Calculating loop amplitudes on a computer

Do-It-Yourself guide

Vitaly Magerya

Institut für Theoretische Physik,
Karlsruher Institut für Technologie
(ITP KIT)

Hamburg, CAPP 2023

Goals

Goal of the lecture:

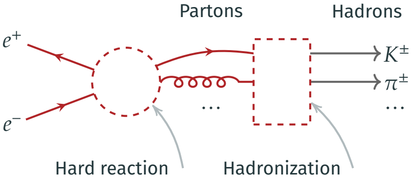
- choose a *multi-loop amplitude*,
- *calculate it* completely on a computer (analytically and numerically),
- *use our own code* (in MATHEMATICA, FORM, with other useful software).

* * *

Alternatives to writing own code (not covered here):

- using existing libraries for one- and multi-loop amplitudes, such as FEYNCALC, FEYNARTS+FORMCALC, ALIBRARY, Q2E, HEPLIB, etc;
- using aumated library generators for 1-loop amplitudes, such as GOSAM, NJET, OPENLOOPS, RECOLA, etc;
- using complete automated packages for event generation, such as MADGRAPH, HELAC, WHIZARD, etc.

What will we calculate?



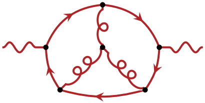
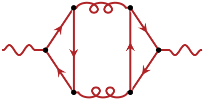
Target quantity: the *total cross-section of e^+e^- annihilation to hadrons*,

$$\sigma(e^+e^- \to \text{hadrons}) = \sigma(e^+e^- \to \text{partons}) = \frac{|M(e^+e^- \to \text{partons})|^2}{4\sqrt{p_{e^-} \cdot p_{e^+}}}.$$

Goal: calculate $\mathcal{O}(\alpha_s^2)$ corrections to it (and then $\mathcal{O}(\alpha_s^3)$ too).
Model: QCD with N_f *massless quarks*, and N_t *massive quarks* of mass m_t.

* * *

As we'll see this translates to calculating diagrams like

 ,  , etc.

Motivation: the R ratio

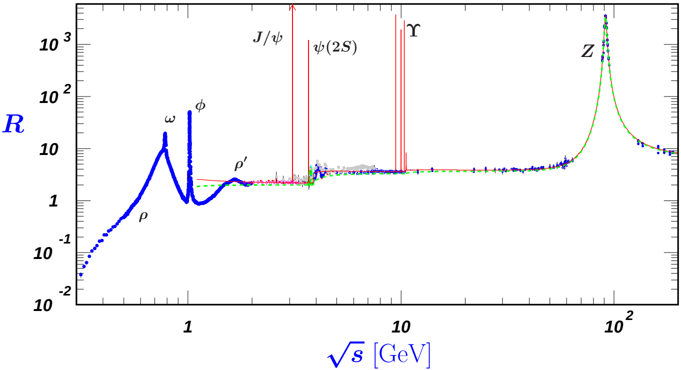


Figure: $R \equiv \sigma(e^+e^- \to \text{hadrons})/\sigma_{\text{leading}}(e^+e^- \to \gamma^* \to \text{muons})$, from Ezhela, Lugovsky, Zenin '03, available at pdg.lbl.gov/2023/hadronic-xsections.

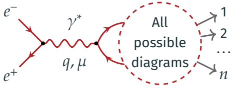
At the leading order R is proportional to the number of QCD colors N_c:

$$R = N_c \sum_{\text{quarks}} e_q \quad \Rightarrow \quad N_c \text{ can be measured via } R.$$

Calculating e^+e^- annihilation to hadrons

The *matrix element squared* of the total cross-section is

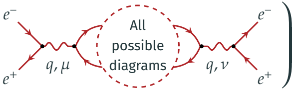
$$|M(e^+e^- \to \text{partons})|^2 = \sum_n \int \mathrm{dPS}_n \left| \sum \right.$$



$$\left. \right|^2 .$$

Often calculating it is easier via the *optical theorem*:

$$|M|^2 = 2\,\mathrm{Re}\Bigg(\Bigg)$$

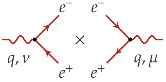


.

Further, we can factorize this into *leptonic and hadronic tensors* (L and H):

$$|M|^2 = -\frac{2}{q^4}\,\mathrm{Re}\Big(L_{\mu\nu}H^{\mu\nu}\Big)\,, \quad q \equiv p_{e^-} + p_{e^+}\,,$$

$$L_{\mu\nu} \equiv$$



$$, \quad H^{\mu\nu} \equiv$$

.

All the loop integration and α_s corrections are in $H^{\mu\nu}$, and $L_{\mu\nu}$ is just

$$L^{\mu\nu} = 4\pi\alpha\left(p^\mu_{e^-}p^\nu_{e^+} + p^\nu_{e^-}p^\mu_{e^+} - g^{\mu\nu}\, p_{e^-}\cdot p_{e^+}\right).$$

Tensor structures and projectors

Because $H^{\mu\nu}(q)$ is a *Lorentz-covariant tensor*, it can only be composed from $g^{\mu\nu}$, q^{μ}, and q^{ν}. Its general structure then must be:

$$H^{\mu\nu}(q) = q^{\mu}q^{\nu}\, F_1(q) + g^{\mu\nu}\, F_2(q).$$

This structure can be further restricted via *Ward identities*:

$$q_{\mu}H^{\mu\nu} = H^{\mu\nu}q_{\nu} = 0 \quad \Rightarrow \quad q^2 F_1 + F_2 = 0,$$

$$H^{\mu\nu}(q) = F_1(q)\, \left(q^{\mu}q^{\nu} - g^{\mu\nu}\, q^2\right).$$

With this form of $H^{\mu\nu}$ the total matrix element squared becomes

$$|M|^2 = -\frac{2}{q^4}\,\mathrm{Re}\left(L_{\mu\nu}H^{\mu\nu}\right) = 4\pi\alpha(2-d)\,\mathrm{Re}\left(F_1(q)\right).$$

To get F_1 from $H^{\mu\nu}$ we must invert the relation by constructing a *projector*:

$$F_1 = P_{\mu\nu}H^{\mu\nu}, \qquad P_{\mu\nu} \equiv \frac{g_{\mu\nu}}{2-d}\frac{1}{q^2}.$$

Calculting scalars like F_1 is simpler than tensors like $H^{\mu\nu}$, and a tensor decomposition along with projector constuction is almost always needed.

Calculation plan

1. Generate *Feynman diagrams* for the process.
   - $F_1 = P$  $+ P$  $+ P$  $+ \ldots$
   - QGRAF with MATHEMATICA output.
2. Apply *Feynman rules*.
   - $F_1 = \int \mathrm{d}^d l_1 \, \mathrm{Tr}\left(\gamma^\mu \not{k}_1 \gamma^\nu \not{k}_2\right) \cdots + \ldots$
   - Custom MATHEMATICA code.
3. Resolve Dirac and color *tensor summation*, convert to the scalar integral families.
   - $F_1 = N_f C_a \cdots I_{123} + \ldots$
   - MATHEMATICA $\longrightarrow$ FORM $\longrightarrow$ MATHEMATICA.
4. Use *IBP relations* to reduce to smaller set of “master integrals”.
   - $F_1 = \left(N_f C_a \cdots + \ldots\right) I_{111} + \ldots$
   - MATHEMATICA $\longrightarrow$ KIRA $\longrightarrow$ MATHEMATICA.
5. Evaluate the master integrals.
   - Numerically: sector decomposition with pySECDEC.
   - Analytically or semi-analytically: differential equations.

General idea: *use MATHEMATICA to glue everything together*.

Diagram generation

Feynman diagram generation with QGRAF

QGRAF is a widely used program for Feynman diagram generation available at `http://cfif.ist.utl.pt/~paulo/qgraf.html`.

To generate diagrams with QGRAF:

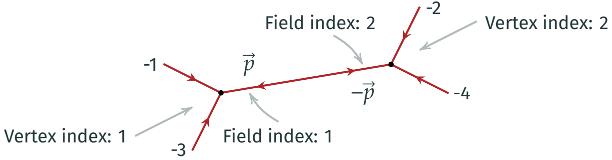
1. Create a QGRAF *model file* with a list of fields and vertices.
   See `qgraf-modfile` for the QCD model we'll use.
2. Create a QGRAF *style file* (output template) defining the output format.
   See `qgraf-stylefile` for the Mathematica output style we will use.
3. Create `qgraf.dat`, defining the incoming particles, outgoing particles, loop count, names of the momenta, the model file, and the style file.
   See `qgraf.dat.example`.
4. Run `qgraf` from the directory where `qgraf.dat` is.
   Note: `qgraf.dat` name is hardcoded (can only be changed since QGRAF 3.6.6 with some restrictions). We'll work around this.

Demo: run `qgraf` manualy; run `example.generate-diagrams.m` to generate the diagrams automatically; run `show-diagrams.m` to view them.

QGRAF result structure

Once QGRAF has generated the list of diagrams, each diagram will have:

- A list of *incoming fields* (legs), and a list of *outgoing fields*.
  - Each field has: a field name, a field index, a vertex id, momentum.
- A list of *propagators*.
  - Each propagator has: a field name, two field indices (start and end), two vertex indices (start and end), momentum.
- A list of *vertices*.
  - Each vertex has: a vertex index, a list of *rays*.
    - Each ray has: a field name, a field index, momentum.



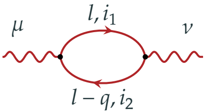
Notes:

- Fields and momenta are always listed as if incoming into the vertex.
- Vertices and internal legs have positive indices.
- External legs have negative indices: -1, -3, -5, ... for incoming particles, -2, -4, -6, ... for outgoing.

Feynman rules

Notation on a computer

Feynman rules *in a book*:



$=$

$$\int \frac{\mathrm{d}^d l}{(2\pi)^d}\, i g_e Q_{f_1} \delta_{i_1 i_2} \delta_{f_1 f_2}\, \mathrm{Tr}\left(\gamma^\mu \frac{\not{l} - \not{q}}{\left(q - l\right)^2 + i0} \gamma^\nu \frac{\not{l}}{l^2 + i0} \right) i g_e Q_{f_2} \delta_{i_2 i_1} \delta_{f_2 f_1}.$$

Feynman rules *on a computer*:

```
In[1] := One Feynman diagram, please???

Syntax::sntxf: "One Feynman diagram"
cannot be followed by ", please???".
```

MATHEMATICA notation, I

To operate on Feynman rules in MATHEMATICA, we need to choose a notation. Any will work; we'll use the following.

Index names:

- μ_i, Lorentz indices, $1 \ldots d$: `lor[`i`]` (with i directly from QGRAF);
- i_i, fundamental color indices, $1 \ldots N_c$: `fun[`i`]`;
- a_i, adjoint color indices, $1 \ldots N_a = N_c^2 - 1$: `adj[`i`]`;
- f_i, light quark flavors (up, down, etc), $1 \ldots N_f$: `flv[`i`]`;
- t_i, heavy quark flavors (e.g. top), $1 \ldots N_t$: `flvt[`i`]`;
- s_i, Dirac (spinor) indices, $1 \ldots 4$: `spn[`i`]`.

Tensors:

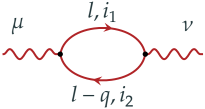
- $f^{a_1 a_2 a_3}$, $SU(N_c)$ structure constants: `colorf[`a_1`,`a_2`,`a_3`]`;
- $T^a_{i_1 i_2}$, $SU(N_c)$ generators, `colorT[`a`,`i_1`,`i_2`]`;
- $\left(\gamma^\mu\right)_{s_1 s_2}$, Dirac matrices: `gammachain[gamma[`$\mu$`],`$s_1$`,`$s_2$`]`;
- $\left(\not{p}\right)_{s_1 s_2}$, Dirac slash notation: `gammachain[slash[p],`s_1`,`s_2`]`.

MATHEMATICA notation, II

- Propagator denominators:
  - Massless: $\frac{1}{p^2+i0} \to$ `den[p]`.
  - Massive: $\frac{1}{p^2-m^2+i0} \to$ `den[p,m2]`.
- Momenta components: $p^\mu \to$ `momentum[p,`μ`]`.
- Scalar products: $p \cdot q \to$ `sp[p,q]`.
- Delta functions (metric tensors):
  - Quark flavor: $\delta_{f_1 f_2} \to$ `deltaf[`f_1`,`f_2`]`.
  - Heavy quark flavor: $\delta_{t_1 t_2} \to$ `deltaft[`t_1`,`t_2`]`.
  - Generic: $\delta_{xy} \to$ `delta[`x`,`y`]`.
- Quark electric charges:
  - Light: $Q_{f_i} \to$ `chargeQ[`f_i`]`.
  - Heavy: $Q_{t_i} \to$ `chargeQt[`t_i`]`.

Everything else (products, sums, powers, etc): the normal MATHEMATICA expressions.

Mathematica notation example



$$=$$

$$\int \frac{\mathrm{d}^d l}{(2\pi)^d}\, i g_e Q_{f_1} \delta_{i_1 i_2} \delta_{f_1 f_2} \,\mathrm{Tr}\left(\gamma^\mu \frac{\not{l} - \not{q}}{\left(q - l\right)^2 + i0} \gamma^\nu \frac{\not{l}}{l^2 + i0} \right) i g_e Q_{f_2} \delta_{i_2 i_1} \delta_{f_2 f_1} =$$

```
(I * ge * chargeQ[flv[1]]
 * delta[fun[1], fun[2]] * deltaf[flv[1], flv[2]]
 * gammachain[gamma[lor[mu]], spn[1], spn[2]]
 * gammachain[slash[l - q], spn[2], spn[3]]
 * den[q - l]
 * gammachain[gamma[lor[nu]], spn[3], spn[4]]
 * gammachain[slash[l], spn[4], spn[1]]
 * den[l]
 * I * ge * chargeQ[flv[2]]
 * delta[fun[2], fun[1]] * deltaf[flv[2], flv[1]])
```

Feynman rules: propagators

Feynman rules for propagators:

- Quark propagator: $i\delta_{f_1 f_2}\delta_{i_1 i_2}\frac{\left(\not{p}\right)_{s_2 s_1}}{p^2}$.
- Heavy quark propagator: $i\delta_{t_1 t_2}\delta_{i_1 i_2}\frac{\left(\not{p}+\mathbb{1}m_t\right)_{s_2 s_1}}{p^2-m^2}$.
- Gluon propagator: $-i\delta_{a_1 a_2}\left(\frac{g_{\mu_1\mu_2}}{p^2}-(\xi-1)\frac{p^{\mu_1}p^{\mu_2}}{\left(p^2\right)^2}\right)$.
- Ghost propagator: $i\delta_{a_1 a_2}\frac{1}{p^2}$.
- Photon propagator: not needed, photons are external for $H^{\mu\nu}$.

Feynman rules: vertices

Feynman rules for vertices:

- Anti-quark/quark/gluon vertex: $i g_s \delta_{f_1 f_2} T^{a_3}_{i_1 i_2} \left(\gamma^{\mu_3}\right)_{s_1 s_2}$.
- Anti-quark/quark/photon vertex: $i g_e Q_{f_1} \delta_{f_1 f_2} \delta_{i_1 i_2} \left(\gamma^{\mu_1}\right)_{s_1 s_2}$.
- Anti-ghost/ghost/gluon vertex: $g_s f^{a_3 a_2 a_1} p_1^{\mu_3}$.
- Three-gluon vertex:

$$g_s f^{a_1 a_2 a_3} \left(g^{\mu_1 \mu_2} \left(p_1 - p_2\right)^{\mu_3} + (123 \to 231) + (123 \to 312) \right).$$

- Four-gluon vertex:

$$-i g_s \left(f^{\nu \mu_1 \mu_2} f^{\nu \mu_3 \mu_4} \left(g^{\mu_1 \mu_3} g^{\mu_2 \mu_4} - g^{\mu_1 \mu_4} g^{\mu_2 \mu_3} \right) + (1342) + (1423) \right).$$

  Note the fresh summation index ν.

See: `example.feynman-rules.m`.

Tensor summation

Dirac tensor summation with FORM

FORM can expand traces of gamma matrices:

$$\mathrm{Tr}\left(\gamma^\mu \not{p}\right) = \left(\gamma^\mu\right)_{s_1 s_2} \left(\not{p}\right)_{s_2 s_1}$$

$= \texttt{gammachain[gamma[lor[mu]],spn[1],spn[2]]}$

$\quad \texttt{*gammachain[slash[p],spn[2],spn[1]]}$

$\rightarrow \texttt{gammachain[gamma[mu],slash[p],spn[1],spn[1]]}$

$\rightarrow \texttt{g\_(}i\texttt{,mu,p)}$

$\quad \texttt{traceN}\ i;$

$\rightarrow \texttt{4*p(mu)}$

$\rightarrow \texttt{4*momentum[p,lor[mu]]}$

On the FORM side:

- i can be any arbitrary (but unique) integer;
- `mu` must be declared as an index: `index mu = d;`
- `p` must be declared as a vector: `vector p;`
- `p` must be a single variable, not an expression (i.e. no `p+q`).

See: `example.dirac-trace*frm`, `example.to-and-from-form.m`.

Color tensor summation with COLOR.H

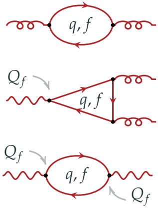
COLOR.H is a FORM package for generic color group ($SU(N)$ and beyond) tensor summation, available at
https://www.nikhef.nl/~form/maindir/packages/color/
Usage in summary:

$$
\begin{aligned}
\mathrm{Tr}\left(T^{a_1}T^{a_1}\right) &= T^{a_1}_{i_1 i_2} T^{a_1}_{i_2 i_1} \\
&= \texttt{colorT[adj[1],fun[1],fun[2]]*} \\
&\quad \texttt{*colorT[adj[2],fun[2],fun[1]]} \\
&\to \texttt{T(fun1,fun2,adj1)*T(fun2,fun1,adj1)} \\
&\quad \texttt{\#include color.h} \\
&\quad \texttt{\#call docolor} \\
&\to \texttt{NA*I2R} \\
&\to N_a T_f
\end{aligned}
$$

See: `example.color-trace.frm`.

Quark flavor summation

Flavor-related factors are complicated by the dependence of charge on flavor (Q_f). Three cases are relevant:



$$\sim N_f, \qquad \sim \sum Q_f, \qquad \sim \sum Q_f^2,$$

and the same three cases for the heavy quarks (N_t, $\sum Q_t$, $\sum Q_t^2$).

No library for this; just some FORM code that:

1. Apply each $\delta_{f_1 f_2}$ factor by renaming f_1 into f_2 (if $f_1 \neq f_2$).
2. Recognize the possible remaining cases ($Q_f^2 \delta_{ff}$, $Q_f \delta_{ff}$, δ_{ff}).

See: `example.flavor-trace.frm`.

IBP reduction

Integration-By-Parts relations

A Feynman *integral family* with N denominators D_i, L loop momenta l_i, and E external momenta p_i, is the set of integrals

$$\underbrace{I_{\nu_1,\nu_2,\dots,\nu_N}}_{\text{indices}} \equiv \int \frac{\mathrm{d}^d l_1}{(2\pi)^d} \dots \frac{\mathrm{d}^d l_L}{(2\pi)^d} \frac{1}{D_1^{\nu_1} \cdots D_N^{\nu_N}},$$

where

$$D_i \equiv \left(l_j \pm p_k \pm \dots\right)^2 - m_i^2 + i0.$$

The idea: shifting any l_k by any vector v should not change I:

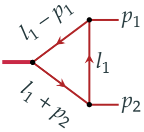
$$\lim_{\alpha\to 0} \frac{\partial}{\partial\alpha} I(l_k \to l_k + \alpha\, v) = \int \mathrm{d}^d\, l_1 \cdots \mathrm{d}^d\, l_L \frac{\partial}{\partial l_k^\mu} \frac{v^\mu}{D_1^{\nu_1} \dots D_N^{\nu_N}} \overset{!}{=} 0.$$

These are the *Integration-By-Parts (IBP) relations*. [Chetyrkin, Tkachov '81]
They hold for each $k = 1 \dots L$, and any v (out of l_i and p_i), including $v = l_k$.
There are $L(L+E)$ unique relations.

IBP relations example

Consider a *massless triangle* topology:

$$I_{a,b,c} \equiv$$



$$\equiv \int \frac{\mathrm{d}^d l}{(2\pi)^d} \frac{1}{\left(l^2\right)^a \left((l-p_1)^2\right)^b \left((l+p_2)^2\right)^c},$$

with $p_1^2 = p_2^2 = 0$, and $p_1 \cdot p_2 = s/2$.

Using the general IBP relation form and

$$v^\mu \frac{\partial}{\partial l^k} \frac{1}{(k^2)^n} = -n \frac{1}{(k^2)^{n+1}} 2 v^\mu \frac{\partial k^\nu}{\partial l^\mu} k_\nu,$$

for this example we get

$$0 = \int \frac{\mathrm{d}^d l}{(2\pi)^d} \frac{1}{\left(l^2\right)^a \left((l-p_1)^2\right)^b \left((l+p_2)^2\right)^c} \times$$
$$\times \left(\frac{\partial v^\mu}{\partial l^\mu} - 2a \frac{v \cdot l}{l^2} - 2b \frac{v \cdot (l-p_1)}{(l-p_1)^2} - 2c \frac{v \cdot (l+p_2)}{(l+p_2)^2} \right).$$

IBP relations example, cont.

Next, express all scalar products with l_i in terms of the denominators:

$$l \cdot l = l^2,$$
$$p_1 \cdot l = \frac{1}{2} l^2 - \frac{1}{2}(l - p_1)^2,$$
$$p_2 \cdot l = \frac{1}{2} l^2 - \frac{1}{2}(l + p_2)^2.$$

This allows rewriting the IBP relations in terms of $I_{a,b,c}$.
Specifically, choosing $v = \{p_1, p_2, l\}$ we get:

$$(b-a)\, I_{a,b,c} - csI_{a,b,c+1} - cI_{a-1,b,c+1} - bI_{a-1,b+1,c} + cI_{a,b-1,c+1} + aI_{a+1,b-1,c} = 0,$$

$$(a-c)\, I_{a,b,c} + bsI_{a,b+1,c} + cI_{a-1,b,c+1} + bI_{a-1,b+1,c} - bI_{a,b+1,c-1} - aI_{a+1,b,c-1} = 0,$$

$$(d - 2a - b - c)\, I_{a,b,c} - cI_{a-1,b,c+1} - bI_{a-1,b+1,c} = 0.$$

Constructing integral families

To rewrite relations between integrals (such as IBP relations) in terms of $I_{\nu_1 \dots \nu_N}$, one must express all scalar products involving the loop momenta l_i,

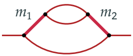
$$s_k \equiv \left\{ l_i \cdot l_j ,\ l_i \cdot p_j \right\}, \qquad k = 1 \dots L\,(L+1)/2 + LE,$$

in terms of the denominators D_i:

$$D_i = M_{ik}\, s_k + K_i \quad \Rightarrow \quad s_k = \left(M^{-1}\right)_{ki} (D_i - K_i)\,.$$

This is only possible if M is invertible, which means:

1. Each IBP family must have exactly $L\,(L+1)/2 + LE$ denominators.
   - A Feynman diagram only has up to $3L + E - 2$ lines, so each family needs $(L-1)\,(E-2+L/2)$ extra denominators not coming from the diagrams. These are "*irreducible numerators*", their indices are negative.
2. All D_i must be linearly independent when viewed as polynomials in s_k.
   - Massive diagrams naturally have terms with dependent denominators:



$$\sim \frac{1}{p^2 - m_1^2} \frac{1}{p^2 - m_2^2}.$$

   - Partial fraction decomposition must be applied before IBP.

See: `example.to-bases.frm`, `example.construct-basis.m`.

Breaking up linearly dependent denominators

In the simple case:

$$\frac{1}{p^2-m_1^2}\frac{1}{p^2-m_2^2} = \frac{1}{m_1^2-m_2^2}\frac{1}{p^2-m_1^2} + \frac{1}{m_2^2-m_1^2}\frac{1}{p^2-m_2^2}.$$

In the general case: *Leinartas' algorithm*. [Leinartas '76; Raichev '12]

1. For each term of the form $C\,D_1^{-\nu_1}\cdots D_N^{-\nu_N}$, check if there is a linear dependence among the denominators, $A_i D_i + B = 0$.
   * For this, decompose D_i into the scalar products $D_i = M_{ik} s_k + K_i$, then all $\vec{A} \in \ker \mathbb{M}^\top$ and $B = -\vec{A}\cdot\vec{K}$ will satisfy the dependence condition.
2. If $B \neq 0$, multiply the term by a factor of
$$1 = -\tfrac{1}{B}\textstyle\sum_i A_i D_i.$$
3. If $B = 0$, choose one denominator D_k and multiply the term by
$$1 = -\frac{1}{A_k D_k}\textstyle\sum_{i\neq k} A_i D_i$$
4. Repeat until no term has linearly dependent denominators.

Newer algorithms: based on algebraic geometry, e.g. MultivariateApart.

[Heller, von Manteuffel '21]

Lorentz Invariance Relations

$I_{\nu_1,\dots,\nu_N}$ should be invariant under any Lorentz rotation of the external momenta. Then, for any $\omega^\mu_\nu = -\omega^\nu_\mu$ and any p_k:

$$\lim_{\alpha\to 0}\frac{\partial}{\partial\alpha} I_{\nu_1,\dots,\nu_N}\left(p_k^\mu \to p_k^\mu + \alpha\,\omega^\mu_\nu p^\nu\right) = \omega^\mu_\nu \left(\sum_i p_k^\mu \frac{\partial}{\partial p_i^\nu}\right) I_{\nu_1,\dots,\nu_N} \stackrel{!}{=} 0.$$

Choosing ω^μ_ν to be all possible antisymmetric combinations of the form

$$\omega^\mu_\nu = p_i^\mu p_{j\nu} - p_i^\nu p_{j\mu},$$

and making the derivatives act on the integrand, we obtain the *Lorentz invariance relations*. [Gehrmann, Remiddi '99]

These follow the same structure, and are in fact linear combinations of the IBP relations. [Lee '08]

Modern software typically constructs both for the reduction.

Integral symmetries

Compare these two integrals:

$$I_1 = \int \frac{\mathrm{d}^d l}{\left(p_1 - l\right)^2 \left(p_1 + p_2 - l\right)^2 l^2} \text{ and } I_2 = \int \frac{\mathrm{d}^d l'}{\left(p_1 + l'\right)^2 \left(p_2 - l'\right)^2 l'^2}.$$

To see that they are equal, use Feynman parameters:

$$I_i = \Gamma\left(3 - \frac{d}{2}\right) \int \mathrm{d}x_1 \mathrm{d}x_1 \mathrm{d}x_1\, x_1 x_2 x_3\, \delta(1 - x_1 - x_2 - x_3)\, \mathscr{U}_i^{3-d} \mathscr{F}_i^{d/2-3},$$

$$\mathscr{U}_1 = x_1 + x_2 + x_3,\ \mathscr{F}_1 = (x_1 + x_2)\, x_3 p_1^2 + (x_1 + x_3)\, x_2 p_2^2 + 2 x_2 x_3 p_1 \cdot p_2,$$

$$\mathscr{U}_2 = x_1 + x_2 + x_3,\ \mathscr{F}_2 = (x_2 + x_3)\, x_1 p_1^2 + (x_1 + x_3)\, x_2 p_2^2 + 2 x_1 x_2 p_1 \cdot p_2.$$

Both expressions become identical under $x_{1,2,3} \leftrightarrow x_{3,2,1}$. In momenta space this corresponds to $l' = l - p_1$. [Pak '11; FEYNSON]

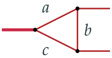
Automated implementation: FEYNSON (github.com/magv/feynson).

Example: `example.feynson.symmetrize.in`.

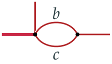
Scaleless (zero) integral detection

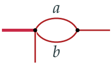
For efficiency, scaleless integrals should be put to zero early.

Consider the triangle family with one off-shell leg:

$I_{a,b,c} \equiv$ 

Two subsectors of this family are zero:

$I_{0,b,c} \equiv$ 

$= 0,$ and $I_{a,b,0} \equiv$ 

$= 0.$

Sufficient criteria (Lee '13): a family (or a subsector) is zero if there are such x-independent k_i, that

$$\sum_i k_i \, x_i \frac{\partial}{\partial x_i} \left(\mathscr{F}(x) + \mathscr{U}(x) \right) = \mathscr{F}(x) + \mathscr{U}(x),$$

where $\mathscr{F}$, $\mathscr{U}$, and x give the corresponding Feynman parameterization.

Implementation: any IBP solver, also Feynson.

Example: `example.feynson.zero-sectors.in`.

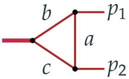
Laporta algorithm

Solving IBP relations "by hand" (with indices as symbolic variables) can be done in simpler cases. For more complicated problems use the *Laporta algorithm*: [Laporta '00]

1. Substitute integer values for the indices ν_i into the IBP relations, obtaining a large linear system with many different $I_{\nu_1 \ldots \nu_N}$.
2. Define an ordering on $I_{\nu_1 \ldots \nu_N}$ from "simple" to "complex" integrals.
   - E.g. $I_{0,1,1} < I_{1,1,0} < I_{1,1,1} < I_{1,2,1} < I_{2,1,1}$, etc.
3. Perform *Gaussian elimination* on the linear system, eliminating the most "complex" integrals first.
4. A small number of "simple" integrals will remain uneliminated.
   - $\Rightarrow$ These are the *master integrals*. The rest will be expressed as their linear combinations.
   - The number of master integrals is always finite. [Smirnov, Petukhov '04]

Solving IBP relations is a *major bottleneck* in cutting edge calculations.

IBP reduction example

$$I_{a,b,c} \equiv$$ 

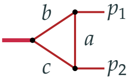
$$, \qquad p_1^2 = p_2^2 = 0, \qquad p_1 \cdot p_2 = s/2.$$

$$\begin{gathered}
(d-6)\,I_{2,1,1} - I_{1,2,1} - I_{1,1,2} = 0\\
s\,I_{1,2,1} - s\,I_{1,1,2} = 0\\
s\,I_{1,1,2} + I_{0,2,1} + I_{0,1,2} = 0\\
(d-4)\,I_{1,1,1} - I_{0,2,1} - I_{0,1,2} = 0\\
-2\,I_{-1,3,1} - I_{-1,2,2} + (d-3)\,I_{0,2,1} = 0\\
-2\,I_{-1,1,1} - s\,I_{0,1,1} = 0\\
-I_{0,3,1} + I_{0,1,3} = 0\\
s\,I_{-1,2,1} - s\,I_{-1,1,2} = 0\\
-I_{-1,2,1} - I_{-1,1,2} + (d-2)\,I_{0,1,1} = 0\\
-I_{-1,2,1} - I_{-1,1,2} - s\,I_{0,1,2} + I_{0,1,1} = 0\\
-I_{-1,2,2} - 2I_{-1,1,3} + (d-3)\,I_{0,1,2} = 0\\
s\,I_{-1,2,2} - 2s\,I_{-1,1,3} + s\,I_{0,1,2} = 0\\
-2\,I_{-1,2,2} - s\,I_{0,2,2} + I_{0,2,1} + I_{0,1,2} = 0\\
-I_{-1,2,2} - 2\,I_{-1,1,3} - 2s\,I_{0,1,3} + I_{0,1,2} = 0\\
-I_{0,2,1} + I_{0,1,2} = 0
\end{gathered}
\quad\Longleftrightarrow\quad
\begin{pmatrix}
d-6 & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot\\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot\\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1}\\ I_{1,2,1}\\ I_{1,1,2}\\ I_{1,1,1}\\ I_{-1,3,1}\\ I_{-1,1,3}\\ I_{-1,2,2}\\ I_{-1,2,1}\\ I_{-1,1,2}\\ I_{-1,1,1}\\ I_{0,3,1}\\ I_{0,1,3}\\ I_{0,2,2}\\ I_{0,1,2}\\ I_{0,2,1}\\ I_{0,1,1}
\end{pmatrix} = 0$$

After Gaussian elimination (2 operations):

$$\begin{pmatrix}
1 & 1/(6-d) & 1/(6-d) & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot\\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot\\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot\\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1}\\ I_{1,2,1}\\ I_{1,1,2}\\ I_{1,1,1}\\ I_{-1,3,1}\\ I_{-1,1,3}\\ I_{-1,2,2}\\ I_{-1,2,1}\\ I_{-1,1,2}\\ I_{-1,1,1}\\ I_{0,3,1}\\ I_{0,1,3}\\ I_{0,2,2}\\ I_{0,1,2}\\ I_{0,2,1}\\ I_{0,1,1}
\end{pmatrix} = 0$$

IBP reduction example

$$I_{a,b,c} \equiv$$ 

$$, \qquad p_1^2 = p_2^2 = 0, \qquad p_1 \cdot p_2 = s/2.$$

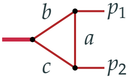
$$
\begin{gathered}
(d-6)\, I_{2,1,1} - I_{1,2,1} - I_{1,1,2} = 0 \\
s\, I_{1,2,1} - s\, I_{1,1,2} = 0 \\
s\, I_{1,1,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
(d-4)\, I_{1,1,1} - I_{0,2,1} - I_{0,1,2} = 0 \\
-2\, I_{-1,3,1} - I_{-1,2,2} + (d-3)\, I_{0,2,1} = 0 \\
-2\, I_{-1,1,1} - s\, I_{0,1,1} = 0 \\
-I_{0,3,1} + I_{0,1,3} = 0 \\
s\, I_{-1,2,1} - s\, I_{-1,1,2} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} + (d-2)\, I_{0,1,1} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} - s\, I_{0,1,2} + I_{0,1,1} = 0 \\
-I_{-1,2,2} - 2I_{-1,1,3} + (d-3)\, I_{0,1,2} = 0 \\
s\, I_{-1,2,2} - 2s\, I_{-1,1,3} + s\, I_{0,1,2} = 0 \\
-2\, I_{-1,2,2} - s\, I_{0,2,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
-I_{-1,2,2} - 2\, I_{-1,1,3} - 2s\, I_{0,1,3} + I_{0,1,2} = 0 \\
-I_{0,2,1} + I_{0,1,2} = 0
\end{gathered}
\quad \Longleftrightarrow \quad
\begin{pmatrix}
d-6 & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix}
= 0
$$

After Gaussian elimination (5 operations):

$$
\begin{pmatrix}
1 & 0 & -2/(d-6) & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix}
= 0
$$

IBP reduction example

$$I_{a,b,c} \equiv$$



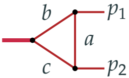
$$, \qquad p_1^2 = p_2^2 = 0, \qquad p_1 \cdot p_2 = s/2.$$

$$
\begin{gathered}
(d-6)\, I_{2,1,1} - I_{1,2,1} - I_{1,1,2} = 0 \\
s\, I_{1,2,1} - s\, I_{1,1,2} = 0 \\
s\, I_{1,1,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
(d-4)\, I_{1,1,1} - I_{0,2,1} - I_{0,1,2} = 0 \\
-2\, I_{-1,3,1} - I_{-1,2,2} + (d-3)\, I_{0,2,1} = 0 \\
-2\, I_{-1,1,1} - s\, I_{0,1,1} = 0 \\
-I_{0,3,1} + I_{0,1,3} = 0 \\
s\, I_{-1,2,1} - s\, I_{-1,1,2} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} + (d-2)\, I_{0,1,1} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} - s\, I_{0,1,2} + I_{0,1,1} = 0 \\
-I_{-1,2,2} - 2I_{-1,1,3} + (d-3)\, I_{0,1,2} = 0 \\
s\, I_{-1,2,2} - 2s\, I_{-1,1,3} + s\, I_{0,1,2} = 0 \\
-2\, I_{-1,2,2} - s\, I_{0,2,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
-I_{-1,2,2} - 2\, I_{-1,1,3} - 2s\, I_{0,1,3} + I_{0,1,2} = 0 \\
-I_{0,2,1} + I_{0,1,2} = 0
\end{gathered}
\iff
\begin{pmatrix}
d-6 & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix}
= 0
$$

After Gaussian elimination (11 operations):

$$
\begin{pmatrix}
1 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2/(ds-6s) & 2/(ds-6s) & \cdot \\
\cdot & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1/s & 1/s & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1/s & 1/s & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix}
= 0
$$

IBP reduction example

$I_{a,b,c} \equiv$ 

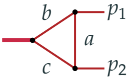
$, \qquad p_1^2 = p_2^2 = 0, \qquad p_1 \cdot p_2 = s/2.$

$$
\begin{aligned}
(d-6)\, I_{2,1,1} - I_{1,2,1} - I_{1,1,2} &= 0 \\
s\, I_{1,2,1} - s\, I_{1,1,2} &= 0 \\
s\, I_{1,1,2} + I_{0,2,1} + I_{0,1,2} &= 0 \\
(d-4)\, I_{1,1,1} - I_{0,2,1} - I_{0,1,2} &= 0 \\
-2\, I_{-1,3,1} - I_{-1,2,2} + (d-3)\, I_{0,2,1} &= 0 \\
-2\, I_{-1,1,1} - s\, I_{0,1,1} &= 0 \\
-I_{0,3,1} + I_{0,1,3} &= 0 \\
s\, I_{-1,2,1} - s\, I_{-1,1,2} &= 0 \\
-I_{-1,2,1} - I_{-1,1,2} + (d-2)\, I_{0,1,1} &= 0 \\
-I_{-1,2,1} - I_{-1,1,2} - s\, I_{0,1,2} + I_{0,1,1} &= 0 \\
-I_{-1,2,2} - 2 I_{-1,1,3} + (d-3)\, I_{0,1,2} &= 0 \\
s\, I_{-1,2,2} - 2s\, I_{-1,1,3} + s\, I_{0,1,2} &= 0 \\
-2\, I_{-1,2,2} - s\, I_{0,2,2} + I_{0,2,1} + I_{0,1,2} &= 0 \\
-I_{-1,2,2} - 2\, I_{-1,1,3} - 2s\, I_{0,1,3} + I_{0,1,2} &= 0 \\
-I_{0,2,1} + I_{0,1,2} &= 0
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{pmatrix}
d-6 & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix} = 0
$$

After Gaussian elimination (62 operations):

$$
\begin{pmatrix}
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2/(ds-6s) & 2/(ds-6s) & \cdot \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1/s & 1/s & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1/s & 1/s & \cdot \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1/(4-d) & 1/(4-d) & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (d-4)/4 & (3-d)/2 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & s/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & \cdot & (d-4)/(2s) & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2-d)/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2-d)/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 3-d \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2-d)/4 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (4-d)/2 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & (d-5)/s & -1/s & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 0 & 0 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & (d-4)/(2s) & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix} = 0
$$

IBP reduction example

$$I_{a,b,c} \equiv$$ 

$$, \qquad p_1^2 = p_2^2 = 0, \qquad p_1 \cdot p_2 = s/2.$$

$$\begin{gathered}
(d-6)\, I_{2,1,1} - I_{1,2,1} - I_{1,1,2} = 0 \\
s\, I_{1,2,1} - s\, I_{1,1,2} = 0 \\
s\, I_{1,1,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
(d-4)\, I_{1,1,1} - I_{0,2,1} - I_{0,1,2} = 0 \\
-2\, I_{-1,3,1} - I_{-1,2,2} + (d-3)\, I_{0,2,1} = 0 \\
-2\, I_{-1,1,1} - s\, I_{0,1,1} = 0 \\
-I_{0,3,1} + I_{0,1,3} = 0 \\
s\, I_{-1,2,1} - s\, I_{-1,1,2} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} + (d-2)\, I_{0,1,1} = 0 \\
-I_{-1,2,1} - I_{-1,1,2} - s\, I_{0,1,2} + I_{0,1,1} = 0 \\
-I_{-1,2,2} - 2I_{-1,1,3} + (d-3)\, I_{0,1,2} = 0 \\
s\, I_{-1,2,2} - 2s\, I_{-1,1,3} + s\, I_{0,1,2} = 0 \\
-2\, I_{-1,2,2} - s\, I_{0,2,2} + I_{0,2,1} + I_{0,1,2} = 0 \\
-I_{-1,2,2} - 2\, I_{-1,1,3} - 2s\, I_{0,1,3} + I_{0,1,2} = 0 \\
-I_{0,2,1} + I_{0,1,2} = 0
\end{gathered}
\iff
\begin{pmatrix}
d-6 & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & d-4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & -2 & \cdot & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & -s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & -1 & \cdot & \cdot & \cdot & \cdot & -s & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & d-3 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2s & s & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -2 & \cdot & \cdot & \cdot & \cdot & \cdot & -s & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -2 & -1 & \cdot & \cdot & \cdot & \cdot & -2s & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & -1 & \cdot
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix} = 0$$

After Gaussian elimination (108 operations, $\sim N^2_{\text{integrals}}$):

$$\begin{pmatrix}
1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (12-4d)/(ds^2-6s^2) \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (6-2d)/(s^2) \\
\cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (6-2d)/(s^2) \\
\cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2d-6)/(ds-4s) \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (d^2-5d+6)/(4s) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & s/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & (-d^2+7d-12)/(2s^2) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2-d)/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (2-d)/2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & (d-3)/s \\
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (d^2-5d+6)/(4s) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & (d^2-7d+12)/(2s) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & (-d^2+9d-18)/(s^2) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & (-d^2+7d-12)/(2s^2) \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & (d-3)/s
\end{pmatrix}
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1} \\ I_{0,1,1}
\end{pmatrix} = 0
\iff
\begin{pmatrix}
I_{2,1,1} \\ I_{1,2,1} \\ I_{1,1,2} \\ I_{1,1,1} \\ I_{-1,3,1} \\ I_{-1,1,3} \\ I_{-1,2,2} \\ I_{-1,2,1} \\ I_{-1,1,2} \\ I_{-1,1,1} \\ I_{0,3,1} \\ I_{0,1,3} \\ I_{0,2,2} \\ I_{0,1,2} \\ I_{0,2,1}
\end{pmatrix} =
\begin{pmatrix}
4(d-3)/((d-6)\,s^2) \\ 2(d-3)/s^2 \\ 2(d-3)/s^2 \\ -2(d-3)/((d-4)\,s) \\ -(d-3)(d-2)/(4s) \\ -(d-3)(d-2)/(4s) \\ -(d-4)(d-3)/(2s) \\ -(2-d)/2 \\ -(2-d)/2 \\ -s/2 \\ (d-4)(d-3)/(2s^2) \\ (d-4)(d-3)/(2s^2) \\ (d-6)(d-3)/s^2 \\ -(d-3)/s \\ -(d-3)/s
\end{pmatrix} I_{0,1,1}$$

IBP software

FIRE6 (bitbucket.org/feynmanIntegrals/fire) [Smirnov, Chukharev ’19]

- *Fast and parallel* Laporta-style IBP reduction implementation.
- Has a (terrible) MATHEMATICA interface, but C++ core.
- Can use modular arithmetic methods to control intermediate expression swell, and for greater parallelizibility (thousands of cores).
- Requires LITERED to discover symmetries within integral families.
- Good for zero- or single-variate reduction at high loop count.

KIRA (kira.hepforge.org): [Maierhöfer, Usovitsch, Uwer ’18]

- *Fast and parallel* Laporta-style IBP reduction implementation.
- Automatically finds symmetries within and between families.
- Optionally uses modular arithmetic via FIREFLY. [Klappert, Lange, et al ’20]
- Good for multivariate reduction.
- Main drawback: high memory use (e.g. 200GB for a 4-loop problem).

IBP software, contd.

LiteRed (inp.nsk.su/~lee/programs/LiteRed) [Lee '13]

- Heuristic-driven IBP relation solution for general indices.
- Written in Mathematica, *easy to use*, but slow, and not parallelizable.
- Contains auxiliary functions for integral differentiation, Feynman parameterization, and dimentional recurrence construction.

Forcer (github.com/benruijl/forcer) [Ruijl, Ueda, Vermaseren '17]

- Hand-crafted reduction for massless 2-point functions up to 4 loops.
- Written in Form, parallelizable.
- The *fastest thing for massless 2-point functions*.

Others:

- FiniteFlow (a library for arbitrary computations). [Peraro '19]
- Ratracer (fast modular equation solved compatible with Kira). [V.M. '22]
- Caravel (a library for amplitude computations). [Cordero, Sotnikov et al '20]
- Reduze, Air, Fmft, Matad, private implementations, etc.

IBP reduction with KIRA

Usage in short: [kira.hepforge.org]

- Define kinematics (`config/kinematics.yaml`).
- List integral families (`config/integralfamilies.yaml`).
- Create a jobs file (e.g. `jobs.yaml`), defining
  - for which integrals to *write down* IBP relations (r and s bounds);
  - for which integrals to *solve* IBP relations (r, s, and d bounds, or a list).
- Get the results as MATHEMATICA (or FORM) substitution tables.

Guide to the notation:

- r is the sum of denominator powers (positive indices);
- s is the sum of numerator powers (negative indices);
- d is the sum of dots (indices ≥ 2);
- t is the number of denominators.

$$\begin{array}{lcccc} & r & s & d & t \\ I_{1,2,1,0,0} & 4 & 0 & 1 & 3 \\ I_{1,1,3,-1,0} & 5 & 1 & 2 & 3 \\ I_{0,2,2,0,0} & 4 & 0 & 2 & 2 \\ I_{0,1,2,-2,0} & 3 & 2 & 1 & 2 \end{array} \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} 11100_2 = \text{Sector id } 7 \\ \left.\vphantom{\begin{array}{c}a\\a\end{array}}\right\} 01100_2 = \text{Sector id } 6 \end{array}$$

See: `kira_example/` and `example.kira.m`.

Rational function arithmetic: classical

$$f(x,y) = \frac{2xy - y^2}{x - y} + \frac{y^3 - 3xy^2}{x^2 - y^2} = ?$$

Computing the result *the classical way*:

1. Common denominator:	$((2xy - y^2)(x + y) + y^3 - 3xy^2)/(x^2 - y^2)$
2. Expand the numerator:	$(2x^2y - xy^2 + 2xy^2 - y^3 + y^3 - 3xy^2)/(x^2 - y^2)$
3. Combine alike terms:	$(2x^2y - 2xy^2)/(x^2 - y^2)$
4. Cancel common factors:	$2xy/(x + y)$
Runtime:	$\mathcal{O}(N^2_{\text{initial monomials}} N_{\text{digits per monomial}})$
Peak memory needed:	$\mathcal{O}(N^2_{\text{initial monomials}} N_{\text{digits per monomial}})$

Note:

- short input (polynomials with up to 2 monomials),
- large intermediate expression (up to 8 monomials per poly),
- short output (up to 2 monomials per poly).

$\Longrightarrow$ The runtime scales with the *intermediate expression size*!
Can the runtime scale only with the *output size* instead?

Rational function arithmetic: interpolation

Computing the result via *interpolation* based on an anzatz:

1. Prepare an ansatz: $f(x,y) = c_1 xy/(x + c_2 y)$
2. Evaluate f (twice): $f(1,1) = 1,\ f(1,2) = 4/3$
3. Solve for c_i: $c_1 = 2,\ c_2 = 1$

Runtime, evaluation: $N_{\text{final monomials}} \times \mathscr{O}(N_{\text{initial monomials}} N_{\text{digits per monomial}})$

Runtime, interpolation: $\mathscr{O}(N^2_{\text{final monomials}} N_{\text{digits per monomial}})$

Peak memory needed: $\mathscr{O}(N^2_{\text{final monomials}} N_{\text{digits per monomial}})$

The runtime scales with the result size (number of final monomials).

- But is still scales with the number of digits in the intermediate expressions.
  Can it be proportional only to $N_{\text{digits per final monomial}}$?

Rational function arithmetic: modular interpolation

Same interpolation, but using *modular arithmetic*:

- Interpolate *keeping the values as integers modulo a prime numer* P_1.
  - E.g. modulo 997: $567 + 678 = 248$; $-1 = 996$; $1/2 = 499$; etc.
- Use *rational number reconstruction* to upgrade c_i from integers to rationals modulo P_1. [Wang '81; Monagan '04]
- Repeat the same with primes P_2, P_3, ...
- Use the *Chinese remainder theorem* to get c_i modulo $P_1 \cdot P_2 \cdot P_3 \cdots$.
- Stop when c_i no longer change.

Runtime: same, but $N_{\text{digits per monomial}} \to N_{\text{digits per final monomial}}$.

This is also *faster on a computer*: all operations are on small integers!

Modular interpolation example

To find a symbolic form of a rational function $f(x_1, \dots, x_N)$:

- *Evaluate* f modulo a prime number *many times*, with x_i set to integers.
- *Reconstruct* the exact symbolic form of f from the obtained values.

Example: if we have an unknown $f(x)$, and we have evaluated

$$f(11) = 139 \pmod{997}, \qquad f(65) = 479 \pmod{997},$$
$$f(38) = 350 \pmod{997}, \qquad f(92) = 115 \pmod{997},$$

then we can use *polynomial interpolation* to find a polynomial form of f:

$$f(x) = 618 + 979\,x + 486\,x^2 + 41\,x^3 \pmod{997},$$

and then *rational function reconstruction* to find an equivalent rational form:

$$f(x) = \frac{996 + 333x}{1 + x} \pmod{997},$$

and finally *rational number reconstruction* to find the rational coefficients:

$$f(x) = \frac{-1 + \frac{2}{3}x}{1 + x} \pmod{997}.$$

Guess that this is the true form of $f(x)$; evaluate more times to verify.

Function reconstruction algorithms

If an anzatz is unknown, multiple *reconstruction algorithms* are available:

- *Univariate* case:
  - Newton interpolation for *dense polynomials*. [Newton 1675; Peraro '16]
    - Number of evaluations $\sim N_{\text{maximal degree}}$.
  - Ben-Or/Tiwari for *sparse polynomials*. [Ben-Or, Tiwari '88]
    - Number of evaluations $\sim 2N_{\text{monomials}}$.
  - Thiele interpolation for *dense rationals*.
    - Number of evaluations $\sim 2N_{\text{maximal degree}}$.
- *Multivariate* case:
  - Newton applied recursively in each variable for *dense polynomials*.
    - Number of evaluations $\sim (N_{\text{maximal degree}})^{N_{\text{scales}}}$.
  - Zippel ($\sim$ recursive Newton with prunning) + early termination for *sparse polynomials*. [Zippel '90; Kaltofen, Lee '03]
    - Number of evaluations $\lesssim N_{\text{scales}}\, N_{\text{maximal degree}}\, N_{\text{monomials}}$.
  - Multivariate Ben-Or/Tiwari for *sparse polynomials*. [Go '06]
    - Number of evaluations $\sim 2N_{\text{monomials}}$.
  - First Thiele, then Zippel and/or Ben-Or/Tiwari for *multivariate rationals* (the FireFly library). [Klappert, Lange '19; Klappert, Klein, Lange '20]

IBP performance checklist

To improve IBP performance:

1. Use modular arithmetic methods. [von Manteuffel, Schabinger '14; Peraro '16]
2. *Make the result smaller*:
   2.1 Reduce whole amplitudes (not individual integrals).
   2.2 Choose master integrals that minimize the result size.
      * Use *d-factorizing bases* that ensure the factorization of d in the denominators of IBP coefficients. [Usovitsch '20; Smirnov, Smirnov '20]
      * Consider *quasi-finite bases*. [von Manteuffel, Panzer, Schabinger '14]
      * Consider *uniform transcendentality bases*, if possible. [Bendle et al '19]

   2.3 Construct a smaller ansatz for the result. [Abreu et al '19; De Laurentis, Page '22]
   2.4 Set some of the variables to fixed numbers.
      * E.g. reduce with m_H^2/m_t^2 set to $12/23$.
      * Or perform IBP reduction separately for each phase-space point, and interpolate in between. [Jones, Kerner et al '18; Chen, Heinrich et al '19, '20]
3. Improve the *evaluation performance*:
   3.1 Combine IBP relations (using syzygies) to eliminate integrals with raised (or lowered) indices. [Gluza, Kajda, Kosower '10; Scahbinger '11]
   3.2 Pre-solve the IBP system to simplify it before solving it (NeatIBP, Blade).
   [Wu, Boehm, Ma, Xu, Zhang '23; Guan, Liu, Ma, Wu '23]

Using Kira with modular reconstruction (FireFly)

Basic idea:

1. Instruct Kira to use FireFly for modular reconstruction.
2. Don't ask for reduction tables for each integral, instead reduce complete expressions for each amplitude.
   Because smaller output $\longrightarrow$ faster reduction.

In Kira this is done in two steps:

1. Instruct Kira to export the IBP equations into files.
2. Add additional equation files, with equations like $\mathrm{AMP}_1 = C_1 I_{123} + C_2 I_{112} + \ldots$, one per each amplitude.
3. Instruct Kira to load all the "user-defined" equations and reduce AMP_i to master integrals.

See: `kira_example_amplitude/` and `example.kira-amplitude.m`.

Integral evaluation via sector decomposition

Sector decomposition in short

$$I = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y \, \left(x+y\right)^{-2+\varepsilon} = ?$$

Problem: the integrand diverges at $x, y \to 0$, can't integrate numerically.

Solution: [Heinrich '08; Binoth, Heinrich '00]

1. Factorize the divergence in x and y with sector decomposition:
   - $I = \int \cdots \times \Big(\underbrace{\theta\left(x>y\right)}_{\text{Sector 1}} + \underbrace{\theta\left(y>x\right)}_{\text{Sector 2}}\Big) = \int_0^1 \mathrm{d}x \int_0^x \mathrm{d}y \, \left(x+y\right)^{-2+\varepsilon} + \begin{pmatrix} x \\ \updownarrow \\ y \end{pmatrix}$
2. Rescale the integration region in each sector back to a hypercube:
   - $I \overset{y \to x y}{=} \int_0^1 \mathrm{d}x \, \underbrace{x^{-1+\varepsilon}}_{\text{Factorized pole}} \int_0^1 \mathrm{d}y \left(1+y\right)^{-2+\varepsilon} + \begin{pmatrix} x \\ \updownarrow \\ y \end{pmatrix}$
3. Extract the pole at $x \to 0$ analytically, expand in ε:
   - $I = -\frac{2}{\varepsilon} \int_0^1 \mathrm{d}y \left(1+y\right)^{-2+\varepsilon} = -\frac{2}{\varepsilon} \int_0^1 \mathrm{d}y \left(\frac{1}{\left(1+y\right)^2} - \frac{\ln\left(1+y\right)}{\left(1+y\right)^2} \varepsilon + \mathscr{O}\left(\varepsilon^2\right) \right)$
4. Integrate each term in ε numerically (they all converge now).

In practice: geometric sector decomposition. [Bogner, Weinzierl '07; Kaneko, Ueda, '09]

Contour deformation in short

$$I \equiv \int \mathrm{d}^n\vec{x}\, \frac{U^\alpha\left(\vec{x}\right)}{F^\beta\left(\vec{x}, \dots\right) + i0}$$

Problem: can't integrate numerically if $F = 0$ inside the integration region.
Solution: *deform $\vec{x}$ into the complex plane* to escape the pole:

$$\vec{x} \to \vec{x} + i\,\vec{\Delta}\left(\vec{x}\right)$$

$$\Rightarrow \begin{cases} F \to F + i\,\Delta\,\partial_x F - \Delta^2\,\partial_x^2 F - i\,\Delta^3\,\partial_x^3 F + \mathcal{O}\left(\Delta^4\right), \\ \operatorname{Im} F \to \Delta\,\partial_x F - \Delta^3\,\partial_x^3 F + \mathcal{O}\left(\Delta^5\right). \end{cases}$$

Choose $\vec{\Delta}\left(\vec{x}\right)$ to enforce the $+i0$ prescription ($\operatorname{Im} F > 0$):

$$\vec{\Delta}\left(\vec{x}\right) = \lambda\,\vec{\partial}_x F\left(\vec{x}\right) \quad \Rightarrow \quad \operatorname{Im} F \approx \lambda\,\left(\partial_x F\right)^2 - \lambda^3\left(\partial_x F\right)^3\,\partial_x^3 F + \mathcal{O}\left(\lambda^5\right) > 0.$$

- Lambda should be small enough that $\operatorname{Im} F > 0$.
- But: larger λ improves convergence (the pole is further away).
- In practice: choose λ heuristically, but decrease it if $\operatorname{Im} F < 0$.
  - Gradient-based λ optimization can be useful. [Winterhalder et al '21]

Sector decomposition software

pySecDec (github.com/gudrunhe/secdec): [Heinrich et al ’23, ’21, ’18, ’17, ’15, ’08, ’00]

- A Python (3.8+) library that generates C++ code for integration.
  - Uses `Format O`n from Form to optimize the integrand expressions.
- Installable via `python3 -m pip install pySecDec`.
- Can integrate on CPUs & GPUs.
- Integration via *Randomized Quasi Monte Carlo* (Qmc); optionally Vegas/Suave/Divonne/Cuhre (Cuba), or Cquad (Gsl).
- Adaptive evaluation of *weighted sums of integrals* (i.e. amplitudes).
  - Evaluating a sum is faster than evaluating each integral separately.

Fiesta (bitbucket.org/feynmanIntegrals/fiesta): [Smirnov et al ’21, ’15, ’13, ’09, ’08]

- Mathematica core, generates/compiles to C++ behind the scenes.
- Monte-Carlo integration (Qmc and others) on both CPUs and GPUs.
- Interprets more than compiles (good at high loops and few scales).

Feyntrop (github.com/michibo/feyntrop): [Borinsky, Munch, Tellander ’23]

- Tropical sampling for quasi-finite integrals (no sector decomposition).
- When applicable, can be faster than e.g. pySecDec, especially at higher loops and deep expansions in ε.

Sector decomposition with pySecDec

Usage overview (details: secdec.readthedocs.io):

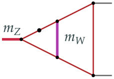
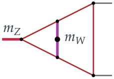
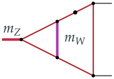
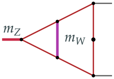
1. Use the Python module `pySecDec` to define your integrals and generate the code for the integration library.
2. Compile the integration library.
3. Import the integration library from Python (or the command line), call it to perform integration.

See: `example.pysecdec.py`, and then `example.pysecdec.m`.

Limitations of sector decomposition

Practical limitations of the method:

- The numerical convergence can be poor in high energy regions, near thresholds, and in other special parameter configurations.
  - Asymptotic expansion helps in some cases (available in pySecDec and Fiesta/`asy2.m`), but not in others.
- Integrals with many propagators and contour deformation will take a lot of time to compile.
- Related integrals can have widely different convergence rates. For example, integration time to 10^{-3} precision with pySecDec:[1]

	orders	t, s		orders	t, s
	ε^{-3} ... ε^{0}	27		ε^{-2} ... ε^{0}	57
	ε^{-2} ... ε^{0}	1230		ε^{-2} ... ε^{0}	>9000

[1] pySecDec 1.5.3, NVidia A100 GPU.

Back to the e^+e^- annihilation

Tying it all together

$$|M(e^+e^- \to \text{partons})|^2 = 4\pi\alpha(2-d)\,\mathrm{Re}\Big(F_1(q)\Big)$$

$$= 4\pi\alpha\,\mathrm{Re}\left(\frac{g_{\mu\nu}}{q^2}\sum\right.$$

$$\left.\vphantom{\frac{g}{q^2}}\right)$$

$$= \,?$$

See: `allthecode.m`, `plot-pysecdec-results.py` or `plot-pysecdec-results.ipynb`.

Differential equations

Method of differential equations

Consider a family of integrals depending on external momenta p_i and masses m_i:

$$I_{\nu_1,\nu_2,\ldots,\nu_N} \equiv \int \frac{\mathrm{d}^d l_1}{(2\pi)^d} \ldots \frac{\mathrm{d}^d l_L}{(2\pi)^d} \frac{1}{D_1^{\nu_1} \ldots D_N^{\nu_N}} = I_{\nu_1,\nu_2,\ldots,\nu_N}\left(\left\{p_i \cdot p_j, m_i^2\right\}\right).$$

With dimensional analysis one of the parameters can be scaled out, making remaining arguments dimensionless:

$$I\left(\left\{p_i \cdot p_j, m_i\right\}\right) = \left(p_1^2\right)^{\frac{d}{2}L - \sum_i \nu_i} I\left(\{x_i\}\right), \qquad \{x_i\} = \left\{\frac{p_i \cdot p_j}{p_1^2}, \frac{m_i^2}{p_1^2}\right\}.$$

Idea: if $\{x_i\} \neq \emptyset$, we can construct differential equations in x_i, and solve them instead of performing the loop integration directly.
(In practice: just set $p_1^2 = 1$, restore it in the end by dimensionality).

Constructing differential equations

Suppose IBP relations were solved, and we have a set of master integrals I_i.

1. Differentiate each I_i by one of the parameters, $x = m_a^2$ or $p_a \cdot p_b$:

$$\partial_{m_a^2} I = \int \frac{\mathrm{d}^d l_1}{(2\pi)^d} \dots \frac{\mathrm{d}^d l_L}{(2\pi)^d} \, \partial_{m_a} \frac{1}{D_1^{\nu_1} \dots D_N^{\nu_N}},$$

$$\partial_{p_a \cdot p_b} I = \left(G^{-1}\right)_{ia} p_i \cdot \left(\partial_{p_b} I\right), \quad \partial_{p_a \cdot p_a} I = \frac{1}{2} \left(G^{-1}\right)_{ia} p_i \cdot \left(\partial_{p_a} I\right),$$

   where G is the Gram matrix: $G_{ij} \equiv p_i \cdot p_j$.

2. Express the derivatives as integrals in the same family:

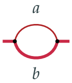
$$\partial_x I_i = \sum_k C_k \, I_{\nu_1^{(k)}, \nu_2^{(k)}, \dots, \nu_N^{(k)}}.$$

3. Use the IBP tables to reduce those integrals back to the master integrals, thus obtaining a linear differential equation system for I_i:

$$\partial_x I_i \overset{\text{IBP}}{=} \sum_j M_{ij} \, I_j.$$

Example: self-energy with one mass

Consider a family of self-energy with one mass integrals

$$I_{a,b} \equiv$$


$$= \int \frac{\mathrm{d}^d l}{(2\pi)^d} \frac{1}{\left(q-l\right)^{2a} \left(l^2 - m^2\right)^b}.$$

This family has two master integrals:

$$I_1 \equiv I_{0,1} =$$

$$= \int \frac{\mathrm{d}^d l}{(2\pi)^d} \frac{1}{l^2 - m^2},$$

$$I_2 \equiv I_{1,1} =$$

$$= \int \frac{\mathrm{d}^d l}{(2\pi)^d} \frac{1}{\left(q-l\right)^2 \left(l^2 - m^2\right)}.$$

Constructing differential equation system:

$$\partial_{m^2} \underbrace{\begin{pmatrix} I_1 \\ I_2 \end{pmatrix}}_{\vec{I}} = \begin{pmatrix} I_{0,2} \\ I_{1,2} \end{pmatrix} \overset{\text{IBP}}{=} \underbrace{\begin{pmatrix} \frac{-2+d}{2m^2} & 0 \\ \frac{2-d}{2m^2\left(m^2-q^2\right)} & \frac{-3+d}{m^2-q^2} \end{pmatrix}}_{\mathbb{M}} \underbrace{\begin{pmatrix} I_1 \\ I_2 \end{pmatrix}}_{\vec{I}}.$$

Solution via the epsilon-form

If we have a differential equation system for $\vec{I}$,

$$\partial_x \vec{I}(x,d) = \mathbb{M}(x,d)\,\vec{I}(x,d), \qquad \text{where } d = 4-2\varepsilon,$$

we can transform to a new basis $\vec{J}$ via the transformation matrix $\mathbb{T}$,

$$\vec{I}(x,\varepsilon) = \mathbb{T}(x,\varepsilon)\,\vec{J}(x,\varepsilon),$$

and for $\vec{J}$ the same differential equation system will look like

$$\partial_x \vec{J} = \mathbb{T}^{-1}\left(\mathbb{M}\mathbb{T} - \partial_x\mathbb{T}\right)\vec{J} \equiv \mathbb{M}'\,\vec{J}.$$

Idea: if this system is in an *ε-form* (has the dependence on ε factorized),

$$\partial_x \vec{J}(x,\varepsilon) = \varepsilon\,\mathbb{S}(x)\,\vec{J}(x,\varepsilon),$$

then the solution for $\vec{J}$ as a series in ε becomes trivial: [Henn '13]

$$\vec{J}(x,\varepsilon) \equiv \varepsilon^{k_0}\sum_{k=0}^{\infty}\varepsilon^k \vec{J}^{(k)}(x),$$

$$\sum \varepsilon^k \vec{J}^{(k)} = \sum \varepsilon^{k+1}\mathbb{S}\,\vec{J}^{(k)} \quad\Rightarrow\quad \vec{J}^{(k)}(x) = \int^x \mathbb{S}(x')\,\vec{J}^{(k-1)}(x')\mathrm{d}x' + \vec{C}^{(k)}.$$

Example, cont.: the epsilon-form

The ε-form can be achieved with a transformation to the following basis $\vec{J}$:

$$\vec{I} = \begin{pmatrix} (-1+2\varepsilon)\, m^2 & 0 \\ -1+\varepsilon & (-1+\varepsilon)\left(m^2 - q^2\right) \end{pmatrix} \vec{J}.$$

Differential equation system in $\vec{J}$ then has the form

$$\partial_{m^2} \vec{J} = \varepsilon \begin{pmatrix} -\frac{1}{m^2} & 0 \\ -\frac{1}{q^2}\frac{1}{m^2} - \frac{1}{q^2}\frac{1}{m^2-q^2} & -\frac{2}{m^2-q^2} \end{pmatrix} \vec{J}.$$

Accordingly, the solution is

$$\vec{J}^{(0)} = \vec{C}^{(0)},$$

$$\vec{J}^{(1)} = \vec{C}^{(1)} + \begin{pmatrix} -C_1^{(0)} \int^{m^2} \frac{\mathrm{d}m'^2}{m'^2} \\ \dots \end{pmatrix},$$

$$\vec{J}^{(2)} = \vec{C}^{(2)} + \begin{pmatrix} -C_1^{(1)} \int^{m^2} \frac{\mathrm{d}m'^2}{m'^2} + C_1^{(0)} \int^{m^2} \frac{\mathrm{d}m'^2}{m'^2} \int^{m'^2} \frac{\mathrm{d}m''^2}{m''^2} \\ \dots \end{pmatrix}.$$

Iterated integrals

The solution to differential equation in an ε-form always come as iterated integrals of the form of *multiple polylogarithms* (a.k.a Goncharov polylogarithms): [Goncharov '98]

$$G(w_1, w_2, \ldots, w_n; x) \equiv \int_0^x \frac{\mathrm{d}t_1}{t_1 - w_1} \int_0^{t_1} \frac{\mathrm{d}t_2}{t_2 - w_2} \cdots \int_0^{t_n} \frac{\mathrm{d}t_n}{t_n - w_n}.$$

Special case for the trailing zeros:

$$G(\underbrace{0, \ldots, 0}_{n}; x) \equiv \frac{1}{n!} \log^n(x).$$

Integration and differention, the simple case:

$$\int \mathrm{d}x \frac{1}{x-a} G(\vec{w}; x) = G(a, \vec{w}; x) + C, \quad \frac{\mathrm{d}}{\mathrm{d}x} G(w_1, \vec{w}; x) = \frac{1}{x - w_1} G(\vec{w}; x).$$

Software: GiNaC, Hpl, Harmpol, HyperInt, PolyLogTools, etc.

Example, cont.: the solution in GPLs

Rewriting the iterated integrals in terms of G:

$$\vec{J}^{(0)} = \vec{C}^{(0)},$$
$$\vec{J}^{(1)} = \vec{C}^{(1)} + \begin{pmatrix} -C_1^{(0)} G(0; m^2) \\ \dots \end{pmatrix},$$
$$\vec{J}^{(2)} = \vec{C}^{(2)} + \begin{pmatrix} -C_1^{(1)} G(0; m^2) + C_1^{(0)} G(0, 0; m^2) \\ \dots \end{pmatrix}.$$

It is trivial to generate this answer to any required order. Note that the answer will only have $w_i \in \{0, q^2\}$. By rescaling the weights (or setting q^2 to 1), this can be turned into $w_i \in \{0, 1\}$, the *harmonic polylogarithms*.

Multiple polylogarithms, more properties

Differentiation in the general case when w_i may depend on x:

$$\frac{\mathrm{d}}{\mathrm{d}x} G(w_1, \ldots, w_n; y) = \sum_i G(w_1, \ldots, \cancel{w_i}, \ldots, w_n; y) \frac{\mathrm{d}}{\mathrm{d}x} \log \frac{w_i - w_{i-1}}{w_i - w_{i+1}},$$

where $w_0 \equiv y$, and $w_{i+1} \equiv 0$.

Integration in the general case when w_i may depend on x is nontrivial, one must first represent G in a way that the integration variable only appears in the last argument. [Brown '08]

Automated via HyperInt (bitbucket.org/PanzerErik/hyperint): [Panzer '14]

```
$ maple
> read "HyperInt.mpl";
> fibrationBasis(Hlog(x,[y,x,y]),[x,y]);
Hlog(x,[y,y,y])-Hlog(x,[y,0,y])
```

Multiple polylogarithms and their relatives

The integral representation of multiple polylogarithms (G) is equivalent to the infinite sum representation (Li):

$$\mathrm{Li}_{m_1,\ldots,m_n}(x_1,\ldots,x_n) = \sum_{i_1>\cdots>i_n>0} \frac{x_1^{i_1}}{i_1^{m_1}} \cdots \frac{x_n^{i_n}}{i_n^{m_n}} =$$

$$= (-1)^n\, G\Big(\underbrace{0,\ldots,0}_{m_1-1},\frac{1}{x_i},\ldots,\underbrace{0,\ldots,0}_{m_2-1},\frac{1}{x_1 x_2\cdots x_n};1\Big).$$

- Note: there are conflicting conventions for the order of the indices in the Li summation. Above is the “physicist” notation (used in e.g. HPL, GINAC, and the MZV datamine).
- The “mathematician” notation is reverse: $0 < i_1 < \cdots < i_n$; it was used by Goncharov, and is also used in HYPERINT. The order of indices in the Multiple Zeta Values is also reversed there.

Multiple polylogarithms and their relatives, II

- *Logarithms* are GPLs with a single weight:

$$\log x = G(0;x), \qquad \log\left(\frac{a-x}{a}\right) = G(a;x).$$

- *Nielsen's generalized polylogarithms* are GPLs with $w_i \in \{0,1\}$:

$$S_{n,p}(x) = (-1)^p\, G(\underbrace{0,\ldots,0}_{n},\underbrace{1,\ldots,1}_{p};x).$$

- *Harmonic polylogarithms* (HPLs) are GPLs with $w_i \in \{0,\pm 1\}$:

$$H_{\ldots,+m,-n,0}(x) = (-1)^m G(\ldots,\underbrace{0,\ldots,0,1}_{m},\underbrace{0,\ldots,0,-1}_{n},0;x).$$

- *Two-dimensional HPLs* are GPLs with $w_i \in \{0,1,1-z,-z\}$.
- *Multiple Zeta Values* $\zeta_{\vec{w}}$ are just $H_{\vec{w}}(1)$ (in the "physicist" notation).

Fixing the integration constants

Differential equations only give the solution up to the integration constants. Finding these constants is the essential difficulty of the method. There are many ways.

- By evaluating the integrals in a limit where they simplify.
  - Large mass limit. Small mass limit. [Smirnov '02]
  - Even massless integrals can be evaluated by adding masses to them, and connecting the large mass limit to the massless limit via the differential equations.
- Using the knowledge of the analytic properties of the integrals.
  - E.g.: enforcing regularity in the kinematic limits. [Gehrmann, Remiddi '00]
- From partial knowledge of the integrals values, such as a Mellin moment.
  - One can integrate over the semi-inclusive integrals to obtain the fully inclusive ones. [Gituliar '15]

Example, cont.: integration constants

First, we consider I_1 (the vacuum bubble) “simple”, and look it up in a book:


$$= -\frac{i\pi^{\frac{d}{2}}}{(2\pi)^d}\Gamma\left(1-\frac{d}{2}\right)\left(m^2-i0\right)^{\frac{d}{2}-1}.$$

For I_2: in the limit of zero mass the massive diagram becomes massless,

$$\lim_{m^2\to 0}$$

$$=$$

.

This massless integral we also know from a book:


$$= \frac{i\pi^{\frac{d}{2}}}{(2\pi)^d}\frac{\Gamma^2\left(\frac{d}{2}-1\right)\Gamma\left(2-\frac{d}{2}\right)}{\Gamma(d-2)}\left(-q^2-i0\right)^{\frac{d}{2}-2}.$$

Then, we can expand these in series’ in ε, compare with the series from ε-form solution, and fix all $\vec{C}^{(k)}$.

See: `example.diff-eq-massive-self-energy.m`.

The fundamental solution

A *fundamental solution* to $\partial_x \vec{I} = \mathbb{M}\,\vec{I}$ is an n by n matrix of independent solutions, such that any solution can be expressed as a linear combination of its columns.

A fundametal solution for a system in an ε-form, $\partial_x \vec{J} = \varepsilon\,\mathbb{S}\,\vec{J}$, can be constructed as a series in ε:

$$\mathbb{W} = \mathbb{1} + \varepsilon \int_{x_0}^{x} \mathrm{d}x'\,\mathbb{S}(x') + \varepsilon^2 \int_{x_0}^{x} \mathrm{d}x'\,\mathbb{S}(x') \int_{x_0}^{x'} \mathrm{d}x''\,\mathbb{S}(x'') + \dots .$$

The general solution is then just $\mathbb{W}$ multiplied by a vector of integration constants $\vec{C}$:

$$\vec{J}(x,\varepsilon) = \mathbb{W}(x,\varepsilon)\,\vec{C}(\varepsilon),$$

where the constants themselves are a series in ε,

$$\vec{C}(\varepsilon) \equiv \varepsilon^{k_0} \sum_{k=0}^{\infty} \varepsilon^k \vec{C}^{(k)}.$$

This is an alternative way to write down a solution for $\vec{J}$, with the benefit of being immediately extendable to the multivariate case.

The multivariate case

If differential equations in multiple variables are considered, and a combined ε-form is achieved,

$$\partial_{x_i} \vec{J}(\vec{x}, \varepsilon) = \varepsilon\, \mathbb{S}_i(\vec{x})\, \vec{J}(\vec{x}, \varepsilon),$$

then writing down the solution in this case can be made easy by:

1. Choosing an integration contour along the axes x_i in an some order, for example from $(0, 0, \dots)$ to $(x_1, 0, \dots)$, then to $(x_1, x_2, 0, \dots)$, etc.
2. Writing down the fundamental solutions along each segment, $\mathbb{W}_i$. Because segments are chosen such that only x_i changes along each, $\mathbb{W}_i$ can be calculated the same as in single-variate case.

The general solution for $\vec{J}$ is then

$$\vec{J}(\vec{x}, \varepsilon) = \mathbb{W}_n(x_1, \dots, x_n, \varepsilon) \cdots \mathbb{W}_n(x_1, \varepsilon)\, \vec{C}(\varepsilon).$$

Overall this result will be a sum of terms of this form:

$$G(\text{arguments depending on } x_1, \dots, x_{n-1}; x_n) \cdots G(\text{constants}; x_1).$$

Epsilon-form software

FUCHSIA (github.com/magv/fuchsia.cpp) [Gituliar, V.M. '17; V.M. '22]

- Uses the Lee algorithm. [Lee '14]
- Initial version in PYTHON/SAGEMATH, newer version in C++/GINAC.
- Can certify if a system is irreducible (with exceptions).
- Can suggest variable changes that will make it reducible.

LIBRA (github.com/rnlg/Libra) [Lee '20]

- Lee algorithm in MATHEMATICA.
- Has a GUI to manually choose transformation steps.

CANONICA (github.com/christophmeyer/CANONICA) [Meyer '17]

- Uses the rational ansatz method (in MATHEMATICA).
  Constructs an ansatz for the transformation matrix up to some powers of the variables, then solves for the coefficients.

EPSILON (github.com/mprausa/epsilon) [Prausa '17]

- Lee algorithm in C++. Single variable only.

INITIAL (github.com/UT-team/INITIAL) [Dlapa, Henn, Yan '20]

- Needs the knowledge of a single uniform transcendentality integral.

See: `example.diff-eq-massive-self-energy.m`.

Differential equations, numerically

When an ε-form can not be achieved, differential equations can be solved numerically instead:

1. Construct differential equations along some line in parameter space.
2. Use the differential equation system to construct a power series anzatz for the integrals at multiple points along this line.
3. Determine the anzatz coefficients at one of the points via boundary conditions.
4. Determine the ansatz coefficients at a nearby point by matching the value of the integral numerically.
5. Move to each point in order.

Available software:

- DiffExpr (gitlab.com/hiddingm/diffexp) [Hidding '20]
- AMFlow (gitlab.com/multiloop-pku/amflow) [Liu, Ma '22]
- SeaSyde (github.com/TommasoArmadillo/SeaSyde) [Armadillo, Bonciani, Devoto, Rana, Vicini '22]

Summary

Summary

We have talked about:

- Computing loop amplitudes with QGRAF, FORM, COLOR.H, FEYNSON.
- MATHEMATICA as the hot glue.
- IBP reduction with KIRA.
- Numerical evaluation with pySECDEC.
- Differential equations, the ε-form, multiple polylogarithms.

Summary

We have talked about:

- Computing loop amplitudes with QGRAF, FORM, COLOR.H, FEYNSON.
- MATHEMATICA as the hot glue.
- IBP reduction with KIRA.
- Numerical evaluation with pySECDEC.
- Differential equations, the ε-form, multiple polylogarithms.

Thank you for your attention.

Backup slides: Lee algorithm

Lee algorithm for finding the epsilon-form

Overall idea: to go from a differential equation system

$$\partial_x \vec{I} = \mathbb{M}\,\vec{I}, \quad \text{where } \mathbb{M}(x,\varepsilon) = \sum_i \frac{\mathbb{A}_i(\varepsilon)}{(x-x_i)^{k_i}},$$

to an ε-form

$$\partial_x \vec{J} = \varepsilon\,\mathbb{S}\,\vec{J}, \quad \text{where } \mathbb{S}(x) = \sum_i \frac{\mathbb{S}_i}{x-x_i},$$

apply a series of simple basis transformation $\vec{I} = \mathbb{T}\,\vec{J}$, such that each brings the system a bit closer to an ε-form. [Lee '14]

1. If a higher pole is present (i.e. $k_i \neq 1$) then use a transformation that reduces the rank of $\mathbb{A}_i$, eventually eliminating it. (“Fuchsification”).
2. Else, for the eigenvalues of $\mathbb{A}_i$ of the form $n + k\varepsilon$, use a transformation that shifts n by ± 1, eventually setting it to zero. (“Normalization”).
3. If all $\mathbb{A}_i$ eigenvalues are proportional to ε, use a transformation that makes the whole $\mathbb{A}_i$ proportional to ε. (“Factorization”).

Lee algorithm, fuchsification

Consider *a “balance” transformation between x_1 and x_2*:

$$\mathbb{T}(x,\varepsilon) = \overline{\mathbb{P}}(\varepsilon) + \frac{x-x_2}{x-x_1}\mathbb{P}(\varepsilon), \quad \text{with } \mathbb{P}^2 = \mathbb{P}, \text{ and } \mathbb{P} + \overline{\mathbb{P}} = \mathbb{1}.$$

If the matrix $\mathbb{M}$ has a pole at $x = x_1$ of power $n > 1$,

$$\mathbb{M} = \mathbb{A}_{-n}\,(x-x_1)^{-n} + \mathbb{A}_{-n+1}\,(x-x_1)^{-n+1} + \dots,$$

then either

1. there is such $\mathbb{P}$ and x_2 , that the transformed $\mathbb{A}_{-n}$ is of lower rank than $\mathbb{A}_{-n}$, while the leading expansion order around $x = x_2$ does increase beyound $n = 1$; or
2. the ε-form cannot be reached by any rational transformation.

Then, a series of balance transformations can decrease the rank of all $\mathbb{A}_{-n}$ with $n \neq 1$ to zero, transforming $\mathbb{M}$ into the *Fuchsian form*:

$$\mathbb{M}(x,\varepsilon) = \sum \frac{\mathbb{A}_i(\varepsilon)}{x-x_i}.$$

Lee algorithm, normalization

Differential equations for master integrals are observed to have a special feature: when transformed into Fuchsian form,

$$\mathbb{M} = \sum \frac{\mathbb{A}_i}{x - x_i},$$

the eigenvalues of $\mathbb{A}_i$ often have the form of $n + k\varepsilon$, where $n \in \mathbb{N}$.
Now, a balance can be found between x_1 and x_2 that does not spoil the Fuchsian form. Such a balance will shift one of the eigenvalues of $\mathbb{A}_1$ by $+1$, one of $\mathbb{A}_2$ by -1.
Then, a series of such balances can transform $\mathbb{M}$ into a *normalized Fuchsian form*: where all $\mathbb{A}_i$ have eigenvalues proportional to ε.

- Sometimes eigenvalues of the form $\frac{1}{2} + n + k\varepsilon$ are encountered. If they are present at up to three different $\mathbb{A}_i$, then it is possible to change the integration variable from x to such y, that the differential equation in y has all the eigenvalues in the form $n + k\varepsilon$.
- Sometimes eigenvalues are not linear in ε. This often means the master integral basis is linearly dependent.

Lee algorithm, factorization

Finally, once all the eigenvalues of $\mathbb{M}$ residues are proportional to ε, then either

1. the whole matrix can be made proportional to ε by a transformation that does not depend on x; or
2. the ε-form can not be reached.

Such transformation is searched for via an ansatz.
Note that on this step, and on all previous ones too, there exists multiple transformations that can achieve the desired form. As a result, the ε-form is not unique.

Lee algorithm, the multivariate case

If a system of differential equations in multiple variables is considered:

$$\partial_{x_i}\vec{I}(\vec{x},\varepsilon) = \mathbb{M}_i(\vec{x},\varepsilon)\,\vec{I}(\vec{x},\varepsilon),$$

then it is useful to have a single transformation $\mathbb{T}(x,\varepsilon)$ that transforms all of the differential equation systems into ε-form simultaneously,

$$\partial_{x_i}\vec{J}(\vec{x},\varepsilon) = \varepsilon\,\mathbb{S}_i(\vec{x})\,\vec{J}(\vec{x},\varepsilon), \qquad \text{with } \vec{I} = \mathbb{T}\,\vec{J}.$$

Single-variable Lee algorithm can be reused for this by:

1. Reducing $\mathbb{M}_1$ to an ε-form with $\mathbb{T}_1(x_1, x_2, \dots, \varepsilon)$.
2. Transforming all the equations with $\mathbb{T}_1$.
3. Reducing $\mathbb{M}_2$ to an ε-form with a such a $\mathbb{T}_2(x_2, \dots, \varepsilon)$ that is independent of x_1 and ε.
4. Transforming all the equations with $\mathbb{T}_2$. This will not spoil the ε-form in x_1 because $\mathbb{T}_2$ does not depend on x_1 or ε.
5. Repeating similarly for the rest of $\mathbb{M}_i$.