Kira, Feynman integral reduction program – new developments

(common work with: Fabian Lange, Philipp Maierhöfer)

Loops and Legs in Quantum Field Theory 2022

Johann Usovitsch

29. April 2022

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
- **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
  - **Differential equations** [Kotikov, 1991; Remiddi, 1997; Henn, 2013; Argeri et al., 2013; Lee, 2015; Meyer, 2016; Moriello, 2019; Hidding, 2020] or **difference equations**[Laporta, 2000; Lee, 2010]
  - Use the method of **sector decomposition** [Heinrich,2008] (pySecDec/expansion by regions [Heinrich, et al., 2021] 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, arXiv:2107.01864], AMFlow [Xiao Liu,Yan-Qing Ma, 2022]
  - **Series expansions method** [Moriello, 2019], DiffExp [Hidding, 2020]

Outline

Introduction

- Kira is a linear solver for sparse linear system of equations with main application to **Feynman integral reduction**
- Kira automatically generates the system of equations and applies symmetries between several integrals and topologies
- We use finite field methods [von Manteuffel, Schabinger, 2015, Peraro, 2016] and the finite field reconstruction libray FireFly [Klappert, Lange, 2019, Klappert, Klein, Lange, 2020]
- **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 (I will talk about this at the end of the talk)

Integration-by-parts (IBP) identities

$$I(a_1,\ldots,a_5) = \int \frac{d^D l_1 d^D l_2}{[l_1{}^2 - m_1^2]^{a_1}[(p_1+l_1)^2]^{a_2}[l_2{}^2]^{a_3}[(p_1+l_2)^2]^{a_4}[(l_2-l_1)^2]^{a_5}}$$

$$\begin{aligned}
\int d^D \boldsymbol{l_1} \ldots d^D \boldsymbol{l_L} \frac{\partial}{\partial(\boldsymbol{l_i})_\mu}\left((q_j)_\mu \frac{1}{[P_1]^{a_1}\ldots[P_N]^{a_N}}\right) &= 0 \\
c_1(\{a_f\},\vec{s},D)I(a_1,\ldots,a_N\boldsymbol{-1}) + \cdots + c_m(\{a_f\},\vec{s},D)I(a_1\boldsymbol{+1},\ldots,a_N) &= 0
\end{aligned}$$

$q_j = p_1,\ldots,p_E,l_1,\ldots,l_L$
$\vec{s} = (\{s_i\},\{m_i^2\})$
m number of terms generated by one IBP identity

Reduction: 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: Seek for recursion relations, LiteRed [Lee, 2012]
- a_f = integers: Sample a system of equations, **Laporta algorithm** [Laporta, 2000]

Laporta algorithm challenges

- The system of equations generated the Laporta way contains many redundant equations ($\sim$ up to billions or 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!**

Finite field reconstruction: Kira + FireFly

- Reconstruction of multivariate rational functions from **samples over finite integer fields** [Schabinger, von Manteuffel, 2014][Peraro, 2016]
- Public implementations available: FireFly [Klappert, Lange, 2019][Klappert, Klein, Lange, 2020], FIRE 6 [Smirnov, Chukharev, 2019] and FiniteFlow [Peraro, 2019]
- **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!

Run time examples

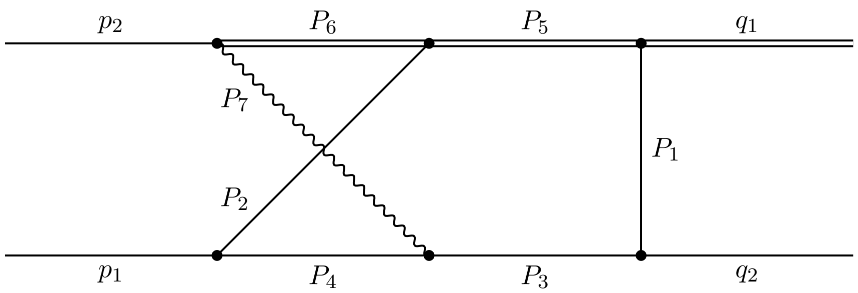
$$P_1 = k_1^2, \quad P_2 = k_2^2, \quad P_3 = k_3^2, \quad P_4 = (p_1 - k_1)^2, \quad P_5 = (p_1 - k_2)^2, \quad P_6 = (p_1 - k_3)^2, \quad P_7 = (p_2 - k_1)^2,$$
$$P_8 = (p_2 - k_2)^2, \quad P_9 = (p_2 - k_3)^2, \quad P_{10} = (k_1 - k_2)^2, \quad P_{11} = (k_1 - k_3)^2, \quad P_{12} = (k_2 - k_3)^2,$$

$$p_1^2 = z z_b, \quad p_2^2 = 1, \quad p_1 p_2 = (1 - z)(1 - z_b)$$

We chose $r = 17$ and $s = 0$ for the benchmark

Mode	Runtime	Memory	Probes	CPU time per probe	CPU time for probes
`run_initiate`	5 h 20 min	128 GiB	-	-	-
`run_triangular` + `run_back_substitution`	> 14 d	~ 540 GB	-	-	-
`run_firefly: true`	6 d 3 h	670 GiB	108500	370 s	100 %
`run_triangular: sectorwise`	36 min	4 GiB	-	-	-
`run_firefly: back`	4 h 54 min	35 GiB	108500	12.2 s	100 %

Reducing the memory footprint with iterative reduction



$r = 7$ and $s = 4$

Mode	Iterative	Runtime	Memory
`Kira` ⊕ `FireFly`	-	18 h	40 GiB
	sectorwise	33 h 15 min	9 GiB

- `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

`--bunch_size=`	Runtime	Memory	CPU time per probe	CPU time for probes
1	18 h	40 GiB	1.73 s	95 %
2	14 h	41 GiB	1.30 s	94 %
4	11 h	46 GiB	1.00 s	93 %
8	10 h 15 min	51 GiB	0.91 s	92 %
16	9 h 45 min	63 GiB	0.85 s	92 %
32	9 h 30 min	82 GiB	0.84 s	92 %
64	9 h 30 min	116 GiB	0.83 s	92 %
`Kira` $\oplus$ `Fermat`	82 h	147 GiB	-	-

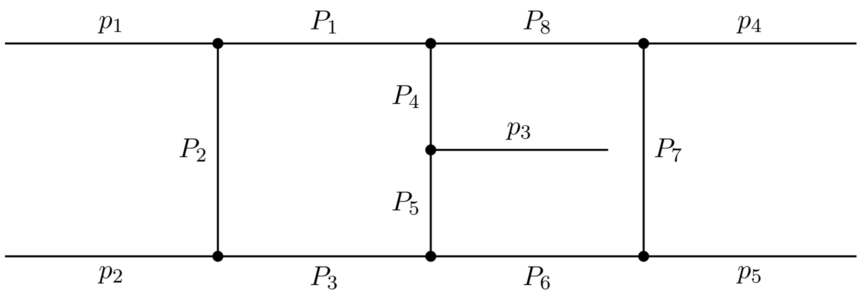
- The runtime of the probes is dominated by the forward elimination
- 48 cores each with hyper-threading disabled
- Coefficient arrays bring sizeable effects in exchange for main memory

Runtime reduction with MPI

# nodes	Runtime	Speed-up	CPU efficiency
1	18 h	1.0	95 %
2	10 h 15 min	1.8	87 %
3	7 h 15 min	2.5	82 %
4	5 h 45 min	3.1	76 %
5	5 h 30 min	3.3	65 %
`Kira` ⊕ `Fermat`	82 h	-	-

- 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

Double-pentagon topology in five-light-parton scattering I



Runtime	Memory	Probes	CPU time per probe	CPU time for probes
12 d	540 GiB	38278000	0.37 s	25 %

- Including d, the reduction of the double-pentagon topology is a six variable problem
- We use a system of equations which 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
- `-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 `iterative_reduction` or by reducing the `-bunch_size` option
- We use Horner form to accelerate the parsing for the coefficients

Double-pentagon topology in five-light-parton scattering III

- The new option `insert_prefactors` would give a factor of 2 improvement in an overall performance if we use the denominators from [J.U, arXiv:2002.08173]. The method to compute these denominators is explained shortly in the summary of [J.U, arXiv:2002.08173], which relies on algebraic reconstruction methods pioneered in [arXiv:1805.01873, arXiv:1712.09737, arXiv:1511.01071]. A second approach to compute the denominator functions should be possible with finite field methods [Heller, von Manteuffel, arXiv:2101.0828].
- The **block triangular form** is much better suited for the reduction than a naïv IBP system of equations as generated by Kira
- Reduction tables are available upon request

Guan, Liu, Ma algorithm – construction of the block triangular form, see arXiv:1912.09294v3

- First step is the Ansatz: $I_1 c_1 + \cdots + I_N c_N = 0$, where I_i are the Feynman integrals and the c_i are polynomials.
- Second step is the Ansatz for the coefficients

  $$c_j(d, \vec{s}) = \sum_{i=0}^{d_{\max}} d^i \sum_{\vec{l} \in \Omega_{k_j}}^{k_{\max}} \hat{c}_j^{i,l_1,\ldots,l_M} s_1^{l_1} \cdots s_M^{l_M}$$

- $\Omega_{k_j} = \{\vec{l} \in \mathbb{N}^M | \sum_{j=0}^{M} l_j = k_j\}$
- We have a linear relation between integrals of different massdimension, thus k_i differ with respect to the integrals of our choice
- The $\hat{c}_j^{i,l_1,\ldots,l_M}$ are unknown rational numbers and are fixed by adjusting the $k_{\max}$ and $d_{\max}$

Guan, Liu, Ma algorithm

- To determine the unknowns $\hat{c}_j^{i,l_1,\ldots,l_M}$ we have to reduce the IBP-system to N master integrals generated the Laporta way as many times as the number of the unknowns $\hat{c}_j^{i,l_1,\ldots,l_M}$ are in the Ansatz.
- Each new sample generates N new non trivial equations.
- Some unknowns turn out to be $\hat{c}_j^{i,l_1,\ldots,l_M}$ undetermined and we can choose them arbitrary.
- **The result** is a system of equations in block triangular form containing as many equations as integrals, which we would like to reduce.
- The coefficients are polynomials of very low degree
- The rational numbers $\hat{c}_j^{i,l_1,\ldots,l_M}$ will be huge
- This system of equations is ideal for the finite field methods applied in Kira
- To implement this algorithm, we estimate roughly 1 week full time work
- We have all tools available

Feynman integral computation with DiffExp

- The naïv usage of DiffExp with arbitrary Feynman integrals is guaranteed to fail:
  - either because we do not know the boundary terms
  - or because, if one works with a basis chosen with a Laporta algorithm one will encounter singular matrices in the dimensional regularization parameter in the system differential equations
- Solution [Ievgen Dubovyk, Ayres Freitas, Janusz Gluza, Krzysztof Grzanka, Martijn Hidding, arXiv:2201.02576]:
  - we work with quasi finite basis of master integrals [Panzer, 2015, von Manteuffel, Panzer, Schabinger, 2015], which fixes the matrices to be finite
  - and we run DiffExp with boundary terms fixed numerically
  - This makes the computation of Feynman integrals with DiffExp **automatic** if we know the boundary terms numerically and we have the IBP reductions
- Numerical boundary terms:
  - Get them from pySecDec in Euclidean regions
  - Get them from AMFlow

Upcoming Features in next Kira Version

Kira's, development release

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

- On https://hepforge.kira.org we provide a static linked Kira executable
- We have a Wiki and a best practice summary on gitlab
- 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]!
- We have automated the permutation of propagators to accelerate the reduction time `permutation_option: 1`
- We improved the speed for the export of the results into the FORM output

Summary and Outlook

- New version of Kira have always new 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
- Computing the block triangular form will allow us to tackle new interesting state of the art problems!
- Explained the automatic usage of DiffExp