

Tools for NLO and Higher Order Corrections to Scattering Processes

Philipp Maierhöfer

Albert-Ludwigs-Universität Freiburg

Presentation and Interview for a Group Leader Position

Max Planck Institut für Physik

München

26 January 2021

Outline

1 Research Activity

- (Not Just) NLO Automation
- Feynman Integral Reduction

2 Software and Computing Achievements and Skills

- Software Development
- IT Skills and Duties

Part I

PART I

Research Activity

Major long-term projects

OpenLoops

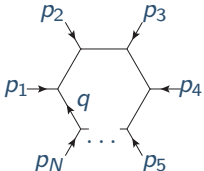
- **Automatic generator for tree and 1-loop scattering amplitudes:** High-performance evaluation of matrix elements up to high multiplicities in the full Standard Model.
- Who needs something like that? Everybody who runs Monte Carlo simulations e.g. for the LHC at next-to-leading or higher order.

Kira

- **Reduction program for multi-loop Feynman integrals:** expresses integrals of a given family as linear combinations of a small set of master integrals.
- Who needs something like that? Everybody who calculates amplitudes (or renormalisation constants, form factors, ...) at 2-loops or higher, or differential equations for master integrals.

OpenLoops Algorithm

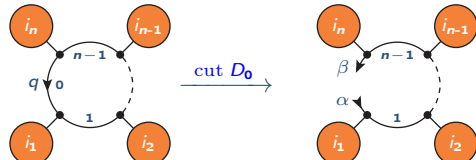
Loop amplitude: **colour factors**, **tensor coefficients**, and **tensor integrals**



$$= \mathcal{C} \sum_{r=0}^R \mathcal{N}_r^{\mu_1 \dots \mu_r} \cdot \underbrace{\int d^d q \frac{q_{\mu_1} \dots q_{\mu_r}}{D_0 D_1 \dots D_{N-1}}}_{\text{tensor integral of rank } r}$$

$$D_i = \left(q + \sum_{\ell=1}^i p_\ell \right)^2 - m_i^2$$

$\mathcal{N}(q) = \sum_r \mathcal{N}_r^{\mu_1 \dots \mu_r} q_{\mu_1} \dots q_{\mu_r}$, cut a loop propagator: $\mathcal{N}(q) = \mathcal{N}_\beta^\alpha(q)$:

$$\int d^d q \frac{\mathcal{N}(q)}{D_0 \dots D_{N-1}} =$$


OpenLoops algorithm: [Cascioli, PM, Possorini] generalises tree-level recursion formulas to a numerical recursion for the **tensor coefficients** (avoid algebra!).

Reduction

Construct processes from **generic building blocks** (only depend on model).

Treatment of tensor integrals: choose between

- Tensor integral reduction with Collier [Denner, Dittmaier, Hofer],
- Integrand reduction with CutTools [Ossola, Papadopoulos, Pittau].

OpenLoops was the first general purpose 1-loop generator that **reached and surpassed the performance of optimised algebraic code**.

Algorithm adopted by MadLoop [Hirschi], similar alg. in Recola [Actis et al.].

Latest development: OpenLoops on-the-fly reduction

$$q^\mu q^\nu = \sum_i \left(A_i^{\mu\nu} + B_{i,\lambda}^{\mu\nu} q^\lambda \right) D_i(q) + \text{terms that integrate to zero.}$$

Apply reduction during the recursion whenever the tensor rank reach 2.

Implemented in OpenLoops 2 [Buccioni, Lang, Lindert, PM, Pozzorini, Zhang, Zoller]

OpenLoops 2 delivers **unprecedented performance and numerical stability**.

Applications

OpenLoops has been interfaced with many Monte Carlo generators: GENEVA/BONSAY/HERWIG/MATRIX/MUNICH/POWHEG-BOX/SHERPA/WHIZARD.

Applications to NLO QCD, NLO EW & loop-induced process, often to demonstrate new capabilities together with Monte Carlo developers.

Among various others that I contributed to:

- NLO QCD $pp \rightarrow t\bar{t} + \leq 3j$: up to 150,000 diagrams per channel, multi-jet merging [Höche et al.] or MiNLO [Hamilton, Nason, Zanderighi].
- NLO EW WZ scattering: $pp \rightarrow \mu^+ \mu^- e^+ \nu_e jj$ [Denner et al.]: 8-point functions, up to 83,000 diagrams per channel, entire gauge sector of the SM, complex mass scheme necessary, more important than QCD corrections.

OpenLoops is extensively used by other users, in particular ATLAS & CMS.

Most NNLO calculations use OpenLoops for real-virtual corrections (stability!), e.g. in MATRIX [Grazzini et al.]. Probably the only provider that can do $2 \rightarrow 4$ real-virtual, e.g. NNLO 3γ [Chawdhry et al.].

Feynman integral reduction

≥ 2 loops: Calculation of Feynman integrals requires case-by-case treatment.
Precision demanded by experiments → see Les Houches NNLO wishlist.

Strategy:

- Express all integrals in terms of master integrals (“reduction”).
- Calculate the master integrals, usually with differential eqs.

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

Only general algorithm for integral reduction: Laporta’s algorithm.

- Integration-by-parts identities relate integrals with different a_i .
- Systematically generate equations with integer values for the a_i .
- Use Gaussian elimination to solve the system.

Public implementations: Reduze [v. Manteuffel], FIRE [A. Smirnov], Kira.

Problem: Huge systems and complicated coefficients (multivariate rational functions) make the procedure inefficient.

Kira

Our solution: Kira [PM, Usovitsch]

- Most actively developed reduction program.
- Most reliable symmetry finder for Feynman integrals on the market.
- Use modular arithmetic to analyse the system.
- Exploit freedom in Gaussian elimination to increase efficiency.
- Sophisticated treatment of coefficients for faster simplification.
- In many cases 1-2 orders of magnitude faster than other programs.

Still significant improvements necessary to tackle 2 → 3 NNLO.

Most promising idea: generation of block-diagonal systems [Guan et al.].

Kira 2 [Klappert, Lange, PM, Usovitsch], interfaced with FireFly [Klappert, Lange]

- Reconstruct rational numbers from their image on a finite field.
- Interpolation of multivariate rational functions.
- Parallelised using MPI to run on HPC clusters.

Part II

PART II

Software and Computing Achievements and Skills

OpenLoops: technical realisation

- Process generator in **Mathematica** (FeynArts [Hahn] for diagrams). Fills file templates to generate process-specific Fortran code.
- Process-independent Fortran library for numerics/tensor handling.
- No hard complexity limits. In practice up to $2 \rightarrow 6$ scattering with reasonable resources. Numerics part could do more!
- Dynamic library loader (via C bindings) loads processes at runtime.

Design decisions reasonable at the time ...

- **Mathematica**: handle algebra, **FeynArts**: already available, **Fortran**: good for number crunching & physicists know it.
- **Code generation** easier than doing everything in-memory.

... and the success proves us right:

- NLO QCD working after just 1.5 years of development.
- 1-2 orders of magnitude faster than other generators at the time.

But . . .

The implementation served us well for 10 years now, but

- Working with the code **more difficult with each new feature**.
Lots of workarounds, e.g. for compatibility.
- New physics **model definition** possible, but **quite involved**.

Example for unwanted complexity:

CutTools originally needed to circumvent limitations of Collier.
Requires **quadruple precision** to rescue numerically unstable points.

Problem: **Fortran lacks generic programming** capabilities
(Fortran 2003 “parametrised derived types” took until 2018 in GCC).

Use a preprocessor to **generate precision-specific code** and interfaces.
Handled within the SCons/Python build system.

OpenLoops: The Next Generator (TNG)

Use a decade of experience and write a new generator!

- Completely in C++.
- Modular design for maintainability and extensibility.
- Fully model-agnostic process generator. Use UFO/NLO-CT models from FeynRules [Alloul et al.] or Rept1L [Lang].
- Recursive in-memory generation of an analytic amplitude representation → can be manipulated, e.g. with diagram filters.
- Hierarchical structure based on momentum/particle type/coupling order/fermionic signs/loop integral/colour structure/helicity cfg. Translated into a sequence of calls to numerical routines.
- Custom memory management for numerical objects on a memory pool. Save allocation time & reduce cache misses (large effect!).

Implementation in progress (currently only by myself):

Tree-level working, including UFO model import (with help of M. Prausa).

Kira/pyRed

Originated from merging two projects with complementary strengths: original Kira [Usovitsch] and pyRed [PM], (both unpublished at the time).

pyRed

- Originally a framework in **Python** to explore reduction algorithms.
- Ported relevant parts to **C++** (→ performance/memory usage).
- Provides the **modular arithmetic** functionality of Kira.
- **High-performance** generator for integration-by-parts/symmetry relations and **Gaussian solver for sparse systems**. Exploits the properties of IBP systems to reach almost $\mathcal{O}(n)$ computational complexity (with coefficients mapped to the finite field).
- **Multi-threaded** using C++11 threads.

Eliminates **redundancy** and chooses subsystems to **solve selected integrals**.

Heavily used to calculate the probes required by FireFly.

High performance computing

High performance computing

- Member of the NEMO technical advisory board since 2017: Lobbying for the needs in theoretical particle physics, (hard- and software requirements/preferences, fair scheduling). Currently discussing options for NEMO 2.
- Experience of HPC cluster usage with different batch systems/schedulers (MOAB/SLURM/SGE).
- Optimisation of software to run on large clusters (e.g. solve problems with file system access latencies).

Bits and pieces

- De-facto admin of hardware that is not on the radar of the IT group: Mac Mini with the SmartBoard, institute Laptops, Wacom tablets.
- Presentation equipment (e.g. technical coordinator for HP² 2018).

Duties that come with software projects

- Software support: Tutoring/tipps, fixes, help with compilation problems (usually problems with the build environment or dependencies), support for MacOS.
- Following development of programming languages (mostly C++ standard). If only people would adopt new compilers earlier ...

Driving force to introduce tools that make the developer's life easier:
Version control, issue tracker, better build systems, code style, ...

- Linux server administration (rented VPS and self-hosted); Intel AMT; basic monitoring,
- Following IT news (regularly at least Heise/Golem/Phoronix).

Better make things work for everybody than just for yourself.

Both in own projects and in others (did you notice when CLN/Ginac or FLint were broken on openSUSE? Or libssh jump hosts?).

Conclusions

- Programming experience in several languages for different purposes.
- Maintaining and developing software that's widely used in the community.
- 1-loop matrix element generators are as important as ever. Focus on NNLO requirements and BSM.
- Feynman integral reduction: Essential tool and one of the bottlenecks in higher order calculation. Improve algorithms, e.g. using finite field techniques to make more $2 \rightarrow 3$ NNLO possible.
- In-house applications are useful to demonstrate capabilities, motivate further development & collaboration. But the tools have priority.
- Experience with system administration and IT support.