Laporta algorithm for multi-loop vs multi-scale problems
(with Philipp Maierhöfer and Peter Uwer)
11th FCC-ee workshop: Theory and Experiments

Johann Usovitsch

09. January 2019
Outline

1 Introduction

2 Implementation - Kira

3 Examples and Challenges

4 New feature

5 Summary and Outlook
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.


- 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: \( N = 5 \)
Introduction

**IBP Identities**

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

Integration-by-parts (IBP) identities:

\[ \int d^d k_1 \ldots d^d k_L \frac{\partial}{\partial (k_i)_\mu} \left( (q_j)_\mu \frac{1}{[P_1]^{a_1} \ldots [P_N]^{a_N}} \right) = 0 \]

\[ c_1(\{a_f\})I(a_1, \ldots, a_N-1) + \cdots + c_m(\{a_f\})I(a_1+1, \ldots, a_N) = 0 \]

\[ q_j = p_1, \ldots, p_E, k_1, \ldots, k_L \]

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, \( i = 1, \ldots, L \) and \( j = 1, \ldots, E + L \)
- \( a_f = \) symbols: Look for recursion relations, LiteRed \cite{Lee2012}
- \( a_f = \) integers: Sample a system of equations, Laporta algorithm \cite{Laporta2000}
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

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

Fire [Smirnov,2008] language

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


- 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
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 $gg\rightarrow H$ at 3-loops non-planar topology

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kira 1.1 (32 cores)</th>
<th>Kira 1.2 (16 cores)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate system of equations</td>
<td>7.9 h</td>
<td>-</td>
</tr>
<tr>
<td>Reduce numerically</td>
<td>3.6 h</td>
<td>-</td>
</tr>
<tr>
<td><strong>Generate and reduce numerically</strong></td>
<td>-</td>
<td>3.4 h</td>
</tr>
<tr>
<td>Build triangular form (thread pools)</td>
<td>26 h</td>
<td>4.8 h</td>
</tr>
<tr>
<td>Backward substitution (heuristics)</td>
<td>18.8 d</td>
<td>4.1 d</td>
</tr>
</tbody>
</table>

- Seed specification: \{r: 10, s: 4, d: 1\}
- Speedup comes from less calls to Fermat: $382,502.520 \times 5$ (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

\[
\begin{align*}
  p_1^2 &= 0 \\
  p_1 - k_1 &= 0 \\
  p_1 - k_1 + k_2 &= 0 \\
  q_2^2 &= 0 \\
  p_2^2 &= 0 \\
  q_1^2 &= m_1^2 \\
  q_1 - k_1 &= 0 \\
  k_2 - p_2 &= 0 \\
  k_2 - p_2 &= 0 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>(T_{Kira\ 1.1}^{m_2^2 = \frac{3}{14} m_1^2})</th>
<th>(T_{Kira\ 1.2}^{m_2^2 = \frac{3}{14} m_1^2})</th>
<th>(T_{Kira\ 1.1})</th>
<th>(T_{Kira\ 1.2})</th>
<th>(T_{\text{Reduce~2}}^{m_2^2 = \frac{3}{14} m_1^2})</th>
<th>(T_{\text{FIRE~5}}^{m_2^2 = \frac{3}{14} m_1^2})</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>2.4 h</td>
<td>1 h</td>
<td>-</td>
<td>11.5 h</td>
<td>2.7 d</td>
<td>23.5 h</td>
</tr>
<tr>
<td>A</td>
<td>35.3 min</td>
<td>28.4 min</td>
<td>10 h</td>
<td>5.8 h</td>
<td>-</td>
<td>22.4 h</td>
</tr>
</tbody>
</table>

- 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

- \( C_j = \sum_{i=1}^{N} c_i \),
- \( N \approx \mathcal{O}(10^2) - (10^5) \)
- Naïve sum gives a snowball effect: Intermediate sum grows to more complicated terms than 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 \( \left\{ \frac{s}{m_1^2}, \frac{t}{m_1^2}, \frac{m_i^2}{m_1^2} \right\}_{i \neq 1}, \ldots \) 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 $C(D, x) \rightarrow \{C(D, x_j)\}$
- 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

$$C(D, x) = \sum_{i=1}^{d_N+1,d_D+1} a_i \prod_{j=1}^{i-1} (x - x_j)$$

- $a_1 = C(D, x_1)$
- $a_2 = \frac{C(D, x_2) - a_1}{x_2 - x_1}$
- $a_3 = \left( \frac{C(D, x_3) - a_1}{x_3 - x_1} - a_2 \right) \frac{1}{x_3 - x_2}$
- \ldots
- $a_{d_N+1} = \left( \frac{C(D, x_{d_N+1}) - a_1}{x_{d_N+1} - x_1} - a_2 \right) \frac{1}{x_{d_N+1} - x_2} - \cdots - a_{d_N} \right) \frac{1}{x_{d_N+1} - x_d_N}$
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
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