Laporta algorithm for multi-loop vs multi-scale problems

(with Philipp Maierhöfer and Peter Uwer)

11th FCC-ee workshop: Theory and Experiments

Johann Usovitsch

European Research Council
Established by the European Commission

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

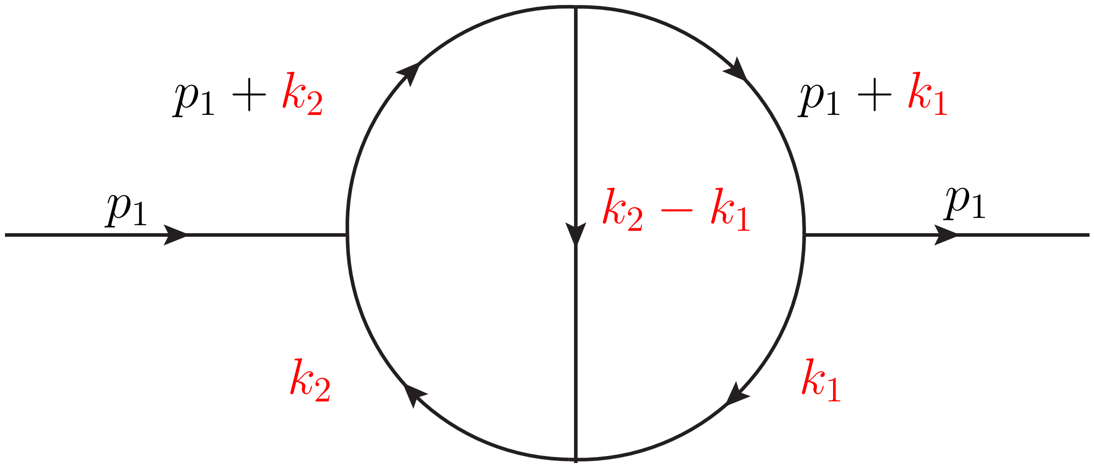
09. January 2019

Outline

Integration-by-parts identities applications

- Integration-by-parts (IBP)[Chetyrkin,Tkachov,1981] and Lorentz invariance [Gehrmann,Remiddi,2000] identities for scalar Feynman integrals are very important in quantum field theoretical computations (multi-loop computations)
- Reduce the number of Feynman integrals to compute, which appear in scattering amplitude computations
- Compute the integrals analytically or numerically with the method of differential equations [Kotikov,1991;Remiddi,1997;Henn,2013;Argeri et al.,2013;Lee,2015;Meyer,2016] or difference equations[Laporta,2000;Lee,2010] (require basis change and IBP reductions)
- Use the method of sector decomposition [Heinrich,2008] (pySecDec [Borowka et al.,2018] and Fiesta4 [Smirnov,2016]) or use the linear reducibility of the integrals (HyperInt [Panzer,2014]) to compute the Feynman integrals analytically or numerically (require basis change and IBP reductions).

Scalar Integrals



$$I(a_1, \ldots, a_5) = \int \frac{d^d k_1 d^d k_2}{[k_1{}^2]^{a_1}[(p_1 + k_1)^2]^{a_2}[k_2{}^2]^{a_3}[(p_1 + k_2)^2]^{a_4}[(k_2 - k_1)^2]^{a_5}}$$

- Integral depends explicitly on the exponents a_f
- Loop momenta: $k_1,\ k_2$, $L = 2$
- Number of the propagators: $N = 5$

IBP Identities

$$I(a_1,\dots,a_5)=\int\frac{d^dk_1\,d^dk_2}{[k_1{}^2]^{a_1}[(p_1+k_1)^2]^{a_2}[k_2{}^2]^{a_3}[(p_1+k_2)^2]^{a_4}[(k_2-k_1)^2]^{a_5}}$$

Integration-by-parts (IBP) identities:

$$\begin{aligned}\int d^d\boldsymbol{k_1}\dots d^d\boldsymbol{k_L}\frac{\partial}{\partial(\boldsymbol{k_i})_\mu}\left((q_j)_\mu\frac{1}{[P_1]^{a_1}\dots[P_N]^{a_N}}\right)&=0\\ c_1(\{a_f\})I(a_1,\dots,a_N\mathbf{-1})+\dots+c_m(\{a_f\})I(a_1\mathbf{+1},\dots,a_N)&=0\end{aligned}$$

$q_j=p_1,\dots,p_E,k_1,\dots,k_L$

Express all integrals with the same set of propagators but with different exponents a_f as a linear combination of some basis integrals (master integrals).

- Gives relations between the scalar integrals with different exponents a_f
- Number of $L(E+L)$ IBP equations, $i=1,\dots,L$ and $j=1,\dots,E+L$
- a_f = symbols: Look for recursion relations, LiteRed [Lee,2012]
- a_f = integers: Sample a system of equations, Laporta algorithm [Laporta,2000]

Laporta Algorithm [Laporta,2000]

Scalar integrals $I(a_1,\ldots,a_5)$ with integer values a_f

Sample system of IBP equations, Reduze [Studerus,Manteuffel,2012] language

- $r=\sum_{f=1}^{N} a_f$ mit $a_f>0,\ f=1,\ldots,N$
- $s=-\sum_{f=1}^{N} a_f$ mit $a_f<0,\ f=1,\ldots,N$
- Seed integrals: $r\in[r_{\min},r_{\max}],\ \ s\in[s_{\min},s_{\max}]$
- $S=\sum_{i=1}^{N}\theta_j\times 2^{j-1}\theta_j=1$ for each $a_f>0$ else $\theta_j=0$
- T topology number

Fire [Smirnov,2008] language

- Avoid reductions of scalar integrals $\notin(r,s)$

- Different public implementations: Air [Lazopoulos,Anastasiou,2004], FIRE [Smirnov,2008] and Reduze [Studerus,Manteuffel,2012] and Kira [Maierhöfer, Usovitsch, Uwer,2017]
- Kira is more powerful the less LiteRed succeeds

Kira version 1.2

Kira is an implementation of the Laporta algorithm

Get Kira gitlab at: https://gitlab.com/kira-pyred/kira.git

- New equation generator which is $\sim 10^L$ faster than Kira 1.1 multi-loop
- Improved parallelization - no openMP
- Compiles on your Mac / New build system: Meson
- Get relations from higher sectors – minimize the number of master integrals
- Start a reduction with a preferred list of master integrals
- Focus the reduction only to a subset of master integrals — set all other coefficients to zero, since Kira 1.0 and 1.1
- New flexible seed notation is introduced, while the old is preserved
- Choose between 8 different integral Laporta orderings
- Coefficient simplifications are based on heuristics
- New feature: Algebraic reconstruction multi-scale
- New feature: User defined system of equations
- Release notes: arXiv:1812.01491

gg→H at 3-loops: integralfamilies.yaml


```
integralfamilies:
  - name: Xhiggs3l1_mmmmmmm00
    loop_momenta: [ l1, l2, l3 ]
    top_level_sectors: [511] # important option
    propagators:
      - [ "l1", "m^2" ]
      - [ "l2", "m^2" ]
      - [ "l3", "m^2" ]
      - [ "l1 - q1", "m^2" ]
      - [ "l2 - q1 - q2", "m^2" ]
      - [ "l1 - l2", 0 ]
      - [ "-l2 + l3 + q1 + q2", 0 ]
      - [ "l1 - l2 + l3", "m^2" ]
      - [ "l1 - l2 + l3 + q2", "m^2" ]
      - { bilinear: [ [ "l1", "l3" ], 0 ] }
      - { bilinear: [ [ "l2", "q1" ], 0 ] }
      - { bilinear: [ [ "l3", "q1" ], 0 ] }
```

gg→H at 3-loops: Old v.s. new jobs.yaml interface

```
jobs:
 - reduce_sectors:
    sector_selection: # Old
     select_recursively: # Old
      - [Xhiggs3l1_mmmmmmm00,511] # Old
    identities: # Old
     ibp: # Old
      - { r: [t, 10], s: [0, 4] } # Old
    reduce: # New
     - {r: 10, s: 4} # New
    select_integrals: # important option
     select_mandatory_recursively: # important option
      - {r: 10, s: 4, d: 1} # important option
```

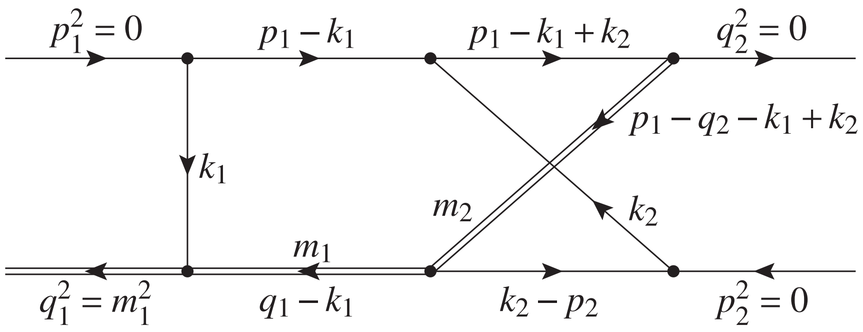
- Kira implicitly knows from integralfamilies.yaml that the user wants to reduce the topology named: Xhiggs3l1_mmmmmmm00
- From `top_level_sectors: [511]` Kira assumes that the user wants to reduce the sector: 511

Reduction of a gg→H at 3-loops non-planar topology

Algorithm	**Kira 1.1** (32 cores)	**Kira 1.2** (16 cores)
Generate system of equations	7.9 h	-
Reduce numerically	3.6 h	-
Generate and reduce numerically	-	3.4 h
Build triangular form (thread pools)	26 h	4.8 h
Backward substitution (heuristics)	18.8 d	4.1 d

- Seed specification: {r: 10, s: 4, d: 1}
- Speedup comes from less calls to Fermat: 382.502.520 x 5 (Kira 1.1) v.s. 981 (Kira 1.2)
- After the numerical reduction over the finite field (integers modulo 64 Bit prime number) is finished, you know the master integrals

Algebraic coefficient simplification



Type	$T_{\text{Kira 1.1}}^{m_2^2=\frac{3}{14}m_1^2}$	$T_{\text{Kira 1.2}}^{m_2^2=\frac{3}{14}m_1^2}$	$T_{\text{Kira 1.1}}$	$T_{\text{Kira 1.2}}$	$T_{\text{Reduze 2}}^{m_2^2=\frac{3}{14}m_1^2}$	$T_{\text{FIRE 5}}^{m_2^2=\frac{3}{14}m_1^2}$
default	2.4 h	1 h	-	11.5 h	2.7 d	23.5 h
A	35.3 min	28.4 min	10 h	5.8 h	-	22.4 h

- default: `select_mandatory_recursively: [{r: 7, s: 4}]`
- A: `select_mandatory_recursively: [{r: 7,s: 4,d: 0}]`
- Reduze 2 A. von Manteuffel and C. Studerus (2012), FIRE 5 A. V. Smirnov (2014) in C++ and using the same Fermat executable.

Algebraic reconstruction

Backward substitution gives: $I(\{a_i\}) = \sum_j^M C_j M_j$, M_j master integral

- $C_j = \sum_{i=1}^N c_i$,
- $N \approx \mathcal{O}(10^2) - (10^5)$
- Naiv sum gives a snow ball effect: Intermediate sum grows to more complicated terms then the final result.
- One solution since Kira 1.0 is to constantly sort the terms c_i and the intermediate sums in their string length.

Second solution since Kira 1.2 is the algebraic reconstruction

- Sample $\sum_{i=1}^N c_i$ by setting at least one parameter $\{\frac{s}{m_1^2}, \frac{t}{m_1^2}, \frac{m_{i\neq 1}^2}{m_1^2}, \ldots\}$ to integer numbers
- Interpolate the final result from these samples

Implementation part 1

Dependence on at least 2 parameters, e.g.: $\{D, x\}$, $x = \frac{s}{m_1^2}$

- Sample once $C(D, x)$ for numeric value in D
- Get $C(x)$ rational function
- Get the degree of the polynomials (numerator and denominator) of $C(x)$ in x: d_N and d_D
- Interpolate the numerator and denominator in x individually with Newtonian approach
- Use $C(x)$ later as a reference point to eliminate sign and numeric prefactor ambiguities
- Original work in this field is based on, see arXiv: 1805.01873 1712.09737 1511.01071 by Yang Zhang and his collaborators

Implementation part 2

Sample $C(D,x)$ $\max(d_N+2, d_D+2)$ for numeric values x_j in x

- Get multiple functions $C(D,x) \to \{C(D,x_j)\}$
- Test that all numerators and denominators have the same number of terms, if not, resample

Interpolate the numerator and the denominator of $C(D,x)$ individually, by using the Newtonian interpolation formula

- $C(D,x) = \sum_{i=1}^{d_N+1, d_D+1} a_i \prod_{j=1}^{i-1} (x - x_j)$
- $a_1 = C(D, x_1)$
- $a_2 = \frac{C(D,x_2)-a_1}{x_2-x_1}$
- $a_3 = (\frac{C(D,x_3)-a_1}{x_3-x_1} - a_2)\frac{1}{x_3-x_2}$
- ...
- $a_{d_N+1} = ((\frac{C(D,x_{d_N+1})-a_1}{x_{d_N+1}-x_1} - a_2)\frac{1}{x_{d_N+1}-x_2} - \cdots - a_{d_N})\frac{1}{x_{d_N+1}-x_{d_N}}$

Implementation part 3

- To activate the algebraic reconstruction use:
  `algebraic_reconstruct: true`
- Kira decides based on heuristics to use the algebraic reconstruction algorithm or not
- Heuristics are: Number of terms in a sum, length of the biggest coefficients
- All implementation details are "hidden under the hood" — await improvements and more benchmarks (code is public)
- at present algebraic reconstruction kicks in only for the coefficients during the backward substitution
- Next Kira version will include the algebraic reconstruction of the whole reduction
- Possible usage: Treat coefficients of the master integrals individually

Summary and Outlook

- Kira version 1.2 is available: https://gitlab.com/kira-pyred/kira.git and includes:
- Fast equation generator
- Improved parallelization
- New flexible seed notation, while the old is preserved
- New feature: Algebraic reconstruction
- Todo list:
- Algebraic reconstruction for the whole system, parallelization across different machines.
- Kira is an all-rounder best in all disciplines: multi-loop, multi-scale and user defined system of equations reductions