Kira: Laporta's Algorithm is Still Alive!

Philipp Maierhöfer

Albert-Ludwigs-Universität Freiburg

Workshop "Taming the Complexity of Multiloop Integrals"
ETH Zürich
7 June 2018

In collaboration with Johann Usovitsch and Peter Uwer

Outline

Kira: origins

Kira [Usovitsch, Uwer]

- Laporta algorithm using Gaussian elimination with pivoting based on an integral and equation ordering.
- For the forward elimination to be efficient, it requires the system of equations to be linearly independent.
- Previously relying in ICE [Kant '13], which was too inefficient.

pyRed [PM]

- Originally a framework to explore reduction algorithms in Python, written as a spare time project in 2013
- Use information gained by analysing the system with coefficients mapped to a finite integer field.
- Though promising, the project was put on ice.

Joined forces in 2016

Selection of linearly independent equations

Computational complexity and pivoting

ICE

- Of the equations with the highest weight integral, pick the lowest weight equation.
- System effectively becomes dense $\rightarrow O(n^3)$ scaling $+$ expensive pivoting, high memory consumption.
- “Good” selection, minimising substitutions.

pyRed

- Pick the equation with the lowest weight, insert known solutions (until it is known for what to solve), and solve it.
- Initial (stable) sorting, theoretically $O(n \log^2 n)$, cheap in practice.
- System stays sparse $\rightarrow \sim O(n)$ scaling, no dynamic pivoting.
- Less optimal selection, compensated by reduced time for selection.

pyRed without py-

Reimplemented the modular arithmetic part in C++

- Operates on 64-bit unsigned integers (63-bit primes).
  Optionally uses array coefficients to check for collisions:
  very common with 32-bit integers in real-world applications,
  but haven't seen any with 64-bits.
- Templated: can use other coefficient classes with reasonable effort.
- Selection of a sub-system which is sufficient to solve specific integrals. Naïve approch: doesn't work wonders,
  but may reduce the system size by a factor ~ 2, sometimes more.
- Fast determination of master integrals.
- New: high performance multi-threaded equation generator.

Can be used stand-alone or within other programs
(but has no stable API and documentation, yet).

Kira reduction steps

```
jobs:
- reduce_sectors:
    sector_selection:
      select_recursively:
      - [topo4,127]
    identities:
      ibp:
      - {r: [t,7], s: [0,4]}
    select_integrals:
      select_masters: "basisChange"
      select_mandatory_list:
      - [topo4, needed.dat]
      select_mandatory_recursively:
      - [topo4,127,1,1]
    run_symmetries: true
    run_initiate: true
    run_pyred: true
    run_triangular: true
    run_back_substitution: true
    data_file: false
    conditional: true
```

Mandadory options in a reduction job
(compatible with Reduze [von Manteuffel])

A sample job file for the topology named “topo4”, sector 127
(big endian binary:
“the first seven propagators may have positive powers”),

Sum of positive powers up to 7 and up to 4 irreducible scalar products.

Other options (not all shown)

Kira

`select_masters: "basisChange"` *new in Kira 1.1*
Instead of choosing master integrals automatically,
prefer those from the file "basisChange" (one integral per line).
Incompleteness and overcompleteness are automatically handled.
Note: no a posteriori basis change, yet.

`select_mandatory_list: [[topo4, needed.dat]]`
Subsystem selection for a list of integrals. Recommendation:
request integrals needed for the amplitude and differential equations.

`select_mandatory_recursively: [[topo4,127,1,1]]`
Subsystem for a seed range (max dots and max scalar products).
Use this to have enough freedom for a later basis change.

Note that in order to determine the master integrals before the reduction,
at least one `select_mandatory_*` option must be provided.
Also use this to confirm that the seed was chosen large enough.

Kira

`run_symmetries: true`
Find symmetry relations (invariance of Symanzik polynomials).

`run_initiate: true`
Generate equations.

`run_pyred: true`
Select independent equations rsp. sub-system. Find master integrals.

`run_triangular: true`
Run the forward elimination.

`run_back_substitution: true`
Run the back substitution.

`data_file: false`
Do not write (redundant) human readable reduction tables to disk.

`conditional: true` *new in Kira 1.1*
Enable checkpointing by using database transactions.
Set this option so that aborted runs can be resumed.

Kira

```
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]]
```

new in Kira 1.1

Calculate only the contributions due to specific master integrals (here numbered from 1 to 31). Sets all other coefficients to zero.

This has two applications

- Reduce memory consumption.
- Parallelise the reduction across several machines.

Perform all steps including the forward elimination, then run the back substitution on several machines with different job files (selecting different master integrals). Combine results afterwards.

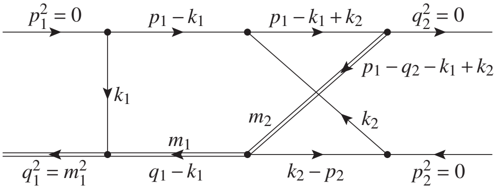
Gives insight how much CPU time goes into which coefficients. Can be used to optimise the the choice of master integrals.

Similar approach described very recently by [Chawdhry, Lim, Mitov].

Kira

```
integralfamilies:
- name: "topo4"
  loop_momenta: [k1,k2]
  top_level_sectors: [127]
  propagators:
  - [ k1,            0 ]
  - [ k2,            0 ]
  - [ p1-k1,         0 ]
  - [ k2-p2,         0 ]
  - [ q1-k1,       m12 ]
  - [ p1-k1+k2,      0 ]
  - [ p1-q2-k1+k2, m22 ]
  - [ k1-q2,         0 ]
  - [ k2-p1-p2,      0 ]
  cut_propagators: [3,5]
```

$\leftarrow$ do not map integrals on sectors which contain “auxiliary” propagators



$\leftarrow$ new in Kira 1.1

works by setting coefficients to zero and disabling cut $\leftrightarrow$ uncut symmetries

Benchmarks I

`topo4` with $m_2^2 = \frac{3}{14} m_1^2$ (no cuts)
on a dual Intel Xeon E5-2680 machine using 11 cores.

Type	s_max	$T_\texttt{pyRed}$	$T_\texttt{Kira}$	$T_\texttt{Reduze}$	$T_\texttt{FIRE}$	$\frac{T_\texttt{pyRed}}{T_\texttt{Kira}}$	$\frac{T_\texttt{Reduze}}{T_\texttt{Kira}}$	$\frac{T_\texttt{FIRE}}{T_\texttt{Kira}}$
default	1	2.8 s	90 s	2.1 h	23 m	0.03	86	15
select	1	2.8 s	24 s	–	19 m	0.11	–	49
default	2	9.8 s	6.6 m	7.2 h	2.3 h	0.02	65	21
select	2	11 s	167 s	–	2.2 h	0.07	–	47
default	3	28 s	43 m	22.8 h	7.6 h	0.01	32	11
select	3	30 s	539 s	–	7.4 h	0.06	–	49
default	4	67 s	2.4 h	2.7 d	23.5 h	0.01	26	10
select	4	70 s	35 m	–	22.4 h	0.03	–	38

"default" jobs are without integral selection,
"select" jobs select `[topo4,127,0,`s_max`]`.

Benchmarks II

2-loop pp $\rightarrow$ Hj with full mass dependence on m_t and m_H

- "4-scale problem": s, t, m_t, m_H (and d).
- 11 topologies, of which 3 are non-planar, embedded in 3 integral families.
- Reduced all integrals required for the amplitude and the differential equations (and enough for basis transformtions).
- Used topology definitions by the Zürich group, no further tuning.
- Finished after 3 weeks in total for all topologies on a dual Intel Xeon E5-2630 v4 (20 cores).

Several groups did not succeed with just `Reduze` and `FIRE`.
(for then non-planars, UZH spent > 1 year unsuccessfully on a dual Intel Xeon E5-2697v2, 24 cores)

Tips & tricks

Some tricks may speed up the reduction significantly

(note that the optimal settings may not be the same across different reduction programs)

- Always use integral selection (`select_mandatory_*`).
- Always start multi-scale problems with the option `--algebra` (not yet automatic). Combines coefficients pairwise with `Fermat` [Lewis] instead of all at once.
- Try out `select_masters_reduction` for really big problems.
- Make use of `conditional: true` (not yet automatic).
- Play with the propagator definition/ordering and see how it affects the performance (in a smaller run). Rule of thumb: short propagators and top sector of the form $111\ldots000$.
- Do the same with different choices of master integrals. Tip: look at the tadpoles.

Kira

Current release of Kira: version 1.1 from 26.04.2018.

Expect the next release until LoopFest 2018 (mid of July).

Things to come (may not all make it into the next release)

- New and much faster equation generator (relevant for large systems).
- Parallel `Fermat` handlers ported from `OpenMP` to `C++` threads (relevant when many rather simple coefficients must be processed).
- More integral orderings (e.g. prefering masters with dots over sps).
- Solve a system of user-provided equations (already possible with some hacking [Prausa]).
- User provided "equation templates": equations with symbolic powers; let Kira insert seeds, (e.g. IBP generating vectors).

Probably later: map an amplitude to topologies / generate differential equations / arbitrary crossings.

Not coming soon: reconstruction of rational functions from finite integers [Peraro; v. Manteuffel, Schabinger].

Conclusions

- Laporta's algorithm is probably not the final solution to the reduction problem. But reports of its death are exaggerated.
- Kira implements a straight forward Gaussian elimination, assisted by modular arithmetic to select equations.
- Let the benchmarks speak for itself. Main focus on problems with several scales. For multi-loop, limits are hit (for now) when the system has billions of equations.
- Make use of all options to tweak the reduction. Keep in mind that the performance often depends on details (topology definition, interplay with integral ordering, . . .).

Laporta's algorithm just works – with little human effort.

Kira is relatively new and may profit from your feedback.
In case of questions, feature requests and bug reports, please contact us.

There's more to come. Stay tuned!