(Semi-)automated methods for solving Feynman integrals through differential equations

Martijn Hidding
Uppsala University

Based on work in collaboration with:
Ievgen Dubovyk, Krzysztof Grzanka, Johann Usovitsch

RADCOR/Loopfest 2021
• In recent years, the method of differential equations has proven to be an exceptionally powerful way of computing Feynman integrals.

• The effectiveness of the differential equations method is especially striking when it is applied to polylogarithmic integral families that admit an $\epsilon$-factorized (canonical) basis.

• Furthermore, numerical approaches to solving the differential equations can be efficient, precise, and may extend to cases beyond multiple polylogarithms or elliptic generalizations thereof.

• Although many individual steps have been automated, some “glue” is still missing. In this talk we will consider some steps towards a full automatization.

Introduction

[Ref: Kotikov, 1991], [Remiddi, 1997]
[Ref: Gehrmann, Remiddi, 2000]

[Ref: Henn, 2013]
e.g.: [Lee, Smirnov, Smirnov, ‘18], [Mandal, Zhao, ‘19], [Moriello, ‘19], [Bonciani, Del Duca, Frellesvig, Henn, MH, Maestri, Moriello, Salvatori, Smirnov, ‘19], [MH ‘20], [Abreu, Ita, Moriello, Page, Tschernow, Zeng ‘20]
Outline of the talk

• The method of differential equations
• Solutions through iterated series expansions
• Overview of an automated computational strategy
• The DiffExp Mathematica package & the Caesar toolbox
• Applications to a 3-loop vertex topology
Differential equations

• We consider a family of Feynman integrals:

\[
I_{a_1, \ldots, a_{n+m}} = \int \left( \prod_{i=1}^{l} \frac{d^d k_i}{i \pi^{d/2}} \right) \frac{\prod_{i=n+1}^{n+m} N_i^{-a_i}}{\prod_{i=1}^{n} D_i^{a_i}}, \quad d = d_{\text{int}} - 2\epsilon \\
D_i = -q_i^2 + m_i^2 - i\delta
\]

and a basis of master integrals \( \vec{I} \). Taking derivatives on kinematic invariants and masses and performing IBP reductions, we obtain:

\[
\partial_{s_j} \vec{I} = M_{s_j} (\{ s_i \}, \epsilon) \vec{I}
\]

• We will proceed by solving these equations iteratively in terms of one-dimensional series expansions, which will allow us to obtain numerical results everywhere in phase-space.

[Introduction, Differential equations and series solutions, 3-loop topology, Conclusion]
Differential equations

• Let us briefly consider the special case of a canonical basis. Under a change of variables \( \tilde{B} = T \tilde{I} \), we have that:

\[
\frac{\partial}{\partial s_i} \tilde{B} = \left[ (\partial_{s_i} T) T^{-1} + TM_{s_i} T^{-1} \right] \tilde{B}.
\]

• For polylogarithmic families, it is conjectured that a \( T \) exists, such that:

\[
\frac{\partial \tilde{B}}{\partial s_i} = \epsilon \frac{\partial \tilde{A}}{\partial s_i} \tilde{B}, \quad d\tilde{B} = \epsilon d\tilde{A} \tilde{B}
\]

where \( \tilde{A} \) does not depend on \( \epsilon \), and such that

\[
\tilde{A} = \sum_{i \in A} C_i \log(l_i)
\]

decomposes as a \( \mathbb{Q} \)-linear combination of logarithms of rat./algebraic functions.

[Henn, 2013]

See also:
[Lee, 1411.0911]
[Prausa, 1701.00725]
[Gituliar, Magerya, 1701.04269]
[Meyer, 1705.06252]
[Dlapa, Henn, Yan, 2002.02340]
Differential equations

• Let us parametrize the differential equations along a one-dimensional path. In other words, we consider: \( \gamma : [0, 1] \rightarrow \mathbb{C}^{|S|} \)

\[ x \mapsto (\gamma_{s_1}(x), \ldots, \gamma_{s_{|S|}}(x)) \]

• Then we have that:

\[ \partial_x \vec{B} = \varepsilon \frac{\partial \tilde{A}(\gamma(x))}{\partial x} \vec{B} \]

\[ \partial_x \vec{B} = \varepsilon \mathbf{A}_x \vec{B} \]

• Upon expanding in \( \varepsilon \), the equations can be solved order-by-order:

\[ \vec{B} = \sum_{i \geq 0} \vec{B}^{(i)} \varepsilon^i \]

\[ \vec{B}^{(i)}(x) = \int_0^x \mathbf{A}_x \vec{B}^{(i-1)}(x') dx' + \vec{B}^{(i)}(x = 0) \]
Differential equations

• Let us expand the matrix $A_x$ in the line parameter. Then we have:

$$A_x = x^r \left[ \sum_{p=0}^{n} c_p x^p + \mathcal{O}(x^{n+1}) \right]$$

• Using integration-by-parts, we find can write for each rational $m$ and integer $n$:

$$\int x^m \log(x)^n = x^{m+1} \sum_{j=1}^{n} c_j \log(x)^j$$

• Thus, we may perform all the integrations in terms of (generalized) series expansions

$$B_j^{(k)}(x) = x^r \sum_{n=0}^{\infty} \sum_{m=0}^{k} c_{mn} x^n \log(x)^m, \quad c_{mn} \in \mathbb{C}, \quad 0 \geq r \in \mathbb{Q}$$

• Although each series solution has a limited range of convergence, we may concatenate such solutions to reach any point in phase-space.
More generally, consider an unsimplified or partially simplified basis $\tilde{f}$, satisfying:

$$\frac{\partial}{\partial x} \tilde{f}(x, \epsilon) = A_x(x, \epsilon) \tilde{f}'(x, \epsilon)$$

We will assume that $A_x$ is finite as $\epsilon$ goes to zero, which gives

$$\partial_x \tilde{f}^{(k)} = A_x^{(0)} \tilde{f}^{(k)} + \sum_{j=0}^{k-1} A_x^{(k-j)} f^{(j)}$$

This can typically be achieved by rescalings of the form:

$$f_i \rightarrow \epsilon^{\rho_i} f_i, \quad \rho_i \in \mathbb{Z}$$

Lastly, upon ordering the integrals sector-wise, we obtain a "block-triangular" form:

$$A_x^{(0)} \sim \begin{array}{c}
\end{array}$$

, which allows us to decompose into differential equations of the form:

$$\partial_x \tilde{g} = M \tilde{g} + \tilde{b}$$

See e.g.:

[Moriello, '19],
[MH, '20]
DiffExp

• DiffExp is a Mathematica package for solving linear systems of differential equations in terms of one-dimensional series expansions.

• Capable of computing “coupled” systems of more than two integrals

• Takes in (any) system of differential equations of the form

\[ \frac{\partial}{\partial s_i} \vec{f}(\{s_j\}, \epsilon) = A_{s_i} \vec{f}(\{s_j\}, \epsilon) \quad A_{s_i}(\{s_j\}, \epsilon) = \sum_{k=0}^{\infty} A_{s_i}^{(k)}(\{s_j\})\epsilon^k \]

• Uses: compute Feynman integrals numerically at high precision. Analytically continue results across thresholds. Transporting boundary conditions from one special point to another.
DiffExp

• Typical usage of the package:

  • Set configuration options using the method \texttt{LoadConfiguration[opts_]}.
  
  • Prepare a list of boundary conditions using \texttt{PrepareBoundaryConditions[bc\_s\_, \ line\_]}.
  
  • Then we can find series solutions along a line using the function:
    \begin{verbatim}
    IntegrateSystem[bc\_s\_prepared\_, \ line\_]
    \end{verbatim}
  
  • Or one can transport the boundary conditions to a new point using:
    \begin{verbatim}
    TransportTo[bc\_s\_prepared\_, \ point\_]
    \end{verbatim}
Example: 3-loop banana graph

• Load DiffExp:

```
Get[FileNameJoin[{NotebookDirectory[],"..","DiffExp.m"}]];
Loading DiffExp version 1.0.7
For questions, email: martijn.hidding@physics.uu.se
For the latest version, see: https://gitlab.com/hiddingm/diffexp
```

• Set the configuration options and load the matrices

```
EqualMassConfiguration = {
    DeltaPrescriptions -> \{t - 16 + i \delta\},
    MatrixDirectory -> NotebookDirectory[] <> "Banana_EqualMass_Matrices/",
    UseMobius -> True, UsePade -> True
};

LoadConfiguration[EqualMassConfiguration];
DiffExp: Loading matrices.
DiffExp: Found files: \{dt_0.m, dt_1.m, dt_2.m, dt_3.m, dt_4.m\}
DiffExp: Kinematic invariants and masses: \{t\}
DiffExp: Getting irreducible factors..
DiffExp: Configuration updated.
```

Equal-mass case:

\[
\begin{align*}
\bar{B}^{\text{banana}} &= \left( \epsilon J_{2211}^{\text{banana}}, \epsilon(1 + 3\epsilon)I_{2111}^{\text{banana}}, \\
&\quad \epsilon(1 + 3\epsilon)(1 + 4\epsilon)I_{1111}^{\text{banana}}, \epsilon^3 I_{1110}^{\text{banana}} \right) \\
I_{a_1a_2a_3a_4}^{\text{banana}} &= \left( \frac{\alpha_s^3}{4\pi^2/2} \right)^{3/2} (m^2) (2^{3} - 2\epsilon) \left( \prod_{i=1}^{4} \int dt_i \right) D_1^{-a_1} D_2^{-a_2} D_3^{-a_3} D_4^{-a_4} \\
D_1 &= -k_1^2 + m^2, \quad D_2 = -k_2^2 + m^2, \\
D_3 &= -k_3^2 + m^2, \quad D_4 = -(k_1 + k_2 + k_3 + p_1)^2 + m^2
\end{align*}
\]

Figure 1: The three-loop unequal mass banana diagram.
3-loop banana graph

- Prepare the boundary conditions along an asymptotic limit:

\[
\text{EqualMassBoundaryConditions} = \left\{ \\
\quad "?", \\
\quad "?", \\
\quad \varepsilon \left(1 + 3 \varepsilon\right) \left(1 + 4 \varepsilon\right) \left(\varepsilon e^3 \text{EulerGamma} e \frac{\text{Gamma}[\varepsilon]^3}{t} - 4 e^3 \text{EulerGamma} e \frac{\text{Gamma}[\varepsilon]^3}{t} \right) \\
\quad + 6 e^3 \text{EulerGamma} e \left(-\frac{1}{t}\right)^{1+\varepsilon} \varepsilon \text{Gamma}[-\varepsilon]^2 \text{Gamma}[\varepsilon]^3 \\
\quad + 8 e^3 \text{EulerGamma} e \left(-\frac{1}{t}\right)^{1+2\varepsilon} \varepsilon \text{Gamma}[-\varepsilon]^3 \text{Gamma}[\varepsilon]^3 \text{Gamma}[2\varepsilon] \\
\quad + 3 e^3 \text{EulerGamma} e \left(-\frac{1}{t}\right)^{1+3\varepsilon} \varepsilon \text{Gamma}[-\varepsilon]^4 \text{Gamma}[3\varepsilon] \\
\quad + e^3 \text{EulerGamma} e \varepsilon^3 \text{Gamma}[\varepsilon]^3 \\
\right\} \\
\text{// PrepareBoundaryConditions[#, <|t| \to -1/x|>] \&;}
\]

DiffExp: Integral 1: Ignoring boundary conditions.
DiffExp: Integral 2: Ignoring boundary conditions.
DiffExp: Assuming that integral 3 is exactly zero at epsilon order 0.
DiffExp: Prepared boundary conditions in asymptotic limit, of the form:

\[
\begin{array}{cccccc}
0[x] & \ldots x + 0[x]^{3/2} & \ldots x + 0[x]^{3/2} & \ldots x + 0[x]^{3/2} & \ldots x + 0[x]^{3/2} \\
\ldots + \sqrt{0[x]} & \sqrt{0[x]} & \ldots + \sqrt{0[x]} & \ldots + \sqrt{0[x]} & \ldots + \sqrt{0[x]}
\end{array}
\]
Next, we transport the boundary conditions:

\[
\text{Transport1} = \text{TransportTo}[\text{EqualMassBoundaryConditions}, \langle |t \to -1| \rangle]; \\
\text{Transport2} = \text{TransportTo}[\text{Transport1}, \langle |t \to x| \rangle, 32, \text{True}];
\]

\[
\text{DiffExp}: \text{Transporting boundary conditions along } \langle \left| t \to \frac{1}{x} \right| \rangle \text{ from } x = 0 \text{ to } x = 1.
\]

\[
\text{DiffExp}: \text{Preparing partial derivative matrices along current line.}
\]

\[
\text{DiffExp}: \text{Determining positions of singularities and branch cuts.}
\]

\[
\text{DiffExp}: \text{Possible singularities along line at positions } \{0\}.
\]

\[
\text{DiffExp}: \text{Analyzing integration segments.}
\]

\[
\text{DiffExp}: \text{Segments to integrate: 3.}
\]

\[
\text{DiffExp}: \text{Integrating segment: } \langle \left| t \to \frac{8 \cdot (-1 + 1 \cdot x)}{x} \right| \rangle.
\]

\[
\text{DiffExp}: \text{Integrated segment 1 out of 3 in 20.8565 seconds.}
\]

\[
\text{DiffExp}: \text{Evaluating at } x = 0.0625
\]

\[
\text{DiffExp}: \text{Current segment error estimate: } 5.14483 \times 10^{-31}
\]

\[
\text{DiffExp}: \text{Total error estimate: } 5.14483 \times 10^{-31}
\]

\[
\text{DiffExp}: \text{Integrating segment: } \langle |t \to -1 + 1 \cdot x| \rangle
\]
Lastly, we plot the result:

```math
ResultsForPlotting = ToPiecewise[Transport2];
Quiet[ReImPlot[
{ResultsForPlotting[[3, 4]] [x], ResultsForPlotting[[3, 5]] [x]}, {x, 0, 32},
    ClippingStyle -> Red, PlotLegends -> {"B_3^{(3)}", "B_3^{(4)}"}, AxesLabel -> {"p^2/m^2"}, PlotRange -> {-700, 850},
    MaxRecursion -> 15, WorkingPrecision -> 100]]
```
3-loop banana graph

• Timing:
  • Moving from $p^2 = -\infty$ to $p^2 = 30$ at a precision of 25 digits takes about 90 sec, where we computed the top sector integrals up to and including order $\epsilon^3$.
  • Moving from $p^2 = -\infty$ to $p^2 = 30$ at a precision of 100 digits takes a bit under 20 min, where we computed the top sector integrals up to and including order $\epsilon^3$.
  • Obtaining 100+ digits at $p^2 = -100$ up to and including order $\epsilon^3$ takes about 2.5 min.

• $B_3^{(k)}$:

$$
\begin{align*}
\theta &= 4.08241320270405960780199146104509733985550125377422243449656379831484823907330199489603248642178129 \\
&\quad - 0.7713150915227857546258559692543676298350939151980774607908277236769934490973612004866036340787026038 \\
&\quad - 15.52268532416518855576696548019433617730937578226039207428302008586262767404183548619606743796239099 \\
&\quad + 78.12509728148001692986790482079302619114776011817121195506011258285334682242128391076363566162968586
\end{align*}
$$
3-Loop banana graph

- We may also compute the fully unequal mass case. We choose the basis:

\[
\mathcal{B}_{\text{banana}} = \left\{ \varepsilon I_{1122}^{\text{banana}}, \varepsilon I_{1212}^{\text{banana}}, \varepsilon I_{1221}^{\text{banana}}, \varepsilon I_{2112}^{\text{banana}}, \varepsilon I_{2121}^{\text{banana}}, \varepsilon I_{2211}^{\text{banana}}, \\
(1 + 3\varepsilon) I_{1112}^{\text{banana}}, (1 + 3\varepsilon) I_{1211}^{\text{banana}}, (1 + 3\varepsilon) I_{2111}^{\text{banana}}, \\
(1 + 3\varepsilon) I_{2211}^{\text{banana}}, (1 + 3\varepsilon)(1 + 4\varepsilon) I_{1111}^{\text{banana}}, \\
3 I_{0111}^{\text{banana}}, 3 I_{1011}^{\text{banana}}, 3 I_{1101}^{\text{banana}}, 3 I_{1110}^{\text{banana}} \right\}
\]

- We provide 55 digits of basis integral \( B_{11} \) below, in the point

\[(p^2 = 50, m_1^2 = 2, m_2^2 = 3/2, m_3^2 = 4/3, m_4^2 = 1)\]

\[
\begin{align*}
B_{11}^{(0)} &= 0 \\
B_{11}^{(1)} &= 5.1972521136965043170129578538563652405618939122389078645 \\
&\quad + i\, 6.8755169535390207501370685645538902299559024551830956594 \\
B_{11}^{(2)} &= -17.95801081120940608995233616989284789878068705389997533 \\
&\quad + i\, 31.743670363369309090840293229901197191350895649494231047 \\
B_{11}^{(3)} &= -121.5101152068177565203392807541216084962880772908306370668 \\
&\quad - i\, 40.769076236020276645377599999171712226537428258529145754746 \\
B_{11}^{(4)} &= 125.6113380236055347545937640047989958323116836268125703923 \\
&\quad - i\, 229.9200257172388589952062757571215176834471783495112755027
\end{align*}
\]

These results were obtained in about 20 minutes on a single CPU-core.
Further automatization

• In the previous example, the boundary conditions were provided as closed-form expressions in $\epsilon$. In general, this requires a manual case-by-case analysis using expansion by regions in the parametric representation.

• Furthermore, the basis was chosen such that the differential equations are finite (and also in precanonical form $A_0 + \epsilon A_1$).

• More generally, we would like to derive the basis, differential equations and boundary terms in an automated way.

[See works by Beneke and Smirnov] & [Jantzen, Smirnov, Smirnov, 1206.0546] for the asy.m package
An automated computational strategy

- Find a basis of (quasi-)finite Feynman integrals.
- Derive a closed linear system of differential equations for the basis.
- Rescale integrals by powers of \( \epsilon \) to make the differential equations finite in \( \epsilon \).
- Compute boundary conditions in a Euclidean point by numerical integration.
- Obtain points in the physical region (and analytically continue) by numerically solving the differential equations using iterated series expansions.
- (Optional) upgrade the boundary conditions to a higher precision by analyzing behavior near thresholds and pseudo-thresholds.
Caesar package

- Together with J. Usovitsch, I am working on a Mathematica toolbox, Caesar, which automates all steps. It works by interfacing with various programs that are already on the market.

  A finite basis is derived in an automated fashion by using Reduze to obtain candidate integrals

  and using Kira to select an independent set.

- The differential equations are computed using inbuilt code, while the dimensional reduction relations are generated using LiteRed.

- pySecDec is used to obtain numerical boundary conditions in the Euclidean region.

- DiffExp is used to obtain results everywhere else.
Application: 3-loop vertex topology (relevant for EW pseudo-observables at Z-boson resonance)

In collaboration with: [Ievgen Dubovyk, Ayres Freitas, Janusz Gluza, Krzysztof Grzanka, MH, Johann Usovitsch]

- We consider the 3-loop topology pictured below:

\[
p_1 \quad m_W \quad m_t \quad p_3 \quad \rightarrow \quad m_W \quad m_t \quad p_3
\]

in the kinematic configuration: \( p_1^2 = 0, \ p_2^2 = 0, \ p_1 \cdot p_2 = s/2 \). We choose the following propagators:

\[
\begin{align*}
D_1 &= m_W^2 - k_3^2 \\
D_5 &= -(k_2 - p_1 - p_2)^2 \\
D_9 &= -(k_2 - k_1)^2 \\
D_2 &= -k_2^2 \\
D_6 &= m_W^2 - (k_3 - p_1 - p_2)^2 \\
N_{10} &= -(k_1 - k_3)^2 \\
D_3 &= -k_1^2 \\
D_7 &= -(k_3 - p_1)^2 \\
N_{11} &= -(k_1 - p_2)^2 \\
D_4 &= -(k_1 - p_1 - p_2)^2 \\
D_8 &= m_t^2 - (k_3 - k_2)^2 \\
N_{12} &= -(k_2 - p_2)^2
\end{align*}
\]

- After IBP-reduction, the top sector collapses. The highest sectors remaining after IBP reduction have 8 propagators and are pictured in the top-right.
Example: 3-loop topology

- The (finite) basis consists of 77 integrals in total. We choose 19 integrals in \( d = 4 \), 53 integrals in \( d = 6 \), and 5 integrals in \( d = 8 \).

- We rescale the integrals by powers of \( \epsilon \) in order to make the differential equations finite as \( \epsilon \to 0 \). The largest power we rescale by is \( \epsilon^{-5} \).

- We set up the system of differential equations, making use of IBP identities and dimensional recurrence relations. The differential equations are \( \sim 12 \) MB before expanding in \( \epsilon \).
**Basis integrals**

<table>
<thead>
<tr>
<th>$d=6-2\epsilon$</th>
<th>$d=6-2\epsilon$</th>
<th>$d=6-2\epsilon$</th>
<th>$d=6-2\epsilon$</th>
<th>$d=6-2\epsilon$</th>
<th>$d=6-2\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2} \epsilon$</td>
<td>$\frac{1}{2} \epsilon$</td>
<td>$\frac{1}{2} \epsilon$</td>
<td>$\frac{1}{2} \epsilon$</td>
<td>$\frac{1}{2} \epsilon$</td>
<td>$\frac{1}{2} \epsilon$</td>
</tr>
<tr>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
<td>$1.2,2,2,2,0,1,1,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{3} \epsilon$</td>
<td>$\frac{1}{3} \epsilon$</td>
<td>$\frac{1}{3} \epsilon$</td>
<td>$\frac{1}{3} \epsilon$</td>
<td>$\frac{1}{3} \epsilon$</td>
<td>$\frac{1}{3} \epsilon$</td>
</tr>
<tr>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
<td>$2,2,2,1,1,1,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{4} \epsilon$</td>
<td>$\frac{1}{4} \epsilon$</td>
<td>$\frac{1}{4} \epsilon$</td>
<td>$\frac{1}{4} \epsilon$</td>
<td>$\frac{1}{4} \epsilon$</td>
<td>$\frac{1}{4} \epsilon$</td>
</tr>
<tr>
<td>$6-2\epsilon$</td>
<td>$6-2\epsilon$</td>
<td>$6-2\epsilon$</td>
<td>$6-2\epsilon$</td>
<td>$6-2\epsilon$</td>
<td>$6-2\epsilon$</td>
</tr>
<tr>
<td>$\frac{1}{5} \epsilon$</td>
<td>$\frac{1}{5} \epsilon$</td>
<td>$\frac{1}{5} \epsilon$</td>
<td>$\frac{1}{5} \epsilon$</td>
<td>$\frac{1}{5} \epsilon$</td>
<td>$\frac{1}{5} \epsilon$</td>
</tr>
<tr>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
<td>$3,0,2,2,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{6} \epsilon$</td>
<td>$\frac{1}{6} \epsilon$</td>
<td>$\frac{1}{6} \epsilon$</td>
<td>$\frac{1}{6} \epsilon$</td>
<td>$\frac{1}{6} \epsilon$</td>
<td>$\frac{1}{6} \epsilon$</td>
</tr>
<tr>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$5,0,2,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{7} \epsilon$</td>
<td>$\frac{1}{7} \epsilon$</td>
<td>$\frac{1}{7} \epsilon$</td>
<td>$\frac{1}{7} \epsilon$</td>
<td>$\frac{1}{7} \epsilon$</td>
<td>$\frac{1}{7} \epsilon$</td>
</tr>
<tr>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$2,0,2,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{8} \epsilon$</td>
<td>$\frac{1}{8} \epsilon$</td>
<td>$\frac{1}{8} \epsilon$</td>
<td>$\frac{1}{8} \epsilon$</td>
<td>$\frac{1}{8} \epsilon$</td>
<td>$\frac{1}{8} \epsilon$</td>
</tr>
<tr>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$4,0,0,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{9} \epsilon$</td>
<td>$\frac{1}{9} \epsilon$</td>
<td>$\frac{1}{9} \epsilon$</td>
<td>$\frac{1}{9} \epsilon$</td>
<td>$\frac{1}{9} \epsilon$</td>
<td>$\frac{1}{9} \epsilon$</td>
</tr>
<tr>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
<td>$0,0,0,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{10} \epsilon$</td>
<td>$\frac{1}{10} \epsilon$</td>
<td>$\frac{1}{10} \epsilon$</td>
<td>$\frac{1}{10} \epsilon$</td>
<td>$\frac{1}{10} \epsilon$</td>
<td>$\frac{1}{10} \epsilon$</td>
</tr>
<tr>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,2,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{11} \epsilon$</td>
<td>$\frac{1}{11} \epsilon$</td>
<td>$\frac{1}{11} \epsilon$</td>
<td>$\frac{1}{11} \epsilon$</td>
<td>$\frac{1}{11} \epsilon$</td>
<td>$\frac{1}{11} \epsilon$</td>
</tr>
<tr>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
</tr>
<tr>
<td>$\frac{1}{12} \epsilon$</td>
<td>$\frac{1}{12} \epsilon$</td>
<td>$\frac{1}{12} \epsilon$</td>
<td>$\frac{1}{12} \epsilon$</td>
<td>$\frac{1}{12} \epsilon$</td>
<td>$\frac{1}{12} \epsilon$</td>
</tr>
<tr>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
<td>$1,0,1,0,0,0,0,0,0,0,0,0$</td>
</tr>
</tbody>
</table>
Numerical boundary conditions using pySecDec

- When all basis integrals are finite, their numerical integration using pySecDec is sped up considerably.
- We compute all basis integrals in the Euclidean region in the point $s = -2, m_W^2 = 4, m_t^2 = 16$, using the Qmc integrator configured with:

```python
lib.use_Qmc(minn=10**7, maxeval=10**9, transform='korobov3', epsabs=1e-12, cputhreads=16)
```

- The computation took between 1/2-1 day on a Ryzen Threadripper Pro 3955WX.
- We find for example: $I_{1,1,1,1,0,1,1,1} = 0.133952666651743990 - 0.13899149646580500 \epsilon + O(\epsilon^2)$

$$\pm \left( 2 \times 10^{-10} + 7 \times 10^{-10} \epsilon \right)$$
Results in the physical region, using DiffExp

- Using DiffExp we may transport from the Euclidean point to any other (real) point in phase-space.

- Transporting from \((s, m_W^2, m_t^2) = (-2, 4, 16)\) to \((s, m_W^2, m_t^2) = \left(1, \left(\frac{401925}{455938}\right)^2, \left(\frac{433000}{227969}\right)^2\right)\), we obtain:

\[
\begin{align*}
I^{d=6-2\epsilon}_{1,1,0,2,0,1,1,2,2,0,0,0} &= (0.125019 + 0.0127438 i) - (0.334035 - 0.0731341 i) \epsilon + (1.81433 + 0.208055 i) \epsilon^2 - (6.08263 - 0.389921 i) \epsilon^3 + O(\epsilon^4) \\
I^{d=4-2\epsilon}_{1,0,1,1,0,1,1,2,1,0,0,0} &= (1.17171 + 1.03298 i) - (3.13434 - 1.43328 i) \epsilon + (5.9312 + 3.04346 i) \epsilon^2 + O(\epsilon^3) \\
I^{d=4-2\epsilon}_{2,0,1,1,0,1,1,2,1,0,0,0} &= (0.912403 + 0.837335 i) - (1.66844 - 1.83869 i) \epsilon + (2.25671 + 3.31779 i) \epsilon^2 + O(\epsilon^3) \\
I^{d=4-2\epsilon}_{1,0,1,1,0,1,1,3,1,0,0,0} &= (0.102616 + 0.123891 i) - (0.137177 - 0.313638 i) \epsilon - (0.0575107 - 0.560502 i) \epsilon^2 + O(\epsilon^3) \\
I^{d=4-2\epsilon}_{1,1,1,0,1,1,1,1,1,0,0,0} &= (1.30731 + 3.42323 i) - (10.0551 - 8.533 i) \epsilon + O(\epsilon^2)
\end{align*}
\]

- The computation involved 16 line segments and took 45 minutes on a single CPU core. The precision of the expansions was \(10^{-17}\), exceeding the precision of the boundary conditions.
Results in the physical region, using DiffExp

• We find that the numerical error of the boundary conditions approximately carries over after transporting from the Euclidean to the physical point.

• For example, at \((s, m_W^2, m_t^2) = (-2, 4, 16)\) we have:

\[
I_{1,1,1,0,1,1,1,0,0,0}^{d=4-2\epsilon} = +0.133952666651744 \pm 2 \times 10^{-10}
\]

• While at \((s, m_W^2, m_t^2) = \left(1, \left(\frac{401925}{455938}\right)^2, \left(\frac{433000}{227969}\right)^2\right)\) we have:

\[
I_{1,1,1,0,1,1,1,0,0,0}^{d=4-2\epsilon} = \left(1.30730596404577 + 3.42322623988039i\right) \pm \left(3 \times 10^{-11} + 2 \times 10^{-9}i\right)
\]
Results in the physical region, using DiffExp

- By concatenating series expansions along line segments, we can plot the results along a line. For example:

- It took about 2 hour and 15 minutes to obtain the results along this line, at a precision of $\sim 10^{-13}$.

- Afterwards, evaluating an integral anywhere along the line takes about 0.01 seconds.
Optional: upgrading the boundary conditions

- Suppose we want to go beyond the precision that pySecDec can provide in the Euclidean region. It turns out that we can lift the boundary conditions to a higher precision by looking at the scaling of the integrals near (pseudo-)thresholds.

- We don’t have to use expansion by regions. Instead, we take the numerical boundary conditions, move around in phase-space and record at which locations there are branch-points or singularities.

- In particular, for each line segment we record presence or absence of terms of the form of $x^{-n}$, $x^{-n/2}$ and $\log(x)^m$, where we let $n \leq 0$.

- Because the boundary conditions are of finite precision, such terms may carry coefficients of the form $10^{-10}$ which we will interpret to be 0 exactly.
Optional: upgrading the boundary conditions

- We get a feeling for which directions to move towards, by looking at the poles of the differential equations. The differential equations have the following poles:

\[
\begin{align*}
mt, & \quad mw, \\
mt + s, & \quad mt - mw, \\
5* mw - 2*s, & \quad 2* mw - s, \\
mt - mw - 3*s, & \quad 2* mt - 4* mw + s, \\
mt^2 - 2* mt* mw + mw^2 - 2* mt* s - 2* mw* s + s^2, & \quad 2* mt* mw - 4* mw^2 + mt* s + 6* mw* s - 2* s^2, \\
2* mt^4 * mw - 8* mt^3 * mw^2 + 12* mt^2 * mw^3 - 8* mt * mw^4 + 2* mw^5 - mt^4 * s + 5* mt^3 * mw * s - 9* mt^2 * mw^2 * s + 7* mt * mw^3 * s - 2* mw^4 * s - mt^2 * mw^2 * s^2 - 3* mt * mw^2 * s^2 + mt^2 * s^3
\end{align*}
\]

- For example, with \( m_t^2 = 16 \), we obtain the following contour plot:
- The green dots represents points between which we transport. In particular, we consider lines from the Euclidean point \((s, m_W^2, m_t^2) = (-2,4,16)\), towards the outer green points. The points have been chosen in order to cross as many of the poles as possible.
Optional: upgrading the boundary conditions

- Adding two additional points that cross $m_t = 0$ as well, we end up with the following 8 points to which we transport from $(s, m_w^2, m_t^2) = (-2, 4, 16)$:

\[
\begin{align*}
\{s \rightarrow -29, m_w^2 \rightarrow 50, m_t^2 \rightarrow 17\} & \quad \{s \rightarrow -90, m_w^2 \rightarrow -50, m_t^2 \rightarrow 17\} \\
\{s \rightarrow 25, m_w^2 \rightarrow 65, m_t^2 \rightarrow 17\} & \quad \{s \rightarrow 90, m_w^2 \rightarrow 20, m_t^2 \rightarrow 17\} \\
\{s \rightarrow 37, m_w^2 \rightarrow -30, m_t^2 \rightarrow 17\} & \quad \{s \rightarrow -26, m_w^2 \rightarrow -18, m_t^2 \rightarrow -10\} \\
\{s \rightarrow -22, m_w^2 \rightarrow 20, m_t^2 \rightarrow -10\} & \quad \{s \rightarrow 70, m_w^2 \rightarrow 40, m_t^2 \rightarrow -10\}
\end{align*}
\]
Optional: upgrading the boundary conditions

Next, we repeat the computation with a set of unfixed boundary conditions:

Lastly, we impose the same behavior around the singular points, which fixes the coefficients:
Optional: upgrading the boundary conditions

• We see that order $\epsilon^6$ has not been fully determined, and we would need to expand up to order $\epsilon^7$ in order to fully fix this order.

• Furthermore, we manually added high precision results for the basis integrals 1, 4, 23 and 26:

$$
\left( \frac{1}{\epsilon^2} \right) I_{4,2,2,2,2,0,0,0,0,0,0}^{d=6-2\epsilon}, \left( \frac{1}{\epsilon^4} \right) I_{4,0,2,2,0,0,0,4,0,0,0,0}^{d=6-2\epsilon}, \left( \frac{1}{\epsilon^4} \right) I_{5,3,0,3,0,0,0,0,3,0,0,0}^{d=8-2\epsilon}, \left( \frac{1}{\epsilon^5} \right) I_{3,0,2,0,0,0,0,3,2,0,0,0}^{d=6-2\epsilon}
$$

which were obtained by integrating the Feynman parametrization analytically.

• We performed the lifting procedure twice by transporting along different lines, in order to check consistency of the results. We obtain the following (preliminary) results at $(s, m_W^2, m_t^2) = (-2, 4, 16)$:

$$
I_{1,1,1,1,0,1,1,1,1,0,0,0}^{d=4} = 0.133952666444160183902749812
$$

at an expected precision of about $10^{-25}$. 


Conclusions

• Without spending significant effort on simplification of the basis, we can numerically solve the differential equations of non-trivial 3-loop Feynman integrals.

• By choosing the basis representatives to be finite integrals, we can obtain precise numerical boundary conditions in the Euclidean region using pySecDec.

• We find that the precision of the boundary conditions in the Euclidean region carries over to the physical region.

• We can upgrade the boundary conditions to a higher precision by reading of the scaling behavior of the integrals around singular points.

• The process can be almost fully automated.
Thank you for listening!