Some obstacles in the calculation of structure functions

J.A.M. Vermaseren

Nikhef

- What does the output look like?
- Mellin transforms.
- Multiple Zeta Values.
- Deriving Relations.
- Running with TFORM.
- Non-Alternating Sums.
- Alternating Sums.
- Conclusions.

What does the output look like?

Before one starts a calculation of structure functions one should worry whether one is in complete control of the mathematics involved. This leads to a number of questions:

- Can I do all the integrals?
- What functions will be in the output?
- Are these functions practical?
- Can the calculation be done in a simpler way?
- How do I make sure there are no errors?
- Etc.

Because this is only a short talk, we skip the historical remarks and go directly to the mathematics we need. We decide that the calculation is much easier in Mellin space and that introduces harmonic sums.

The 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 whole problem can be set up in such a way that the integrals are trivial and there remain nested sums involving harmonic sums. Some of the master integrals are determined by difference equations.

Converting to Bjorken x-space is done with a so-called inverse Mellin transform. Studying the Mellin transform led to the finding of a new class of functions which were called the harmonic polylogarithms.

It was not very difficult to write a procedure for the inverse Mellin transform and to express the results in terms of the Hpl's.

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

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

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

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

The systematic treatment of the harmonic sums has led to new ways of computing Feynman diagrams. The best known method is now the use of the Mellin Barnes transform, which introduces new integrals, but allows one to do the Lorenz integrals in a rather direct way. Then the new integrals can be done by studying their contour and closing it. This then leads to sums over the pole contributions of the Gamma functions in the formulas. In the end the whole problem is reduced to a summation problem, rather than an integration problem. Very often the sums are much easier than the original integrals.

If this would be all, there would be no need for THIS talk.

Mellin transforms

Let us have a look at one of the inverse Mellin transforms. We have a FORM library for the sums and the Hpl's and it contains routines for Mellin transforms and inverse Mellin transforms. Hence we take the inverse transform of a single sum:

```
 #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 =
    - 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
    - 1/2*sign_(N)*H(R(1,0,0),x)*[1+x]^-1*z2
    + 21/20*H(R(-1),x)*[1-x]^-1*z2^2
    + H(R(-1,-3,0),x)*[1-x]^-1
    + 3/2*H(R(-1,0),x)*[1-x]^-1*z3
    + 1/2*H(R(-1,0,0),x)*[1-x]^-1*z2
   ;
```

We see a number of constants like ζ_2, ζ_3, ζ_5 and a strange number named s_6 which is defined as $S_{-5,-1}(\infty)$. Actually originally there were many more constants. The 'raw' run gives:

```
#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 =
    - sign_(N)*H(R(1,0,0),x)*Htab2(0,-1)*[1+x]^-1
    - sign_(N)*Htab5(0,-1,0,0,-1)*[1+x]^-1
    - sign_(N)*Htab5(0,-1,0,0,1)*[1+x]^-1
    + sign_(N)*Htab5(0,-1,1,0,0)*[1+x]^-1
    - 2*sign_(N)*Htab5(0,0,-1,0,1)*[1+x]^-1
    - 3*sign_(N)*Htab5(0,0,0,-1,1)*[1+x]^-1
    - 3*sign_(N)*Htab5(0,0,0,1,-1)*[1+x]^-1
```

```
- sign_(N)*Htab5(0,0,1,0,-1)*[1+x]^-1
+ sign_(N)*Htab5(0,1,-1,0,0)*[1+x]^-1
+ sign_(N)*Htab5(0,1,0,-1,0)*[1+x]^-1
+ sign_(N)*Htab5(0,1,0,0,-1)*[1+x]^-1
+ sign_(N)*Htab5(1,0,-1,0,0)*[1+x]^-1
+ 2*sign_(N)*Htab5(1,0,0,-1,0)*[1+x]^-1
+ 3*sign_(N)*Htab5(1,0,0,0,-1)*[1+x]^-1
- H(R(-1),x)*Htab4(0,0,-1,0)*[1-x]^-1
+ H(R(-1,-3,0),x)*[1-x]^-1
- H(R(-1,0),x)*Htab3(0,-1,0)*[1-x]^-1
- H(R(-1,0,0),x)*Htab2(-1,0)*[1-x]^-1
+ 6*Htab5(-1,-1,0,0,0)*[1-x]^-1
+ 5*Htab5(-1,0,-1,0,0)*[1-x]^-1
+ 3*Htab5(-1,0,0,-1,0)*[1-x]^-1
+ 4*Htab5(0,-1,-1,0,0)*[1-x]^-1
+ 3*Htab5(0,-1,0,-1,0)*[1-x]^-1
+ 2*Htab5(0,0,-1,-1,0)*[1-x]^-1
+ Htab5(0,0,-1,0,-1)*[1-x]^-1
+ Htab6(-1,0,-1,0,0,-1)
+ Htab6(-1,0,-1,0,0,1)
+ 2*Htab6(-1,0,0,-1,0,1)
+ 3*Htab6(-1,0,0,0,-1,1)
+ 3*Htab6(-1,0,0,0,1,-1)
```

```
 + Htab6(-1,0,0,1,0,-1)
 + 2*Htab6(0,-1,-1,0,0,-1)
 + 2*Htab6(0,-1,-1,0,0,1)
 + Htab6(0,-1,0,-1,0,-1)
 + 3*Htab6(0,-1,0,-1,0,1)
 + 2*Htab6(0,-1,0,0,-1,-1)
 + 5*Htab6(0,-1,0,0,-1,1)
 + 3*Htab6(0,-1,0,0,1,-1)
 + Htab6(0,-1,0,1,0,-1)
 + 4*Htab6(0,0,-1,-1,0,1)
 + 5*Htab6(0,0,-1,0,-1,1)
 + 3*Htab6(0,0,-1,0,1,-1)
 + Htab6(0,0,-1,1,0,-1)
 + 6*Htab6(0,0,0,-1,-1,1)
 + 3*Htab6(0,0,0,-1,1,-1)
;
```

The Htab objects are Hpl's in one in which for instance Htab6(0,0,0,-1, 1,-1) stands for $H_{-4,1,-1}(1)$. The numbers are related to the sums in infinity.

The interesting thing is that there are relations between these constants. If we use these relations, the final answer is much shorter as we could see before.

Clearly the shorter answer is much more informative. Hence we need to know these relations and express all these constants in terms of a minimal set. This is also already an old problem that goes back to Euler. Euler however didn't have computers and didn't do QCD loop corrections.

How many of these constants are there?

We can define a unified notation as in:

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

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.

In this notation counting is easy: there are $2 \times 3^{w-1}$ sums in infinity and $4 \times 3^{w-2}$ are finite. Hence at weight 6 we have 324 finite sums. In the end we can express them all in combinations of 8 different constants in such a way that there are in total 13 combinations. And often many of those combinations vanish. The number is given by the w-th Fibonacci F_w number when $F_0 = F_1 = 1$ and $F_w = F_{w-1} + F_{w-2}$.

The problem in this talk is: "How to go from $4 \times 3^{w-2}$ to F_w?"

Multiple Zeta Values

The sums in infinity (or the Hpl's in one) are called Multiple Zeta Values. Some of them also go by the name Euler Zagier sums. They have become interesting to the mathematicians in the 1990's, but then mainly the sums with positive indices only. We also need the sums that can have negative indices. Gastmans and Troost (1981) managed to give the relations for all weight 4 and a number of weight 5 sums. We need basically weight $= 2\times$ (number of loops).

The number of MZV's that exists is $2 \times 3^{w-1}$ for the alternating sums and 2^{w-1} for the non-alternating ones. If we remove the divergent sums these numbers become $4 \times 3^{w-2}$ and 2^{w-2} respectively. This means that for weight 6 there are 324 (16) constants that we have to determine.

Unfortunately this is where the mathematicians run into trouble. There is no known constructive way to take one of these constants and express it into a basis that consists of a minimal set.

The only two ways that are currently known are:

- Write down all algebraic relations for these objects and solve the system of equations.
- Guess a relation and fit the coefficients with a program like PSLQ after computing all objects in the relation numerically to a very large number of digits. Broadhurst has done much of this in the 1990's.

Deriving Relations

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

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

The harmonic polylogarithms obey a 'shuffle' algebra as in

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

When we take the limit $N \to \infty$ or $x \to 1$ the sums and the Hpl's can be expressed into each other and we obtain MZV's. Hence the MZV's obey both relations. It should be noted that when there are negative indices the sum of the indices becomes a bit more complicated:

$$a + b \;\to\; \sigma_a \sigma_b (|a| + |b|) = \sigma_a b + \sigma_b a$$

in which σ_a is the sign of a.

When all indices are positive, these are the only relations that we know about. Example:

$$S_2(\infty) S_2(\infty) \;=\; 2 S_{2,2}(\infty) - S_4(\infty)$$

which tells us that $S_{2,2}(\infty)$ can be expressed in terms of ζ_2^2 and ζ_4.

$$\begin{aligned} H_2(1) H_2(1) \;&=\; H_{0,1}(1) H_{0,1}(1) \\ &=\; 2 H_{0,1,0,1}(1) + 4 H_{0,0,1,1}(1) \\ &=\; 2 H_{2,2}(1) + 4 H_{3,1}(1) \end{aligned}$$

For the alternating sums there are more relations.

$$S_{n_1,\cdots,n_p}(N) \;=\; 2^{n_1+\cdots+n_p-p} \sum_{\pm} S_{\pm n_1,\cdots,\pm n_p}(2N)$$

They are called the doubling relations. When $n \to \infty$ and the sums are finite, this gives useful relations.

For the alternating sums there is yet another category of relations which we call the generalized doubling formulas (GDF's). They are based on similar principles but we have only a computer algorithm to generate them. No closed formula.

Running with TFORM

Trying to solve large systems of equations can be quite a challenge. And because we want to reach the limits of what is possible we need the most powerful program we can lay our hands on. Of course we have FORM, but there is the newer TFORM that can make use of multi-core machines. This gives added power.

We use the MZV program as a test under extreme conditions for TFORM. This has enabled us to both

- Test and improve TFORM.
- Improve the program for solving sets of equations.

Unfortunately this is a short talk and we cannot go into the details of the program. They will become available in a future publication.
We will run three types of programs.

1. A full expression of all MZV's in a minimal basis.
2. An expression of all MZV's in a minimal basis modulus a prime number.
3. A determination of a basis modulus a prime number and expression of all MZV's of a given weight in terms of this basis, dropping all terms that are products of lower weight objects.

We also separate the programs in two categories. In the first we determine all MZV's of a given weight, and in the second we determine only a subset in which the depth is at most a given number.

Non-Alternating Sums

Mathematicians are mostly interested in the MZV's with only positive indices. Yet they have not yet even proved how many of them are independent. There are quite a few conjectures. One of them is by Hoffmann who says that taking all sums with indices 2 and/or 3 gives a basis when we don't reduce sums as much as possible to products of lower weight objects but express everything as nested sums of the same weight. With computer programs that treat this as a matrix problem Kaneko, Noro and Tsurumaki managed to prove this to weight 20.
Using algebraic techniques and TFORM we managed to extend this to weight 24.

Broadhurst and Kreimer had a better conjecture that tells that when one tries to express everything as much as possible in terms of products of lower weight objects, and the remaining objects in terms of sums with as low a depth (number of nested sums), how many objects are left for each weight and depth. But which elements to select as a basis also they cannot tell. We have managed to confirm their formula to weight 26, although there are some assumptions there. For lower depths we can go to much higher weights.

Some of the parameters of the runs are rather impressive:

W	size	output	num	CPU	real	Eff.	Rat.
16	10.9M	10.6M	21	254	59	4.29	1.05
17	30M	29M	19	690	149	4.62	1.11
18	86M	77M	25	3491	700	4.98	1.51
19	218M	205M	27	9460	1855	5.10	1.54
20	756M	552M	31	65640	11086	5.92	2.30
21	1.63G	1.55G	39	165561	27771	5.96	2.24
22	8.05G	4.00G	36	2276418	326489	6.97	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 largest number of decimal digits in either a numerator or a denominator in the output.

W	size	output	CPU	real	Eff.
16	1.7M	1.2M	300	57	5.25
17	5.6M	3.2M	713	134	5.32
18	14.4M	7.2M	2706	465	5.82
19	39M	19M	6901	1206	5.72
20	104M	45M	30097	4819	6.25
21	239M	114M	75302	12379	6.08
22	767M	280M	449202	65644	6.84
23	2.17G	734M	992431	151337	6.56
24	8.04G	1.77G	9251325	1268247	7.29

Runs on a 8-core Xeon computer at 3 GHz and with 32 Gbytes of memory. CPU is the total CPU time of all processors together in seconds, real is the elapsed time in seconds and Eff. is the pseudo efficiency, defined by the CPU time divided by the real time.

W	D	size	output	CPU	real	Eff.
23	7	1.55G	89M	61447	9579	6.41
24	8	673M	380M	536921	72991	7.36
25	7	6.37G	244M	369961	50197	7.37
26	8	38.3G	1160M	4786841	651539	7.35

Runs on a 8-core Xeon computer at 3 GHz and with 32 Gbytes of memory. D indicates the maximum depth (see text). We reran at 23 and 24 to have information for extrapolation purposes.

Alternating Sums

For physicists the alternating sums are more interesting. Here one cannot go to such a high weights. Actually the mathematicians never got beyond weight 8. And they didn't make those results available. The reason could be that they never used the doubling formula. This formula is needed for the first time at weight 8. At weight 11 there are even more formula's needed or the Broadhurst conjecture will break down. These are the generalized doubling formulas we mentioned before. They are very nice in that they solve also other problems that existed in the derivation of the alternating MZV's.

In total we could run all the MZV's to weight 12.

W	variables	eqns	remaining	size	output	time
4	36	57	1	4.3K	2.0K	0.06
5	108	192	2	21K	8.9K	0.12
6	324	665	2	98K	42K	0.37
7	972	2205	4	472K	219K	1.71
8	2916	7313	5	2.25M	1.15M	7.78
9	8748	23909	8	11M	6.3M	50
10	26244	77853	11	58M	36M	353
11	78732	251565	18	360M	213M	3266
12	236196	809177	25	3.1G	1.29G	47311

Runs on a 8-core Xeon computer at 3 GHz and with 32 Gbytes of memory. The column 'eqns' gives the number of equations that was considered.

We see that the size of the outputs becomes a bigger problem than the running time.

An example of a special relation is the following:

$$
\begin{aligned}
H_{8,2,1,1} = & -\frac{1593344}{47475}H_{-11,-1} + \frac{10624}{28485}H_{-9,-3} + \frac{56896}{712125}H_{-7,-5} \\
& + \frac{64}{243}H_{-3}^4 + \frac{194772992}{2421225}H_{-9}H_{-3} + \frac{56203264}{712125}H_{-7}H_{-5} \\
& + \frac{21504}{1583}\zeta_2 H_{-9,-1} - \frac{768}{1583}\zeta_2 H_{-7,-3} - \frac{8660992}{299187}\zeta_2 H_{-7}H_{-3} \\
& - \frac{529216}{39575}\zeta_2 H_{-5}^2 + \frac{512}{171}\zeta_2^2 H_{-7,-1} - \frac{512}{2565}\zeta_2^2 H_{-5,-3} \\
& - \frac{98624}{12825}\zeta_2^2 H_{-5}H_{-3} - \frac{352}{315}\zeta_2^3 H_{-3}^2 \\
& - \frac{59755910459266246}{18760001932546875}\zeta_2^6
\end{aligned}
$$

In the case of nonalternating sums one can express this object only as a four-fold (depth 4) sum. When the alternating sums are included double sums suffice.

For limited depth and using modulus calculus we could go further and when we are only interested in those terms that don't involve products of lower weight objects we could even go to weight 18, depth ≤ 6. This is more important than it seems as apparently there must be some interesting relations combining the results of this run with the non-alternating results for weight 18. We have however not yet completed the study of these results.

One thing that comes out though that there is one nonalternating sum with depth 6 that can be expressed in alternating sums of depth 4 and one nonalternating sum of depth 4 that is a basis element that can be expressed in terms of other basis elements of depth 4 and alternating sums of depth 2. These are very important findings for people who try to study these sums.

weight	constants	remaining	running time [sec]	output [Mbyte]
13	56940	22	2611	
14	90564	37	12716	51
15	138636	35	55204	87
16	205412	66	206951	214
17	295916	55	789540	288
18	416004	109	2622157	711

Summary of the runs at $d = 6$ in modular arithmetic, dropping all terms that are products of lower weight objects. Times refer to an 8 Xeon core machine at 3 GHz and 32 GBytes of memory.

The weight 18 run here was rather impressive. It took one month on an 8 core Xeon machine, working its way through a combined total of more than 7×10^{12} terms or 7 TeraTerms!

Conclusions

We have now tables for the MZV's that will cover all cases up to 6-loop coefficient functions.

The holy grail in the field of MZV's is an algorithm to express each MZV into an unique basis in a constructive way. That way we would have a (hopefully) small procedure rather than giant tables. Thus far this has not been found.

The only practical way to reduce combinations of MZV's to a minimal set is by tabulating expressions for all of them and substituting from the tables. The construction of these tables can currently only be done by brute force. We have applied the force of TFORM to it.....