OpenLoops: The Next Generator

Philipp Maierhöfer

Albert-Ludwigs-Universität Freiburg

Loops and Legs in Quantum Field Theory

St. Goar

3 May 2018

In collaboration with Mario Prausa

Outline

Demands on a matrix element generator

model agnosic
- QCD & EW Standard Model
- BSM and effective theories: import new models

usability
- Process generator included: fast and at runtime (no code generation)
- Easy interfacing

NLO matrix element generator

default applications: performance
- NLO high multiplicity
- Loop-induced
- NNLO (1-loop)^2
- Tree and loop correlators
- Real corrections

applications with increased numerical stability requirements
- NNLO Real-virtual corrections
- Balance performance vs. numerical stability: different reduction techniques, multi-precision (error estimates)

OpenLoops 1/2 [Cascioli, Buccioni, Lindert, PM, Pozzorini, Zoller]

OpenLoops prooved its value in many highly non-trivial applications.

At the time when the OpenLoops development started (end of 2010), the focus was on producing a working tool as quickly as possible.

With success: after $\sim$ 1.5 years, NLO QCD was working.

- Rely on an existing tool for diagram generation: FeynArts [Hahn].
- Use Mathematica to generate process specific code.
  Most program logic is in the generator.
- Fortran for numerics: many physicists know and love it.
- Publish code for processes (generated and validated by us) separatley from the generic library.

This comes with a couple problems from which we are suffering now.

$\rightarrow$ Learn from several years of real-world experience.

Technical issues

Need to keep a stable interface between processes and generic library, otherwise we need to re-generate all processes and maintain several repositories with different versions (will happen for the first time with `OpenLoops 2`).

The code style deteriorated with the implemented hacks to circumvent this problem and retain compatibility.

For state-of-the-art applications, `FeynArts` is too slow and too memory hungry. Even worse, in long runs `Mathematica` tends to die randomly.

Difficult to replace the old code generator. Would require us to program towards the peculiarities of the process interface.

Integration of new physics models more difficult than anticipated in the beginning. Require two models: `FeynArts` and `OpenLoops`; inconvenient parameter handling.

Solution: It's time to rewrite `OpenLoops` from scratch.

Design goals

- In-memory High-level abstract amplitude representation (possibility for humans to understand and manipulate the representation).
- High performance numerics (use custum memory management).
- Support different algorithms, reduction & integral libraries (cf. OL1).
- Multi-precision: arbitrary (number of) floating point data types, specified at compile time, without explicit code duplication.
  $\rightarrow$ Require a language with good generic programming capabilites.

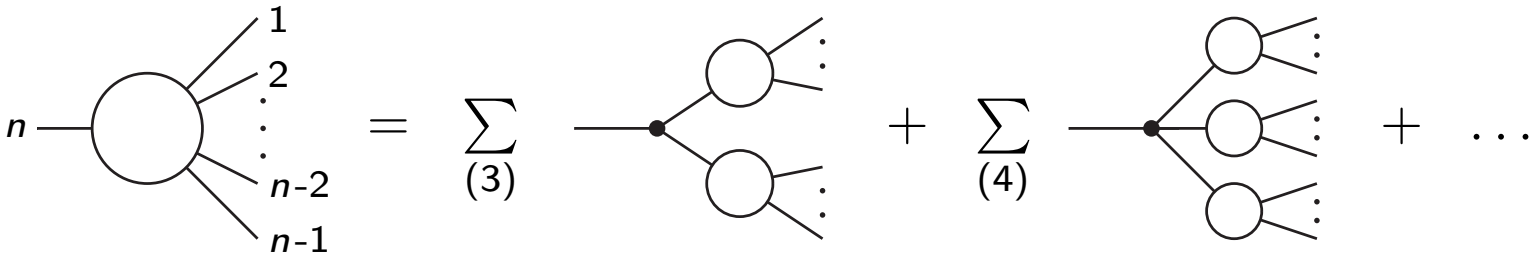
A future-proof platform – and a playground to explore e.g.

- Multi-threading, requires thread-safe reduction/integral libraries.
- Explicit vectorisation (SIMD).
- GPU computing? (expect problems with memory bandwidth).
- Extension to 2-loops.

Language of choice: C++

Topology and diagram generation: tree-level

First, we need a diagram generator. The easiest way is to generate topologies by recursive partitioning of external legs.



The sums run over all partitions of $\{1, 2, \ldots, n\text{-}1\}$ into $2\,(3, \ldots)$ non-empty sets for $3\,(4, \ldots)$-point vertices.

Assign particles according to the Feynman rules, starting from the external particles $\{1, 2, \ldots, n\text{-}1\}$; assign colour and helicity states.

Generate serialised numeric calls:
No need to traverse the tree during evaluation.

This looks a lot like Dyson-Schwinger recursion,
but it is actually not decided here if we use diagrams or currents.

Different ways to treat colour

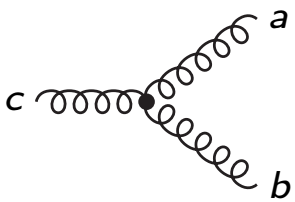
Diagrammatic: Factorise colour from the amplitude, i.e. do not perform colour reduction until the diagram has been built completely.

Common colour basis choices

- Trace basis: products of $T^a_{ij} T^b_{jk} \dots$ and traces thereof. Natural choice for quark lines. "Standard" Feynman rules.
- Colour flow basis: adjoint indices $\rightarrow$ pairs of fundamental indices, $a = \binom{i_a}{j_a}$. Basis elements $\delta^{i_a}_{j_x} \delta^{i_b}_{j_a} \dots$, colour ordered Feynman rules.

All choices with fixed basis produce more terms than necessary

- Diagrams: usually more colour structures than basis elements.
- Feynman rules in colour flow basis:
  colour structures split up, loosing symmetry properties.



$$\propto f^{abc} \propto \underbrace{\mathrm{Tr}(T^a T^b T^c) - \mathrm{Tr}(T^c T^b T^a)}_{\text{trace basis}} \propto \underbrace{\delta^{i_a}_{j_b} \delta^{i_b}_{j_c} \delta^{i_c}_{j_a} - \delta^{i_a}_{j_c} \delta^{i_b}_{j_a} \delta^{i_c}_{j_b}}_{\text{colour flow basis}}$$

Colour treatment in TNG

Solution in TNG: do not commit to a particular colour basis

- Possibility to decide locally which basis to use for each "current".
- Requires extra numerical steps: building colour linear combinations.
- Minimise the number of colour stripped objects to calculate.

Decompose colour factors $\mathcal{C}_i$ into colour basis $\mathcal{B}_j$, $\mathcal{C}_i = \sum_j \mathcal{B}_j \alpha_{ji}$.

$\mathcal{C}_1$  $+ \mathcal{C}_2$  $+ \ldots = \mathcal{B}_1 \big(\alpha_{11}$  $+ \alpha_{12}$  $+ \ldots \big)$

$+ \mathcal{B}_2(\ldots) + \ \ldots \ + \mathcal{B}_{k>k_0} \cdot 0.$

Choose $\mathcal{B}_j$ such that as many as possible appear with coefficient zero.

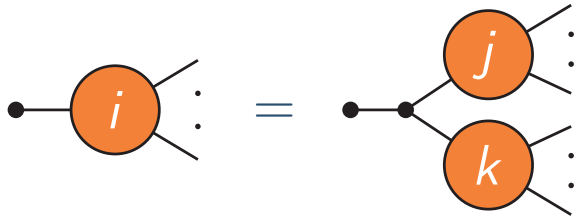
The colour reduction library is exchangeable: just need to provide the same interface (multiplication, contraction of indices, extract coefficients of linear combinations of basis elements).

Interfaces to particular program components provide the possibility to plug in better solutions (better bases, new gauge groups) later.

Tree numerics

Numerical evaluation of tree diagrams: array of callable objects

- External wave functions.
  For now: Dirac fermions, vector bosons, scalars.
- Vertices & propagators: combine colour stripped tree wave functions with vertices from "standard" Feynman rules



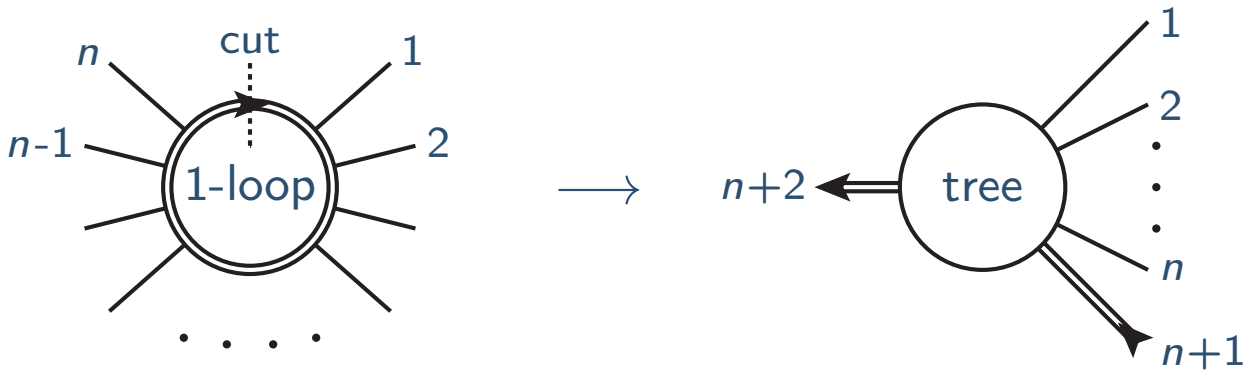
$$w^\beta(i) = \frac{X^\beta_{\gamma\delta}}{p_i^2 - m_i^2}\, w^\gamma(j)\, w^\delta(k)$$

  Evaluation follows the steps of the n-1 $\to$ 1 diagram generator.
  (cf. OL1: build tree diagrams as $n \to 0$ "outside to inside")
- Build linear combinations to decompose wave functions into a local colour basis $\mathcal{B}_i$. Contribution to $\mathcal{B}_i$:

- Interference: $\sum_{i,j} \left(\bullet\!\!-\!\!\boxed{b_i}\!\!: \right)^* \mathcal{K}_{ij} \left(\bullet\!\!-\!\!\boxed{b_j}\!\!: \right)$, $\mathcal{K}_{ij} = \sum_{\text{colour}} \mathcal{B}_i^\dagger \mathcal{B}_j$

Loop diagram generation

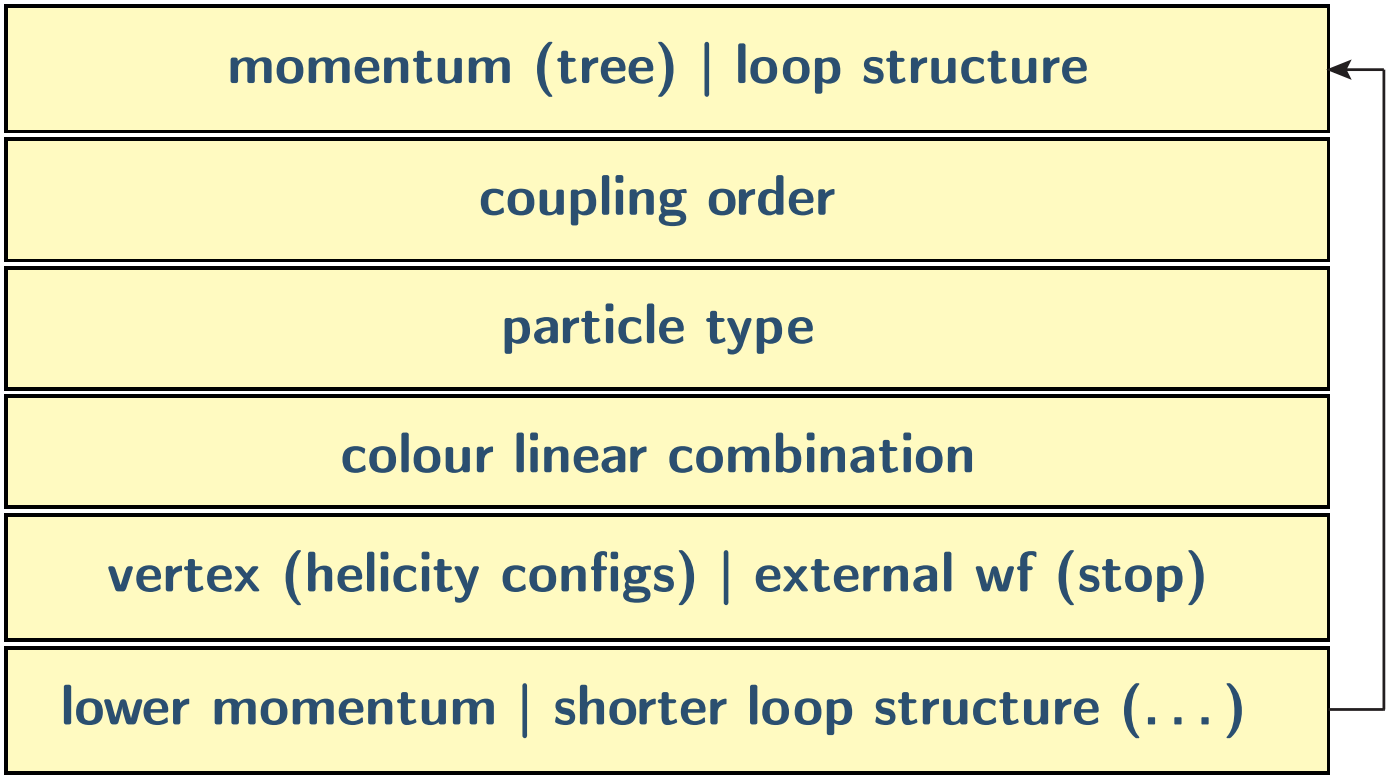
Generate n-point loop amplitudes as n+2-point tree amplitudes, closing the loops by connecting the two "loop particles" n+1 and n+2.



- Remove redundant contributions using the same cutting rule as in OL1 to choose the cut position and build direction.
- Treat special cases: two-point functions, symmetry factors, tadpoles, external wave function corrections.
- Group contributions according to their loop structure (momenta and masses of loop propagators) $\rightarrow$ different tensor integrals.

Hierarchy of diagram structures

A hierarchical structure ensures that each object is calculated only once (no identification & look-up needed as in OL1).



Apply selectors: interference selection, diagram selection (e.g. resonances).

Reference counters to decide which objects are not needed $\rightarrow$ discard.

Loop numerics I

Constructing the numerator $\mathcal{N}(q)$ of a loop amplitude involves the same steps as the tree calculation, but with different numeric routines for the "loop wave functions".

$$\mathcal{N}^{\beta}_{n,\alpha}(q) = X^{\beta}_{\gamma\delta}(q)\, \mathcal{N}^{\gamma}_{n\text{-}1,\alpha}(q)\, w^{\delta}(n), \qquad \mathcal{N}^{\beta}_{0,\alpha} = (\mathbb{1})^{\beta}_{\alpha}$$

Factor out the loop momentum q from the numerator polynomial and work with polynomial coefficients $\mathcal{N}^{\beta}_{\alpha}(n,q)$

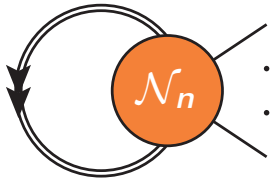
$$\mathcal{N}^{\beta}_{\alpha}(n,q) = \sum_{r=0}^{n} \mathcal{N}^{\beta}_{\mu_1 \ldots \mu_r;\alpha}(n)\, q^{\mu_1} \ldots q^{\mu_r}, \qquad X^{\beta}_{\gamma\delta} = Y^{\beta}_{\gamma\delta} + q^{\nu}\, Z^{\beta}_{\nu;\gamma\delta}$$

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

Based on an idea by [van Hameren '09] for multi-gluon amplitudes.
Also adopted by `MadLoop` [Hirschi] and `Recola` [Actis et al.].

Loop numerics II

When all tree structures have been attached to the loop, close it by contracting the indeces of the cut loop particle



$$\mathcal{N}_{\mu_1 \dots \mu_r} = \mathcal{N}^{\alpha}_{\mu_1 \dots \mu_r ; \alpha}$$

Interfere with the Born amplitude on the level of $\mathcal{N}_{\mu_1 \dots \mu_r}$ (not shown here), perform reduction to scalar integrals and evaluate them.
Contribution to the amplitude from a specific tensor integral:

$$\mathcal{N}_{\mu_1 \dots \mu_r} \cdot \int dq \, \frac{q_1 \dots q_r}{D_0 \dots D_n}$$

For now: Tensor integrals with `Collier` [Denner, Dittmaier, Hofer]
or `OPP` reduction with `CutTools` [Ossola, Papadopoulos, Pittau]
and `OneLOop` [van Hameren] for scalar integrals.

Later: On-the-fly integrand reduction [Buccioni, Pozzorini, Zoller] (Max' talk).

Physics model definition

Model definition: no serious practical limitations (I'm aware of)

- Generic power counting framework for user-defined couplings.
- Parameters which use analytic formulas and dependency tacking.

The operators which describe the interaction vertices of the theory are needed as numerical routines for each all combinations of incoming/outgoing particles. These routines are generated automatically (may also be written/optimised by hand).
(see also `ALOHA` [de Aquino et al.], `COMIX` [Höche et al.] and `REPT1L` [Lang]).

Define Feynman rules: particles entering the vertex, coupling order, colour structure, numeric routine and parameters which enter it.

For model definition, rely on external tools:
`FeynRules/NLO-CT` [Alloul et al.], `REPT1L`.

Models can be defined at runtime through direct import of UFO files.
If all operators are available models work without compilation.

First benchmarks: tree-level only

	Generate		Evaluate		
Process	**TNG**	**OL**	**TNG**	**OL**	**OL/TNG**
pure EW	[ms]	[s]	[μs]	[μs]	
$d\bar{d} \to e^+e^-$	0.075	1	1.47	1.97	1.34
$d\bar{d} \to e^+e^-\gamma$	0.165	1	3.03	3.66	1.21
$d\bar{d} \to e^+e^-\nu_e\bar{\nu}_e$	0.315	2	4.57	5.06	1.11
$d\bar{d} \to e^+e^-\nu_e\bar{\nu}_e\gamma$	0.914	11	9.43	20.6	2.18
$d\bar{d} \to e^+e^-\nu_e\bar{\nu}_e d\bar{d}$	2.94	182	29.1	151	5.19
pure QCD					
$d\bar{d} \to t\bar{t}$	0.081	1	2.24	2.88	1.29
$d\bar{d} \to t\bar{t}g$	0.283	1	4.97	5.19	1.04
$d\bar{d} \to t\bar{t}gg$	1.51	6	30.1	28.8	0.96
$d\bar{d} \to t\bar{t}ggg$	22.9	87	463	568	1.23

colour and helicity summed matrix elements, i7-4790, gcc 7.3 (preliminary)

TNG wins over OL in almost all cases; by far for EW multi-leg.

Colour treatment needs optimisation (generation & evaluation).
Lots of potential to optimise the generator for performance.

Summary

A new 1-loop generator is in the making – OpenLoops: TNG

- Building on years of experience with OpenLoops.
- Fully "dynamic": no code generation for processes,
  not even for models if operators are already implemented.
- Completely in C++.
- Tree-level is working, including UFO import for BSM models.
- Loop generator is almost complete,
  numerics including CT+R2 still requires some work
- First benchmarks look very promising, even though it hasn't been optimised for performance much
  (expected to generalise to 1-loop coefficients).
- Ready for first applications probably within 2018.
- Will be released as open source as soon as possible.