Some algebraic methods in Field Theory

J.A.M. Vermaseren

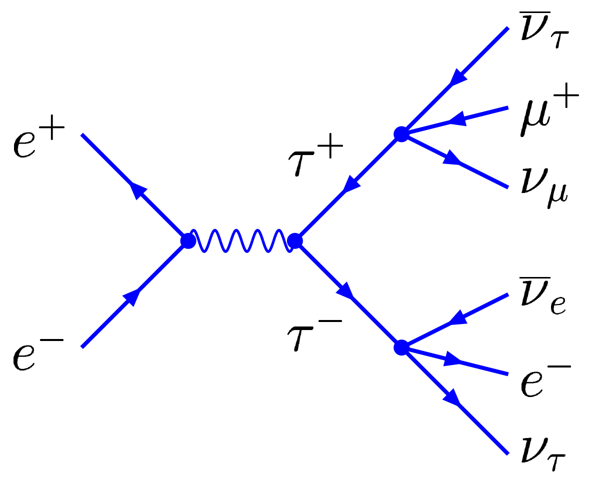
Nikhef

- Some simple examples.
- Harmonic sums and multiple zeta values.
- Solving linear equations.
- Difference equations.
- Planning ahead.

Some simple examples

Once it became clear that perturbative field theory was there to stay, the need for automated formula processing became obvious. This started with simple things like traces of Feynman diagrams.

Assume the reaction with the following Feynman diagram:



To take the traces with full spin-spin correllations is something that most people would consider too complicated if they would have to do this by hand. There are some tricks and if one is very good at this things, one can do it by hand, but then there is also the chance of mistakes. Hence:

```
    V   p1,p2,Q,q1,q2,p3,p4,p5,p6,p7,p8;
    I   m1,m2,m3;
    I   n1,n2,n3;
    S   emass,tmass,mass4,mass5,mass7,mass8;
    L   F =
        (g_(1,p2)-emass)*g_(1,m1)
        *(g_(1,p1)+emass)*g_(1,n1)
        *g_(2,p3)*g_(2,m2)*g7_(2)
        *(g_(2,q1)+tmass)*g_(2,m1)
        *(-g_(2,q2)+tmass)*g_(2,m3)*g7_(2)*g_(2,p6)
        *g_(2,n3)*g7_(2)*(-g_(2,q2)+tmass)*g_(2,n1)
        *(g_(2,q1)+tmass)*g_(2,n2)*g7_(2)
        *(g_(3,p4)+mass4)*g_(3,m2)*g7_(3)
        *(g_(3,p5)-mass5)*g_(3,n2)*g7_(3)
        *(g_(4,p7)+mass7)*g_(4,m3)*g7_(4)
        *(g_(4,p8)-mass8)*g_(4,n3)*g7_(4)
        ;
    trace4,4;
    trace4,3;
    trace4,1;
    trace4,2;
    print +f +s;
    .end

Time =       0.00 sec    Generated terms =        164
               F         Terms in output =         27
                         Bytes used      =       1384
```

```
F =
    - 524288*p1.p2*q1.q2*p3.p4*p5.p7*p6.p8*tmass^2
    + 524288*p1.p2*q1.p5*q2.p7*p3.p4*p6.p8*tmass^2
    + 524288*p1.p2*q1.p7*q2.p5*p3.p4*p6.p8*tmass^2
    + 1048576*p1.q1*p2.q2*q1.p5*q2.p7*p3.p4*p6.p8
    + 524288*p1.q1*p2.q2*p3.p4*p5.p7*p6.p8*tmass^2
    - 524288*p1.q1*p2.p7*q1.p5*q2.q2*p3.p4*p6.p8
    - 524288*p1.q1*p2.p7*q2.p5*p3.p4*p6.p8*tmass^2
    + 1048576*p1.q2*p2.q1*q1.p5*q2.p7*p3.p4*p6.p8
    + 524288*p1.q2*p2.q1*p3.p4*p5.p7*p6.p8*tmass^2
    - 524288*p1.q2*p2.p5*q1.q1*q2.p7*p3.p4*p6.p8
    - 524288*p1.q2*p2.p5*q1.p7*p3.p4*p6.p8*tmass^2
    - 524288*p1.p5*p2.q2*q1.q1*q2.p7*p3.p4*p6.p8
    - 524288*p1.p5*p2.q2*q1.p7*p3.p4*p6.p8*tmass^2
    + 262144*p1.p5*p2.p7*q1.q1*q2.q2*p3.p4*p6.p8
    + 524288*p1.p5*p2.p7*q1.q2*p3.p4*p6.p8*tmass^2
    + 262144*p1.p5*p2.p7*p3.p4*p6.p8*tmass^4
    - 524288*p1.p7*p2.q1*q1.p5*q2.q2*p3.p4*p6.p8
    - 524288*p1.p7*p2.q1*q2.p5*p3.p4*p6.p8*tmass^2
    + 262144*p1.p7*p2.p5*q1.q1*q2.q2*p3.p4*p6.p8
    + 524288*p1.p7*p2.p5*q1.q2*p3.p4*p6.p8*tmass^2
    + 262144*p1.p7*p2.p5*p3.p4*p6.p8*tmass^4
    + 262144*q1.q1*q2.q2*p3.p4*p5.p7*p6.p8*emass^2
    - 524288*q1.q1*q2.p5*q2.p7*p3.p4*p6.p8*emass^2
    + 1048576*q1.q2*q1.p5*q2.p7*p3.p4*p6.p8*emass^2
    - 524288*q1.p5*q1.p7*q2.q2*p3.p4*p6.p8*emass^2
    + 1048576*q1.p5*q2.p7*p3.p4*p6.p8*emass^2*tmass^2
    + 262144*p3.p4*p5.p7*p6.p8*emass^2*tmass^4
   ;
```

The above is a rather simple example. It becomes more complicated when we introduce diagrams with loops and reactions involving many diagrams. In this case we may have to spend all our time developing symbolic programs to do our calculations by computer. An example is the Mincer package for the manipulations of massless three loop propagator graphs. It took a while to develop it and make it efficient, but now everybody can use it and it gives relatively quick results. An example:

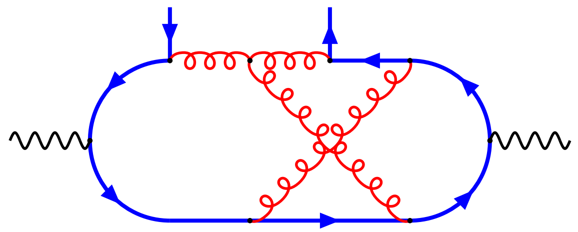


Diagram d9c in the qaqa dataset. We use the $\delta^{\mu\nu}$ projection in our test runs.

Of this diagram we run so-called Mellin moments. This means that we take derivatives w.r.t. the quark momentum P and put P equal to zero. This leaves very complicated propagator-like diagrams, especially when there are many derivatives. The number of derivatives determines an expansion in Mellin moments that allows us to reconstruct functions we are interested in to a given accuracy. In this case the problem involves deep inelastic scattering and there were more than 10000 diagrams in total.

Hence it is important to see to what derivative or 'power' we can go.

```
#define POW "6"
#define PROJ "0"
#define CURRENT "F2"
#include- mincer.h
.global
#include- diagram.h
        ;
.sort
#call treatqaqa(‘POW’)
.sort
#call integral(‘TOPO’)
.sort
#call trim(‘TOPO’)
print;
.end
   d9c =
      12520988335127/893025000 - 1507544/10125*ep^-3
       - 33324538/50625*ep^-2 - 9676515073/31893750*ep^-1
       - 46848/175*ep^-1*z3 - 440320/7*z5 - 70272/175*z4
       + 21504323128/496125*z3;

  24.78 sec out of 24.79 sec
```

N	Real time(sec)	Largest size(Gbytes)
10	59	0.195
12	157	0.504
14	369	1.18
16	888	2.65
18	2188	5.71
20	6290	11.47
22	18304	22.7
24	41695	43.0

Running diagram d9c of the qaqa dataset in the d projection with Mincer. We are using TFORM on 8 Xeon cores.

The ‘Largest size’ refers to a phenomenon called intermediate expression swell, which is the annoying property that intermediate steps may produce rather lengthy formulas, even though at the beginning and the end the formulas may be rather concise.

At the time this reaction was first completed (mid 90's) computers limited us to $N \leq 10$. At later times complete runs have been made to $N = 16$. These calculations have been quite instrumental in determining α_S accurately.

I hope this has convinced you of the need to use symbolic processing when we do computations in perturbative field theory.

Harmonic sums and multiple zeta values.

One of the recent developments in the evaluation of loop-integrals is that one can rewrite many integrals to (nested) sums. This can be done by a variety of ways.

1. One studies Mellin moments. Each moment is an integral. One can however construct recursions that map the integrals onto a set of master integrals, and these master integrals are determined by difference equations.
2. Variations of Mellin-Barnes transformations.

In general whatever integrals remain in the system are relatively trivial, but there is a number of nested sums left which may or may not be simple. These sums can be either to a finite (as in the Mellin moment case) or to an infinite upper bound (usually in the case of the M-B transformation). Especially in the case of the finite upper bound it is rather easy to verify the results.

Often sums are much easier to verify numerically than integrals.

What functions do we encounter?

Harmonic sums are defined by:

$$\begin{aligned}
S_m(N) &= \sum_{i=1}^{N} \frac{1}{i^m} \\
S_{-m}(N) &= \sum_{i=1}^{N} \frac{(-1)^i}{i^m} \\
S_{m,m_2,\cdots,m_p}(N) &= \sum_{i=1}^{N} \frac{1}{i^m} S_{m_2,\cdots,m_p}(i) \\
S_{-m,m_2,\cdots,m_p}(N) &= \sum_{i=1}^{N} \frac{(-1)^m}{i^m} S_{m_2,\cdots,m_p}(i)
\end{aligned}$$

This is a notation that is also suitable for computers. There is a difference here between various definitions as there are also people using $i-1$ for the argument of the S in the recursive formula. Those sums we call Z-sums.

The harmonic polylogarithms are defined by:

$$\begin{aligned} H(0;x) &= \ln x \\ H(1;x) &= \int_0^x \frac{dx'}{1-x'} = -\ln(1-x) \\ H(-1;x) &= \int_0^x \frac{dx'}{1+x'} = \ln(1+x) \end{aligned}$$

and the functions

$f(0;x) = \frac{1}{x}, \qquad f(1;x) = \frac{1}{1-x}, \qquad f(-1;x) = \frac{1}{1+x}$

If $\vec{a}_w$ is an array with w elements, all with value a, then:

$$\begin{aligned} H(\vec{0}_w;x) &= \frac{1}{w!}\ln^w x \\ H(a,\vec{m}_w;x) &= \int_0^x dx' \; f(a;x') \; H(\vec{m}_w;x') \end{aligned}$$

These functions are intimately related. The harmonic sums are the Mellin transforms of the harmonic polylogarithms.

$$\int_0^1 dx\ x^N H_{\vec{m}}(x) = \sum_{\vec{p}} c_{\vec{p},\vec{m}}\ S_{\vec{p}}(N)$$

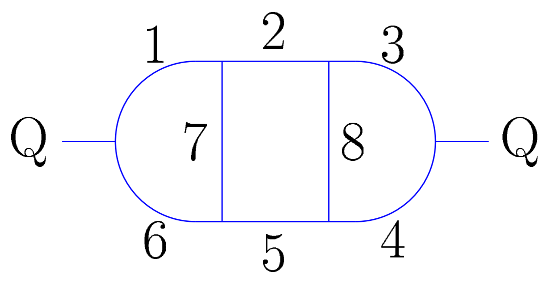
With suitable definitions of the concept of weight, the terms on both sides of the equation have the same weight.
The sums in infinity and the integrals in one are related and are called multiple zeta values.

```
 #define SIZE "6"
 #include- harmpol.h
 Off statistics;
 .global
 Local F = S(R(-1,3,-2),N);
 #call invmel(S,N,H,x)
 Print +f +s;
 .end

F =
    + H(R(-1,-3,0),x)*[1-x]^-1
    - 1/2*sign_(N)*H(R(1,0,0),x)*[1+x]^-1*z2
    + 1/2*H(R(-1,0,0),x)*[1-x]^-1*z2
    + 3/2*H(R(-1,0),x)*[1-x]^-1*z3
    + 21/20*H(R(-1),x)*[1-x]^-1*z2^2
    - 51/32*[1-x]^-1*z5
    + 3/4*[1-x]^-1*z2*z3
    - 7/2*s6
    + 51/32*z5*ln2
    - 33/64*z3^2
    + 9/4*z2*z3*ln2
    + 121/840*z2^3
    - 51/32*sign_(N)*[1+x]^-1*z5
    + 3/4*sign_(N)*[1+x]^-1*z2*z3
   ;

0.28 sec out of 0.33 sec
```

Example:



$$
\begin{aligned}
LA_{22}^{(131)} &= \int d^Dp_1 d_2^D d^Dp_3 \frac{(2P{\cdot}p_2)^8(2p_7{\cdot}p_8)^3}{p_1^2\ (p_2^2)^9\ p_3^2\ p_4^2\ p_5^2\ p_6^2\ p_7^2\ p_8^2} \\
&= (Q^2)^{1-6\epsilon}(\frac{2P{\cdot}Q}{Q^2})^8 \times \\
&\quad (\frac{436990847}{326592000} + \frac{1}{56}\epsilon^{-2} + \frac{107}{3360}\epsilon^{-1} - \frac{39}{35}\zeta_3)
\end{aligned}
$$

This took 45 sec. with FORM on a P2800 computer.

The general N form of the above diagram:

$$LA_{22}^{(131)} = \int d^D p_1 d_2^D d^D p_3 \frac{(2P\cdot p_2)^N (2p_7\cdot p_8)^3}{p_1^2 \; (p_2^2)^{N+1} \; p_3^2 \; p_4^2 \; p_5^2 \; p_6^2 \; p_7^2 \; p_8^2} =$$

```
+theta(-2+N)*(+den(-1+N)*(23/8+ep^-2+1/3*ep^-1+4/3*z3)
              +den(-1+N)^2*(-7/2-ep^-1)
              +den(-1+N)^3*(3)
              +den(-1+N)^3*S(R(1),-1+N)*(2/3)
              +den(-1+N)^2*S(R(1),-1+N)*(-31/18)
              +den(-1+N)*S(R(1),-1+N)*(37/36+ep^-1)
              +den(-1+N)*S(R(1,1),-1+N)*(1)
              +den(-1+N)*S(R(2),-1+N)*(-29/18)
              +den(-1+N)*S(R(3),-1+N)*(-2/3) )
+theta(-1+N)*(+den(N)*(-17/18-ep^-2-1/3*ep^-1+16/3*z3)
              +den(N)^2*(-73/18+ep^-1)
              +den(N)^3*(11/3)
              +den(N)^3*S(R(1),N)*(8/3)
              +den(N)^2*S(R(1),N)*(-29/9)
              +den(N)*S(R(1),N)*(23/9-ep^-1)
              +den(N)*S(R(1,1),N)*(-1)
              +den(N)*S(R(2),N)*(5/9)
              +den(N)*S(R(3),N)*(-8/3) )
+theta(N)*(   +den(1+N)*(4*z3)
              +den(1+N)^2*(8*z3)
              +den(1+N)^4*S(R(1),1+N)*(4)
              +den(1+N)^3*S(R(1),1+N)*(2/3)
              +den(1+N)^2*S(R(2),1+N)*(-2/3)
              +den(1+N)^2*S(R(3),1+N)*(-4)
              +den(1+N)*S(R(1),1+N)*(-8*z3)
              +den(1+N)*S(R(1,2),1+N)*(2/3)
              +den(1+N)*S(R(1,3),1+N)*(4)
              +den(1+N)*S(R(2,1),1+N)*(-2/3)
              +den(1+N)*S(R(3,1),1+N)*(-4) )
+delta(-1+N)*(+(1001/648+4/3*ep^-3-2/9*ep^-2-13/108*ep^-1-1/3*z3) )
+delta(N)*(   +(-1087/81-1/3*ep^-3-1/9*ep^-2-97/54*ep^-1+20/3*z3) );
```

The harmonic sums often enter problems by means of expansions of the Γ-function:

$$\begin{aligned}
\Gamma(n+1+\epsilon) &= n!\,\Gamma(1+\epsilon)(1+Z_1(n)\epsilon+Z_{1,1}(n)\epsilon^2+Z_{1,1,1}(n)\epsilon^3 \\
&\quad +Z_{1,1,1,1}(n)\epsilon^4+\cdots) \\
\Gamma(-n+\epsilon) &= \frac{(-1)^n}{\epsilon n!}\Gamma(1+\epsilon)(1+S_1(n)\epsilon+S_{1,1}(n)\epsilon^2+S_{1,1,1}(n)\epsilon^3 \\
&\quad +S_{1,1,1,1}(n)\epsilon^4+\cdots)
\end{aligned}$$

Solving linear equations

When we look at the inverse Mellin transform before we have a relatively short answer (14 terms). But this takes into account that there are many relations between the harmonic sums in infinity (or the hpl's in one). If we don't use these relations we have the result

```
+ H(R(-1,-3,0),x)*[1-x]^-1
- sign_(N)*H(R(1,0,0),x)*Z(-2)*[1+x]^-1
+ H(R(-1,0,0),x)*Z(-2)*[1-x]^-1
+ 2*H(R(-1,0),x)*Z(-3)*[1-x]^-1
+ 3*H(R(-1),x)*Z(-4)*[1-x]^-1
- sign_(N)*Z(-2,-3)*[1+x]^-1
+ 6*Z(-4,-1,1)       + 3*Z(-4,1,-1)
+ 5*Z(-3,-2,1)       + 4*Z(-3,-1,2)
+ Z(-3,1,-2)         + 3*Z(-3,2,-1)
- Z(-2,-3)*[1-x]^-1  + 2*Z(-2,-3,-1)
+ 5*Z(-2,-3,1)       + Z(-2,-2,-2)
+ 3*Z(-2,-2,2)       + 2*Z(-2,-1,-3)
+ 2*Z(-2,-1,3)       + Z(-2,2,-2)
+ 3*Z(-2,3,-1)       + 3*Z(-1,-4,1)
+ 2*Z(-1,-3,2)       + Z(-1,-2,-3)
+ Z(-1,-2,3)         + Z(-1,3,-2)
+ 3*Z(-1,4,-1)
```

Now we have 27 terms!

How to reduce one into the other?

To some extend this is a problem like calculating beta functions. One has to do this once for a whole theory and a given order in perturbation theory.

Here the order of the perturbation theory is related to the weight of the MZV's. Each extra order gives usually two extra in the weight of the sums.

Hence the order to do perturbative calculations in QCD is:

1. Get MZV's, sums and hpl's to the necessary weight.
2. Get the beta function to the needed order.
3. Get the needed anomalous dimensions (if any).
4. Calculate physical processes.

The harmonic sums obey a ‘stuffle’ algebra which is based on properties of sums:

$$\begin{aligned} S_{a,b}(N)S_{c,d}(N) &= S_{a,b,c,d}(N) + S_{a,c,b,d}(N) + S_{a,c,d,b}(N) \\ &\quad + S_{c,a,b,d}(N) + S_{c,a,d,b}(N) + S_{c,d,a,b}(N) \\ &\quad - S_{a+c,b,d}(N) - S_{a,c+b,d}(N) - S_{a,c,b+d}(N) \\ &\quad - S_{c,a,b+d}(N) - S_{c,a+d,b}(N) + S_{a+c,b+d}(N) \end{aligned}$$

The harmonic polylogarithms obey a ‘shuffle’ algebra as in

$$\begin{aligned} H_{a,b}H_{c,d} &= H_{a,b,c,d} + H_{a,c,b,d} + H_{a,c,d,b} \\ &\quad + H_{c,a,b,d} + H_{c,a,d,b} + H_{c,d,a,b} \end{aligned}$$

The multiple zeta functions (MZV's) are either Z-sums in infinity or harmonic polylogarithms in 1 (for positive indices they are identical). We can define a unified notation as in:

$$\begin{aligned} H_{0,0,1,0,-1} &= H_{3,-2} \\ Z_{7,2,-1} &= Z_{0,0,0,0,0,0,1,0,1,-1} \end{aligned}$$

The notation with the 0,1,-1 we call integral notation and the other notation we call sum notation. The number of indices in the integral notation is the weight, and the number of indices in the sum notation is the depth.

The algebraic relations show that these objects are not independent over the rational numbers. Given an appropriate basis, the MZV's of a given weight can be expressed in the basis elements for this weight and products of basis elements of lower weight, such that the sum of their weights is equal to the weight of the object we look at. Questions: what is a good basis for each weight and how do we express each object in terms of this basis? (This is over the rationals) There are many conjectures including one by Zagier about the size of the basis, but there is only one method known to check and to express elements in the basis:

Write down all elements and all relations due to the algebras and solve this system. (For the alternating sums there are more relations based on the property that for finite sums $Z(\infty) = Z(2\infty)$).

There are two systems to study: we call them the alternating sums (includes negative indices) and the non-alternating sums (only positive indices). Of the first there are $2\ 3^{w-1}$ and of the second there are 2^{w-1}. How far can we take FORM (and specifically TFORM) with this and how does that compare with dedicated programs written in C?

First the alternating sums. We want to construct tables that contain all sums of a given weight, including the divergent ones. This is the status as of 2007 (but rerun as a benchmark in 2008):

W	variables	eqns	remaining	size	output	time
4	36	57	1			
5	108	192	2			
6	324	665	2			
7	972	2205	4			
8	2916	7313	5			
9	8748	23909	8	11M	11M	56
10	26244	77853	11	58M	56M	406
11	78732	251565	18	360M	284M	3623
12	236196	809177	25	3.1G	1.48G	50926

Times in real time on a computer with 8 Xeon cores at 3 GHz and 4 Gbytes of memory per core, running TFORM.

With the new techniques used for the non-alternating sums, probably this can be taken further, but the storage of the answer becomes a problem. Weight 12 needs already 1.48 Gbyte.

The big fun started this year, trying to run the non-alternating sums and eventually trying to see how far one can go in determining the solution space for them. There exists a conjecture of Zagier and some people claim that interesting things are to happen at either W=23 or W=24. At the start of the project the status was a calculation by Kaneko, Noro and Tsurumaki who treated this as a (sparse) matrix problem.

W	size	time(sec)
16	72M	150
17	288M	880
18	1.2G	5000
19	4.6G	33000
20	18G	245000

They cannot go further for now as they need the whole matrix to be in memory. Maybe sooner or later they can do W=21. All calculus is modulus the 15-bit prime 31991. They confirm the Zagier conjecture for W=20.

This is what we get with TFORM for a 31-bit prime number (2147479273).

W	size	output	time(sec)
16	1.7M	1.2M	57
17	5.6M	3.2M	134
18	14.4M	7.2M	465
19	39M	19M	1206
20	104M	45M	4819
21	239M	114M	12379
22	767M	280M	65644
23	2.17G	734M	151337
24	8.04G	1.77G	1268247

Runs on a 8-core Xeon computer at 3 GHz and with 32 Gbytes of memory. Size is the maximum size of the master expression during the run, output is the size of the master expression in the end and time is the elapsed time in seconds.

These runs involved solving equations with a total of 2^{W-3} unknowns. In the worst case the program processed more than $4\ 10^{12}$ terms!

Under the assumption that no basis element has a depth (number of nested sums) of more than 1/3 of the weight, we can go even further:

W	size	output	time(sec)
23	1.55G	89M	9579
24	673M	380M	72991
25	6.37G	244M	50197
26	38.3G	1160M	651539

It is actually possible to run the complete expressions over the rational numbers. In that case we obtain:

W	size	output	num	real	Rat.
16	10.9M	10.6M	21	59	1.05
17	30M	29M	19	149	1.11
18	86M	77M	25	700	1.51
19	218M	205M	27	1855	1.54
20	756M	552M	31	11086	2.30
21	1.63G	1.55G	39	27771	2.24
22	8.05G	4.00G	36	326489	4.97

Runs to obtain full results for all MZV's of a given weight. Rat. refers to the ratio in real time between this run and the run modulus a 31-bits prime to just determine the size of the basis. Num is the number of digits in the largest number (either a numerator or a denominator).

This confirms the Zagier conjecture to W=22. To W=24 it is confirmed modulus 2147479273, and to W=26 it is confirmed modulus 2147479273 and under the assumption that no basis element needs to have a depth of more than $W/3$.

It also shows that FORM is quite a good tool for these types of calculations.

We (J. Blümlein, D. Broadhurst and JV) have more results, both for the alternating and the non-alternating sums. A paper is being prepared.

All results and the programs will be made available in a 'datamine' which will be pointed at from the FORM website once it exists.

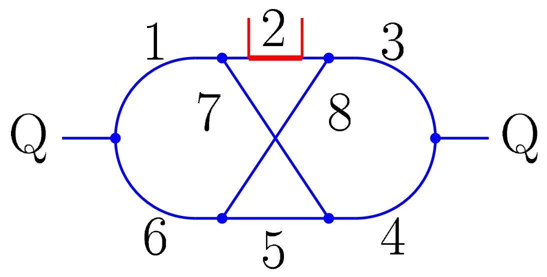
Difference equations

One technique that is gaining popularity at the moment is the solving of difference equations. These are equations of the type

$$a_0(N)\ F(N) + a_1(N)\ F(N-1) + \cdots + a_m(N)\ F(N-m)\ =\ G(N)$$

When m is equal to one this is commonly called a recursion. These equations are related to differential equations by an inverse Mellin transform.
These equations occur when we reduce loopintegrals to a limited set of master integrals together with integrals of a simpler type. Such reduction equations have usually a recursive nature. They may not simplify the masterintegrals, but they may give difference equations for them.
This means that we have to solve these equations.

An example of such an integral is



$$NO_{22}(N) \times (\frac{2P{\cdot}Q}{Q{\cdot}Q})^N \; = \int d^Dp_1 d_2^D d^D p_3 \frac{(2P{\cdot}p_2)^N}{p_1^2 \; (p_2^2)^{N+1} \; p_3^2 \; p_4^2 \; p_5^2 \; p_6^2 \; p_7^2 \; p_8^2}$$

We want the result for any value of N. We have a program that can give the result for a limited number of fixed values of N, like $N = 0 \cdots 9$:

$$\begin{aligned}
F(0) &= +20\zeta_5 \\
F(1) &= +8\zeta_3 \\
F(2) &= -\frac{5}{3} + \frac{4}{3}\zeta_3 + \frac{20}{3}\zeta_5 \\
F(3) &= -\frac{8}{27} + \frac{40}{9}\zeta_3 \\
F(4) &= -\frac{5321}{4320} + \zeta_3 + 4\zeta_5 \\
F(5) &= -\frac{64}{225} + \frac{2072}{675}\zeta_3 \\
F(6) &= -\frac{6471461}{6804000} + \frac{7}{9}\zeta_3 + \frac{20}{7}\zeta_5 \\
F(7) &= -\frac{41224}{165375} + \frac{25832}{11025}\zeta_3 \\
F(8) &= -\frac{14795817599}{19203609600} + \frac{205}{324}\zeta_3 + \frac{20}{9}\zeta_5 \\
F(9) &= -\frac{61197376}{281302875} + \frac{939752}{496125}\zeta_3
\end{aligned}$$

For this diagram the equation is a third order equation with:

$$\begin{aligned}
G(N) &= 12S_{-2}(N)((-4N+2)(-1)^N+1)+12S_2(N) \\
&\quad +24(1-(-1)^N)/N-12(1-(-1)^N)/N^2 \\
a_0(N) &= N(N-2\epsilon)(N+2\epsilon)(N+1-2\epsilon)(3N+3+2\epsilon)/2 \\
a_1(N) &= (N-2\epsilon)(15N\epsilon+4N\epsilon^3-3N-18N^2\epsilon \\
&\quad -10N^2\epsilon^2+9N^2+5N^3\epsilon-9N^3+3N^4 \\
&\quad -2\epsilon+6\epsilon^2-8\epsilon^3+8\epsilon^4)/2 \\
a_2(N) &= (N-1)(12N\epsilon-28N\epsilon^2-160\epsilon^3 \\
&\quad -60N^2\epsilon+44N^2\epsilon^2+52N^3\epsilon+6N^3 \\
&\quad +6N^4+8\epsilon^2+56\epsilon^3-112\epsilon^4)/4 \\
a_3(N) &= (N-1)(N-2)(3N+2\epsilon)(N-1+3\epsilon)(N-1+6\epsilon)/2
\end{aligned}$$

The way to solve such equations is by guessing the category of functions that make up the answer. We then write

$$F(N) = \sum_p c_p F_p(N)$$

in which we sum p over all possibilities inside this category and c_p is the corresponding numerical coefficient.
Next we substitute this function in the equation and bring everything to an unique standard form. Then each different function will have a coefficient that is a linear combination of the coefficients c_p and as a result we have a large set of linear equations to solve. In combination with the equations generated by the boundary conditions these equations will determine all c_p, provided we guessed right.

The set of function that we need may be very large as it involves

- Harmonic sums up to a given weight.
- The arguments of the harmonic sums may have an offset as in $N-i$. We have to select a range of values for i.
- There can be powers of ϵ. Typically for each extra power of ϵ we need an extra weight for the sums.
- There can be powers of ζ_3, ζ_4, ζ_5.

The result is that for typical three loop propagator graphs that we needed to consider there may be thousands of possibilities and hence thousands of equations to solve. In the case of the diagram we consider here we had more than 40000 variables.

The FORM procedure trysolve does exactly this. We provide it with the equation, the boundary values and the range of guesses and then the procedure constructs the Ansatz function, the equations and solves them. Out of the solution it constructs the answer and then it tests whether this answer is consistent with possible excess boundary values we provide.

In the case of our diagram the answer is rather simple (this is exceptional):

$$
\begin{aligned}
F(N) \;=\; \theta(N)\frac{1+(-1)^N}{2}\frac{1}{1+N}\big(&+20\zeta_5 + 12S_{-3,-2}(N+1) \\
&+4S_{-3,2}(N+1) + 8S_{-2}(N+1)\zeta_3 + 4S_{-2,-3}(N+1) \\
&-4S_{-2,3}(N+1) + 8S_{2}(N+1)\zeta_3 + 4S_{2,-3}(N+1) \\
&-4S_{2,3}(N+1) + 12S_{3,-2}(N+1) + 4S_{3,2}(N+1)\big) \quad (1)
\end{aligned}
$$

Nowadays there is also more mathematical interest in this. There is for instance the Sigma package by C. Schneider.

Planning ahead.

At the moment there is much interest in numerical integration by sector decomposition. People make first algebraic programs to divide integrals in multi-dimensional space into regions for which the numerical integral can be computed relatively easily.
An example is a recent package by Smirnov^2 and Tentyukov in which some categories of loop integrals are dealt with by a Mellin Barnes transformation and subsequent sector decomposition. The final integrals are then done by Monte Carlo techniques.
Unfortunately the package is for Mathematica which is way too expensive for people who are not at a university with a site license. Also in the end it turns out that Mathematica cannot handle the really big expressions anyway.
Fortunately there are more people working on this subject programming for free systems.

Solving recursions, solving sets of equations, etc will become more and more important. This will involve also Gröbner basis techniques in the future. Be on the lookout for those.
This will require better polynomial handling techniques in systems like FORM or GiNaC.