Faster IBP solving via RATRACER and tricks

Vitaly Magerya

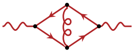
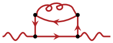
Institute for Theoretical Physics (ITP),
Karlsruhe Institute of Technology (KIT)

Hamburg
January 24, 2023

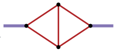
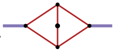
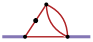
A typical calculation in HEP

Basic steps of calculating scattering matrix elements:

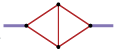
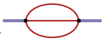
1. List Feynman diagrams for the process.

   $\Rightarrow \mathscr{M} =$  $+$  $+ \ldots$

2. Project the diagrams onto scalar integrals.

   $\Rightarrow \mathscr{M} = C_1$  $+ C_2$  $+ C_3$  $+ \ldots$

3. Reduce to master integrals via *Integration-by-Part (IBP) relations*.

   $\Rightarrow \mathscr{M} = C'_1$  $+ C'_2$  $+ \ldots$

4. Evaluate the loop integrals.
   - Analytically:
     - Direct integration (by hand, with HYPERINT, etc).
     - Differential equations for master integrals (e.g. FUCHSIA, LIBRA). *Needs IBP.*
     - Dimensional recurrence relations (SUMMERTIME, DREAM). *Needs IBP.*
   - Numerically:
     - Sector decomposition (PYSECDEC, FIESTA, etc).
     - Mellin-Barnes representation (MB, AMBRE, etc).
     - Numerical differential equations (DIFFEXP, AMFLOW, etc). *Needs IBP.*

5. Integrate over the kinematics.

IBP relations

An *IBP integral family* with *L loop momenta* $\vec{l}_i$, and *E external momenta* $\vec{p}_i$, is the set of Feynman integrals

$$I_{\nu_1,\nu_2,\ldots,\nu_N} \equiv \int \frac{\mathrm{d}^d\vec{l}_1 \cdots \mathrm{d}^d\vec{l}_L}{D_1^{\nu_1} \cdots D_N^{\nu_N}},$$

where ν_i are the "indices" (denominator powers), their number is

$$N \equiv L\,(L+1)\,/2 + LE,$$

and the denominators themselves are

$$D_i \equiv \left(\text{linear combination of } \vec{l}_j \text{ and } \vec{p}_k\right)^2 - m_i^2 + i0.$$

The idea: shifting $\vec{l}_k$ by any vector $\vec{v}$ should not change I:

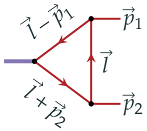
$$\lim_{\alpha\to 0} \frac{\partial}{\partial\alpha} I\left(\vec{l}_k \to \vec{l}_k + \alpha\vec{v}\right) = \int \mathrm{d}^d\vec{l}_1 \cdots \mathrm{d}^d\vec{l}_L \frac{\partial}{\partial l_k^\mu} \frac{v^\mu}{D_1^{\nu_1} \ldots D_N^{\nu_N}} = 0.$$

These are the *IBP relations* (for all k and all $\vec{v}$, $L\,(L+E)$ in total).

IBP relations example

For an example massless triangle topology:

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



$$\equiv \int \frac{\mathrm{d}^d \vec{l}}{\left(\vec{l}^2\right)^a \left((\vec{l}-\vec{p}_1)^2\right)^b \left((\vec{l}+\vec{p}_2)^2\right)^c},$$

$$\vec{p}_1^{\,2} = \vec{p}_2^{\,2} = 0, \qquad 2\,\vec{p}_1 \cdot \vec{p}_2 = s.$$

For $k = 1$ and $\vec{v} = \vec{p}_1$:

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

For $k = 1$ and $\vec{v} = \vec{p}_2$:

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

For $k = 1$ and $\vec{v} = \vec{l}$:

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

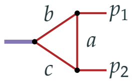
Laporta algorithm

To solve IBP relations in practice 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 $I_{\nu_1 \dots \nu_N}$.
2. Define an ordering on $I_{\nu_1 \dots \nu_N}$ from "simple" to "complex" integrals.
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.

IBP reduction example

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

$$, \qquad p_1^2 = p_2^2 = 0, \qquad 2p_1p_2 = s.$$

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

IBP reduction as a bottleneck

To match LHC experimental precision the theory requires 2-loop corrections. For future colliders: 3-loop corrections. [Freitas '21]

	1 loop	2 loops	3 loops
4 legs	~100 diagrams 3 families 4 denominators 2 mass scales	~2K diagrams 24 families 7+2 denominators 2 mass scales	~50K diagrams 219 families 10+5 denominators 2 mass scales
5 legs	~1K diagrams 12 families 5 denominators 5 mass scales	~30K diagrams 180 families 8+3 denominators 5 mass scales	~800K diagrams 2355 families 11+7 denominators 5 mass scales

- Lines in a Feynman diagram: $N_{\text{lines}} = 3L + E - 2$.
- Denominators per family: $N_{\text{denominators}} = L\,(L+1)\,/2 + LE$.
- Mass scales per family: $N_{\text{scales}} = E\,(E-1)\,/2 - 1 + N_{\text{massive legs + masses}}$.

Optimistic *reduction time estimate* (symbolic reduction):

$$T \sim \frac{1}{\text{performance}_S} N_{\text{families}} \underbrace{N^2_{\text{integrals per family}}}_{\exp(N_{\text{denominators}})} \underbrace{N^2_{\text{monomials per term}}}_{\exp(N_{\text{scales}})} N_{\text{digits per monomial}}$$

Improving IBP reduction time

$$T \sim \frac{1}{\text{performance}_S} N_{\text{families}} N^2_{\text{integrals per family}} N^2_{\text{monomials per term}} N_{\text{digits per monomial}}$$

Strategies to reduce T:

1. 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 using quasi-finite bases. [von Manteuffel, Panzer, Schabinger '14]
   - $\Rightarrow$ Lowers $N_{\text{monomials per term}}$ and $N_{\text{digits per monomial}}$.
2. Combine IBP relations (using syzygies) to eliminate integrals with raised (or lowered) indices. [Gluza, Kajda, Kosower '10; Scahbinger '11]
   - ⋆ Pre-solves the IBP system symbolically, lowering $N^2_{\text{integrals per family}}$.
3. Set some of the variables to fixed numbers.
   - ⋆ E.g. reduce with m_H^2 set to $\frac{12}{23}\, m_t^2$.
   - $\Rightarrow$ Decreases N_{scales} and $N_{\text{monomial per term}}$, but increases $N_{\text{digits per monomial}}$.
4. Set all of the variables to numbers.
   - ⋆ I.e. perform IBP reduction separately for each phase-space point, and interpolate in between. [Jones, Kerner et al '18; Chen, Heinrich et al '19, '20]
   - $\Rightarrow$ Decreases $N_{\text{monomial per term}}$ to 1, but increases $N_{\text{digits per monomial}}$.
5. Use *modular arithmetic methods*. [von Manteuffel, Schabinger '14]

Modular arithmetic methods

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.

Performance of IBP via modular arithmetic

For modular methods, total time for *reconstruction*:

$$\frac{1}{\text{performance}_R} N_{\text{families}} \underbrace{\cancel{N^2_{\text{integrals per family}}}}_{\to N_{\text{amplitudes}} N_{\text{masters}}} N^2_{\text{monomials per term}} N_{\text{digits per monomial}}$$

Total time for *evaluation*:

$$\overbrace{\frac{1}{\text{performance}_E} N_{\text{families}} N^2_{\text{integrals per family}}}^{T_{\text{single evaluation}}} \overbrace{N^{\cancel{2}}_{\text{monomials per term}} N_{\text{digits per monomial}}}^{N_{\text{evaluations}}}$$

- On conventional computers, $\text{performance}_E > \text{performance}_S$.
- The evaluation can be naturally parallelized.
- Reducing whole amplitudes instead of individual integrals is needed to minimize the reconstruction time.

Available software

IBP solvers that use modular arithmetic:

- FINRED (private implementation) [von Manteuffel et al]
- FIRE6 [Smirnov, Chuharev '19]
  - Does not provide multivariate reconstruction.
- KIRA when used with FIREFLY [Klappert, Lange, Maierhöfer, Usovitsch '20; Klappert, Klein, Lange '20]
- FINITEFLOW [Peraro '19]
  - A framework for arbitrary computations.
- CARAVEL [Cordero, Sotnikov et al '20]
  - A framework for amplitude computations.

Now also introducing:

- RATRACER (with KIRA and FIREFLY) [V.M. '22]

RATRACER overview

RATRACER (“Rational Tracer”): a program for *solving systems of linear equations* using modular arithmetic, *faster*. [V.M. ’22; github.com/magv/ratracer]

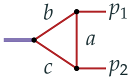
- Accepts arbitrary systems of linear equations: IBP relations, dimensional recurrence relations, amplitude definitions, etc.
- Initially developed for solving IBP systems for massive 5-point 2-loop diagrams.
- *Improves the evaluation* time, reconstruction is still done by FIREFLY.
- Intended usage:
  1. Use KIRA (or LITERED, or custom code) to export IBP relations.
  2. Use RATRACER to load them and solve them.

The performance improvement is based on simple observations:

- IBP equation solvers that use modular arithmetic (KIRA+FIREFLY, FIRE6) spend lots of time managing data structures that represent equations.
- Each evaluation re-creates the same data structures in the same way.

$\Longrightarrow$ Eliminate the data structures: *record the list of arithmetic operations* (“trace”) performed during the first evaluation, and just *replay the list* for subsequent evaluations.

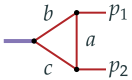
Rational traces

For $I_{a,b,c}(s,d) \equiv$  the *trace* of the IBP solution might look like:

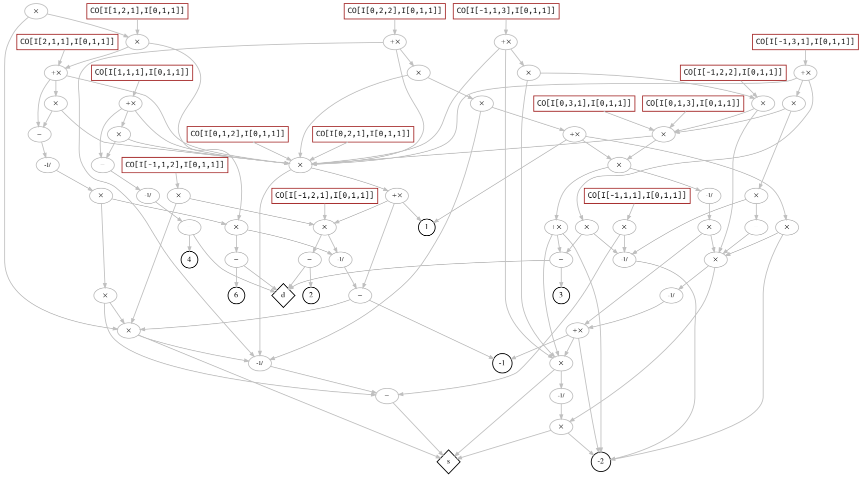
```
t0 = var 'd'
t1 = int 4
t2 = sub t0 t1
t3 = int 1
t4 = var 's'
t5 = neg t4
t6 = int 6
t7 = sub t0 t6
t8 = int -1
t9 = int 2
t10 = int -2
t11 = sub t0 t9
t12 = int 3
t13 = sub t0 t12
t14 = mul t4 t10
t15 = neginv t5
t16 = mul t4 t15
t17 = sub t8 t16
t18 = mul t5 t16
t19 = neginv t17
t20 = mul t7 t19
[...]
```

```
t54 = addmul t53 t27 t44
t55 = mul t25 t44
t56 = addmul t55 t25 t44
t57 = mul t23 t44
t58 = addmul t57 t23 t44
t59 = mul t20 t58
t60 = mul t16 t59
save t60 as CO[I[1,1,2],I[0,1,1]]
save t59 as CO[I[1,2,1],I[0,1,1]]
save t58 as CO[I[2,1,1],I[0,1,1]]
save t56 as CO[I[1,1,1],I[0,1,1]]
save t54 as CO[I[-1,1,3],I[0,1,1]
save t52 as CO[I[-1,2,2],I[0,1,1]
save t51 as CO[I[-1,3,1],I[0,1,1]
save t46 as CO[I[0,1,3],I[0,1,1]]
save t49 as CO[I[0,2,2],I[0,1,1]]
save t47 as CO[I[-1,1,2],I[0,1,1]
save t46 as CO[I[0,3,1],I[0,1,1]]
save t42 as CO[I[-1,2,1],I[0,1,1]
save t44 as CO[I[0,1,2],I[0,1,1]]
save t44 as CO[I[0,2,1],I[0,1,1]]
save t45 as CO[I[-1,1,1],I[0,1,1]
```

Rational traces

For $I_{a,b,c}(s,d) \equiv$ 

the *trace* of the IBP solution might look like:



Trace optimizations

Given a trace, RATRACER can optimize it using:

- *Constant propagation:*

$$\begin{cases}\texttt{t11 = int 2}\\ \texttt{t12 = int 3}\\ \texttt{t13 = mul t11 t12}\end{cases} \Rightarrow \begin{cases}\texttt{t11 = int 2}\\ \texttt{t12 = int 3}\\ \texttt{t13 = int 6}\end{cases}$$

- *Trivial operation simplification:*

$$\begin{cases}\texttt{t11 = int -1}\\ \texttt{t12 = mul t11 t7}\end{cases} \Rightarrow \begin{cases}\texttt{t11 = int -1}\\ \texttt{t12 = neg t7}\end{cases}$$

- *Common subexpression elimination:*

$$\begin{cases}\texttt{t11 = add t5 t7}\\ \texttt{t12 = add t5 t7}\end{cases} \Rightarrow \begin{cases}\texttt{t11 = add t5 t7}\\ \texttt{t12 = t11}\end{cases}$$

- *Dead code elimination:*

$$\begin{cases}\texttt{t11 = add t5 t7}\\ \texttt{[..., t11 is unused]}\end{cases} \Rightarrow \begin{cases}\texttt{nop}\\ \texttt{[...]}\end{cases}$$

  - Especially useful if a user wants to select a subset of the outputs.

- *“Finalization”:*

$$\begin{cases}\texttt{t11 = add t5 t6}\\ \texttt{t12 = add t11 t7}\\ \texttt{[..., t11 is unused]}\end{cases} \Rightarrow \begin{cases}\texttt{t11 = add t5 t6}\\ \texttt{t11 = add t11 t7}\\ \texttt{[...]}\end{cases}$$

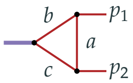
  - Needed to minimize the temporary memory needed for the evaluation.

Trace transformations

Given a trace, RATRACER can:

- Set some of the variables to numbers or expressions.
  - E.g. set `mh2` to `12/23*mt2`, `d` to `4-2*eps`, or `s` to `13600`.
  - No need to remake the IBP system just to set a variable to a number.
- Select any subset of the outputs, and drop operations that don't contribute to them (via dead code elimination).
  - Can be used to split the trace into parts.
    - Each part can be reconstructed separately (e.g. on a different machine).
  - See master-wise and sector-wise reduction in other solvers.
- Expand the result into a series in any variable.
  - E.g. in practice only the leading few orders in ε are needed.
    - $\Rightarrow$ Expand in ε up to e.g. $\mathcal{O}\left(\varepsilon^0\right)$, don't compute the higher orders.
  - Done before the reconstruction, so one less variable to reconstruct in, but potentially more expressions (depending on the truncation order).

Truncated series expansion

For $I_{a,b,c}(s,d) \equiv$  before expansion:

- Variables to reconstruct in: s and d.
- Trace outputs: "`CO[I[1,1,1],I[0,1,1]]`", etc:

$$I_{1,1,1} = \texttt{CO[I[1,1,1],I[0,1,1]]}\, I_{0,1,1}.$$

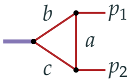
After expansion in ε to $\mathcal{O}\left(\varepsilon^0\right)$:

- Variables to reconstruct in: only s.
- Trace outputs: "`ORDER[CO[I[1,1,1],I[0,1,1],eps^-1]`", etc:

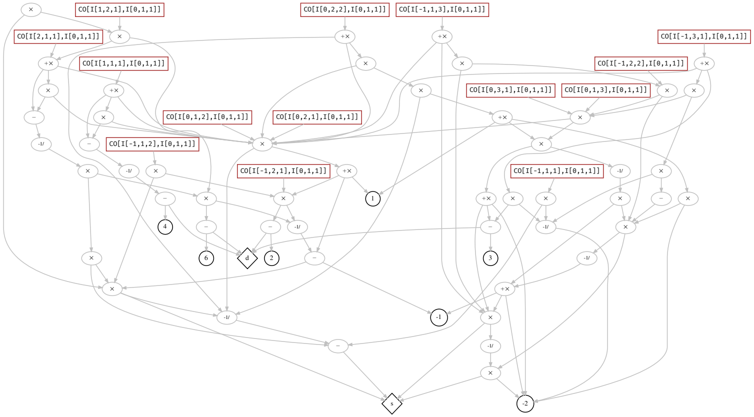
$$\begin{aligned} I_{1,1,1} &= \texttt{ORDER[CO[I[1,1,1],I[0,1,1],eps\^{}-1]}\, \varepsilon^{-1}\, I_{0,1,1} \\ &+ \texttt{ORDER[CO[I[1,1,1],I[0,1,1],eps\^{}0]}\, \varepsilon^{0}\, I_{0,1,1}. \end{aligned}$$

- Might be slower to evaluate, but fewer evaluations are needed.
  - $\Longrightarrow$ The more complicated the problem, the higher the speedup.

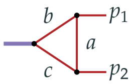
Truncated series expansion: the graph



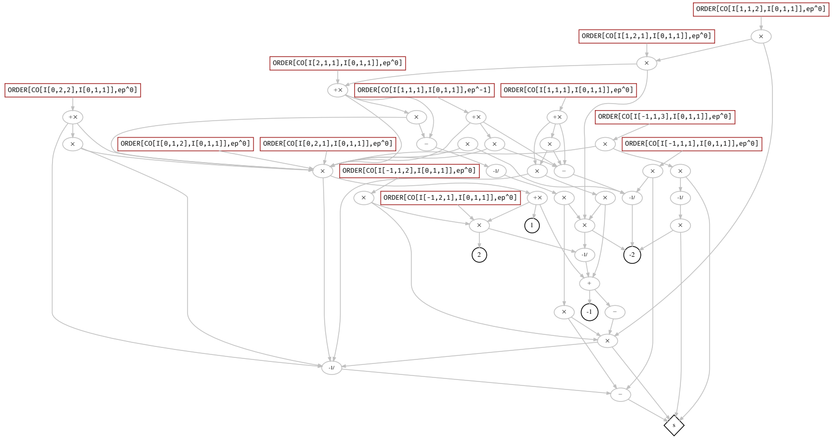
Expansion:	–	$\mathscr{O}\left(\varepsilon^0\right)$	$\mathscr{O}\left(\varepsilon^1\right)$	$\mathscr{O}\left(\varepsilon^2\right)$
Operations:	61	40	73	87
Outputs:	15	13	26	33



Truncated series expansion: the graph

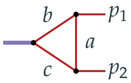


Expansion:	–	$\mathcal{O}(\varepsilon^0)$	$\mathcal{O}(\varepsilon^1)$	$\mathcal{O}(\varepsilon^2)$
Operations:	61	40	73	87
Outputs:	15	13	26	33

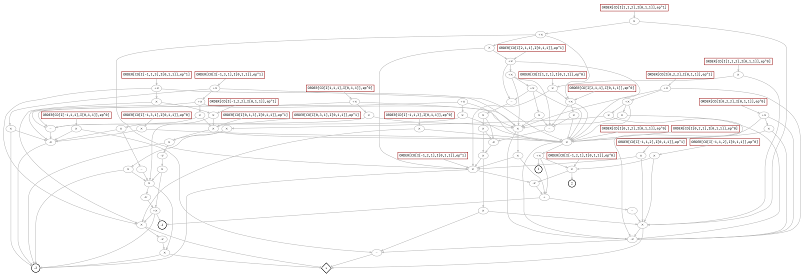


ORDER[CO[I[1,1,2],I[0,1,1]],ep^0]
ORDER[CO[I[1,2,1],I[0,1,1]],ep^0]
ORDER[CO[I[2,1,1],I[0,1,1]],ep^0]
ORDER[CO[I[0,2,2],I[0,1,1]],ep^0]
ORDER[CO[I[1,1,1],I[0,1,1]],ep^-1]
ORDER[CO[I[1,1,1],I[0,1,1]],ep^0]
ORDER[CO[I[-1,1,3],I[0,1,1]],ep^0]
ORDER[CO[I[0,1,2],I[0,1,1]],ep^0]
ORDER[CO[I[0,2,1],I[0,1,1]],ep^0]
ORDER[CO[I[-1,1,1],I[0,1,1]],ep^0]
ORDER[CO[I[-1,1,2],I[0,1,1]],ep^0]
ORDER[CO[I[-1,2,1],I[0,1,1]],ep^0]
1
2
-2
-1
s

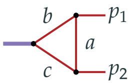
Truncated series expansion: the graph



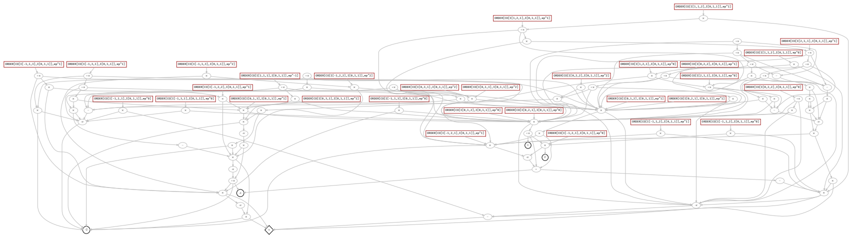
Expansion:	–	$\mathscr{O}\left(\varepsilon^0\right)$	$\mathscr{O}\left(\varepsilon^1\right)$	$\mathscr{O}\left(\varepsilon^2\right)$
Operations:	61	40	73	87
Outputs:	15	13	26	33



Truncated series expansion: the graph

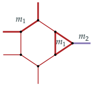
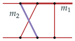
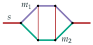
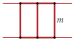


Expansion:	–	$\mathscr{O}\left(\varepsilon^0\right)$	$\mathscr{O}\left(\varepsilon^1\right)$	$\mathscr{O}\left(\varepsilon^2\right)$
Operations:	61	40	73	87
Outputs:	15	13	26	33



RATRACER benchmarks

For IBP reduction of every integral (i.e. not single amplitudes):

	Evaluation speedup vs. KIRA+FIREFLY	$\frac{t_{\text{reconstruction}}}{t_{\text{evaluation}}}$	Total speedup vs. KIRA+FIREFLY	Total speedup vs. KIRA+FERMAT	Total speedup vs. FIRE6	$\mathscr{O}(\varepsilon^0)$ speedup
	20	3.3	5.2	1.2	∞?	3.2
	7.8	1/3.3	6.0	37	∞?	2.7
	26	25	1.7	1/3.3	1.8	2.3
	9.6	8.8	5.2	2.6	8.8	4.3

[github.com/magv/ibp-benchmark]

Resulting performance:

- Consistent ~10x speedup in modular evaluation over KIRA+FIREFLY.
- Up to *~5x speedup in total reduction time* over KIRA+FIREFLY for complicated examples, 1x-30x over KIRA+FERMAT, ∞x over FIRE6.
- A *~3x speedup on top of that* with ε expansion.

Problems with big traces

Problem:

- The size of a trace is proportional to the number of operations.
    - I.e. $\sim N^2_{\text{integrals}}$ for sparse IBP systems.
    - $\Rightarrow$ Megabytes to hundreds of gigabytes for problems of interest.
- Computer memory is expensive and limited.

Solution:

1. Always keep the trace on disk, never load it fully into memory.
    - Compress it on disk for storage (via ZSTD, GZIP, LZMA, etc).
2. During the evaluation read the trace piece by piece.
3. During the optimization make sure the algorithms have bounded memory usage.

$\Rightarrow$ Multi-GB traces are supported easily in RATRACER.

Guessing the denominators

The *denominators of IBP coefficients factorize* into few unique factors.
If some candidate factors are known, then we can find the powers of those factors in each coefficient: [Heller, von Manteuffel '21]

1. Choose a factor to search for, e.g. $(d-6)$.
2. Set all variables to random values, e.g. $d = 95988281$, $s = 75579811$.
   $\Rightarrow$ $(d-6) = 95988275 = 5^2 \cdot 103 \cdot 37277$.
3. Evaluate the IBP solution using these numbers.
   - ⋆ E.g. $\texttt{CO}\,[I_{2,1,1}\,, I_{0,1,1}] = \frac{383953112}{548314574947073136171275} = \frac{2^3 \cdot 1117 \cdot 42967}{5^2 \cdot 103 \cdot 37277 \cdot 75579811^2}$.
4. Find common prime factors, identify their powers.
   - ⋆ $\texttt{CO}\,[I_{2,1,1}\,, I_{0,1,1}] \sim (d-6)^{-1}\, s^{-2}$.

Automated implementation: `toos/guessfactors` from RATRACER.

To find the set of possible factors:

- ⋆ Reconstruct a simpler subset of the coefficients. (A few per sector).
- $\Rightarrow$ Easy with RATRACER, just select individual outputs.

Once the factors are found, *speedup the reconstruction* by dividing them out from the expressions.

- ⋆ I.e. reconstruct $\texttt{CO}\,[I_{2,1,1}\,, I_{0,1,1}] / (d-6) / s^2$, not just $\texttt{CO}\,[I_{2,1,1}\,, I_{0,1,1}]$.

Usage for IBP reduction

1. Use Kira to generate the IBP equations.

```
$ cat >config/integralfamilies.yaml <<EOF
integralfamilies:
  - name: "I"
    loop_momenta: [l]
    top_level_sectors: [b111]
    propagators:
      - ["l", 0]
      - ["l-p1", 0]
      - ["l+p2", 0]
EOF
$ cat >config/kinematics.yaml <<EOF
kinematics:
 outgoing_momenta: [p1, p2]
 kinematic_invariants: [[s, 2]]
 scalarproduct_rules:
  - [[p1,p1], 0]
  - [[p2,p2], 0]
  - [[p1,p2], "s/2"]
# symbol_to_replace_by_one: s
EOF
```

```
$ cat >export-equations.yaml <<EOF
jobs:
 - reduce_sectors:
    reduce:
     - {sectors: [b111], r: 4, s: 1}
    select_integrals:
     select_mandatory_recursively:
      - {sectors: [b111], r: 4, s: 1}
    run_symmetries: true
    run_initiate: input
EOF
$ kira export-equations.yaml
```

2. Use Ratracer to create a trace with the solution.

```
$ ratracer \
    load-equations input_kira/I/SYSTEM_I_0000000007.kira.gz \
    load-equations input_kira/I/SYSTEM_I_0000000006.kira.gz \
    solve-equations choose-equation-outputs --maxr=4 --maxs=1 \
    optimize finalize save-trace I.trace.gz
```

3. Optionally expand the outputs into a series in ε.

```
$ ratracer \
    set d '4-2*eps' load-trace I.trace.gz \
    to-series eps 0 \
    optimize finalize save-trace I.eps0.trace.gz
```

4. Use Ratracer (+FireFly) to reconstruct the solution.

```
$ ratracer \
    load-trace I.eps0.trace.gz \
    reconstruct --to=I.solution.txt --threads=8 --inmem
```

Usage as a library

RATRACER is built to support custom user-defined traces.

Any rational algorithm can be turned into a trace (via the C++ API).

Usage:

```
#include <ratracer.h>
int main() {
    Tracer tr = tracer_init();

    Value x = tr.var(tr.input("x"));
    Value y = tr.var(tr.input("y"));

    Value x_sqr =
        tr.pow(x, 2);
    Value expr =
        tr.add(x_sqr, tr.mulint(y, 3));

    /* expr = x^2 + 3y */
    tr.add_output(expr, "expr");

    tr.save("example.trace.gz");
    return 0;
}
```

API:

```
struct Value { uint64_t id; uint64_t val; };
struct Tracer {
    Value var(size_t idx);
    Value of_int(int64_t x);
    Value of_fmpz(const fmpz_t x);
    bool is_zero(const Value &a);
    bool is_minus1(const Value &a);
    Value mul(const Value &a, const Value &b);
    Value mulint(const Value &a, int64_t b);
    Value add(const Value &a, const Value &b);
    Value addint(const Value &a, int64_t b);
    Value sub(const Value &a, const Value &b);
    Value addmul(const Value &a,
                 const Value &b1,
                 const Value &b2);
    Value inv(const Value &a);
    Value neginv(const Value &a);
    Value neg(const Value &a);
    Value pow(const Value &base, long exp);
    Value div(const Value &a, const Value &b);
    void assert_int(const Value &a, int64_t n);
    void add_output(const Value &src, const char *name);
    size_t input(const char *name, size_t len);
    size_t input(const char *name);
    int save(const char *path);
    void clear();
};
Tracer tracer_init();
```

RATRACER future plans

For very large examples *main memory speed becomes the bottleneck* for the modular evaluation. So:

- Investigate optimizing traces to improve memory access patterns.

For other examples RATRACER speeds up the evaluation so much that the *modular reconstruction in FIREFLY becomes the bottleneck*. So:

- Reduce the overhead in FIREFLY to speed up simpler examples.
- Improve paralelizability in FIREFLY to help with complicated examples.

$\Longrightarrow$ Ongoing collaboration with FIREFLY and KIRA authors.

Once denominators are known, the results can be partial-fractioned:

- Known to reduce final expression size in some cases. [De Laurentis, Page '22]
- Pending implementation & investigation.

Summary

Solving IBP equation systems faster:

- Spend time on choosing better master basis.
- IBP-reduce amplitudes, not individual integrals.
- Use Kira + Ratracer + FireFly.
- Expand the coefficients in a series in ε up to the needed order.
- Guess the denominators of the coefficients.

Ratracer:

- Faster modular evaluation of linear system solutions.
- Trace optimization, transformation, slicing and dicing.
- Available at github.com/magv/ratracer
- TODO: faster evaluation, faster reconstruction, more tricks.