Computational challenges in experimental mathematics

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

21 Jul 2014

Numerical reproducibility in high-performance computing

A December 2012 ICERM workshop on reproducibility in high-performance computing noted:

> *Numerical round-off error and numerical differences are greatly magnified as computational simulations are scaled up to run on highly parallel systems. As a result, it is increasingly difficult to determine whether a code has been correctly ported to a new system, because computational results quickly diverge from standard benchmark cases. And it is doubly difficult for other researchers, using independently written codes and distinct computer systems, to reproduce published results.*

- V. Stodden, D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider and W. Stein, "Setting the default to reproducible: Reproducibility in computational and experimental mathematics," Jan 2013, available at `http://www.davidhbailey.com/dhbpapers/icerm-report.pdf`.

Analysis of collisions at the Large Hadron Collider

- The 2012 discovery of the Higgs boson at the ATLAS experiment in the LHC relied crucially on the ability to track charged particles with exquisite precision (10 microns over a 10m length) and high reliability (over 99% of roughly 1000 charged particles per collision correctly identified).
- Software: 5 millions line of C++ and python code, developed by roughly 2000 physicists and engineers over 15 years.
- Recently, in an attempt to speed up the calculation, researchers found that merely changing the underlying math library resulted in some collisions being missed or misidentified.

Questions:

- How serious are these numerical difficulties?
- How can they be tracked down?
- How can the library be maintained, producing numerically reliable results?

U.C. Berkeley's CORVETTE project and the "Precimonious" tool

Objective: Develop software facilities to find and ameliorate numerical anomalies in large-scale computations:

- Facilities to test the level of numerical accuracy required for an application.
- Facilities to delimit the portions of code that are inaccurate.
- Facilities to search the space of possible code modifications.
- Facilities to repair numerical difficulties, including usage of high-precision arithmetic.
- Facilities to navigate through a hierarchy of precision levels (32-bit, 64-bit, 80-bit or higher as needed).

The current version of this tool is known as "Precimonious."

- C. Rubio-Gonzalez, C. Nguyen, H. D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey and C. Iancu, "Precimonious: Tuning assistant for floating-point precision," *Proceedings of SC13*, 2013.

Innocuous example where high precision is required for reproducible results

Problem: Find a polynomial to fit the data $(1, 1048579, 16777489, 84941299, 268501249, 655751251, 1360635409, 2523398179, 4311748609)$ for arguments $0, 1, \cdots, 8$. The usual approach is to solve the linear system:

$$\begin{bmatrix} n & \sum_{k=1}^{n} x_k & \cdots & \sum_{k=1}^{n} x_k^n \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 & \cdots & \sum_{k=1}^{n} x_k^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} x_k^n & \sum_{k=1}^{n} x_k^{n+1} & \cdots & \sum_{k=1}^{n} x_k^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} y_k \\ \sum_{k=1}^{n} x_k y_k \\ \vdots \\ \sum_{k=1}^{n} x_k^n y_k \end{bmatrix}$$

A 64-bit computation (e.g., using Matlab, Linpack or LAPACK) fails to find the correct polynomial in this instance, even if one rounds results to nearest integer.

However, if Linpack routines are converted to use double-double arithmetic (31-digit accuracy), the above computation quickly produces the correct polynomial:

$$f(x) \;=\; 1 + 1048577x^4 + x^8 \;=\; 1 + (2^{20} + 1)x^4 + x^8$$

Innocuous example, continued

The result on the previous page can be obtained with double precision using Lagrange interpolation or the Demmel-Koev algorithm. But few scientists, outside of expert numerical analysts, are aware of these schemes.

Besides, even these schemes fail for higher-degree problems. For example:
$(1, 134217731, 8589938753, 97845255883, 549772595201,$
$2097396156251, 6264239146561, 15804422886323, 35253091827713,$
$71611233653971, 135217729000001, 240913322581691, 409688091758593)$
is generated by:

$$f(x) \;=\; 1 + 134217729x^6 + x^{12} \;=\; 1 + (2^{27} + 1)x^6 + x^{12}$$

Neither the Lagrange or Demmel-Koev algorithms get the right answer with double precision. In contrast, a straightforward Linpack scheme, implemented with double-double arithmetic, works fine for this and a wide range of similar problems.

Numerical analysis experts or high precision?

2010 U.C. Berkeley statistics:

- 870 seniors graduated in the Division of Mathematical and Physical Sciences (including Mathematics, Physics and Statistics), the College of Chemistry and the College of Engineering (including Computer Science).
- Many graduates in other fields (biology, economics, finance, medicine, psychology, etc.) will also do significant numerical computing in their professional work.
- 219 students enrolled in two sections of Math 128A, a one-semester introductory numerical analysis course required of applied math majors.
- Only 24 enrolled in Math 128B, a more advanced course.

Conclusion: Only about 2% of Berkeley graduates that will do scientific computing in their career work have had advanced training in numerical analysis.

Other applications where high-precision arithmetic is useful or essential

1. Planetary orbit calculations (32 digits).
2. Supernova simulations (32–64 digits).
3. Certain components of climate modeling (32 digits).
4. Coulomb n-body atomic system simulations (32–120 digits).
5. Schrodinger solutions for lithium and helium atoms (32 digits).
6. Electromagnetic scattering theory (32–100 digits).
7. Scattering amplitudes of fundamental particles (32 digits).
8. Discrete dynamical systems (32 digits).
9. Theory of nonlinear oscillators (64 digits).
10. The Taylor algorithm for ODEs (100–600 digits).
11. Ising integrals from mathematical physics (100–1000 digits).
12. Problems in experimental mathematics (100–50,000 digits).

- D. H. Bailey, R. Barrio, and J. M. Borwein, "High precision computation: Mathematical physics and dynamics," *Applied Mathematics and Computation*, vol. 218 (2012), pg. 10106–10121.

High-precision arithmetic in experimental mathematics

Typical methodology:

1. Compute various mathematical entities (limits, infinite series sums, definite integrals, etc.) to high precision, typically 100–10,000 digits.
2. Use an integer relation algorithm such as "PSLQ" to recognize these numerical values in terms of well-known mathematical constants.
3. Devise formal mathematical proofs of the discovered relations.

This approach has been extremely fruitful — literally thousands of new results.

What's more, this is a very accessible form of research — even undergraduates (if they have good programming skills) can make state-of-the-art contributions.

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.

- 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.

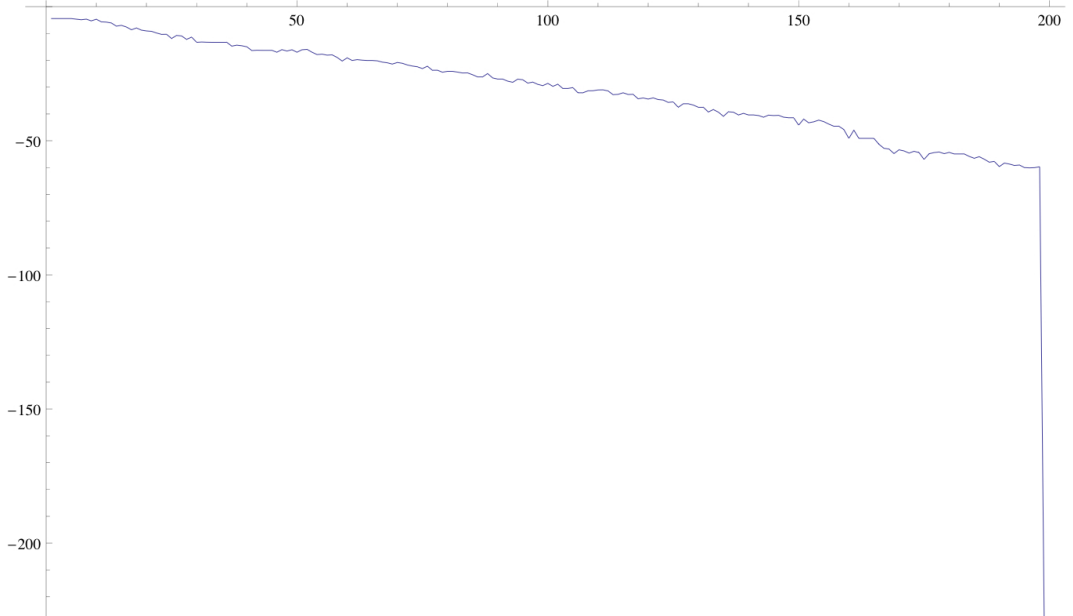
PSLQ, continued

- 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



Simple application of PSLQ: Find minimal polynomial of algebraic number

Suppose we are given an algebraic number α, i.e., α is a root of an algebraic equation with integer coefficients: $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$. How can we reconstruct the minimal polynomial of α, knowing just the degree n (or some estimate of n) and the numerical value of α?

Solution: Compute the vector $x = (1, \alpha, \alpha^2, \cdots, \alpha^n)$, and then apply PSLQ to this $(n+1)$-long vector.

Example: Find the minimal polynomial for this constant, assuming that it is of degree no more than 10:

$$87.055259441185777057425040060690481806178409055793\ldots$$

Answer:

$$0 = 1 - 88\alpha + 92\alpha^2 - 872\alpha^3 + 1990\alpha^4 - 872\alpha^5 + 92\alpha^6 - 88\alpha^7 + \alpha^8$$

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. 897–925.

Some other BBP-type formulas found using PSLQ

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

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

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

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

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

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

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.

- 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.

Ising integrals from mathematical physics

We studied these three families of integrals: C_n arise in quantum field theory (see next viewgraph); D_n arise in Ising theory of mathematical physics (referred to us by Craig Tracy of U.C. Davis); E_n are the numerators of D_n:

$$C_n := \frac{4}{n!} \int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$D_n := \frac{4}{n!} \int_0^\infty \cdots \int_0^\infty \frac{\prod_{i<j} \left(\frac{u_i - u_j}{u_i + u_j}\right)^2}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$E_n = 2 \int_0^1 \cdots \int_0^1 \left(\prod_{1 \le j < k \le n} \frac{u_k - u_j}{u_k + u_j} \right)^2 \mathrm{d}t_2 \, dt_3 \cdots \mathrm{d}t_n$$

where in the last line $u_k = t_1 t_2 \cdots t_k$.

- D. H. Bailey, J. M. Borwein and R. E. Crandall, "Integrals of the Ising class," *Journal of Physics A: Mathematical and General*, vol. 39 (2006), pg. 12271–12302.

Limiting value of C_n: What is this number?

Key observation: The C_n integrals can be converted to one- dimensional integrals involving the modified Bessel function $K_0(t)$:

$$C_n \;=\; \frac{2^n}{n!}\int_0^\infty t K_0^n(t)\,\mathrm{d}t$$

High-precision numerical values, computed using this formula and tanh-sinh quadrature, approach a limit. For example:

$$C_{1024} \;=\; 0.630473503374386796122040192710878904354587078 7\ldots$$

What is this number? We copied the first 50 digits into the online Inverse Symbolic Calculator (ISC) at `http://carma-lx1.newcastle.edu.au:8087`. The result was:

$$\lim_{n\to\infty} C_n \;=\; 2e^{-2\gamma}.$$

where γ denotes Euler's constant. This is now proven.

Other Ising integral evaluations found using PSLQ

$$
\begin{aligned}
D_2 &= 1/3 \\
D_3 &= 8 + 4\pi^2/3 - 27\,\mathrm{Li}_{-3}(2) \\
D_4 &= 4\pi^2/9 - 1/6 - 7\zeta(3)/2 \\
E_2 &= 6 - 8\log 2 \\
E_3 &= 10 - 2\pi^2 - 8\log 2 + 32\log^2 2 \\
E_4 &= 22 - 82\zeta(3) - 24\log 2 + 176\log^2 2 - 256(\log^3 2)/3 \\
&\quad +16\pi^2\log 2 - 22\pi^2/3 \\
E_5 &= 42 - 1984\,\mathrm{Li}_4(1/2) + 189\pi^4/10 - 74\zeta(3) - 1272\zeta(3)\log 2 \\
&\quad +40\pi^2\log^2 2 - 62\pi^2/3 + 40(\pi^2\log 2)/3 + 88\log^4 2 \\
&\quad +464\log^2 2 - 40\log 2
\end{aligned}
$$

where $\zeta(x)$ is the Riemann zeta function and $\mathrm{Li}_n(x)$ is the polylogarithm function.

The Ising integral E_5

We were able to reduce E5 to an extremely complicated 3-D integral (see right).

We computed this integral to 250-digit precision, using a highly parallel, high-precision 3-D quadrature program. Then we used a PSLQ program to discover the evaluation given on the previous page.

We also computed D_5 to 500 digits, but were unable to identify it. In March 2014, Erik Panzer proved our formula for E_5 and found a similar evaluation for D_5.

1. D. H. Bailey, J. M. Borwein and R. E. Crandall, "Integrals of the Ising class," *J. Physics A: Math. and Gen.*, vol. 39 (2006), pg. 12271–12302.
2. E. Panzer, "Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals," manuscript, 2014.

$$
\begin{aligned}
E_5 \;=\; & \int_0^1\int_0^1\int_0^1 \left[2(1-x)^2(1-y)^2(1-xy)^2(1-z)^2(1-yz)^2(1-xyz)^2\right. \\
& \left(-\left[4(x+1)(xy+1)\log(2)\left(y^3z^3x^7-y^4z^2(4(y+1)z+3)x^6-y^3z\left(y^2+1\right)z^2+4(y+\right.\right.\right. \\
& 1)z+5)x^5+y^2\left(4y(y+1)z^3+3\left(y^2+1\right)z^2+4(y+1)z-1\right)x^4+y\left(z\left(z^2+4z\right.\right. \\
& \left.+5\right)y^2+4\left(z^2+1\right)y+5z+4\right)x^3+\left(\left(-3z^2-4z+1\right)y^2-4zy+1\right)x^2-(y(5z+4) \\
& \left.+4)x-1)\right]/\left[(x-1)^3(xy-1)^3(xyz-1)^3\right]+\left[3(y-1)^2y^4(z-1)^2z^2(yz\right. \\
& -1)^2x^6+2y^3z\left(3(z-1)^2z^3y^5+z^2\left(5z^3+3z^2+3z+5\right)y^4+(z-1)^2z\right. \\
& \left(5z^2+16z+5\right)y^3+\left(3z^5+3z^4-22z^3-22z^2+3z+3\right)y^2+3\left(-2z^4+z^3+2\right. \\
& z^2+z-2\left.\right)y+3z^3+5z^2+5z+3\left.\right)x^5+y^2\left(7(z-1)^2z^4y^6-2z^3\left(z^3+15z^2\right.\right. \\
& \left.+15z+1\right)y^5+2z^2\left(-21z^4+6z^3+14z^2+6z-21\right)y^4-2z\left(z^5-6z^4-27z^3\right. \\
& -27z^2-6z+1\left.\right)y^3+\left(7z^6-30z^5+28z^4+54z^3+28z^2-30z+7\right)y^2-2\left(7z^5\right. \\
& \left.+15z^4-6z^3-6z^2+15z+7\right)y+7z^4-2z^3-42z^2-2z+7\left.\right)x^4-2y\left(z^3\left(z^3\right.\right. \\
& \left.-9z^2-9z+1\right)y^6+z^2\left(7z^4-14z^3-18z^2-14z+7\right)y^5+z\left(7z^5+14z^4+3\right. \\
& z^3+3z^2+14z+7\left.\right)y^4+\left(z^6-14z^5+3z^4+84z^3+3z^2-14z+1\right)y^3-3\left(3z^5\right. \\
& \left.+6z^4-z^3-z^2+6z+3\right)y^2-\left(9z^4+14z^3-14z^2+14z+9\right)y+z^3+7z^2+7z \\
& \left.+1\right)x^3+\left(z^2\left(11z^4+6z^3-66z^2+6z+11\right)y^6+2z\left(5z^5+13z^4-2z^3-2z^2\right.\right. \\
& \left.+13z+5\right)y^5+\left(11z^6+26z^5+44z^4-66z^3+44z^2+26z+11\right)y^4+\left(6z^5-4\right. \\
& z^4-66z^3-66z^2-4z+6\left.\right)y^3-2\left(33z^4+2z^3-22z^2+2z+33\right)y^2+\left(6z^3+26\right. \\
& z^2+26z+6\left.\right)y+11z^2+10z+11\left.\right)x^2-2\left(z^2\left(5z^3+3z^2+3z+5\right)y^5+z\left(22z^4\right.\right. \\
& \left.+5z^3-22z^2+5z+22\right)y^4+\left(5z^5+5z^4-26z^3-26z^2+5z+5\right)y^3+\left(3z^4-\right. \\
& \left.22z^3-26z^2-22z+3\right)y^2+\left(3z^3+5z^2+5z+3\right)y+5z^2+22z+5\left.\right)x+15z^2+2z \\
& +2y(z-1)^2(z+1)+2y^3(z-1)^2z(z+1)+y^4z^2\left(15z^2+2z+15\right)+y^2\left(15z^4\right. \\
& \left.-2z^3-90z^2-2z+15\right)+15\left.\right]/\left[(x-1)^2(y-1)^2(xy-1)^2(z-1)^2(yz-1)^2\right. \\
& \left.(xyz-1)^2\right]-\left[4(x+1)(y+1)(yz+1)\left(-z^2y^4+4z(z+1)y^3+\left(z^2+1\right)y^2\right.\right. \\
& -4(z+1)y+4x\left(y^2-1\right)\left(y^2z^2-1\right)+x^2\left(z^2y^4-4z(z+1)y^3-\left(z^2+1\right)y^2\right. \\
& \left.+4(z+1)y+1\right)-1\left.\right)\log(x+1)\left.\right]/\left[(x-1)^3x(y-1)^3(yz-1)^3\right]-\left[4(y+1)(xy\right. \\
& \left.+1)(z+1)\left(x^2\left(z^2-4z-1\right)y^4+4x(x+1)\left(z^2-1\right)y^3-\left(x^2+1\right)\left(z^2-4z-1\right)\right.\right. \\
& \left.y^2-4(x+1)\left(z^2-1\right)y+z^2-4z-1\right)\log(xy+1)\left.\right]/\left[x(y-1)^3y(xy-1)^3(z-\right. \\
& \left.1)^3\right]-\left[4(z+1)(yz+1)\left(x^3y^5z^7+x^2y^4(4x(y+1)+5)z^6-xy^3\left((y^2+\right.\right.\right. \\
& \left.1)x^2-4(y+1)x-3\right)z^5-y^2\left(4y(y+1)x^3+5\left(y^2+1\right)x^2+4(y+1)x+1\right)z^4+ \\
& y\left(y^2x^3-4y(y+1)x^2-3\left(y^2+1\right)x-4(y+1)\right)z^3+\left(5x^2y^2+y^2+4x(y+1)\right. \\
& \left.\left.y+1\right)z^2+\left((3x+4)y+4\right)z-1\right)\log(xyz+1)\left.\right]/\left[xy(z-1)^3z(yz-1)^3(xyz-1)^3\right]\left.\right] \\
& /\left[(x+1)^2(y+1)^2(xy+1)^2(z+1)^2(yz+1)^2(xyz+1)^2\right]\,dx\,dy\,dz
\end{aligned}
$$

Box integrals

The following integrals appear in numerous applications:

$$B_n(s) := \int_0^1 \cdots \int_0^1 \left(r_1^2 + \cdots + r_n^2\right)^{s/2} \, \mathrm{d}R$$

$$\Delta_n(s) := \int_0^1 \cdots \int_0^1 \left((r_1 - q_1)^2 + \cdots + (r_n - q_n)^2\right)^{s/2} \, \mathrm{d}R\mathrm{d}Q$$

- $B_n(1)$ is average distance of a random point from the origin.
- $\Delta_n(1)$ is average distance between two random points.
- $B_n(-n+2)$ is average electrostatic potential in an n-cube whose origin has a unit charge.
- $\Delta_n(-n+2)$ is average electrostatic energy between two points in a uniform n-cube of charged "jellium."
- Recently integrals of this type have arisen in neuroscience, e.g. the average distance between synapses in a mouse brain.

- D. H. Bailey, J. M. Borwein and R. E. Crandall, "Box integrals," *Journal of Computational and Applied Mathematics*, vol. 206 (2007), pg. 196–208.

Sample evaluations of box integrals

n	s	$B_n(s)$
any	even $s \geq 0$	rational, e.g., : $B_2(2) = 2/3$
1	$s \neq -1$	$\frac{1}{s+1}$
2	-4	$-\frac{1}{4} - \frac{\pi}{8}$
2	-3	$-\sqrt{2}$
2	-1	$2\log(1+\sqrt{2})$
2	1	$\frac{1}{3}\sqrt{2} + \frac{1}{3}\log(1+\sqrt{2})$
2	3	$\frac{7}{5}\sqrt{2} + \frac{3}{20}\log(1+\sqrt{2})$
2	$s \neq -2$	$\frac{2}{2+s}{}_2F_1\left(\frac{1}{2}, -\frac{s}{2}; \frac{3}{2}; -1\right)$
3	-5	$-\frac{1}{6}\sqrt{3} - \frac{1}{12}\pi$
3	-4	$-\frac{3}{2}\sqrt{2}\arctan\frac{1}{\sqrt{2}}$
3	-2	$-3G + \frac{3}{2}\pi\log(1+\sqrt{2}) + 3\ \mathrm{Ti}_2(3-2\sqrt{2})$
3	-1	$-\frac{1}{4}\pi + \frac{3}{2}\log\left(2+\sqrt{3}\right)$
3	1	$\frac{1}{4}\sqrt{3} - \frac{1}{24}\pi + \frac{1}{2}\log\left(2+\sqrt{3}\right)$
3	3	$\frac{2}{5}\sqrt{3} - \frac{1}{60}\pi - \frac{7}{20}\log\left(2+\sqrt{3}\right)$

Here F is hypergeometric function; G is Catalan; Ti is Lewin's inverse-tan function.

Lessons learned from the Ising integral, box integral and similar projects

- High-precision computation (typically 400–4000 digits), combined with the PSLQ algorithm, is very effective in evaluating and identifying definite integrals.
- The tanh-sinh scheme is very effective in evaluating a broad range of integrals to very high precision, even including functions with singularities at endpoints.
- For most of these problems, the numerical integration is the pacing challenge.
- Often highly parallel supercomputers are required to obtain sufficiently high-precision results.
- By comparison, running PSLQ to identify numerical values is usually very cheap.

Thus, more efficient algorithms are needed to evaluate the following to high precision:

- Basic arithmetic operations and transcendental functions.
- A wide range of special functions.
- Ordinary one-dimensional integrals.
- Multi-dimensional integrals (this is particularly expensive).

High-precision evaluation of transcendental and special functions

1. Basic transcendentals—exp, log, sin, cos, tan, hyperbolic functions—and the corresponding inverse functions.
2. Gamma, digamma, polygamma, incomplete gamma, beta and incomplete beta functions.
3. Riemann zeta function, polylogarithms and Dirichlet L-functions.
4. Bessel functions (first, second and third kinds, modified, etc.).
5. Hypergeometric functions.
6. Airy functions.
7. Elliptic integral functions.
8. Jacobian elliptic functions and Weierstrass elliptic/modular functions.
9. Theta functions.

Also, derivatives are needed for all of the above.

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.

- 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.

A fast series to compute $\pi_2(x, y)$

The following series, due to the late Richard Crandall, can be used to rapidly compute $\pi_2(x, y)$ to high precision:

$$\phi_2(x,y) \;=\; \frac{1}{4\pi}\log\frac{\cosh(\pi x)+\cos(\pi y)}{\cosh(\pi x)-\cos(\pi y)} - \frac{2}{\pi}\sum_{\mathbf{m}\in\mathbb{O}^+}\frac{\cosh(\pi m x)\cos(\pi m y)}{m(1+e^{\pi m})}.$$

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 corresponding to $x = y = 1/32$ 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.

128-degree polynomial corresponding to $x = y = 1/32$

$$
\begin{aligned}
&- 1 + 21888\alpha + 5893184\alpha^2 + 15077928064\alpha^3 - 3696628330464\alpha^4 - 287791501240448\alpha^5 - 30287462976198976\alpha^6 + 4426867843186404992\alpha^7 \\
&- 554156920878198587888\alpha^8 + 107315457336691335745428\alpha^9 + 12004873192870905025004 8\alpha^{10} + 437699921157776551272665 6\alpha^{11} - 279045693458194222125366432\alpha^{12} \\
&+ 187475862877801180938543348 48\alpha^{13} - 643310226865188446831485766208\alpha^{14} + 120471172259227877284434966554 88\alpha^{15} - 117230595100328033884939566091384\alpha^{16} \\
&+ 667772184328316952814362214365568\alpha^{17} - 413066173471328814403740993269651 2\alpha^{18} + 723136262393839647652749462265304 32\alpha^{19} \\
&- 189142057120586161209180276180914108 8\alpha^{20} + 387708817305534714705906410608726864 64\alpha^{21} - 577943965397394779947709633563006963008\alpha^{22} \\
&+ 627979638207448514084765060480161455987 2\alpha^{23} - 504389076783312437984488492451561368012 32\alpha^{24} + 305806320133365055812520453224169520739712\alpha^{25} \\
&- 144100717193471533676922484813827081259129 6\alpha^{26} + 555461735623272864708582294664264026949747 2\alpha^{27} - 202800244301707051070006302617737590706473 28\alpha^{28} \\
&+ 995417207399951050118612643085518671645838 08\alpha^{29} - 754081464712315412970559119390477134883548736\alpha^{30} \\
&+ 627195864689543436587480243513641192202233612 8\alpha^{31} - 459313493148156253394426902909129484801941501 72\alpha^{32} \\
&+ 280907040806572157908285324812126135484630889344\alpha^{33} - 142727378291697253257629900959675542314911105913 6\alpha^{34} \\
&+ 605518029967373723193280444323007740829172390873 6\alpha^{35} - 216099109391645533161019943019529887930132911355 84\alpha^{36} \\
&+ 654332757365969149092928383757376859599521411802 88\alpha^{37} - 169928170513492897108417040254326115991438719391296\alpha^{38} \\
&+ 385709310577705218843549196766620216295554031550592\alpha^{39} - 801233230832691550861608914233661767474963249815792\alpha^{40} \\
&+ 170621055729103077207440218312332725133327106151616 0\alpha^{41} - 442121059435135710250578418183124217406326551938496\alpha^{42} \\
&+ 144441995858663299156438881875973835402336197186197776\alpha^{43} - 509684785301999563884879134179051256657384094261120 32\alpha^{44} \\
&+ 169891313454945514927724813351516976839425267825908096\alpha^{45} - 506612996672385619931633440499093959534203673546181440\alpha^{46} \\
&+ 133057338820432656514454519283409678846993289718569689 6\alpha^{47} - 306950163844404584140795143264505977613508948940313888 8\alpha^{48} \\
&+ 622663639764675225769234935154287263403239891773667315 2\alpha^{49} - 111333834916311260597617527344854345043970408904494855 04\alpha^{50} \\
&+ 176018233099192604719436483554791829832092485540837525 76\alpha^{51} - 247230274439950821260540124923236035442268133440226877 12\alpha^{52} \\
&+ 311410437176792898080812707666113557266957359149956816 64\alpha^{53} - 359824303896705515502047999055994768668687656478521892 48\alpha^{54}
\end{aligned}
$$

128-degree polynomial corresponding to $x = y = 1/32$, continued

$$+ 402925839201178982686349145065742471701537282543307686 4\alpha^{55} - 485121882143639762904708688962520089798963108831329672 48\alpha^{56}$$
$$+ 692751122140951499772883106328685359667055677280559584 00\alpha^{57} - 114516830148561378617778209682642099604147034577152904128\alpha^{58}$$
$$+ 195760470467323759899736578743283333538805684128806803072\alpha^{59} - 317349593507106729834513764473487031789280056911012860320\alpha^{60}$$
$$+ 468944248086031450001465269696090117959962662732817675648\alpha^{61} - 622467103741378906100611838210632752408312516281305008960\alpha^{62}$$
$$+ 738516443137003178837650661261546833168555909499151978624\alpha^{63} - 781916756680856373187881889706233393197646662361906135622\alpha^{64}$$
$$+ 738516443137003178837650661261546833168555909499151978624\alpha^{65} - 622467103741378906100611838210632752408312516281305008960\alpha^{66}$$
$$+ 468944248086031450001465269696090117959962662732817675648\alpha^{67} - 317349593507106729834513764473487031789280056911012860320\alpha^{68}$$
$$+ 195760470467323759899736578743283333538805684128806803072\alpha^{69} - 114516830148561378617778209682642099604147034577152904128\alpha^{70}$$
$$+ 692751122140951499772883106328685359667055677280559584 00\alpha^{71} - 485121882143639762904708688962520089798963108831329672 48\alpha^{72}$$
$$+ 402925839201178982686349145065742471701537282543307686 4\alpha^{73} - 359824303896705515502047999055994768668687656478521892 48\alpha^{74}$$
$$+ 311410437176792898080812707666113557266957359149956816 64\alpha^{75} - 247230274439950821260540124923236035442268133440226877 12\alpha^{76}$$
$$+ 176018233099192604719436483554791829832092485540837525 76\alpha^{77} - 111333834916311260597617527344854345043970408904494855 04\alpha^{78}$$
$$+ 622663639764675225769234935154287263403239891773667 3152\alpha^{79} - 306950163844404584140795143264505977613508948940313888 8\alpha^{80}$$
$$+ 133057338820432656514454519283409678846993289718569689 6\alpha^{81} - 506612996672385619931633440499093959534203673546181440\alpha^{82}$$
$$+ 169891313454945514927724813351516976839425267825908096\alpha^{83} - 509684785301999563884879134179051256657384094261120 32\alpha^{84}$$
$$+ 144441995858663299156438881875973835402336197181961977 6\alpha^{85} - 442121059435135710250578418183124217406326355193849 6\alpha^{86}$$
$$+ 170621055729103077207440218312332725133327106151616 0\alpha^{87} - 801233230832691550861608914233661767474963249815792\alpha^{88}$$
$$+ 385709310577705218843549196766620216295554031550592\alpha^{89} - 169928170513492897108417040254326115991438719391296\alpha^{90}$$
$$+ 654332757365969149092928383757376859595952141180288\alpha^{91} - 216099109391645533161019943019529887930132911355 84\alpha^{92}$$
$$+ 605518029967373723193280444323007740829172390873 6\alpha^{93} - 142727378291697253257629900959675542314911105913 6\alpha^{94}$$

128-degree polynomial corresponding to $x = y = 1/32$, continued

$$
\begin{aligned}
&+ 28090704080657215790828532481212613548463088934 4\alpha^{95} - 45931349314815625339442690290912948480194150172\alpha^{96} \\
&+ 62719586468954343658748024351364119220223361 28\alpha^{97} - 75408146471231541297055911939047713488354873 6\alpha^{98} \\
&+ 99541720739995105011861264308551867164583808\alpha^{99} - 20280024430170705107000630261773759070647328\alpha^{100} \\
&+ 55546173562327286470858229466426402694974 72\alpha^{101} - 14410071719347153367692248481382708125912 96\alpha^{102} \\
&+ 30580632013336505581252045322416952073971 2\alpha^{103} - 50438907678331243798448849245156136801232\alpha^{104} \\
&+ 62797963820744851408476506048016145598 72\alpha^{105} - 57794396539739477994770963356300696300 8\alpha^{106} \\
&+ 38770881730553471470590641060872686464\alpha^{107} - 18914205712058616120918027618091410 88\alpha^{108} + 72313626239383964765274946226530432\alpha^{109} \\
&- 41306617347132881440374099326965 12\alpha^{110} + 66777218432831695281436221436556 8\alpha^{111} - 11723059510032803388493956609138 4\alpha^{112} \\
&+ 12047117225922787728443496655488\alpha^{113} - 64331022686518844683148576620 8\alpha^{114} + 18747586287780118903854334848\alpha^{115} - 27904569345819422212536643 2\alpha^{116} \\
&+ 43769992115777655127266 56\alpha^{117} + 12004873192870905025004 8\alpha^{118} + 10731545733669133574528\alpha^{119} - 55415692087819858788 8\alpha^{120} + 442686784318640499 2\alpha^{121} \\
&- 30287462976198976\alpha^{122} - 287791501240448\alpha^{123} - 3696628330464\alpha^{124} + 15077928064\alpha^{125} + 5893184\alpha^{126} + 21888\alpha^{127} - \alpha^{128}
\end{aligned}
$$

A conjectured formula for the degree

Jason Kimberley has observed empirically, based on our computations to date, that the degree $m(d)$ for the polynomial corresponding to $\phi_2(1/d, 1/d)$ appears to be given for odd primes by $m(4k+1) = (2k) \cdot (2k)$ and $m(4k+3) = (2k+2) \cdot (2k+1)$.

If we set $m(2) = 1/2$, for notational convenience, then it seems that for any prime factorization of an integer greater than 2:

$$m\left(\prod_{i=1}^{k} p_i^{e_i}\right) \stackrel{?}{=} 4^{k-1} \prod_{i=1}^{k} p_i^{2(e_i-1)}\, m(p_i).$$

To determine whether or not this formula holds, even larger computer runs (and even higher precision) will be required.

Lessons from the Poisson lattice project

- Finding a fast series for $\phi_2(x, y)$ was a critical step in this research.
- This problem is a good example where the PSLQ computation, not the initial numerical evaluation, is the pacing challenge.
- This problem is also a good example of the need for extremely high precision — 50,000 to 100,000 digits.
- We wanted to run larger examples, but run times were unreasonable (several days), and in one case the run failed due to an obscure bug in the ARPREC package — "very reliable" isn't good enough.
- A faster, highly parallel version of PSLQ is needed for very large problems such as these.

Analysis of Mordell-Tornheim-Witten sums

In three recent papers, the present authors and colleagues (including the late Richard Crandall) explored sums such as

$$\omega(s_1, \ldots, s_{K+1}) \quad := \quad \sum_{m_1, \ldots, m_K > 0} \frac{1}{m_1^{s_1} \cdots m_K^{s_K} (m_1 + \cdots + m_K)^{s_{K+1}}}$$

and, more generally,

$$\begin{aligned} \omega(s_1, \ldots, s_M \mid t_1, \ldots, t_N) \quad &:= \sum_{\substack{m_1, \ldots, m_M, n_1, \ldots, n_N > 0 \\ \sum_{j=1}^M m_j = \sum_{k=1}^N n_k}} \prod_{j=1}^M \frac{1}{m_j^{s_j}} \prod_{k=1}^N \frac{1}{n_k^{t_k}} \\ &= \frac{1}{2\pi} \int_0^{2\pi} \prod_{j=1}^M \mathrm{Li}_{s_j}\left(e^{i\theta}\right) \prod_{k=1}^N \mathrm{Li}_{t_k}\left(e^{-i\theta}\right) \mathrm{d}\theta. \end{aligned}$$

Here the *polylogarithm of order s* is denoted by $\mathrm{Li}_s(z) := \sum_{n \geq 1} z^n / n^s$.

We also explored *character* variants of these sums.

Computation of Mordell-Tornheim-Witten sums

To compute these sums to high precision required high-precision evaluations of:

- A new scheme for computing derivatives of polylogarithms *with respect to the order* for real and complex arguments.
- The ζ function and its derivatives at integer arguments (positive and negative).
- Bernoulli numbers.
- The Γ function and derivatives.
- Definite integrals of complex functions.

and much more.

For full details, see:

- David H. Bailey, Jonathan M. Borwein and Richard E. Crandall, "Computation and theory of extended Mordell-Tornheim-Witten sums," *Mathematics of Computation*, vol. 83, no. 288 (Jul 2014), pg. 1795–1821.
- David H. Bailey and Jonathan M. Borwein, "Computation and theory of extended Mordell-Tornheim-Witten sums II," 05 May 2014, available at `http://www.davidhbailey.com/dhbpapers/mtw2.pdf`.
- David H. Bailey and Jonathan M. Borwein, "Computation and structure of character polylogarithms with applications to Mordell-Tornheim-Witten sums," *Mathematics of Computation*, to appear, 20 Feb 2014, available at `http://www.davidhbailey.com/dhbpapers/mtw3.pdf`.

Summary

- Very high-precision arithmetic is an increasingly important component of modern experimental mathematics and mathematical physics research.
- Precision levels of several hundred digits are commonplace; some problems require as much as 50,000 to 100,000 digits.
- Highly parallel computer implementations are often required for pacing problems.
- Faster and even easier-to-use open-source software is needed for high-precision arithmetic, supporting computations with tens or hundreds of thousands of digits and a wide range of functions.
- Research is needed into faster algorithms for very high-precision evaluation of integrals (especially multi-dimensional integrals).
- Research is needed into new, faster algorithms for very high-precision evaluation of special functions.
- Research is also needed into highly parallel implementations of PSLQ.

This talk is available at `http://www.davidhbailey.com/dhbtalks/dhb-icerm2014.pdf`.