FR-PHENO-2018-15
TCDMATH 18-18

Kira 1.2 Release Notes

Philipp Maierhöfer[*,a] and Johann Usovitsch[†,b]

[a] *Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, 79104 Freiburg, Germany*
[b] *Trinity College Dublin, School of Mathematics, Dublin 2, Ireland*

Abstract

We present the Feynman integral reduction program `Kira 1.2` and describe its new features and other changes w.r.t. the previous versions. The main new features include a much faster equation generator, more flexible seed specification options, several predefined integral orderings, the reduction of systems of user-provided equations and a new technique to simplify coefficients by sampling variables over integers. Furthermore, we provide a collection of recommendations on how to use the program efficiently. This version has overall improvements in runtime for all reduction tasks compared to the previous versions of `Kira`.

[*]E-mail: philipp.maierhoefer@physik.uni-freiburg.de
[†]E-mail: usovitsj@maths.tcd.ie

1 Introduction

`Kira`, first introduced in [1], is an implementation of the Laporta algorithm [2] to reduce Feynman integrals to a basis of master integrals. In this approach, large systems of relations between Feynman integrals, namely integration-by-parts [3] and Lorentz invariance [4] identities, and symmetry relations, are generated and solved by Gaussian elimination. A complexity criterion defines an ordering on the Feynman integrals, so that more complicated integrals are systematically expressed in terms of simpler integrals. Laporta's algorithm has become a standard method for the integral reduction task in calculations of multi-loop scattering amplitudes e.g. for the Large Hadron Collider, and constitutes one of the bottlenecks in such calculations. Though alternative reduction techniques have been proposed and applied to specific problems, see e.g. [5–8], to date programs based on Laporta's algorithm [9–11] pose the only general purpose tools suited for large scale applications. The continuous improvement of these tools is thus essential to keep up with the increasing demands in the applications.

In `Kira`, the solver for systems of linear equations is assisted by modular arithmetic to remove redundant equations from the system and to select subsystems of equations in order to reduce specific selected integrals. The employed forward elimination tries to minimise the number of elimination steps, and the simplification of coefficients is optimised e.g. by recursive pairwise combinations of coefficients. The details of the algorithms used are described in [1].

In section 2 of this article we describe how to obtain `Kira` and how to compile and install it on your system. Section 3 explains new features, fixes and changed default behaviour in `Kira 1.2`. In section 4 we provide a collection of tips on how to use `Kira` efficiently. Appendices A and B provide further help regarding the installation of `Kira` and its dependencies.

Note that this is not a complete user manual. In particular, for the conventions used in the topology and kinematics definition, please refer to the original `Kira` publication [1]. Even though no systematic introduction on the structure of job files is provided in these release notes, together with the examples that are shipped with the source code, they should contain enough information to cover the majority of use cases.

2 Installation

Kira can either be built using the Autotools build system or the Meson build system [12]. We recommend Meson if a sufficiently recent version is available on your system.

2.1 Prerequisites

Platform requirements

Linux x86_64 or Mac OS X.

Compiler requirements

A C++ compiler supporting the C++14 standard and a C compiler supporting the C11 standard, e.g. gcc version 5.1 or later, or LLVM version 3.4 or later.

Dependencies

Kira requires the following packages to be installed on the system:

- GiNaC [13, 14] (which itself requires CLN [15]),
- yaml-cpp [16],
- zlib [17].

In addition, the program Fermat [18] is required to run Kira. If the Fermat executable is not found automatically by Kira or a specific Fermat installation should be used, the path to the Fermat executable can be provided by setting the environment variable FERMATPATH=/path/to/fer64.

Please note that GiNaC, CLN and yaml-cpp must have been compiled with the same compiler which is used to compile Kira. Otherwise the linking step will most likely fail. If you are using the system compiler, you can usually install these packages via your system's package manager. However, if you are using a different compiler, this means in practice that you also have to build these packages from source and set the environment variables C_PATH, LD_LIBRARY_PATH and PKG_CONFIG_PATH according to the installation prefix. See appendix B for more details.

2.2 Obtaining Kira

The latest version of Kira is available from our Git repository at GitLab under the URL https://gitlab.com/kira-pyred/kira. To obtain the source code of the latest release version, clone the repository with

```
git clone https://gitlab.com/kira-pyred/kira.git -b release
```

checking out the release branch. Release versions are also available as Git tags (starting with Kira 1.2). The master branch of the repository contains the latest pre-release version, receiving more frequent updates with new features and fixes. To obtain the source code of the latest pre-release version, clone the repository with

```
git clone https://gitlab.com/kira-pyred/kira.git
```

checking out the master branch. Please refer to the changelog for information about what changed w.r.t. the release. Packages of the release versions as compressed Tar archives are available from

`https://www.physik.hu-berlin.de/de/pep/tools`.

2.3 Compiling Kira with the Autotools build system

If you obtained Kira from the Git repository, you first need to run

```
autoreconf -i
```

Then compile and install with

```
./configure --prefix=/install/path
make
make install
```

where the optional `--prefix` argument sets the installation prefix.

2.4 Compiling Kira with the Meson build system

Since Kira 1.2 one may optionally use the Meson build system instead of Autotools to build Kira.

```
meson --prefix=/install/path builddir
cd builddir
ninja
ninja install
```

where `builddir` is the build directory. Specifying the installation prefix with `--prefix` is optional. Meson version 0.46 or later is recommended[1]. See appendix A for more information about how to obtain a current version of Meson.

[1] Some older versions may work as well, but this has not been tested.

3 Changes and new features in Kira 1.2

3.1 Build system

- A new build system based on `Meson` has been introduced.
- The dependency on `OpenMP` has been removed. Instead, `Kira` now uses threads directly.
- Support for Mac OS X has been improved. `Kira` now compiles with Apple's version of `LLVM` (which lacks `OpenMP` support).

3.2 New equation generator

The generator for integration-by-parts, Lorentz invariance and symmetry equations has been rewritten. The performance of the equation generator has been improved by typically three orders of magnitude.

3.3 Changed default options

- The option `data_file: false` in the job file is now default, i.e. the reduction rules are no more written to plain text files. This reduces disk usage by approximately a factor 2. The old behaviour can be restored by setting `data_file: true`.
- The option `conditional: true` in the job file is now default. With this option (introduced in `Kira 1.1`), aborted reductions are automatically resumed when the job is restarted.
- The command line option `--algebra` has been removed. This option was used to choose between two different algorithms to simplify a sum of terms. The choice is now made automatically for every simplification based on the number of terms in the sum and the size of the terms.

3.4 New seed notation

A more flexible notation for seeds has been introduced.

```
jobs:
  - reduce_sectors:
      reduce:
        - {topologies: [T1,...], sectors: [S1,...],
           r: rmax, s: smax, d: dmax}
        - {...} # can be used multiple times
```

- `rmax`: the maximal sum of positive propagator powers in the seed.
- `smax`: the maximal negative sum of negative propagator powers in the seed.
- `dmax` (optional): the maximal number of dots. On a sector with L lines (L is the number of propagators with positive integer exponents), seeds will be generated up to $r = \min(\texttt{rmax}, L + \texttt{dmax})$. If omitted, only `rmax` will limit r. This option should usually only be used in integral selectors, but not in IBP seeds.
- `topologies` (optional): `T1` is the name of a topology. Several topologies may be selected. If omitted, all topologies defined in the project will be selected.
- `sectors` (optional): `S1` is a sector number. Several sectors may be selected. If omitted, the sectors specified as `top_level_sectors` in the topology definition in `integralfamilies.yaml` will be used (individually for each topology).

The same notation can be used in `select_mandatory_recursively` and the export options `kira2form` and `kira2math`.

N.B.: The old seed notation may still be used. However, we recommend using the new notation. See the section about best practices for the advantages of the new notation.

3.5 Integral ordering

The integral ordering can now be chosen by the user from eight predefined orderings. The ordering can either be chosen on the command line with the option `--integral_ordering=<value>` or in the job file with the option `integral_ordering: <value>`.

See `examples/topo4/advanced_jobs_reduction.yaml` for an example. Previous versions of `Kira` used ordering 1 exclusively, which is still the default.

Integral orderings

1. (default) Scalar products are regarded as simpler as dots. If there is more than one master integral in a sector, this produces a basis with irreducible scalar products (negative indices) and no increased propagator powers (positive indices greater than 1). If master integrals with increased propagator powers appear this usually indicates that the number of scalar products in the seed was chosen too small.

2. Dots are regarded as simpler than scalar products. This produces a basis with only positive indices. If master integrals with irreducible scalar products appear this usually indicates that the value of `rmax` in the seed was chosen too small.

3. The first complexity criterion is the sum of dots and scalar products. If the sum is equal, scalar products are regarded as simpler.

4. Like 3., but if the sum is equal, dots are regarded as simpler.

Furthermore, in orderings 1–4, sectors are ordered w.r.t. the sector number. Orderings 5–8 are the same as 1–4, but with sectors ordered by the number of lines first. Furthermore, in orderings 5–8, subsectors of the defined top-level sectors are always regarded as simpler than sectors outside the top-level sectors with the same number of lines. This prevents a possible issue which is described in section 3.13.

Note that changing the integral ordering may affect the runtime significantly in either direction, depending on the topology and its exact parametrisation. We recommend to investigate the behavior in a test run.

3.6 Basis change

The feature `preferred_masters` (introduced in `Kira 1.1`) to provide a list of integrals which should be preferred as master integrals has been improved and bugs have been fixed:

- Eliminating redundant integrals from the list of preferred master integrals now works.

- In some cases, the mapping of integrals was done incorrectly and an integral with all indices zero appeared as a master integral.

Note that a posteriori basis changes are not supported, i.e. preferred masters must be given before the reduction is performed.

N.B.: The option `select_masters` is an alias of `preferred_masters`.

3.7 Topology initialisation

In the first run, all topologies which are defined in `integralfamilies.yaml` will be initialised. I.e. trivial sectors, symmetries and mappings will be determined for all topologies in the order in which they are defined. If an individual topology is reduced, master integrals will now be mapped to previous topologies, even if they are not reduced in the same run.

In previous `Kira` versions, only topologies selected in the job file were initialised (in the order they were given in the job file). This lead to an error if, in a subsequent run, a topology was added to the job file. Furthermore, the mapping to previous topologies was only done when the topologies were reduced together.

3.8 Improved symmetry detection

- The detection of symmetries due to the permutation of external momenta now takes into account sign flips of the momenta. Only symmetries that leave the invariants unchanged are considered.

- Previous versions of `Kira` discarded certain symmetries if they were expected to be redundant. This has been disabled, because in some cases symmetries which are not redundant were dropped. Now all symmetries are always taken into account.

- In certain cases, `Kira 1.1` tried to use symmetries which do not permute the propagators but shift their masses. While the symmetries are in principle valid, they are not handled correctly. Since these symmetries are unwanted in a reduction, they have been disabled.

3.9 Export of reduction rules to Mathematica or Form

In `Kira 1.1`, reduction rules for integrals in trivial sectors were missing in the exported file. This has been fixed.

3.10 Sectorwise forward elimination

For large systems it may be helpful to perform the forward elimination sectorwise. To activate this, set `run_triangular: sectorwise` (instead of `run_triangular: true`) in your job file. If `run_triangular: sectorwise` is set and a run is aborted before the triangular form is achieved, `Kira` will resume the forward elimination when the job is restarted.

3.11 Algebraic reconstruction

In reduction problems where the coefficients contain at least two variables, the new option `algebraic_reconstruction: true` can be used to enable the algebraic reconstruction of coefficients sampled over integers [7, 19, 20]. The algorithm is only applied during the back substitution.

If the option `algebraic_reconstruction: true` is set `Kira` will decide to which coefficients the algorithm will be applied based on the number of terms in the expression and the size of the expressions.
For an example see `examples/topo4/advanced_jobs_reduction.yaml`.

3.12 Solving user-defined systems of equations

One of the most requested features by `Kira` users is the possibility to use the equation solver on externally provided equations[2]. In `Kira 1.2` an interface is provided to solve systems of user-provided linear equations. The option `reduce_user_defined_system` comes in two flavours:

- Import the user-defined system of equations without preparing the config files `integralfamilies.yaml` and `kinematics.yaml`. The system will be sorted according to `Kira`'s equation ordering and then be solved. Topologies will be automatically defined with the names used in the equation file in the order of their appearance. For integrals with more than one integer index, the orderings defined in section 3.5 can be used.

- Prepare the config files in the `config` folder in the same way as you would in an ordinary integration-by-parts reduction. Based on the definitions in the config files, trivial integrals will be set to zero in the imported system. The system will be sorted and then solved.

Options like `select_integrals`, `preferred_masters` and those to control the individual reduction steps apply as usual.

In cases where the objects in the equations are not integer-indexed Feynman integrals or where a different ordering is desired than `Kira`'s build-in integral orderings, single-indexed objects may be used, e.g. `T[n]`, where `n` is a 32-bit signed integer. In the default ordering, `T[0]` will be regarded as the simplest object, followed by descending negative indices. Positive indices are regarded as more complex than any negative index with increased complexity for ascending values. This way, 2^{32} distinguishable objects may be defined per "topology".

Some examples can be found in `examples/userDefinedSystem1` and `examples/userDefinedSystem2`.

Possible applications include but are not limited to integrals appearing in amplitudes that do not have the standard form of Feynman integrals, or coefficients in the expansion of Feynman integrals in some parameter.

[2]Employing some undocumented tricks this has always been possible and in fact been done by several users, e.g. in [21].

3.13 Finding "magic relations"

Some relations between Feynman integrals are only found when sectors with more lines than the integrals themselves are taken into account. In some cases, those higher sectors may not appear in the physical problem at hand. A straight-forward way to find these relations is, of course, to include those higher sectors in the reduction (possibly with a lower seed) and excluding them in the integral selection.

However, if the option `top_level_sectors` is used in the topology definition, the symmetrisation of higher sectors will be skipped. The new option `magic_relations: true` in the topology definition forces `Kira` to include symmetries of sectors which are not subsectors of the top-level sectors in a way that the sectors are preferably mapped onto the top-level sectors. From experience it is sufficient to symmetrise only sectors with up to the same number of lines as the top-level sectors.

An example is provided in `examples/topbox` where the number of master integrals in sector 93 is reduced from 5 to 4 when sector 127 is included in the seed and the option `magic_relations: true` is set.

Note that when top-level sectors are embedded into a topology which do not have a sector number expressible as $2^n - 1$ with integer n, it may happen that such "magic relations" reduce a selected integral to integrals which lie outside of the top-level sectors. In such cases, the number of master integrals may increase, because these additional sectors will then be reduced as well. This can be prevented by using a different sector ordering which regards top-level sectors and their subsectors as simpler than all other sectors. The integral orderings 5–9 as described in section 3.5 exhibit this property and may therefore resolve the issue.

3.14 Merging of database files

`Kira` now provides the job file option `merge` to merge several reduction databases into a single database file.

If a reduction was done independently to different subsets of master integrals with the option `select_masters_reduction`, this can be used to merge the database files with the complementary parts of the reduction into one database with the complete reduction. For an example see section 4.5 and the example `examples/topo7_parallel`. In this example, we perform the reduction to the first half of the master integrals in the project directory `examples/topo7_parallel` and to the second half in the directory `examples/topo7_parallel2`. Then we can merge the database from `topo7_parallel2` into the one from `topo7_parallel` by running the job file

```
jobs:
 - merge:
    files2merge:
     - ../topo7_parallel2/results/kira.db
```

in the directory `examples/topo7_parallel`. Note that the path to the database to merge into is not listed in this file, but implicitly taken from the project directory from which the merge job is executed. Afterwards, the fully reduced integrals can be extracted from the merged database with `kira2math` or `kira2form` as usual.

3.15 Cut propagators

For completeness, we describe the option `cut_propagators` already introduced in `Kira 1.1`. To declare specific propagators of a topology as "cut", set the option

```
cut_propagators: [n1,n2,n3,...]
```

for the topology in `integralfamilies.yaml`, where `[n1,n2,n3,...]` is the list of the numbers of the cut propagators (propagators are numbered starting with 1). Example:

```
integralfamilies:
  - name: "topo7"
    loop_momenta: [k1,k2]
    top_level_sectors: [127]
    propagators:
      - ["-k1", 0]          #1
      - ["k2", 0]           #2
      - ["-k1+k2", 0]       #3
      - ["k1+q2", "m2^2"]   #4
      - ["k2-p2", 0]        #5
      - ["-k1+p1+p2", 0]    #6
      - ["k2-p1-p2", 0]     #7
      - ["k1-p2", 0]        #8
      - ["k2-q2", 0]        #9
    cut_propagators: [3,4]
```

Here, the 3rd and 4th propagator will be treated as cut. This means that during the reduction all integrals in which a cut propagator has non-positive power are set to zero.

4 Best practices

4.1 Topology definition

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. Define the propagators in such a way that

1. propagators have short linear combinations of loop momenta and the shortest linear combinations appear earlier in the list of propagators,

2. analogously, keep linear combinations of external momenta short with ascending length,

3. massless propagators appear earlier in the list.

Always define `top_level_sectors` for the topologies. This will ensure that sectors are only mapped on subsectors of the defined top-level sectors. Also, these are the sectors which are automatically selected when omitted in the seed specification (see section 3.4). Sectors which are not subsectors of the defined top-level sectors will not be symmetrised or mapped. This reduces the amount of time spent on symmetry detection. Note that 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 in order to symmetrise them. See also "magic relations" in section 3.13.

These are only rough guidelines based on experience. Finding the optimal topology representation is a non-trivial task and we do not know a general prescription. If you are dealing with a level of complexity where the topology representation may be crucial to complete the reduction successfully on the available hardware, try different representations and run smaller jobs (i.e. with smaller seeds) and measure the effect.

4.2 Integral selection

Always use the options to select the integrals that should be reduced: `select_mandatory_list` and/or `select_mandatory_recursively` (see e.g. `examples/topo7/jobs2.yaml`).

Recommendation:
Provide a list of all integrals needed for the amplitude and (if applicable) the differential equations and pass it to `select_mandatory_list`. If you intend

to (manually) perform a transformation of the integral basis after the reduction, use the option `select_mandatory_recursively` to select integrals up to a certain number of dots and scalar products that you expect to suffice for the transformation.

When mandatory selectors have been defined, `Kira` prints the list of master integrals after the selection has been performed and before starting the forward elimination. This step is much faster than the full reduction, i.e. one can obtain the list of master integrals in a quick run. It is possible to interrupt the reduction after printing the master integrals by setting the option `run_initiate: true` in the job file and (if present) `run_triangular` and `run_back_substitution` to `false` in order to stop.

In time-consuming reductions, we recommend to always check that this list of master integrals looks sensible before running the full reduction, e.g. that it does not contain unexpected integrals like integrals that lie on the seed edge although there are no masters in the same sector with one dot or one scalar product less.

The reported list of master integrals at this stage contains all integrals which appear in the reduction formulas of mandatory selected integrals and the selected integrals themselves if there was no reduction formula found. Of course, these master integrals can be used to determine integrals which enter the differential equations for the master integrals before doing the actual reduction.

Optional selectors as described in [1] are not available in `Kira 1.2`. They may be reintroduced in a later release.

4.3 Advanced seed selection

With the new seed selector described in section 3.4, one may generate and select a system of equations where the subsectors may have more dots or scalar products than the higher sectors (with more lines).

See `example/topo7/jobs2.yaml`:

```
jobs:
 - reduce_sectors:
    reduce:
     - {topologies: [topo7], sectors: [127], r: 7, s: 2}
     - {topologies: [topo7], sectors: [63], r: 8, s: 3}
    select_integrals:
     select_mandatory_recursively:
      - {topologies: [topo7], sectors: [127], r: 7, s: 2, d: 0}
      - {topologies: [topo7], sectors: [63], r: 8, s: 3, d: 2}
```

Here we demonstrate that we can generate a system of equations for the sector 63 with up to `r=8` and 3 scalar products while the higher sector 127 is reduced up to `r=0` and 2 scalar products. Such a reduction is impossible with the old seed selector notation. Also the job files are shorter and topologies and sectors are chosen automatically according to the topology definition.

See `example/topo7/jobs1.yaml`:

```
jobs:
 - reduce_sectors:
    reduce:
     - {r: 7, s: 2}
```

where `topo7` is implicitly chosen with sector 127 and `topo7x` with sector 508 as defined in the topology definition with `top_level_sectors` set. An equivalent job file with the old seed notation looks like this:

```
jobs:
 - reduce_sectors:
    sector_selection:
     select_recursively:
      - [topo7,127]
      - [topo7x,508]
    identities:
     ibp:
      - {r: [t,7], s: [0,2]}
```

4.4 Resuming aborted reduction jobs

In the following stages of the reduction `Kira` automatically creates checkpoints from which an aborted calculation can be resumed:

- After the search for the symmetries and trivial sectors of every topology.
- After generating the system of equations and selecting a subsystem of linearly independent equations.
- After the forward elimination, i.e. when the system is in triangular form. In case the option `run_triangular: sectorwise` has been set, the state will be saved after every completed sector. Usually, the forward elimination is quite fast. If checkpointing is required anyway, `run_triangular: sectorwise` must be set.

- During the back substitution, intermediate results will be committed to the database regularly.

When an aborted job is restarted, it will automatically continue from the last checkpoint if the option `conditional: true` is set. This is the default unless the option `select_masters_reduction` is used (see section 4.5 for details). Symmetries and trivial sectors will always be reused, independent of `conditional`.

With the following options, different reduction steps may be started individually:

- `run_symmetries: true/false`,
- `run_initiate: true/false`,
- `run_triangular: true/false/sectorwise`,
- `run_back_substitution: true/false`.

Note that the bulk of the CPU time usually goes into the back substitution. However, there is currently no option to terminate a job gracefully after a specified amount of time. Instead the job may just be terminated by the means of the operating system (e.g. by a batch system with a time limit per job).

4.5 Individual reductions to selected master integrals

Since `Kira 1.1` it is possible to split the back substitution into several reduction jobs where each job calculates the coefficients of a given subset of master integrals only, effectively setting all other master integrals to zero. The full reduction is then obtained by merging the results from the individual reductions (i.e. summing the right-hand sides of the reduction formulas for each reduced integral). Similar approaches have been described in [20, 22]. There are two scenarios in which this technique is useful.

- Reduce the memory consumption during the back substitution by calculating coefficients for different master integrals sequentially in a single run. In order to use this feature, create an entry of the following form in your job file:

```
select_integrals:
 select_masters_reduction:
  - [topo7, [1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31]]
  - [topo7, [2,4,6,8,10,12,14,16,18,20,22,24,26,28,30]]
```

In this example, we have 31 master integrals for the example topology `topo7` and the reduction will be performed in two sequential steps, first to the master integrals with odd numbers, then to those with even numbers. The merging into complete reduction formulas will be done automatically in this case. Note that in this mode resuming a reduction job with `conditional: true` is not possible. If resuming is forced nevertheless, it will most likely produce wrong results.

- Parallelise the reduction across different machines by letting each machine handle the coefficients of a subset of the master integrals. In order to do this, some manual work is required. We recommend to run the forward elimination (option `run_triangular: true` or `sectorwise`) on one machine and then create a copy of the working directory for each back substitution job. Then run the back substitution jobs on all machines in parallel.

  In this mode resuming a reduction is possible (for every back substitution job individually). Note that if `select_masters_reduction` is used, the option `conditional: true` must be set explicitly to resume a job (checkpoints are created in any case, see section 4.4).

It lies in the user's responsibility to make sure that the combined sets of master integrals listed in `select_masters_reduction` is complete. This means that the list of master integrals must be determined before performing the back substitution. The fastest way to find the master integrals with `Kira` is to run a job with the option `run_initiate: true` and perform an integral selection with the options `select_mandatory_recursively` and/or `select_mandatory_list`. The list of master integrals will then be written to the file `master` in the directory `results/TOPO` for topology `TOPO`. The numbers of the master integrals in the option `select_masters_reduction` refer to the position in this list (starting with 1).

An interesting observation is that the runtime to calculate the coefficients of different master integrals varies strongly. From experience, the master integrals with the smallest number of lines take longest. Therefore, when the back substitution is parallelised across several machines, those integrals should be evenly distributed among the jobs. An example to illustrate the usage of the option `select_masters_reduction` to parallelise the reduction across several machines is provided in `examples/topo7_parallel`. See `examples/topo7_parallel/readme.txt` for details. When the reduction to all master integrals is complete, the databases can be merged as described in section 3.14.

4.6 Command line arguments

Use `--parallel=n` where `n` is at most the number of physical cores (i.e. not counting hyperthreading logical cores) to run `Kira` on `n` CPU cores in parallel.

In `Kira 1.2` option the `--algebra` has been removed. The option is now activated automatically where deemed useful as described in section 3.3.

4.7 Integral ordering and basis choice

Before a large run, try different integral orderings (e.g. ordering 1 and 2) and measure the performance and memory consumption. Note that different orderings may require different seeds. In particular, in orderings which produce a "dot basis", `rmax` must be chosen large enough so that the seed includes the master integrals in each reduced sector.

Instead or in addition to changing the integral ordering it may also be useful to choose a different basis of master integrals with the option `preferred_masters`. The choice of the basis can strongly impact the runtime, because the complexity of the coefficients depends on the basis.

Acknowledgments

We thank Peter Uwer for his continuing support and discussions. We thank Yang Zhang who inspired the implementation of the algebraic reconstruction for his support. We thank Fabian Lange and Mario Prausa for their comments on the manuscript and for testing the new release. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 647356 (CutLoops). J.U. acknowledges support by the "Phenomenology of Elementary Particle Physics beyond the Standard Model" group at Humboldt-Universität zu Berlin for providing computing resources. P.M. acknowledges support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG. Last but not least we would like to thank all users who helped improving `Kira` by sending us their comments, feature requests and bug reports.

A Obtaining Meson and Ninja

If no sufficiently recent version of `Meson` is available from your distribution's package repository, the latest version can be installed system-wide with

```
pip3 install meson
```

rsp. with

```
pip3 install --user meson
```

in the user's home directory. `Python 3.5` or later is required. The `Ninja` binary can be downladed from `https://ninja-build.org`.

B Specifying a compiler and a non-default install location

If you are using a compiler which is not your system compiler, in general you will also need to compile the dependencies with that compiler. If the dependencies are not installed in a default location, the search paths must be set accordingly.

Set the compiler you want to use (here: `g++-X` for `C++` and `gcc-X` for `C`) and the installation prefix:

```
export CXX=g++-X
export CC=gcc-X
PREFIX=/install/prefix
export LD_LIBRARY_PATH=$PREFIX/lib64:\
  $PREFIX/lib:$LD_LIBRARY_PATH
export CPATH=$PREFIX/include:$CPATH
export PKG_CONFIG_PATH=$PREFIX/lib64/pkgconfig:\
  $PREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
```

To compile `CLN` with this compiler and install it to the location specified by `$PREFIX`, extract the `CLN` source package, change into the `CLN` directory and run

```
./configure --prefix=$PREFIX
make
make install
```

Extract the GiNaC source package, change into the GiNaC directory and run

```
./configure --prefix=$PREFIX
make
make install
```

Extract the yaml-cpp source package, change into the yaml-cpp directory and run

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=$PREFIX -DBUILD_SHARED_LIBS=ON ..
make
make install
```

Change into the Kira directory and run (of course, autotools may be used as well, see section 2.3)

```
meson --prefix=$PREFIX builddir
cd builddir
ninja
ninja install
```

To run Kira, LD_LIBRARY_PATH must be set, so that the same libraries are linked at startup as were linked at compile time. Then run

```
$PREFIX/bin/kira
```

Optionally, you may set the PATH environment variable so that the full path is not required when starting Kira.

```
PATH=$PREFIX/bin:$PATH
kira
```

It may be convenient to set and export LD_LIBRARY_PATH and PATH in your shell configuration (e.g. ~/.bashrc if you are using Bash).

References

[1] P. Maierhöfer, J. Usovitsch and P. Uwer, *Kira—A Feynman integral reduction program*, *Comput. Phys. Commun.* **230** (2018) 99–112, [1705.05610].

[2] S. Laporta, *High precision calculation of multiloop Feynman integrals by difference equations*, *Int.J.Mod.Phys.* **A15** (2000) 5087–5159, [hep-ph/0102033].

[3] K. G. Chetyrkin and F. V. Tkachov, *Integration by Parts: The Algorithm to Calculate beta Functions in 4 Loops*, *Nucl. Phys.* **B192** (1981) 159–204.

[4] T. Gehrmann and E. Remiddi, *Differential equations for two loop four point functions*, *Nucl. Phys.* **B580** (2000) 485–518, [hep-ph/9912329].

[5] A. V. Smirnov and V. A. Smirnov, *Applying Grobner bases to solve reduction problems for Feynman integrals*, *JHEP* **01** (2006) 001, [hep-lat/0509187].

[6] R. N. Lee, *LiteRed 1.4: a powerful tool for reduction of multiloop integrals*, *J. Phys. Conf. Ser.* **523** (2014) 012059, [1310.1145].

[7] K. J. Larsen and Y. Zhang, *Integration-by-parts reductions from unitarity cuts and algebraic geometry*, *Phys. Rev.* **D93** (2016) 041701, [1511.01071].

[8] D. A. Kosower, *Direct Solution of Integration-by-Parts Systems*, *Phys. Rev.* **D98** (2018) 025008, [1804.00131].

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

[10] A. von Manteuffel and C. Studerus, *Reduze 2 - Distributed Feynman Integral Reduction*, 1201.4330.

[11] A. V. Smirnov, *FIRE5: a C++ implementation of Feynman Integral REduction*, *Comput. Phys. Commun.* **189** (2015) 182–191, [1408.2372].

[12] J. Pakkanen, *The Meson Build System*, https://mesonbuild.com.

[13] C. W. Bauer, A. Frink and R. Kreckel, *Introduction to the GiNaC framework for symbolic computation within the C++ programming language*, *J. Symb. Comput.* **33** (2000) 1, [cs/0004015].

[14] J. Vollinga, *GiNaC: Symbolic computation with C++*, *Nucl. Instrum. Meth.* **A559** (2006) 282–284, [hep-ph/0510057].

[15] B. Haible and R. B. Kreckel, *CLN - Class Library for Numbers, version 1.3.4*, http://www.ginac.de/CLN.

[16] *yaml-cpp*, https://github.com/jbeder/yaml-cpp.

[17] J.-L. Gailly and M. Adler, *ZLIB*, http://zlib.net.

[18] R. H. Lewis, *Computer Algebra System Fermat*, http://www.bway.net/lewis.

[19] J. Böhm, A. Georgoudis, K. J. Larsen, M. Schulze and Y. Zhang, *Complete sets of logarithmic vector fields for integration-by-parts identities of Feynman integrals*, *Phys. Rev.* **D98** (2018) 025023, [1712.09737].

[20] J. Böhm, A. Georgoudis, K. J. Larsen, H. Schönemann and Y. Zhang, *Complete integration-by-parts reductions of the non-planar hexagon-box via module intersections*, *JHEP* **09** (2018) 024, [1805.01873].

[21] R. V. Harlander, Y. Kluth and F. Lange, *The two-loop energy–momentum tensor within the gradient-flow formalism*, *Eur. Phys. J.* **C78** (2018) 944, [1808.09837].

[22] H. A. Chawdhry, M. A. Lim and A. Mitov, *Two-loop five-point massless QCD amplitudes within the IBP approach*, 1805.09182.