_{\text{NLO}} + \alpha^2\, \mathcal{A}_{\text{NNLO}} + \dots$$

Why automation?

- avoid human mistakes
- the complexity of the computation is often too high (for a human)
  - . . . but it can be managed by a computer
- every new process requires the computation of new amplitudes

  $\Rightarrow$ implement a universal algorithm once
  and let the computer do the job every time
- testing new models becomes easier
- experimenting with new/different computational techniques is easier within automated frameworks
- . . .

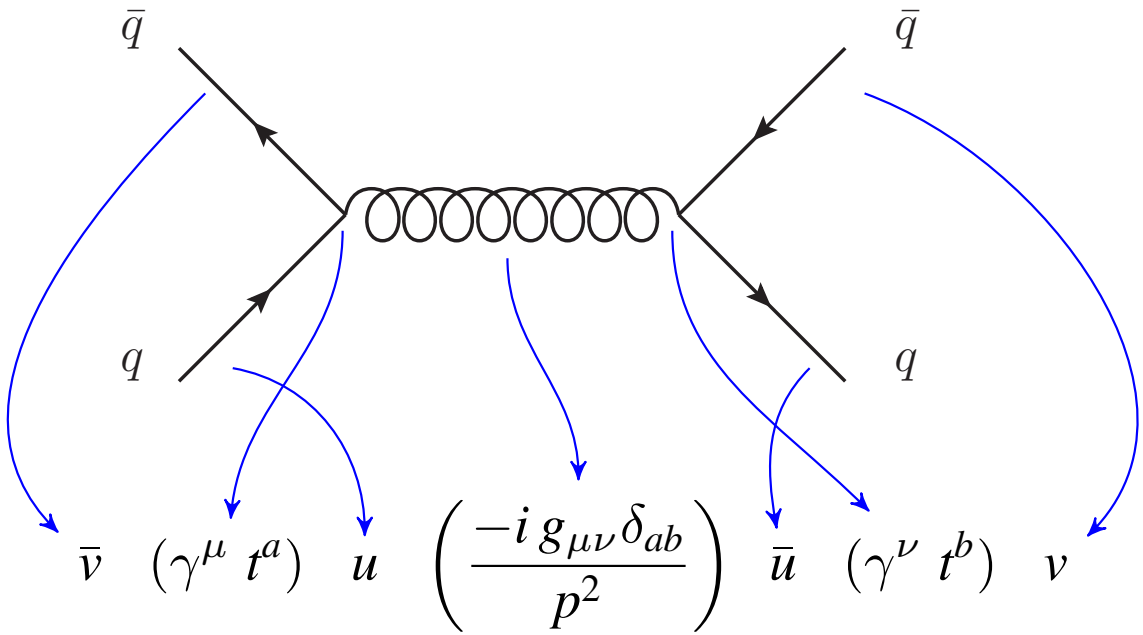
Outline

How to compute a scattering amplitude

- A amplitude is a sum of contributions called Feynman diagrams

$$\mathcal{A} = \sum_i \mathcal{D}_i$$

- Each diagram has a representation as a graph
  - Feynman rules: each vertex, external and internal line correspond to an algebraic factor



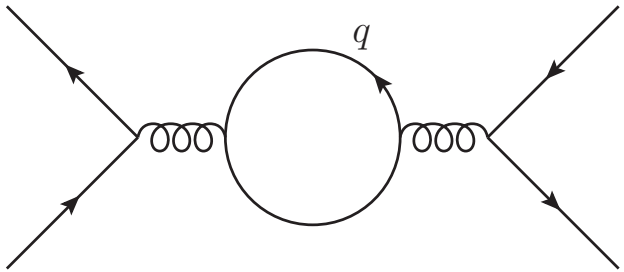
How to compute a scattering amplitude

A strategy:

- draw the Feynman diagrams
  - easy with the right software (FEYNARTS [T. Hahn], QGRAF [P. Nogueira], . . .)
- substitute the Feynman rules
  - use a CAS (e.g. FORM [J. Vermaseren])
  - not always easy (because of renorm. schemes, counter-terms, etc. . .)
  - . . . but they must be implemented only once per theoretical model
- work out the computation
  - typically easy at leading order (LO)
    - usually pure algebra
  - difficult at next-to-leading order (NLO) and beyond
    - involves the computation of integrals

Amplitudes and loop integrals

- Integrals come from diagrams with loops



  - the momentum q running in the loop is not fixed by momentum conservation
  - from QM: all the values of q give a contribution

$$\mathcal{A} \sim \langle \psi_{out} | \psi_{in} \rangle \sim \sum_q \langle \psi_{out} | q \rangle \langle q | \psi_{in} \rangle, \qquad \text{continuum} : \sum_q \to \int d^4 q$$

- Integration should be done in d dimensions: $d^4 q \to d^d q$
  - physical observables are finite for $d \to 4$ (amplitudes are not!)

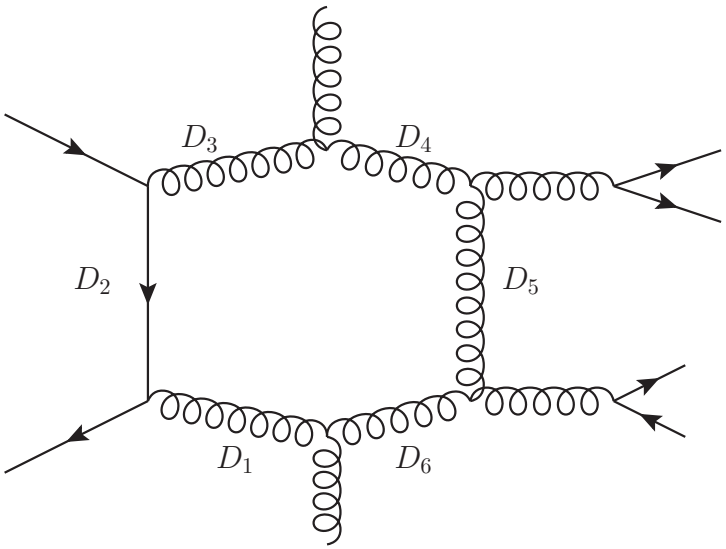
Outline

One-loop amplitudes

- The integrand of a generic n-point one-loop integral:
  - is a rational function in the components of the loop momentum q
  - polynomial numerator $\mathcal{N}$

$$\mathcal{M}_n = \int d^d q \; \mathcal{I}_n, \qquad \mathcal{I}_n \equiv \frac{\mathcal{N}(q)}{D_1 \dots D_n}$$

  - quadratic polynomial denominators D_i
    - they correspond to Feynman loop propagators

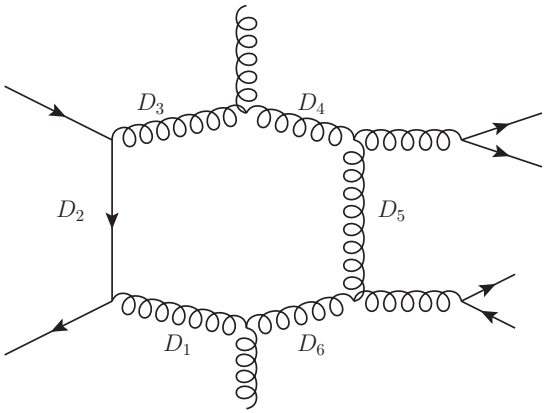


Loop denominators

From the Feynman rules:

$$D_i = (q + p_i)^2 - m_i^2$$

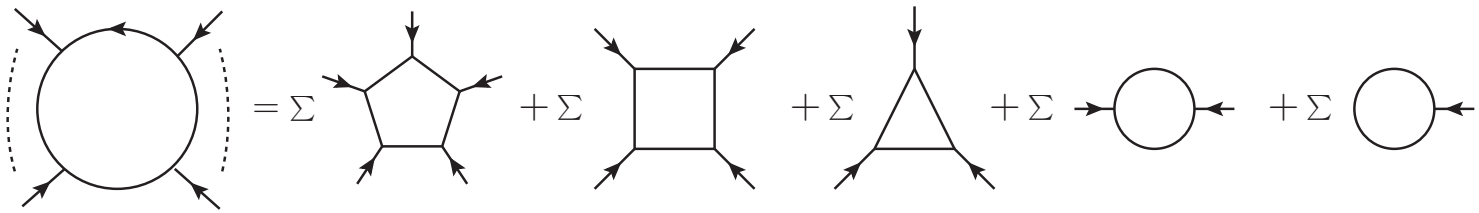
Integrand reduction of one-loop amplitudes



$$\mathcal{M}_n = \int d^d q \, \frac{\mathcal{N}(q)}{D_1 \dots D_n}$$

- In multi-particle scattering amplitudes
  - the number of diagrams (i.e. integrals to be computed) is higher
    - up to thousands or even more
  - the numerators $\mathcal{N}$ are more complex
  - the number of loop denominators is high
    - can be equal to the number of external legs
- Integrand reduction of one-loop amplitudes
  - rewrite the integrand as a sum of "simpler" integrands
  - use an algorithm suited for automation

The Integrand reduction of one-loop amplitudes



- Every one-loop integrand, can be decomposed as
  [Ossola, Papadopoulos, Pittau (2007); Ellis, Giele, Kunszt, Melnikov (2008)]

$$\frac{\mathcal{N}}{D_1 \cdots D_n} = \sum_{j_1 \cdots j_5} \frac{\Delta_{j_1 j_2 j_3 j_4 j_5}}{D_{j_1} D_{j_2} D_{j_3} D_{j_4} D_{j_5}} + \sum_{j_1 \cdots j_4} \frac{\Delta_{j_1 j_2 j_3 j_4}}{D_{j_1} D_{j_2} D_{j_3} D_{j_4}} + \cdots + \sum_{j_1} \frac{\Delta_{j_1}}{D_{j_1}}$$

  - a sum of integrands with 5 or less loop denominators

- The residues $\Delta_{i_1 \cdots i_k}$
  - are polynomials in the components of q
  - have a known, universal parametric form
  - are parametrized by unknown, process-dependent coefficients

  $\Rightarrow$ can be completely determined with a polynomial fit

- Choice of 4-dimensional basis for an m-point residue

$$e_1^2 = e_2^2 = 0\,, \qquad e_1 \cdot e_2 = 1\,, \qquad e_3^2 = e_4^2 = \delta_{m4}\,, \qquad e_3 \cdot e_4 = -(1-\delta_{m4})$$

- Coordinates: $\mathbf{z} = (z_1, z_2, z_3, z_4, z_5) \equiv (x_1, x_2, x_3, x_4, \mu^2)$

$$q^\mu_{\text{4-dim}} = -p^\mu_{i_1} + x_1\, e_1^\mu + x_2\, e_2^\mu + x_3\, e_3^\mu + x_4\, e_4^\mu\,, \qquad \bar{q}^2 = q^2_{\text{4-dim}} - \mu^2$$

- Generic numerator

$$\mathcal{N} = \sum_{j_1,\dots,j_5} \alpha_{\vec{j}}\, z_1^{j_1}\, z_2^{j_2}\, z_3^{j_3}\, z_4^{j_4}\, z_5^{j_5}\,, \qquad (j_1 \dots j_5) \quad \text{such that} \quad \text{rank}(\mathcal{N}) \leq \#\text{ loop-denom.}$$

- Residues

$$\begin{aligned}
\Delta_{i_1 i_2 i_3 i_4 i_5} &= c_0\, \mu^2 \\
\Delta_{i_1 i_2 i_3 i_4} &= c_0 + c_1 x_4 + \mu^2 (c_2 + c_3 x_4 + \mu^2 c_4) \\
\Delta_{i_1 i_2 i_3} &= c_0 + c_1 x_3 + c_2 x_3^2 + c_3 x_3^3 + c_4 x_4 + c_5 x_4^2 + c_6 x_4^3 + \mu^2 (c_7 + c_8 x_3 + c_9 x_4) \\
\Delta_{i_1 i_2} &= c_0 + c_1 x_2 + c_2 x_3 + c_3 x_4 + c_4 x_2^2 + c_5 x_3^2 + c_6 x_4^2 + c_7 x_2 x_3 + c_9 x_2 x_4 + c_9 \mu^2 \\
\Delta_{i_1} &= c_0 + c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4
\end{aligned}$$

- It can be easily extended to higher-rank numerators

The Integrand reduction of one-loop amplitudes

- After integration
  - some terms vanish and do not contribute to the amplitude
    $\Rightarrow$ spurious terms
  - non-vanishing terms give Master Integrals (MIs)
  - the amplitude is a linear combination of known MIs
- The coefficients of this linear combination
  - can be identified with some of the coefficients which parametrize the polynomial residues

The Integrand reduction of one-loop amplitudes

- After integration
  - some terms vanish and do not contribute to the amplitude
    $\Rightarrow$ spurious terms
  - non-vanishing terms give Master Integrals (MIs)
  - the amplitude is a linear combination of known MIs
- The coefficients of this linear combination
  - can be identified with some of the coefficients which parametrize the polynomial residues

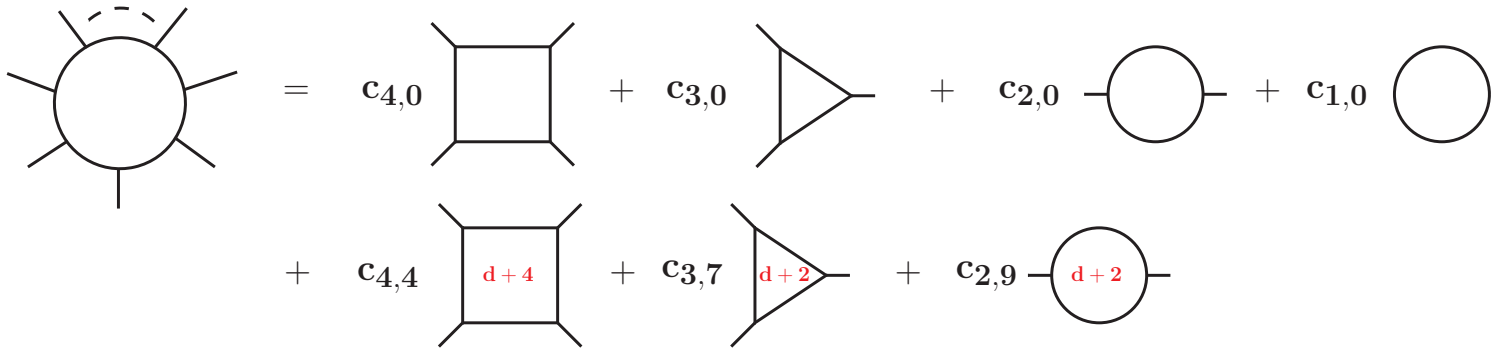
  $\Rightarrow$ reduction to MIs $\equiv$ polynomial fit of the residues

The Integrand reduction of one-loop amplitudes

- After integration
  - some terms vanish and do not contribute to the amplitude
    $\Rightarrow$ spurious terms
  - non-vanishing terms give Master Integrals (MIs)
  - the amplitude is a linear combination of known MIs
- The coefficients of this linear combination
  - can be identified with some of the coefficients which parametrize the polynomial residues

  $\Rightarrow$ reduction to MIs $\equiv$ polynomial fit of the residues

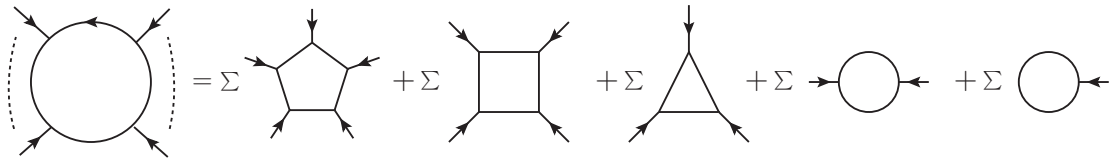
★ any one-loop amplitude can be computed with a polynomial fit

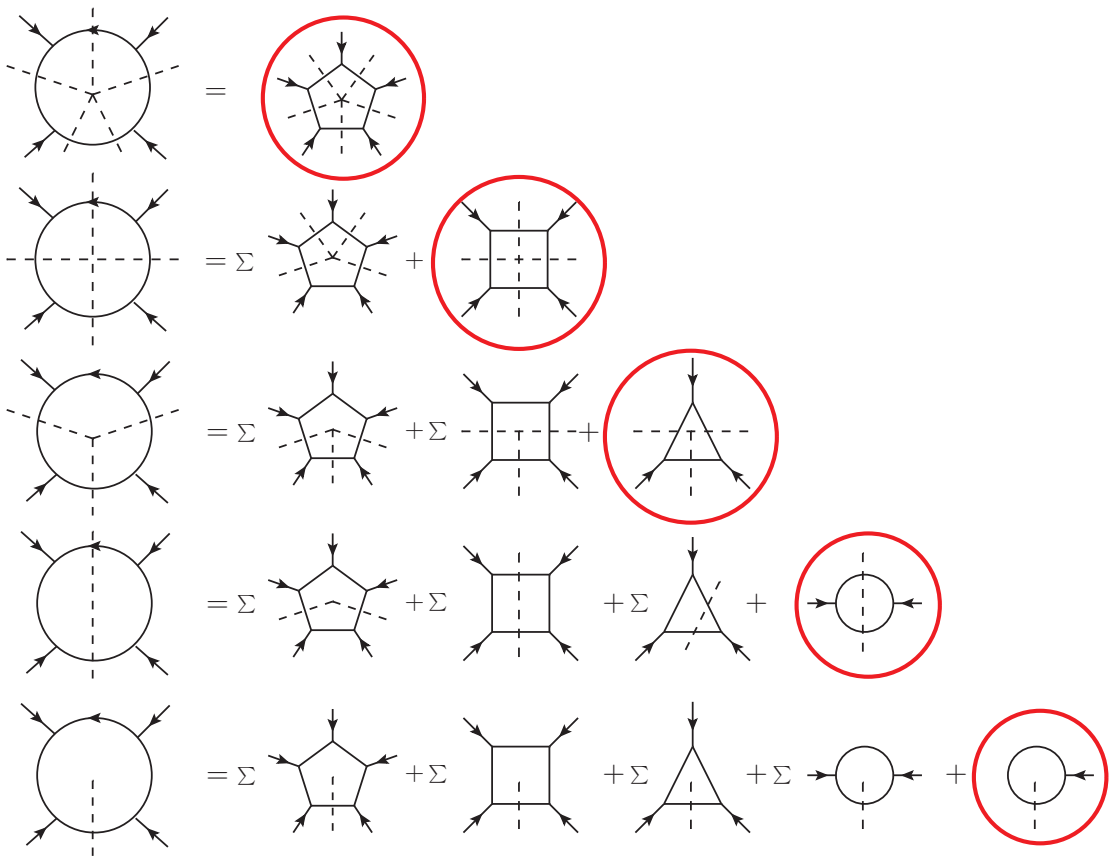


Fit-on-the-cut at one-loop

[Ossola, Papadopoulos, Pittau (2007)]

Integrand decomposition:





Fit-on-the cut

- fit m-point residues on m-ple cuts
- Cutting a loop propagator means

$$\frac{1}{D_i} \to \delta(D_i)$$

i.e. putting it on-shell

Automation of one-loop computation

In several one-loop packages we can distinguish three phases:

1. Generation
   - generate the integrand
   - cast it in a suitable form for reduction
   - write it in a piece of source code (e.g. FORTRAN or C/C++)
2. Compilation
   - compile the code
3. Run-time
   - use a reduction library in order to compute the integrals

Automation of one-loop computation in GoSam

GoSam is a Python package which:

- generates analytic integrands
  - using QGraf and Form
- writes them into Fortran90 code
- can use different reduction algorithms at run-time

Automation of one-loop computation in GoSam

GoSam is a Python package which:

- generates analytic integrands
  - using QGraf and Form
- writes them into Fortran90 code
- can use different reduction algorithms at run-time
  - Samurai (d-dim. integrand reduction)
    - faster than Golem95 but numerically less stable
    - current default (in GoSam-1.0)

Automation of one-loop computation in GoSam

GoSam is a Python package which:

- generates analytic integrands
  - using QGraf and Form
- writes them into Fortran90 code
- can use different reduction algorithms at run-time
  - Samurai (d-dim. integrand reduction)
    - faster than Golem95 but numerically less stable
    - current default (in GoSam-1.0)
  - Golem95 (tensor reduction)
    - slower than Samurai but more stable
    - default rescue-system for unstable points

Automation of one-loop computation in GoSam

GoSam is a Python package which:

- generates analytic integrands
  - using QGraf and Form
- writes them into Fortran90 code
- can use different reduction algorithms at run-time
  - Samurai (d-dim. integrand reduction)
    - faster than Golem95 but numerically less stable
    - current default (in GoSam-1.0)
  - Golem95 (tensor reduction)
    - slower than Samurai but more stable
    - default rescue-system for unstable points
  - Ninja
    - implements a new integrand reduction algorithm
    - fast (2 to 5 times faster than Samurai)
    - stable (in worst cases $\mathcal{O}(1/1000)$ unstable points)
    - new default (in GoSam-2.0)

Integrand reduction via Laurent expansion (NINJA)

P. Mastrolia, E. Mirabella, T.P. (2012)

The integrand reduction via Laurent expansion:

- fits residues by taking their asymptotic expansions on the cuts
  - elaborating ideas first proposed by Forde and Badger
- yields diagonal systems of equations for the coefficients
- requires the computation of fewer coefficients
- subtractions of higher point residues is simplified
  - implemented as corrections at the coefficient level

Integrand reduction via Laurent expansion (NINJA)

P. Mastrolia, E. Mirabella, T.P. (2012)

The integrand reduction via Laurent expansion:

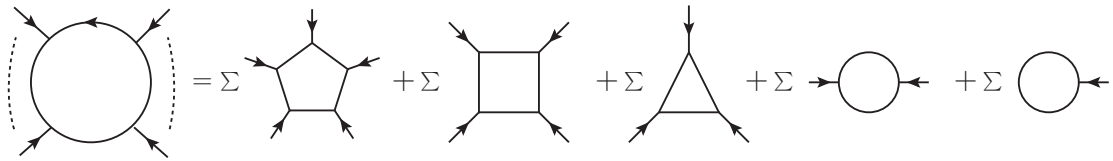
- fits residues by taking their asymptotic expansions on the cuts
  - elaborating ideas first proposed by Forde and Badger
- yields diagonal systems of equations for the coefficients
- requires the computation of fewer coefficients
- subtractions of higher point residues is simplified
  - implemented as corrections at the coefficient level

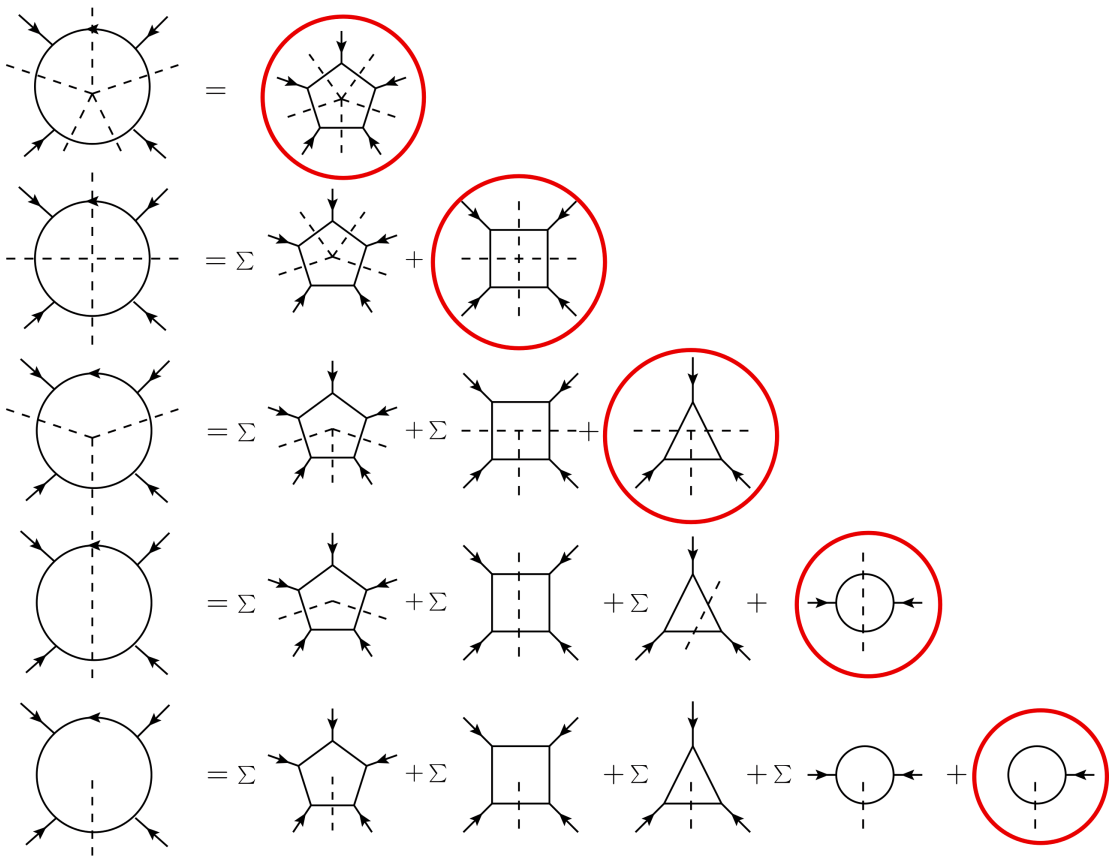
★ Implemented in the semi-numerical C++ library NINJA [T.P. (2014)]

- Laurent expansions via a simplified polynomial-division algorithm
- interfaced with the package GOSAM
- is a faster and more stable integrand-reduction algorithm

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:

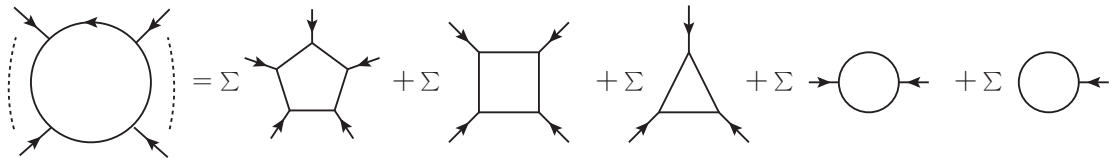


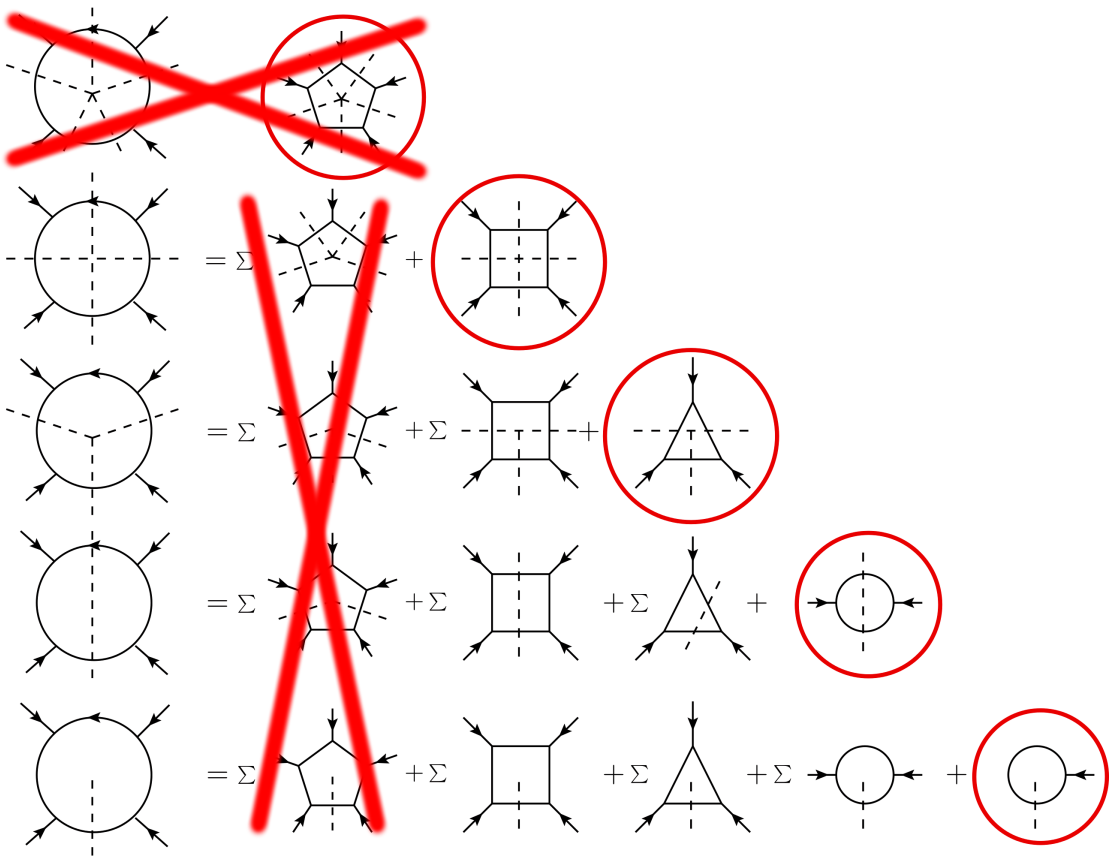


Laurent-expansion method

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



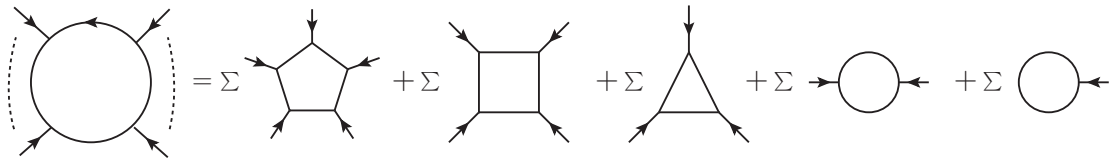


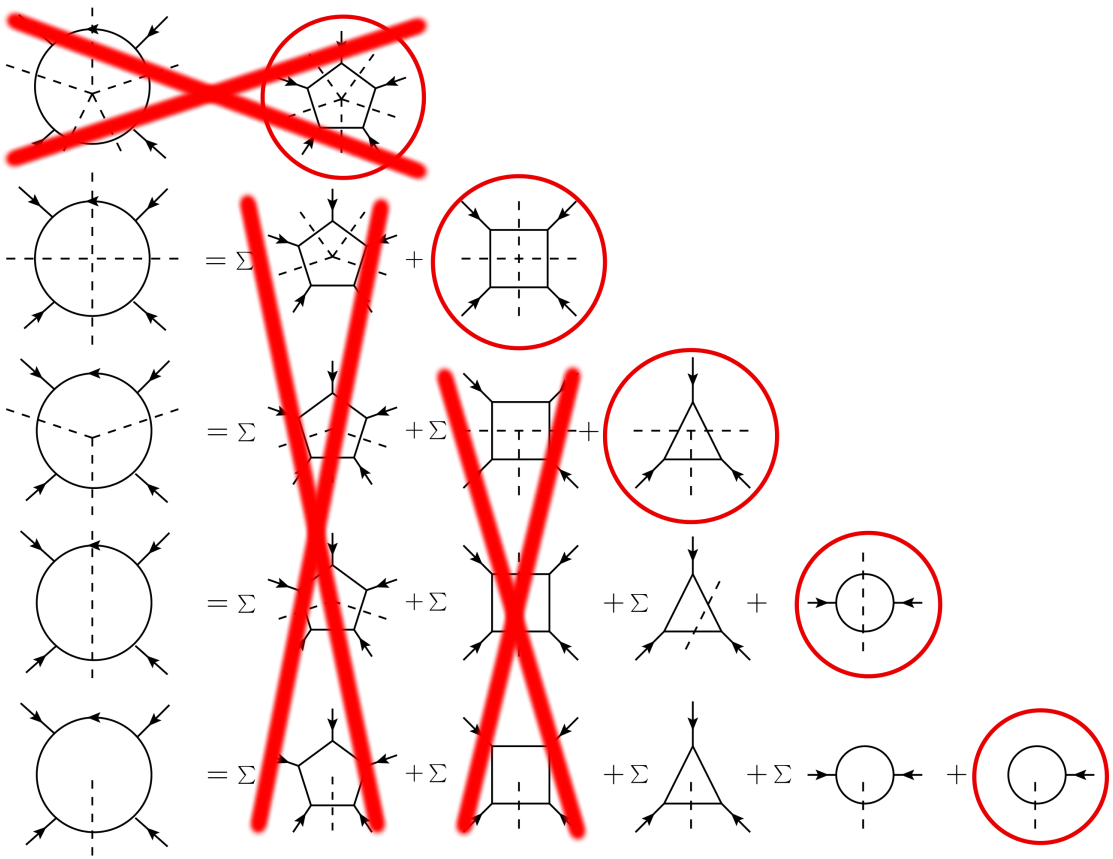
Laurent-expansion method

- pentagons not needed

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



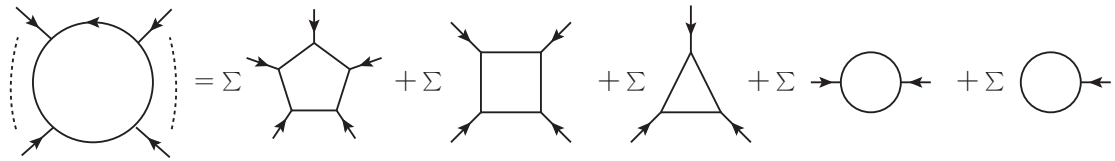


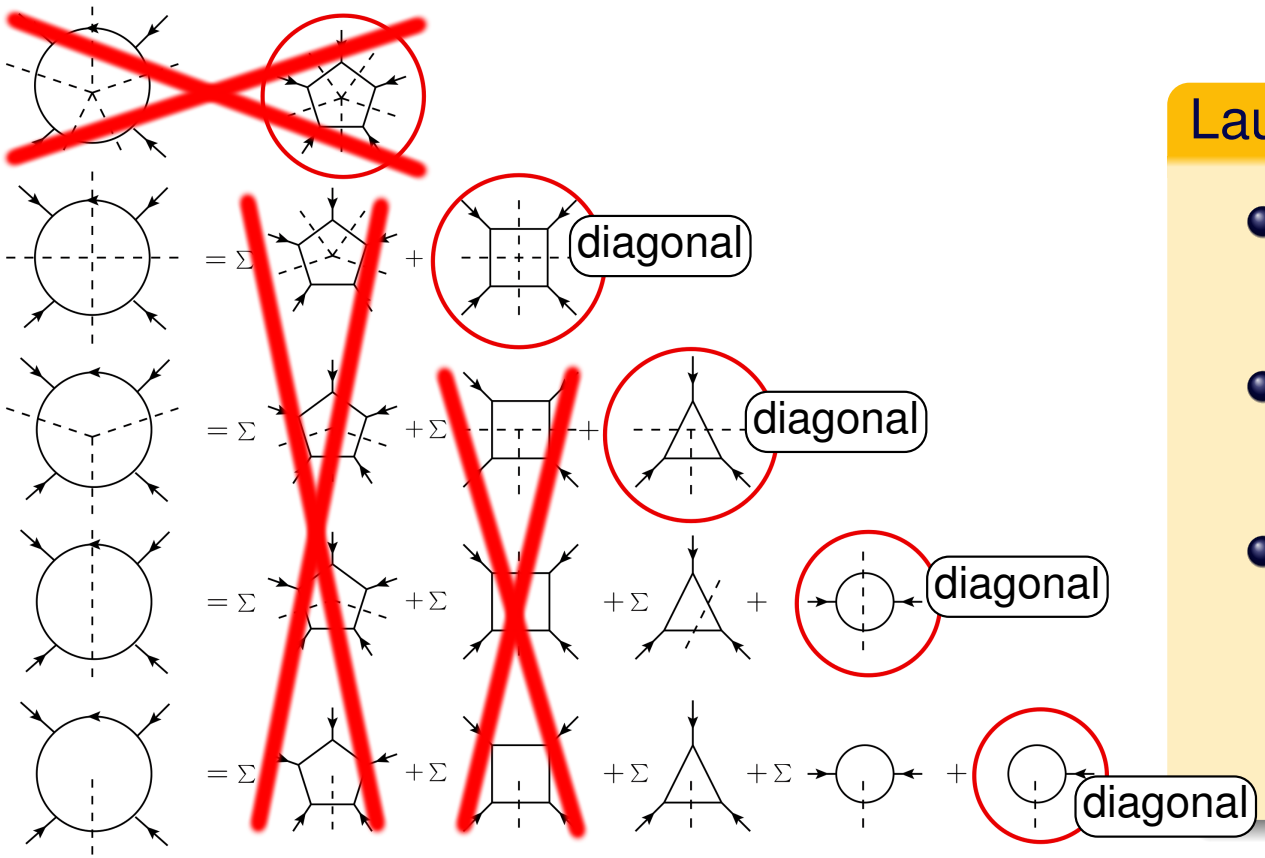
Laurent-expansion method

- pentagons not needed
- boxes never subtracted

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



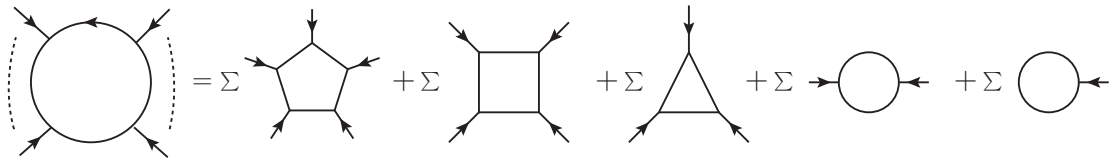


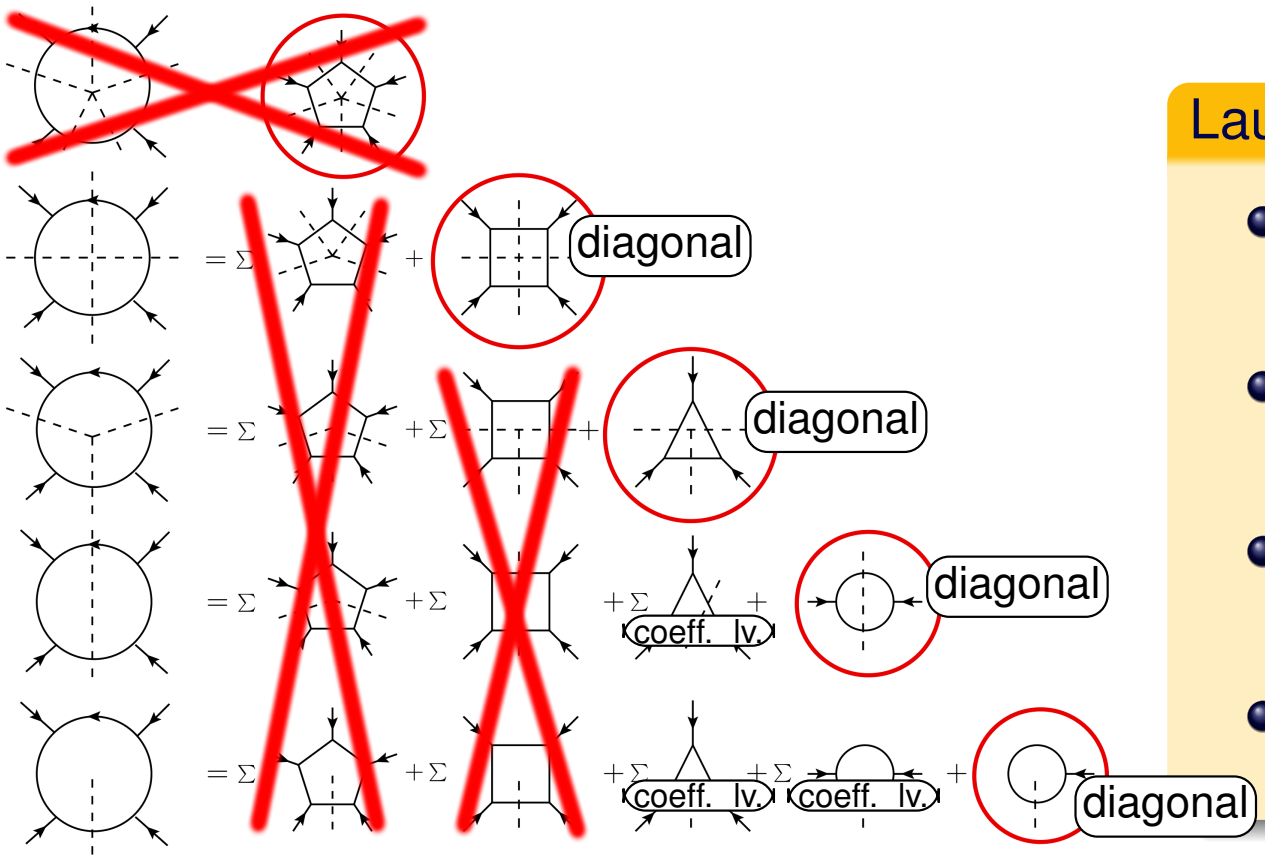
Laurent-expansion method

- pentagons not needed
- boxes never subtracted
- diagonal systems of equations

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:





Laurent-expansion method

- pentagons not needed
- boxes never subtracted
- diagonal systems of equations
- subtractions at coefficient level

Benchmarks of GoSam + Ninja

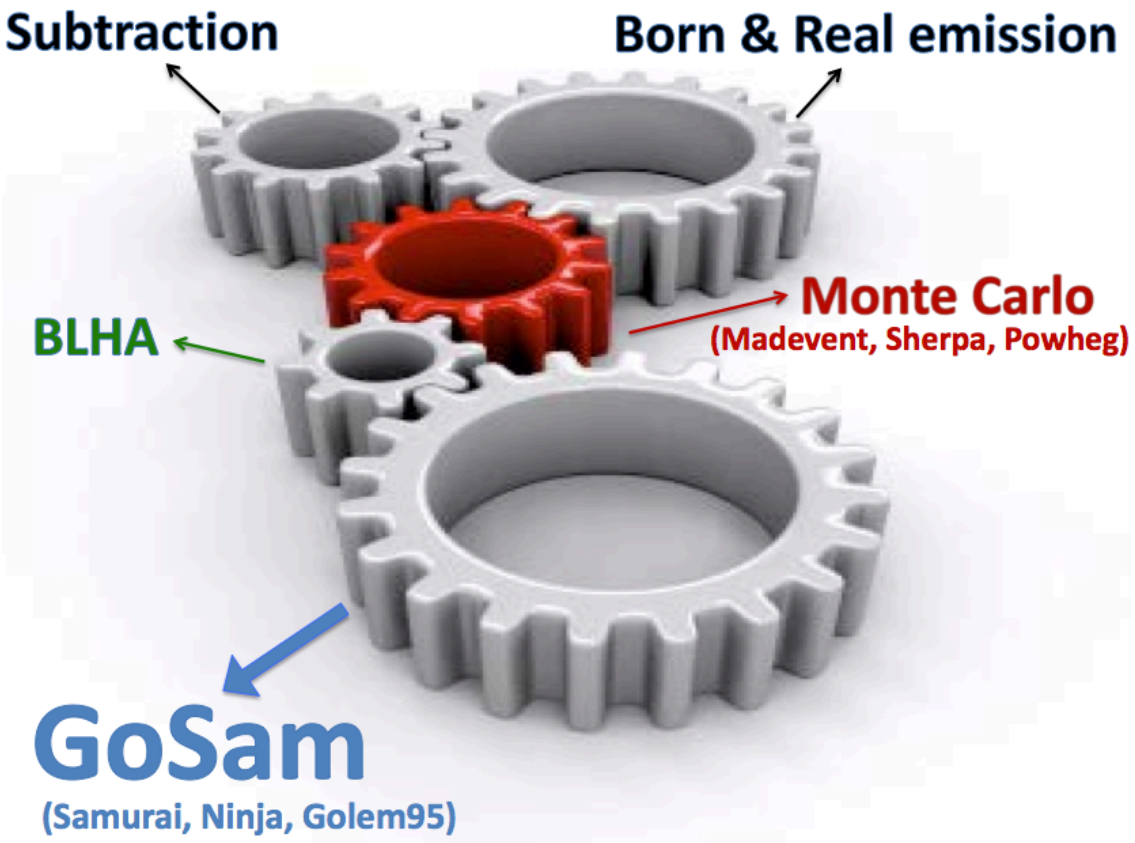
H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T.P. (2013)

Benchmarks: GoSam + Ninja			
Process		# NLO diagrams	ms/event[a]
$W+3j$	$d\bar{u} \to \bar{\nu}_e e^- ggg$	1 411	226
$Z+3j$	$d\bar{d} \to e^+ e^- ggg$	2 928	1 911
$t\bar{t}b\bar{b}$ $(m_b \neq 0)$	$d\bar{d} \to t\bar{t}b\bar{b}$	275	178
	$gg \to t\bar{t}b\bar{b}$	1 530	5 685
$t\bar{t}+2j$	$gg \to t\bar{t}gg$	4 700	13 827
$W\,b\,\bar{b}+1j$ $(m_b \neq 0)$	$u\bar{d} \to e^+ \nu_e b\bar{b}g$	312	67
$W\,b\,\bar{b}+2j$ $(m_b \neq 0)$	$u\bar{d} \to e^+ \nu_e b\bar{b}s\bar{s}$	648	181
	$u\bar{d} \to e^+ \nu_e b\bar{b}d\bar{d}$	1 220	895
	$u\bar{d} \to e^+ \nu_e b\bar{b}gg$	3 923	5 387
$H+3j$ in GF	$gg \to Hggg$	9 325	8 961
$t\,\bar{t}\,H+1j$	$gg \to t\bar{t}Hg$	1 517	1 505
$H+3j$ in VBF	$u\bar{u} \to Hgu\bar{u}$	432	101
$H+4j$ in VBF	$u\bar{u} \to Hggu\bar{u}$	1 176	669
$H+5j$ in VBF	$u\bar{u} \to Hgggu\bar{u}$	15 036	29 200

more processes in arXiv:1312.6678

[a] Timings refer to full color- and helicity-summed amplitudes, using an Intel Core i7 CPU @ 3.40GHz, compiled with `ifort`.

From amplitudes to observables with GoSam



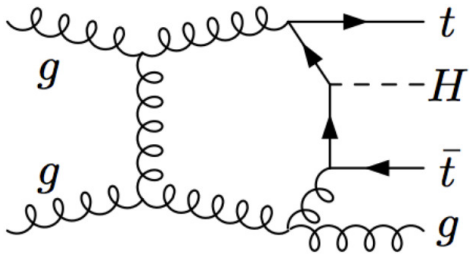
The GoSam collaboration:

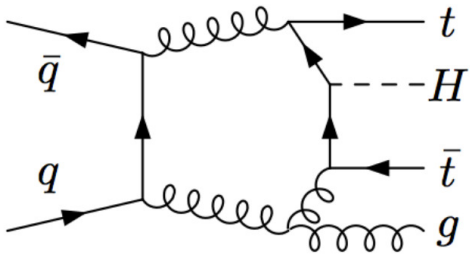
G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, J. Reichel , J. Schlenk, J. F. von Soden-Fraunhofen, T. Reiter, F. Tramontano, T.P.

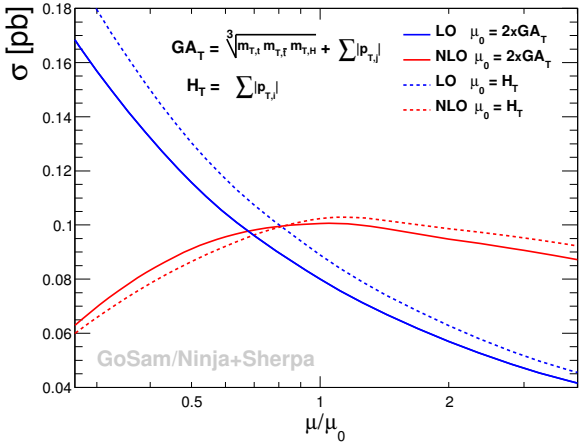
Application: $pp \to t\bar{t}H + jet$ with GoSam + Ninja

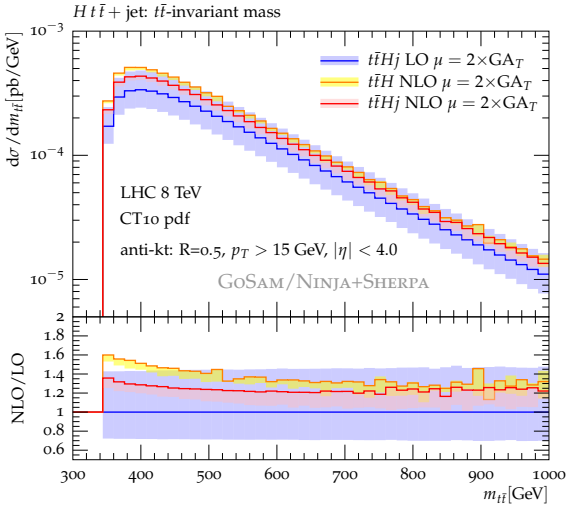
H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2013)

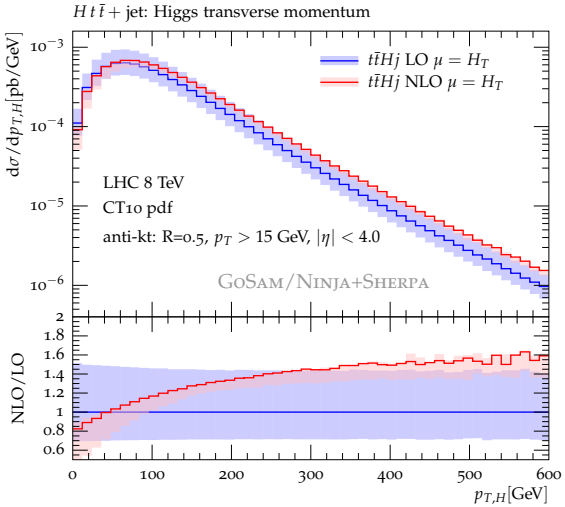
- Interfaced with the Monte Carlo Sherpa

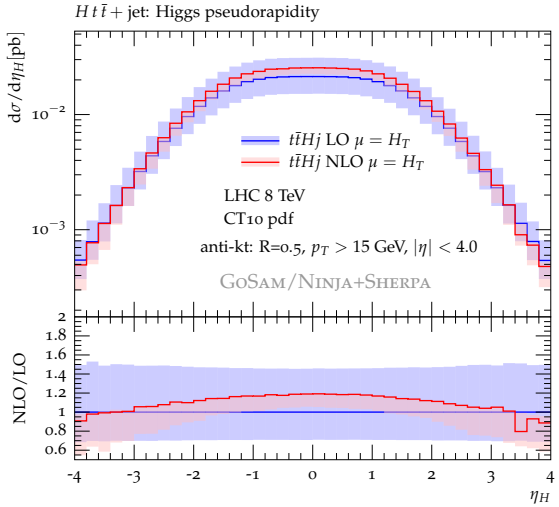












Outline

Higher-loop techniques

- Integrand reduction can be extended to higher loop
  P. Mastrolia, G. Ossola (2011)
  - using multivariate polynomial division techniques
    Y. Zhang (2012); S. Badger, H. Frellesvig, Y. Zhang (2012-13);
    P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2012-13)
- The Master Integrals are not all known at 2 or higher loops
- Recent developments in the computation of higher-loop integrals:
  - differential equations
    J. Henn (2013); Henn, Smirnov, Melnikov (2013-14);
    M. Argeri, S. Di Vita, P. Mastrolia, E. Mirabella, J. Schlenk, U. Schubert,
    L. Tancredi (2014)
  - numerical techniques (e.g. SecDec)
    S. Borowka, G. Heinrich (2013-14)

Outline

Summary & Conclusions

- Automated one-loop calculations via integrand-reduction
  - allow to compute any one-loop integral in any QFT
  - are implemented in several public codes (e.g. CUTTOOLS, SAMURAI, NINJA)
  - are producing phenomenological results for LHC (e.g. with GOSAM, FORMCALC [T. Hahn et al.])
- At higher loop
  - we have a framework which extends integrand reduction to any perturbative order
  - many interesting results from both analytic and numerical techniques
  - work is still in progress but things seem to be promising

THANK YOU
FOR YOUR ATTENTION