Semi-analytic and algebraic techniques for Integrand Reduction

Tiziano Peraro

Max-Planck-Institut für Physik, München, Germany

New Frontiers in Theoretical Physics
Cortona, Italy
28–31 May 2014

MAX-PLANCK-GESELLSCHAFT

Alexander von Humboldt
Stiftung / Foundation

Introduction and motivation

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 ℓ-loop integral is a rational function:
  - polynomial numerator $\mathcal{N}_{i_1\cdots i_n}$

$$\mathcal{M}_n = \int d^d\bar{q}_1 \cdots d^d\bar{q}_\ell \;\; \mathcal{I}_{i_1\cdots i_n}, \qquad \mathcal{I}_{i_1\cdots i_n} \equiv \frac{\mathcal{N}_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}}$$

  - loop propagators $\to$ quadratic polynomial denominators D_i
- The integrand-reduction algorithm leads to

$$\mathcal{I}_{i_1\cdots i_n}(\bar{q}_1,\cdots,\bar{q}_\ell) \equiv \frac{\mathcal{N}_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}} = \underbrace{\frac{\Delta_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}} + \cdots + \sum_{k=1}^{n}\frac{\Delta_{i_k}}{D_{i_k}}}_{\text{they must be irreducible}} + \Delta_\emptyset$$

- The residues $\Delta_{i_1\cdots i_k}$ are irreducible polynomials in $\bar{q}_i$
  - universal topology-dependent parametric form
  - the coefficients of the parametrization are process-dependent

INTEGRAND REDUCTION $\equiv$ a smart/rigorous partial fraction decomposition

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
    $\Rightarrow$ 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

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
    $\Rightarrow$ 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

Integrand reduction via polynomial division

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

Integrand reduction via *polynomial division*: the recursive formula

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

$$\mathcal{I}_{i_1\cdots i_n} \equiv \frac{\mathcal{N}_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}} = \sum_{k} \mathcal{I}_{i_1\cdots i_{k-1} i_{k+1}\cdots i_n} + \frac{\Delta_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}}$$

- Fit-on-the-cut approach
  - from a generic $\mathcal{N}$, get the parametric form of the residues Δ
  - determine the coefficients sampling on the cuts (impose $D_i = 0$)
  - residues can be built from tree-level amplitudes [see W. Torres' talk]
- Divide-and-Conquer approach
  - generate the $\mathcal{N}$ of the process
  - compute the residues by iterating the polynomial division algorithm

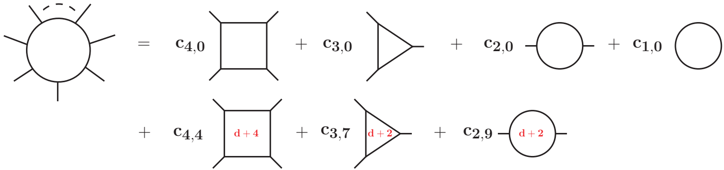
The one-loop decomposition

At one-loop we reproduce a well known result:

- the integrand decomposition
  [Ossola, Papadopoulos, Pittau (2007); Ellis, Giele, Kunszt, Melnikov (2008)]

$$\mathcal{I}_{i_1\cdots i_n} = \frac{\mathcal{N}_{i_1\cdots i_n}}{D_{i_1}\cdots D_{i_n}} = \sum_{j_1\cdots j_5} \frac{\Delta_{j_1j_2j_3j_4j_5}}{D_{j_1}D_{j_2}D_{j_3}D_{j_4}D_{j_5}} + \sum_{j_1j_2j_3j_4} \frac{\Delta_{j_1j_2j_3j_4}}{D_{j_1}D_{j_2}D_{j_3}D_{j_4}}$$
$$+ \sum_{j_1j_2j_3} \frac{\Delta_{j_1j_2j_3}}{D_{j_1}D_{j_2}D_{j_3}} + \sum_{j_1j_2} \frac{\Delta_{j_1j_2}}{D_{j_1}D_{j_2}} + \sum_{j_1} \frac{\Delta_{j_1}}{D_{j_1}}$$

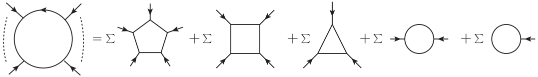
- the integral decomposition

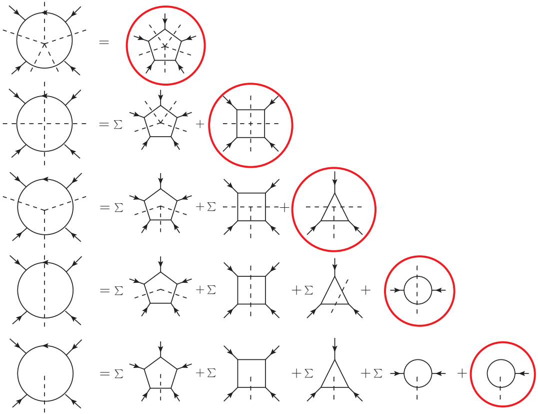


- all the Master Integrals are known!

Fit-on-the-cut at 1-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)

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

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

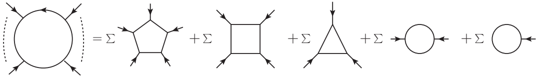
★ Implemented in the semi-numerical C++ library NINJA [T.P. (2014)]

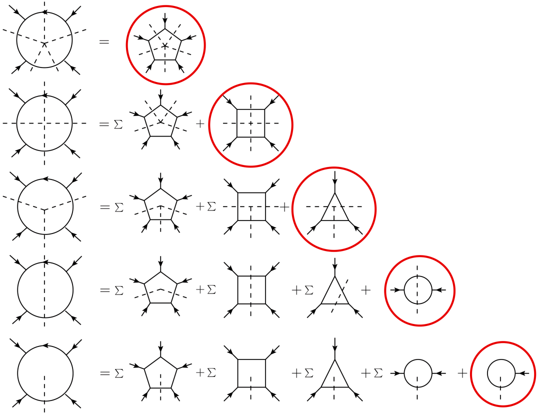
- Laurent expansions via a simplified polynomial-division algorithm
- interfaced with the package GOSAM
- interface with FORMCALC [T. Hahn et al.] under development
- is a faster and more stable integrand-reduction algorithm

★ NINJA is public ⇒ ninja.hepforge.org

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:

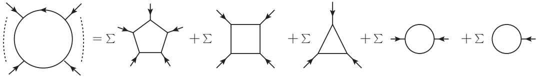


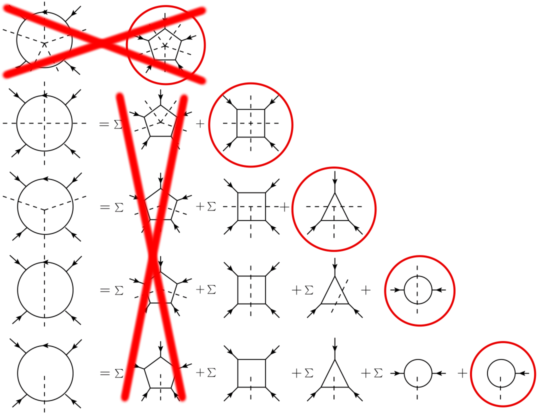


Laurent-expansion method

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



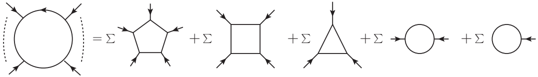


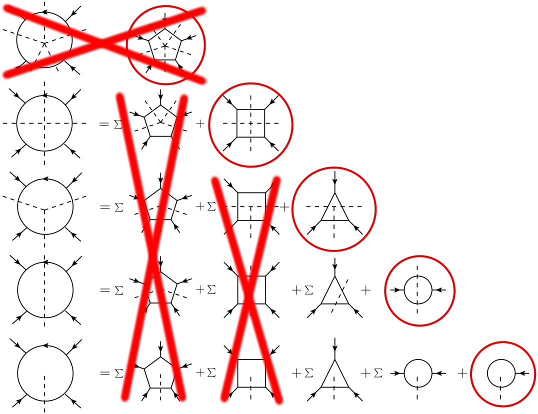
Laurent-expansion method

- pentagons not needed

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



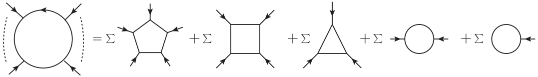


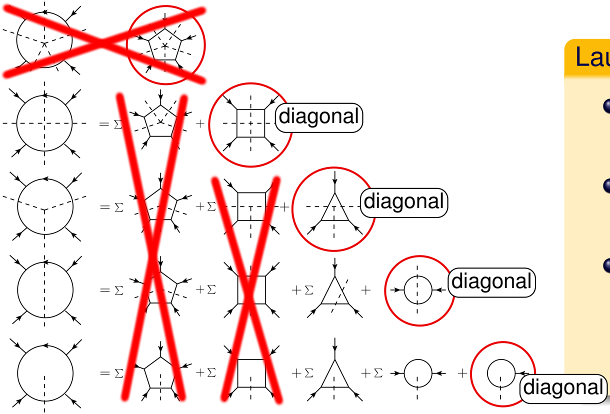
Laurent-expansion method

- pentagons not needed
- boxes never subtracted

Integrand reduction via Laurent expansion (NINJA)

Integrand decomposition:



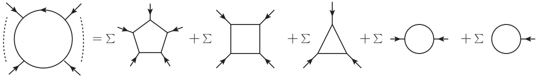


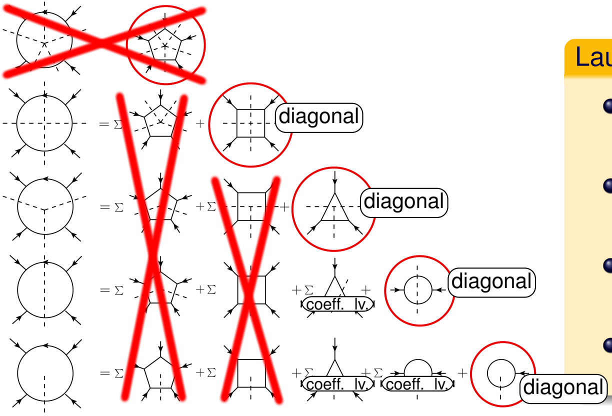
Laurent-expansion method

- 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

Automation of one-loop computation in GOSAM

GOSAM is a PYTHON package which:

- generates analytic integrands
- 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 $\mathcal{O}(1/1000)$ unstable points)
    - current default in GOSAM-2.0 $\leftarrow$ just released

Benchmarks of GOSAM + NINJA

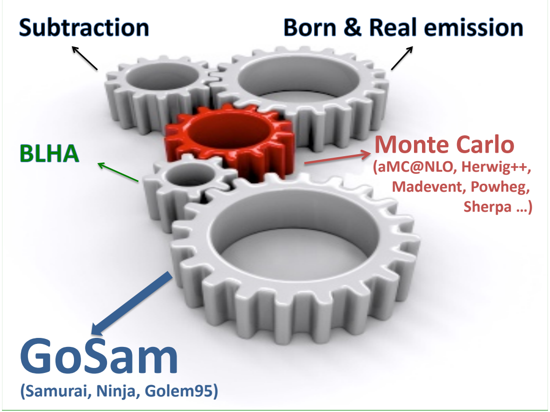
H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola and T.P. (2013)

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



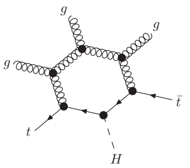
The GOSAM collaboration:

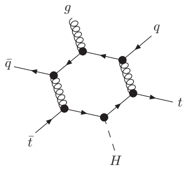
G. Cullen, H. van Deurzen, N. Greiner, G. Heinrich, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, J. Reichel , J. Schlenk, J. F. von Soden-Fraunhofen, T. Reiter, F. Tramontano, T.P.

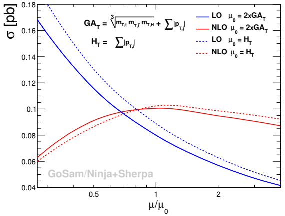
Application: $pp \to t\bar{t}H + jet$ with GOSAM + NINJA

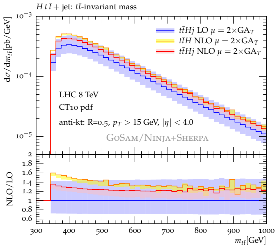
H. van Deurzen, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2013)

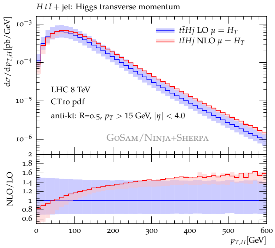
- Interfaced with the Monte Carlo SHERPA

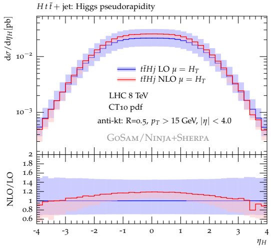






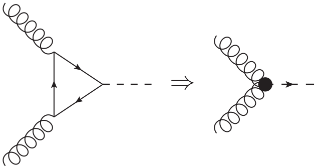






Application: $pp \to H + jets$ in GF with GOSAM + NINJA

- $m_t \to \infty$ approximation



- effective couplings $H + (2,3,4)gl.$
- higher-rank integrands $\Rightarrow$ extension of int. red. methods
  [P. Mastrolia, E. Mirabella,T.P.(2012), H. van Deurzen (2013)]

- $H + 2j$ (GOSAM+SAMURAI+SHERPA)
  [H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, J. F. von Soden-Fraunhofen, F. Tramontano, T.P.(2013)]
- $H + 3j$ (GOSAM+SAMURAI+SHERPA+MADGRAPH4/MADEVENT)
  [G. Cullen, H. van Deurzen, N. Greiner, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, F. Tramontano, T.P.(2013)]
- new analysis with ATLAS-like cuts, using NINJA for the reduction
  [G. Cullen, H. van Deurzen, N. Greiner, J. Huston, G. Luisoni, P. Mastrolia, E. Mirabella, G. Ossola, F. Tramontano, J. Winter, V. Yundin, T.P. (preliminary, 2014)]

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

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

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

- ATLAS-like cuts

$$R = 0.4, \qquad p_{t,jet} > 30\text{GeV}, \qquad |\eta_{jet}| < 4.4$$

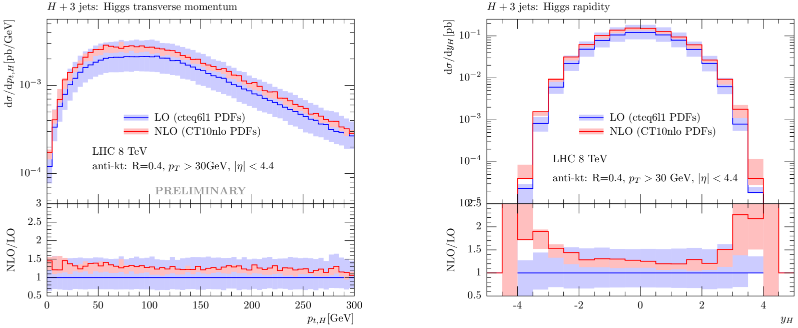
- total cross section

$$\sigma_{LO}^{(H+2j)}([\text{pb}]) = 1.23^{+37\%}_{-24\%}, \qquad \sigma_{LO}^{(H+3j)}([\text{pb}]) = 0.381^{+53\%}_{-32\%}$$
$$\sigma_{NLO}^{(H+2j)}([\text{pb}]) = 1.590^{-4\%}_{-7\%}, \qquad \sigma_{NLO}^{(H+3j)}([\text{pb}]) = 0.485^{-3\%}_{-13\%}$$

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

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

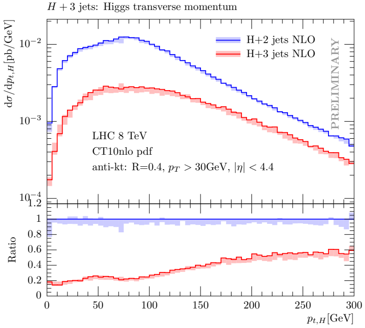
$$\mu_F = \mu_R = \frac{\hat{H}_T}{2} = \frac{1}{2}\left(\sqrt{m_H^2 + p_{t,H}^2} + \sum_{jets} |p_{t,jet}|^2\right)$$

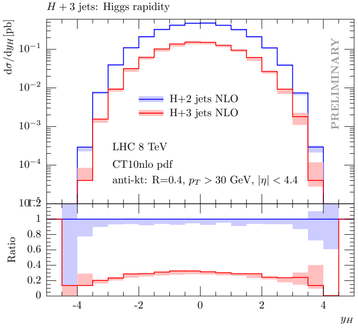


Application: $pp \to H + jets$ in GF with GOSAM + NINJA

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

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





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, 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

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

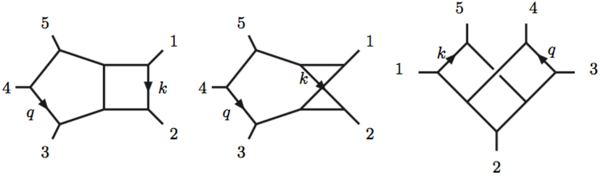
- The integrand-level approach to scattering amplitudes at one-loop
  - can be used to compute any amplitude in any QFT
  - has been implemented in several codes, some of which public [SAMURAI, CUTTOOLS, NINJA]
  - has produced (and is still producing) results for LHC [GOSAM, FORMCALC, BLACKHAT, MADLOOP,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

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

P. Mastrolia, G. Ossola (2011); P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2012)



- 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

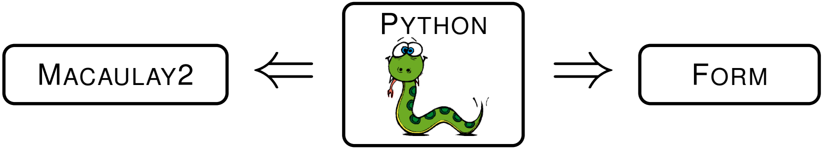
$\mathcal{N}=8$ linear combination of 8, 7 and 6-denominators MIs

Divide-and-Conquer approach

P. Mastrolia, E. Mirabella, G. Ossola, T.P. (2013)

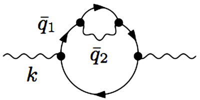
The divide-and-conquer approach to the integrand reduction

- 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



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

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

- iterating the polynomial division algorithm on the numerator we get

$$\mathcal{N}_{11234} = \Delta_{11234} + \Delta_{1234} D_1 + \Delta_{1134} D_2 + \Delta_{1124} D_3 + \Delta_{1123} D_4 + \Delta_{234} D_1^2 + \Delta_{114} D_2 D_3 + \Delta_{113} D_2 D_4$$

- the integrand decomposition becomes

$$\begin{aligned} \mathcal{I}_{11234} = \frac{\mathcal{N}_{11234}}{D_1^2 D_2 D_3 D_4} = \frac{\Delta_{11234}}{D_1^2 D_2 D_3 D_4} + \frac{\Delta_{1234}}{D_1 D_2 D_3 D_4} + \frac{\Delta_{1134}}{D_1^2 D_3 D_4} + \frac{\Delta_{1124}}{D_1^2 D_2 D_4} \\ + \frac{\Delta_{1123}}{D_1^2 D_2 D_3} + \frac{\Delta_{234}}{D_2 D_3 D_4} + \frac{\Delta_{114}}{D_1^2 D_4} + \frac{\Delta_{113}}{D_1^2 D_3} \end{aligned}$$

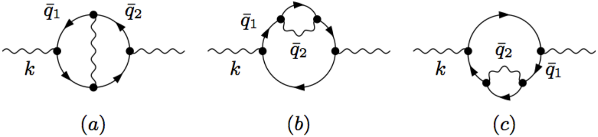
$$\Delta_{11234} = 16 m^2 \left(k^2 + 2\, m^2 - k^2 \epsilon \right) \qquad \Delta_{1134} = -\,16 m^2 \,(1-\epsilon)$$

$$\Delta_{1234} = 16 \left[(q_2 \cdot k)(1-\epsilon)^2 + m^2 \right] \qquad \Delta_{113} = -\,\Delta_{114} = \Delta_{234} = 8\,(1-\epsilon)^2$$

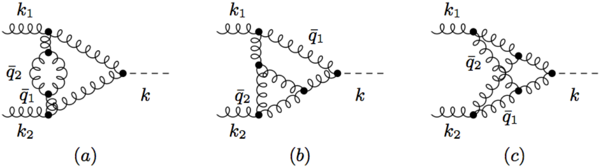
$$\Delta_{1124} = -\,\Delta_{1123} = 8\,(1-\epsilon) \left[k^2 (1-\epsilon) + 2\, m^2 \right]$$

Examples of divide-and-conquer approach

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



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



From Master Integrands to Master Integrals

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

- Independent integrands can be linearly dependent at the integral level
  - further identities exist between integrals
  - traditional approach: Integration by Part (IBP)

$$\int \frac{\partial}{\partial \bar{q}_i^\mu} \frac{\mathcal{N}(\bar{q}_i)^\mu}{D_{i_1} \cdots D_{i_n}} = 0$$

- A 2-step strategy
  1. use integrand reduction first
     $\Rightarrow$ integrals with higher multiplicity should be reduced
  2. then apply IBP
     $\Rightarrow$ 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?

From Master Integrands to Master Integrals

IBP identities can be found by combining

- integrand reduction of “special” integrands
- dimensional recurrence relations of respective integrals

- “Special” integrands can be Shouten polynomials [see L. Tancredi’s talk]
- They satisfy dimensional recurrence relations
  - easily found using Schwinger parameters

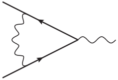
$$\mathcal{I}[S(4; q_1, \ldots, q_\ell, k_1, \ldots, k_{n-1})] \propto \mathcal{I}^{(d+2)}$$
$$\mathcal{I}[S(-2\epsilon; \vec{\mu}_1, \ldots, \vec{\mu}_\ell)] \propto \mathcal{I}^{(d+2)}$$

$\Rightarrow$ integrand reduction of l.h.s. + dim. shifts, from lower to higher point integrals, gives IBP-like or PV-like relations

Examples of IBP via int. red. + dim. shifts

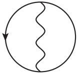
$$\mathcal{I}_{01}[\mathcal{N}] = \frac{\mathcal{N}}{D_0 D_1}$$

$$(d-3)\,\mathcal{I}_{01} = \frac{1}{2\,m^2}\,(d-2)\,\mathcal{I}_1$$



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

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



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

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

Summary and Outlook

- 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 knwon 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,. . .)
  - 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