Developments since Kira 2.0

Fabian Lange[1,2], Philipp Maierhöfer[3] and Johann Usovitsch[4⋆]

1 Institut für Theoretische Teilchenphysik, Karlsruhe Institute of Technology (KIT), Wolfgang-Gaede Straße 1, 76128 Karlsruhe, Germany
2 Institut für Astroteilchenphysik, Karlsruhe Institute of Technology (KIT), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
3 Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, 79104 Freiburg, Germany
4 Theoretical Physics Department, CERN, 1211 Geneva, Switzerland
* johann.usovitsch@cern.com

October 30, 2021

15th International Symposium on Radiative Corrections: Applications of Quantum Field Theory to Phenomenology, FSU, Tallahasse, FL, USA, 17-21 May 2021
doi:10.21468/SciPostPhysProc.?

Abstract

Last year we released version `2.0` of the Feynman integral reduction program `Kira`. In this contribution we first report on changes and new features since then and, secondly, on new features for upcoming releases.

1 Introduction

The reduction of Feynman integrals to a smaller set of master integrals is a crucial part of high-precision calculations in theoretical particle physics. This can be achieved by using linear relations between integrals as provided by *integration-by-parts* (IBP) identities [1, 2] and *Lorentz-invariance* (LI) identities [3]. To this end, `Kira` is an implementation of the Laporta algorithm [4] and was first published in Ref. [5]. The algebraic expressions were solely handled by the computer algebra system `Fermat` [6]. Last year, we released version `2.0` of `Kira` [7], which, as headline feature, additionally offers finite field interpolation and rational reconstruction techniques (see Refs. [8, 9] in the context of IBPs) provided by the library `FireFly` [10, 11]. There are several other public implementations of the Laporta algorithm, see e.g. Refs. [12–14].

In this contribution we highlight some of the new features already implemented since last year's release of version `2.0` in Sect. 2 and then present some of the planned features for upcoming releases in Sect. 3. We assume that the reader is familiar with `Kira` and refer to Refs. [5, 7] otherwise.

2 New and changed features since Kira 2.0

In this section we highlight some of the new and changed features implemented in the versions `2.1` and `2.2` of `Kira`:

- We introduced a version number for the database. Databases produced prior to `Kira 2.1` cannot be used for export jobs with `kira2form` and `kira2formfill` unless they are upgraded with the command line argument `--force_database_format=fermat|firefly`. Three cases have to be distinguished:

  1. If the database was produced with `Fermat`, use the command line argument `--force_database_format=fermat`.
  2. If the database was produced with `FireFly` without `insert_prefactors`, use the command line argument `--force_database_format=firefly`.
  3. If the database was produced with `FireFly` and `insert_prefactors`, it cannot be upgraded. Instead, remove the database and rerun the job that produced it. If `firefly_saves` still exists, most of the computational steps are skipped.

  Export jobs with `kira2math` and `kira2file` are not affected by this change.

- With the new `kira2formfill` job the results can now be exported as `fill` statements to fill `TableBase` objects in `FORM` [15].

- The numerators of the rational functions exported with `kira2form` and `kira2formfill` are now enclosed by `num()`.

- The prefactors submitted through the option `insert_prefactors` are now enclosed by `prefactor[]` when employing `kira2math` and by `prefactor()` when using `kira2form` and `kira2formfill`.

- We added the option `run_initiate: masters` to stop the reduction after the master integrals have been identified without writing the system to disk.

- The weight bits for user-defined systems with indexed integrals `T[a,b,c,...]` are now automatically adjusted. Before, Kira crashed if the weights did not fit into the default representation.

- We changed the output format for the master integrals to make it more readable by both humans and other programs. This mainly affects the output to the console, `kira.log`, and `results/<TOPOLOGY>/masters(.final)`.

- The command line option `--parallel/-p` now accepts the arguments `physical` and `logical` to exploit all physical or logical cores, respectively. It is still possible to manually choose any number of threads. If the option is given to `Kira` without argument, it uses `physical` by default. Under `macOS`, `physical` refers to the number of logical cores.

Finally, we emphasize that we opened a wiki at

https://gitlab.com/kira-pyred/kira/-/wikis/Home

to serve as first contact point for users. It already contains sections about best practice and troubleshooting.

3 Features in upcoming releases

In this section we briefly present improved and new features for the upcoming releases of `Kira`.

3.1 Faster export of the results

The export performance of the results to different formats, especially for `FORM` [15], is improved significantly. This is achieved by replacing the pipe to `Perl` by `C++ std::regex`. Tab. 1 shows the improvements for the example `topo7`.

Table 1: Export of results to different formats with and without reconstructing the mass set to 1 for the example `topo7` with $r = 7$ and $s = 2$. The numbers are still preliminary.

Mode	`Kira 2.2`	`Kira 2.X`	Improvement factor
`-kira2math`	0.22 s	0.22 s	1
`-kira2form`	12.21 s	0.43 s	28
`-kira2math` `reconstruct_mass: true`	8.33 s	2.41 s	3.5
`-kira2form` `reconstruct_mass: true`	20.61 s	2.61 s	8

3.2 Reorder propagators to increase the performance

The definition of the topology can have an impact on the size of the system generated by `Kira` and on the performance of the reduction. Not only redefinitions of the propagators can change the performance, even a simple reordering can drastically impact it. In Ref. [16] we described rough guidelines for the topology definition, which we repeat in the following for clarity:

1. Always define `top_level_sectors` for the topologies. This ensures that sectors are only mapped on subsectors of the defined top-level sectors. If you are trying to find relations by reducing sectors which are higher than those in the physical problem, the higher sectors must be included in the top-level sectors or the `magic_relations` have to be enabled in order to symmetrize them.

2. If a topology has only one top-level sector, order the propagators such that the sector number is $2^n - 1$ (if the sector has n lines), i.e. the propagators at the end of the list appear only as irreducible scalar products.

3. Define the propagators in such a way that propagators have short linear combinations of loop momenta and the shortest linear combinations appear earlier in the list of propagators.

4. Analogously, keep linear combinations of external momenta short with ascending length.

5. Massless propagators should appear earlier in the list.

These guidelines are based on experience and will not always produce the best definition. However, they should serve as a good starting point. If the topology definition is crucial to complete the reduction, one should run smaller jobs first to experimentally find the preferred definition.

In Ref. [17] one of the authors of this contribution encountered some extreme examples in the calculation of four-loop on-shell integrals for the relation of a heavy quark defined in the $\overline{\text{MS}}$ and the on-shell scheme allowing for a second non-zero quark mass in the loops. We show one example topology in Fig. 1. The setup used in Ref. [17] automatically defines its

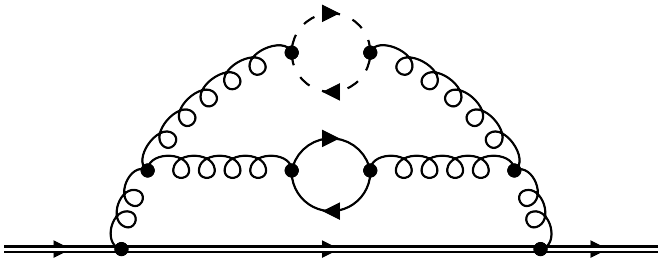

Figure 1: Example topology from Ref. [17]. The double line represents quarks with mass m_1, the single line quarks with mass m_2, and the dashed line massless quarks. Produced with `FeynGame` [18].

propagators as

$$\begin{gathered} P_1 = p_1^2 - m_1^2, \quad P_2 = (p_1 - q)^2, \quad P_3 = (p_2 + p_3)^2, \quad P_4 = (p_1 - p_3 + p_4 - q)^2, \\ P_5 = (p_3 - p_4)^2, \quad P_6 = (p_3 + q)^2, \quad P_7 = (p_2 + p_4)^2, \quad P_8 = p_2^2 - m_2^2, \\ P_9 = (-p_1 + p_2 + p_3 - p_4 + q)^2 - m_2^2, \quad P_{10} = p_3^2, \quad P_{11} = p_4^2, \quad P_{12} = (p_1 + p_3)^2, \\ P_{13} = (p_1 + p_4)^2, \quad P_{14} = (p_1 + p_2)^2, \end{gathered} \tag{1}$$

where p_i are the loop momenta, q is the external momentum, and m_1, m_2 are the two quark masses. P_3, P_6, P_7, P_{12}, P_{13}, and P_{14} are auxiliary propagators which only appear in the numerators. With this definition, `Kira` generates 10 204 834 linearly independent equations with 436 973 786 terms in total to reduce the selected integrals and one reduction over a finite field with `pyRed` takes about 12 771 s. Following the guidelines above and reordering the propagators to

$$\begin{gathered} P_1 = p_3^2, \quad P_2 = p_4^2, \quad P_3 = p_1^2 - m_1^2, \quad P_4 = p_2^2 - m_2^2, \quad P_5 = (p_1 - q)^2, \quad P_6 = (p_3 - p_4)^2, \\ P_7 = (p_1 - p_3 + p_4 - q)^2, \quad P_8 = (-p_1 + p_2 + p_3 - p_4 + q)^2 - m_2^2, \quad P_9 = (p_3 + q)^2, \\ P_{10} = (p_2 + p_4)^2, \quad P_{11} = (p_2 + p_3)^2, \quad P_{12} = (p_1 + p_3)^2, \quad P_{13} = (p_1 + p_4)^2, \\ P_{14} = (p_1 + p_2)^2 \end{gathered} \tag{2}$$

only 3 752 490 equations with 57 696 557 terms are generated, corresponding to a reduction by 63 % and 87 %, respectively. The runtime of a reduction with `pyRed` reduces to 43 s, i.e. the performance increases by a factor of 297. We want to stress that this is certainly an extreme example and the performance gains can even be negligible for other examples.

For the convenience of the users of `Kira`, we plan to implement the option to reorder the propagators internally, i.e. to allow the internal order to differ from the definition in `integralfamilies.yaml`. This allows for reaping the potential performance gains without keeping track of the order when communicating with other programs. We plan to implement two versions: first, allow the users to set the ordering manually and, secondly, automatically reorder the propagators according to the guidelines above.

3.3 Generate a system of equations for later reduction as user-defined system

With the option `run_initiate: input`, the generated system of linearly independent equations is stored in the `Kira` readable `.kira` format for further processing. In contrast to the

existing option `generate_input`, which was introduced to reduce the memory requirements by considering only a subset of the sectors at the same time, with the option `run_initiate: input` the user is able to select equations to reduce only requested integrals. One idea is to use these files `.kira` in combination with additionally generated `.kira` files containing extra relations provided by the user. The additional relations can be extra symmetry relations, a system of differential equations, or amplitudes.

3.4 Set variables to rational numbers and/or choose a prime field

The complexity of reductions can be greatly reduced by setting some or all of the variables to rational numbers. While the result is no longer general, it might be helpful to first reduce the simpler problem to get a feeling for the reduction and the structure of the result. We plan to allow the users to set the variables to rational numbers at different stages of `Kira` so that, for example, the system can be generated in full generality and then be solved with the variables replaced by rational numbers.

Additionally, we want to allow the users to choose a prime and solve the system one or several times over that prime field. By storing these results to disk, they could then be processed by other programs if the users are only interested in the solutions over specific prime fields but not the algebraic result.

4 Conclusions

In this contribution we presented the new and changed features already implemented since the release of `Kira 2.0` and some of the features planned for upcoming releases. We hope that these, together with the ever-present bug fixes, will improve the experience of users of `Kira`.

Acknowledgments

Funding information The research of F.L. was supported by the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) through the Collaborative Research Centre TRR 257 funded through grant 396021762.

References

[1] F. V. Tkachov, *A theorem on analytical calculability of 4-loop renormalization group functions*, Phys. Lett. B **100**, 65 (1981), doi:10.1016/0370-2693(81)90288-4.

[2] K. G. Chetyrkin and F. V. Tkachov, *Integration by parts: The algorithm to calculate β-functions in 4 loops*, Nucl. Phys. **B192**, 159 (1981), doi:10.1016/0550-3213(81)90199-1.

[3] T. Gehrmann and E. Remiddi, *Differential equations for two-loop four-point functions*, Nucl. Phys. **B580**, 485 (2000), doi:10.1016/S0550-3213(00)00223-6, hep-ph/9912329.

[4] S. Laporta, *High-precision calculation of multiloop Feynman integrals by difference equations*, Int.J.Mod.Phys. **A15**, 5087 (2000), doi:10.1016/S0217-751X(00)00215-7, hep-ph/0102033.

[5] P. Maierhöfer, J. Usovitsch and P. Uwer, *Kira—A Feynman integral reduction program*, Comput. Phys. Commun. **230**, 99 (2018), doi:10.1016/j.cpc.2018.04.012, 1705.05610.

[6] R. H. Lewis, *Computer Algebra System Fermat*, URL https://home.bway.net/lewis.

[7] J. Klappert, F. Lange, P. Maierhöfer and J. Usovitsch, *Integral reduction with Kira 2.0 and finite field methods*, Comput. Phys. Commun. **266**, 108024 (2021), doi:10.1016/j.cpc.2021.108024, 2008.06494.

[8] A. von Manteuffel and R. M. Schabinger, *A novel approach to integration by parts reduction*, Phys. Lett. **B744**, 101 (2015), doi:10.1016/j.physletb.2015.03.029, 1406.4513.

[9] T. Peraro, *Scattering amplitudes over finite fields and multivariate functional reconstruction*, JHEP **12**, 030 (2016), doi:10.1007/JHEP12(2016)030, 1608.01902.

[10] J. Klappert and F. Lange, *Reconstructing rational functions with FireFly*, Comput. Phys. Commun. **247**, 106951 (2020), doi:10.1016/j.cpc.2019.106951, 1904.00009.

[11] J. Klappert, S. Y. Klein and F. Lange, *Interpolation of dense and sparse rational functions and other improvements in FireFly*, Comput. Phys. Commun. **264**, 107968 (2021), doi:10.1016/j.cpc.2021.107968, 2004.01463.

[12] C. Anastasiou and A. Lazopoulos, *Automatic integral reduction for higher order perturbative calculations*, JHEP **07**, 046 (2004), doi:10.1088/1126-6708/2004/07/046, hep-ph/0404258.

[13] A. von Manteuffel and C. Studerus, *Reduze 2 – Distributed Feynman Integral Reduction* (2012), 1201.4330.

[14] A. V. Smirnov and F. S. Chukharev, *FIRE6: Feynman Integral REduction with modular arithmetic*, Comput. Phys. Commun. **247**, 106877 (2020), doi:10.1016/j.cpc.2019.106877, 1901.07808.

[15] J. A. M. Vermaseren, *New features of FORM* (2000), math-ph/0010025.

[16] P. Maierhöfer and J. Usovitsch, *Kira 1.2 Release Notes* (2018), 1812.01491.

[17] M. Fael, F. Lange, K. Schönwald and M. Steinhauser, *A semi-analytic method to compute Feynman integrals applied to four-loop corrections to the* $\overline{\text{MS}}$*-pole quark mass relation*, JHEP **09**, 152 (2021), doi:10.1007/JHEP09(2021)152, 2106.05296.

[18] R. V. Harlander, S. Y. Klein and M. Lipp, *FeynGame*, Comput. Phys. Commun. **256**, 107465 (2020), doi:10.1016/j.cpc.2020.107465, 2003.00896.