Computation and theory of extended Mordell-Tornheim-Witten sums

David H. Bailey
http://www.davidhbailey.com
Computational Research Dept., Lawrence Berkeley National Lab (retired)
Computer Science Department, University of California, Davis

14 Dec 2013

The PSLQ integer relation algorithm

Let (x_n) be a given vector of real numbers. An integer relation algorithm either finds integers (a_n) such that

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \quad = \quad 0$$

(to within the "epsilon" of the arithmetic being used), or else finds bounds within which no relation can exist.

The "PSLQ" algorithm of mathematician-sculptor Helaman Ferguson is the most widely used integer relation algorithm.

Integer relation detection requires very high precision (at least $n \times d$ digits, where d is the size in digits of the largest a_k), both in the input data and in the operation of the algorithm.

1. H.R.P. Ferguson, D.H. Bailey and S. Arno, "Analysis of PSLQ, An Integer Relation Finding Algorithm," *Mathematics of Computation*, vol. 68, no. 225 (Jan 1999), pg. 351–369.
2. D.H. Bailey and D.J. Broadhurst, "Parallel Integer Relation Detection: Techniques and Applications," *Mathematics of Computation*, vol. 70, no. 236 (Oct 2000), pg. 1719–1736.

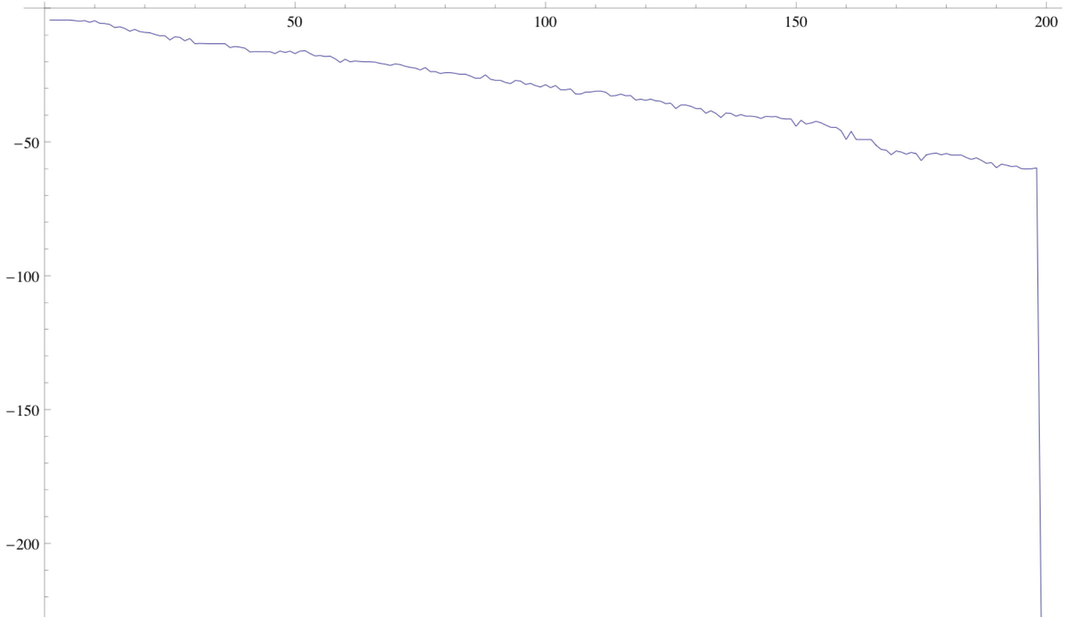
PSLQ, continued

- PSLQ constructs a sequence of integer-valued matrices B_n that reduce the vector $y = x \cdot B_n$, until either the relation is found (as one of the columns of matrix B_n), or else precision is exhausted.
- A relation is detected when the size of smallest entry of the y vector suddenly drops to roughly "epsilon" (i.e. 10^{-p}, where p is the number of digits of precision).
- The size of this drop can be viewed as a "confidence level" that the relation is not a numerical artifact: a drop of 20+ orders of magnitude almost always indicates a real relation.

Efficient variants of PSLQ:

- 2-level and 3-level PSLQ perform almost all iterations with only double precision, updating full-precision arrays as needed. They are hundreds of times faster than the original PSLQ.
- Multi-pair PSLQ dramatically reduces the number of iterations required. It was designed for parallel systems, but runs faster even on 1 CPU.

Decrease of $\log 10(\min |y_i|)$ in multipair PSLQ run



The first major PSLQ discovery: The BBP formula for π

In 1996, a PSLQ program discovered this new formula for π:

$$\pi = \sum_{n=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

This formula permits one to compute binary (or hexadecimal) digits of π beginning at an arbitrary starting position, using a very simple scheme that requires only standard 64-bit or 128-bit arithmetic.

In 2004, Borwein, Galway and Borwein proved that no base-n formulas of this type exist for π, except when $n = 2^m$.

BBP-type formulas (discovered with PSLQ) are now known for numerous other mathematical constants.

1. D.H. Bailey, P.B. Borwein and S. Plouffe, "On the rapid computation of various polylogarithmic constants," *Mathematics of Computation*, vol. 66, no. 218 (Apr 1997), pg. 903–913.
2. J.M. Borwein, W.F. Galway and D. Borwein, "Finding and excluding b-ary Machin-type BBP formulae," *Canadian Journal of Mathematics*, vol. 56 (2004), pg. 1339–1342.

Some other BBP-type formulas found using PSLQ

$$\pi^2 = \frac{1}{8}\sum_{k=0}^{\infty}\frac{1}{64^k}\left(\frac{144}{(6k+1)^2} - \frac{216}{(6k+2)^2} - \frac{72}{(6k+3)^2} - \frac{54}{(6k+4)^2} + \frac{9}{(6k+5)^2}\right)$$

$$\pi^2 = \frac{2}{27}\sum_{k=0}^{\infty}\frac{1}{729^k}\left(\frac{243}{(12k+1)^2} - \frac{405}{(12k+2)^2} - \frac{81}{(12k+4)^2} - \frac{27}{(27k+5)^2} - \frac{72}{(12k+6)^2} - \frac{9}{(12k+7)^2} - \frac{9}{(12k+8)^2} - \frac{5}{(12k+10)^2} + \frac{1}{(12k+11)^2}\right)$$

$$\zeta(3) = \frac{1}{1792}\sum_{k=0}^{\infty}\frac{1}{2^{12k}}\left(\frac{6144}{(24k+1)^3} - \frac{43008}{(24k+2)^3} + \frac{24576}{(24k+3)^3} + \frac{30720}{(24k+4)^3} - \frac{1536}{(24k+5)^3} + \frac{3072}{(24k+6)^3} + \frac{768}{(24k+7)^3} - \frac{3072}{(24k+9)^3} - \frac{2688}{(24k+10)^3} - \frac{192}{(24k+11)^3} - \frac{1536}{(24k+12)^3} - \frac{96}{(24k+13)^3} - \frac{672}{(24k+14)^3} - \frac{384}{(24k+15)^3} + \frac{24}{(24k+17)^3} + \frac{48}{(24k+18)^3} - \frac{12}{(24k+19)^3} + \frac{120}{(24k+20)^3} + \frac{48}{(24k+21)^3} - \frac{42}{(24k+22)^3} + \frac{3}{(24k+23)^3}\right)$$

D.H. Bailey, J.M. Borwein, A. Mattingly and G. Wightwick, "The computation of previously inaccessible digits of Pi^2 and Catalan's constant," *Notices of the American Mathematical Society*, vol. 60 (2013), no. 7, pg. 844-854.

Algebraic numbers in Poisson potential functions and lattice sums

Lattice sums arising from the Poisson equation have been studied widely in mathematical physics and also in image processing. We numerically discovered, and then proved, that for rational (x, y), the two-dimensional Poisson potential function satisfies

$$\phi_2(x, y) = \frac{1}{\pi^2} \sum_{m,n \text{ odd}} \frac{\cos(m\pi x)\cos(n\pi y)}{m^2 + n^2} = \frac{1}{\pi} \log \alpha$$

where α is an *algebraic number*, i.e., the root of an integer polynomial

$$0 = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_n\alpha^n$$

The minimal polynomials for these α were found by PSLQ calculations, with the $(n+1)$-long vector $(1, \alpha, \alpha^2, \cdots, \alpha^n)$ as input, where $\alpha = \exp(8\pi\phi_2(x, y))$. PSLQ returned the vector of integer coefficients $(a_0, a_1, a_2, \cdot, a_n)$ as output.

1. D.H. Bailey, J.M. Borwein, R.E. Crandall and J. Zucker, "Lattice sums arising from the Poisson equation," *Journal of Physics A: Mathematical and Theoretical*, vol. 46 (2013), pg. 115201, `http://www.davidhbailey.com/dhbpapers/PoissonLattice.pdf`.
2. D.H. Bailey and J.M. Borwein, "Compressed lattice sums arising from the Poisson equation: Dedicated to Professor Hari Sirvastava," *Boundary Value Problems*, vol. 75 (2013), DOI: 10.1186/1687-2770-2013-75, `http://www.boundaryvalueproblems.com/content/2013/1/75`.

Samples of minimal polynomials found by PSLQ

k	Minimal polynomial for $\exp(8\pi\phi_2(1/k,1/k))$
5	$1 + 52\alpha - 26\alpha^2 - 12\alpha^3 + \alpha^4$
6	$1 - 28\alpha + 6\alpha^2 - 28\alpha^3 + \alpha^4$
7	$-1 - 196\alpha + 1302\alpha^2 - 14756\alpha^3 + 15673\alpha^4 + 42168\alpha^5 - 111916\alpha^6 + 82264\alpha^7$ $-35231\alpha^8 + 19852\alpha^9 - 2954\alpha^{10} - 308\alpha^{11} + 7\alpha^{12}$
8	$1 - 88\alpha + 92\alpha^2 - 872\alpha^3 + 1990\alpha^4 - 872\alpha^5 + 92\alpha^6 - 88\alpha^7 + \alpha^8$
9	$-1 - 534\alpha + 10923\alpha^2 - 342864\alpha^3 + 2304684\alpha^4 - 7820712\alpha^5 + 13729068\alpha^6$ $-22321584\alpha^7 + 39775986\alpha^8 - 44431044\alpha^9 + 19899882\alpha^{10} + 3546576\alpha^{11}$ $-8458020\alpha^{12} + 4009176\alpha^{13} - 273348\alpha^{14} + 121392\alpha^{15}$ $-11385\alpha^{16} - 342\alpha^{17} + 3\alpha^{18}$
10	$1 - 216\alpha + 860\alpha^2 - 744\alpha^3 + 454\alpha^4 - 744\alpha^5 + 860\alpha^6 - 216\alpha^7 + \alpha^8$

The minimal polynomial for $\exp(8\pi\phi_2(1/32,1/32))$ has degree 128, with individual coefficients ranging from 1 to over 10^{56}. This PSLQ computation required 10,000-digit precision. See next page.

Other polynomials required up to 50,000-digit precision.

Degree-128 minimal polynomial for $\exp(8\pi\phi_2(1/32, 1/32))$

$$-1 + 21888\alpha + 5893184\alpha^{2} + 15077928064\alpha^{3} - 3696628330464\alpha^{4} - 287791501240448\alpha^{5} - 30287462976198976\alpha^{6}$$
$$+4426867843186404992\alpha^{7} - 554156920878198587888\alpha^{8} + 10731545733669133574528\alpha^{9}$$
$$+120048731928709050250048\alpha^{10} + 4376999211577765512726656\alpha^{11} - 27904569345819422212536632\alpha^{12}$$
$$+1874758628778011890385433484 8\alpha^{13} - 643310226865188446831485766208\alpha^{14}$$
$$+12047117225922787772844349665548 8\alpha^{15} - 1172305951003280338849395660913 84\alpha^{16}$$
$$+667772184328316952814362214365568\alpha^{17} - 4130661734713288144037409932696512\alpha^{18}$$
$$+72313626239383964765274946226530432\alpha^{19} - 189142057120586161209180276180914 1088\alpha^{20}$$
$$+3877088173055347147059064106087268 6464\alpha^{21} - 577943965397394779947709633563006963008\alpha^{22}$$
$$+6279796382074485140847650604801614559872\alpha^{23} - 50438907678331243798448849245156136801232\alpha^{24}$$
$$+305806320133365055812520453224169520739712\alpha^{25} - 1441007171934715336769224848138270812591296\alpha^{26}$$
$$+5554617356232728647085822946642640269497472\alpha^{27} - 20280024430170705107000630261773759070647328\alpha^{28}$$
$$+99541720739995105011861264308551867164583808\alpha^{29} - 754081464712315412970559119390477134883548736\alpha^{30}$$
$$+6271958646895434365874802435136411922022336128\alpha^{31} - 45931349314815625339442690290912948480194150172\alpha^{32}$$
$$+280907040806572157908285324812126135484630889344\alpha^{33} - 1427273782916972532576299009596755423149111059136\alpha^{34}$$
$$+6055180299673737231932804443230077408291723908736\alpha^{35} - 21609910939164553316101994301952988793013291135584\alpha^{36}$$
$$+65433275736596914909292838375737685959952141180288\alpha^{37} - 169928170513492897108417040254326115991438719391296\alpha^{38}$$
$$+385709310577705218843549196766620216295554031550592\alpha^{39} - 801233230832691550861608914233661767474963249815792\alpha^{40}$$
$$+1706210557291030772074402183123327251333271061516160\alpha^{41} - 4421210594351357102505784181831242174063263551938496\alpha^{42}$$
$$+14444199585866329915643888187597383540233619718619776\alpha^{43} - 50968478530199956388487913417905125665738409426112032\alpha^{44}$$
$$+169891313454945514927724813351516976839425267825908096\alpha^{45} - 506612996672385619931633440499093959534203673546181440\alpha^{46}$$
$$+1330573388204326565144545192834096788469932897185696896\alpha^{47} - 3069501638444045841407951432645059776135089489403138888\alpha^{48}$$
$$+6226636397646752257692349351542872634032398917736673152\alpha^{49} - 11133383491631126059761752734485434504397040890449485504\alpha^{50}$$
$$+17601823309919260471943648355479182983209248554083752576\alpha^{51} - 24723027443995082126054012492323603544226813344022687712\alpha^{52}$$
$$+31141043717679289808081270766611355726695735914995681664\alpha^{53} - 35982430389670551550204799905599476866868765647852189248\alpha^{54}$$
$$+40292583920117898286863491450657424717015372825433076864\alpha^{55} - 48512188214363976290470868896252008979896310883132967248\alpha^{56}$$
$$+69275112214095149977288310632868535966705567728055958400\alpha^{57} - 114516830148561378617778209682642099604147034577152904128\alpha^{58}$$
$$+195760470467323759899736578743283333538805684128806803072\alpha^{59} - 317349593507106729834513764473487031789280056911012860320\alpha^{60}$$
$$+468944248086031450001465269696090117959962662732817675648\alpha^{61} - 622467103741378906100611838210632752408312516281305008960\alpha^{62}$$
$$+738516443137003178837650661261546833168555909499151978624\alpha^{63} - 781916756680856373187881889706233393197646662361906135622\alpha^{64}$$
$$+738516443137003178837650661261546833168555909499151978624\alpha^{65} - 622467103741378906100611838210632752408312516281305008960\alpha^{66}$$
$$+468944248086031450001465269696090117959962662732817675648\alpha^{67} - 317349593507106729834513764473487031789280056911012860320\alpha^{68}$$
$$+195760470467323759899736578743283333538805684128806803072\alpha^{69} - 114516830148561378617778209682642099604147034577152904128\alpha^{70}$$
$$+69275112214095149977288310632868535966705567728055958400\alpha^{71} - 48512188214363976290470868896252008979896310883132967248\alpha^{72}$$
$$+40292583920117898286863491450657424717015372825433076864\alpha^{73} - 35982430389670551550204799905599476866868765647852189248\alpha^{74}$$
$$+31141043717679289808081270766611355726695735914995681664\alpha^{75} - 24723027443995082126054012492323603544226813344022687712\alpha^{76}$$
$$+17601823309919260471943648355479182983209248554083752576\alpha^{77} - 11133383491631126059761752734485434504397040890449485504\alpha^{78}$$
$$+6226636397646752257692349351542872634032398917736673152\alpha^{79} - 3069501638444045841407951432645059776135089489403138888\alpha^{80}$$
$$+1330573388204326565144545192834096788469932897185696896\alpha^{81} - 506612996672385619931633440499093959534203673546181440\alpha^{82}$$
$$+169891313454945514927724813351516976839425267825908096\alpha^{83} - 50968478530199956388487913417905125665738409426112032\alpha^{84}$$
$$+14444199585866329915643888187597383540233619718619776\alpha^{85} - 4421210594351357102505784181831242174063263551938496\alpha^{86}$$
$$+1706210557291030772074402183123327251333271061516160\alpha^{87} - 801233230832691550861608914233661767474963249815792\alpha^{88}$$
$$+385709310577705218843549196766620216295554031550592\alpha^{89} - 169928170513492897108417040254326115991438719391296\alpha^{90}$$
$$+65433275736596914909292838375737685959952141180288\alpha^{91} - 21609910939164553316101994301952988793013291135584\alpha^{92}$$
$$+6055180299673737231932804443230077408291723908736\alpha^{93} - 1427273782916972532576299009596755423149111059136\alpha^{94}$$
$$+280907040806572157908285324812126135484630889344\alpha^{95} - 45931349314815625339442690290912948480194150172\alpha^{96}$$
$$+6271958646895434365874802435136411922022336128\alpha^{97} - 754081464712315412970559119390477134883548736\alpha^{98}$$
$$+99541720739995105011861264308551867164583808\alpha^{99} - 20280024430170705107000630261773759070647328\alpha^{100}$$
$$+5554617356232728647085822946642640269497472\alpha^{101} - 1441007171934715336769224848138270812591296\alpha^{102}$$
$$+305806320133365055812520453224169520739712\alpha^{103} - 50438907678331243798448849245156136801232\alpha^{104}$$
$$+6279796382074485140847650604801614559872\alpha^{105} - 577943965397394779947709633563006963008\alpha^{106}$$
$$+38770881730553471470590641060872686464\alpha^{107} - 1891420571205861612091802761809141088\alpha^{108}$$
$$+72313626239383964765274946226530432\alpha^{109} - 4130661734713288144037409932696512\alpha^{110}$$
$$+667772184328316952814362214365568\alpha^{111} - 117230595100328033884939566091384\alpha^{112}$$
$$+120471172259227877728443496655488\alpha^{113} - 643310226865188446831485766208\alpha^{114}$$
$$+18747586287780118903854334848\alpha^{115} - 27904569345819422212536632\alpha^{116}$$
$$+4376999211577765512726656\alpha^{117} + 120048731928709050250048\alpha^{118} + 10731545733669133574528\alpha^{119}$$
$$-554156920878198587888\alpha^{120} + 4426867843186404992\alpha^{121} - 30287462976198976\alpha^{122}$$
$$-287791501240448\alpha^{123} - 3696628330464\alpha^{124} + 15077928064\alpha^{125} + 5893184\alpha^{126} + 21888\alpha^{127} - \alpha^{128}$$

High-precision tanh-sinh numerical integration

Given $f(x)$ defined on $(-1, 1)$, define $g(t) = \tanh(\pi/2 \sinh t)$. Then setting $x = g(t)$ yields

$$\int_{-1}^{1} f(x)\, dx \quad = \quad \int_{-\infty}^{\infty} f(g(t))g'(t)\, dt \;\approx\; h \sum_{j=-N}^{N} w_j f(x_j),$$

where $x_j = g(h_j)$ and $w_j = g'(h_j)$. Since $g'(t)$ goes to zero very rapidly for large t, the product $f(g(t))g'(t)$ typically is a nice bell-shaped function, so that the simple summation above converges very rapidly. Reducing h by half typically doubles the number of correct digits.

We have found that tanh-sinh is the best general-purpose integration scheme for functions with vertical derivatives or singularities at endpoints, or for any function at very high precision (> 1000 digits). Otherwise we use Gaussian quadrature.

1. D.H. Bailey, X.S. Li and K. Jeyabalan, "A Comparison of Three High-Precision Quadrature Schemes," *Experimental Mathematics*, vol. 14 (2005), no. 3, pg. 317–329.
2. H. Takahasi and M. Mori, "Double Exponential Formulas for Numerical Integration," *Publications of RIMS*, Kyoto University, vol. 9 (1974), pg. 721-741.

Extended Mordell-Tornheim-Witten functions

In this study, we examine

$$\omega\left(\begin{array}{c|c} s_1,\ldots,s_M & t_1,\ldots,t_N \\ d_1,\ldots,d_M & e_1,\ldots e_N \end{array}\right) := \sum_{\substack{m_1,\ldots,m_M,n_1,\ldots,n_N>0 \\ \sum_{j=1}^M m_j=\sum_{k=1}^N n_k}} \prod_{j=1}^M \frac{(-\log m_j)^{d_j}}{m_j{}^{s_j}} \prod_{k=1}^N \frac{(-\log n_k)^{e_k}}{n_k{}^{t_k}}$$

$$= \frac{1}{2\pi}\int_0^{2\pi} \prod_{j=1}^M \mathrm{Li}_{s_j}^{(d_j)}\left(e^{i\theta}\right) \prod_{k=1}^N \mathrm{Li}_{t_k}^{(e_k)}\left(e^{-i\theta}\right) \mathrm{d}\theta,$$

where the s-th outer derivative of a polylogarithm is denoted $\mathrm{Li}_s^{(d)}(z) := \left(\frac{\partial}{\partial s}\right)^d \mathrm{Li}_s(z)$. These are a generalization of the class of Mordell-Tornheim-Witten sums, which appear in many contexts, including combinatorics, number theory and mathematical physics.

1. D.H. Bailey and J.M. Borwein, "On Eulerian log-gamma integrals and Tornheim-Witten zeta functions," *Ramanujan Journal*, 27 Feb 2013, available at http://www.davidhbailey.com/dhbpapers/log-gamma.pdf.
2. D.H. Bailey, J.M. Borwein and A. Mattingly, "Computation and theory of extended Mordell-Tornheim-Witten sums II," manuscript, available from author.

Computation of polylogarithm outer derivatives to high precision

Fix $k = 0, 1, 2 \dots$ and $m = 1, 2 \dots$. For $|\log z| < 2\pi$ and $\mathcal{L} = \log(-\log z)$

$$\mathrm{Li}_{k+1}^{(m)}(z) \quad = \quad \textstyle\sum_{0 \le n \ne k} \zeta^{(m)}(k+1-n)\frac{\log^n z}{n!} + m!\, c_{k,m}(\mathcal{L})\frac{\log^k z}{k!}.$$

Here, for $k \geq 1$

$$\begin{aligned} c_{k,j}(\mathcal{L}) &= \frac{(-1)^j}{j!}\gamma_j - b_{k,j+1}(\mathcal{L}) \\ b_{k,m}(\mathcal{L}) &:= \sum_{\substack{p+t+q=j \\ p,t,q \ge 0}} \frac{\mathcal{L}^p}{p!}\frac{\Gamma^{(t)}(1)}{t!}(-1)^t f_{k,q}, \end{aligned}$$

where γ_j are Stieltjes gamma constants, and $f_{k,q}$ are defined recursively by $f_{0,0} = 1$, $f_{0,q} = 0$ $(q > 0)$, $f_{k,0} = 1$ $(k > 0)$ and $f_{k,q} = \sum_{h=0}^{q}(-1)^h f_{k-1,q-h}/k^h$.

Additional computational machinery is required to compute derivatives of zeta function at negative integers and derivatives of gamma function at 1.

Why not just use Maple or Mathematica?

Performing the computations just described is very demanding. We use custom-written code and the ARPREC high-precision arithmetic package (due to DHB and others). Neither Mathematica nor Maple is up to the task:

1. Maple cannot compute numerical values of outer derivatives of polylogarithms.
2. Mathematica computes numerical values of outer derivatives of polylogarithms, but in some cases they are not correct to the specified number of digits.
3. Mathematica refuses to compute $\zeta^{(m)}(0)$ for $m >= 4$. It returns the expression unevaluated. Wolfram Alpha produces analytic expressions for these, but will only give numerical to a few digits unless one signs up for Wolfram Alpha Pro.
4. Mathematica's computations of $\zeta^{(m)}(-n)$ to high precision often produce strange internal errors.
5. Our custom-written code, using ARPREC high-precision software is many times faster than either Maple or Mathematica for performing high-precision numerical integration (quadrature).

Order three omega constants

For nonnegative a, b, c, q, r, s, we focus on this particular class of omega constants:

$$\omega_{a,b,c}(q,r,s) = \omega\left(\begin{array}{cc|c} q\ , & r & s \\ a\ , & b & c \end{array}\right) = \int_0^\infty \left(\frac{x^{s-1}}{\Gamma(s)}\right)^{(c)} \mathrm{Li}_q^{(a)}(e^{-x})\,\mathrm{Li}_r^{(b)}(e^{-x})\,\mathrm{d}x,$$

which is valid when $q \geq 0, r \geq 0, s > 0$, with $q + r + s > 2$, and $a \geq 0, b \geq 0, c \geq 0$. Here the notation $(\cdot)^{(c)}$ denotes the c-th partial derivative of the expression in parentheses with respect to s. For computational purposes, it is necessary to break the integral into two pieces, and then using the substitution $u = e^{-x}$ for the second:

$$\begin{aligned} \omega_{a,b,c}(q,r,s) &= \int_0^1 \left(\frac{x^{s-1}}{\Gamma(s)}\right)^{(c)} \mathrm{Li}_q^{(a)}(e^{-x})\,\mathrm{Li}_r^{(b)}(e^{-x})\,\mathrm{d}x \\ &= \quad + \int_0^{1/e} \left(\frac{(-\log u)^{s-1}}{\Gamma(s)}\right)^{(c)} \mathrm{Li}_q^{(a)}(u)\,\mathrm{Li}_r^{(b)}(u)\,\frac{\mathrm{d}u}{u}. \end{aligned}$$

Small sample of relations discovered by PSLQ among omega constants

$$
\begin{aligned}
0 &= \omega_{0,0,3}(0,1,2) - \omega_{0,3,0}(0,1,2) - \omega_{0,3,0}(1,0,2) + \zeta^{(3)}(3) \\
0 &= \omega_{0,1,1}(0,1,3) + \omega_{0,1,1}(1,0,3) + \omega_{0,1,1}(2,0,2) - \omega_{0,1,1}(2,1,1) \\
0 &= -\omega_{0,2,1}(0,1,3) - \omega_{0,2,1}(0,2,2) - \omega_{0,2,1}(1,0,3) + \omega_{0,2,1}(1,2,1) \\
0 &\stackrel{?}{=} \omega_{0,0,3}(0,1,3) - \omega_{0,1,2}(0,2,2) - \omega_{0,2,1}(0,2,2) - \omega_{0,3,0}(0,1,3) - \omega_{0,3,0}(0,2,2) \\
&\quad -\omega_{0,3,0}(1,0,3) + 2\omega_{1,2,0}(0,1,3) + \omega_{1,2,0}(0,2,2) + 2\omega_{1,2,0}(1,0,3) + \omega_{1,2,0}(2,0,2) \\
0 &= \omega_{1,3,0}(0,1,3) + \omega_{1,3,0}(1,0,3) + \omega_{1,3,0}(2,0,2) - \omega_{1,3,0}(2,1,1) \\
0 &\stackrel{?}{=} -2\omega_{0,0,4}(0,1,3) - \omega_{0,0,4}(0,2,2) + 4\omega_{0,4,0}(0,1,3) + 2\omega_{0,4,0}(0,2,2) + 4\omega_{0,4,0}(1,0,3) \\
&\quad +\omega_{0,4,0}(2,0,2) - 3\zeta^{(4)}(4) \\
0 &\stackrel{?}{=} -\omega_{1,2,0}(0,1,4) - \omega_{1,2,0}(1,0,4) - \omega_{1,2,0}(2,0,3) + \omega_{1,2,0}(2,1,2) \\
0 &\stackrel{?}{=} -3\omega_{0,0,4}(0,1,4) - \omega_{0,0,4}(0,2,3) + 6\omega_{0,4,0}(0,1,4) + 5\omega_{0,4,0}(0,2,3) + 3\omega_{0,4,0}(0,3,2) \\
&\quad +6\omega_{0,4,0}(1,0,4) + \omega_{0,4,0}(2,0,3) - 4\zeta^{(4)}(5)
\end{aligned}
$$

Provable relations among the order-three omegas

The omegas satisfy relations such as

$$
\begin{aligned}
\omega_{a,b,c}(q,r,s) &= \omega_{b,a,c}(r,q,s) \\
\omega_{a,b,c}(r,s,t-1) &= \omega_{a,b,c}(r-1,s,t) + \omega_{a,b,c}(r,s-1,t) \\
\omega_{a,b,0}(s,t,0) - \omega_{a,0,b}(t,0,s) - \omega_{a,0,b}(s,0,t) &= \zeta^{(a+b)}(t+s) \\
\omega_{a,b,c}(r,s,t) &= \sum_{i=1}^{r} \binom{r+s-i-1}{s-1} \omega_{a,b,c}(i,0,N-i) \\
&\quad + \sum_{i=1}^{s} \binom{r+s-i-1}{r-1} \omega_{a,b,c}(0,i,N-i)
\end{aligned}
$$

We conjecture that all relations found by PSLQ are generated by these relations.

Summary

This talk is available at http://www.davidhbailey.com/dhbtalks/dhb-wcnt13.pdf.