High-Precision Computation: Applications and Challenges

David H. Bailey
http://www.davidhbailey.com
Computational Research Dept., Lawrence Berkeley Natl. Lab.
Computer Science Dept., University of California, Davis

7 April 2013

How much precision is needed?

For many applications, 32-bit IEEE floating-point arithmetic is adequate:

- Most graphics and gaming applications.
- Most applications involving experimental data.
- Most signal processing applications.

However, some definitely require 64-bit IEEE arithmetic:

- Certain graphical calculations (e.g., to determine whether two polygons or curved figures intersect).
- Certain structural problems (elasticity calculations, etc.).

For additional discussion and details:

1. W. Kahan, "A floating-point trick to solve boundary-value problems faster," May 2007, `http://www.eecs.berkeley.edu/~wkahan/Math128/FloTrik.pdf`.
2. W. Kahan, "On the cost of floating-point computation without extra-precise arithmetic," Nov 2004, `http://www.eecs.berkeley.edu/~wkahan/Qdrtcs.pdf`.

Aren't 64 bits enough?

For some other algorithms and applications, even 64-bit IEEE arithmetic isn't sufficient:

- Certain applications with highly ill-conditioned linear systems.
- Large summations, especially those involving cancellations.
- Long-time, iterative simulations.
- Large-scale, highly parallel simulations — numerical sensitivities are magnified.
- Resolving small-scale phenomena.
- Studies in experimental mathematics.

The notion that high-precision arithmetic is not only useful, but is in fact essential for numerous important applications has met significant resistance in the computing community.

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.

Innocuous example where standard precision is inadequate

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:

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

Algorithm changes versus double-double?
Double-double is the pragmatic choice

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—most people use home-grown code.

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:

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

In contrast, a straightforward Linpack scheme, implemented with double-double arithmetic, works fine for this and a wide range of similar problems.

Numerical analysis expertise among U.C. Berkeley grads

Of the 2010 U.C. Berkeley graduating class, 870 were in disciplines expected to use technical computing:

- Division of Mathematical and Physical Sciences (Mathematics, Physics, Statistics).
- College of Chemistry.
- College of Engineering (including Computer Science).
- Other fields with significant computing: biology, geology, medicine and social sciences.

Enrollment in numerical analysis:

- Math 128A (Introductory numerical analysis required of math majors): 219.
- Math 128B (A more advanced course): 24.

Conclusion: Fewer than 2% of Berkeley grads who will do technical computing have had rigorous training in numerical analysis!

Reproducibility in scientific computing

The report of the recent NSF-funded Workshop on Reproducibility in Computational and Experimental Mathematics 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.*

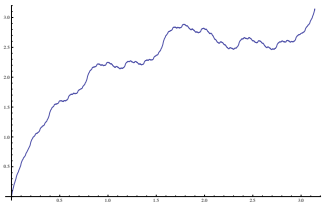
The report concluded that high-precision arithmetic was one of the best ways to ameliorate these numerical difficulties.

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," `http://www.davidhbailey.com/dhbpapers/icerm-report.pdf`.

Enhancing reproducibility with high-precision arithmetic

Problem: Find the arc length of the irregular function $g(x) = x + \sum_{0 \le k \le 10} 2^{-k} \sin(2^k x)$, over the interval $(0, \pi)$ (using 10^7 abscissa points).

- If this computation is done with ordinary double precision arithmetic, the calculation takes 2.59 seconds and yields the result 7.073157029008510.
- If it is done using all double-double arithmetic, it takes 47.39 seconds seconds and yields the result 7.073157029007832.
- But if only the summation is changed to double-double, the result is identical to the double-double result (to 15 digits), yet the computation only takes 3.47 seconds.



Graph of $g(x) = x + \sum_{0 \le k \le 10} 2^{-k} \sin(2^k x)$, over $(0, \pi)$.

Techniques for high-precision computation

- Data is stored as a string of ints or floats, with initial words holding the string length and binary exponent.
- For modest precision levels (< 1000 digits):
  1. Use conventional grade-school schemes for basic arithmetic.
  2. Use conventional Taylor series schemes for transcendentals.
- For extra-high precision levels (> 1000 digits):
  1. Use FFT-based multiplication and Taylor series division.
  2. Use quadratically convergent algorithms for transcendentals.
- Operator overloading (available in Fortran-90, C++ and other languages) can be exploited to construct high-level translation interfaces, which greatly simplify conversion of existing code.

1. D.H. Bailey, Y. Hida, X.S. Li and B. Thompson, "ARPREC: An arbitrary precision computation package," `http://www.davidhbailey.com/dhbpapers/arprec.pdf`.
2. Y. Hida, X.S. Li and D.H. Bailey, "Algorithms for quad-double precision floating point arithmetic," *15th IEEE Symposium on Computer Arithmetic*, IEEE Computer Society, 2001, pg. 155–162.

Free software for high-precision computation

1. ARPREC. Arbitrary precision, with numerous algebraic and transcendental functions. High-level interfaces for C++ and Fortran-90. `http://crd.lbl.gov/~dhbailey/mpdist`.
2. GMP. Produced by a volunteer effort and distributed under the GNU license. `http://gmplib.org`.
3. MPFR. C library for multiple-precision floating-point computations with exact rounding, based on GMP. `http://www.mpfr.org`.
4. MPFR++. High-level C++ interface to MPFR. `http://perso.ens-lyon.fr/nathalie.revol/software.html`.
5. MPFUN90. Similar to ARPREC, but is written entirely in Fortran-90 and provides only a Fortran-90 interface. `http://crd.lbl.gov/~dhbailey/mpdist`.
6. QD. Performs "double-double" (31 digits) and "quad-double" (62 digits) arithmetic. High-level interfaces for C++ and Fortran-90. `http://crd.lbl.gov/~dhbailey/mpdist`.

U.C. Berkeley's CORVETTE project

Developing software tools to find and ameliorate numerical anomalies in large-scale computations:

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

ARITH21 talks on CORVETTE research:

1. Hong Diep Nguyen and Jim Demmel (UC Berkeley, USA), "Numerical Accuracy and Reproducibility at ExaScale."
2. Hong Diep Nguyen and James Demmel (UC Berkeley, USA), "Fast Reproducible Floating-Point Summation."

Some applications where high precision is essential

1. Planetary orbit calculations (32 digits).
2. Supernova simulations (32–64 digits).
3. 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. Detecting "strange" nonchaotic attractors (32 digits).
11. The Taylor algorithm for ODEs (100–600 digits).
12. Ising integrals from mathematical physics (100–1000 digits).
13. Problems in experimental mathematics (100–50,000 digits).

Long-term planetary orbit calculations

Researchers have recognized for centuries that planetary orbits exhibit chaotic behavior:

> *"The orbit of any one planet depends on the combined motions of all the planets, not to mention the actions of all these on each other. To consider simultaneously all these causes of motion and to define these motions by exact laws allowing of convenient calculation exceeds, unless I am mistaken, the forces of the entire human intellect."* *[Isaac Newton, Principia, 1687]*

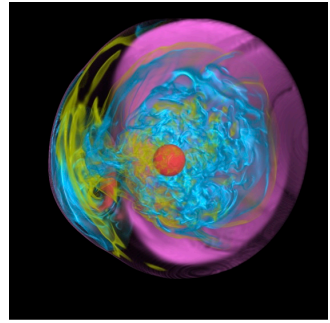
Long-term simulations of planetary orbits using double precision do fairly well for long periods, but then fail at certain key junctures.

Researchers have found that double-double or quad-double arithmetic is required to avoid severe inaccuracies, even if other techniques are employed to reduce numerical error.

G. Lake, T. Quinn and D.C. Richardson, "From Sir Isaac to the Sloan survey: Calculating the structure and chaos due to gravity in the universe," *Proc. of the 8th ACM-SIAM Symp. on Discrete Algorithms*, 1997, pg. 1–10.

Supernova simulations

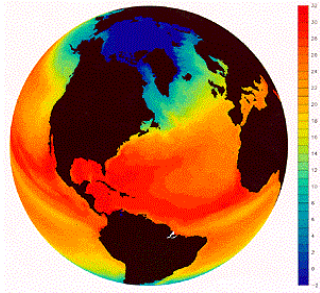
- Researchers at LBNL have used quad-double arithmetic to solve for non-local thermodynamic equilibrium populations of iron and other atoms in the atmospheres of supernovas.
- Iron may exist in several species, so it is necessary to solve for all species simultaneously.
- Since the relative population of any state from the dominant state is proportional to the exponential of the ionization energy, the dynamic range of these values can be very large.
- The quad-double portion now dominates the entire computation.



P.H. Hauschildt and E. Baron, "The numerical solution of the expanding stellar atmosphere problem," *Journal Computational and Applied Mathematics*, vol. 109 (1999), pg. 41–63.

Climate modeling: High-precision for reproducibility

- Climate and weather simulations are fundamentally chaotic: if microscopic changes are made to the current state, soon the future state is quite different.
- In practice, computational results are altered even if minor changes are made to the code or the system.
- This numerical variation is a major nuisance for code maintenance.
- He and Ding found that by using double-double arithmetic in two key inner loops, most of this numerical variation disappeared.



Y. He and C. Ding, "Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications," *Journal of Supercomputing*, vol. 18, no. 3 (Mar 2001), pg. 259–277.

Coulomb n-body atomic system simulations

- Alexei Frolov of Queen's University in Canada has used high-precision arithmetic to solve a generalized eigenvalue problem that arises in Coulomb n-body interactions.
- Matrices are typically 5,000 x 5,000 and are very nearly singular.
- Computations typically involve massive cancellation, and high-precision arithmetic must be employed to obtain numerically reproducible results.
- Frolov has also computed elements of the Hamiltonian matrix and the overlap matrix in four- and five-body systems.
- These computations typically require 120-digit arithmetic.

Frolov: "We can consider and solve the bound state few-body problems ... beyond our imagination even four years ago."

A.M. Frolov and D.H. Bailey, "Highly accurate evaluation of the few-body auxiliary functions and four-body integrals," *Journal of Physics B*, vol. 36, no. 9 (14 May 2003), pg. 1857–1867.

Taylor's method for the solution of ODEs

Taylor's method is one of the oldest numerical schemes for solving ODEs, but in recent years has re-emerged as a leading choice for problems in computational dynamics community. Consider the initial value problem $y' = f(t, y)$. The solution at time $t = t_i$ is:

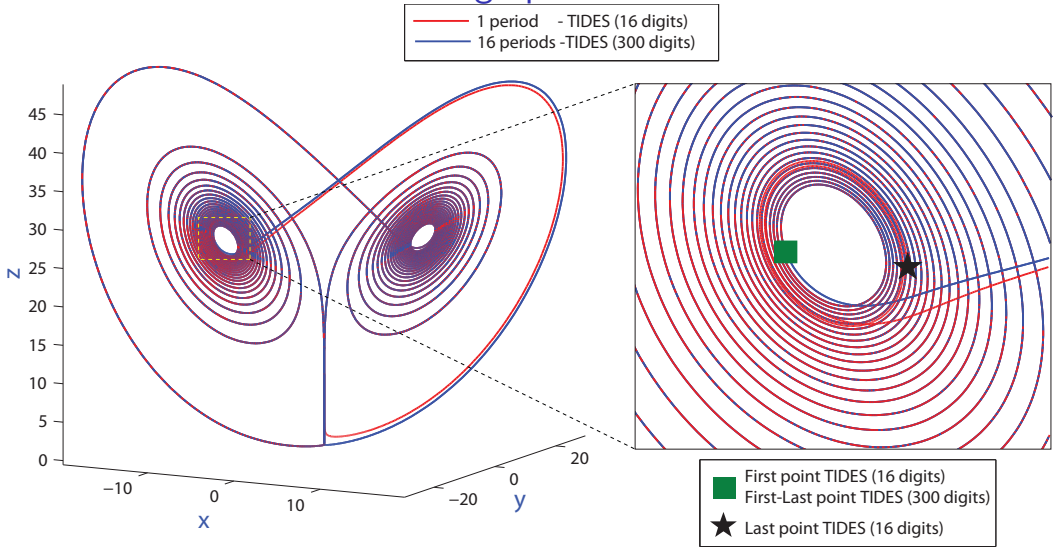
$$\begin{aligned} y(t_0) &=: y_0, \\ y(t_i) &\simeq y_{i-1} + f(t_{i-1}, y_{i-1})\, h_i + \cdots + \frac{1}{n!} \frac{d^{n-1} f(t_{i-1}, y_{i-1})}{dt^{n-1}} h_i^n \\ &=: y_i \end{aligned}$$

The Taylor coefficients here may be found using automatic differentiation methods.

One significant advantage with Taylor's method is that it can be easily implemented using high-precision arithmetic. When this is done, Taylor's method typically gives superior results, compared with other available schemes.

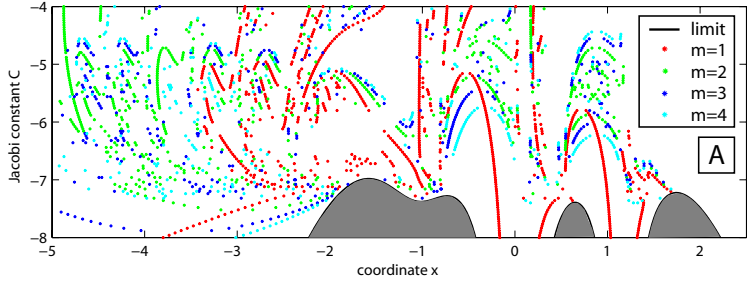
A. Abad, R. Barrio, F. Blesa and M. Rodriguez, "TIDES: a Taylor series Integrator for Differential EquationS," preprint, 2010.

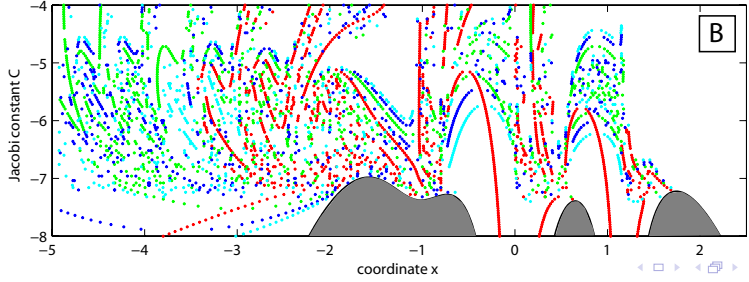
Taylor's method for ODEs with high-precision arithmetic



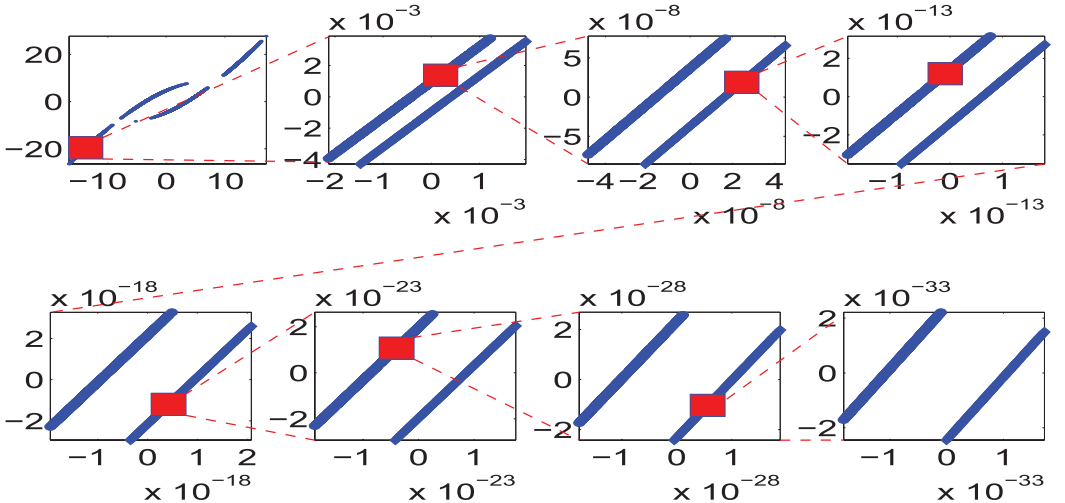
Numerical integration of the L25-R25 unstable periodic orbit for the Lorenz model during 16 time periods using the TIDES code with 300 digits, and 1 time periods using

Computing the "skeleton" of periodic orbits





Fractal properties of Lorenz attractors



On the first plot, the intersection of an arbitrary trajectory on the Lorenz attractor with the section $z = 27$. The plot shows a rectangle in the x-y plane. All later plots zoom in on a tiny region (too small to be seen by the unaided eye) at the center of the

Lions-Mercer iterations

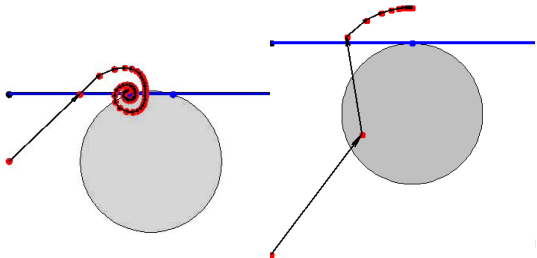
The Lions-Mercer iteration, also known as the Douglas-Rachford or Feinup iteration, is defined by "reflect, reflect and average":

$$x \mapsto T(x) \quad := \quad \frac{x + R_A\left(R_B(x)\right)}{2}$$

In the simple 2-D case of a horizontal line of height α, we obtain the explicit iteration:

$$x_{n+1} \quad := \quad \cos\theta_n, \quad y_{n+1} := y_n + \alpha - \sin\theta_n, \quad (\theta_n := \arg z_n)$$

For $0 < \alpha < 1$, spiraling is ubiquitous. Compare $\alpha = 0.95$ on left with $\alpha = 1.0$ on right:



Exploring iterations using Cinderella

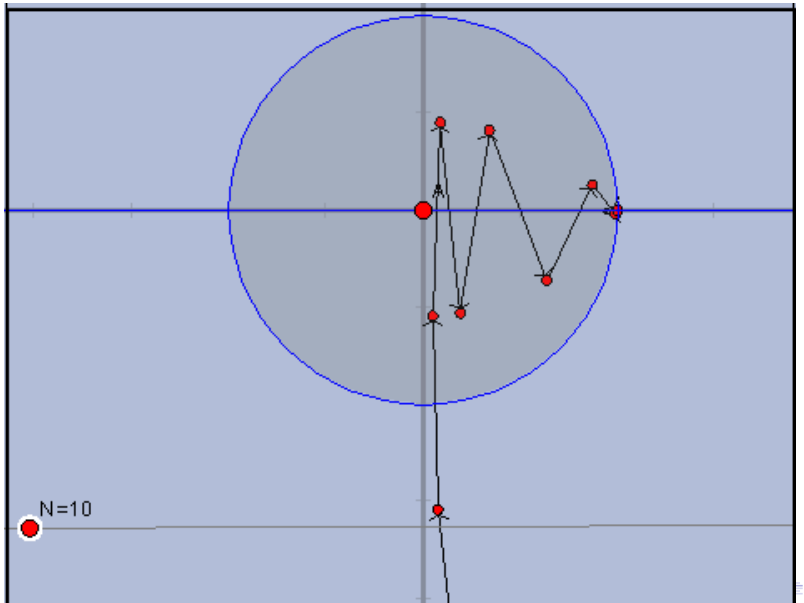
Iterations such as this, as well as many other graphical phenomena, may be explored using the Cinderella online tool: `http://www.cinderella.de`.

Two applets have been defined, working with Cinderella, for exploring Lions-Mercer iterations:
A1. `http://users.cs.dal.ca/?jborwein/reflection.html`
A2. `http://users.cs.dal.ca/?jborwein/expansion.html`

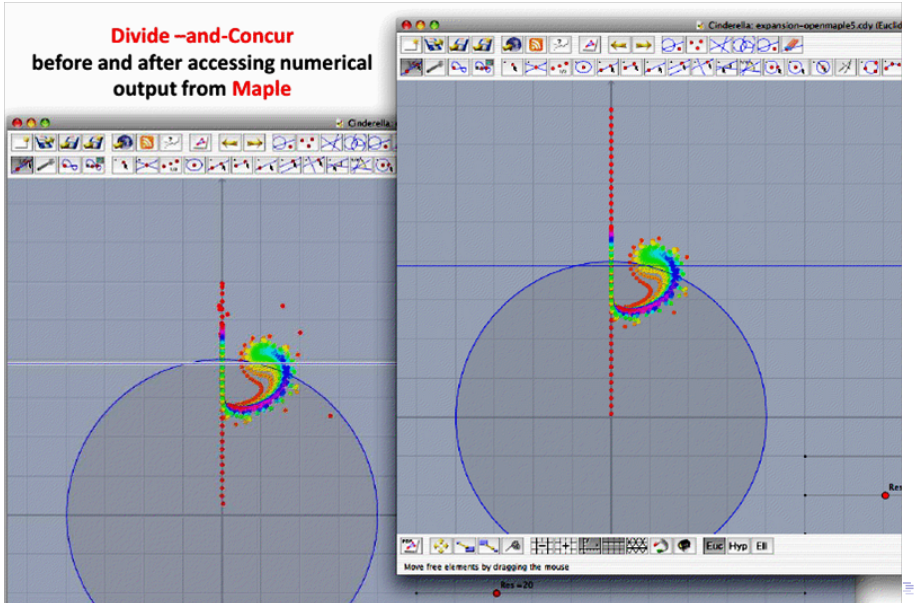
With Applet A1, we observe that as long as the iterate is outside the unit circle, the next point is always closer to the origin; once inside the circle the iterates never leave and converge to $(1, 0)$.

Applet A2 demonstrates clear numerical degradation, compared with similar run using high-precision arithmetic.

Iterations with Applet A1



Iterations with Applet A2: Double vs multiple precision



Experimental mathematics: Discovering new mathematical results by computer

Methodology:

1. Compute various mathematical entities (limits, infinite series sums, definite integrals, etc.) to high precision, typically 100–10,000 digits.
2. Use algorithms such as PSLQ to recognize these numerical values in terms of well-known mathematical constants.
3. When results are found experimentally, seek formal mathematical proofs of the discovered relations.

Many results have recently been found using this methodology, both in pure mathematics and in mathematical physics.

> *"If mathematics describes an objective world just like physics, there is no reason why inductive methods should not be applied in mathematics just the same as in physics." – Kurt Godel*

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.

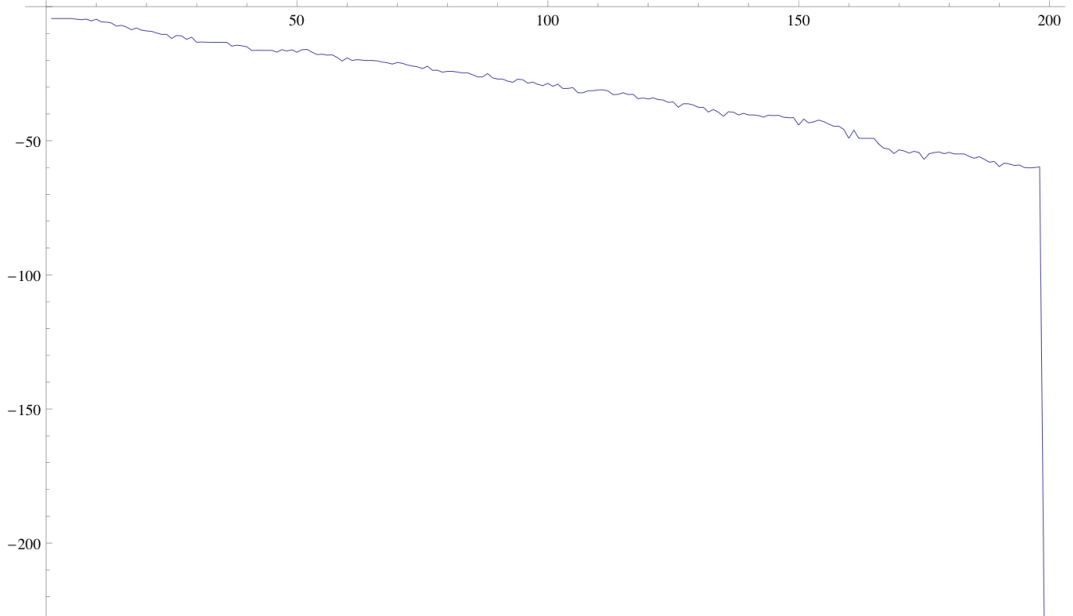
PSLQ, continued

- 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

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

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

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

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.

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.

Ising integrals from mathematical physics

We recently applied our methods to study three classes of integrals (one of which was referred to us by Craig Tracy of U.C. Davis) that arise in the Ising theory of mathematical physics:

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

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

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

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

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

where ζ is the Riemann zeta function and $Li_n(x)$ is the polylog function. D_2, D_3 and D_4 were originally provided to us by Craig Tracy, who hoped that our tools could help identify D_5.

The Ising integral E_5

We were able to reduce E5, which is a 5-D integral, 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. The digits are available if anyone wishes to further explore.

$$
\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(\left(y^2+1\right)z^2+4(y+\right.\right.\right.\\
& 1)z+5\big)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\big)x^3+\left(\left(-3z^2-4z+1\right)y^2-4zy+1\right)x^2-(y(5z+4)\\
& +4)x-1)\big]/\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\big)y+3z^3+5z^2+5z+3\big)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\big)y^3+\left(7z^6-30z^5+28z^4+54z^3+28z^2-30z+7\right)y^2-2\left(7z^5\right.\\
& +15z^4-6z^3-6z^2+15z+7\big)y+7z^4-2z^3-42z^2-2z+7\big)x^4-2y\left(z^3\left(z^3\right.\right.\\
& -9z^2-9z+1\big)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\big)y^4+\left(z^6-14z^5+3z^4+84z^3+3z^2-14z+1\right)y^3-3\left(3z^5\right.\\
& +6z^4-z^3-z^2+6z+3\big)y^2-\left(9z^4+14z^3-14z^2+14z+9\right)y+z^3+7z^2+7z\\
& +1\big)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\big)y^3-2\left(33z^4+2z^3-22z^2+2z+33\right)y^2+\left(6z^3+26\right.\\
& z^2+26z+6\big)y+11z^2+10z+11\big)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.\\
& 22z^3-26z^2-22z+3\big)y^2+\left(3z^3+5z^2+5z+3\right)y+5z^2+22z+5\big)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.\\
& -2z^3-90z^2-2z+15\big)+15\big]/\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.\\
& +4(z+1)y+1)-1)\log(x+1)\big]/\left[(x-1)^3x(y-1)^3(yz-1)^3\right]-\left[4(y+1)(xy\right.\\
& +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.\\
& y^2-4(x+1)\left(z^2-1\right)y+z^2-4z-1\big)\log(xy+1)\big]/\left[x(y-1)^3y(xy-1)^3(z-\right.\\
& 1)^3\big]-\left[4(z+1)(yz+1)\left(x^3y^5z^7+x^2y^4(4x(y+1)+5)z^6-xy^3\left(\left(y^2+\right.\right.\right.\\
& 1)x^2-4(y+1)x-3\big)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.y+1\right)z^2+\left((3x+4)y+4\right)z-1\big)\log(xyz+1)\big]/\left[xy(z-1)^3z(y-1)^3(xyz-1)^3\right]\big)\big]\\
& /\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}
$$

Recursions in Ising integrals

Consider the 2-parameter class of Ising integrals (which arises in quantum field theory for odd k):

$$C_{n,k} = \frac{4}{n!} \int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^{k+1}} \frac{du_1}{u_1} \cdots \frac{du_n}{u_n}$$

After computing 1000-digit numerical values for all n up to 36 and all k up to 75, we discovered (using PSLQ) linear relations in the rows of this array. For example, when $n = 3$:

$$\begin{aligned}
0 &= C_{3,0} - 84C_{3,2} + 216C_{3,4} \\
0 &= 2C_{3,1} - 69C_{3,3} + 135C_{3,5} \\
0 &= C_{3,2} - 24C_{3,4} + 40C_{3,6} \\
0 &= 32C_{3,3} - 630C_{3,5} + 945C_{3,7} \\
0 &= 125C_{3,4} - 2172C_{3,6} + 3024C_{3,8}
\end{aligned}$$

Similar recursions have been found (and proven) for all n.

1. D.H. Bailey, D. Borwein, J.M. Borwein and R.E. Crandall, "Hypergeometric Forms for Ising-Class Integrals," *Experimental Mathematics*, vol. 16 (2007), pg. 257–276.

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.

Elliptic integral functions

Research with "ramble" integrals led us to consider these integrals:

$$I(n_0, n_1, n_2, n_3, n_4) \quad := \quad \int_0^1 x^{n_0} K^{n_1}(x) K'^{n_2}(x) E^{n_3}(x) E'^{n_4}(x) \mathrm{d}x,$$

where K, K', E, E' are elliptic integral functions:

$$\begin{aligned} K(x) &:= \int_0^1 \frac{\mathrm{d}t}{\sqrt{(1-t^2)(1-x^2t^2)}} \\ K'(x) &:= K(\sqrt{1-x^2}) \\ E(x) &:= \int_0^1 \frac{\sqrt{1-x^2t^2}}{\sqrt{1-t^2}}\,\mathrm{d}t \\ E'(x) &:= E(\sqrt{1-x^2}) \end{aligned}$$

1. J. Wan, "Moments of products of elliptic integrals," *Advances in Applied Mathematics*, vol. 48 (2012), pg. 121–141, `http://carma.newcastle.edu.au/jamesw/mkint.pdf`.
2. D.H. Bailey and J.M. Borwein, "Hand-to-hand combat with thousand-digit integrals," *Journal of Computational Science*, vol. 3 (2012), pg. 77–86, `http://www.davidhbailey.com/dhbpapers/combat.pdf`.

Relations found among the I integrals

Thousands of relations have been found among the I integrals. For example, among the class with $n_0 \leq D_1 = 4$ and $n_1 + n_2 + n_3 + n_4 = D_2 = 3$ (a set of 100 integrals), we found that all can be expressed in terms of an integer linear combination of 8 simple integrals. Some examples:

$$\begin{aligned}
81\int_0^1 x^3K^2(x)E(x)\mathrm{d}x &\stackrel{?}{=} -6\int_0^1 K^3(x)\mathrm{d}x - 24\int_0^1 x^2K^3(x)\mathrm{d}x \\
&\quad +51\int_0^1 x^3K^3(x)\mathrm{d}x + 32\int_0^1 x^4K^3(x)\mathrm{d}x \\
-243\int_0^1 x^3K(x)E(x)K'(x)\mathrm{d}x &\stackrel{?}{=} -59\int_0^1 K^3(x)\mathrm{d}x + 468\int_0^1 x^2K^3(x)\mathrm{d}x \\
&\quad +156\int_0^1 x^3K^3(x)\mathrm{d}x - 624\int_0^1 x^4K^3(x)\mathrm{d}x - 135\int_0^1 xK(x)E(x)K'(x)\mathrm{d}x \\
-20736\int_0^1 x^4E^2(x)K'(x)\mathrm{d}x &\stackrel{?}{=} 3901\int_0^1 K^3(x)\mathrm{d}x - 3852\int_0^1 x^2K^3(x)\mathrm{d}x \\
&\quad -1284\int_0^1 x^3K^3(x)\mathrm{d}x + 5136\int x^4K^3(x)\mathrm{d}x - 2592\int_0^1 x^2K^2(x)K'(x)\mathrm{d}x \\
&\quad -972\int_0^1 K(x)E(x)K'(x)\mathrm{d}x - 8316\int_0^1 xK(x)E(x)K'(x)\mathrm{d}x.
\end{aligned}$$

Algebraic numbers in Poisson potential functions associated with 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 *algebraic*, 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 $\alpha = \exp(\pi\phi_2(x,y))$. PSLQ returns the vector of integer coefficients $(a_0, a_1, a_2, \cdots, a_n)$ 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 $\exp(8\pi\phi_2(1/32, 1/32))$

$$
\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 + 10731545733669133574528\alpha^9 \\
&+120048731928709050250048\alpha^{10} + 4376999211577765512726656\alpha^{11} - 279045693458194222125366432\alpha^{12} \\
&+18747586287780118903854334848\alpha^{13} - 643310226865188446831485766208\alpha^{14} \\
&+12047117225922787728443496655488\alpha^{15} - 117230595100328033884939566091384\alpha^{16} \\
&+667772184328316952814362214365568\alpha^{17} - 4130661734713288144037409932696512\alpha^{18} \\
&+72313626239383964765274946226530432\alpha^{19} - 1891420571205861612091802761809141088\alpha^{20} \\
&+38770881730553471470590641060872686464\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} \\
&+12047117225922787728443496655488\alpha^{113} - 643310226865188446831485766208\alpha^{114} \\
&+18747586287780118903854334848\alpha^{115} - 279045693458194222125366432\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}
\end{aligned}
$$

Cautionary example

These constants agree to 42 decimal digits, but are NOT equal:

$$\int_0^\infty \cos(2x) \prod_{n=1}^{\infty} \cos(x/n)\,\mathrm{d}x \;=$$

$$0.392699081698724154807830422909937860524645434187231595926$$

$$\frac{\pi}{8} \;=$$

$$0.392699081698724154807830422909937860524646174921888227621$$

Richard Crandall has shown that this integral is merely the first term of a very rapidly convergent series that converges to $\pi/8$:

$$\frac{\pi}{8} \;=\; \sum_{m=0}^{\infty} \int_0^\infty \cos[2(2m+1)x] \prod_{n=1}^{\infty} \cos(x/n)\,\mathrm{d}x$$

1. D.H. Bailey, J.M. Borwein, V. Kapoor and E. Weisstein, "Ten Problems in Experimental Mathematics," *American Mathematical Monthly*, vol. 113, no. 6 (Jun 2006), pg. 481–409.
2. R.E. Crandall, "Theory of ROOF Walks," 2007, available at `http://people.reed.edu/~crandall/papers/ROOF.pdf`.

Limitations of *Maple* and *Mathematica*

Maple and *Mathematica* are our first choices whenever symbolic or numeric computations are required. However, both have limitations and bugs.

For example, in a study of Mordell-Tornheim-Witten sums (which arise in mathematical physics), we required high-precision numeric values of derivatives with respect to the order s of polylogarithms:

$$\frac{\partial \mathrm{Li}_s(z)}{\partial s}, \quad \text{where} \quad \mathrm{Li}_s(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^s}$$

Maple was not able to numerically evaluate these derivatives at all. *Mathematica*, when asked for 4000 digits, returned only 400 correct digits (at some arguments).

D.H. Bailey, J.M. Borwein and R.E. Crandall, "Computation and theory of extended Mordell-Tornheim-Witten sums," *Mathematics of Computation*, *Ramanujan Journal*, 27 Feb 2013, DOI 10.1007/s11139-012-9427-1, `http://www.davidhbailey.com/dhbpapers/BBC.pdf`.

What is needed for high-precision floating-point software?

- A high-performance, rock-solid-reliable arithmetic engine, with precision scalable to 1,000,000 digits or more.
- A separate package for modest precision (32 and 64 digits)?
- FFT-based multiplication for > 1000 digits.
- A thread-safe design to facilitate multicore parallel processing, and a pathway to extend to graphics processing units (GPUs).
- A comprehensive library of tuned transcendentals: not just sin, cos, exp, etc., but also the gamma function, polylogs (with real and complex arguments), Bessel functions, etc.
- A robust high-level language interface for C++, Fortran-90 and possibly several other languages as well.
- Interfaces for *Maple* and *Mathematica*.

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