Feynman Integral Reduction with Modular Arithmetic

Philipp Maierhöfer

Albert-Ludwigs-Universität Freiburg

A Loop Summit
Cadenabbia, Italy
27 July 2021

In collaboration with
J. Klappert, F. Lange and J. Usovitsch

Outline

Introduction

Express required integrals of the form

$$T(a_1, \ldots, a_N) = \int \frac{\mathrm{d}^d \ell_1 \cdots \mathrm{d}^d \ell_L}{D_1^{a_1} D_2^{a_2} \cdots D_N^{a_N}},$$

as linear combinations of a small set of master integrals ("reduction").
Linear relations between integrals with different a_i:

- Integration-by-parts identities [Chetyrkin, Tkachov '81],
- Lorentz invariance identities [Gehrmann, Remiddi '00],
- Symmetry relations (finder based on graph polynomials, see [Pak '11])

Arbitrary propagators $D_i = \sum \ell_j \cdot \ell_k + \sum \ell_j \cdot p_k + inv.$ allowed, but symmetrisation with $a_i < 0$ only works if the momentum flow is given.

Laporta algorithm

Define integral ordering, generate equations and use Gaussian elimination.

Public (general purpose) implementations

`Reduze` [v. Manteuffel], `FIRE` [A. Smirnov], `Kira` [Klappert, Lange, PM, Usovitsch].

5 Years of Kira

Kira collaboration founded at Loops & Legs 2016, merging the previously unpublished projects `pyRed` [PM] and `Kira` [Usovitsch].

→ Boost Laporta's algorithm:

- Analyse the system using modular arithmetic.
- Exploit freedom in Gaussian elimination and simplifications of coefficients to increase efficiency.

Often an order of magnitude improvement wrt. other available solutions.

Central new feature of Kira 2:

Interpolation of multivariate rational functions (see [Peraro '16]) and rational reconstruction with `FireFly` [Klappert, Lange '19].

For similar approaches see `FIRE 6` [A. Smirnov, Chukharev] (≤ 2 variables), `FinRed` [v. Manteuffel] (private), `FiniteFlow` [Peraro '19] (reconstruction library in `C++` & `Mathematica` toolbox).

FireFly

Main tasks performed by FireFly

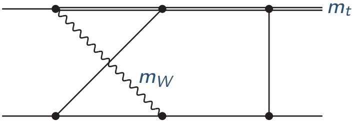
- Multivariate rational interpolation: Zippel algorithm.
- Requires univariate rational interpolation: Racing algorithm Ben-Or/Tiwari (sparse, $\mathcal{O}(2T)$, $T = \#$ terms) vs. Newton (dense). Different algorithms can win for different variables!
- Zippel algorithm improved with "pruning" to never perform worse than nested Newton (used in `FiniteFlow`).
- Rational reconstruction: Chinese remainder theorem.
- Search for univariate factors (requires `FLINT` [Hart, Johansson, Pancratz]).

`FireFly` asks a "black-box function" to evaluate "probes", i.e. to solve the system for a given prime and numerical values of the variables until all coefficient functions are fully reconstructed.

Parallelisation with MPI

Rational interpolation over finite fields opens the door to massive parallelisation, because probes can be evaluated independently.

Example:



# 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` $\oplus$ `Fermat`	82 h	-	-

$r = 7$, $s = 4$, $2\times$ Intel Xeon Platinum 8160 per node (48 cores)

At some point, the parallelisation is hampered by the fact that the sparse algorithms can't process arbitrary probes.

Coefficient arrays

The task of the solver is to evaluate the coefficients.
In some sense, everything else can be considered overhead.

→ Reduce overhead by evaluating multiple coefficients at once.

`--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` ⊕ `Fermat`	82 h	147 GiB	-	-

Should always be used unless memory is a bottleneck.

IBP systems

Naïve observations

- Systems are sparse.
- Block triangular, but large blocks (one per sector).
- Many integrals that won't be fully reduced (one dot and/or one scalar product more than the seed integrals).
- Blocks may need information from higher blocks for full reduction ($\rightarrow$ Kira's `magic_relations` option, use with care).
- Growth of coefficients (& number of terms) in intermediate steps.

Analytic solution

Back substitution dominates the runtime.

Numerical solution

Forward elimination dominates runtime, but low computational complexity wrt. the number of equations ($\mathcal{O}(n^{\sim 1})$ due to sparsity). Back substitution is very fast.

Hybrid analytic/numerical solver

Analytic forward elimination + numerical back substitution can pay off through drastically reduced sampling time (and reduced memory requirements).

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

$P_1 = k_1^2$, $P_2 = k_2^2$, $P_3 = k_3^2$, $P_4 = (p_1 - k_1)^2$, $P_5 = (p_1 - k_2)^2$, $P_6 = (p_1 - k_3)^2$, $P_7 = (p_2 - k_1)^2$, $P_8 = (p_2 - k_2)^2$, $P_9 = (p_2 - k_3)^2$, $P_{10} = (k_1 - k_2)^2$, $P_{11} = (k_1 - k_3)^2$, $P_{12} = (k_2 - k_3)^2$, $p_1^2 = zz_b$, $p_2^2 = 1$, $p_1 p_2 = (1 - z)(1 - z_b)$, **sector 4095**, $r = 17$, $s = 0$. | **2× Intel Xeon Gold 6138 (40 cores)**

From experience this is a good strategy for up to 3 scales.

Better integral relations than IBP?

How to construct systems that are fast to solve
if the hybrid solver is not feasible?

→ Find linear relations between Feynman integrals that don't involve increased powers of propagators (scalar products) ("unitarity compatible IBPs" or "IBP generating vectors").

Systematic construction: Syzygies and/or algebraic geometry.
[Gluza, Kajda, Kosower '10; Schabinger '11; Ita '15; Larsen, Zhang '15; Böhm, Georgoudis, Larsen, Schulze, Zhang '17; Böhm, Georgoudis, Larsen, Schönemann, Zhang '18; Bendle, Böhm, Decker, Georgoudis, Pfreundt, Rahn, Wasser, Zhang '19; von Manteuffel, Panzer, Schabinger '20; . . .]

Alternative: generation of block-triangular systems with an ansatz
[Guan, Liu, Ma '19].

Custom integral relations [based on Guan, Liu, Ma 1912.09294]

An "almost" triangular system can be constructed from an ansatz:

With an appropriately chosen set of integrals $\mathcal{I}_i$, $i = 1, \dots i_{\max}$, write

$$0 = \sum_i P_i \mathcal{I}_i, \qquad P_i = \sum_{a_0, a_1, \dots \geq 0} q_{i, a_0, a_1, \dots} \cdot d^{a_0} s_1^{a_1} s_2^{a_2} \dots$$

with the dimensionful invariants s_k, $a_0 \leq a^0_{\max}$, $a_1 + a_2 + \dots \leq a_{\max}$.

Suppose we already know the reduction to the master integrals M_j, then

$$0 = \sum_i P_i \mathcal{I}_i = \sum_j Z_j M_j \quad \Rightarrow \quad Z_j = 0$$

gives us linear constraints on the integer polynomial coefficients $q_{i, a_0, a_1, \dots}$.

$\rightarrow$ Solve for these coefficients, choosing some of them as free parameters. Exploit freedom to find several independent relations from the same ansatz.

Custom integral relations (cont'd)

What does "appropriate" mean in the choice of the $\mathcal{I}_i$?

$$0 = \sum_i P_i \mathcal{I}_i = \sum_j Z_j M_j \quad \Rightarrow \quad Z_j = 0$$

Choose the integrals $\mathcal{I}_i$ such that

- non-trivial linear relations between them exist (involving the highest $\mathcal{I}_i$ that we want to solve this identity for) ...
- ... with polynomials P_i of low degree.

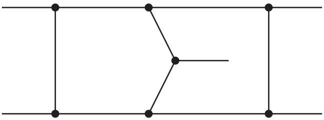
We don't need the full reduction, but only numerical samples to reconstruct the relations $Z_j = 0$.

Since the degree of the polynomials is small, only few pobes are needed to fit the integer coefficients. $\Rightarrow$ Construction of the relations is cheap.

May achieve similarly suitable systems for efficient sampling as approaches based on syzygies, but with much simpler mathematics.

Example

Massless double pentagon (5-scale problem) [system taken from 1912.09294.]



$\sim 4 \cdot 10^7$ probes,
0.37 s per probe,
25 % CPU time for probes (!)
12 days runtime on 40 cores

Most complicated coefficient has degree $\frac{87}{50}$, without factor scan $\frac{87}{85}$.

Further potential for optimisation:

- Smaller block sizes can be achieved with better $\mathcal{I}_i$ choice.
- Expect another factor 2 with factorised denominators [Usovitsch '20; A. Smirnov, V. Smirnov '20, Heller, von Manteuffel '21].
- Better choice of master integrals?

Kira 2.2

Kira 2.2 ist out since 30 June 2021

- Mostly bug fixes.
  Most importantly: `Kira 2.1` missed some symmetries.
- Improved output options/formats.
- Removed limitations of the user-defined system solver.
- Started implementing tests, but still a lot to do for good coverage.

Check it out at `https://gitlab.com/kira-pyred/kira.git`.

- Binaries for Linux available on `https://kira.hepforge.org`
  (all features enables except MPI).
- For bug reports and feature requests use the issue tracker on GitLab.
- Check out the best practices wiki on GitLab.

Conclusions & Outlook

Kira 2 uses FireFly for rational interpolation & reconstruction

- Opened the door to various other new features.
- Different reduction strategies have a huge impact on the performance.
- Lots of screws to optimise and balance performance/memory usage.
- But not clear a priori which strategy is best for a given problem.

Going beyond ordinary IBPs

- Generation of block-triangular systems will be the next major feature.
- Shown to work efficiently. Implementation in C++ in progress.

Minor things to come soon: Differential equations / Custom templates to generate equations / (Auto?) optimisation of the integral ordering.

Everything is open source and usable.