Modern Summation Techniques Applied to Loop Integrals

Mark Round

DESY

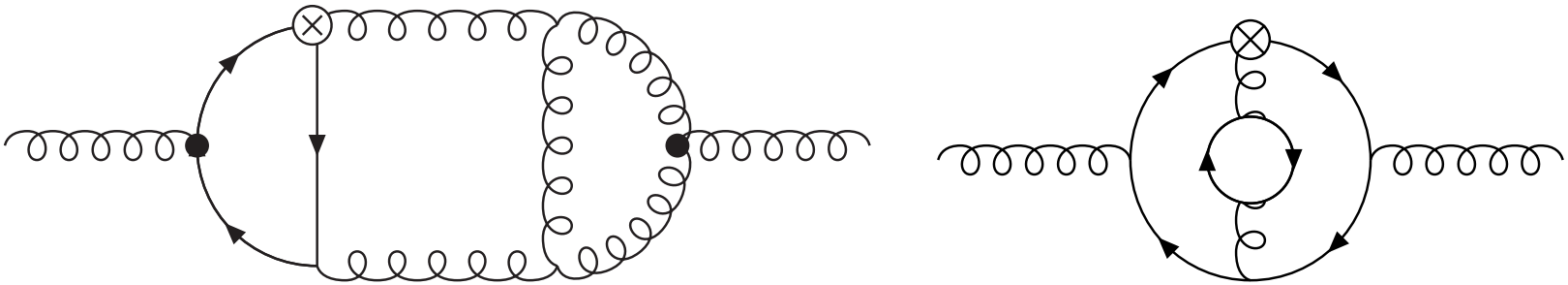
Talk Outline

1. A Particle Physics Motivation
2. Existing Summation Methods
3. A Refined Holonomic Approach
4. QCD Example Sum Results
5. Recent Application
6. Summary

High Precision QFT Calculations

QCD

- Consider perturbative QFT. Such as corrections to the high energy proton sub-structure from heavy quarks at 3-loops (and for $p^2 >> m^2_{\rm HQ}$). Bierenbaum, Blümlein, Klein '09



- Such diagrams yield nasty integrals which are extremely hard to work with. A simpler form is needed. Our goal is to simplify the object.
- Physically speaking, the only restriction on the Feynman diagrams is that they contain atmost one mass.

By 'simplify' parameter integrals here think of expressing the integrals in terms of some basis functions such as rational functions, the harmonic numbers,

$$S_1(n) = \sum_i^n \frac{1}{i},$$

their generalisations and other chosen objects.

Symbolic summation algorithms are significantly stronger than integration algorithms (at the moment!).
Therefore if we can obtain a sum representation of an integral we can exploit powerful routines using a computer to express the input sums in terms of 'simple' sums such as the harmonic sum.
The driving force between these ideas is that ultimately a computer will perform the summation.

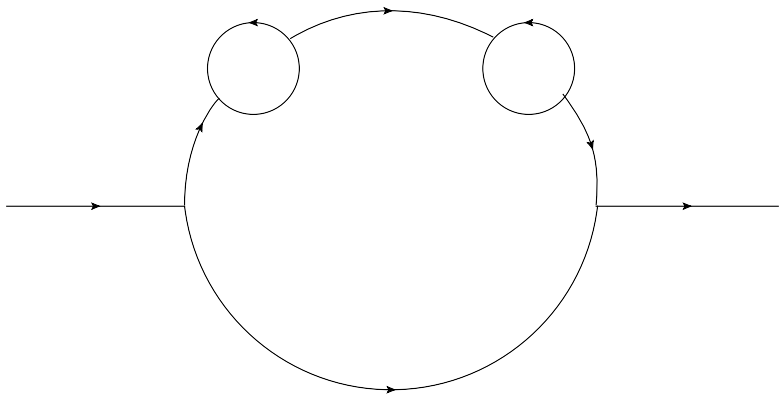
High Precision QFT Calculations

First Algorithmic Steps

- Momentum integrals are straightforward but the Feynman parameters integrals are not. Notice the operator insertion (the $\otimes$ in the diagram) will give us a Mellin parameter, n. Diagrams obey homogeneous difference equations in n. Blümlein, Kauers, Klein, Schneider '09
- A first step in trying to understand the remaining integrals is to convert them into definite nested hypergeometric and harmonic multi-sums. Blümlein & Kurth '98 and Vermaseren '98
- There is an algorithm, under the assumptions listed, to do this!

  Blümlein, Klein, Schneider & Stan '12

As a specific example of a diagram that contributes to the heavy 3-loop Wilson coefficients consider diagrams of the following topology class,



(Almost) all such diagrams have been computed at 3-loops; for the heavy fermion diagrams using symbolic summation techniques. Manuscript in preparation

An algorithm to obtain a sum representation for the example topology is straightforward,

- Introduce Feynman Parameters
- Integrate momenta
- Recognise integrals as being special functions

Other more complicated topologies require a more sophisticated algorithm!

Applying the Feynman parameters to an individual diagram of the bubble class will give parameter integrals that lead to special functions; for example,

$$ {}_2F_1(a,b,c;z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \int_0^1 x^{b-1}(1-x)^{c-b-1}(1-zx)^{-a}dx $$

$$ {}_3F_2\left[\begin{matrix} a_1, a_2, c \\ b_1, d \end{matrix}; z\right] = \frac{\Gamma(d)}{\Gamma(c)\Gamma(d-c)} \int_0^1 x^{c-1}(1-x)^{d-c-1} {}_2F_1(a_1,a_2,b_1;xz)dx $$

are frequently encountered for the bubble topology.

Special functions frequently admit sum representations. For the example topology class we frequently need,

$$ {}_2F_1(a,b,c;z) = \sum_{i=0}^{\infty} \frac{a_i b_i}{c_i} \frac{z^i}{i!} $$

$$ {}_3F_2\left[\begin{matrix} a_1, a_2, c \\ b_1, d \end{matrix}; z\right] = \sum_{i=0}^{\infty} \frac{(a_1)_i (a_2)_i c_i}{(b_1)_i d_i} \frac{z^i}{i!} $$

where,

$$ x_n = \frac{\Gamma(x+n)}{\Gamma(x)}. $$

Thus we arrive at a sum representation for the diagram.

- Introduce binomial expansions of Feynman parameter brackets,

$$(x_1 + x_2)^n, \quad n \in \mathbb{Z}$$

  to simplify parameter integration.
- Apply the residue theorem to evaluate a Mellin-Barnes integral.
- ...

Statement of Problem

Integrals over Feynman parameters can be converted into sums, for example through the methods discussed.
The integration problem has been transformed into a summation problem.
Now attempt to use symbolic summation tools to express the sum form of the diagram in terms of basis functions.
Today I will talk about a new summation algorithm.

Statement of Problem

Defintions

A definite sum is of the form,

$$\sum_{i=0}^{n} f(n,i)$$

and a multi-sum is nested if,

$$\sum_{i}^{m}\sum_{j}^{i} f(n,i,j).$$

We will be interested in definite nested multi-sums which are of the form,

$$\sum_{i}^{n}\sum_{j}^{i} f(n,i,j)$$

Statement of Problem

Any given Feynman diagram (with atmost one mass) can be expressed as a definite nested multi-sum,

$$F_n(\epsilon) = \sum_{i_1=\alpha_1}^{L_1(n)} \cdots \sum_{i_m=\alpha_m}^{L_m(n,i_1,\ldots i_{m-1})} \sum_{j_1=\beta_1}^{\infty} \cdots \sum_{j_N=\beta_N}^{\infty} f(\epsilon, n, i_1, \ldots, i_m, j_1, \ldots, j_n).$$

Aim

Given a definite nested multi-sum find, in terms of indefinite nested multi-sums, another (simpler) representation accurate to a given order in ϵ.

The dimensional regularisation parameter is denoted by ϵ and n is a Mellin parameter.

Statement of Problem

Illustration ($<$ 10 minutes of CPU time)

$$\sum_{i=2}^{n+1}\sum_{j=2}^{n+2-i}\sum_{a=0}^{\infty}\sum_{b=0}^{\infty}\frac{\binom{n+2}{i}\binom{n+2-i}{j}(-1)^{i+j}B[i,j](j+i-1)}{(j+i+a+b)(i+j+a+b-1)(j+b)(i+b)}$$

$$\begin{aligned}
= \quad & \frac{2n^3+5n^2+4n-2}{n+1}S_2+\left[\frac{5}{2}S_2-\frac{n^2+2n+2}{n+1}\right]S_1^2-2nS_{2,1} \\
- \quad & 2n(n+1)\zeta_3-S_1^3+\frac{1}{4}S_1^4-\frac{1}{4}S_2^2+2(n+1)S_3-\frac{1}{2}S_4-2S_{3,1} \\
+ \quad & [4\zeta_3+(2n-1)S_2+2S_3-2(n+1)-4S_{2,1}]\,S_1+2S_{2,1,1}
\end{aligned}$$

where $S_{a,\ldots,b}=S_{a,\ldots,b}(n)$ & $B[x,y]=\Gamma[x]\Gamma[y]/\Gamma[x+y]$ for convenience and $\epsilon=0$ for this sum.

- We have gone from a tough to simple, if long, expression.

Existing Tools

Sigma

- To solve summation problems one needs a general toolkit to apply. Of the examples available, we will use a collection of algorithms implemented in Mathematica. Schneider '04, '05... related to Karr '81 & Chyzak '00
- Sigma computes linear recurrences with polynomial coefficients.
- The scaling of CPU time with 'input difficulty' is dangerously high.
- We must adopt algorithms that 'divide and conquer' summation problems.
- In addition, one must massage inputs to dial scaling parameters thus making a computation possible.

Existing Tools

Sigma

Consider a summation problem,

$$\sum_{i=0}^{n} F(i).$$

Suppose that there exists a function G,

$$F(i) = G(i+1) - G(i), \text{(telescoping equation)}$$

$$\begin{aligned}\sum_{i=0}^{n} F(i) &= \sum_{i=0}^{n} G(i+1) - G(i), \\ &= G(n+1) - G(0)\end{aligned}$$

which is analogous to integration. The telescoping equation can be solved by numerous technologies depending on the properties of F. Sigma is perhaps the most powerful such technology available at the moment.

Sigma works by encoding recurrences into a difference-field. A DF is a field $\mathbb{F}$ and a shift operator σ,

$$\sigma(h(x)) = h(x+1)$$
$$\sigma(\alpha) = \alpha \quad \forall \alpha \in \mathbb{K}.$$

In this language, $f \in \mathbb{F}$ and we search for a $g \in \mathbb{F}$ such that,

$$\sigma(g) - g = f.$$

Choose the harmonic numbers for F and let f represent the harmonic numbers in the difference field. Then,

$$F(i) = \sum_{j=1}^{i} \frac{1}{j}$$

$$F(i+1) = F(i) + \frac{1}{i+1},$$

$$\sigma f = f + \frac{1}{i+1}$$

We should also define i,

$$\sigma i = i + 1$$

Sigma computes a denominator bound on $g \in \mathbb{F} = \mathbb{K}(i)(f)$. Then Sigma computes a degree bound. Finally Sigma solves the linear system for the coefficients of the numerator and denominator polynomials.
Sigma can also solve the more general creative telescoping equation,

$$\sigma g - g = \sum_{k=0}^{m} c_k f_k$$

where the f_k correspond to shifts in another parameter, $F(n+k, i)$.
Solutions are now in the form of recurrences,

$$G(n+1) - G(0) = \sum_{k=0}^{m} c_k \left(\sum_{i=0}^{n} F(n+k, i) \right)$$

Existing Tools

Sigma

Normally in the literature the creative telescoping solution is written using a notation for the sum,

$$\mathcal{S}(n) = \sum_{i=0}^{n} F(i).$$

Then we can write,

$$G(n+1) - G(0) = c_0\mathcal{S}(n) + c_1\mathcal{S}(n+1) + \ldots + c_m\mathcal{S}(n+m)$$

or even in the notation of sequences,

$$\mathcal{G}(n) = p_0 a_n + \ldots + p_m a_{n+m}$$

In this talk $\mathcal{G}$ will be rational and may include sums and the p_i will be polynomials – examples later.

Existing Tools

Direct Approach

$$F_n(\epsilon) = \sum_{i_1=\alpha_1}^{L_1(n)} \cdots \sum_{i_m=\alpha_m}^{L_m(n,i_1,\ldots i_m)} \sum_{j_1=\beta_1}^{\infty} \cdots \sum_{j_N=\beta_N}^{\infty} f(\epsilon, n, \{i\}, \{j\})$$

$$\downarrow$$

Apply MultiSum package (tough)

$$\downarrow$$

$$g_0(\epsilon, n)F_n \; + \; \ldots \; + \; g_r(\epsilon, n)F_{n+r} \quad = \quad G(\epsilon, n)$$

Developed and used for example by Wegschaider '08, Blümlein, Klein, Schneider & Stan '12

Existing Tools

Sum by Sum Approach

$$F_n(\epsilon) = \sum_{i_1=\alpha_1}^{L_1(n)} \cdots \sum_{i_m=\alpha_m}^{L_m(n,i_1,\ldots i_m)} \sum_{j_1=\beta_1}^{\infty} \cdots \sum_{j_N=\beta_N}^{\infty} f(\epsilon, n, \{i\}, \{j\})$$

Zoom to the innermost sum

$$\hat{F}_n(\epsilon) = \sum_{j_N=\beta_N}^{\infty \text{ or } L_N} f(\epsilon, n, \{i\}, \{j\})$$

Find a recurrence for the inner sum,

$$g_0(\epsilon, n)\hat{F}_n + \ldots + g_r(\epsilon, n)\hat{F}_{n+r} = G(\epsilon, n)$$

Solve Recurrence and substitute in solution

Refined Holonomic Approach

$$F_n(\epsilon) = \sum_{i_1=\alpha_1}^{L_1(n)} \cdots \sum_{i_m=\alpha_m}^{L_m(n,i_1,\ldots i_m)} \sum_{j_1=\beta_1}^{\infty} \cdots \sum_{j_N=\beta_N}^{\infty} f(\epsilon, n, \{i\}, \{j\})$$

Zoom to the innermost sum

$$\hat{F}_n(\epsilon) = \sum_{j_N=\beta_N}^{\infty \text{ or } L_N} f(\epsilon, n, \{i\}, \{j\})$$

Replace innermost sum with the sequence and its recurrences

Find a recurrence for the inner sum,

$$g_0(\epsilon, n)\hat{F}_n + \ldots + g_r(\epsilon, n)\hat{F}_{n+r} = G(\epsilon, n)$$

A Worked Example

Consider the following definite nested multi-sum,

$$F_n = \sum_{i_1=0}^{n-2} \sum_{i_2=0}^{n-i_1-2} \frac{4 i_1! (-1)^{i_2} (n - i_1 - 1)(1 + i_2)!}{(1 + i_1 + i_2)^2 (2 + i_1 + i_2)^2 (i_1 + i_2)!} \binom{n - i_1 - 2}{i_2}$$

To find a recurrence for F_n, first re-write the sum,

$$4 \sum_{i_1=0}^{n-2} i_1! (n - i_1 - 1) a_{i_1,n}$$

$$a_{i_1,n} = \sum_{i_2=0}^{n-i_1-2} \frac{(-1)^{i_2} (1 + i_2)!}{(1 + i_1 + i_2)^2 (2 + i_1 + i_2)^2 (i_1 + i_2)!} \binom{n - i_1 - 2}{i_2}$$

now apply a recurrence finding algorithm, e.g. Sigma.

A Worked Example

One finds that,

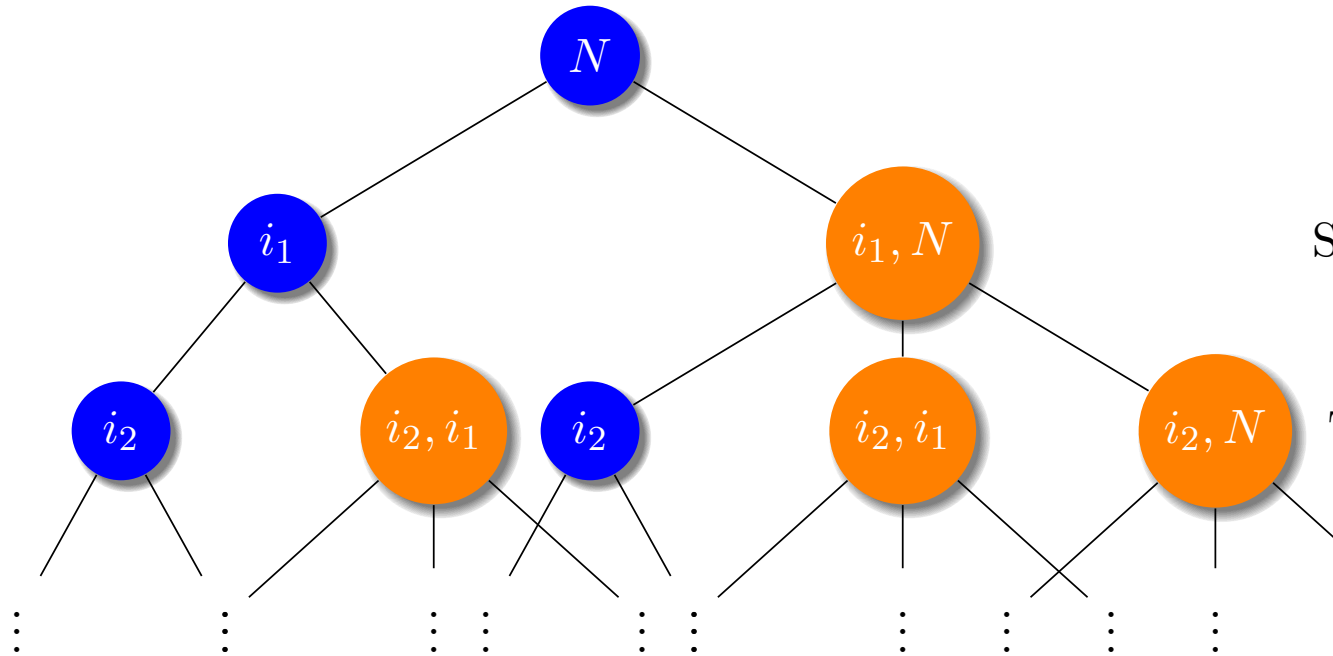
$$(2+i_1+n+i_1 n)a_{i_1,n} - (2+i_1-n)(1+i_1+n)a_{i_1+1,n} = \frac{1}{(1+i_1^2)i_1!},$$
$$-(1+i_1-n)(1+2i_1+i_1 n)a_{i,n} - (1+n)(1+i_1+i_1 n)a_{i,n+1} = -\frac{1}{(1+n)i_1!}.$$

- The two recurrences plus initial conditions specify the summand — a **holonomic** sequence.
- The right-hand sides are non-zero, which is usuall called **'refined'**.

Combining them one finds, for example, a zero-order recurrence,

$$F_n = \frac{4n}{2+n} - \frac{8S_1(n)}{(1+n)(2+n)}, \quad S_1(n) = \sum_{i=1}^{n} \frac{1}{i}.$$

Recurrence Tree



Input Sum, $F = \sum_{i_1=0}^{N} a_{i_1}$

Second Sum, $a_{i_1} = \sum_{i_2=0}^{i_1} b_{i_2}$

Third Sum, $b_{i_2} = \sum_{i_3=0}^{i_2} c_{i_3}$

Conceptual Advantages

- Consider a sum, in the refined approach, defined by a zeroth-order recurrence. It has the most complicated rhs possible. Essentially that is the sum-by-sum approach!
- At the other extreme, the class of problem sums obey homogeneous (rhs $=0$) recurrences of order ~ 35.
- The refined approach allows the structure of the problem to be distributed between the order of the recurrence and the complexity of the recurrence right-hand side.

This freedom is a motivating concept to explore the refined approach.

Conceptual Advantages

- Using the sum-by-sum approach one must expand the summand in ϵ as a first step. Such an expansion blows up the size of the summand quickly.
- By holding the epsilon expansion inside the recurrence one can compute, with a small extra overhead, all the coefficients in an expansion at once.
- One is motivated to propose that at higher and higher orders in ϵ the refined approach will become increasingly quicker than the sum-by-sum approach.
  This scaling is a second motivating concept to explore the refined approach.

Conceptual Advantages

Example Sum

Consider the following double sum,

$$F_n(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j1=0}^{n-3-j_0} \frac{(-1)^{j_1}(1+j_1)(j_0+j_1)!\left(1-\frac{\epsilon}{2}\right)_{j_0}\left(3-\frac{\epsilon}{2}\right)_{j_1}}{(4-\epsilon)_{j_0+j_1}\left(4+\frac{\epsilon}{2}\right)_{j_0+j_1}} \binom{n-j_0-2}{j_1+1}$$

This is the appropriate input for the refined approach. For the sum-by-sum approach one must expand the summand.

$$F_n(\epsilon) = F_n^0(\epsilon) + \epsilon F_n^1(\epsilon) + \ldots$$

$$F_n^0(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j1=0}^{n-3-j_0} \frac{(-1)^{j_1}(1+j_1)(j_0+j_1)!\,(1)_{j_0}\,(3)_{j_1}}{(4)_{j_0+j_1}\,(4)_{j_0+j_1}} \binom{n-j_0-2}{j_1+1}$$

$$F_n^1(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j1=0}^{n-3-j_0} \frac{3(-1)^{j_1} j_0!(2+j_1)!(1+j_1)(j_0+j_1)!}{((3+j_0+j_1)!)^2} \binom{n-j_0-2}{j_1+1}$$

$$\times\left(1+\frac{3}{1+j_1}+\frac{3}{2+j_1}-\frac{3}{1+j_0+j_1}-\frac{3}{2+j_0+j_1}-\frac{3}{3+j_0+j_1}\right.$$

$$\left.+3S_1(j_0)+3S_1(j_1)-3S_1(j_0+j_1)\right)$$

Computing Recurrences

Effects of Recurrence Order

- Passing Sigma sums that are defined in terms of high-order recurrences generically decreases the speed drastically. For example, a sum defined in terms of an order 8 recurrence is simply too high to compute with.
- In addition, it is often true that recurrence order grows. A sum defined by a fourth order recurrence will probably obey a 4+ order recurrence.
- In this way one can lose control of the tree of recurrences for a multi-sum making the algorithm unworkable.

Computing Recurrences

Handling Right-hand Sides

Passing Sigma a sum, defined in terms of recurrences accurate to some order in ϵ, one obtains a recurrence to the requested order in ϵ.

$$g_2(\epsilon, j, n)a_{j+2} + \ldots + g_0(\epsilon, j, n)a_j = 3j^2 + \sum_{i=1}^{j} \frac{1}{n+i+j} + \epsilon \sum_{i=1}^{j} \frac{S_1(n)}{j(j-1+i)}$$

- Generically the right-hand side of the recurrence, $G(\epsilon, n)$, will contain definite nested multi-sums which must be converted to indefinite objects using either algorithm.
  (Unprocessed right-hand sides are extremely slow to work with.)
- Processing the right-hand side can be trivial or tough.

In practice finding and simplifying a recurrence take similar amounts of time due to optimisation.

One is looking to balance several factors.

1. Time to compute recurrences.
2. Time to simplify recurrences.
3. Growth of computation time for later recurrences (controlling the tree)

Current Standard Lore

Here is our current generic thinking,

1. Try to compute a recurrence, with a simple right-hand side ($\Rightarrow$ high-order), provided the recurrence is no more than order 2 and that the system of equations hidden in Sigma takes less than x minutes to solve. (Naïve difference field theory)
2. Else, compute a recurrence of minimal order; no restrictions on right-hand side. (Improved difference field theory)
3. Simplify right-hand side, expand in ϵ.

Illustrative Result

It is not correct to think of one approach as quickest/best in general. Take the example sum,

$$S = \sum_{i_1=0}^{n-2} \sum_{i_2=0}^{n-i_1-2} \frac{4 i_1! (-1)^{i_2} (n - i_1 - 1)(1 + i_2)!}{(1 + i_1 + i_2)^2 (2 + i_1 + i_2)^2 (i_1 + i_2)!} \left(\begin{array}{c} n - i_1 - 2 \\ i_2 \end{array} \right)$$

The refined approach is 4 times faster than the current 'state-of-the-art' sum-by-sum algorithm and some examples can be as much as 10 times faster.

Comparison

- Experimental tests show that large, complicated input sums e.g. quintuple sums are quicker with the sum-by-sum approach. The tree of recurrences can not be well controlled (yet!).
- Small sums are quicker with the refined approach e.g. double or triple sums. However the refined approach is far from an exhausted method at the moment.

QCD Sum Examples

Easy Double Sum

$$F_n(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j1=0}^{n-3-j_0} \frac{(-1)^{j_1}(1+j_1)(j_0+j_1)!\left(1-\frac{\epsilon}{2}\right)_{j_0}\left(3-\frac{\epsilon}{2}\right)_{j_1}}{(4-\epsilon)_{j_0+j_1}\left(4+\frac{\epsilon}{2}\right)_{j_0+j_1}} \binom{n-j_0-2}{j_1+1}$$

Computed to leading order in ϵ takes 30 seconds with the refined approach and 180 seconds with the sum-by-sum approach.

The innermost sum obeys a second order homogeneous recurrence

$$g_2(\epsilon, j_0, n)a_{j_0+2,n} + g_1(\epsilon, j_0, n)a_{j_0+1,n} + g_0(\epsilon, j_0, n)a_{j_0,n} = 0$$

where

$$\begin{aligned} g_0 &= (n - j_0 - 3)(j_0 + 1)(2j_0 + 2 - \epsilon) \\ g_1 &= (2nj_0 - 2n + \epsilon n - 4j_0^2 - j_0\epsilon - 16j_0 - 18 - \epsilon + \epsilon^2)(n - j_0 - 2) \\ g_2 &= 2(n - j_0 - 3)(j_0 - n + 2)(j_0 + 2 + \epsilon) \end{aligned}$$

The whole sum obeys a zeroth order recurrence.

$$g_0(\epsilon, n) F_n = G(\epsilon, n)$$

where

$$g_0(\epsilon, n) = -(2+\epsilon)(4+\epsilon)(n+2-\epsilon)(n+1-\epsilon)$$

$$G(\epsilon, n) = 36(4-n)(n+1)$$

$$\times \frac{(2+n)(36+n(n(205+n(n(n(n-11)+52)-134))-179))}{n^2(n-3)^2(n-2)^2(n-1)^2}$$

$$+\epsilon\Bigg[6n(n(n(n(n(-n(n(n(n(n(n(n(5n-108)+1004)-5238)+16628)$$

$$-31758)+29602)+13158)-85481)+145890)-166526)+128184)$$

$$+27648)28080)-5184)-\frac{72}{n}(n+3)S_1(n)\Bigg]$$

QCD Sum Examples

Tough Triple Sum

$$\sum_{j=0}^{n-2}\sum_{j_1=0}^{j-2}\sum_{j_2=1}^{n-j+j_1-2}\frac{(-1)^{j_1+j_2}j^2(j+j_1-2)!(j_1-\epsilon+3)!(1+j_2)!}{(j-j_1+j_2)!(n-j+j_1+2-2\epsilon)!}$$
$$\times\frac{(n-j-\epsilon-2)!\left(n-j+j_1+2-\frac{\epsilon}{2}\right)!\left(n-j+j_1-j_2+1-\frac{\epsilon}{2}\right)!}{\left(n-j+j_1+4+\frac{\epsilon}{2}\right)!\left(n-j+j_1+2+\frac{\epsilon}{2}\right)!}$$
$$\times\binom{j-2}{j_1}\binom{n-j+j_1-2}{j_2}$$

In this case all the recurrences are first order. Computed from order ϵ^{-1} to ϵ^2 takes 3,500 seconds with the refined approach and 4,750 seconds with the sum-by-sum approach.

Epilogue

- The refined holonomic summation approach was put into a Mathematica package, ρSum, and was used to solve almost all the bubble topologies shown earlier. (10,000s of sums in total)
- There is a manuscript in preparation on the refined holonomic algorithm and the technical details of optimisation.

Summary

1. A refined approach to multi-summation has the capability to interpolate between existing multi-summation techniques.
2. With regards to ϵ expansions, there are good reasons to pursue the refined approach.
3. Although in the earlier stages of development, the method can already compete with (mature) exisiting technologies in non-trivial examples.

Definition of Summand

Definition

Let $\{f(n,k)\}_{n\geq 0,k\geq 0}$ be a multivariate (here bivariate) sequence over a field $\mathbb{F}$. f is hypergeometric in n and k if $\exists d \in \mathbb{Z}^+$ and $r_1(x,y), r_2(x,y) \in \mathbb{F}(x,y)$ such that $f(n+1,k)/f(k) = r_1(n,k)$ and $f(n,k+1)/f(n,k) = r_2(n,k)\,\forall n,k \geq d$.

Definition

Restrict to sequences where:
$\exists d \in \mathbb{Z}^+$, $a_i, b_i, m_i \in \mathbb{Z}$ and $c_i \in \mathbb{F}$ for $i \in [1,r]$ where $a_i n + b_i k + c_i \notin \mathbb{Z}^- \forall n,k \geq d$ and $\exists\, p(x,y), q(x,y) \in \mathbb{F}[x,y]$ where $q \neq 0$ factors linearly over $\mathbb{F}$:
$f(n,k) = p(n,k)/q(n,k) \prod_{i=1}^{r} \Gamma(a_i n + b_i k + c_i)^{m_i}\, \forall n,k \geq d$.

n.b. $\mathbb{Z}^{\pm}$ both include zero.

Statement of Problem

Summary

Allow the summand to be as defined, times harmonic numbers and linear combinations of such terms.

$$F_n(\epsilon) = \sum_{i_1=\alpha_1}^{L_1(n)} \cdots \sum_{i_m=\alpha_m}^{L_m(n,i_1,\ldots i_m)} \sum_{j_1=\beta_1}^{\infty} \cdots \sum_{j_N=\beta_N}^{\infty} f(\epsilon, n, i_1, \ldots, i_m, j_1, \ldots, j_N)$$

where $n \in \mathbb{N}$ and $n \geq n_0 \in \mathbb{N}$, all the upper bounds, L_k, are integer-linear in n, $\{i\}$ and $\{j\}$. All the lower bounds are specified constants, $\{\alpha\}, \{\beta\} \in \mathbb{N}$. The summand, F, is hypergeometric with respect to n, $\{i\}$ and $\{j\}$.

Aim

Find an expression for a definite nested multi-sum, using the outlined summand, in terms of indefinite nested multi-sums.

Return