

# New Multi-Sum Algorithms for Feynman Integrals

Mark Round

Research Institute for Symbolic Computation (RISC)  
Johannes Kepler Universität  
Linz, Austria



LHCPhenoNet Mid-Term Meeting, Ravello

LHCphenOnet

In collaboration with Carsten Schneider and Johannes Blümlein.

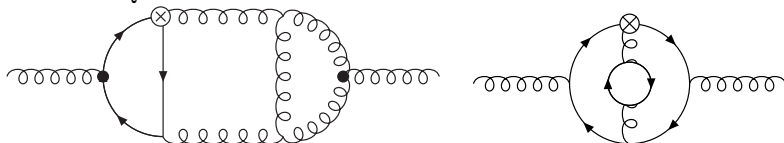
# Talk Outline

- ① Sums in Particle Physics
- ② Existing Summation Methods
- ③ A Refined Holonomic Approach
- ④ QCD Sum Results
- ⑤ 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  $p^2 \gg m_{HQ}^2$ ). 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 at most one mass.

# 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, KAUSERS, 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 to do this! BLÜMLEIN, KLEIN, SCHNEIDER & STAN '12

# Statement of Problem

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

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

## Aim

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

The dimensional regularisation parameter is denoted by  $\epsilon$  and  $n$  is a Mellin parameter.

# Statement of Problem

Illustration (< 10 minutes of CPU time)

$$\begin{aligned} & \sum_{i=2}^{n+1} \sum_{j=2}^{n+2-i} \sum_{a=0}^{\infty} \sum_{b=0}^{\infty} \frac{\binom{n+2}{i} \binom{n+2-i}{j} (-1)^{i+j} B[i, j] (j+i-1)}{(j+i+a+b)(i+j+a+b-1)(j+b)(i+b)} \\ &= \frac{2n^3 + 5n^2 + 4n - 2}{n+1} S_2 + \left[ \frac{5}{2} S_2 - \frac{n^2 + 2n + 2}{n+1} \right] S_1^2 - 2n S_{2,1} \\ &- 2n(n+1) \zeta_3 - S_1^3 + \frac{1}{4} S_1^4 - \frac{1}{4} S_2^2 + 2(n+1) S_3 - \frac{1}{2} S_4 - 2S_{3,1} \\ &+ [4\zeta_3 + (2n-1) S_2 + 2S_3 - 2(n+1) - 4S_{2,1}] S_1 + 2S_{2,1,1} \end{aligned}$$

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

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

# Experimental Nature of the Subject

Consider integration,

$$\int_0^1 x(1-x)dx \quad \int_0^1 x^\alpha(1-x)dx \quad \int_0^1 x^\alpha(1-x)^\beta dx$$

The first two integrals are trivial to perform, the third can not be expressed in terms of elementary functions.

- Problems become much tougher as the number of free parameters increases and difficulty is not clear by inspection.
- By inspecting the integral's form one can not see if the structure fits one algorithm or another.
- At first glance, it is not obvious how a given algorithm will perform on integrals (or sums).

# Experimental Nature of the Subject

- Feynman parameter integrals have proved to be very tough; sum representations of such integrals are correspondingly tough.
- Choosing the optimal algorithm, that contains the correct mathematical ideas, is very important.
- One would like a toolkit of summation algorithms that capture the different types of mathematical structure that can arise in Feynman diagrams.

# 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 co-efficients.
- 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

## Direct Approach

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

↓  
Apply MultiSum package (tough)

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

Developed and used for example by WEGSCHAIDER '08, BLÜMLEIN, KLEIN, SCHNEIDER & STAN '12

# Existing Tools

## Sum by Sum Approach

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

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

Solve Recurrence and  
substitute in solution

# Refined Holonomic Approach

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

Represent innermost  
sum with its recur-  
rences and its initial  
values

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

# A Worked Example

Consider the following definite nested multi-sum,

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

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,

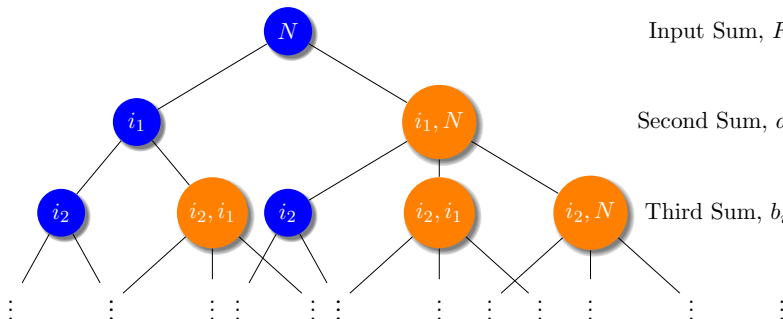
$$\begin{aligned}(2+i_1+n+i_1n)a_{i_1,n} - (2+i_1-n)(1+i_1+n)a_{i_1+1,n} &= \frac{1}{(1+i_1^2)i_1!}, \\ -(1+i_1-n)(1+2i_1+i_1n)a_{i,n} - (1+n)(1+i_1+i_1n)a_{i,n+1} &= -\frac{1}{(1+n)i_1!}.\end{aligned}$$

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

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 ( $\text{rhs} = 0$ ) recurrences of order  $\lesssim 35$ .
- 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  $\epsilon$  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 co-efficients in an expansion at once.
- One is motivated to propose that at higher and higher orders in  $\epsilon$  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,

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

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

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

# Conceptual Advantages

## Example Sum

$$F_n^0(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j_1=0}^{n-3-j_0} \frac{(-1)^{j_1} (1+j_1)(j_0+j_1)! (1)_{j_0} (3)_{j_1}}{(4)_{j_0+j_1} (4)_{j_0+j_1}} \binom{n-j_0-2}{j_1+1}$$
$$F_n^1(\epsilon) = \sum_{j_0=0}^{n-5} \sum_{j_1=0}^{n-3-j_0} \frac{3(-1)^{j_1} j_0! (2+j_1)! (1+j_1)(j_0+j_1)!}{((3+j_0+j_1)!)^2} \binom{n-j_0-2}{j_1+1}$$
$$\times \left( 1 + 3S_1(2+j_0) + 3S_1(j_1) - 3S_1(3+j_0+j_1) \right)$$

# 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  $\epsilon$ , one obtains a recurrence to the requested order in  $\epsilon$ .

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

- 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 rhs are usually outside the algorithm's input class.)
- 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.

# Computing Recurrences

## Managing the Recurrence Tree

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)

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  $\epsilon$ .

# Illustrative Result

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

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

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.

- 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.

# Applications to Perturbative QCD

In the last few months the summations algorithm has been applied to problems arising in QCD. Several diagrams that can not be found in the literature have been attempted. The algorithm achieved full solutions to several Feynman diagrams. That gave further areas of development and refinement.

# QCD Sum Examples

## Easy Double Sum

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

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

# QCD Sum Examples

## Easy Double Sum

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

$$g_0 = (n - j_0 - 3)(j_0 + 1)(2j_0 + 2 - \epsilon)$$

$$g_1 = (2nj_0 - 2n + \epsilon n - 4j_0^2 - j_0\epsilon - 16j_0 - 18 - \epsilon + \epsilon^2)(n - j_0 - 2)$$

$$g_2 = 2(n - j_0 - 3)(j_0 - n + 2)(j_0 + 2 + \epsilon)$$

# QCD Sum Examples

## Easy Double Sum

The whole sum obeys a zeroth order recurrence.

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

where

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

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

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

$$+ \epsilon \left[ 6n(n(n(n(n(-n(n(n(n(n(n(n(n(5n - 108) + 1004) - 5238) + 16628) - 31758) + 29602) + 13158) - 85481) + 145890) - 166526) + 128184) + 27648)28080) - 5184) - \frac{72}{n}(n + 3)S_1(n) \right]$$

# QCD Sum Examples

## Tough Triple Sum

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

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

At the moment the algorithm is being tested on just over 7,000 sums that originate from a 3-loop QCD diagram that is not yet available in the literature.

Currently half the sums have been solved and the remaining half are being computed.

- ① A refined approach to multi-summation has the capability to interpolate between existing multi-summation techniques.
- ② With regards to  $\epsilon$  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) existing technologies in non-trivial examples.
- ④ The algorithm is now being used to solve new diagrams that as yet have not appeared in the literature.

# 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  $f(n+1, k)/f(n, k) = r_1(n, k)$  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.

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

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.