One-loop scattering amplitudes via Laurent expansion with NINJA

Tiziano Peraro

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

NLO Users-of-Sherpa meeting
10 January 2014

Outline

1. Introduction and motivation
2. The integrand reduction of one-loop amplitudes
3. NINJA: Integrand reduction via Laurent expansion
4. Summary and Outlook

Introduction and motivation

The goal

Implementation of a reduction algorithm for one-loop amplitudes which

- can be applied to processes with many external legs
- allows the presence of massive external and internal particles
- is reasonably stable and fast
- is suited for automation

NINJA

NINJA is a C++ library which

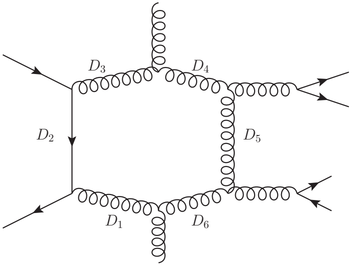
- implements the Larent expansion method for one-loop amplitudes
- is fast, stable and can be applied to any one-loop integrand

The Integrand reduction of one-loop amplitudes

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

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

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



$$D_i = (\bar{q} + p_i)^2 - m_i^2$$

The Integrand reduction of one-loop amplitudes

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

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

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

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

- the decomposition has been recently extended to higher-loops using techniques based on multivariate polynomial division
  [Y. Zhang (2012), P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2012)]

- 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

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

- 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

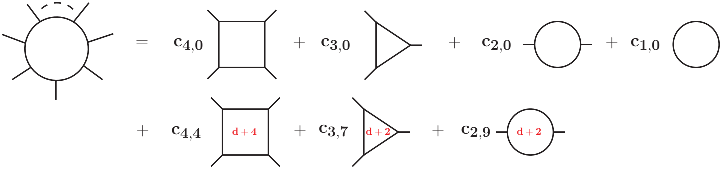
  $\Rightarrow$ reduction to MIs $\equiv$ polynomial fit of the residues

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

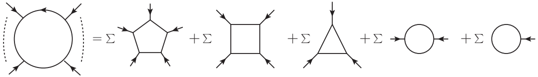
★ any one-loop amplitude can be computed with a polynomial fit

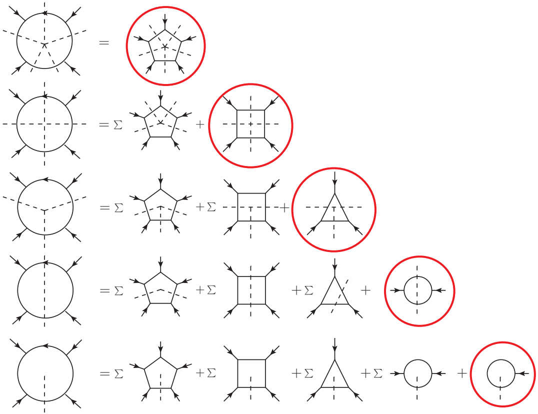


Fit-on-the-cut at one-loop

[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

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:

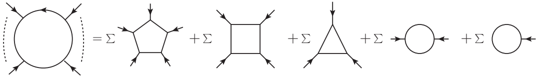
- fits residues by taking their asymptotic expansions on the cuts
  - elaborating ideas first proposed by Forde and Badger
- yields diagonal systems of equations for the coefficients
- requires the computation of fewer coefficients
- subtractions of higher point residues is simplified
  - implemented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

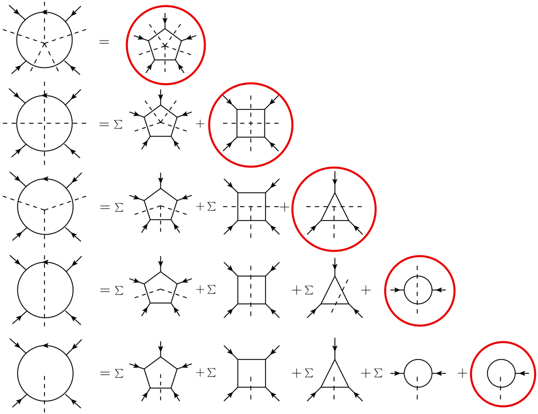
★ Implemented in the semi-numerical C++ library NINJA

- 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:

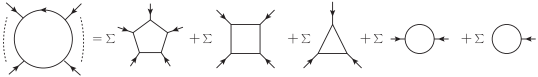


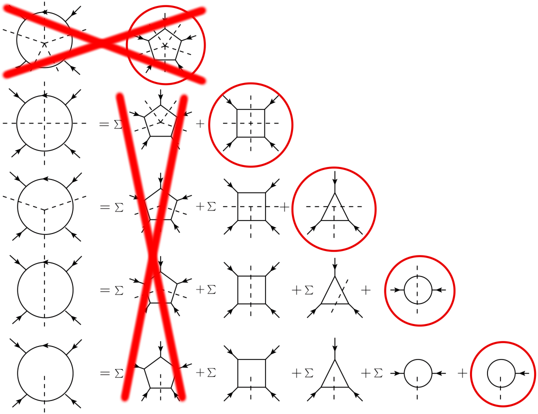


Laurent-expansion method

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



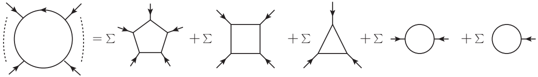


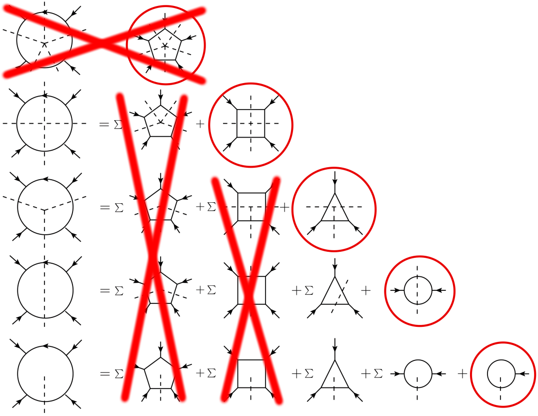
Laurent-expansion method

- pentagons not needed

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



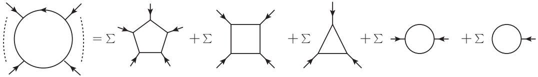


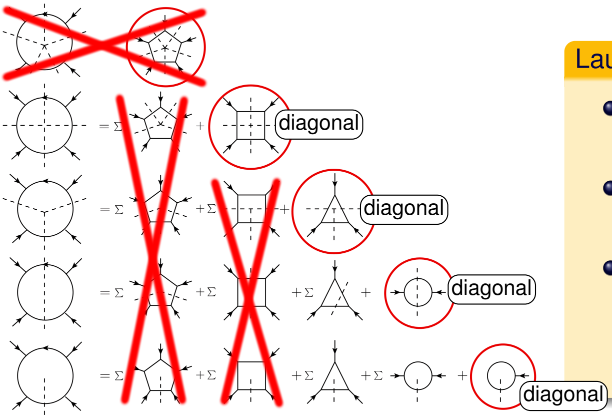
Laurent-expansion method

- pentagons not needed
- boxes never subtracted

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



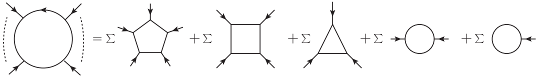


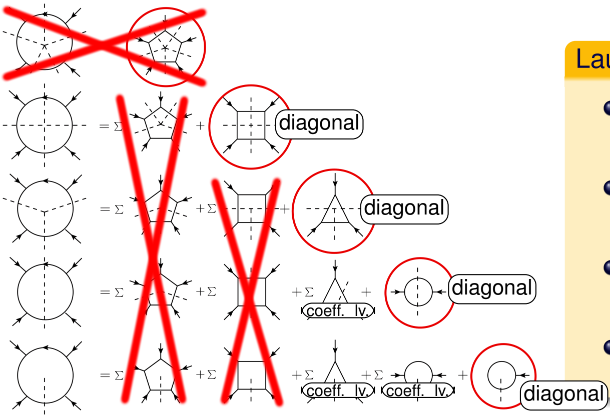
Laurent-expansion method

- 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

Example: One-loop bubbles via Laurent expansion

- The residue of a bubble

$$\Delta_{ij}(q) = b_0 + b_1\,(q\cdot e_2) + b_2\,(q\cdot e_2)^2 + b_3\,(q\cdot e_3) + b_4\,(q\cdot e_3)^2 + b_5\,(q\cdot e_4) + b_6\,(q\cdot e_4)^2 + b_7\,(q\cdot e_2)(q\cdot e_3) + b_8\,(q\cdot e_2)(q\cdot e_4) + b_9\,\mu^2$$

- solutions of a double cut $D_i = D_j = 0$, parametrized by the free variables t, x and μ^2

$$q_+ = x\,e_1 + (\alpha_0 + x\,\alpha_1)e_2 + t\,e_3 + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2\,t}\,e_4$$

$$q_- = x\,e_1 + (\alpha_0 + x\,\alpha_1)e_2 + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2\,t}\,e_3 + t\,e_4$$

- in the limit $t \to \infty$

$$\left.\frac{\mathcal{N}(q_\pm)}{\prod_{m\neq i,j} D_m}\right|_{\text{cut}} = \Delta_{ij} + \sum_k \frac{\Delta_{ijk}}{D_k} + \sum_{kl} \frac{\Delta_{ijkl}}{D_k D_l} + \sum_{klm} \frac{\Delta_{ijklm}}{D_k D_l D_m} = \Delta_{ij} + \sum_k \frac{\Delta_{ijk}}{D_k} + \mathcal{O}(1/t)$$

Example: One-loop bubbles via Laurent expansion

- In the asymptotic limit $t \to \infty$
  - the integrand

$$\left.\frac{\mathcal{N}(q_\pm)}{\prod_{m\neq i,j,k} D_m}\right|_{\text{cut}} = n_0^\pm + n_6^\pm\,\mu^2 + n_1^\pm\,x + n_2^\pm\,x^2 + \left(n_3^\pm + n_4^\pm x\right)t + n_5^\pm\,t^2 + \mathcal{O}(1/t)$$

  - the subtraction term

$$\frac{\Delta_{ijk}(q_\pm)}{D_k} = \tilde{b}_0^{k,\pm} + \tilde{b}_6^{k,\pm}\,\mu^2 + \tilde{b}_1^{k,\pm}\,x + \tilde{b}_2^{k,\pm}\,x^2 + \left(\tilde{b}_3^{k,\pm} + \tilde{b}_4^{k,\pm}x\right)t + \tilde{b}_5^{k,\pm}\,t^2 + \mathcal{O}(1/t)$$

    - $\tilde{b}_i^{k,\pm}$ are known functions of the triangle coefficients
  - the residue

$$\Delta_{ij}(q_+) = b_0 + b_9\,\mu^2 + b_1\,x + b_2 x^2 - \left(b_5 + b_8 x\right)t + b_6\,t^2 + \mathcal{O}(1/t)$$

$$\Delta_{ij}(q_-) = b_0 + b_9\,\mu^2 + b_1\,x + b_2 x^2 - \left(b_3 + b_7 x\right)t + b_4\,t^2 + \mathcal{O}(1/t)$$

  - by comparison, applying subtractions at the coefficient level

$$b_0 = n_0^\pm - \sum_k \tilde{b}_0^{k,\pm}, \quad b_1 = n_1^\pm - \sum_k \tilde{b}_1^{k,\pm}, \quad b_3 = -n_3^- + \sum_k \tilde{b}_3^{k,-}, \quad \ldots$$

Semi-numerical implementation in NINJA

- The input is the numerator $\mathcal{N}$ cast in (three or) four different forms
  - leading terms of parametric expansions of the numerator
  - coefficients of the expansion written to an array $\mathcal{N}$ `[]`
  - all easily (and very quickly) obtained from its analytic expression
- The PYTHON script `ninjanumgen` uses FORM-4 to
  - automatically compute expansions from a FORM expression of $\mathcal{N}$
  - generate optimized source code needed as input for NINJA
- NINJA at run-time
  - computes parametric on-shell solutions and Laurent expansions for every multiple cut
  - implements subtractions at coefficient level
  - multiplies the obtained coefficients with the MI's
- Semi-numeric Laurent expansion via polynomial division
  - expansion of numerator $\mathcal{N}$ `[]` / denominators D_i

Semi-numerical implementation in NINJA

```
// Numerator: can be generated using the script ninjanumgen
class MyNumerator : public ninja::Numerator {
public:

  // evaluates the numerator N(q, μ^2) - same as Samurai
  virtual Complex evaluate(q, μ^2, ...);

  // (optional) expansion for 4-ple cut rational term q^μ → v_⊥^μ + O(1)
  virtual Complex muExpansion(v_⊥, ...);

  // expansion for triangles and tadpoles q^μ → v_0^μ + t v_3^μ + (β + μ^2)/(2 t) v_4^μ
  virtual void t3Expansion(v_0, v_3, v_4, β, ..., Complex N[]);

  // expansion for bubbles q^μ → v_1^μ + x v_2^μ + t v_3^μ + (β_0 + β_1 x + β_2 x^2 + μ^2)/(2 t) v_4^μ
  virtual void t2Expansion(v_1, v_2, v_3, v_4, β_i, ..., Complex N[]);
};
```

—

note: `t2Expansion` is `t3Expansion` with: $v_0 \to v_1^\mu + x\, v_2^\mu,\ \beta \to \beta_0 + \beta_1 x + \beta_2 x^2$

Semi-numerical implementation in NINJA

Master Integrals:

- are called via a generic interface
  - ⇒ any user-defined library of Master Integrals can be used
- the library of MI's to be used can be specified at run time
- NINJA provides the interface for two default libraries
  - ONELOOP library [A. van Hameren] wrapper + caching
    - computed MI's are cached by NINJA
    - constant-time lookup from their arguments
  - LOOPTOOLS library [T. Hahn]
    - an internal cache is already present ⇒ interface is a simple wrapper

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 [P. Nogueira] and FORM [J. Vermaseren et al.]
- 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
  - GOLEM95 (tensor reduction)
    - slower than SAMURAI but more stable
    - default rescue-system for unstable points
  - NINJA
    - fast (2 to 5 times faster than SAMURAI)
    - stable (in worst cases $\mathcal{O}(1/1000)$ unstable points)

Benchmarks of GOSAM + NINJA

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

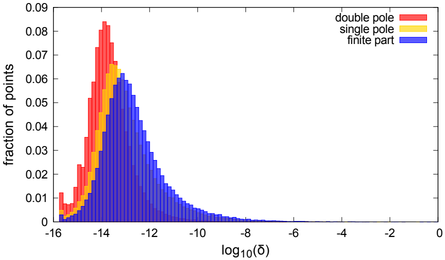
Benchmarks: GOSAM + NINJA			
Process		# NLO diagrams	ms/event[a]
$W+3j$	$d\bar{u} \to \bar{\nu}_e e^- ggg$	1 411	226
$Z+3j$	$d\bar{d} \to e^+ e^- ggg$	2 928	1 911
$t\bar{t}b\bar{b}\,(m_b \neq 0)$	$d\bar{d} \to t\bar{t}b\bar{b}$	275	178
	$gg \to t\bar{t}b\bar{b}$	1 530	5 685
$t\bar{t}+2j$	$gg \to t\bar{t}gg$	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

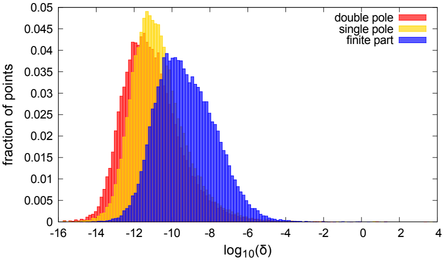
[a] Timings refer to full color- and helicity-summed amplitudes, using an Intel Core i7 CPU @ 3.40GHz, compiled with `ifort`.

Stability of NINJA

- $H + 4j$ in VBF ($u\bar{u} \to Hggu\bar{u}$)



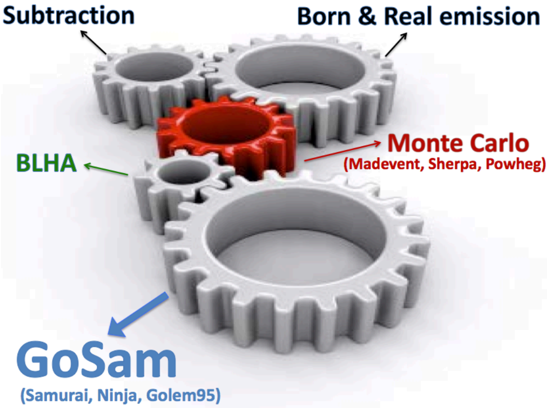
- $t\bar{t}H + 1j$ ($gg \to t\bar{t}Hg$)



Rate of unstable points, i.e. with error $\delta > \delta_{\text{threshold}}$ on the finite part:

$\delta_{\text{threshold}}$	$u\bar{u} \to Hggu\bar{u}$	$gg \to t\bar{t}Hg$
10^{-3}	0.02%	0.06%
10^{-4}	0.04%	0.16%
10^{-5}	0.08%	0.56%

From amplitudes to observables with GOSAM



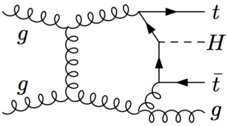
The GOSAM collaboration:

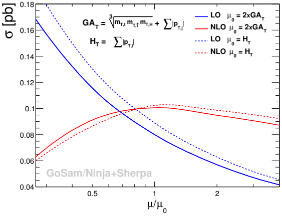
G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, J. Reichel , J. Schlenk, J. F. von Soden-Fraunhofen, T. Reiter, F. Tramontano, T.P.

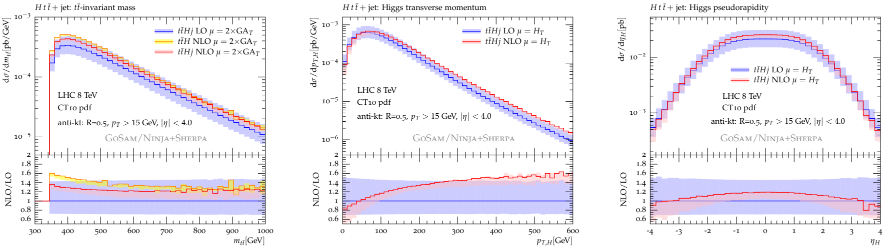
Application: $pp \to t\bar{t}H + jet$ with GOSAM + NINJA

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

- Interfaced with the Monte Carlo SHERPA







Summary and Outlook

- Summary of NINJA
  - implements the Integrand Reduction via Laurent expansion method
  - has good performance and stability
  - is already producing phenomenological results with GOSAM
  - will soon be public, both as standalone and within GOSAM-2.0

- Outlook
  - improve one-loop generation (recursion, global abbreviations,. . .)
  - treatment of (few) remaining unstable points within NINJA
  - alternative approach: fully automated algebraic one-loop

THANK YOU
FOR YOUR ATTENTION

BACKUP SLIDES

One-loop boxes via Laurent expansion

- The residue of a box reads

$$\Delta_{ijkl}(q,\mu^2) = d_0 + d_2\mu^2 + d_4\,\mu^4 + (d_1 + d_3\mu^2)(q\cdot v_\perp)$$

- d_0 via 4-dimensional 4ple cuts [Britto, Cachazo, Feng (2004)]
- d_4 from d-dimensional 4-ple cuts in the limit $\mu^2 \to \infty$ [S. Badger (2008)]
  - d-dimensional solutions of a 4-ple cut

$$q_\pm = a^\mu \pm \sqrt{\alpha + \frac{\mu^2}{\beta^2}}\, v_\perp^\mu = \pm\frac{\sqrt{\mu^2}}{\beta}\, v_\perp^\mu + \mathcal{O}(1)$$

  - the integrand in the asymptotic limit $\mu^2 \to \infty$ of the cut-solutions

$$\left.\frac{\mathcal{N}(q,\mu^2)}{\prod_{m\neq i,j,k,l} D_m}\right|_{\text{cut}} = d_4\,\mu^4 + \mathcal{O}(\mu^3)$$

- d_1, d_2, d_3 are spurious and do not need to be computed

One-loop triangles via Laurent expansion

- The residue of a triangle

$$\Delta_{ijk}(q) = c_0 + c_7\,\mu^2 + (c_1 + c_8\mu^2)\,(q\cdot e_3) + c_2\,(q\cdot e_3)^2 + c_3\,(q\cdot e_3)^3$$
$$+ (c_4 + c_9\mu^2)\,(q\cdot e_4) + c_5\,(q\cdot e_4)^2 + c_6\,(q\cdot e_4)^3$$

- solutions of a triple cut $D_i = D_j = D_k = 0$ parametrized by the free variables t and μ^2

$$q_+^\mu = a^\mu + t\,e_3^\mu + \frac{\alpha+\mu^2}{2\,t}\,e_4^\mu, \qquad q_-^\mu = a^\mu + \frac{\alpha+\mu^2}{2\,t}\,e_3^\mu + t\,e_4^\mu$$

- in the limit $t \to \infty$ [Forde (2007)]

$$\left.\frac{\mathcal{N}(q_\pm)}{\prod_{m\neq i,j,k} D_m}\right|_{\text{cut}} = \Delta_{ijk} + \sum_l \frac{\Delta_{ijkl}}{D_l} + \sum_{lm}\frac{\Delta_{ijklm}}{D_l D_m}$$
$$= \Delta_{ijk} + d_1^\pm + d_2^\pm\,\mu^2 + \mathcal{O}(1/t)$$

with $d_i^+ + d_i^- = 0$

One-loop triangles via Laurent expansion

- In the asymptotic limit $t \to \infty$

$$\left.\frac{\mathcal{N}(q_\pm)}{\prod_{m\neq i,j,k} D_m}\right|_{\text{cut}} = (d_1^\pm + d_2^\pm\,\mu^2) + \Delta_{ijk} + \mathcal{O}(1/t) \qquad \text{with } d_i^+ + d_i^- = 0$$

  - the integrand

$$\left.\frac{\mathcal{N}(q_\pm)}{\prod_{m\neq i,j,k} D_m}\right|_{\text{cut}} = n_0^\pm + n_4^\pm\,\mu^2 + (n_1^\pm + n_5^\pm\,\mu^2)\,t + n_2^\pm\,t^2 + n_3^\pm\,t^3 + \mathcal{O}(1/t)$$

  - the residue

$$\Delta_{ijk}(q_+) = c_0 + c_7\,\mu^2 - (c_4 + c_9\,\mu^2)\,t + c_5\,t^2 - c_6\,t^3 + \mathcal{O}(1/t)$$
$$\Delta_{ijk}(q_-) = c_0 + c_7\,\mu^2 - (c_1 + c_8\,\mu^2)\,t + c_2\,t^2 - c_3\,t^3 + \mathcal{O}(1/t)$$

  - by comparison we get

$$c_0 = \frac{n_0^+ + n_0^-}{2}, \quad c_1 = -n_1^-, \quad c_2 = n_2^-, \quad c_3 = -n_3^-, \quad \ldots$$

Rotation method for error estimation

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

- Definitions

A :	numerical result for the amplitude
A_{rot} :	numerical result for the amplitude with rotated kinematics
A_{ex} :	exact result for the amplitude $\sim$ amplitude in quad. prec.

- the exact error is defined as

$$\delta_{ex} = \left| \frac{A_{ex} - A}{A_{ex}} \right|$$

- the estimated error is defined as

$$\delta_{rot} = 2 \left| \frac{A_{rot} - A}{A_{rot} + A} \right|$$

- one can check that $\delta_{rot} \sim \delta_{ex}$

Rotation method for error estimation

A validation of the rotation method

- example: $W\,b\,\bar{b}+1\,j\ \ (u\bar{d}\to e^{+}\nu_e b\bar{b}g)$, with $m_b \neq 0$