Integral Reduction with Kira 2 and Finite Field Methods
based on: Computer Physics Communications 266 (2021) 108024
(with Jonas Klappert, Fabian Lange, Philipp Maierhöfer)
RADCOR-LoopFest 2021

Johann Usovitsch

19. May 2021
Outline

1 Introduction

2 Main feature: finite field reconstruction
   - Combining algebraic forward elimination with finite field reduction
   - Reducing the memory footprint with iterative reduction
   - Runtime reduction with coefficient arrays
   - Runtime reduction with MPI
   - Double-pentagon topology in five-light-parton scattering

3 Summary and outlook
Introduction

- Kira is a linear solver for sparse linear system of equations with main application to **Feynman integral reduction**
- **The development of Kira** is dedicated to extend the range of feasible high precision calculations and help to study many state-of-the-art problems
- Kira should be used to built more advanced tools to compute Feynman integrals
- To do so, YAML allows to write human readable interface to Kira
Feynman integral reduction 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 to a **small basis** of master integrals

- **Compute these integrals** analytically or numerically with the methods of
  - Use the method of **sector decomposition** [Heinrich, 2008] (pySecDec [Borowka et al., 2018] and Fiesta4 [Smirnov, 2016])
  - Use the **linear reducibility** of the integrals (HyperInt [Panzer, 2014]) to compute the Feynman integrals analytically or numerically
  - **Auxiliary mass flow integrals** [Xin Guan, Xiao Liu, Yan-Qing Ma, 2020]
Feynman integral

\[ T(a_1, \ldots, a_N) = \int \left( \prod_{i=1}^{L} d^d \ell_i \right) \frac{1}{P_1^{a_1} P_2^{a_2} \cdots P_N^{a_N}}, \quad N = \frac{L}{2}(L + 1) + LE \]  

- \( P_j = q_j^2 - m_j^2, \ j = 1, \ldots, N \), are the inverse propagators
- The momenta \( q_j \) are linear combinations of the loop momenta \( \ell_i, \ i = 1, \ldots, L \) for an \( L \)-loop integral, and external momenta \( p_k, \ k = 1, \ldots, E \) for \( E + 1 \) external legs
- The \( m_j \) are the propagator masses
- The \( a_j \) are the (integer) propagator powers
- sector number: \( S = \sum_{j=1}^{N} 2^{j-1} \theta(a_j - \frac{1}{2}) \)

Integrals with the same sector number share the same number of positive propagator powers. **Example:** Integrals \( T(1,1,0,2) \) and \( T(1,2,-1,1) \) belong to the same sector, while the integral \( T(0,0,1,2) \) belongs to a different sector.
Relations from higher sectors

- Relations between the integrals in one sector exist which cannot be generated with the IBP- or LI-identities or symmetry relations for the same sector
- More relevant: the number of master integrals for one specific sector may be further reduced by taking symmetry relations from sectors with more positive propagator powers

Kira specific:

- In case a sector, which is missing the additional relation, is the same sector as listed in `top_level_sectors`, then the option `top_level_sectors` restricts Kira not to generate symmetry relations beyond the `top_level_sector` and we still miss the additional relation
- For this scenario the option `magic_relations` exists. It loosens the restrictions of the option `top_level_sectors` and allows Kira to built symmetry relations for sectors with more positive propagator powers, which generate the additional relation
- We will explain this in more detail in our gitlab Wiki
Automatic generation of equations

- The default behavior of Kira is to generate a system of equations for each integral in the list of preferred master integrals or master equations.
- The idea is: Kira guarantees that for any choice of master integrals the integrals appearing in the system of differential equations are reduced to the minimal set of preferred master integrals.
- Advanced trick: in the study of magic relations we may list one integral in the preferred list of master integrals, which represents the sector responsible for the generation of magic relations. Which is needed if multi-topology reductions are considered.
- We promise to bring up soon a feature to switch on/off this behavior.
Laporta algorithm challenges

- The system of equations generated the Laporta way contains many redundant equations.
- The number of equations may go up to billions and more.
- The coefficients are polynomials in the dimension $D$ and many different scales $\{s_{12}, s_{23}, m_1, m_2, ..\}$.
- Solving linear system of equations generated with the Laporta algorithm are CPU, disk and memory expensive computations.
- Make trade offs to finish the reduction, e.g.: decrease the CPU costs but increase memory or disk costs.
- Explore algorithmic improvements!
Main feature: finite field reconstruction

Finite field reconstruction: Kira + FireFly

- Reconstruction of multivariate rational functions from **samples over finite integer fields** [Schabinger, von Manteuffel, 2014][T. Peraro, 2016]
- **FireFly has been combined with Kira’s native finite field linear solver**
- Furthermore Kira supports MPI: to utilize the new parallelization opportunities now available with finite field methods
- **Side note:** the collaboration [Dominik Bendle, Janko Boehm, Murray Heymann, Rourou Ma, Mirko Rahn, Lukas Ristau, Marcel Wittmann, Zihao Wu, Yang Zhang, 2021] implements semi-numeric row reduced echelon form. They play with Laporta ordering in intermediate steps to improve the reduction time for the forward elimination!
What is special about FireFly I

- FireFly uses **Zippel algorithm** in the multivariate case instead of the nested Newton algorithm to interpolate the Polynomials.
- In IBP reductions we will meet two kind of Polynomials:
  - $E_1 = 1 + s + s^2 + t + t^2 + st$, homogeneous in the variables which are of the same mass dimension. Note: we set in this example one variable to 1.
  - $(1+d+d^2)(1+s+s^2) = E_2 = 1+d+d^2+ds+d^2s+s^2+ds^2+d^2s^2$, again we set one variable to one.
- For the polynomial $E_1$ the Zippel algorithm needs just 6+4 probes compared to the nested Newton 9+7 probes.
- For the $E_2$ the nested Newton interpolation and the Zippel algorithm are of the same complexity.
- In general the algorithmic complexity for Zippel is $O(n \times D \times T)$ and for nested Newton $O(D^n)$, with $n$ number of variables, $D$ the maximal degree and $T$ the number of terms.
What is special about FireFly II

- Main observation: IBP reductions involving 2 scales have the complexity of $E_2 = 1 + d + d^2 + s + ds + d^2s + s^2 + ds^2 + d^2s^2$, thus FIRE6 and FiniteFlow should be on par with FireFly (probably).
- IBP reductions involving 3 or more scales have the complexity of $E_1 = 1 + s + s^2 + t + t^2 + st$, thus FireFly should be better suited compared to other public codes.
- Further algorithmic improvements over brute force approaches implemented in FireFly are efficient algorithms for rational functions, racing algorithms, scan for univariate factors and many more.
Kira + FireFly, strategy plans

- The run time for the sampling over finite field is **always** dominated by the forward elimination of the reduction.
- **Strategy**: compute the rational functions appearing in the forward elimination phase first with Kiras native forward elimination algorithm using Fermat.
- Use this system of equations in triangular form as an input to reconstruct the final rational functions appearing in the backward substitution.
Combining algebraic forward elimination with finite field reduction - Hybrid I

\[ P_1 = k_1^2, \quad P_2 = k_2^2, \quad P_3 = k_3^2, \quad P_4 = (p_1 - k_1)^2, \quad P_5 = (p_1 - k_2)^2, \quad P_6 = (p_1 - k_3)^2, \quad P_7 = (p_2 - k_1)^2, \]
\[ P_8 = (p_2 - k_2)^2, \quad P_9 = (p_2 - k_3)^2, \quad P_{10} = (k_1 - k_2)^2, \quad P_{11} = (k_1 - k_3)^2, \quad P_{12} = (k_2 - k_3)^2, \]

\[ p_1^2 = z z b, \quad p_2^2 = 1, \quad p_1 p_2 = (1 - z)(1 - z b) \]

We chose \( r = 17 \) and \( s = 0 \) for the benchmark

<table>
<thead>
<tr>
<th>Mode</th>
<th>Runtime</th>
<th>Memory</th>
<th>Probes</th>
<th>CPU time per probe</th>
<th>CPU time for probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_initiate</td>
<td>5 h 20 min</td>
<td>128 GiB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>run_triangular +</td>
<td>&gt; 14 d</td>
<td>~540 GB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>run_back_substitution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>run_firefly: true</td>
<td>6 d 3 h</td>
<td>670 GiB</td>
<td>108500</td>
<td>370 s</td>
<td>100 %</td>
</tr>
<tr>
<td>run_triangular: sectorwise</td>
<td>36 min</td>
<td>4 GiB</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>run_firefly: back</td>
<td>4 h 54 min</td>
<td>35 GiB</td>
<td>108500</td>
<td>12.2 s</td>
<td>100 %</td>
</tr>
</tbody>
</table>
Combining algebraic forward elimination with finite field reduction - Hybrid II

- The complexity to generate the equations is in this example fixed (We documented in the most recent Kira paper how to optimize this step)
- The hybrid method is 20 times less expensive in main memory management and 27 times more efficient in CPU time usage
- The forward elimination generates a new system of equations which is smaller and faster to evaluate than the original IBP system of equations
- The hybrid method is usually good idea for reductions up to 3 scales
- We encourage everyone to play with different reduction strategies available within Kira to get the best results
Reducing the memory footprint with iterative reduction

\[ r = 7 \text{ and } s = 4 \]

<table>
<thead>
<tr>
<th>Mode</th>
<th>Iterative</th>
<th>Runtime</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kira ⊕ FireFly</td>
<td>-</td>
<td>18 h</td>
<td>40 GiB</td>
</tr>
<tr>
<td>sectorwise</td>
<td>33 h 15 min</td>
<td>9 GiB</td>
<td></td>
</tr>
</tbody>
</table>

- `iterative_reduction`: sectorwise — one sector at a time
- `iterative_reduction`: masterwise — one master integral at a time
- Works well with the options `run_back_substitution` and `run_firefly`
- Independent study confirms the efficiency of this method

[Chawdhry, Lim, Mitov, 2018]

- Sacrifice the CPU time for 4 times less main memory consumption
Runtime reduction with coefficient arrays

<table>
<thead>
<tr>
<th>--bunch_size=</th>
<th>Runtime</th>
<th>Memory</th>
<th>CPU time per probe</th>
<th>CPU time for probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18 h</td>
<td>40 GiB</td>
<td>1.73 s</td>
<td>95 %</td>
</tr>
<tr>
<td>2</td>
<td>14 h</td>
<td>41 GiB</td>
<td>1.30 s</td>
<td>94 %</td>
</tr>
<tr>
<td>4</td>
<td>11 h</td>
<td>46 GiB</td>
<td>1.00 s</td>
<td>93 %</td>
</tr>
<tr>
<td>8</td>
<td>10 h 15 min</td>
<td>51 GiB</td>
<td>0.91 s</td>
<td>92 %</td>
</tr>
<tr>
<td>16</td>
<td>9 h 45 min</td>
<td>63 GiB</td>
<td>0.85 s</td>
<td>92 %</td>
</tr>
<tr>
<td>32</td>
<td>9 h 30 min</td>
<td>82 GiB</td>
<td>0.84 s</td>
<td>92 %</td>
</tr>
<tr>
<td>64</td>
<td>9 h 30 min</td>
<td>116 GiB</td>
<td>0.83 s</td>
<td>92 %</td>
</tr>
</tbody>
</table>

Kira ⊕ Fermat | 82 h    | 147 GiB | -                  | -                  |

- The runtime of the probes is dominated by the forward elimination
- 48 cores each with hyper-threading disabled
- Coefficient arrays bring sizeable effects in exchange for main memory
Main feature: finite field reconstruction

Runtime reduction with MPI

Option run_firefly: true and Intel® MPI is used

The first prime number suffers in the performance because FireFly cannot process arbitrary probes

New probes are scheduled based on intermediate results

Remark: the user should use less nodes for the first prime number

<table>
<thead>
<tr>
<th># nodes</th>
<th>Runtime</th>
<th>Speed-up</th>
<th>CPU efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18 h</td>
<td>1.0</td>
<td>95%</td>
</tr>
<tr>
<td>2</td>
<td>10 h 15 min</td>
<td>1.8</td>
<td>87%</td>
</tr>
<tr>
<td>3</td>
<td>7 h 15 min</td>
<td>2.5</td>
<td>82%</td>
</tr>
<tr>
<td>4</td>
<td>5 h 45 min</td>
<td>3.1</td>
<td>76%</td>
</tr>
<tr>
<td>5</td>
<td>5 h 30 min</td>
<td>3.3</td>
<td>65%</td>
</tr>
<tr>
<td>Kira ⊕ Fermat</td>
<td>82 h</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Including $d$, the reduction of the double-pentagon topology is a six variable problem.

We use the system of equations is in block-triangular form taken from [Xin Guan, Xiao Liu, Yan-Qing Ma, 2019], which is of the size of 72 MB, best value I could find comparing to other methods. And no simplifications where yet applied.

We benchmark the reduction of all integrals including five scalar products.
Double-pentagon topology in five-light-parton scattering II

- FireFly’s factor scan improves the denominators
- \(-\text{bunch\_size} = 128\) option is used to improve the speed
- 40 cores with hyperthreading enabled
- The most complicated master integral coefficient has a maximum degree in the numerator of 87 and in the denominator of 50
- The database of the reduction occupies 25 GiB of disk space
- The number of required probes \(10^7\) is computed fast due to the block triangular structure of the system of equations

[Xin Guan, Xiao Liu, Yan-Qing Ma, 2020]

- Main memory reduction can be achieved with the options \texttt{iterative\_reduction} or by reducing the \(-\text{bunch\_size}\) option
- We use Horner form to accelerate the parsing for the coefficients

- The **block triangular form** is much better suited for the reduction than a naive IBP system of equations as generated by Kira
- Reduction tables are available upon request
Upcoming Features in next Kira Version

Kira’s, development release

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

- Interface to user defined IBP-identities
- Documentation to symbolic IBP-identities
- On https://hepforge.kira.org we provide a static linked Kira executable
- We have launched a Wiki and a best practice summary on gitlab
Summary and Outlook

- Many parallelization improvements
- Kira is an all-rounder for multi-scale as well as for multi-loop computations
- Kira utilize the finite field methods and helps to tailor it to your needs
- The examples should help you to find the balance yourself
- Bunches should be used if there is unused memory on the system
- MPI if there are more computers or cluster nodes available
- Many features are driven by the high energy physics community demands
- We plan to go for the block triangular form: `run_triangular: block`, which finds a small and fast to evaluate system of equations for general topologies [Xin Guan, Xiao Liu, Yan-Qing Ma, 2020]! It is a paper about auxiliary mass flow integrals, but the ideas to generate block triangular form relations are independent from this formalism
- Be prepared to find more algorithmic improvements in Kira in the near future