Targeting Multi-Loop Integrals with Neural Networks

Ramon Winterhalder

May 12, 2022

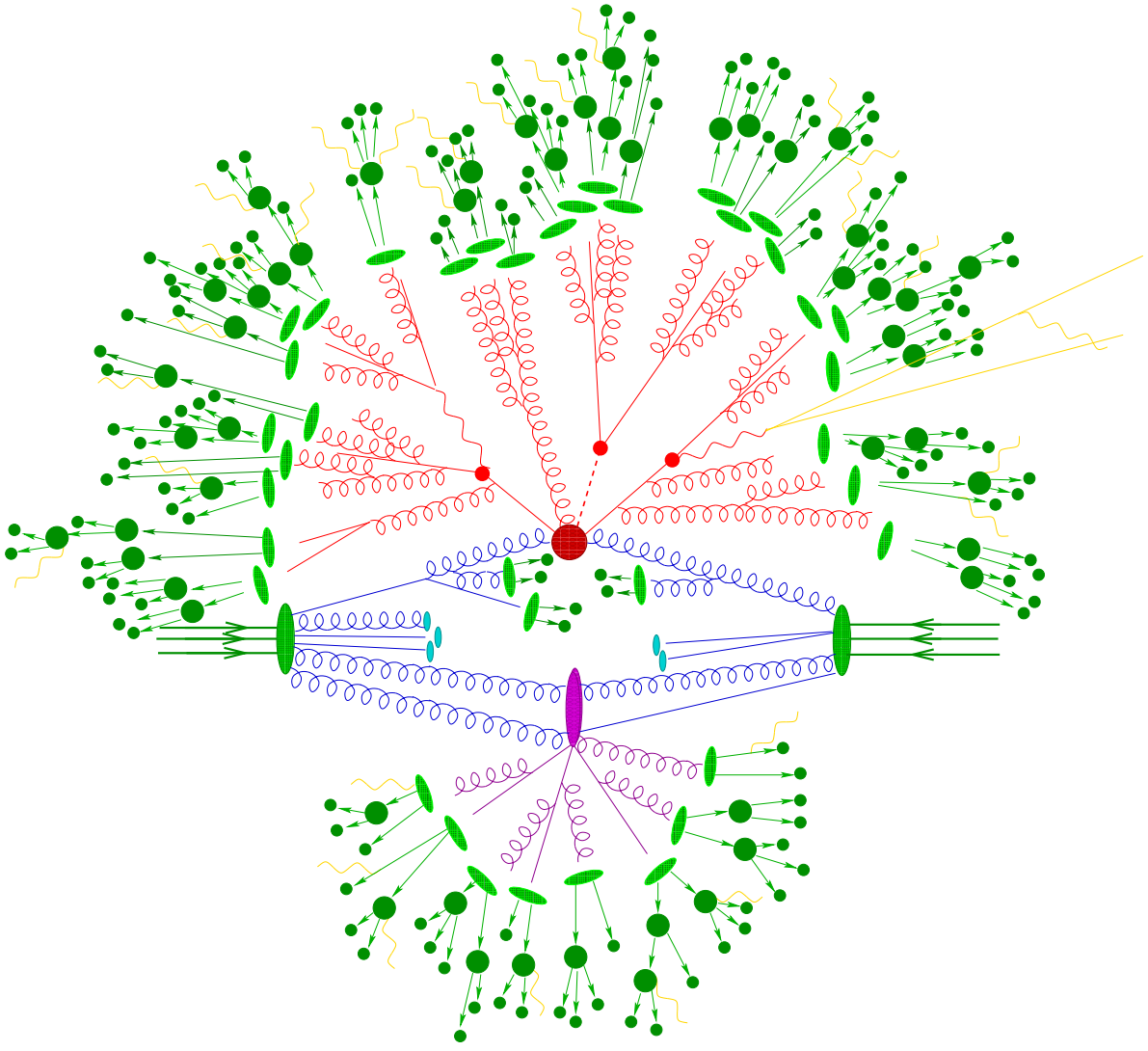
5th IML Workshop – CERN 2022

Based on [2112.09145], SciPost Phys. 12, 129 (2022)

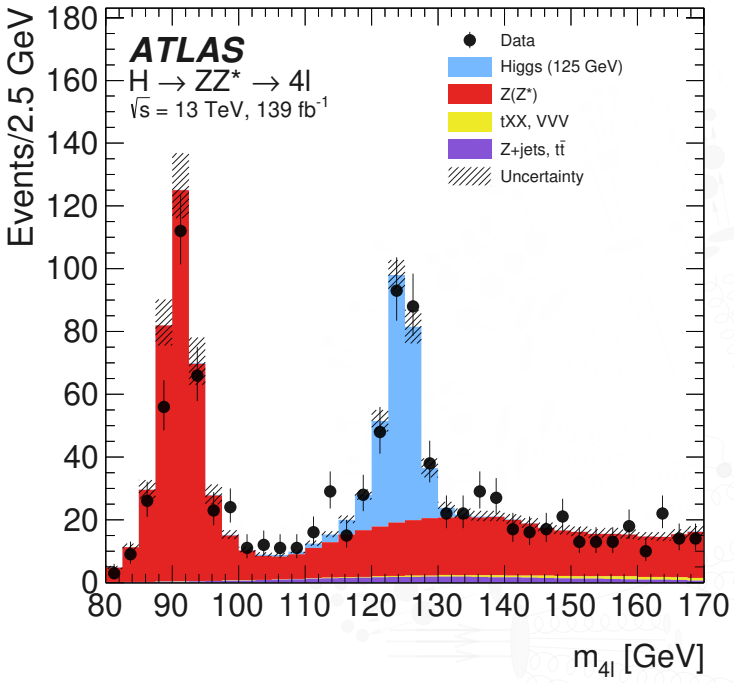
in collaboration with Vitaly Magerya, Emilio Villa, Stephen P. Jones, Matthias Kerner, Anja Butter, Gudrun Heinrich, and Tilman Plehn

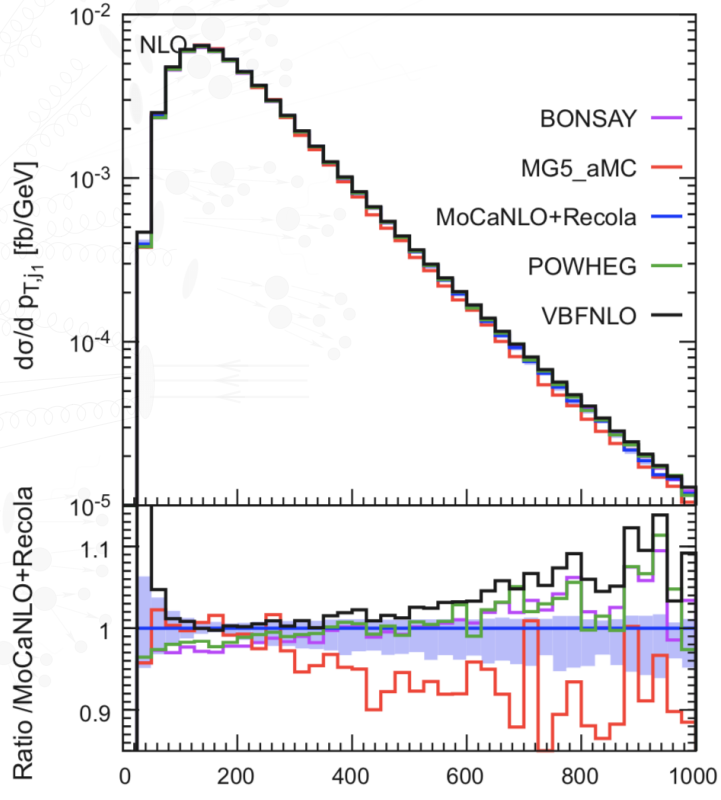
UCLouvain

Data analysis in HEP



Data analysis in HEP





Theory predictions in HEP

$$\mathcal{A}_{n,\{\lambda,c,\dots\}}(p_a, p_b | p_1, \dots, p_n) : \mathbb{M} \to \mathbb{C}$$

Quantum numbers:
spin, colour charge etc.

Kinematics:
Momenta in Minkowski space, masses, etc.

$$\sigma_n = \frac{1}{\text{flux}} \sum_{a,b} \int \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \int \mathrm{d}\Phi_n \, \left\langle |\mathcal{A}_n(p_a, p_b | p_1, \dots, p_n)|^2 \right\rangle$$

Cross section:
more generally, differential observables*

PDFs:
convolution over all possible initial state configurations

Phase-space integral:
over final state kinematics

Squared amplitude:
summed over final states, averaged over initial states

Theory predictions in HEP

Outline for today

→ Focus on **calculation of amplitudes**

Quantum numbers:
spin, colour charge etc.

Kinematics:
Momenta in Minkowski space, masses, etc.

$$\sigma_n = \frac{1}{\text{flux}} \sum_{a,b} \int \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \int \mathrm{d}\Phi_n \, \left\langle |\mathcal{A}_n(p_a, p_b | p_1, \ldots, p_n)|^2 \right\rangle$$

Cross section:
more generally, differential observables*

PDFs:
convolution over all possible initial state configurations

Phase-space integral:
over final state kinematics

Squared amplitude:
summed over final states, averaged over initial states

Theory predictions in HEP

Outline for today

→ Focus on **calculation of amplitudes**

Towards high precision

Fixed-order perturbative expansion

$$\mathcal{A}_n = \mathcal{A}_n^{(0)} + \underbrace{\mathcal{A}_n^{(1)} + \mathcal{A}_n^{(2)} + \dots}_{\text{loop amplitudes}}$$

Theory predictions in HEP

Outline for today

→ Focus on **calculation of amplitudes**

Towards high precision

Fixed-order perturbative expansion

$$\mathcal{A}_n = \mathcal{A}_n^{(0)} + \underbrace{\mathcal{A}_n^{(1)} + \mathcal{A}_n^{(2)} + \dots}_{\text{loop amplitudes}}$$

Calculation of loop amplitudes

For complicated integrals

analytic solution impractical

→ numerical evaluation in Feynman parameter space

Multi-loop integral

Feynman parametrization

$$G = \frac{(-1)^\nu\,\Gamma(\nu - LD/2)}{\prod_{j=1}^{N}\Gamma(\nu_j)} \int \left(\prod_{j=1}^{N} \mathrm{d}x_j\, x_j^{\nu_j - 1}\right) \delta\left(1 - \sum_{l=1}^{N} x_l\right) \frac{\mathcal{U}^{\nu-(L+1)D/2}}{\mathcal{F}^{\nu-LD/2}}$$

with $\mathcal{U} \equiv \det(M)$ and $\mathcal{F} \equiv \det(M)\left[\sum_{i,j=1}^{L} Q_i \left(M^{-1}\right)_{ij} Q_j - J - \mathrm{i}\delta\right]$.

$\mathcal{U}$: polynomial in x_i only

$\mathcal{F}$: depends on x_i and kinematic invariants s_{ij}, m_i^2

Multi-loop integral

Feynman parametrization

$$G = \frac{(-1)^\nu \, \Gamma(\nu - LD/2)}{\prod_{j=1}^{N} \Gamma(\nu_j)} \int \left(\prod_{j=1}^{N} \mathrm{d}x_j \, x_j^{\nu_j - 1} \right) \delta\left(1 - \sum_{l=1}^{N} x_l\right) \frac{\mathcal{U}^{\nu-(L+1)D/2}}{\mathcal{F}^{\nu-LD/2}} ;$$

with $\mathcal{U} \equiv \det(M)$ and $\mathcal{F} \equiv \det(M) \left[\sum_{i,j=1}^{L} Q_i \left(M^{-1}\right)_{ij} Q_j - J - \mathrm{i}\delta \right]$.

$\mathcal{U}$: polynomial in x_i only

$\mathcal{F}$: depends on x_i and kinematic invariants s_{ij}, m_i^2

Caveats

Careful treatment of singularities in the integral!

Singularity structure

UV and IR singularities

show up as poles $1/\epsilon^{\alpha}$

(1a) Overall UV poles: $\Gamma(\nu - LD/2) \propto \Gamma(n\epsilon)$

(1b) UV subdivergencies: arise from $\mathcal{U}(\vec{x}) = 0$ for some $x_i = 0$.

(2) IR divergencies (soft and collinear):
arise from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ for some $x_i = 0$.

Singularity structure

UV and IR singularities

show up as poles $1/\epsilon^{\alpha}$

(1a) Overall UV poles: $\Gamma(\nu - LD/2) \propto \Gamma(n\epsilon)$

(1b) UV subdivergencies: arise from $\mathcal{U}(\vec{x}) = 0$ for some $x_i = 0$.

(2) IR divergencies (soft and collinear):
arise from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ for some $x_i = 0$.

Threshold-type singularities

(3) $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ inside integration region

Singularity structure

UV and IR singularities

show up as poles $1/\epsilon^{\alpha}$

(1a) Overall UV poles: $\Gamma(\nu - LD/2)$

(1b) UV subdivergencies: arise from

(2) IR divergencies (soft and collin

arise from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ fo

Sector decomposition

algorithm to isolate and subtract poles

→ public tool pySecDec [Heinrich et al, '10,'17,'19, '21]

and expand integral

$$G = \sum_{j=-2L}^{n} C_j \epsilon^j$$

Threshold-type singularities

(3) $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ inside integration region

Singularity structure

UV and IR singularities

show up as poles $1/\epsilon^{\alpha}$

(1a) Overall UV poles: $\Gamma(\nu - LD/2)$

(1b) UV subdivergencies: arise from

(2) IR divergencies (soft and collin

arise from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ fo

Sector decomposition

algorithm to isolate and subtract poles

→ public tool pySecDec [Heinrich et al, '10,'17,'19, '21]

and expand integral

$$G = \sum_{j=-2L}^{n} C_j \epsilon^j$$

Threshold-type singularities

(3) $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ inside integration region

Parameter integrals

numerical integration of finite C_j

Singularity structure

UV and IR singularities

show up as poles $1/\epsilon^{\alpha}$

(1a) Overall UV poles: $\Gamma(\nu - LD/2)$

(1b) UV subdivergencies: arise from

(2) IR divergencies (soft and collin

arise from $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ fo

Sector decomposition

algorithm to isolate and subtract poles

→ public tool pySecDec [Heinrich et al, '10,'17,'19, '21]

and expand integral

$$G = \sum_{j=-2L}^{n} C_j \epsilon^j$$

Threshold-type singularities

(3) $\mathcal{F}(x, s_{ij}, m_i^2) = 0$ inside integration region

Parameter integrals

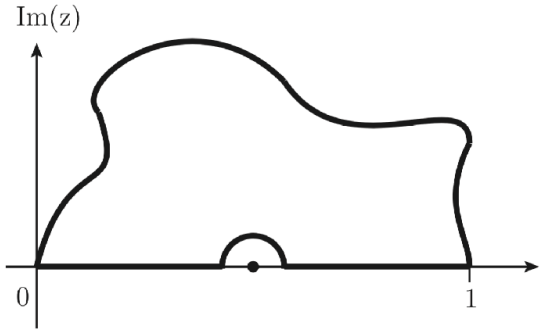
numerical integration of finite C_j

→ needs procedure to avoid poles!

Contour Deformation

Neural Contour Deformation

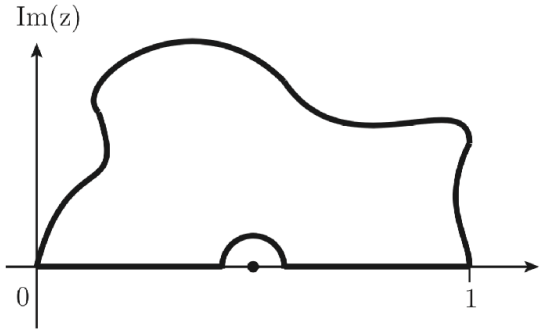
Standard contour deformation



Cauchy theorem:

$$\oint_c \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \mathcal{I}(\vec{x}) + \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

Standard contour deformation



Cauchy theorem:

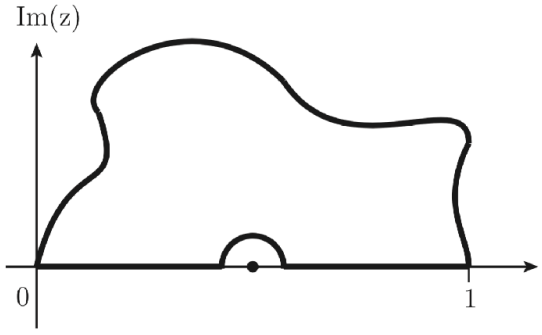
$$\oint_C \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \mathcal{I}(\vec{x}) + \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

Transformation: $x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x})$

Deformation parameter: λ_j

$$\tau_j = \lambda_j x_j (1 - x_j) \frac{\partial \mathcal{F}(\vec{x})}{\partial x_j}$$

Standard contour deformation



Cauchy theorem:

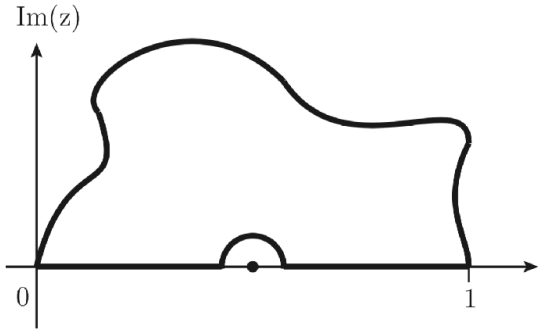
$$\oint_c \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \mathcal{I}(\vec{x}) + \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

Transformation: $x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x})$ Deformation parameter: λ_j

$$\tau_j = \lambda_j x_j (1 - x_j) \frac{\partial \mathcal{F}(\vec{x})}{\partial x_j}$$

$$\mathcal{F}(\vec{z}) = \mathcal{F}(\vec{x}) - \mathrm{i} \sum_j \lambda_j x_j (1 - x_j) \left(\frac{\partial \mathcal{F}(\vec{x})}{\partial x_j} \right)^2 - \frac{1}{2} \sum_{j,k} \tau_j \tau_k \frac{\partial^2 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k}$$
$$+ \frac{\mathrm{i}}{6} \sum_{j,k,l} \tau_j \tau_k \tau_l \frac{\partial^3 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k \partial x_l} + \cdots$$

Standard contour deformation



Cauchy theorem:

$$\oint_c \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \mathcal{I}(\vec{x}) + \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

Transformation: $x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x})$ Deformation parameter: λ_j

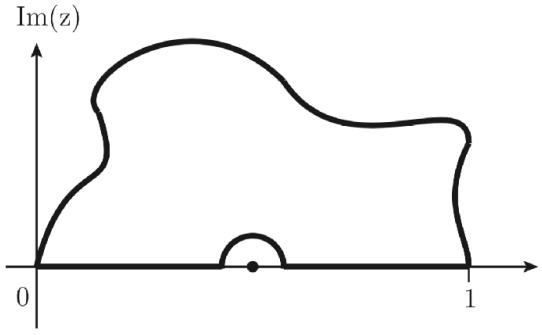
$$\tau_j = \lambda_j x_j (1 - x_j) \frac{\partial \mathcal{F}(\vec{x})}{\partial x_j}$$

$$\mathcal{F}(\vec{z}) = \mathcal{F}(\vec{x}) - \mathrm{i} \sum_j \lambda_j x_j (1 - x_j) \left(\frac{\partial \mathcal{F}(\vec{x})}{\partial x_j} \right)^2 - \frac{1}{2} \sum_{j,k} \tau_j \tau_k \frac{\partial^2 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k}$$

$$+ \frac{\mathrm{i}}{6} \sum_{j,k,l} \tau_j \tau_k \tau_l \frac{\partial^3 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k \partial x_l} + \cdots$$

always positive

Standard contour deformation



Cauchy theorem:

$$\oint_C \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z}) = \int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j \, \mathcal{I}(\vec{x}) + \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j \, \mathcal{I}(\vec{z})$$

Transformation: $x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x})$ Deformation parameter: λ_j

$$\tau_j = \lambda_j x_j (1 - x_j) \frac{\partial \mathcal{F}(\vec{x})}{\partial x_j}$$

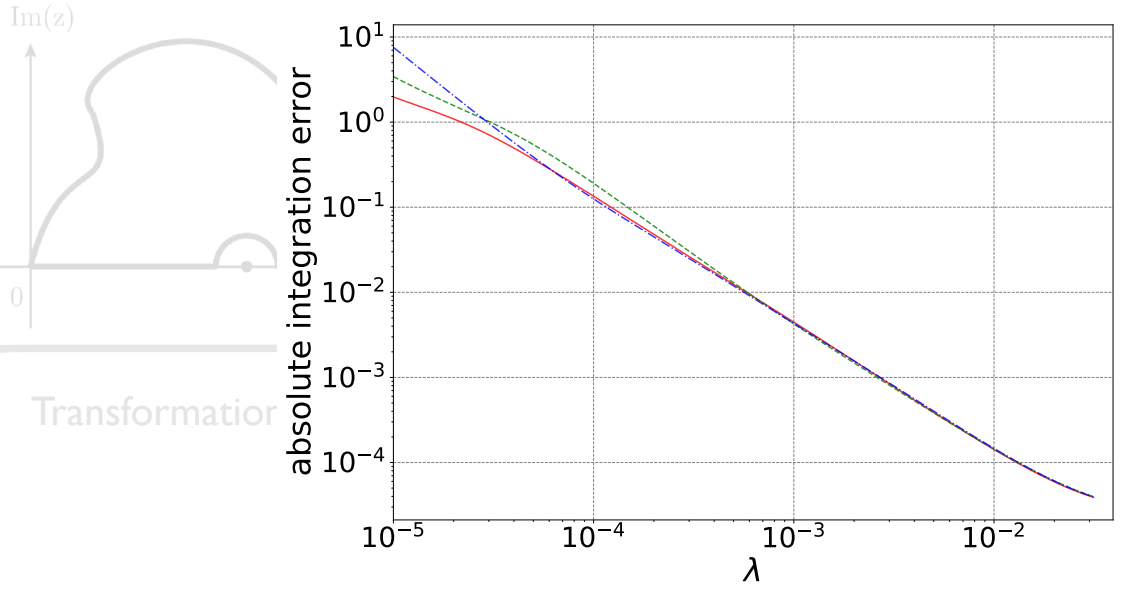
$$\mathcal{F}(\vec{z}) = \mathcal{F}(\vec{x}) - \mathrm{i} \sum_j \lambda_j x_j (1 - x_j) \left(\frac{\partial \mathcal{F}(\vec{x})}{\partial x_j} \right)^2 - \frac{1}{2} \sum_{j,k} \tau_j \tau_k \frac{\partial^2 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k}$$

$$+ \frac{\mathrm{i}}{6} \sum_{j,k,l} \tau_j \tau_k \tau_l \frac{\partial^3 \mathcal{F}(\vec{x})}{\partial x_j \partial x_k \partial x_l} + \cdots$$

always positive

Does **not** spoil sign for small enough λ

Standard contour deformation



$$\mathcal{F}(\vec{z}) = \mathcal{F}(\vec{x}) - \mathrm{i}\sum_j \lambda_j x_j(1-x_j)\left(\frac{\partial \mathcal{F}(\vec{x})}{\partial x_j}\right)^2 - \frac{1}{2}\sum_{j,k}\tau_j\tau_k\frac{\partial^2\mathcal{F}(\vec{x})}{\partial x_j \partial x_k}$$
$$+ \frac{\mathrm{i}}{6}\sum_{j,k,l}\tau_j\tau_k\tau_l\frac{\partial^3\mathcal{F}(\vec{x})}{\partial x_j\partial x_k\partial x_l} + \cdots$$

always positive

Does **not** spoil sign for small enough λ

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1-x_j)\frac{\partial F(\vec{x})}{\partial x_j}, \qquad \rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1 - x_j)\frac{\partial F(\vec{x})}{\partial x_j}, \qquad \rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Loss function

$$L = \sigma_I^2 = \frac{\langle (\mathcal{I}/\rho - I_m)^2 \rangle_x}{m-1}$$

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1-x_j)\frac{\partial F(\vec{x})}{\partial x_j}, \qquad \rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Loss function

$$L = \sigma_I^2 = \frac{\langle (\mathcal{I}/\rho - I_m)^2 \rangle_x}{m-1} + L_{\text{sign}}$$

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1-x_j)\frac{\partial F(\vec{x})}{\partial x_j}, \qquad \rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Loss function

$$L = \sigma_I^2 = \frac{\langle (\mathcal{I}/\rho - I_m)^2 \rangle_x}{m-1} + L_{\text{sign}}$$

Note: Making $\lambda_j \to \lambda_j(\vec{x})$ local has negligible effects! $\to \Lambda$-Glob

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1-x_j)\frac{\partial F(\vec{x})}{\partial x_j}, \qquad \rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Loss function

$$L = \sigma_I^2 = \frac{\langle |\mathcal{I}/\rho - I_m|^2 \rangle_x}{m-1} + L_{\text{sign}}$$

Note: Making $\lambda_j \to \lambda_j(\vec{x})$ local has negligible effects! $\to \Lambda$-Glob

Note II: The integrand is a complex-valued function!

Λ-Glob algorithm

Complex shift

$$x_j \to z_j(\vec{x}) = x_j - \mathrm{i}\tau_j(\vec{x}),$$

$$\tau_j = \lambda_j x_j(1-x_j)\frac{\partial F(\vec{x})}{\partial x_j},$$

$$\rho(\vec{z}(\vec{x})) = \left|\frac{\partial \vec{z}(\vec{x})}{\partial \vec{x}}\right|^{-1}$$

$$I = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j(\vec{x})\,\frac{\mathcal{I}(\vec{z})}{\rho(\vec{z})} \quad \Rightarrow I \simeq I_m = \frac{1}{m}\sum_{i=1}^{m}\frac{\mathcal{I}(\vec{x}_i)}{\rho(\vec{x}_i)} = \langle \mathcal{I}/\rho \rangle_x$$

Loss function

$$L = \sigma_I^2 = \frac{\langle |\mathcal{I}/\rho - I_m|^2 \rangle_x}{m-1} + L_{\text{sign}}$$

Note: Making $\lambda_j \to \lambda_j(\vec{x})$ loca

Note II: The integrand is

Caution!

Careful treatment of complex derivatives!

Complex derivatives in TensorFlow

Consider the Jacobian matrix

$$J = \begin{pmatrix} 1 + \mathrm{i}\lambda_0 & 0 \\ 0 & 1 + \mathrm{i}\lambda_1 \end{pmatrix}$$

Check GitHub and Colab for details!

Next we calculate its determinant

$$\det J = (1 + \mathrm{i}\lambda_0)(1 + \mathrm{i}\lambda_1) = (1 - \lambda_0\lambda_1) + \mathrm{i}(\lambda_0 + \lambda_1)$$

And afterwards its absolute square

$$L = \det J \cdot (\det J)^* = (1 - \lambda_0\lambda_1)^2 + (\lambda_0 + \lambda_1)^2$$

And finally we take the gradient with respect to $\vec{\lambda}$ and obtain

$$\nabla_\lambda L = \left(2\lambda_0(1 + \lambda_1)^2, 2\lambda_1(1 + \lambda_0)^2\right)$$

Now evaluating this gradient at $\vec{\lambda} = (2, 1)$ yields $\nabla_\lambda L\Big|_{\vec{\lambda}=(2,1)} = (10, 8)$.

Complex derivatives in TensorFlow

Consider the Jacobian matrix

$$J = \begin{pmatrix} 1 + \mathrm{i}\lambda_0 & 0 \\ 0 & 1 + \mathrm{i}\lambda_1 \end{pmatrix}$$

Check GitHub and Colab for details!

Next we calculate its determinant

$$\det J = (1 + \mathrm{i}\lambda_0)(1 + \mathrm{i}\lambda_1) = (1 - \lambda_0\lambda_1) + \mathrm{i}(\lambda_0 + \lambda_1)$$

And afterwards its absolute square

$$L = \det J \cdot (\det J)^* = (1 - \lambda_0\lambda_1)^2 + (\lambda_0 + \lambda_1)^2$$

And finally we take the gradient with respect to $\vec{\lambda}$ and obtain

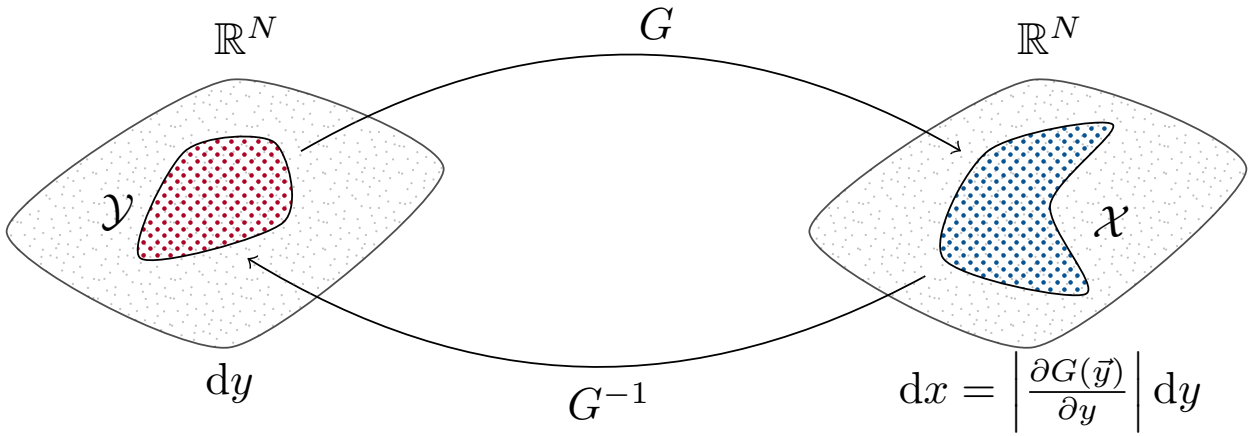
$$\nabla_\lambda L = \left(2\lambda_0(1 + \lambda_1)^2, 2\lambda_1(1 + \lambda_0)^2\right)$$

Now evaluating this gradient at $\vec{\lambda} = (2, 1)$ yields $\nabla_\lambda L\Big|_{\vec{\lambda}=(2,1)} = (10, 8)$.

However: Tensorflow obtains $\nabla_\lambda L\Big|_{\vec{\lambda}=(2,1)} = (-14, -8.8)$

Normalizing flow

Neural importance sampling Gao et al. [2001.05486, 2001.10028], Bothmann et al. [2001.05478]

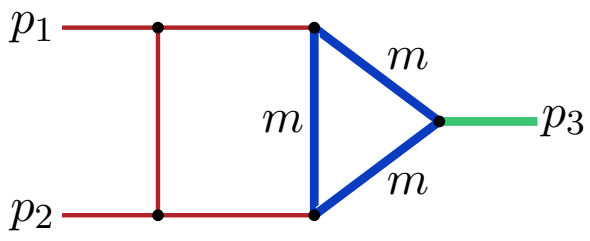


$$x_j \to x_j(\vec{y}) = G_j(\vec{y}), \qquad g(\vec{x}(\vec{y})) = \left|\frac{\partial G(\vec{y})}{\partial y}\right|^{-1}$$

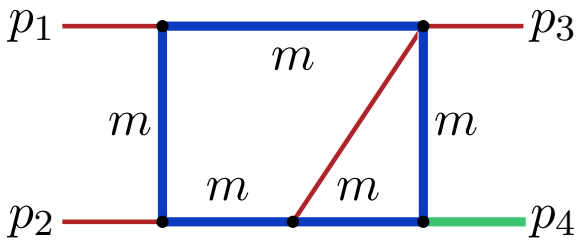
$$I_N = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathcal{I}(\vec{y}_i)}{g(\vec{y}_i)\rho(\vec{y}_i)} = \langle \mathcal{I}/(\rho g)\rangle_y$$

Results – Two-loop integrals

Example diagrams



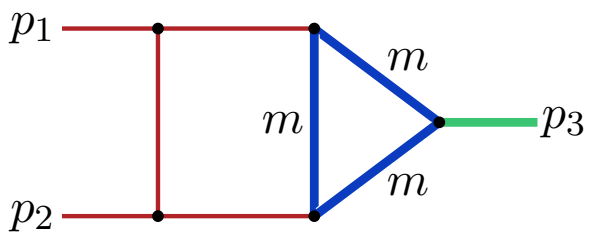
- NLO correction to $gg \to h$
- 5 feynman parameters
- 2 kinematic invariants: s, m^2

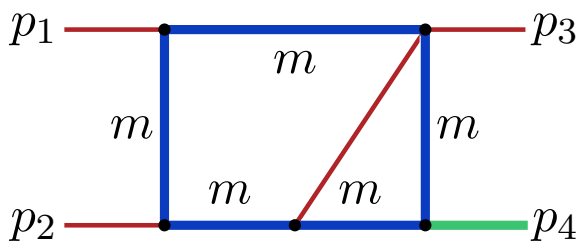


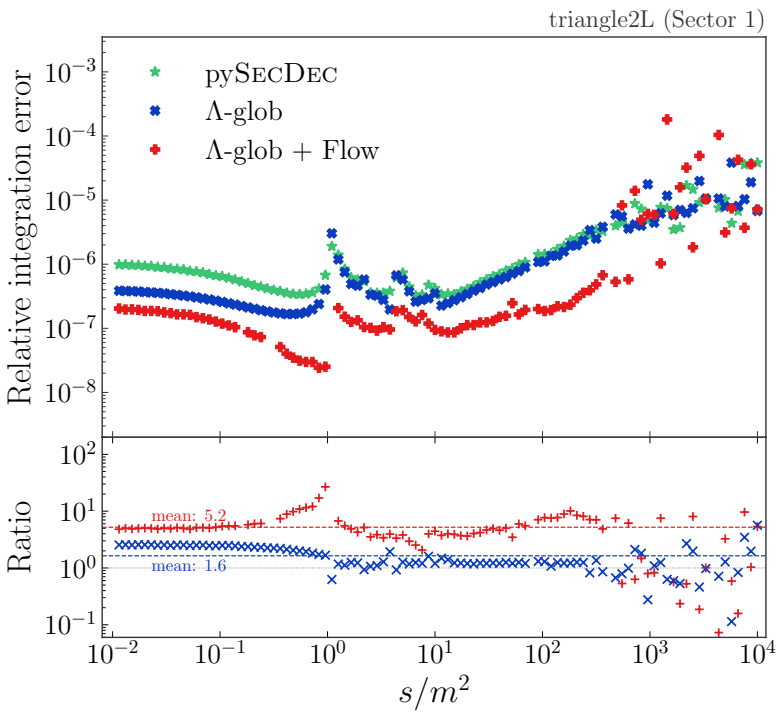
- $gg \to hj$ at two loops
- 5 feynman parameters
- 4 kinematic invariants: s, t, p_4^2, m^2

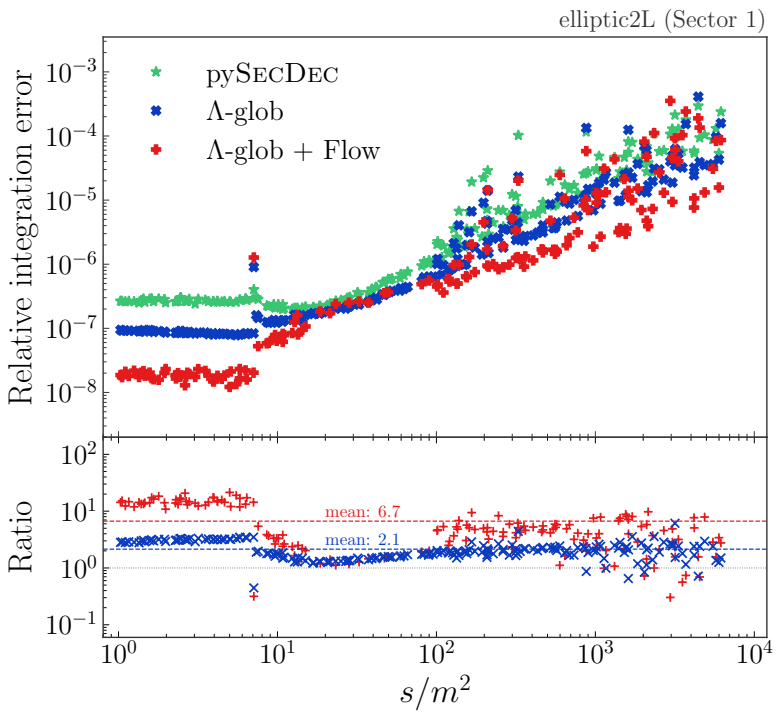
Results – Two-loop integrals

Example diagrams









Conclusion and Outlook

Summary

- Improved precision of numerical loop integrals!
- Two-step procedure:
  (a) Λ-Glob: complex shift
  (b) Normalizing Flow: importance sampling reals

Outlook

- Further investigation of more complicated integrals
- Possibly check integrals for which standard pySecDec fails.
- Integrate into full workflow of pySecDec