Integrand reduction techniques at one and higher loops

Tiziano Peraro

Max-Planck-Institut für Physik, München, Germany

Higgs Centre for Theoretical Physics – Edinburgh
9 May 2014
Outline

1. Introduction and motivation
2. Integrand reduction via polynomial division
3. Application at one-loop
4. Integrand reduction via Laurent expansion (NINJA)
5. Higher loops
6. Summary and Outlook
Motivation

- Theoretical understanding of **scattering amplitudes**
  - basic **analytic/algebraic structure** of loop integrands and integrals
- Need of **theoretical predictions** for colliders (LHC)
  - probing large phase space $\Rightarrow$ several **external legs**
  - need of NLO or higher accuracy $\Rightarrow$ computations at the **loop level**
- **Automation** of methods for predictions in perturbative QFT

We developed a coherent framework for the **integrand decomposition** of Feynman integrals

- based on simple concepts of **algebraic geometry**
- applicable at all loops
Integrand reduction

- The integrand of a generic $\ell$-loop integral:
  - is a rational function in the components of the loop momenta $\vec{q}_i$
  - polynomial numerator $N_{i_1 \ldots i_n}$

$$\mathcal{M}_n = \int d^d \vec{q}_1 \cdots d^d \vec{q}_\ell \quad \mathcal{I}_{i_1 \ldots i_n}, \quad \mathcal{I}_{i_1 \ldots i_n} \equiv \frac{N_{i_1 \ldots i_n}}{D_{i_1} \cdots D_{i_n}}$$

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

$$D_i = \left( \sum_j (-)^{s_{ij}} \vec{q}_j + p_i \right)^2 - m_i^2$$

$$\vec{q}_i = q_i + \vec{\mu}_i$$

$d$-dimensional $4$-dimensional $(-2\epsilon)$-dimensional

$$\vec{q}_i \cdot \vec{q}_j = (q_i \cdot q_j) - \mu_{ij}$$
Integrand reduction

The idea

Manipulate the integrand and reduce it to a linear combination of “simpler” integrands.

- The integrand-reduction algorithm leads to

\[ \mathcal{I}_{i_1 \cdots i_n} \equiv \frac{\mathcal{N}_{i_1 \cdots i_n}}{D_{i_1} \cdots D_{i_n}} = \frac{\Delta_{i_1 \cdots i_n}}{D_{i_1} \cdots D_{i_n}} + \cdots + \sum_{k=1}^{n} \frac{\Delta_{i_k}}{D_{i_k}} + \Delta_\emptyset \]

- The residues \( \Delta_{i_1 \cdots i_k} \) are irreducible polynomials in \( \bar{q}_i \)
  - can’t be written as a combination of denominators \( D_{i_1}, \ldots, D_{i_k} \)
  - universal topology-dependent parametric form
  - the coefficients of the parametrization are process-dependent
From integrands to integrals

- By integrating the integrand decomposition

\[ \mathcal{M}_n = \int d^d \bar{q}_1 \cdots d^d \bar{q}_\ell \left( \frac{\Delta_{i_1\cdots i_n}}{D_{i_1} \cdots D_{i_n}} + \cdots + \sum_{k=1}^{n} \frac{\Delta_{i_k}}{D_{i_k}} + \Delta_\emptyset \right) \]

- some terms vanish and do not contribute to the amplitude
  ⇒ spurious terms
- non-vanishing terms give Master Integrals (MIs)

- The amplitude is a linear combination of MIs
- The coefficients of this linear combination can be identified with some of the coefficients which parametrize the polynomial residues
From integrands to integrals

- By integrating the integrand decomposition

\[
\mathcal{M}_n = \int d^d \vec{q}_1 \cdots d^d \vec{q}_\ell \left( \frac{\Delta_{i_1 \cdots i_n}}{D_{i_1} \cdots D_{i_n}} + \cdots + \sum_{k=1}^{n} \frac{\Delta_{i_k}}{D_{i_k}} + \Delta_\emptyset \right)
\]

- 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 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 one-loop decomposition

At one loop the result is well known:

- the **integrand** decomposition
  
  $\mathcal{I}_{i_1 \ldots i_n} = \frac{N_{i_1 \ldots i_n}}{D_{i_1} \cdots D_{i_n}} = \sum_{j_1 \ldots j_5} \frac{\Delta_{j_1 j_2 j_3 j_4 j_5}}{D_{j_1} D_{j_2} D_{j_3} D_{j_4} D_{j_5}} + \sum_{j_1 j_2 j_3 j_4} \frac{\Delta_{j_1 j_2 j_3 j_4}}{D_{j_1} D_{j_2} D_{j_3} D_{j_4}}$

  
  $+ \sum_{j_1 j_2 j_3} \frac{\Delta_{j_1 j_2 j_3}}{D_{j_1} D_{j_2} D_{j_3}} + \sum_{j_1 j_2} \frac{\Delta_{j_1 j_2}}{D_{j_1} D_{j_2}} + \sum_{j_1} \frac{\Delta_{j_1}}{D_{j_1}}$

- the **integral** decomposition

  $= c_{4,0} + c_{3,0} + c_{2,0} + c_{1,0}$

  $+ c_{4,4}^{d+4} + c_{3,7}^{d+2} + c_{2,9}^{d+2}$

- all the Mater Integrals are known!
Integrand reduction and polynomials

- At $\ell$-loops we want to achieve the integrand decomposition:

$$ I_{i_1 \ldots i_n}(\bar{q}_1, \ldots, \bar{q}_\ell) \equiv \frac{N_{i_1 \ldots i_n}}{D_{i_1} \cdots D_{i_n}} = \frac{\Delta_{i_1 \ldots i_n}}{D_{i_1} \cdots D_{i_n}} + \cdots + \sum_{k=1}^{n} \frac{\Delta_{i_k}}{D_{i_k}} + \Delta_{\emptyset} $$

they must be irreducible

- We trade $(\bar{q}_1, \ldots, \bar{q}_\ell)$ with their coordinates $z \equiv (z_1, \ldots, z_m)$

  ⇒ numerator and denominators ≡ polynomials in $z$

$$ I_{i_1 \ldots i_n}(z) \equiv \frac{N_{i_1 \ldots i_n}(z)}{D_{i_1}(z) \cdots D_{i_n}(z)} $$

⇒ Integrand reduction ≡ problem of multivariate polynomial division

The problem of the determination of the residues of a generic diagram has been solved. [Y. Zhang (2012), P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2012-14)]
Residues via polynomial division


- Define the **Ideal** of polynomials

\[ \mathcal{J}_{i_1 \cdots i_n} \equiv \langle D_{i_1}, \ldots, D_{i_n} \rangle = \left\{ p(z) : p(z) = \sum_j h_j(z)D_j(z), \ h_j \in P[z] \right\} \]

- Take a Gröbner basis \( G_{\mathcal{J}_{i_1 \cdots i_n}} \) of \( \mathcal{J}_{i_1 \cdots i_n} \)

\[ G_{\mathcal{J}_{i_1 \cdots i_n}} = \{ g_1, \ldots, g_s \} \quad \text{such that} \quad \mathcal{J}_{i_1 \cdots i_n} = \langle g_1, \ldots, g_s \rangle \]

- Perform the **multivariate polynomial division** \( \mathcal{N}_{i_1 \cdots i_n} / G_{\mathcal{J}_{i_1 \cdots i_n}} \)

\[ \mathcal{N}_{i_1 \cdots i_n}(z) = \sum_{k=1}^{n} \mathcal{N}_{i_1 \cdots i_{k-1}i_{k+1} \cdots i_n}(z)D_{i_k}(z) + \Delta_{i_1 \cdots i_n}(z) \]

\[ \underbrace{\text{quotient } \in \mathcal{J}_{i_1 \cdots i_n}}_{\text{remainder}} \]

- The remainder \( \Delta_{i_1 \cdots i_n} \) is **irreducible** \( \Rightarrow \) can be identified with the **residue**
Integrand reduction via polynomial division

Recursive Relation for the integrand decomposition


The recursive formula

\[
N_{i_1 \ldots i_n} = \sum_{k=1}^{n} N_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_n} D_{i_k} + \Delta_{i_1 \ldots i_n}
\]

\[
I_{i_1 \ldots i_n} \equiv \frac{N_{i_1 \ldots i_n}}{D_{i_1} \ldots D_{i_n}} = \sum_k I_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_n} + \frac{\Delta_{i_1 \ldots i_n}}{D_{i_1} \ldots D_{i_n}}
\]

- **Fit-on-the-cut approach**
  - from a generic \( N \), get the **parametric form** of the residues \( \Delta \)
  - determine the **coefficients** sampling on the cuts (impose \( D_i = 0 \))

- **Divide-and-Conquer approach**
  - generate the \( N \) of the process
  - compute the residues by **iterating** the polynomial division algorithm
Fit-on-the-cut approach

[Ossola, Papadopoulos, Pittau (2007)]

The decomposition of the numerator

\[ N_{i_1 \cdots i_n} = \sum_{k=0}^{n} \sum_{\{j_1 \cdots j_k\}} \Delta_{j_1 \cdots j_k} \prod_{h \in \{i_1 \cdots i_n\} \setminus \{j_1 \cdots j_k\}} D_h. \]

- Fit the coefficients of the residues sampling on the multiple cuts
- First step: $n$-ple cut
  - impose $D_{i_1} = \cdots = D_{i_n} = 0$
  - \[ \Delta_{i_1 \cdots i_n} = N_{i_1 \cdots i_n} \]
- Further steps: $k$-ple cut
  - impose $D_{i_1} = \cdots = D_{i_k} = 0$ for any subset $\{i_1 \cdots i_k\}$
  - \[ \Delta_{i_1 \cdots i_k} = \frac{N_{i_1 \cdots i_n} - \text{higher-point contributions}}{\prod_{h \neq i_1, \ldots, i_k} D_h} \]
Fit-on-the-cut approach: The reducibility criterion

What happens if a cut has no solution?

The reducibility criterion

- If a cut $D_{i_1} = \cdots = D_{i_k} = 0$ has no solutions, the associated residue vanishes. In other words, any numerator is completely reducible.
- This generally happens with overdetermined systems i.e. when the number of cut denominators is higher than the one of loop coordinates.

- When $D_{i_1} = \cdots = D_{i_k} = 0$ has no solution:

$$
\Delta_{i_1 \ldots i_k} = 0 \quad \Rightarrow \text{no need to perform the fit}
$$

$$
\mathcal{N}_{i_1 \ldots i_n} = \sum_{k=1}^{n} \mathcal{N}_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_n} D_{i_k}
$$

$$
\mathcal{I}_{i_1 \ldots i_n} = \sum_{k} \mathcal{I}_{i_1 \ldots i_{k-1} i_{k+1} \ldots i_n}
$$
The maximum-cut theorem

- We define maximum-cut, a cut where
  
  \#(cut-denominators) \equiv \#(components-of-loop-momenta)

- In non-special kinematic configurations it has a finite number of solutions
  
  \#(coefficients-of-the-residue) = \#(solutions-of-the-cut)

- The fit-on-the-cut approach therefore gives a number of equations which is equal to the number of unknown coefficients.
Fit-on-the-cut approach: The maximum-cut theorem

Examples:

<table>
<thead>
<tr>
<th>diagram</th>
<th>$\Delta$</th>
<th>$n_s$</th>
<th>diagram</th>
<th>$\Delta$</th>
<th>$n_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td>$c_0$</td>
<td>1</td>
<td><img src="image2" alt="Diagram" /></td>
<td>$c_0 + c_1 z$</td>
<td>2</td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td>$\sum_{i=0}^{3} c_i z^i$</td>
<td>4</td>
<td><img src="image4" alt="Diagram" /></td>
<td>$\sum_{i=0}^{3} c_i z^i$</td>
<td>4</td>
</tr>
<tr>
<td><img src="image5" alt="Diagram" /></td>
<td>$\sum_{i=0}^{7} c_i z^i$</td>
<td>8</td>
<td><img src="image6" alt="Diagram" /></td>
<td>$\sum_{i=0}^{7} c_i z^i$</td>
<td>8</td>
</tr>
</tbody>
</table>
Fit-on-the-cut approach

Pros:
- each multiple cut projects out the corresponding residue
- the systems of equations for the coefficients are much smaller
- can be implemented either analytically or numerically
- very successful application at one-loop

Cons:
- at higher-loops the solutions of the cuts can be difficult to find
- it cannot be applied in to all integrands/topologies
  - if we have e.g. quadratic propagators the formula yields
    \[
    \frac{\mathcal{N}_{i_1 \ldots i_n} - \text{higher-point contributions}}{\prod_{h \neq i_1, \ldots, i_k} D_h} = \frac{0}{0}
    \]
Fit-on-the-cut approach

Pros:
- each multiple cut projects out the corresponding residue
  ⇒ the systems of equations for the coefficients are much smaller
- can be implemented either analytically or numerically
- very successful application at one-loop

Cons:
- at higher-loops the solutions of the cuts can be difficult to find
- it cannot be applied to all integrands/topologies
- if we have e.g. quadratic propagators the formula yields

\[ n \neq i_1, \ldots, i_k \]

\[ \prod_{h \neq i_1, \ldots, i_k} D_h = 0 \]

**OBSERVATION:** these issues are not present in the divide-and-conquer approach which instead can be applied to any integrand
One-loop decomposition from polynomial division


- Start from the most general one-loop amplitude in \( d = 4 - 2\epsilon \)
- Apply the recursive formula for the integrand decomposition
  \( \Rightarrow \) it reproduces the OPP result
  \[ \text{[Ossola, Papadopoulos, Pittau (2007); Ellis, Giele, Kunszt, Melnikov (2008)]} \]
- Drop the spurious terms
  \( \Rightarrow \) Get the most general integral decomposition (well known result)

\[
\begin{align*}
\text{Diagram} & = c_{4,0} + c_{3,0} + c_{2,0} + c_{1,0} \\
& + c_{4,4}^{d+4} + c_{3,7}^{d+2} + c_{2,9}^{d+2}
\end{align*}
\]
One-loop decomposition from polynomial division

At one loop in $4 - 2\epsilon$ dimensions:

- **5 coordinates** $\mathbf{z} = (z_1, z_2, z_3, z_4, z_5)$
- **4 components** $(z_1, z_2, z_3, z_4)$ of $q$ w.r.t. a 4-dimensional basis
- $z_5 = \mu^2$ encodes the $(-2\epsilon)$-dependence on the loop momentum

- **we start with**
  \[
  \mathcal{I}_n \equiv \mathcal{I}_{1\cdots n} = \frac{\mathcal{N}_{1\cdots n}(\mathbf{z})}{D_1(\mathbf{z}) \cdots D_n(\mathbf{z})}
  \]
  most general 1-loop numerator
  generic 1-loop denominators

- **if** $m > 5$ any integrand $\mathcal{I}_{i_1\cdots i_m}$ is reducible (reducibility criterion)
  \[
  \mathcal{I}_{i_1\cdots i_m} = \sum_k \mathcal{I}_{i_1\cdots i_{k-1}i_{k+1}\cdots i_m} \quad \Rightarrow \quad \Delta_{i_1\cdots i_m} = 0 \quad \text{for} \quad m > 5
  \]

- **for** $m \leq 5$ the polynomial-division algorithm gives the already-known parametric form of the residues $\Delta_{ijk\cdots}$
Choice of 4-dimensional basis for an \( m \)-point residue

\[
e_1^2 = e_2^2 = 0, \quad e_1 \cdot e_2 = 1, \quad e_3^2 = e_4^2 = \delta_{m4}, \quad e_3 \cdot e_4 = -(1 - \delta_{m4})
\]

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

\[
q^\mu = -p_{i1}^\mu + x_1 e_1^\mu + x_2 e_2^\mu + x_3 e_3^\mu + x_4 e_4^\mu, \quad \bar{q}^2 = q^2 - \mu^2
\]

Generic numerator

\[
N_{i_1 \ldots i_m} = \sum_{j_1, \ldots, j_5} \alpha_j z_1^{j_1} z_2^{j_2} z_3^{j_3} z_4^{j_4} z_5^{j_5}, \quad (j_1 \ldots j_5) \quad \text{such that} \quad \text{rank}(N_{i_1 \ldots i_m}) \leq m
\]

Residues

\[
\Delta_{i_1 i_2 i_3 i_4 i_5} = c_0
\]
\[
\Delta_{i_1 i_2 i_3 i_4} = c_0 + c_1 x_4 + \mu^2 (c_2 + c_3 x_4 + \mu^2 c_4)
\]
\[
\Delta_{i_1 i_2 i_3} = c_0 + c_1 x_3 + c_2 x_3^2 + c_3 x_3^3 + c_4 x_4 + c_5 x_4^2 + c_6 x_4^3 + \mu^2 (c_7 + c_8 x_3 + c_9 x_4)
\]
\[
\Delta_{i_1 i_2} = c_0 + c_1 x_2 + c_2 x_3 + c_3 x_4 + c_4 x_2^2 + c_5 x_3^2 + c_6 x_4^2 + c_7 x_2 x_3 + c_9 x_2 x_4 + c_9 \mu^2
\]
\[
\Delta_{i_1} = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4
\]

It can be easily extended to higher-rank numerators
Fit-on-the-cut at 1-loop

Integrand decomposition:

\[
\sum + \sum + \sum + \sum = \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum + \sum
\]

Fit-on-the-cut

fit \(m\)-point residues on \(m\)-ple cuts
The integrand reduction via Laurent expansion:
[P. Mastrolia, E. Mirabella, T.P. (2012)]

- fits residues by taking their asymptotic expansions on the cuts
- 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)

The integrand reduction via Laurent expansion:
[P. Mastrolia, E. Mirabella, T.P. (2012)]

- fits residues by taking their asymptotic expansions on the cuts
- 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
- interface with FORMCALC [T. Hahn et al.] under development
- is a faster and more stable integrand-reduction algorithm
The integrand reduction via Laurent expansion:
[P. Mastrolia, E. Mirabella, T.P. (2012)]

- fits residues by taking their asymptotic expansions on the cuts
- 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
- interface with FORMCALC [T. Hahn et al.] under development
- is a faster and more stable integrand-reduction algorithm

★ NINJA is public ⇒ ninja.hepforge.org
Integrand decomposition:

\[ \sum = \sum + \sum + \sum + \sum + \sum \]

Laurent-expansion method
Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:

Laurent-expansion method

- pentagons not needed
Integrand reduction via Laurent expansion (\textsc{NINJA})

Integrand decomposition:

\[
\sum = \sum + \sum + \sum
\]

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

- the integrand in the asymptotic limit \(\mu^2 \to \infty\) of the cut-solutions
  \[
  \left. \frac{\mathcal{N}(q, \mu^2)}{\prod_{m \neq i,j,k,l} D_m} \right|_{\text{cut}} = d_4 \mu^4 + O(\mu^3)
  \]

- \(d_1, d_2, d_3\) 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 \( \mu^2 \)

\[ q^\mu_+ = a^\mu + t e_3^\mu + \frac{\alpha + \mu^2}{2t} e_4^\mu, \quad q^\mu_- = a^\mu + \frac{\alpha + \mu^2}{2t} e_3^\mu + t e_4^\mu \]

in the limit \( t \to \infty \)

\[ \left. \frac{\mathcal{N}(q_\pm)}{\prod_{m \neq i,j,k} D_m} \right|_{\text{cut}} = \Delta_{ijk} + \sum_l \frac{\Delta_{ijkl}}{D_l} + \sum_{lm} \frac{\Delta_{ijklm}}{D_lD_m} \]

\[ = \Delta_{ijk} + d_i^+ + d_i^- \mu^2 + \mathcal{O}(1/t) \]

with \( d_i^+ + d_i^- = 0 \)
One-loop triangles via Laurent expansion

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

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

- the integrand

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

- 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

\[
c_0 = \frac{n_0^+ + n_0^-}{2}, \quad c_1 = -n_1^-, \quad c_2 = n_2^-, \quad c_3 = -n_3^-, \ldots
\]
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 \(\mu^2\)
  \[ q_+ = x e_1 + (\alpha_0 + x \alpha_1) e_2 + t e_3 + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2 t} e_4 \]
  \[ q_- = x e_1 + (\alpha_0 + x \alpha_1) e_2 + \frac{\beta_0 + \beta_1 x + \beta_2 x^2 + \mu^2}{2 t} e_3 + t e_4 \]

- in the limit \(t \to \infty\)
  \[ \left. \frac{\mathcal{N}(q_{\pm})}{\prod_{m \neq i,j} D_m} \right|_{\text{cut}} = \Delta_{ij} + \sum_k \frac{\Delta_{ijk}}{D_k} + \sum_{kl} \frac{\Delta_{ijkl}}{D_k D_l} + \sum_{klm} \frac{\Delta_{ijklm}}{D_k D_l D_m} = \Delta_{ij} + \sum_k \frac{\Delta_{ijk}}{D_k} + O(1/t) \]
One-loop bubbles via Laurent expansion

- In the asymptotic limit $t \to \infty$
  - the integrand
    \[
    \frac{\mathcal{N}(q^\pm)}{\prod_{m \neq i,j,k} D_m} \bigg|_{\text{cut}} = n_0^\pm + n_6^\pm \mu^2 + n_1^\pm x + n_2^\pm x^2 + \left( n_3^\pm + n_4^\pm x \right) t + n_5^\pm t^2 + \mathcal{O}(1/t)
    \]
  - 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 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
  - performs Laurent expansions via pol. div.
  - 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, µ²) – same as Samurai
    virtual Complex evaluate(q, µ², ...);

    // (optional) expansion for 4-ple cut rational term q⁻→ t v₂ + O(1)
    virtual void muExpansion(v₂,..., Complex N);

    // expansion for triangles and tadpoles q⁻→ v₀ + t v₃ + β + µ²/2 t v₄
    virtual void t3Expansion(v₀, v₃, v₄, β,..., Complex N[]);

    // expansion for bubbles q⁻→ v₁ + x v₂ + t v₃ + β₀ + β₁ x + β₂ x² + µ²/2 t v₄
    virtual void t2Expansion(v₁, v₂, v₃, v₄, βᵢ,..., Complex N[]);
};

```

Note: t2Expansion is t3Expansion with: v₀ → v₁ + x v₂, β → β₀ + β₁ x + β₂ x²
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

Higher-rank:
- support for higher-rank $r = n + 1$
- higher-rank MI’s (can but) do not need to be provided
Automaton 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
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
    - former default in GoSAM-1.0
  - 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 $O(1/1000)$ unstable points)
    - current default in GoSAM-2.0 ← just released
### Benchmarks of GoSAM + Ninja


<table>
<thead>
<tr>
<th>Process</th>
<th># NLO diagrams</th>
<th>ms/event(^a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(W + 3j)</td>
<td>(d\bar{u} \rightarrow \bar{\nu}_e e^- ggg)</td>
<td>1 411</td>
</tr>
<tr>
<td>(Z + 3j)</td>
<td>(d\bar{d} \rightarrow e^+ e^- ggg)</td>
<td>2 928</td>
</tr>
<tr>
<td>(\bar{t}\bar{b}\bar{b} (m_b \neq 0))</td>
<td>(d\bar{d} \rightarrow \bar{t}\bar{b}\bar{b})</td>
<td>275</td>
</tr>
<tr>
<td></td>
<td>(gg \rightarrow \bar{t}\bar{b}\bar{b})</td>
<td>1 530</td>
</tr>
<tr>
<td>(\bar{t} + 2j)</td>
<td>(gg \rightarrow \bar{t}gg)</td>
<td>4 700</td>
</tr>
<tr>
<td>(Wb\bar{b} + 1j (m_b \neq 0))</td>
<td>(u\bar{d} \rightarrow e^+ \nu_e b\bar{b}g)</td>
<td>312</td>
</tr>
<tr>
<td>(Wb\bar{b} + 2j (m_b \neq 0))</td>
<td>(u\bar{d} \rightarrow e^+ \nu_e b\bar{b}s)</td>
<td>648</td>
</tr>
<tr>
<td></td>
<td>(u\bar{d} \rightarrow e^+ \nu_e b\bar{b}d)</td>
<td>1 220</td>
</tr>
<tr>
<td></td>
<td>(u\bar{d} \rightarrow e^+ \nu_e b\bar{b}gg)</td>
<td>3 923</td>
</tr>
<tr>
<td>(H + 3j) in GF</td>
<td>(gg \rightarrow Hggg)</td>
<td>9 325</td>
</tr>
<tr>
<td>(i\bar{t} H + 1j)</td>
<td>(gg \rightarrow \bar{t}Hg)</td>
<td>1 517</td>
</tr>
<tr>
<td>(H + 3j) in VBF</td>
<td>(u\bar{u} \rightarrow Hgu\bar{u})</td>
<td>432</td>
</tr>
<tr>
<td>(H + 4j) in VBF</td>
<td>(u\bar{u} \rightarrow Hggu\bar{u})</td>
<td>1 176</td>
</tr>
<tr>
<td>(H + 5j) in VBF</td>
<td>(u\bar{u} \rightarrow Hgguu\bar{u})</td>
<td>15 036</td>
</tr>
</tbody>
</table>

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 ($u\bar{u} \rightarrow Hggu\bar{u}$)
- $t\bar{t}H + 1j$ ($gg \rightarrow t\bar{t}Hg$)

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

<table>
<thead>
<tr>
<th>$\delta_{\text{threshold}}$</th>
<th>$u\bar{u} \rightarrow Hggu\bar{u}$</th>
<th>$gg \rightarrow t\bar{t}Hg$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>0.02%</td>
<td>0.06%</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>0.04%</td>
<td>0.16%</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>0.08%</td>
<td>0.56%</td>
</tr>
</tbody>
</table>
From amplitudes to observables with GoSAM

The GoSAM collaboration:

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


- Interfaced with the Monte Carlo SHERPA
Integrand reduction via Laurent expansion (NINJA)

**Application:** $pp \rightarrow H + jets$ in GF with GOSAM + NINJA

- $m_t \rightarrow \infty$ approximation
- Effective couplings $H + (2, 3, 4)gl.$
- Higher-rank integrands $\Rightarrow$ extension of int. red. methods

- $H + 2j$ (GOSAM + SAMURAI + SHERPA)

- $H + 3j$ (GOSAM + SAMURAI + SHERPA + MADGRAPH4/MADEVENT)

- **new analysis with ATLAS-like cuts, using NINJA for the reduction**
Application: \( pp \rightarrow H + jets \) in GF with \( \text{GOSam} + \text{NINJA} \)

- new distributions using \text{NINJA} (preliminary)
  - better accuracy
  - better performance

\[
\mu_F = \mu_R = \frac{\hat{H}_T}{2} = \frac{1}{2} \left( \sqrt{m_H^2 + p_{t,H}^2} + \sum_{\text{jets}} \left| p_{t,jet} \right|^2 \right)
\]

- ATLAS-like cuts

\[
R = 0.4, \quad p_{t,jet} > 30\text{GeV}, \quad |\eta_{\text{jet}}| < 4.4
\]

- total cross section

\[
\sigma_{\text{LO}}^{(H+2j)}([\text{pb}]) = 1.23^{+37\%}_{-24\%}, \quad \sigma_{\text{LO}}^{(H+3j)}([\text{pb}]) = 0.381^{+53\%}_{-32\%}
\]

\[
\sigma_{\text{NLO}}^{(H+2j)}([\text{pb}]) = 1.590^{-4\%}_{-7\%}, \quad \sigma_{\text{NLO}}^{(H+3j)}([\text{pb}]) = 0.485^{-3\%}_{-13\%}
\]
Application: $pp \rightarrow H + jets$ in GF with GOSAM + NINJA

- new distributions using NINJA (preliminary)
  - better accuracy
  - better performance

$$
\mu_F = \mu_R = \frac{\hat{H}_T}{2} = \frac{1}{2} \left( \sqrt{m_H^2 + p_{T,H}^2} + \sum_{jets} |p_{T,jet}|^2 \right)
$$

$LHC 8 \text{ TeV}$

anti-kt: $R=0.4$, $p_T > 30 \text{ GeV}$, $|\eta| < 4.4$

$H + 3 \text{ jets}$: Higgs transverse momentum

$H + 3 \text{ jets}$: Higgs rapidity

T. Peraro (MPI - München)
Application: \( pp \rightarrow H + jets \) in GF with GoSAM + Ninja

- new distributions using Ninja (preliminary)
  - better accuracy
  - better performance

\[ \mu_F = \mu_R = \frac{\hat{H}_T}{2} = \frac{1}{2} \left( \sqrt{m_H^2 + p_{t,H}^2} + \sum_{\text{jets}} |p_{t,jet}|^2 \right) \]

\[ d\sigma/dp_{t,H} \quad \text{(pb/GeV)} \]

\[ \frac{d\sigma}{dy_H} \quad \text{(pb)} \]
Extension to higher loops

- The integrand-level approach to scattering amplitudes at one-loop
  - can be used to compute any amplitude in any QFT
  - has been implemented in several codes, some of which public
    [Samurai, CutTools, Ninja]
  - has produced (and is still producing) results for LHC
    [GoSam, FormCalc, BlackHat, MadLoop, NJets, OpenLoop ...]

- At two or higher loops
  - no general recipe is available
  - the standard and most successful approach is the Integration By Parts (IBP) method, but it becomes difficult for high multiplicities
Extension to higher loops

- The integrand-level approach to scattering amplitudes at one-loop can be used to compute any amplitude in any QFT.
  - has been implemented in several codes, some of which public
    - [SAMURAI, CUTFOLS, NINJA]
  - has produced (and is still producing) results for LHC
    - [GoSam, FORMCALC, BLACKHAT, MADLOOP, NJETS, OpenLoop…]

- At two or higher loops
  - no general recipe is available
  - the standard and most successful approach is the Integration By Parts (IBP) method, but it becomes difficult for high multiplicities.

The integrand-level approach might be a tool for understanding the structure of multi-loop scattering amplitudes and a method for their evaluation.
Extension to higher loops

- The integrand-level approach to scattering amplitudes at one-loop can be used to compute any amplitude in any QFT.
- It has been implemented in several codes, some of which are public:
  - [SAMURAI, CUTTOOLS, NINJA]
  - [GOSAM, FORMCALC, BLACKHAT, MADLOOP, NJETS, OPENLOOP...]

- At two or higher loops:
  - No general recipe is available.
  - The standard and most successful approach is the Integration By Parts (IBP) method, but it becomes difficult for high multiplicities.

The integrand-level approach might be a tool for understanding the structure of multi-loop scattering amplitudes and a method for their evaluation.

...we are moving the first steps in this direction.
\( \mathcal{N} = 4 \) SYM and \( \mathcal{N} = 8 \) SUGRA amplitudes


- Examples in \( \mathcal{N} = 4 \) SYM and \( \mathcal{N} = 8 \) SUGRA amplitudes \((d = 4)\)
  - generation of the integrand
    - graph based [Carrasco, Johansson (2011)]
    - unitarity based [U. Schubert (Diplomarbeit)]
  - fit-on-the-cut approach for the reduction

- Results:
  \( \mathcal{N} = 4 \) linear combination of 8 and 7-denominators MIs
  \( \mathcal{N} = 8 \) linear combination of 8, 7 and 6-denominators MIs
Divide-and-Conquer approach


The divide-and-conquer approach to the integrand reduction

- does not require the knowledge of the solutions of the cut
- can always be used to perform the reduction in a finite number of purely algebraic operations
- has been automated in a PYTHON package which uses MACAULAY2 and FORM for algebraic operations

also works in special cases where the fit-on-the-cut approach is not applicable (e.g. in presence of double denominators)
Divide-and-Conquer approach: a simple example

\[ I_{11234} = \frac{N_{11234}}{D_1 D_2 D_3 D_4} \]

- \( D_1 = \bar{q}_1^2 - m^2 \)
- \( D_2 = (\bar{q}_1 - k)^2 - m^2 \)
- \( D_3 = \bar{q}_2^2 \)
- \( D_4 = (\bar{q}_1 + \bar{q}_2)^2 - m^2 \)
Divide-and-Conquer approach: a simple example

\[ I_{1234} = \frac{N_{11234}}{D_1^2D_2D_3D_4} \]

\( D_1 = \bar{q}_1^2 - m^2 \),
\( D_2 = (\bar{q}_1 - k)^2 - m^2 \),
\( D_3 = \bar{q}_2^2 \),
\( D_4 = (\bar{q}_1 + \bar{q}_2)^2 - m^2 \)

- Basis \( \{ e_i \} \equiv \{ k, k_\perp, e_3, e_4 \} \) and coordinates \( z = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, \mu_{11}, \mu_{12}, \mu_{22}) \)

\[ q_1 = \sum_i x_i e_i, \quad q_2 = \sum_i y_i e_i, \quad (\bar{q}_i \cdot \bar{q}_j) = (q_i \cdot q_j) - \mu_{ij} \]
Divide-and-Conquer approach: a simple example

\[ \mathcal{I}_{11234} = \frac{\mathcal{N}_{11234}}{D_1^2 D_2 D_3 D_4} \]

\[ D_1 = \bar{q}_1^2 - m^2 , \]
\[ D_2 = (\bar{q}_1 - k)^2 - m^2 , \]
\[ D_3 = \bar{q}_2^2 , \]
\[ D_4 = (\bar{q}_1 + \bar{q}_2)^2 - m^2 \]

- Basis \( \{e_i\} \equiv \{k, k_{\perp}, e_3, e_4\} \) and coordinates \( z = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, \mu_{11}, \mu_{12}, \mu_{22}) \)

\[ q_1 = \sum_i x_i e_i , \quad q_2 = \sum_i y_i e_i , \quad (\bar{q}_i \cdot \bar{q}_j) = (q_i \cdot q_j) - \mu_{ij} \]

- Division of \( \mathcal{N}_{11234} \) modulo \( \mathcal{G}_J_{11234} = \mathcal{G}_J_{1234} \)

\[ \mathcal{N}_{11234} = \mathcal{N}_{1234} D_1 + \mathcal{N}_{1134} D_2 + \mathcal{N}_{1124} D_3 + \mathcal{N}_{1123} D_4 + \Delta_{11234} \]

quotients

remainder
Divide-and-Conquer approach: a simple example

\[
I_{1234} = \frac{N_{11234}}{D_1 D_2 D_3 D_4}
\]

\[
D_1 = \bar{q}_1^2 - m^2 ,
D_2 = (\bar{q}_1 - k)^2 - m^2 ,
D_3 = \bar{q}_2^2 ,
D_4 = (\bar{q}_1 + \bar{q}_2)^2 - m^2
\]

- Basis \( \{ e_i \} \equiv \{ k, k_\perp, e_3, e_4 \} \) and coordinates \( \mathbf{z} = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, \mu_{11}, \mu_{12}, \mu_{22}) \)

\[
q_1 = \sum_i x_i e_i , \quad q_2 = \sum_i y_i e_i , \quad (\bar{q}_i \cdot \bar{q}_j) = (q_i \cdot q_j) - \mu_{ij}
\]

- Division of \( N_{11234} \) modulo \( G_{\mathcal{J}_{1234}} \) (\( = G_{\mathcal{J}_{1234}} \))

\[
N_{11234} = N_{1234} D_1 + N_{1134} D_2 + N_{1124} D_3 + N_{1123} D_4 + \Delta_{11234}
\]

quotients

remainder

- Division of \( N_{i_1 i_2 i_3 i_4} \) modulo \( G_{\mathcal{J}_{i_1 i_2 i_3 i_4}} \), e.g.

\[
N_{1234} / G_{\mathcal{J}_{1234}} \quad \Rightarrow \quad N_{1234} = Q_{234}^{(1234)} D_1 + Q_{134}^{(1234)} D_2 + Q_{124}^{(1234)} D_3 + Q_{123}^{(1234)} D_4 + \Delta_{1234}
\]

quotients

remainder

\[
N_{1134} / G_{\mathcal{J}_{1134}} \quad \Rightarrow \quad N_{1134} = Q_{134}^{(1134)} D_1 + Q_{114}^{(1134)} D_3 + Q_{113}^{(1134)} D_4 + \Delta_{1134}
\]

quotients

remainder
**Divide-and-Conquer approach: a simple example**

\[ I_{11234} = \frac{N_{11234}}{D_1 D_2 D_3 D_4} \]

\[ D_1 = \vec{q}_1^2 - m^2, \]
\[ D_2 = (\vec{q}_1 - \vec{k})^2 - m^2, \]
\[ D_3 = \vec{q}_2^2, \]
\[ D_4 = (\vec{q}_1 + \vec{q}_2)^2 - m^2 \]

- **Basis** \( \{e_i\} \equiv \{k, k_\perp, e_3, e_4\} \) and coordinates \( z = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4, \mu_{11}, \mu_{12}, \mu_{22}) \)

\[ q_1 = \sum_i x_i e_i, \quad q_2 = \sum_i y_i e_i, \quad (\vec{q}_i \cdot \vec{q}_j) = (q_i \cdot q_j) - \mu_{ij} \]

- **Division of** \( N_{11234} \) **modulo** \( G_{\mathcal{J}_{11234}} \) (**=** \( G_{\mathcal{J}_{1234}} \))

\[ N_{11234} = N_{1234} D_1 + N_{1134} D_2 + N_{1124} D_3 + N_{1123} D_4 + \Delta_{11234} \]

- **Division of** \( N_{i_1 i_2 i_3 i_4} \) **modulo** \( G_{\mathcal{J}_{i_1 i_2 i_3 i_4}} \)

\[ N_{11234} = \frac{N_{234} D_1^2 + N_{134} D_1 D_2 + N_{124} D_1 D_3 + N_{123} D_1 D_4 + N_{114} D_2 D_3 + N_{113} D_2 D_4 + \Delta_{1234} D_1 + \Delta_{1134} D_2 + \Delta_{1124} D_3 + \Delta_{1123} D_4 + \Delta_{11234}}{\text{(sums of) quotients}} + \Delta_{11234} \]

\[ + \Delta_{1234} D_1 + \Delta_{1134} D_2 + \Delta_{1124} D_3 + \Delta_{1123} D_4 + \Delta_{11234} \]

\[ \text{(remains)} \]
Divide-and-Conquer approach: a simple example

- after a further step (division $N_{i_1i_2i_3}/G_{i_1i_2i_3}$) no quotient remains

$$N_{11234} = \Delta_{11234} + \Delta_{1234}D_1 + \Delta_{1134}D_2 + \Delta_{1124}D_3 + \Delta_{1123}D_4 + \Delta_{234}D_1^2 + \Delta_{114}D_2D_3 + \Delta_{113}D_2D_4$$

- the integrand decomposition becomes

$$\mathcal{I}_{11234} = \frac{N_{11234}}{D_1^2D_2D_3D_4} = \frac{\Delta_{11234}}{D_1^2D_2D_3D_4} + \frac{\Delta_{1234}}{D_1D_2D_3D_4} + \frac{\Delta_{1134}}{D_1^2D_3D_4} + \frac{\Delta_{1124}}{D_1^2D_2D_4}$$

$$+ \frac{\Delta_{1123}}{D_2^2D_2D_3} + \frac{\Delta_{234}}{D_2D_3D_4} + \frac{\Delta_{114}}{D_2^2D_4} + \frac{\Delta_{113}}{D_2^2D_3}$$

$$\Delta_{11234} = 16m^2 \left( k^2 + 2m^2 - k^2 \epsilon \right) ,$$

$$\Delta_{1234} = 16 \left[ (q_2 \cdot k)(1 - \epsilon)^2 + m^2 \right] ,$$

$$\Delta_{1124} = - \Delta_{1123} = 8 (1 - \epsilon) \left[ k^2(1 - \epsilon) + 2m^2 \right] ,$$

$$\Delta_{1134} = - 16m^2 (1 - \epsilon) ,$$

$$\Delta_{113} = - \Delta_{114} = \Delta_{234} = 8 (1 - \epsilon)^2 .$$
Examples of divide-and-conquer approach

- Photon self-energy in massive QED, \((4 - 2\epsilon)\)-dimensions

![Diagrams](image1)

- Diagrams entering \(gg \rightarrow H\), in \((4 - 2\epsilon)\)-dimensions

![Diagrams](image2)
Higher loops

Additional relations between integrals

P. Mastrolia, G. Ossola, T.P. (work in progress)

- The integrals given by the integrand reduction can be further reduced with additional identities
  - traditional approach: Integration by Part (IBP)

\[
\int \frac{\partial}{\partial q_i^\mu} \frac{N(q_i)^\mu}{D_1 \cdots D_n} = 0
\]

- A 2-step strategy
  1. use integrand reduction first
     ⇒ integrals with higher multiplicity should be reduced
  2. then apply IBP
     ⇒ could be easier after integrand reduction

- Can we instead see IBPs from Integrand Reduction?
  - Can we recover IBPs from int. red. relations computed in step 1?
Higher loops

Dimensionally shifted integrals

One-loop case:

- with $v^\mu_\perp = \epsilon^\mu_{\mu_1 \cdots \mu_{n-1}} k_1^{\mu_1} \cdots k_{n-1}^{\mu_{n-1}}$, we can prove

$$ I_{1 \cdots n}[\mu^2] = -\epsilon I_{1 \cdots n}^{(d+2)}, $$

$$ I_{1 \cdots n}[(q \cdot v_\perp)^2] = I_{1 \cdots n}[\epsilon(q, k_1, \ldots, k_{n-1})^2] = -\frac{v_\perp^2}{2} I_{1 \cdots n}^{(d+2)} $$

- perform integrand reduction of $I_{1 \cdots n}[(q \cdot v_\perp)^2]$ from $n = 1$ to higher-points
  - we can reuse the same pol. divisions of integrand reduction
- if $I_{1 \cdots n}[(q \cdot v_\perp)^2]$ reducible at integrand level $\Rightarrow$ then $I_{1 \cdots n}$ reducible at integral level
  - we get an homogeneous equation with integrals in $d + 2$
  - after $d \rightarrow d - 2$ we get an IBP relation
Example: one-loop tadpole

Tadpoles: $\mathcal{N} = \epsilon(q)^2 = q^2$:

\[ \mathcal{I}_0 = \frac{q^2}{D_0}, \quad D_0 = \bar{q}^2 - m^2 \]

- No external vectors ⇒ use special case

\[ \mathcal{I}_0[q^2] = -2 \mathcal{I}_0^{(d+2)} \]

- After integrand reduction

\[ -2 \mathcal{I}_0^{(d+2)} = \mathcal{I}_0[q^2] = \mathcal{I}_0[\mu^2] + m^2 \mathcal{I}_0 \]

\[ = \frac{d - 4}{2} \mathcal{I}_0^{(d+2)} + m^2 \mathcal{I}_0 \]

Dimensional shift for tadpoles

\[ d \mathcal{I}_0^{(d+2)} = 2 m^2 \mathcal{I}_0 \]
Example: one-loop bubble

\[ \mathcal{I}_{01}[(q \cdot v_{\perp})^2] = \frac{\mathcal{N}}{D_0 D_1} \]

\[ \mathcal{N} = \epsilon(q, k)^2 = q^2 k^2 - (q \cdot k)^2 \]

\[ D_0 = \bar{q}^2 \]

\[ D_1 = \bar{q}^2 + 2(q \cdot k) \]

**Integrand reduction**

\[ \mathcal{N} = m^2 \mu^2 + D_0 \left( \frac{1}{2} m^2 + \frac{1}{2} ((q + k) \cdot k) \right) + D_1 \left( -\frac{1}{2} (q \cdot k) \right) \]

\[ -\frac{3m^2}{2} \mathcal{I}_{01}^{(d+2)} = \mathcal{I}_{01}[\mathcal{N}] = m^2 \mathcal{I}_{01}[\mu^2] + \frac{m^2}{2} \mathcal{I}_1 = \frac{d-4}{2} m^2 \mathcal{I}_{01}^{(d+2)} - \frac{d}{4} \mathcal{I}_1^{(d+2)} \]

**The result in \( d + 2 \) dimensions**

\[ (d - 1) \mathcal{I}_{01}^{(d+2)} = \frac{1}{2m^2} d \mathcal{I}_1^{(d+2)} \]

Result in \( d \) dimensions

\[ (d - 3) \mathcal{I}_{01} = \frac{1}{2m^2} (d - 2) \mathcal{I}_1 \]
Example: one-loop triangle

\[ \mathcal{I}_{012}[N] = \frac{N}{D_0 D_1 D_2} \]

\[ D_0 = \bar{q}^2 \]
\[ D_1 = \bar{q}^2 + 2(q \cdot k_1) \]
\[ D_2 = \bar{q}^2 - 2(q \cdot k_2) \]

With a similar procedure, from \( \mathcal{I}_{012}[\epsilon(q, k_1, k_2)^2] \) and \( \mathcal{I}_{12}[\epsilon(q, k_1 + k_2)^2] \), we get

\[ (2 - d) \mathcal{I}_{012}^{(d+2)} = \mathcal{I}_{12} \]
\[ (1 - d) \mathcal{I}_{12}^{(d+2)} = \frac{4m^2 - s}{2} \mathcal{I}_{12} + \mathcal{I}_1 \]

The result in \( d + 2 \) dimensions

\[ (2 - d) \mathcal{I}_{012}^{(d+2)} = \frac{2}{4m^2 - s} \left( (1 - d) \mathcal{I}_{12}^{(d+2)} + \frac{d}{2} \mathcal{I}_{1}^{(d+2)} \right) \]

Result in \( d \) dimensions

\[ (4 - d) \mathcal{I}_{012} = \frac{2}{4m^2 - s} \left( (3 - d) \mathcal{I}_{12} + \frac{d - 2}{2} \mathcal{I}_1 \right) \]
Higher loops

- At one-loop we used

\[ \mathcal{I}[\epsilon(q, k_1, \ldots, k_{n-1})^2], \quad \mathcal{I}[\mu^2] \]

- At higher loops we should use

\[ \mathcal{I}[\epsilon(q_1, \ldots, q_\ell, k_1, \ldots, k_{n-1})^2], \quad \mathcal{I}[\epsilon(\vec{\mu}_1, \ldots, \vec{\mu}_\ell)]. \]

- Relations for integrals in $\mu^2$ can be easily found at any loop using Schwinger parametrization
Example: two-loop

\[ \mathcal{I}_{123}[\mathcal{N}] = \frac{\mathcal{N}}{D_1 D_2 D_3} \]

\[ D_1 = \bar{q}_1^2 - m^2 = q_1^2 - m^2 - \mu_{11} \]
\[ D_2 = \bar{q}_2^2 - m^2 = q_2^2 - m^2 - \mu_{22} \]
\[ D_3 = (\bar{q}_1 - \bar{q}_2)^2 = (q_1 - q_2)^2 - \mu_{11} - \mu_{22} + 2 \mu_{12} \]

- The integrand reduction gives

\[ -3 \mathcal{I}_{123}^{(d+2)} = \mathcal{I}_{123}[\epsilon(q_1, q_2)^2] = \mathcal{I}_{123}[(q_1 \cdot q_2)^2 - q_1^2 q_2^2] \]
\[ = \frac{1}{4} \mathcal{I}_{123}[4 \mu_{12}^2 - 4 \mu_{11} \mu_{22}] + m^2 \mathcal{I}_{123}[2 \mu_{12} - \mu_{11} - \mu_{22}] - \frac{m^2}{2} \mathcal{I}_{12} \]}

- Integrals in \( \mu_{ij} \)

\[ \mathcal{I}_{123}[4 \mu_{12}^2 - 4 \mu_{11} \mu_{22}] = -2 \epsilon(1 + 2 \epsilon) \mathcal{I}_{123}^{(d+2)} \]
\[ \mathcal{I}_{123}[2 \mu_{12} - \mu_{11} - \mu_{22}] = \frac{4 - d}{d} \mathcal{I}_{12} \]

Final result in \( d \) dimensions

\[ \mathcal{I}_{123} = \frac{d - 2}{2m^2(d - 3)} \mathcal{I}_{12} \]
Summary

- we have a framework for the all-loop reduction at the integrand level
- the integrand is decomposed via multivariate polynomial division
- at one loop it reproduces well known results (OPP)
- one-loop reduction is improved by Laurent expansion (NINJA)
- algebraic reduction at any loop via divide-and-conquer approach
- IBPs via integrand reduction and $d$-shifts

Outlook

- improve one-loop generation (recursion, global abbreviations, ...)
- treatment of (few) remaining unstable points within NINJA
- application of int. red. + $d$-shifts a full two-loop QED/QCD process
- fully automated analytic one-loop via divide-and-conquer
THANK YOU
FOR YOUR ATTENTION
BACKUP SLIDES
Rotation method for error estimation


- 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: \( W b \bar{b} + 1j \ (ud \rightarrow e^+ \nu_e b \bar{b} g) \), with \( m_b \neq 0 \)

![Graph](image_url)