CALICO: parametric annihilators for loop integrals & special functions

Gaia Fontana (University of Zürich)

In collaboration with Giuseppe Bertolini & Tiziano Peraro

Universität Zürich[UZH]

MathemAmplitudes 2025, Mainz — 24/09/2025

arXiv:2506.13653

Menu of the day

- Introduction & notation
- Parametric annihilators
    - How do we compute them?
- A similar technique for differential operators
- Applications
- Conclusions & outlook

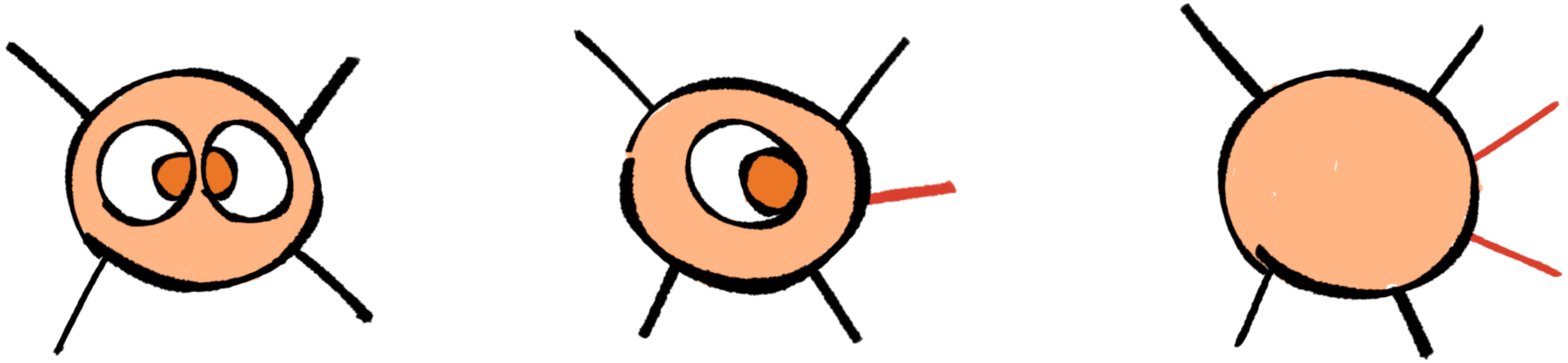
Introduction

Precision era @ colliders

- Precision physics as
  - **test** of the Standard model
  - gate to **new physics**

- High-Lumi upgrade of LHC :
- theory and experiments must have comparable uncertainties
  - needed: %-level accuracy:
    - perturbation theory @ NNLO and often N3LO
    - diagrams with increasing no. of **loops, legs & mass scales**

$@\mathcal{O}(\alpha_s^2)$:



Loop Integrals

- LEGO® blocks of perturbative QFT beyond tree level
- Key ingredient of phenomenological predictions
- Rich and interesting mathematical structures

Thousands of loop integrals appear when studying perturbative predictions!

- **Crucial:** Finding relations between them
- Loop integrals admit various integral representations with different tradeoffs and mathematical properties

This work

- Study & elaborate on the method of **parametric annihilators** for finding integral identities,
  - Focus on **parametric representations** of loop integrals
  - Extend applications to **different** representations
  - Similar technique for finding **differential equations**
- Provide an **implementation** of annihilators & differential operators based on modern linear solvers relying on cutting edge **finite-fields** techniques
- Implementation in the public Mathematica package: **`CALICO`**

```
Computing Annihilators from Linear Identities
Constraining (differential) Operators
```

Some useful definitions

A list of variables: $\mathbf{z} = (z_1, \ldots, z_n)$

A list of exponents: $\alpha = (\alpha_1, \ldots, \alpha_n)$

$\longrightarrow$ $\mathbf{z}^\alpha = \prod_j z_j^{\alpha_j}$ Monomials; $|\alpha| = \sum_j \alpha_j$ total degree

We are interested in **integral families** of the form

These include **many parametrizations** of loop integrals

$$I_\alpha = \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z})\, u(\mathbf{z})$$

Rational factors raised to integer powers

Twist
Multivalued

$$\begin{cases} u(\mathbf{z}) = \prod_j B_j(\mathbf{z})^{\gamma_j} \\ u(\mathbf{z}) = \exp F(\mathbf{z}) \prod_j B_j(\mathbf{z})^{\gamma_j} \end{cases}$$

Assumption:

$\varphi_\alpha(\mathbf{z})$ closed under monomial multiplication & differentiation

What do we want to do?

- Finding and solving **linear relations** satisfied by integrals having the form of I_α
- Express integrals within a family as a linear combination of a set of independent **master integrals (MIs)**

$$I_\alpha = \sum_{\beta \in \mathrm{MIs}} c_{\alpha\beta} I_\beta$$

Crucial ingredient are **Integration-By-Parts identities (IBP):**

$$\int \mathrm{d}^n \mathbf{z}\, \partial_j \Big(\varphi_\alpha(\mathbf{z})\, u(\mathbf{z}) \Big) = 0$$

```
(Regulated integrals vanish at the integration boundary)
```

$$\int \mathrm{d}^n\mathbf{z}\,\partial_j\Big(\varphi_\alpha(\mathbf{z})\,u(\mathbf{z})\Big) = 0$$

By expanding the LHS of the IBP relation we get :

$$\int \mathrm{d}^n\mathbf{z}\,\partial_j\Big(\varphi_\alpha(\mathbf{z})\,u(\mathbf{z})\Big) = \int \mathrm{d}^n\mathbf{z}\;(\partial_j\varphi_\alpha(\mathbf{z}))\;u(\mathbf{z}) + \int \mathrm{d}^n\mathbf{z}\,\varphi_\alpha(\mathbf{z})\;(\partial_j \log u(\mathbf{z}))\;u(\mathbf{z})$$

We get:

- Non-trivial identities among the integrals
- $\partial_j \log u(\mathbf{z})$ usually creates terms that cannot be absorbed in $\varphi_\alpha(\mathbf{z})$

not a relation within the original integral family!

Parametric annihilators

Integral identities via parametric annihilators

Parametric annihilator of order o of $u(\mathbf{z})$

[Baikov (1996); Lee (2014); Bitoun, Bogner, Klausen, Panzer (2017)]

$$\hat{A} = c_0(\mathbf{z}) + \sum_j c_j(\mathbf{z})\,\partial_j + \sum_{j_1 \le j_2} c_{j_1 j_2}(\mathbf{z})\,\partial_{j_1}\partial_{j_2} + \cdots + \sum_{j_1 \le \cdots \le j_o} c_{j_1 \cdots j_o}(\mathbf{z})\,\partial_{j_1} \cdots \partial_{j_o}$$

$c_{j_1 j_2 \dots}(\mathbf{z})$ polynomials in $\mathbf{z}$

Such that $\boxed{\hat{A}\,u(\mathbf{z}) = 0}$

$$\hat{A}\,u(\mathbf{z}) = 0$$

For any annihilator $\hat{A}$, we have infinitely many integral identities

$$\int \mathrm{d}^n\mathbf{z}\,\varphi_\alpha(\mathbf{z})\,\hat{A}\,u(\mathbf{z}) = 0,\ \ \forall\alpha$$

symbolic α

Using IBPs on derivatives, we get a **template identity** for symbolic α

$$\int u\,(\varphi_\alpha c_0) - \sum_j \int u\,(\partial_j c_j \varphi_\alpha) + \cdots + (-1)^o \sum_{j_1\leq\cdots\leq j_o} \int u\,\partial_{j_1}\cdots\partial_{j_o}(c_{j_1\cdots j_o}\varphi_\alpha) = 0$$

All the integrals belong to the family I_α

Laporta algorithm

[Chetyrkin, Tkachov (1981); Laporta (2000)]

$$\int u\,(\varphi_\alpha c_0) - \sum_j \int u\,(\partial_j c_j \varphi_\alpha) + \cdots + (-1)^o \sum_{j_1 \leq \cdots \leq j_o} \int u\,\partial_{j_1} \cdots \partial_{j_o}(c_{j_1 \cdots j_o} \varphi_\alpha) = 0$$

Seeding the template eq.s : replacing symbolic α with integer numbers

- Applying each template identity to a large number of seed integrals: obtain a **linear system** of equations
- Choice of an **ordering**: express **complex** integrals as a function of **simple** ones
- solving it: reduction to **master integrals**★

$$I_\alpha = \sum_{\beta \in \mathrm{MIs}} c_{\alpha\beta} I_\beta$$

Computational bottleneck in state-of-the-art calculations

★ Additional relations may exist, such as symmetry relations

Properties of parametric annihilators

$\hat{A}$ is an annihilator $\Rightarrow$ $\mathbf{z}^\beta \hat{A}$ and $\partial_j \hat{A}$ are annihilators

Set of annihilators of $u(\mathbf{z})$ is a D-module

$$\int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\mathbf{z}^\beta \hat{A}\, u(\mathbf{z})\right) = \int \mathrm{d}^n\mathbf{z} \left(\mathbf{z}^\beta \varphi_\alpha(\mathbf{z})\right) \hat{A}\, u(\mathbf{z})$$

$$\int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\partial_j \hat{A}\, u(\mathbf{z})\right) = -\int \mathrm{d}^n\mathbf{z} \left(\partial_j \varphi_\alpha(\mathbf{z})\right) \hat{A}\, u(\mathbf{z})$$

$\varphi_\alpha(\mathbf{z})$ closed under monomial multiplication & differentiation

interested in a **minimal** set of **generators**:

set of annihilators $\hat{A}$ independent modulo linear combinations of $\mathbf{z}^\beta \hat{A}$ and $\partial_j \hat{A}$

Differential operators

Differential equations

[Barucchi, Ponzano (1973); Kotikov (1991); Bern, Dixon, Kosower (1994); Gehrmann, Remiddi (2000)]

- Integrals in the form of I_α also depend on additional **free parameters** x (e.g. kinematic invariants)
- Studying of **analytic structure** & their numerical or analytical **evaluation**
- Reducing the derivative of MIs with respect to x to MIs, write a **system of differential equations** satisfied by the MIs themselves

$$\partial_x I_\alpha = \sum_{\beta \in \mathrm{MIs}} M_{\alpha\beta} I_\beta, \qquad \text{for } \alpha \in \mathrm{MIs}.$$

x **free parameter of the integrals**

Differential equations via differential operators

- Derive an operator $\hat{O}_x$ that realizes differentiation with respect to x

$$\hat{O}_x = c_0^{(x)}(\mathbf{z}) + \sum_j c_j^{(x)}(\mathbf{z})\,\partial_j + \sum_{j_1 \leq j_2} c_{j_1 j_2}^{(x)}(\mathbf{z})\,\partial_{j_1}\partial_{j_2}$$

$$+ \cdots + \sum_{j_1 \leq \cdots \leq j_o} c_{j_1 \cdots j_o}^{(x)}(\mathbf{z})\,\partial_{j_1} \cdots \partial_{j_o}$$

$c_{j_1 j_2 \cdots}^{(x)}$ polynomials

Such that $\boxed{\hat{O}_x\, u(\mathbf{z}) = \partial_x\, u(\mathbf{z})}$

Differential equations via differential operators

- Explicitly, integrating by parts all derivatives in $\hat{O}_x$

$$\partial_x I_\alpha = \int (\hat{O}_x u)\,\varphi_\alpha + \int u\,(\partial_x \varphi_\alpha)$$

$$= \int u\,(\varphi_\alpha c_0^{(x)}) - \sum_j \int u\,(\partial_j c_j^{(x)} \varphi_\alpha) + \cdots$$

$$+ (-1)^o \sum_{j_1 \leq \cdots \leq j_o} \int u\,\partial_{j_1} \cdots \partial_{j_o} (c^{(x)}_{j_1 \cdots j_o} \varphi_\alpha) + \int u\,(\partial_x \varphi_\alpha)$$

- Our preferred way to compute derivatives
  - Alternatively: recurrence relations

How to compute parametric annihilators

Computing annihilators via linear constraints

- Computing parametric annihilators up to a certain order and polynomial degree
- implemented in the `CALICO` package

Step 1 : From annihilators to syzygies

$$\boxed{\mathbf{f}(\mathbf{z})} \cdot \boxed{\mathbf{g}(\mathbf{z})} = 0$$ Syzygy equation

known polynomials

$$\mathbf{f}(\mathbf{z}) = \{f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})\}$$

unknown polynomials

$$\mathbf{g}(\mathbf{z}) = \{g_1(\mathbf{z}), \ldots, g_n(\mathbf{z})\}$$

Syzygy sol.s form a module:

$\mathbf{g}^{(k)}(\mathbf{z})$ set of solutions $\Rightarrow \mathbf{g}(\mathbf{z}) = \sum_k p_j(\mathbf{z})\mathbf{g}^{(k)}(\mathbf{z})$ is also a solution

$p_j(\mathbf{z})$ arbitrary polynomials

Start from $\hat{A}\, u(\mathbf{z}) = 0 \;\Rightarrow$ Identify $c_{j_1,\ldots,j_k}(\mathbf{z})$ as the unknown polynomials $\mathbf{g}(\mathbf{z})$ of a syzygy $\mathbf{f}(\mathbf{z}) \cdot \mathbf{g}(\mathbf{z}) = 0$

$$\boxed{\frac{1}{u(\mathbf{z})}\hat{A}\, u(\mathbf{z}) = 0}$$

- Insert the expression for $\hat{A} = c_0(\mathbf{z}) + \sum c_j(\mathbf{z})\partial_j + \ldots$
- Collect LHS under a common denominator
- Imposing vanishing of the numerator $\Rightarrow$ obtain a syzygy equation for $c_{j_1,\ldots,j_k}(\mathbf{z})$

- Example : **first-order annihilator**

$$\hat{A} = c_0(\mathbf{z}) + \sum_j c_j(\mathbf{z})\,\partial_j \qquad\qquad u(\mathbf{z}) = \prod_j B_j(\mathbf{z})^{\gamma_j}$$

$$c_0(\mathbf{z}) \prod_k B_k(\mathbf{z}) + \sum_{j=1}^{n} c_j(\mathbf{z}) \sum_k \gamma_k \left(\partial_j B_k(\mathbf{z})\right) \prod_{l \neq k} B_l(\mathbf{z}) = 0$$

Form of a syzygy equation

$$\mathbf{f}(\mathbf{z}) \cdot \mathbf{g}(\mathbf{z}) = 0$$

Step 2: Solving syzygy equations via linear constraints

[Schabinger (2015)]

- Ansatz for the solution

$$\mathbf{g}(\mathbf{z}) = \sum_j \sum_\alpha c_{j\alpha}\, \mathbf{z}^\alpha\, \hat{e}_j$$

$|\alpha| \leq d$ for some max degree d

$\hat{e}_j$ unit vector in the j-th direction

- Plug it into $\mathbf{f}(\mathbf{z}) \cdot \mathbf{g}(\mathbf{z}) = 0$

$\Rightarrow \Big\{$ **Linear system for $c_{j\alpha}$**

- Impose coeff.s of each monomial $\mathbf{z}$ vanish

CALICO uses the efficient linear solver of FiniteFlow, based on finite-field methods

[Peraro (2019)]

Applications

Hypergeometric ${}_2F_1$

$$I_\alpha = \int_0^1 \mathrm{d}z\, \varphi_\alpha(z)\, u(z)$$

$$\varphi_\alpha(z) = z^\alpha, \qquad u(z) = z^{b_2-1}\,(1-z)^{b_3-b_2-1}\,(1-x\,z)^{-b_1}$$

Related to the hypergeometric ${}_2F_1$

$$I_\alpha = \frac{\Gamma(b_2+\alpha)\Gamma(b_3-b_2)}{\Gamma(b_3+\alpha)}\,{}_2F_1(b_1,\, b_2+\alpha,\, b_3+\alpha;\, x)$$

1. 1 first-order annihilator → reduction to 2 MIs $\{I_0, I_1\}$
2. First-order differential operator

$$\partial_x \begin{pmatrix} I_1 \\ I_0 \end{pmatrix} = \begin{pmatrix} \frac{b_1\,x - b_3}{(1-x)\,x} & \frac{b_2}{(1-x)\,x} \\ \frac{b_1 - b_3}{1-x} & \frac{b_2}{1-x} \end{pmatrix} \cdot \begin{pmatrix} I_1 \\ I_0 \end{pmatrix}$$

Generalised to Hypergeometric ${}_{n+1}F_n$

Loop integrals

Momentum-space representation

$$J_\alpha = J_{\alpha_1 \cdots \alpha_n} = \int \prod_{i=1}^{\ell} \frac{\mathrm{d}^d k_i}{i\pi^{d/2}} \frac{1}{D_1^{\alpha_1} \ldots D_n^{\alpha_n}}$$

$$D_{F,j} = l_j \cdot v_j - m_j^2$$

$$D_{F,j} = l_j^2 - m_j^2$$

l_j linear combination of k_j,
v_j linear combination of p_j

$D_j s$ are generalised denominators

- **Proper denominators:** D_j such that $\alpha_j > 0$
- Irreducible scalar products (ISPs): D_j such that $\alpha_j \leq 0$

IBPs in momentum space

[Tkachov (1981), Chetyrkin, Tkachov (1981)]

$$\int \prod_{i=1}^{\ell} \frac{\mathrm{d}^d k_i}{i\pi^{d/2}} \frac{\partial}{\partial k_j^\mu} \frac{v^\mu}{D_1^{\alpha_1} \ldots D_n^{\alpha_n}} = 0, \quad \text{with } v^\mu = k_i^\mu, p_i^\mu$$

Parametric representations of loop integrals

Baikov

$$I_\alpha = \int \mathrm{d}^n \mathbf{z} \, \frac{1}{\mathbf{z}^\alpha} \, B(\mathbf{z})^\gamma$$

Also **Loop-by-Loop Baikov** & **Duals of loop integrals**

Lee-Pomeransky

$$I_\alpha = \int \mathrm{d}^n \mathbf{z} \left(\prod_{j=1}^{n} \frac{1}{\Gamma(\alpha_j)} \right) \mathbf{z}^{\alpha-1} G(\mathbf{z})^{-d/2}$$

$$G(\mathbf{z}) = \mathcal{U}(\mathbf{z}) + \mathcal{F}(\mathbf{z})$$

Schwinger

$$I_\alpha = \int \mathrm{d}^n \mathbf{z} \left(\prod_{j=1}^{n} \frac{1}{\Gamma(\alpha_j)} \right) \mathbf{z}^{\alpha-1} \exp\left[-\mathcal{F}(\mathbf{z})/\mathcal{U}(\mathbf{z}) \right] \mathcal{U}(\mathbf{z})^{-d/2}$$

Baikov representation

$$I_\alpha = \int \mathrm{d}^n \mathbf{z}\, \varphi_\alpha(\mathbf{z})\, u(\mathbf{z})$$

Reminder

Baikov

$$I_\alpha = \int \mathrm{d}^n \mathbf{z}\, \frac{1}{\mathbf{z}^\alpha}\, B(\mathbf{z})^\gamma$$

Loop-By-Loop Baikov

$$I_\alpha = \int \mathrm{d}^n \mathbf{z}\, \frac{1}{\mathbf{z}^\alpha} \prod_{j=1}^{2\ell-1} B_j(\mathbf{z})^{\gamma_j}$$

Change of integration vars from loop momenta to generalised denominators

- Require a full set of ISPs
- $\deg B = \min(\ell + e, 2\ell)$
- Less variables in LBL (Baikov change of var.s is done one loop at a time)

Intersection theory in a nutshell

[Mastrolia, Mizera (2019)] [Brunello, Cacciatori, Caron-Huot, Chestnov, Crisanti, Duhr, Frellesvig, GF, Gasparotto, Giroux, Laporta, Maggio, Mandal, Mastrolia, Matsubara-Heo, Mattiazzi, Mizera, Munch, Peraro, Pokraka, Porkert, Semper, Smith, Sohnle, Stavinski, Takayama (2019 – Ongoing)]

Introduction of a scalar product, **intersection number**, between

- **Integrals I_α**

$$I_\alpha = \int \mathrm{d}^n\mathbf{z}\, \frac{1}{\mathbf{z}^\alpha}\, B(\mathbf{z})^\gamma$$

- **Dual integrals $I^\star_\alpha$**

$$I^\star_\alpha = \int \mathrm{d}^n\mathbf{z}\, \frac{1}{\mathbf{z}^\alpha}\, B(\mathbf{z})^{-\gamma}$$

- Calculation of intersection numbers $\longrightarrow$ standard procedure: **recursive** algorithm in the integration var.s

- Focus on **dual integrals**
  - Linear relations
  - Differential equations (necessary step in the intersection numbers calculation)

Dual integrals & regulators

$$I^\star_\alpha = \int \mathrm{d}^n\mathbf{z}\,\frac{1}{\mathbf{z}^\alpha}\,B(\mathbf{z})^{-\gamma}$$

$$\left.\begin{array}{l} \bullet\ (z_1,\ldots,z_m)\ \text{proper denominators} \\ \bullet\ (z_{m+1},\ldots,z_n)\ \text{ISPs} \end{array}\right\}\ \alpha_j \le 0 \text{ for } j>m$$

Problem: need of additional regulators $\frac{1}{z_j^{\alpha_j}} \to \frac{1}{z_j^{\alpha_j-\rho_j}}$ for $j \le m$

[GF, Peraro (2023)]

$$I^\star_\alpha = \underbrace{\prod_{j=1}^{m}\rho_j^{\Theta(\alpha_j-1/2)}\int \mathrm{d}^n\mathbf{z}\,\frac{1}{\mathbf{z}^{\alpha-\rho}}}_{\varphi_\alpha(\mathbf{z})}\,\underbrace{B(\mathbf{z})^{-\gamma}}_{u(\mathbf{z})}$$

work on the leading coefficients $\rho_j \to 0$

Reduction of dual integrals

- Method of annihilator can be applied straightforwardly (Baikov, LBL Baikov)
- Tmp ids require the knowledge of $\varphi_\alpha(\mathbf{z})$ under
  - Multiplication by monomials

$$z_j^{\beta_j}\,\varphi_\alpha(\mathbf{z}) = \begin{cases} \varphi_{\alpha-\beta_j\hat{e}_j}(\mathbf{z}) & \text{if } \alpha_j > \beta_j \text{ or } \alpha_j \leq 0 \\ 0 & \text{if } 0 < \alpha_j \leq \beta_j . \end{cases}$$

  - Differentiation

$$\partial_j\,\varphi_\alpha(\mathbf{z}) = \begin{cases} -(\alpha_j - \delta_{\alpha_j 0})\,\varphi_{\alpha+\hat{e}_j}(\mathbf{z}) & \text{if } j \leq m \\ -\alpha_j\,\varphi_{\alpha+\hat{e}_j}(\mathbf{z}) & \text{if } j > m. \end{cases}$$

- Approach tested for both
  - Reduction of dual integrals
  - Computation of DEQ satisfied by dual integrals in fewer variables
- Successful comparison on all examples of [GF, Peraro (2023)]

Lee-Pomeransky and Schwinger rep.s

Lee-Pomeransky

$$I_\alpha = \int \mathrm{d}^n\mathbf{z} \left(\prod_{j=1}^{n} \frac{1}{\Gamma(\alpha_j)} \right) \mathbf{z}^{\alpha-1} G(\mathbf{z})^{-d/2}$$

$$G(\mathbf{z}) = \mathcal{U}(\mathbf{z}) + \mathcal{F}(\mathbf{z})$$

Schwinger

$$I_\alpha = \int \mathrm{d}^n\mathbf{z} \left(\prod_{j=1}^{n} \frac{1}{\Gamma(\alpha_j)} \right) \mathbf{z}^{\alpha-1} \exp\left[-\mathcal{F}(\mathbf{z})/\mathcal{U}(\mathbf{z}) \right] \mathcal{U}(\mathbf{z})^{-d/2}$$

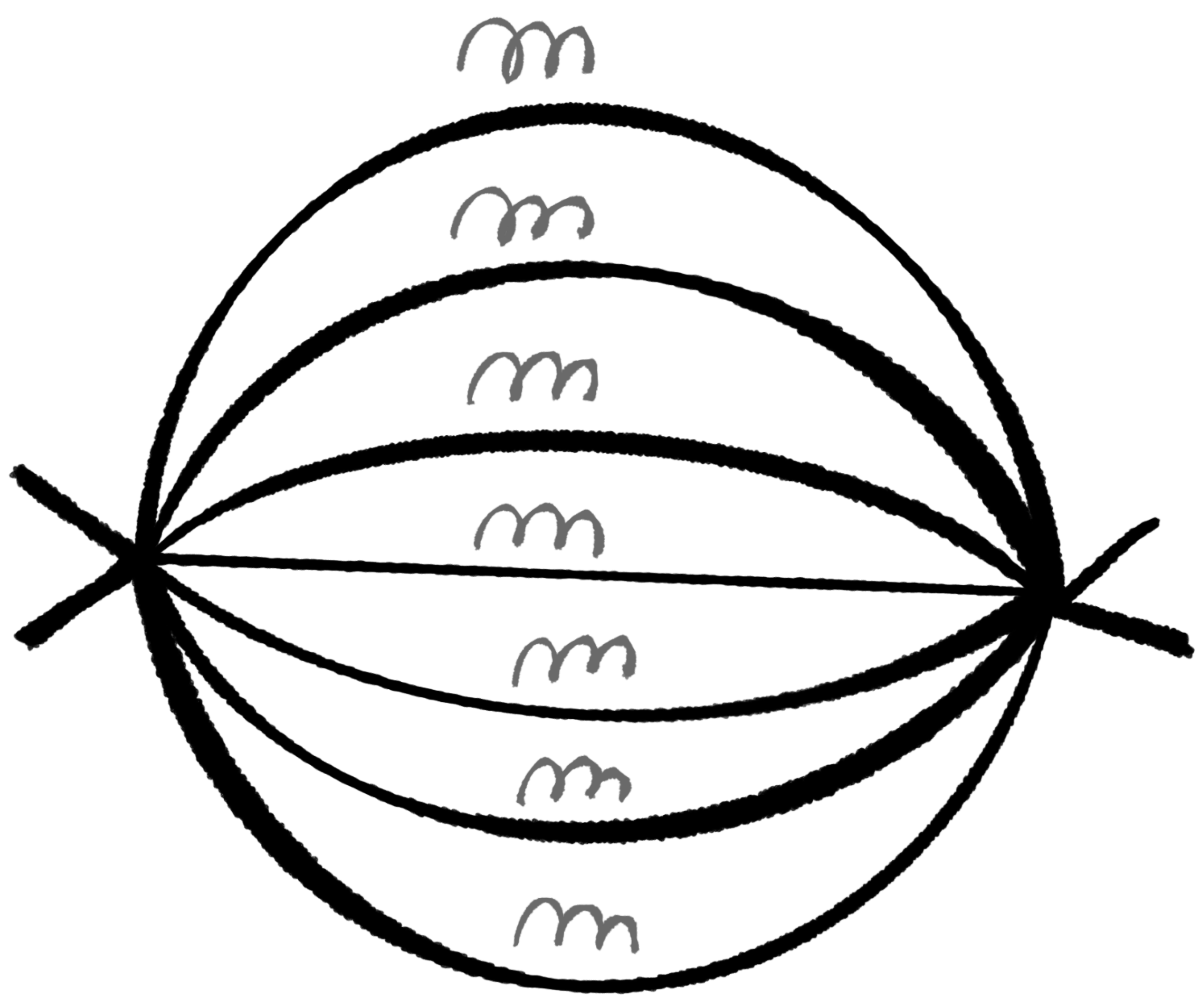
- n integration variables (# number of proper denominators)
- no need of introducing ISPs (can be added if required)
- $\deg G = \ell + 1$ at ℓ loops

- twist includes exp. factor
- empirically:
  - annihilators have lower degree
  - More frequent use of 2nd order ann.
  - tmp id.s often fewer and simpler

L loop bananas

ℓ-loop & one internal mass m, defined by the set of $\ell + 1$ proper denominators:

$$D_j = k_j^2 - m^2 \quad \text{for } j = 1,\ldots,\ell$$
$$D_{\ell+1} = (k_1 + \cdots + k_\ell - p)^2 - m^2$$



Momentum space has $(\ell + 2)(\ell - 1)/2$ ISPs → for $\ell = 6$, it has 20 ISPs

Use Lee-Pomeranski or Schwinger representation to

- Reduce to MIs
- Derive DEQs

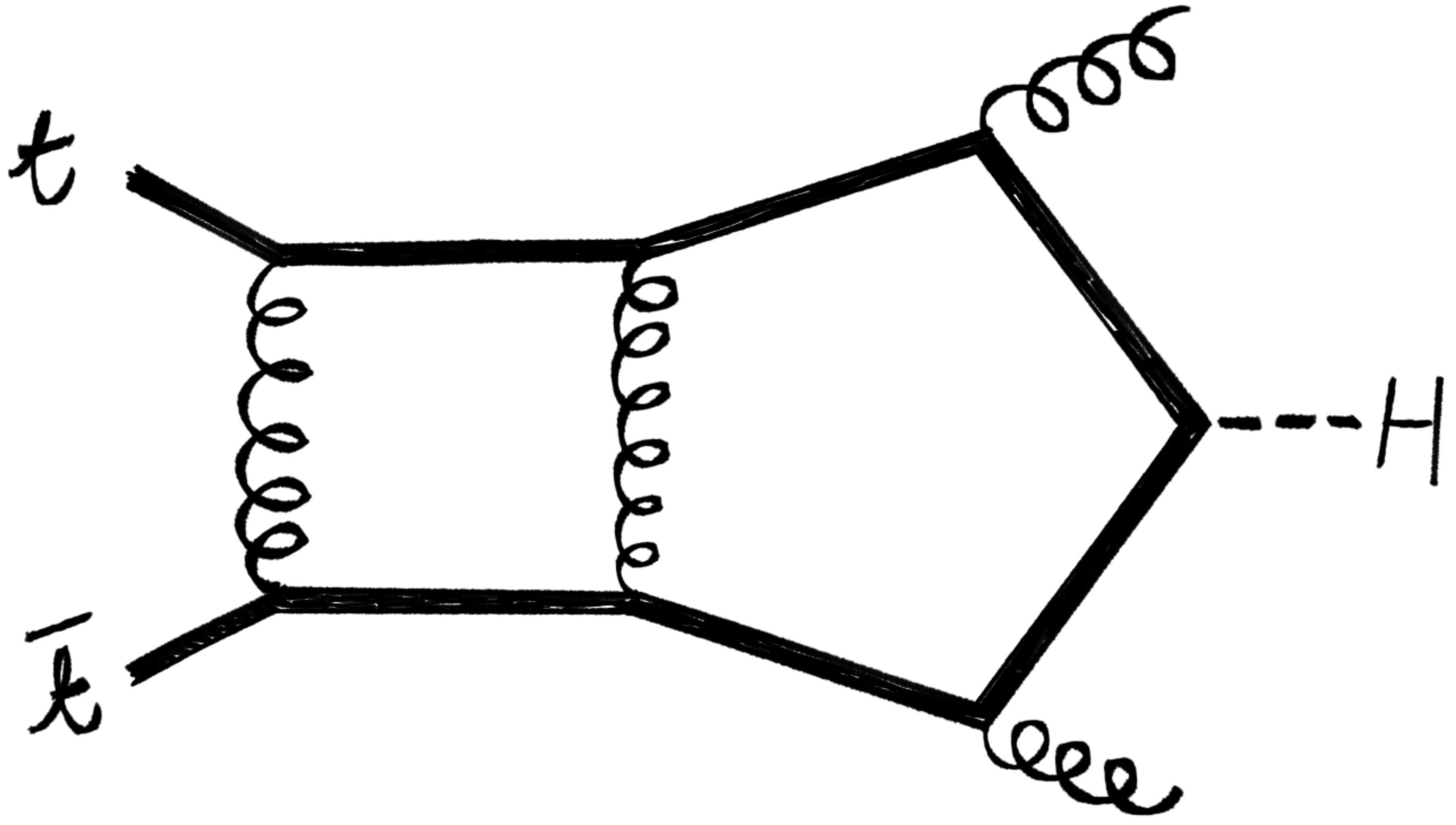
Without the need of additional ISPs!

Done in a couple of minutes on a laptop up to 6 loops (mostly spent computing annihilators and template eqs)

Family for $t\bar{t}H$ production

Cutting-edge example

- Many different scales
- Many external legs



Using the integral representations:

- Schwinger (More efficient!)
- Lee-Pomeranski

Tested: numerical reduction on a laptop with up to 3 extra powers of denominators

Finding relations between integrals with constant numerator and higher powers of proper denominators

Useful for finding integrals with good properties

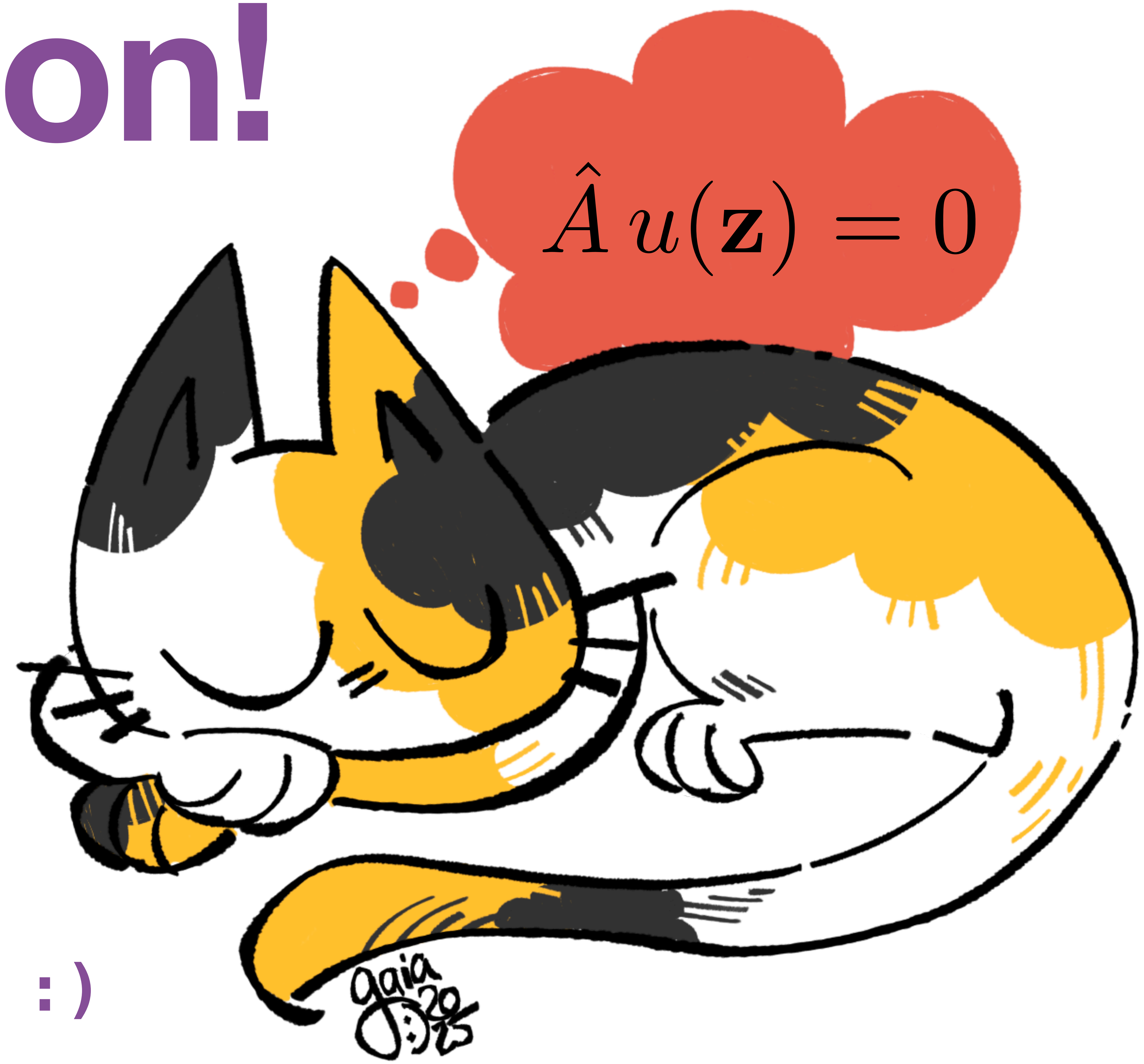
- Quasi-finite [von Manteuffel, Panzer, Schabinger (2014)]
- Pure functional form [Henn (2013)]

Conclusions & Outlook

- Parametric annihilators are a useful tool for finding linear relations between integrals
- Allow to use integral parametrizations tailored to specific problems
  - New applications to LBL Baikov, duals of loop integrals, Schwinger parametrisation
- Introduced similar technique for deriving differential equations
- Implementation will be released in the public package `CALICO`

★**Bonus**: can also solve syzygy equations and polynomial decomposition problems

Thank you for your attention!



… more drawings @qftoons :)

Backup

L loop bananas — CALICO code snippet

This example computes differential equations (DEs) for the L-loop equal mass banana integral family, using the Schwinger representation.

In[1]:=
```
<<CALICO`
```

All public symbols exported by the package are prefixed by
CAT (“Computing AnnihilaTors”)

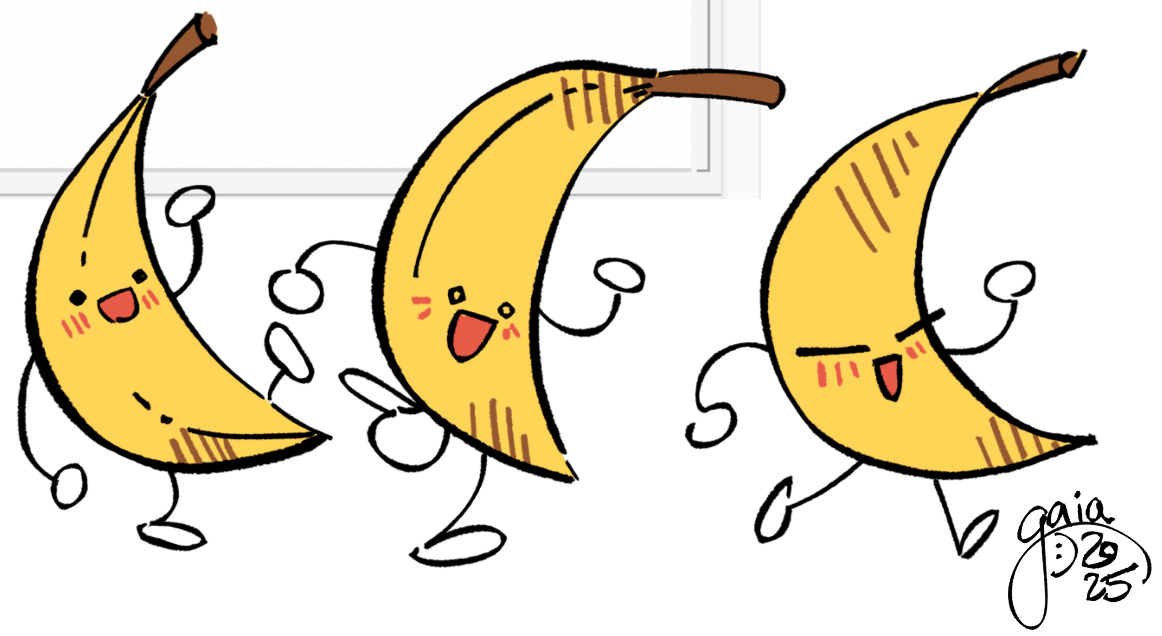
Set up of the integral family

You can set the number of loops here. The example has been tested up to L=6.

```
In[2]:= L=4

Out[2]= 4
```

Define loop denominators and the twist:

```
In[3]:= loopmoms = k/@Range[L];
dens = Join[ Table[{loopmoms[[ii]],m2},{ii,1,L}], {{Sum[loopmoms[[ii]],{ii,1,L}]-p,m2}} ]

Out[4]= {{k[1], m2}, {k[2], m2}, {k[3], m2}, {k[4], m2}, {-p + k[1] + k[2] + k[3] + k[4], m2}}

In[5]:= {u,f,g,zs} = CATUFGPolys[
  dens,
  k/@Range[L],
  {p^2 → s},
  z
];
```



Twist output: uses symbolic polynomials, whose analytic expression is substituted at a later stage

full analytic expression can also be used instead

```
In[6]:= twist = CATSchwinger[u,f,d,zs]
```

Out[6]=

$$\text{CATTwist}\Big[e^{-\frac{\text{CATFPoly}[z[1],z[2],z[3],z[4],z[5],m2,s]}{\text{CATUPoly}[z[1],z[2],z[3],z[4],z[5],m2,s]}}\,\text{CATUPoly}[z[1],z[2],z[3],z[4],z[5],m2,s]^{-d/2},$$

$$\{\text{CATFPoly}\to(\#1^2\,\#2\,\#3\,\#4\,\#6+\#1\,\#2^2\,\#3\,\#4\,\#6+\#1\,\#2\,\#3^2\,\#4\,\#6+\#1\,\#2\,\#3\,\#4^2\,\#6+$$
$$\#1^2\,\#2\,\#3\,\#5\,\#6+\#1\,\#2^2\,\#3\,\#5\,\#6+\#1\,\#2\,\#3^2\,\#5\,\#6+\#1^2\,\#2\,\#4\,\#5\,\#6+\#1\,\#2^2\,\#4\,\#5\,\#6+$$
$$\#1^2\,\#3\,\#4\,\#5\,\#6+5\,\#1\,\#2\,\#3\,\#4\,\#5\,\#6+\#2^2\,\#3\,\#4\,\#5\,\#6+\#1\,\#3^2\,\#4\,\#5\,\#6+$$
$$\#2\,\#3^2\,\#4\,\#5\,\#6+\#1\,\#2\,\#4^2\,\#5\,\#6+\#1\,\#3\,\#4^2\,\#5\,\#6+\#2\,\#3\,\#4^2\,\#5\,\#6+\#1\,\#2\,\#3\,\#5^2\,\#6+$$
$$\#1\,\#2\,\#4\,\#5^2\,\#6+\#1\,\#3\,\#4\,\#5^2\,\#6+\#2\,\#3\,\#4\,\#5^2\,\#6-\#1\,\#2\,\#3\,\#4\,\#5\,\#7\,\&),$$
$$\text{CATUPoly}\to(\#1\,\#2\,\#3\,\#4+\#1\,\#2\,\#3\,\#5+\#1\,\#2\,\#4\,\#5+\#1\,\#3\,\#4\,\#5+\#2\,\#3\,\#4\,\#5\,\&)\}\Big]$$

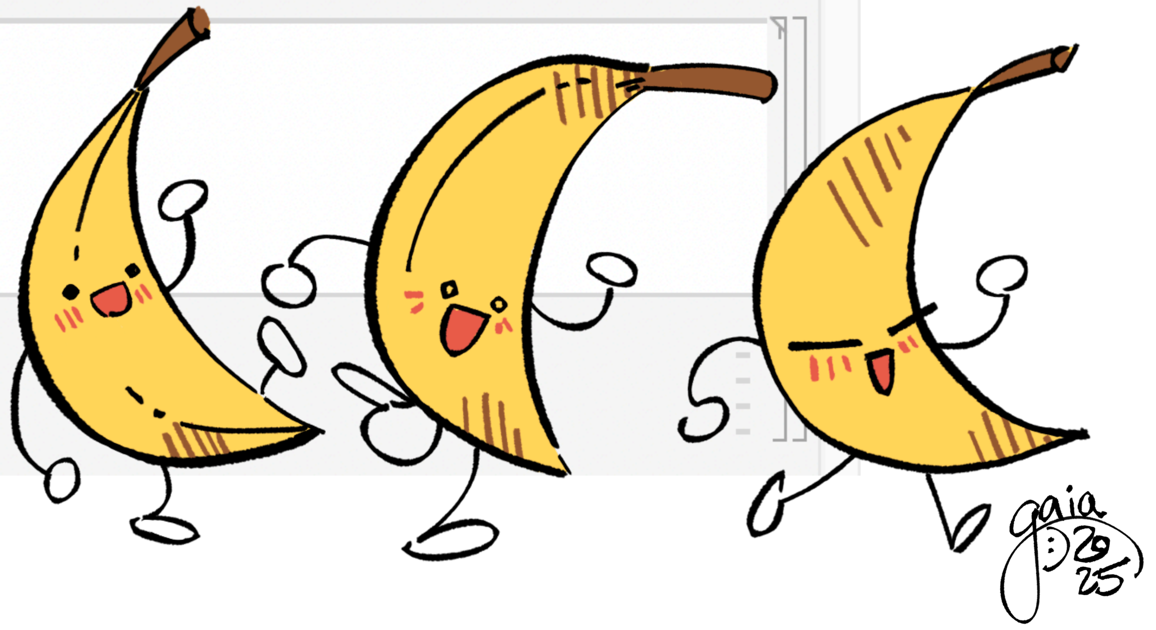
```
In[28]:= (* computing first-order annihilators*)
maxdegree = L+1;
maxorder = 1;
ann = CATAnnihilator[twist,zs,maxdegree,maxorder]
```

Returns the polynomials $c_{j_1 j_2 \ldots}(z)$

```
Out[30]= {{{}, {}, {{d z[4] + 2 m2 z[4]^2 - d z[5] - 2 m2 z[5]^2, 0, 0, 0, 2 z[4]^2, -2 z[5]^2},
   {d z[3] + 2 m2 z[3]^2 - d z[5] - 2 m2 z[5]^2, 0, 0, 2 z[3]^2, 0, -2 z[5]^2},
   {d z[2] + 2 m2 z[2]^2 - d z[5] - 2 m2 z[5]^2, 0, 2 z[2]^2, 0, 0, -2 z[5]^2},
   {d z[1] + 2 m2 z[1]^2 - d z[5] - 2 m2 z[5]^2, 2 z[1]^2, 0, 0, 0, -2 z[5]^2}}, {},
  {{d z[2] × z[3] × z[4] + 2 m2 z[1] × z[2] × z[3] × z[4] + 2 d z[2] × z[3] × z[5] +
     2 m2 z[1] × z[2] × z[3] × z[5] + 2 d z[2] × z[4] × z[5] + 2 m2 z[1] × z[2] × z[4] × z[5] +
     2 d z[3] × z[4] × z[5] + 2 m2 z[1] × z[3] × z[4] × z[5] + (5 m2 - s) z[2] × z[3] × z[4] × z[5] +
     d z[2] z[5]^2 + d z[3] z[5]^2 + 2 m2 z[2] × z[3] z[5]^2 + d z[4] z[5]^2 + 2 m2 z[2] × z[4] z[5]^2 +
     2 m2 z[3] × z[4] z[5]^2 + 2 m2 z[2] z[5]^3 + 2 m2 z[3] z[5]^3 + 2 m2 z[4] z[5]^3,
    2 z[1] × z[2] × z[3] × z[4] + 2 z[1] × z[2] × z[3] × z[5] + 2 z[1] × z[2] × z[4] × z[5] +
     2 z[1] × z[3] × z[4] × z[5] + z[2] × z[3] × z[4] × z[5],
    z[2] × z[3] × z[4] × z[5], z[2] × z[3] × z[4] × z[5], z[2] × z[3] × z[4] × z[5],
```

etc,etc, . . .

```
In[13]:= (*degree → number of first order annihilators*)
Table[ deg → Length[ann⟦1,deg+1⟧], {deg,0,Length[ann⟦1⟧]-1}]

Out[13]= {0 → 0, 1 → 0, 2 → 4, 3 → 0, 4 → 5, 5 → 0}
```



In[31]:=
```
(*check at order 2*)
maxdegree = 2;
maxorder = 2;
ann2 = CATAnnihilator[twist,zs,maxdegree,maxorder,"MinOrder"→2,"KnownSolutions"→ann]
```

Out[33]=
```
{{}, {{}, {}, {{2 (-4 d + 5 d^2) + 4 (-m2 + 2 d m2) z[1] +
   4 (-m2 + 2 d m2) z[2] + 4 m2^2 z[1] × z[2] + 4 (-m2 + 2 d m2) z[3] +
   4 m2^2 z[1] × z[3] + 4 m2^2 z[2] × z[3] + 4 (-m2 + 2 d m2) z[4] + 4 m2^2 z[1] × z[4] +
   4 m2^2 z[2] × z[4] + 4 m2^2 z[3] × z[4] + (13 d m2 - d s) z[5] + 4 m2^2 z[1] × z[5] +
   4 m2^2 z[2] × z[5] + 4 m2^2 z[3] × z[5] + 4 m2^2 z[4] × z[5] + 2 (5 m2^2 - m2 s) z[5]^2,
  4 (-1 + 2 d) z[1] + 4 m2 z[1] × z[2] + 4 m2 z[1] × z[3] + 4 m2 z[1] × z[4] +
   d z[5] + 4 m2 z[1] × z[5] + 2 m2 z[5]^2, 4 (-1 + 2 d) z[2] + 4 m2 z[1] × z[2] +
   4 m2 z[2] × z[3] + 4 m2 z[2] × z[4] + d z[5] + 4 m2 z[2] × z[5] + 2 m2 z[5]^2,
```
etc,etc, . . .

Merging the two solutions

In[23]:=
```
(*combine all*)
ann = CATAnnihilatorMerge[ann,ann2];
```

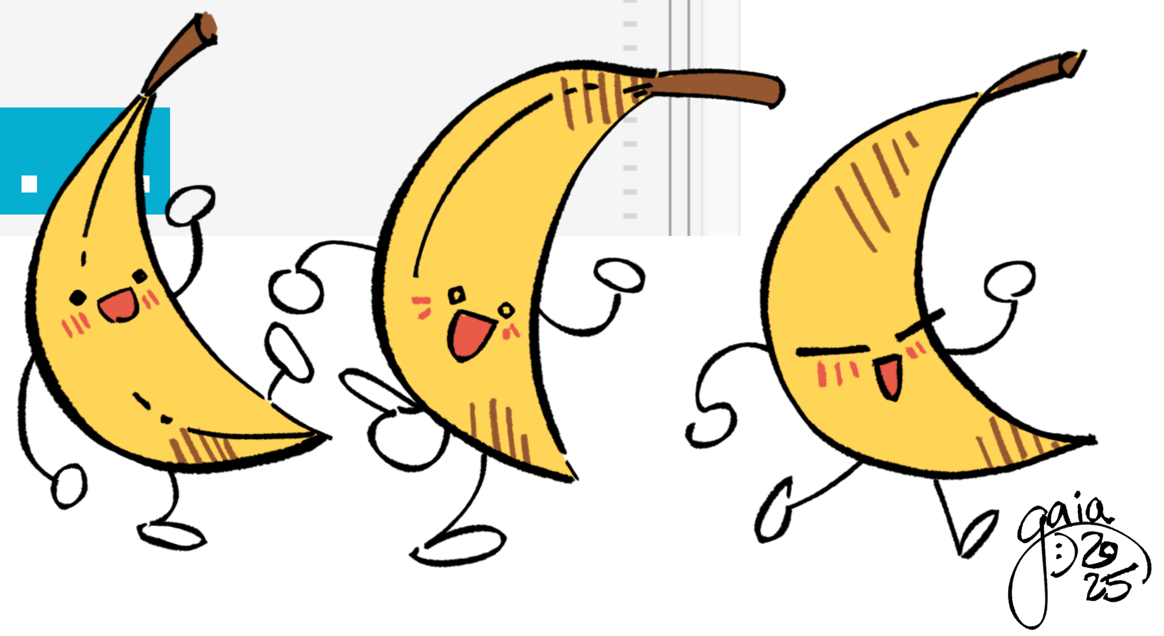
In[24]:=
```
(*degree → number of second order annihilators*)
Table[ deg → Length[ann[[2,deg+1]]], {deg,0,Length[ann[[2]]]-1}]
```

Out[24]=
```
{0 → 0, 1 → 0, 2 → 1}
```

Template equations

```
In[27]:= tmpeq = CATSchwingerIdsFromAnnihilators["family",ann,zs]

Out[27]= {2 m2 CATInt[family, {#1, #2, #3, 2 + #4, #5}] #4 (1 + #4) +
    CATInt[family, {#1, #2, #3, 1 + #4, #5}] #4 (2 - d + 2 #4) -
    2 m2 CATInt[family, {#1, #2, #3, #4, 2 + #5}] #5 (1 + #5) -
    CATInt[family, {#1, #2, #3, #4, 1 + #5}] #5 (2 - d + 2 #5),
   2 m2 CATInt[family, {#1, #2, 2 + #3, #4, #5}] #3 (1 + #3) +
    CATInt[family, {#1, #2, 1 + #3, #4, #5}] #3 (2 - d + 2 #3) -
    2 m2 CATInt[family, {#1, #2, #3, #4, 2 + #5}] #5 (1 + #5) -
    CATInt[family, {#1, #2, #3, #4, 1 + #5}] #5 (2 - d + 2 #5),
   2 m2 CATInt[family, {#1, 2 + #2, #3, #4, #5}] #2 (1 + #2) +
    CATInt[family, {#1, 1 + #2, #3, #4, #5}] #2 (2 - d + 2 #2) -
    2 m2 CATInt[family, {#1, #2, #3, #4, 2 + #5}] #5 (1 + #5) -
    CATInt[family, {#1, #2, #3, #4, 1 + #5}] #5 (2 - d + 2 #5),
   2 m2 CATInt[family, {2 + #1, #2, #3, #4, #5}] #1 (1 + #1) +
    CATInt[family, {1 + #1, #2, #3, #4, #5}] #1 (2 - d + 2 #1) -
    2 m2 CATInt[family, {#1, #2, #3, #4, 2 + #5}] #5 (1 + #5) -
    CATInt[family, {#1, #2, #3, #4, 1 + #5}] #5 (2 - d + 2 #5),
```

etc,etc, . . .



Here we use FiniteFlow to solve the system, following section 6.1 of https://arxiv.org/pdf/1905.08019.
This example is simple enough that we can also easily find reduction tables with

FFSparseSolve[Thread[eqs==0],ints,"NeededVars"->neededints]

but the overall strategy below is generally more efficient, especially in more complex cases.

FiniteFlow to solve the tmp eq.s & derive DEQ

```
In[33]:= FFNewGraph["g","in",params]
Out[33]= True

In[34]:= FFAlgSparseSolver["g","ibps",{"in"},params,Thread[eqs==0],ints,
170      "NeededVars"→neededints]
Out[34]= True

In[35]:= FFSolverOnlyHomogeneous["g","ibps"]

In[36]:= FFGraphOutput["g","ibps"]

In[37]:= ibplearn = FFSparseSolverLearn["g",ints];
```

The independent unknowns of the system are the master integrals (MIs)

Master integrals

```
In[38]:= mis = "IndepVars"/.ibplearn
Out[38]= {CATInt[family, {2, 2, 2, 1, 1}],
          CATInt[family, {2, 2, 1, 1, 1}], CATInt[family, {2, 1, 1, 1, 1}],
          CATInt[family, {1, 1, 1, 1, 1}], CATInt[family, {1, 1, 1, 1, 0}]}
```

Differential equations via recurrence relations

$$\partial_x I_\alpha = \int \mathrm{d}^n\mathbf{z} \left(\partial_x \varphi_\alpha(\mathbf{z})\right) B(\mathbf{z})^\gamma + \gamma \int \mathrm{d}^n\mathbf{z}\, \varphi_\alpha(\mathbf{z}) \left(\partial_x B(\mathbf{z})\right) B(\mathbf{z})^{\gamma-1}$$

$\in$ **family** $I_\alpha = \int \varphi_\alpha B^\gamma$

$\in$ **shifted integral family** $I_\alpha = \int \varphi_\alpha B^{\gamma-1}$

- Need to derive a recurrence relation to shift $\gamma - 1 \to \gamma$
- Easy shift: $\gamma \to \gamma + 1$

$$I_\alpha^{(\gamma+1)} = \int \mathrm{d}^n\mathbf{z} \left(B(\mathbf{z})\varphi_\alpha(\mathbf{z})\right) B(\mathbf{z})^\gamma \quad \Rightarrow \quad I_\alpha^{(\gamma+1)} = \sum_{\beta \in \mathrm{MIs}} R^{(+)}_{\alpha\beta}(\gamma)\, I_\beta, \;\; \alpha \in \mathrm{MIs}$$

- Shift back $\gamma \to \gamma - 1$

$$I_\alpha^{(\gamma-1)} = \sum_{\beta \in \mathrm{MIs}} R^{(-)}_{\alpha\beta}(\gamma)\, I_\beta, \;\; \alpha \in \mathrm{MIs}$$

$$R^{(-)}_{\alpha\beta}(\gamma) = \left[R^{(+)}(\gamma-1)\right]^{-1}_{\alpha\beta}$$

Annihilators & syzygy

Syzygy-based approach

- Example: $u(\mathbf{z}) = B(\mathbf{z})^\gamma$
- IBP: $$0 = \sum_{j=1}^{n} \int \mathrm{d}^n\mathbf{z}\, \partial_j \Big(a_j(\mathbf{z}) \varphi_\alpha(\mathbf{z}) B(\mathbf{z})^\gamma \Big)$$

Does not belong to the family I_α

$$= \int \mathrm{d}^n\mathbf{z} \left[B(\mathbf{z})^\gamma \sum_{j=1}^{n} \partial_j \Big(\varphi_\alpha(\mathbf{z}) a_j(\mathbf{z}) \Big) + B(\mathbf{z})^\gamma \varphi_\alpha(\mathbf{z}) \boxed{\left(\gamma \frac{1}{B(\mathbf{z})} \sum_{j=1}^{n} a_j(\mathbf{z}) \partial_j B(\mathbf{z}) \right)} \right]$$

- Choosing $$\boxed{\sum_{j=1}^{n} a_j(\mathbf{z}) \partial_j B(\mathbf{z}) = b(\mathbf{z})\, B(\mathbf{z})}$$

Syzygy equation to determine $a_j(\mathbf{z}), b(\mathbf{z})$

- Solution $a^\star(\mathbf{z}), b^\star(\mathbf{z})$ $$\int \mathrm{d}^n\mathbf{z}\, B(\mathbf{z})^\gamma \left[\sum_{j=1}^{n} \partial_j \Big(\varphi_\alpha(\mathbf{z}) a_j^\star(\mathbf{z}) \Big) + \gamma \varphi_\alpha(\mathbf{z}) b^\star(\mathbf{z}) \right] = 0$$

[Larsen, Zhang (2017); Sameshima (2019)]

First-order annihilator approach

[Bertolini (2024)]

- First-order annihilator

$$\hat{A} = c_0(\mathbf{z}) + \sum_j c_j(\mathbf{z})\partial_j$$

- Considering

$$B(\mathbf{z})^{1-\gamma}\hat{A}u(\mathbf{z}) = 0$$

- Condition on the $c_j(\mathbf{z})$s

$$c_0(\mathbf{z})\,B(\mathbf{z}) + \gamma\sum_{j=1}^{n} c_j(\mathbf{z})\,\partial_j B(\mathbf{z}) = 0$$

same as

$$\sum_{j=1}^{n} a_j(\mathbf{z})\partial_j B(\mathbf{z}) = b(\mathbf{z})\,B(\mathbf{z})$$

choosing

$$b(\mathbf{z}) = -c_0(\mathbf{z}),\;\; a_j(\mathbf{z}) = \gamma\, c_j(\mathbf{z})$$

Drawing connections

By inserting $\quad b(\mathbf{z}) = -c_0(\mathbf{z}),\;\; a_j(\mathbf{z}) = \gamma\, c_j(\mathbf{z})$

in $$\int \mathrm{d}^n\mathbf{z}\, B(\mathbf{z})^\gamma \left[\sum_{j=1}^{n} \partial_j\Big(\varphi_\alpha(\mathbf{z}) a_j^\star(\mathbf{z})\Big) + \gamma \varphi_\alpha(\mathbf{z}) b^\star(\mathbf{z})\right] = 0$$

We get same id.s as $$\int u\,(\varphi_\alpha c_0) - \sum_j \int u\,(\partial_j c_j \varphi_\alpha) = 0$$

sygyzy method yields the same integral identities as first-order annihilators

- More syzgy equations can be used to constrain the form of the identities that are generated [Gluza, Kajda, Kosower (2010);Ita (2015);Larsen, Zhang (2015); Wu, Bohem, Ma, Xu, Zhang (2023)]. Additional constraints can be applied to both the solutions of syzygy and annihilators of any order o

- Annihilator method generalizes the one based on syzygy eq.s
  - Syzygy $\Longleftrightarrow$ annihilators if $o = 1$

THE END