Using OpenLoops for one-loop scattering amplitudes

Philipp Maierhöfer

Physik-Institut
Universität Zürich

HP2: High Precision for Hard Processes

Firenze, 3 September 2014

In collaboration with
F. Cascioli, J. Lindert, and S. Pozzorini

Outline

OpenLoops

NLO automation

- Feasibility of $2 \to 4$ NLO QCD corrections is well established.
- Move from fixed order and proof of concept calculations to full simulations for experimental analyses.
- Develop tools of general applicability with focus on generic features rather than individual processes.
- Performance is crucial when NLO should become the default accuracy for LHC analyses.

With the right tools, the users can spend more time doing physics instead of solving technical problems.

Many more or less generic tools have been developed

Collier, CutTools, OneLOop, Samurai;
BlackHat, FormCalc, GoSam, HELAC-NLO, MadLoop, MCFM, NJet, OpenLoops, Recola, VBFNLO;
Herwig++, MadGraph/aMC@NLO, POWHEG, Pythia, Sherpa

Different views on loop amplitudes

$$\mathcal{A} = \int d^d q \, \frac{\sum_{r=0}^{R} \mathcal{N}_r^{\mu_1 \dots \mu_r} \cdot q_{\mu_1} \dots q_{\mu_r}}{D_0 \; D_1 \; \dots \; D_{N-1}}$$

Covariant decomposition + tensor integrals

$$\int d^d q \, \frac{q_{\mu_1} \dots q_{\mu_r}}{D_0 \dots D_{N-1}} \equiv P^{k,r}_{q_{\mu_1} \dots q_{\mu_r}} \, T^{k,r} \quad \Rightarrow \quad \mathcal{A} = \sum_r N^{k,r} \cdot T^{k,r}$$

with monomials $P^{k,r}$, built from the metric and external momenta.
Calculate $N^{k,r} = \mathcal{N}_r^{\mu_1 \dots \mu_r} P^{k,r}_{q_{\mu_1} \dots q_{\mu_r}}$ analytically in d dimensions.

Limited by huge expressions and expensive algebraic simplifications.

Tree recursion + OPP reduction [Ossola, Papadopoulos, Pittau]

$$\mathcal{A} = \int d^d q \, \frac{N(q)}{D_0 \; D_1 \; \dots \; D_{N-1}}$$

Calculate the numerator N for fixed q which satisfy on-shell conditions.
Original idea: do this with recursive generators for tree amplitudes.

Multiple evaluations for different q limit the performance.

The OpenLoops perspective

$$\mathcal{A} = \sum_{r=0}^{R} \mathcal{N}_r^{\mu_1 \dots \mu_r} \cdot \int d^d q \, \frac{q_{\mu_1} \dots q_{\mu_r}}{D_0 \, D_1 \, \dots \, D_{N-1}}$$

Use a numerical recursion for the tensor components of $\mathcal{N}_r^{\mu_1 \dots \mu_r}$, which encode the loop momentum dependence of the numerator.

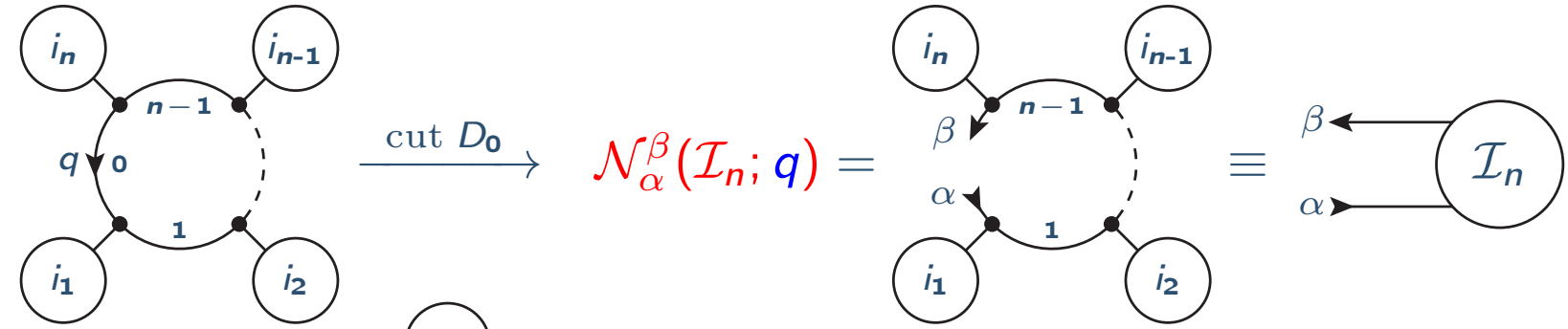
Inspired by van Hameren's ['09] work on multi-gluon amplitudes, where a Dyson-Schwinger-like recursion is used.

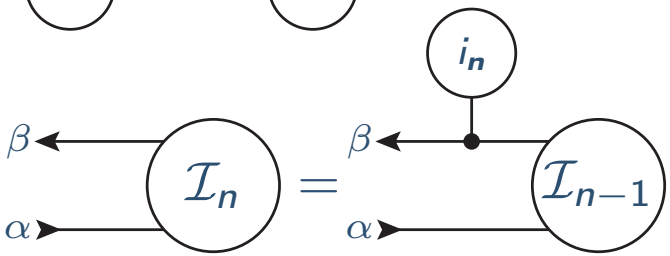
Naturally works with both, tensor integral reduction
[Melrose; Passarino, Veltman; Denner, Dittmaier; Binoth et al.]
and OPP reduction via $N(q) = \mathcal{N}_r^{\mu_1 \dots \mu_r} q_{\mu_1} \dots q_{\mu_r}$.
$\rightarrow$ 1-2 orders of magnitude improvement wrt. tree recursion.

Introduced in the context of OpenLoops:
perform colour and helicity summation before reduction.

With OpenLoops
OPP is reduction almost as efficient as tensor integrals.

A one-loop diagram is an ordered set of trees i_k with wave functions $w^\delta(i_k)$, connected along the loop by vertices $X^\beta_{\gamma\delta}$.





$$\mathcal{N}^\beta_\alpha(\mathcal{I}_n;q) = X^\beta_{\gamma\delta}(q)\,\mathcal{N}^\gamma_\alpha(\mathcal{I}_{n-1};q)\,w^\delta(i_n)$$

with the loop momentum q separated from the coefficients

$$\mathcal{N}^\beta_\alpha(\mathcal{I}_n;q) = \sum_{r=0}^{n} \mathcal{N}^\beta_{\mu_1\ldots\mu_r;\alpha}(\mathcal{I}_n)\,q^{\mu_1}\ldots q^{\mu_r}, \qquad X^\beta_{\gamma\delta} = Y^\beta_{\gamma\delta} + q^\nu Z^\beta_{\nu;\gamma\delta}$$

Leads to the recursion formula for "open loops" polynomials $\mathcal{N}^\beta_{\mu_1\ldots\mu_r;\alpha}$:

$$\mathcal{N}^\beta_{\mu_1\ldots\mu_r;\alpha}(\mathcal{I}_n) = \left[Y^\beta_{\gamma\delta}\,\mathcal{N}^\gamma_{\mu_1\ldots\mu_r;\alpha}(\mathcal{I}_{n-1}) + Z^\beta_{\mu_1;\gamma\delta}\,\mathcal{N}^\gamma_{\mu_2\ldots\mu_r;\alpha}(\mathcal{I}_{n-1})\right] w^\delta(i_n)$$

OpenLoops: technical setup

- FeynArts [Hahn] to generate Feynman diagrams
- Mathematica to generate process specific Fortran code
- Process independent Fortran library
- Rational terms of type R_2 from couterterm-like Feynman rules
  [Draggiotis, Garzelli, Malamos, Papadopoulos, Pittau '09, '10; Shao, Zhang, Chao '11]
- QCD corrections to Standard Model processes,
  EW corrections in preparation

Default reduction library: `CutTools` [Ossola, Papadopoulos, Pittau],
with scalar integrals from `OneLOop` [van Hameren].
Requires quad precision to treat numerical instabilities.

Thanks to R. Pittau for permission to distribute `CutTools` under GPL.

Alternatively, interfaces are available for

- OPP reduction with `Samurai` [Mastrolia, Ossola, Reiter, Tramontano]
- Tensor integral reduction with `Collier` [Denner, Dittmaier, Hofer]
  (unpublished). Numerically more stable in double precision than
  OPP reduction thanks to Denner-Dittmaier reduction techniques.

Performance

process	diags	size/MB	time/ms
$u\bar{u} \to t\bar{t}$	11	0.1	0.27(0.16)
$u\bar{u} \to W^+W^-$	12	0.1	0.14
$u\bar{d} \to W^+g$	11	0.1	0.24
$u\bar{u} \to Zg$	34		0.75
$gg \to t\bar{t}$	44	0.2	1.6(0.7)
$u\bar{u} \to t\bar{t}g$	114	0.4	4.8(2.4)
$u\bar{u} \to W^+W^-g$	198	0.4	3.4
$u\bar{d} \to W^+gg$	144	0.5	4.0
$u\bar{u} \to Zgg$	408		17
$gg \to t\bar{t}g$	585	1.2	40(14)
$u\bar{u} \to t\bar{t}gg$	1507	3.6	134(101)
$u\bar{u} \to W^+W^-gg$	2129	2.5	89
$u\bar{d} \to W^+ggg$	1935	4.2	120
$u\bar{u} \to Zggg$	5274		524
$gg \to t\bar{t}gg$	8739	16	1460(530)

Measured on an i7-3770K (single thread) with gfortran 4.8 -O0, dynamic (ifort static $\sim$30% faster), tensor integral reduction with Collier.

Colour and helicity summed.

W/Z production includes leptonic decays and non-resonant contributions.

$t\bar{t}$ production numbers in brackets are for massless decays.

CutTools provides similar performance for $2 \to 4$, but is slower for lower multiplicities. In complicated processes, the quad precision evaluations can affect the average runtime significantly.

Applications

NLO: focus on applicability to experiments, "beyond fixed order"

- MEPS@NLO for $\ell\ell\nu\nu + 0, 1\,\text{jets}$ [Cascioli, Höche, Krauss, PM, Pozzorini, Siegert]
- LO merging for (loop-induced) $HH + 0, 1\,\text{jets}$ [PM, Papaefstathiou]
- NLO $W^+W^-b\bar{b}$ with $m_b > 0$ [Cascioli, Kallweit, PM, Pozzorini]
- MEPS@NLO $W^+W^-W^\pm + 0, 1\,\text{jets}$ [Höche, Krauss, Pozzorini, Schönherr, Thompson, Zapp]
- MC@NLO for $t\bar{t}b\bar{b}$ with $m_b > 0$ [Cascioli, PM, Moretti, Pozzorini, Siegert]
- MEPS@NLO for $t\bar{t} + 0, 1, 2\,\text{j}$ [Höche, Krauss, PM, Pozzorini, Schönherr, Siegert]

NNLO: for real-virtual corrections, partly for (1-loop)^2

- $Z\gamma$ [Grazzini, Kallweit, Rathlev, Torre]
- ZZ [Cascioli, Gehrmann, Grazzini, Kallweit, PM, von Manteuffel, Pozzorini, Rathlev, Tancredi, Weihs]
- W^+W^- ($\rightarrow$ talk by D. Rathlev) [Gehrmann, Grazzini, Kallweit, PM, von Manteuffel, Pozzorini, Rathlev, Tancredi]
- $t\bar{t}$ ($\rightarrow$ talk by G. Abelof) [Abelof, Gehrmann-De Ridder, PM, Pozzorini]

Installation

OpenLoops is available for download from hepforge

`http://openloops.hepforge.org`

Requirements: `gfortran 4.6` or later, `Python 2.x` ($x \geq 4$)

The easiest way to install and keep your installation up to date is to pull a copy from the subversion repository

```
svn checkout \
  http://openloops.hepforge.org/svn/OpenLoops/branches/public \
  OpenLoops
```

compile

```
cd OpenLoops
./scons
```

go to the website and look up the processes which are available for download and install the ones you need, e.g. for Z + up to three jets:

```
./scons auto=ppzj,ppzjj,ppzjjj
```

Use OpenLoops in your own programs

OpenLoops can be used via

- a Binoth Les Houches Accord (BLHA) interface
- its native interface in `Fortran` and `C`

If you want to interface your Monte Carlo tool with OpenLoops and you do not (yet) support the BLHA, the native interface tries to make things as easy as possible for you.

Mapping of equivalent partonic channels is done internally.

Some examples are included in the "examples" directory in the OpenLoops installation folder.

Interface, features and options are documented on the web page.

demonstration: installation and native interface

Interfaces to Monte Carlo programs

OpenLoops provides the matrix elements

Tree level for Born and real corrections, including spin and colour correlations, 1-loop and (1-loop)^2. For NLO simulations these must be supplemented by phase space integration and a subtraction procedure, possibly a parton shower, multi-jet merging, hadronisation, ...

- **Sherpa** ($\rightarrow$ talk by F. Krauss)
  Interface included in the official release 2.1.0 (or later).
  Full-featured event generator, `Sherpa+OpenLoops` is well tested and used in several published simulations.
- **Herwig++** ($\rightarrow$ talk by S. Plätzer)
  BLHA interface, coming soon.
- Monte Carlo by S. Kallweit (to be published). Parton level, efficient integration, particularly used in NNLO calculations.

demonstration: Sherpa example

Sherpa example

Setup for the Sherpa example

- `Rivet 2.1.2` (includes `FastJet`, `HepMC`, `YODA`)
- `Sherpa 2.1.0`, configured with the option
  ```
  ./configure --enable-openloops=<OL_PREFIX> \
      [other options]
  ```
  (`OL_PREFIX` can be modified in the Sherpa run card)
- Fixed order NLO QCD corrections to $pp \to W^-(\to e^-\bar{\nu}_e)j$.
- Set `Loop_Generator=OpenLoops` in the run card.
- Basic `Rivet` analyses to create distributions for $m_{T,\ell}$, $p_{T,\ell}$, $p_{T,j}$.

Use minimal integrator optimisation and 10^5 weighted events.

Some run cards for non-trivial simulations, including MEPS@NLO merging, are available on our web page. More to come!

Conclusions, part 1

In summary, there are basicly two use cases

- Obviously, people who want to do simulations for LHC experiments. Sherpa+OpenLoops uses standard Sherpa runcards, no special OpenLoops knowledge is required.
- People who need 1-loop matrix elements in their own calculations. Particularly useful for NNLO calculations where the real-virtual matrix elements suffer from severe numerical instabilities in soft and collinear regions. Solutions include a good choice of technical parameters, usage of several reduction libraries and if needed rescue systems which are tailored to the problem. We can help!

Processes are available for installation using the integrated downloader

- Many more will be added very soon.
- The process generator will be included in a later release.

Conclusions

OpenLoops is a fast and flexible generator for tree and 1-loop matrix elements.

- Recursive construction of loop momentum polynomials.
- Uses `CutTools` and `OneLOop`. Numerical instabilities are detected and cured with quadruple precision.
  An interface to the tensor integral reduction library `Collier` is available. `Collier` itself will be included as soon as it is public.
- Easy to interface with your own programs.
- Interfaced to `Sherpa` for full automation of NLO simulations, including parton shower, multi-jet merging, ...

Publicly available from http://openloops.hepforge.org

Please send your feedback, questions, suggestions, ...,
to `openloops@projects.hepforge.org`