Automation of NNLO Amplitude Construction in OpenLoops

Natalie Schär

June 1, 2021

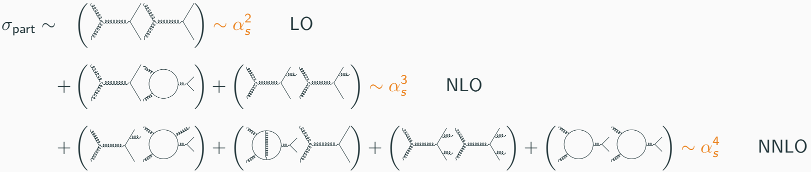
in collaboration with:
S. Pozzorini and M. F. Zoller

NNLO in OpenLoops

Monte Carlo Simulation contains:

- Hard Scattering Amplitudes → OpenLoops
- PDFs, Parton Shower, Hadronisation, Underlying Events

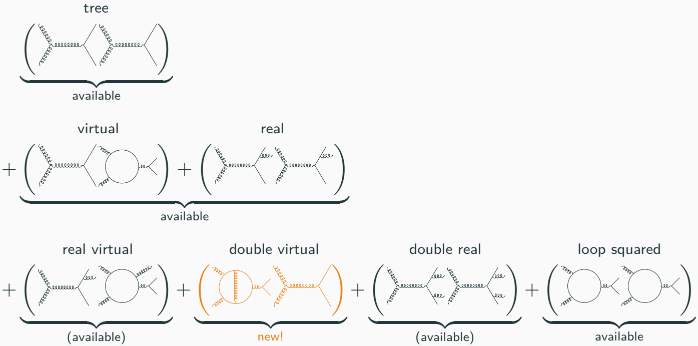
OpenLoops constructs Amplitudes from Feynman Diagrams.



NNLO required for LHC.

Components to NNLO

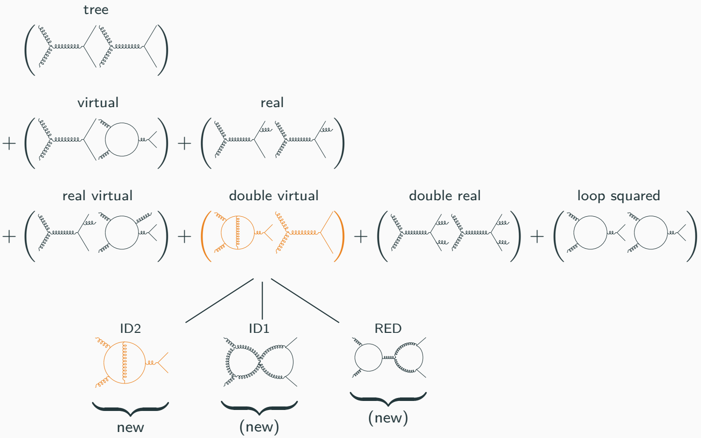
Some components to NNLO are a available in the public version of OpenLoops:



Double virtual required for NNLO.

Components to NNLO

Distinguish three types of double virtual diagrams.

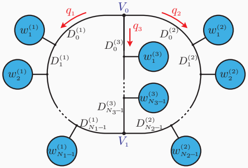


Components to NNLO

For one diagram Γ:

renormalized Amplitude (D dimensions) $= \mathcal{M}_{2,\Gamma}$(4d numerator, D-dim denominator)

$+$ rational and UV counterterms (in lower loop diagrams)

$\mathcal{M}_{2,\Gamma} =$ 

$$= \underbrace{C_{2,\Gamma}}_{\textit{color}} \sum_{r_1=0}^{R_1} \sum_{r_2=0}^{R_2} \underbrace{\mathcal{N}_{\mu_1 \cdots \mu_{r_1} \nu_1 \cdots \nu_{r_2}}}_{\text{tensor coefficient}} \underbrace{\int \mathrm{d}\bar{q}_1 \int \mathrm{d}\bar{q}_2 \, \frac{q_1^{\mu_1} \cdots q_1^{\mu_{r_1}} \, q_2^{\nu_1} \cdots q_2^{\nu_{r_2}}}{\mathcal{D}^{(1)}(\bar{q}_1)\, \mathcal{D}^{(2)}(\bar{q}_2)\, \mathcal{D}^{(3)}(\bar{q}_3)} \bigg|_{q_3 \to -(q_1+q_2)}}_{\text{tensor integral}}$$

In OpenLoops:

- tensor coefficients constructed numerically $\rightarrow$ in 4 dimensions
- restore coefficients to D dimensions by rational counterterms
- denominators kept analytical

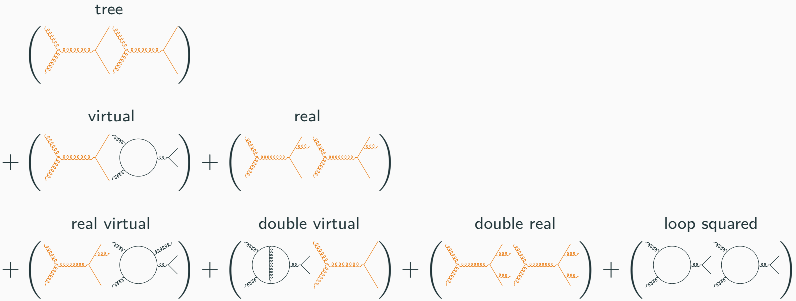
Components to NNLO Calculation

$$\mathcal{M}_{2,\Gamma} = c_{2,\Gamma} \sum_{r_1=0}^{R_1} \sum_{r_2=0}^{R_2} \underbrace{\mathcal{N}_{\mu_1\cdots\mu_{r_1}\,\nu_1\cdots\nu_{r_2}}}_{\text{tensor coefficient}} \underbrace{\int \mathrm{d}\bar{q}_1 \int \mathrm{d}\bar{q}_2 \, \frac{q_1^{\mu_1}\cdots q_1^{\mu_{r_1}}\; q_2^{\nu_1}\cdots q_2^{\nu_{r_2}}}{\mathcal{D}^{(1)}(\bar{q}_1)\,\mathcal{D}^{(2)}(\bar{q}_2)\,\mathcal{D}^{(3)}(\bar{q}_3)}\Big|_{q_3\to-(q_1+q_2)}}_{\text{tensor integral}}$$

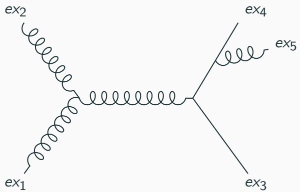
- In this talk: numerical construction of $\mathcal{N}$ from universal Feynman rules (dressing) in 4d, 2-loop irreducible (ID1, ID2), reducible (RED) diagrams
- further tasks:
  - Renormalization, Rational Terms
  - Reduction (reduction to scalar master integrals, scalar integral evaluation/library)
  - Treatment of IR divergences.

Outline

Tree Algorithm in OpenLoops



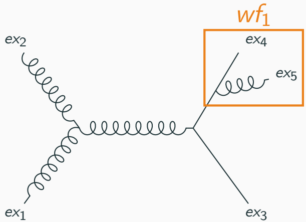
Tree Algorithm in OpenLoops: Example



start with external wavefunctions

ex_1, ex_2, ex_3, ex_4, ex_5

Tree Algorithm in OpenLoops: Example

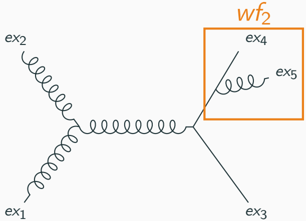


Combine ex_4, ex_5 into subtree wf_1:

```
wf1=vert_QV_A(ex4,ex5)
```

(Q=fermion, A=anti-fermion, V=boson)

Tree Algorithm in OpenLoops: Example

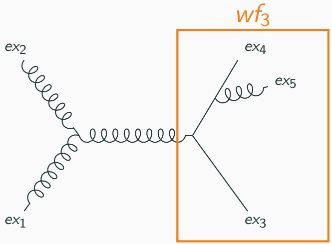


Add propagator to wf1:

```
wf1=vert_QV_A(ex4,ex5)
wf2=prop_Q_A(wf1)
```

(Q=fermion, A=anti-fermion, V=boson)

Tree Algorithm in OpenLoops: Example

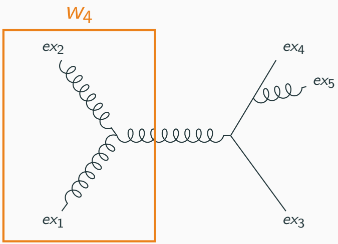


Add next external leg:

```
wf1=vert_QV_A(ex4,ex5)
wf2=prop_Q_A(wf1)
wf3=vert_QA_V(wf2,ex3)
```

(Q=fermion, A=anti-fermion, V=boson)

Tree Algorithm in OpenLoops: Example

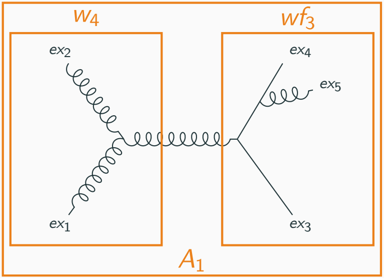


same on the other side:

```
wf1=vert_QV_A(ex4,ex5)
wf2=prop_Q_A(wf1)
wf3=vert_QA_V(wf2,ex3)

wf4=vert_VV_V(ex1,ex2)
```

(Q=fermion, A=anti-fermion, V=boson)

Tree Algorithm in OpenLoops: Example



contract into full diagram, multiply denominator:

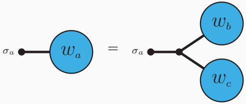
```
wf1=vert_QV_A(ex4,ex5)
wf2=prop_Q_A(wf1)
wf3=vert_QA_V(wf2,ex3)

wf4=vert_VV_V(ex1,ex2)

A1 =cont_VV(wf3, wf4)*den
```

(Q=fermion, A=anti-fermion, V=boson)

Tree Level Algorithm: Generalized

Recursively construct subtrees (=vertex+propagator):

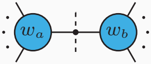
$$w_a^{\sigma_a}(k_a, h_a) =$$



$$= \underbrace{\frac{X^{\sigma_a}_{\sigma_b \sigma_c}(k_b, k_c)}{k_a^2 - m_a^2}}_{\text{model dependent}} \underbrace{w_b^{\sigma_b}(k_b, h_b)\, w_c^{\sigma_c}(k_c, h_c)}_{\text{process dependent}}$$

Then contract into full diagram:

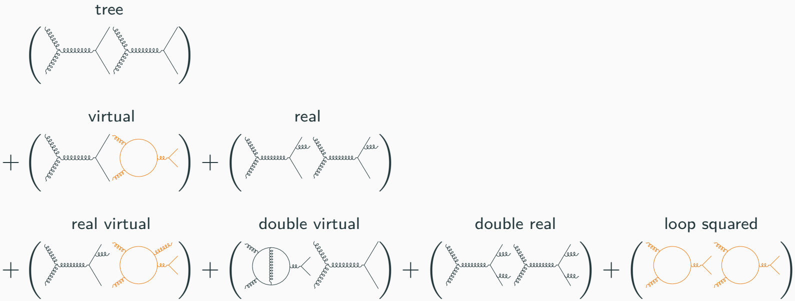
$$\mathcal{M}_{0,\Gamma}(h) =$$



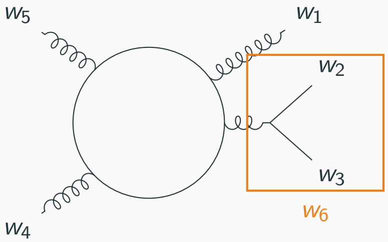
$$= C_{0,\Gamma} \cdot w_a^{\sigma_a}(k_a, h_a)\, \delta_{\sigma_a \sigma_b} \widetilde{w}_b^{\sigma_b}(k_b, h_b)$$

- diagrams constructed using universal feynman rules
- subtrees appearing in multiple diagrams are recycled

One Loop Algorithm in OpenLoops



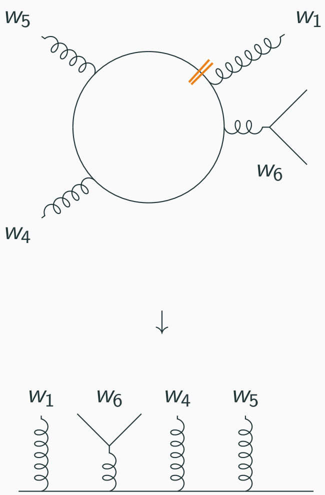
One Loop Algorithm: Example



external subtrees constructed in tree level algorithm (in combination with tree level diagrams):

$w_2, w_3 \to w_6$

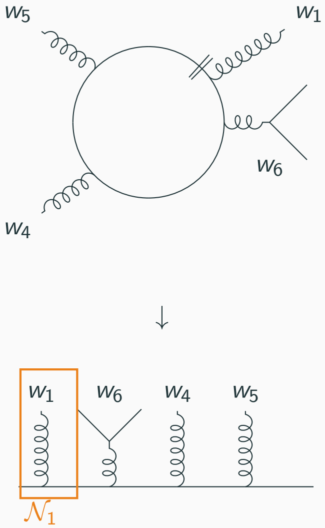
One Loop Algorithm: Example



Open Loop:

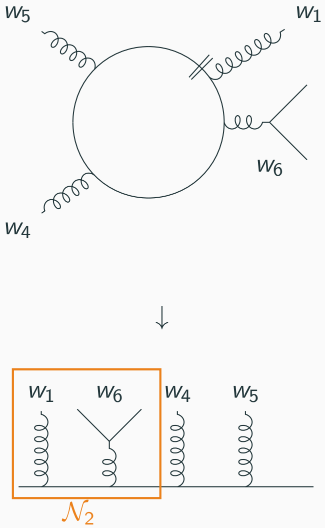
Diagram factorizes into chain of segments: $\mathcal{N} = S_1 \cdots S_N$

One Loop Algorithm: Example



Dress first segment (=vertex+propagator+subtree) S_1 attaching the external wavefunction w_1.

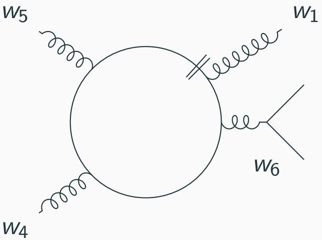
$\mathcal{N}_0 = \mathbb{1}$
$\mathcal{N}_1 = \mathcal{N}_0 \cdot S_1(w_1)$



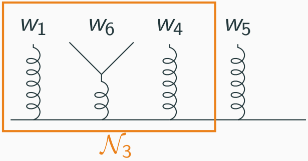
Dress second segment attaching the subtree w_6.

$$\mathcal{N}_0 = \mathbb{1}$$
$$\mathcal{N}_1 = \mathcal{N}_0 \cdot S_1(w_1)$$
$$\mathcal{N}_2 = \mathcal{N}_1 \cdot S_2(w_6)$$

One Loop Algorithm: Example



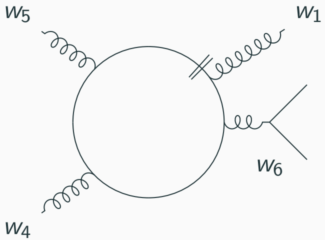
↓



Dress third segment.

$$\mathcal{N}_0 = \mathbb{1}$$
$$\mathcal{N}_1 = \mathcal{N}_0 \cdot S_1(w_1)$$
$$\mathcal{N}_2 = \mathcal{N}_1 \cdot S_2(w_6)$$
$$\mathcal{N}_3 = \mathcal{N}_2 \cdot S_3(w_4)$$

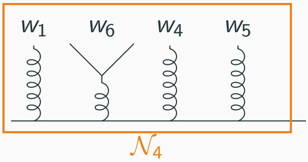
One Loop Algorithm: Example



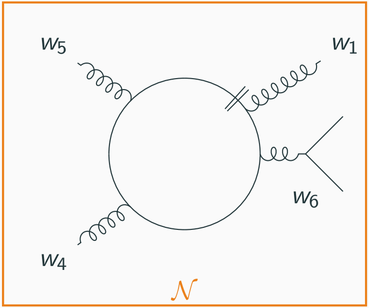
Dress last segment.

$$
\begin{aligned}
\mathcal{N}_0 &= \mathbb{1} \\
\mathcal{N}_1 &= \mathcal{N}_0 \cdot S_1(w_1) \\
\mathcal{N}_2 &= \mathcal{N}_1 \cdot S_2(w_6) \\
\mathcal{N}_3 &= \mathcal{N}_2 \cdot S_3(w_4) \\
\mathcal{N}_4 &= \mathcal{N}_3 \cdot S_4(w_5)
\end{aligned}
$$

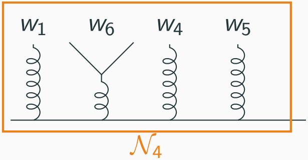
↓



One Loop Algorithm: Example



↑



Close the loop (contract open Lorentz/spinor indices).

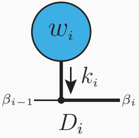
$$\mathcal{N}_0 = \mathbb{1}$$
$$\mathcal{N}_1 = \mathcal{N}_0 \cdot S_1(w_1)$$
$$\mathcal{N}_2 = \mathcal{N}_1 \cdot S_2(w_6)$$
$$\mathcal{N}_3 = \mathcal{N}_2 \cdot S_3(w_4)$$
$$\mathcal{N}_4 = \mathcal{N}_3 \cdot S_4(w_5) = \mathcal{N}_4{}_{\beta_0}^{\beta_N}$$

$$\mathcal{N} = Tr(\mathcal{N}_4{}_{\beta_0}^{\beta_N})$$

One Loop Algorithm: Generalized

Segments (=vertex+propagator+subtree(s)) can always be written as:

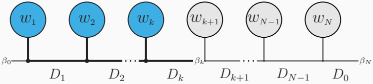
$$\left[S_i(q_1,h_i)\right]^{\beta_i}_{\beta_{i-1}} =$$



$$= \left\{ \left[Y^i_{\sigma_i}\right]^{\beta_i}_{\beta_{i-1}} + \left[Z^i_{\nu;\sigma_i}\right]^{\beta_i}_{\beta_{i-1}} \underbrace{q_1^\nu}_{\text{rank increased by 1}} \right\} w_i^{\sigma_i}(k_i,h_i)$$

Partially constructed chain (open loop):

$$\mathcal{N}_n(q_1,\hat{h}^{(1)}_k) \quad = \quad \prod_{i=1}^{k} S_i(q_1,h_i) =$$



Recursion step: $\mathcal{N}_n = \mathcal{N}_{n-1} \cdot S_n$

- Diagrams factorize into segments
- Universal Feynman Rules (encoded in Y,Z)

Helicities and Rank

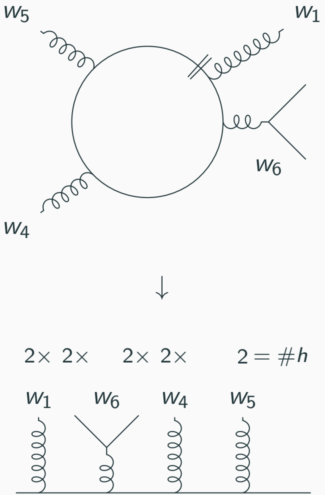
- Final result: scattering probability density $\sim \sum_h |M|^2$
- Born-Loop interference required (for virtual, real virtual etc.)
- Multiplication with Born and color factor in the beginning of construction possible $\rightarrow$ start with maximal helicities of any diagrams
- $\mathcal{U}_0(h) = 2\sum_{col} \mathcal{M}_0^* C$

Helicities may be summed after each dressing step (exploiting factorization):

$$\sum_h \mathcal{U}_0 Tr(\mathcal{N}(h)) = \sum_{h_N} \left[\cdots \sum_{h_2} \left[\sum_{h_1} \mathcal{U}_0(h) S_1(h_1) \right] S_2(h_2) \cdots \right] S_N(h_N)$$

- (in renormalizable theories) each segment:
  - increases rank by 1 (or 0)
  - decreases total helicities by a factor of # helicities of wavefunction in the segment
- minimal helicities with maximal rank $\rightarrow$ efficient, complexity is kept low in final recursion steps

Helicities and Rank: Example

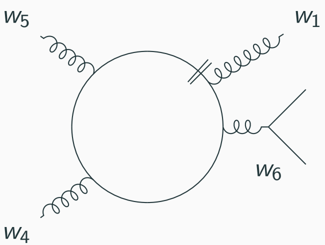


each segment:

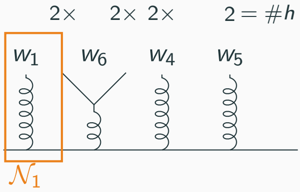
- increases rank by 1
- decreases total helicities by a factor of # helicities of wavefunction in the segment

helicities=32,
rank=0

Helicities and Rank: Example



$\downarrow$

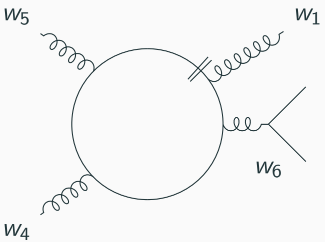


each segment:

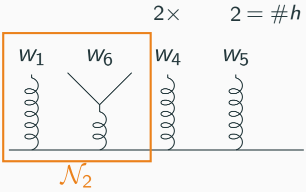
- increases rank by 1
- decreases total helicities by a factor of # helicities of wavefunction in the segment

helicities=16,
rank=1

Helicities and Rank: Example



$\downarrow$

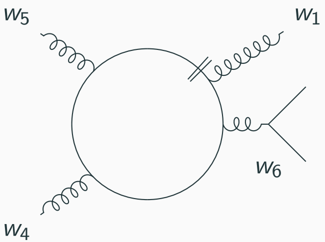


each segment:

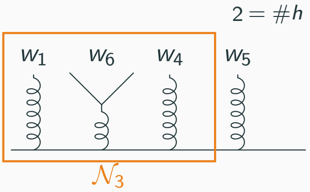
- increases rank by 1
- decreases total helicities by a factor of # helicities of wavefunction in the segment

helicities=4,
rank=2

Helicities and Rank: Example



↓

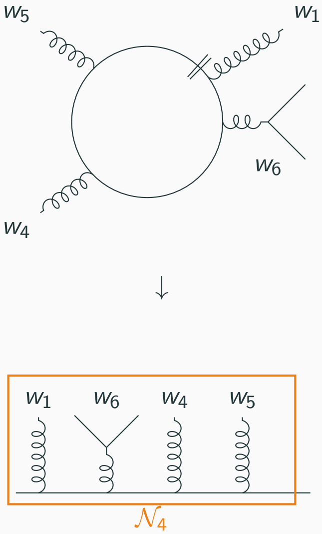


each segment:

- increases rank by 1
- decreases total helicities by a factor of # helicities of wavefunction in the segment

helicities=2,
rank=3

Helicities and Rank: Example



each segment:

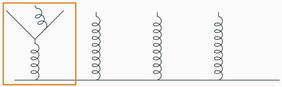
- increases rank by 1
- decreases total helicities by a factor of # helicities of wavefunction in the segment

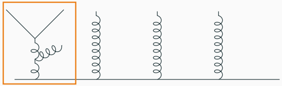
helicities=1,
rank=4

Merging

Example:

- After one dressing step subsequent dressing steps are identical.
- Topology (scalar propagators) is identical for both diagrams.
- Diagrams can be merged.





For diagrams A,B with identical segments after n dressing steps (exploit factorization):

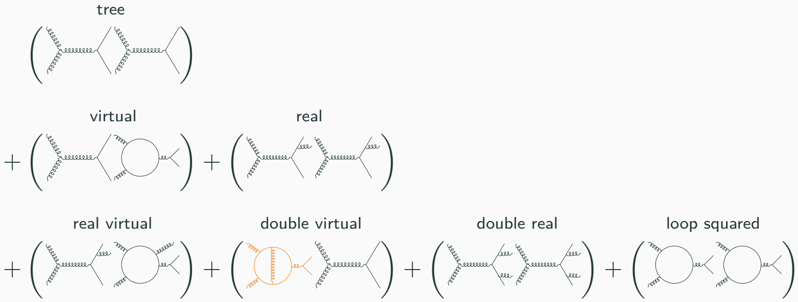
$$\mathcal{U}_{A,B} = \mathcal{U}_0\, Tr(\mathcal{N}_{A,B}) = \text{numerator} \cdot \text{Born} \cdot \text{color}$$

$$\begin{aligned}\mathcal{U}_A + \mathcal{U}_B &= (\mathcal{U}_{n,A} \cdot S_{n+1} \cdots S_N) + (\mathcal{U}_{n,B} \cdot S_{n+1} \cdots S_N) \\ &= (\mathcal{U}_{n,A} + \mathcal{U}_{n,B}) \cdot S_{n+1} \cdots S_N\end{aligned}$$

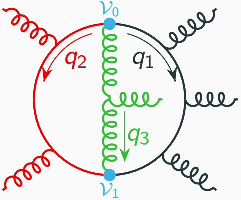
Only perform dressing steps n+1 to N once.

Highly efficient way of dressing a large number of diagrams for complicated processes.

Two Loop Algorithm in OpenLoops



Two Loop Algorithm: Components



- chain 1 = longest chain
- chain 2 = middle chain
- chain 3 = shortest chain
- $\mathcal{V}_0, \mathcal{V}_1$= vertices connecting chains
- q_1, q_2, q_3 = loop momenta $q_3 = -q_1 - q_2$

Diagram factorizes into 3 chains and 2 vertices (matrix multiplications, indices suppressed):

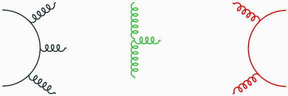
$$\mathcal{N}(q_1, q_2) = \left[\mathcal{N}^{(1)}(q_1)\right] \left[\mathcal{N}^{(2)}(q_2)\right] \left[\mathcal{N}^{(3)}(q_3)\right] \left[\mathcal{V}_0(q_1, q_2)\right] \left[\mathcal{V}_1(q_1, q_2)\right] \Big|_{q_3 \to -(q_1+q_2)}$$

Each chain in factorizes into segments

$$\mathcal{N}^{(i)}(q_i) = S_0^{(i)}(q_i)\ S_1^{(i)}(q_i) \cdots S_{N_i-1}^{(i)}(q_i)$$

Factorization results in freedom of choice for dressing algorithm.

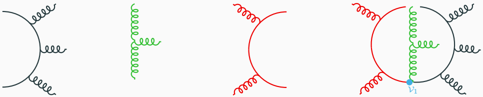
Two Loop Algorithm: Naive Approach



1. dress chains $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(2)}(q_2)$, $\mathcal{N}^{(3)}(q_3)$

$$\left[\mathcal{N}^{(1)}(q_1)\right]_{\beta_0^{(1)}}^{\beta_{N_1}^{(1)}} \left[\mathcal{N}^{(2)}(q_2)\right]_{\beta_0^{(2)}}^{\beta_{N_2}^{(2)}} \left[\mathcal{N}^{(3)}(q_3)\right]_{\beta_0^{(3)}}^{\beta_{N_3}^{(3)}}$$

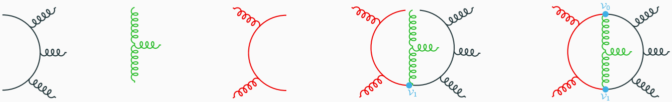
Two Loop Algorithm: Naive Approach



1. dress chains $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(2)}(q_2)$, $\mathcal{N}^{(3)}(q_3)$
2. combine with vertex $\mathcal{V}_1$, closing indices $\beta^{(1)}_{N_1},\beta^{(2)}_{N_2},\beta^{(3)}_{N_3}$

$$\left[\mathcal{N}^{(1)}(q_1)\right]_{\beta_0^{(1)}}^{\beta_{N_1}^{(1)}}\left[\mathcal{N}^{(2)}(q_2)\right]_{\beta_0^{(2)}}^{\beta_{N_2}^{(2)}}\left[\mathcal{N}^{(3)}(q_3)\right]_{\beta_0^{(3)}}^{\beta_{N_3}^{(3)}}\left[\mathcal{V}_1(q_1,q_2)\right]_{\beta_{N_1}^{(1)}\beta_{N_2}^{(2)}\beta_{N_3}^{(3)}}$$

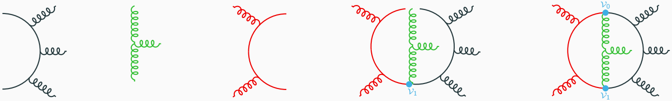
Two Loop Algorithm: Naive Approach



1. dress chains $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(2)}(q_2)$, $\mathcal{N}^{(3)}(q_3)$
2. combine with vertex $\mathcal{V}_1$, closing indices $\beta^{(1)}_{N_1}, \beta^{(2)}_{N_2}, \beta^{(3)}_{N_3}$
3. combine with vertex $\mathcal{V}_0$, closing indices $\beta^{(1)}_{0}, \beta^{(2)}_{0}, \beta^{(3)}_{0}$

$$\left[\mathcal{N}^{(1)}(q_1)\right]^{\beta^{(1)}_{N_1}}_{\beta^{(1)}_0} \left[\mathcal{N}^{(2)}(q_2)\right]^{\beta^{(2)}_{N_2}}_{\beta^{(2)}_0} \left[\mathcal{N}^{(3)}(q_3)\right]^{\beta^{(3)}_{N_3}}_{\beta^{(3)}_0} \left[\mathcal{V}_1(q_1, q_2)\right]_{\beta^{(1)}_{N_1}\beta^{(2)}_{N_2}\beta^{(3)}_{N_3}} \left[\mathcal{V}_0(q_1, q_2)\right]^{\beta^{(1)}_0\beta^{(2)}_0\beta^{(3)}_0}$$

Two Loop Algorithm: Naive Approach



1. dress chains $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(2)}(q_2)$, $\mathcal{N}^{(3)}(q_3)$
2. combine with vertex $\mathcal{V}_1$, closing indices $\beta^{(1)}_{N_1}, \beta^{(2)}_{N_2}, \beta^{(3)}_{N_3}$
3. combine with vertex $\mathcal{V}_0$, closing indices $\beta^{(1)}_{0}, \beta^{(2)}_{0}, \beta^{(3)}_{0}$
4. map momenta, loop over helicities

$$\left[\mathcal{N}^{(1)}(q_1)\right]^{\beta^{(1)}_{N_1}}_{\beta^{(1)}_0}\left[\mathcal{N}^{(2)}(q_2)\right]^{\beta^{(2)}_{N_2}}_{\beta^{(2)}_0}\left[\mathcal{N}^{(3)}(q_3)\right]^{\beta^{(3)}_{N_3}}_{\beta^{(3)}_0}\left[\mathcal{V}_1(q_1,q_2)\right]_{\beta^{(1)}_{N_1}\beta^{(2)}_{N_2}\beta^{(3)}_{N_3}}\left[\mathcal{V}_0(q_1,q_2)\right]^{\beta^{(1)}_0\beta^{(2)}_0\beta^{(3)}_0}\Big|_{q_3\to-(q_1+q_2)}$$

Two Loop Algorithm: Observations and Challenges

$$\left[\mathcal{N}^{(1)}(q_1)\right]_{\beta_0^{(1)}}^{\beta_{N_1}^{(1)}} \left[\mathcal{N}^{(2)}(q_2)\right]_{\beta_0^{(2)}}^{\beta_{N_2}^{(2)}} \left[\mathcal{N}^{(3)}(q_3)\right]_{\beta_0^{(3)}}^{\beta_{N_3}^{(3)}} \left[\mathcal{V}_0(q_1,q_2)\right]^{\beta_0^{(1)}\beta_0^{(2)}\beta_0^{(3)}} \left[\mathcal{V}_1(q_1,q_2)\right]_{\beta_{N_1}^{(1)}\beta_{N_2}^{(2)}\beta_{N_3}^{(3)}}\Big|_{q_3\to-(q_1+q_2)}$$

1. dress chains $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(2)}(q_2)$, $\mathcal{N}^{(3)}(q_3)$
2. combine with vertex $\mathcal{V}_1$, closing indices $\beta_{N_1}^{(1)}\beta_{N_2}^{(2)}\beta_{N_3}^{(3)}$
3. combine with vertex $\mathcal{V}_0$, closing indices $\beta_0^{(1)},\beta_0^{(2)},\beta_0^{(3)}$
4. map momenta, loop over helicities

Observations:

- step 2. is performed for 6 open spinor/Lorentz indices
- step 3. is preformed for 3 open spinor/Lorentz indices
- in step 2,3 we have maximal ranks, as all chains have been fully dressed
- the mapping in step 4 is performed for maximal ranks
- all dressing steps are performed for all helicities

This is very inefficient.

Cost Simulation for Two Loop Algorithm

- factorization: freedom of order in combining chains and vertices
- full algorithm: N recursion steps with partially dressed numerators
  $\mathcal{N}_n = \mathcal{N}_{n-1} X_n$,
  with building blocks $X_n \in \{S_k^{(i)}, \mathcal{V}_j, \mathcal{N}^{(i)}, \mathcal{M}_0^* C\}$
- CPU cost $\sim$ # multiplications
- $\rightarrow$ cost simulation tracking # components and multiplications
- test different variants to determine most efficient algorithm for two loop diagrams

Two Loop Algorithm in OpenLoops

0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type

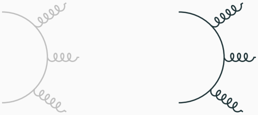
Two Loop Algorithm in OpenLoops

0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type

1a. Initial Condition for chain 1 (longest chain): Born $\times$ color factor. Start with maximal # helicities.

$$\mathcal{U}_0^{(1)} = 2 \sum_{col} \mathcal{C} \mathcal{M}_0^*$$

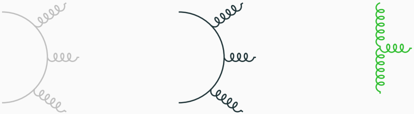
Two Loop Algorithm in OpenLoops



0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type
1a. Initial Condition for chain 1 (longest chain): Born $\times$ color factor. Start with maximal # helicities.
1b. Dress ($\mathcal{N}^{(1)}(q_1)$ $\times$ Born $\times$ color) summing helicities at each vertex (as at one loop).

$$\mathcal{U}_n^{(1)} = \mathcal{U}_{n-1}^{(1)} S_n^{(1)}, \qquad \mathcal{U}_0^{(1)} = 2\sum_{col} C\mathcal{M}_0^*$$

Two Loop Algorithm in OpenLoops



0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type

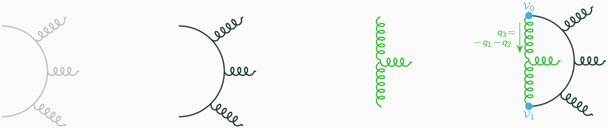
1a. Initial Condition for chain 1 (longest chain): Born $\times$ color factor. Start with maximal # helicities.

1b. Dress ($\mathcal{N}^{(1)}(q_1) \times$ Born $\times$ color) summing helicities at each vertex (as at one loop).

2. Dress $\mathcal{N}^{(3)}(q_3)$, start with no helicities, new helicities enter at each vertex.

$$\mathcal{N}^{(3)}_n(q_3) = \mathcal{N}^{(3)}_{n-1} S^{(3)}_n, \qquad \mathcal{N}^{(3)}_0 = \mathbb{1},$$

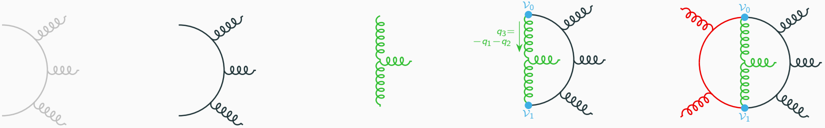
Two Loop Algorithm in OpenLoops



0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type
1a. Initial Condition for chain 1 (longest chain): Born $\times$ color factor. Start with maximal # helicities.
1b. Dress ($\mathcal{N}^{(1)}(q_1)$ $\times$ Born $\times$ color) summing helicities at each vertex (as at one loop).
2. Dress $\mathcal{N}^{(3)}(q_3)$, start with no helicities, new helicities enter at each vertex.
3. Attach $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(3)}(q_3)$ to $\mathcal{V}_0$ and $\mathcal{V}_1$, map $q_3 \to -q_1 - q_2$, sum hels of $\mathcal{N}^{(3)}(q_3), \mathcal{V}_1, \mathcal{V}_0$.

$$[\mathcal{U}^{(13)}]^{\beta^{(2)}_{N_2}}_{\beta^{(2)}_0} = [\mathcal{U}^{(1)}]^{\beta^{(1)}_{N_1}}_{\beta^{(1)}_0} [\mathcal{N}^{(3)}]^{\beta^{(3)}_{N_3}}_{\beta^{(3)}_0} \left[\mathcal{V}_0(q_1, q_3)\right]^{\beta^{(1)}_0 \beta^{(2)}_0 \beta^{(3)}_0} \left[\mathcal{V}_1(q_1, q_3)\right]_{\beta^{(1)}_{N_1} \beta^{(2)}_{N_2} \beta^{(3)}_{N_3}} \Big|_{q_3 \to -(q_1 + q_2)}$$

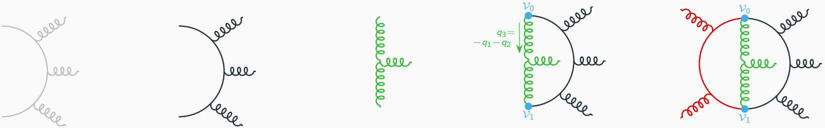
Two Loop Algorithm in OpenLoops



0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type
1a. Initial Condition for **chain 1** (longest chain): Born $\times$ color factor. Start with maximal # helicities.
1b. Dress ($\mathcal{N}^{(1)}(q_1)$ $\times$ Born $\times$ color) summing helicities at each vertex (as at one loop).
2. Dress $\mathcal{N}^{(3)}(q_3)$, start with no helicities, new helicities enter at each vertex.
3. Attach $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(3)}(q_3)$ to $\mathcal{V}_0$ and $\mathcal{V}_1$, map $q_3 \to -q_1 - q_2$, sum hels of $\mathcal{N}^{(3)}(q_3)$,$\mathcal{V}_1$,$\mathcal{V}_0$.
4. Attach $\mathcal{N}^{(2)}(q_2)$ segments to previously constructed object, sum helicities at each vertex.

$$\mathcal{U}_n^{(123)} = \mathcal{U}_{(n-1)}^{(123)} S_n^{(2)}, \qquad \mathcal{U}_0^{(123)} = \mathcal{U}^{(13)} = \mathcal{U}^{(1)}(q_1)\mathcal{N}^{(3)}(q_3)\mathcal{V}_0(q_1,q_2)\mathcal{V}_1(q_1,q_2)$$

Two Loop Algorithm in OpenLoops



0. Sort chains by length: $N_1 \geq N_2 \geq N_3$, choose order of $\mathcal{V}_0$, $\mathcal{V}_1$ by vertex type

1a. Initial Condition for **chain 1** (longest chain): Born $\times$ color factor. Start with maximal # helicities.

1b. Dress ($\mathcal{N}^{(1)}(q_1) \times$ Born $\times$ color) summing helicities at each vertex (as at one loop).

2. Dress $\mathcal{N}^{(3)}(q_3)$, start with no helicities, new helicities enter at each vertex.

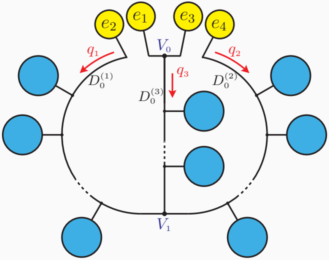
3. Attach $\mathcal{N}^{(1)}(q_1)$, $\mathcal{N}^{(3)}(q_3)$ to $\mathcal{V}_0$ and $\mathcal{V}_1$, map $q_3 \to -q_1 - q_2$, sum hels of $\mathcal{N}^{(3)}(q_3), \mathcal{V}_1, \mathcal{V}_0$.

4. Attach $\mathcal{N}^{(2)}(q_2)$ segments to previously constructed object, sum helicities at each vertex.

This algorithm is two orders of magnitude faster than the naive approach.

Pseudotree Test

Test validity and numerical stability of two loop algorithm without computing tensor integrals.

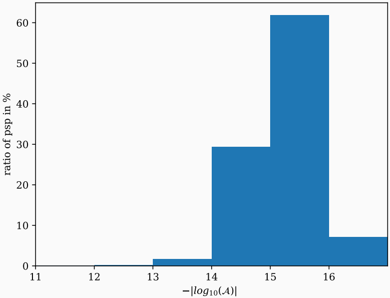


- Insert pseudo wavefunctions $e_1, e_2, e_3, e_4 \rightarrow$ saturate indices
- set q_1, q_2 to random (constant) values, contract tensor coefficients $\mathcal{N}_{\mu_1 \ldots \mu_{r_1} \nu_1 \ldots \nu_{r_2}}$ with fixed-value tensor integrand $\frac{q_1^{\mu_1} \cdots q_1^{\mu_{r_1}} q_2^{\nu_1} \cdots q_1^{\nu_{r_2}}}{\mathcal{D}(q_1, q_2)}$
- $\rightarrow$ compare with well tested tree level algorithm
- establish quad precision as benchmark, perfect (16 digit) agreement at quad precision

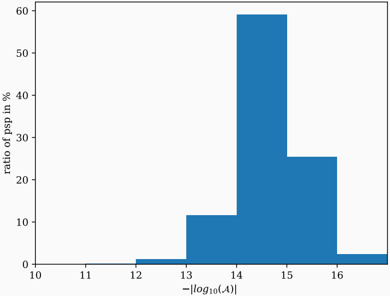
Accuracy

Two loop algorithm using pseudotree mode for 10^5 uniform random phase space points. Numerical stability of double (dp) vs quad (qp) precision scattering probability density $\mathcal{W}_{02} = \sum_{hel} \sum_{col} 2Re[\mathcal{M}_0^* \mathcal{M}_2]$:

Process: $gg \to \bar{t}t$



Process: $d\bar{d} \to u\bar{u}g$

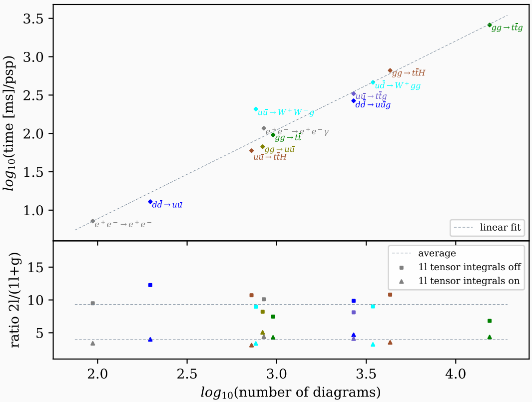


$$\text{Relative Error: } \mathcal{A} = \frac{|\mathcal{W}_{02}^{dp} - \mathcal{W}_{02}^{qp}|}{Min(|\mathcal{W}_{02}^{dp}|, |\mathcal{W}_{02}^{qp}|)}$$

Excellent numerical stability. Essential for full calculation (tensor integral reduction will be main source of instabilities).

Timings for Two Loop Tensor Coefficients

QED, QCD and SM (NNLO QCD) processes (single intel i7-6600U, 2.6 GHz, 16GB RAM, 1000 psp)



- $2 \to 2$ process: 6-100ms/psp
- $2 \to 3$ process: 60-2500ms/psp

Runtime $\propto$ # diagrams

time/psp/diagram $\sim 150\ \mu s$

Constant ratios between NNLO virtual (2l) and real-virtual (1l+g):

$$\frac{\text{2l (tensor coefficients)}}{\text{1l+g (tensor coefficients)}} \sim 9$$

$$\frac{\text{2l (tensor coefficients)}}{\text{1l+g (full calculation)}} \sim 4$$

Strong CPU performance, comparable to real-virtual corrections in OpenLoops.

Conclusion

New algorithm for two loop tensor coefficients:

- Excellent numerical stability
- Highly efficient, comparable to real virtual corrections
  - determined most efficient algorithm through cost simulation
  - exploit factorization of two loop diagrams into chains and vertices for ideal order
  - exploit factorization of chains and on the fly helicity summation for efficient treatment of individual building blocks.
  - merging and recycling of dressing steps.
- Fully implemented for NNLO QED and QCD Corrections to SM (reducible and irreducible)
- Fully generic algorithm

next steps

- UV counterterms and rational counterterms
- tensor integrals (reduction and evaluation)

End

Factorization into Segments

$$\mathcal{N}(q_1, q_2) = \left[\mathcal{N}^{(1)}(q_1)\right]_{\beta_0^{(1)}}^{\beta_{N_1}^{(1)}} \left[\mathcal{N}^{(2)}(q_2)\right]_{\beta_0^{(2)}}^{\beta_{N_2}^{(2)}} \left[\mathcal{N}^{(3)}(q_3)\right]_{\beta_0^{(3)}}^{\beta_{N_3}^{(3)}} \cdot$$
$$\cdot \left[\mathcal{V}_0(q_1, q_2)\right]^{\beta_0^{(1)}\beta_0^{(2)}\beta_0^{(3)}} \left[\mathcal{V}_1(q_1, q_2)\right]_{\beta_{N_1}^{(1)}\beta_{N_2}^{(2)}\beta_{N_3}^{(3)}} \Big|_{q_3 \to -(q_1+q_2)}$$

$$\mathcal{N}^{(i)}(q_i)_{\beta_0^{(i)}}^{\beta_{N_i}^{(i)}} = S_0^{(i)}(q_i)_{\beta_0^{(i)}}^{\beta_1^{(i)}} \; S_1^{(i)}(q_i)_{\beta_1^{(i)}}^{\beta_2^{(i)}} \cdots S_{N_i-1}^{(i)}(q_i)_{\beta_{N_i-1}^{(i)}}^{\beta_{N_i}^{(i)}}$$

Helicities

There are three ways of treating helicities along the three chains of a two-loop 1PI diagram:

- Global helicity loop (like in OpenLoops 1) $\rightarrow$ this is sure to be the most inefficient.
- "Down" method (represented by downward arrows): Use on-the-fly helicity summation (like in OpenLoops 2), i.e. the number of active helicities is reduced in each step. Requires interference with Born before. After each step we have a helicity array with the d.o.f. of the undressed segments.
- "Up" method (represented by upward arrow): Helicity arrays are constructed for the d.o.f. of the already dressed segments and extended in each dressing step by the d.o.f. of the attached subtree(s).

Rank Optimization Example

Before mapping:
Chain 3 (green) has rank 2, V0V1 have rank 0
$\rightarrow q_3^2 = (-q_1 - q_2)^2 = q_1^2 - 2q_1q_2 + q_2^2$

rank in q_1 is increased by 2 AND rank in q_2 is increased by 0
OR
rank in q_1 is increased by 0 AND rank in q_2 is increased by 2
OR
rank in q_1 is increased by 1 AND rank in q_2 is increased by 1

maximum ranks in q_1 and q_2 are not independent,
superfluous ranks can be removed

ranks

	component	label
r=0	1	1
r=1	q_0	2
r=1	q_1	3
r=1	q_2	4
r=1	q_3	5
r=2	q_0^2	6
r=2	$q_1 q_2$	7
r=2	$q_1 q_3$	8
r=2	$q_1 q_4$	9
r=2	q_2^2	10
...	...	...
r=2	q_3^2	15