Computer discovery and analysis of large Poisson polynomials using 64,000-digit arithmetic

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

Collaborators:
Jonathan M. Borwein (deceased 2 Aug 2016), University of Newcastle, Australia
Richard E. Crandall (deceased 20 Aug 2012), Apple Computers, USA
Jason Kimberley, University of Newcastle, Australia
Watson Ladd, University of California, Berkeley, USA

February 7, 2017

Standing on the shoulders of giants

The following study is multidisciplinary experimental mathematics, as it crucially relies on many highly talented contributions:

- Richard Crandall's work applying the Poisson equation to image enhancement.
- Jon Borwein's insight in how to analyze and rapidly compute the Poisson formula.
- Work by Brent, Zimmermann, Lefevre and other developers of the MPFR package, which was used to perform 64,000-digit arithmetic.
- An enormous software infrastructure behind our computer code:
    - GNU compilers and Apple's Berkeley Unix software, to handle 190,000 lines of code.
    - Fortran custom datatypes and operator overloading, to handle multiprecision code.
    - OpenMP software, to handle 20,000 core-hours of parallel processing.
- Ferguson's PSLQ algorithm (as far as we are aware, this study involves the largest computations ever done using a variant of the PSLQ algorithm).
- A key observation by Jason Kimberley of the University of Newcastle, Australia.
- A concluding proof by Watson Ladd, a graduate student at U.C. Berkeley.

The PSLQ integer relation algorithm

Let $X = (x_k)$ be an $(m+1)$-long real or complex vector. An integer relation algorithm such as PSLQ finds a nontrivial integer vector $A = (a_k)$ such that

$$a_0 x_0 + a_1 x_1 + \cdots + a_m x_m = 0.$$

- The multipair PSLQ algorithm is a more efficient and parallelizable variant of PSLQ, the most widely used integer relation algorithm (other researchers use a variant of the LLL algorithm).
- Integer relation detection (by any algorithm) requires very high precision: at least $(m+1) \cdot \max_k \log_{10} |a_k|$ digits, both in the input data and the algorithm.

- 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.
- D. H. Bailey and D. J. Broadhurst, "Parallel integer relation detection: Techniques and applications," *Mathematics of Computation*, vol. 70, no. 236 (Oct 2000), 1719–1736.

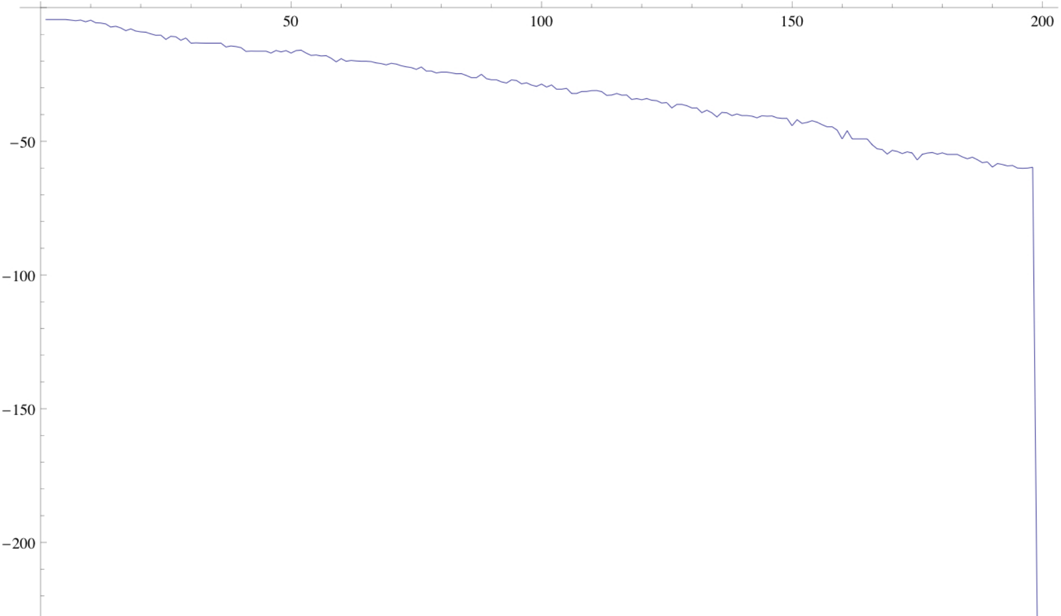
PSLQ, continued

- 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



Application of multipair PSLQ

One simple but important application of multipair PSLQ is to recognize a computed numerical value as the root of an integer polynomial of degree m.

Example: The following constant is suspected to be an algebraic number:

$\alpha = 1.232688913061443445331472869611255647068988824547930576057634684778\ldots$

What is its minimal polynomial?

Method: Compute the vector $(1, \alpha, \alpha^2, \cdots, \alpha^m)$ for $m = 30$, then input this vector to multipair PSLQ.

Answer (using 250-digit arithmetic):

$$
\begin{aligned}
0 = {} & 697 - 1440\alpha - 20520\alpha^2 - 98280\alpha^3 - 102060\alpha^4 - 1458\alpha^5 + 80\alpha^6 - 43920\alpha^7 \\
& + 538380\alpha^8 - 336420\alpha^9 + 1215\alpha^{10} - 80\alpha^{12} - 56160\alpha^{13} - 135540\alpha^{14} - 540\alpha^{15} \\
& + 40\alpha^{18} - 7380\alpha^{19} + 135\alpha^{20} - 10\alpha^{24} - 18\alpha^{25} + \alpha^{30}
\end{aligned}
$$

High-precision 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 $w_j = g'(h_j)$.

Features:

- Reducing h by half typically *doubles* the number of correct digits.
- Works well even for functions with blow-up singularities at endpoints.
- The cost of computing abscissas and weights increases only *linearly* with the number N of subdivisions, which is must faster than with most other methods.

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-Ð741.

Ising integrals: Classic experimental math study led by Jon Borwein

We applied our methods to study three classes of integrals: C_n are connected to quantum field theory, D_n arise in the Ising theory of mathematical physics, while the E_n integrands are derived from 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:

$$C_{1024} = 0.6304735033743867961220401927108789043545870787\ldots$$

What is this number? We copied the first 50 digits into the online Inverse Symbolic Calculator (ISC) at `https://isc.carma.newcastle.edu.au`. 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

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

where ζ is the Riemann zeta function and $\mathrm{Li}_n(x)$ is the polylog function. E_5 remained a "numerical conjecture" for several years, but was proven in March 2014 by Erik Panzer.

E_5 was reduced to a 3-D integral of a very complicated integrand, which was evaluated using tanh-sinh quadrature to 250-digit arithmetic, using over 1000 CPU-hours on a highly parallel system. The PSLQ calculation required only seconds.

The Poisson potential function: Jon Borwein at his best

In 2012, Richard Crandall, while investigating techniques to sharpen images, noted that each pixel was given by a form of the 2-D Poisson potential function:

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

In a 2013 study, Jon Borwein, myself, Crandall and Zucker numerically discovered, and then proved the intriguing fact that for rational (x, y),

$$\phi_2(x, y) = \frac{1}{\pi} \log \alpha$$

where α is *algebraic*, i.e., the root of a some integer polynomial of degree m.

By computing high-precision numerical values of $\phi_2(x, y)$ for various specific rational x and y, and applying a multipair PSLQ program, we were able to produce the explicit minimal polynomials for α in numerous specific cases.

- 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), 115201.

Samples of minimal polynomials found by multipair PSLQ

s	Minimal polynomial corresponding to $x = y = 1/s$:
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	$1 - 216\alpha + 860\alpha^2 - 744\alpha^3 + 454\alpha^4 - 744\alpha^5 + 860\alpha^6 - 216\alpha^7 + \alpha^8$

These computations are very expensive. The case $x = y = 1/32$, for instance, required 10,000-digit arithmetic and ran for 45 hours. Other runs, using even higher precision, ultimately failed, evidently due to subtle program bugs. Help!

Kimberley's formula for the degree of the polynomial

After being introduced to the problem by Jon Borwein, Jason Kimberley of the University of Newcastle observed that the degree $m(s)$ of the minimal polynomial associated with the case $x = y = 1/s$ appears to be given by the following:

Set $m(2) = 1/2$. Otherwise for primes p congruent to 1 mod 4, set $m(p) = \mathrm{int}^2(p/2)$, where int denotes greatest integer, and for primes p congruent to 3 mod 4, set $m(p) = \mathrm{int}\,(p/2)(\mathrm{int}\,(p/2)+1)$. Then for any other positive integer s whose prime factorization is $s = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$,

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

Does Kimberley's formula hold for larger s? Why?

What is the true mathematical connection between the pair of rationals (x, y) and the algebraic number α?

Three improvements to the Poisson polynomial computation program

1. MPFUN2015: A new thread-safe multiprecision package.
   - Speedup: 3X
2. A new 3-level multipair PSLQ program.
   - Speedup: 4.2X
3. Parallel implementation on a 16-core system.
   - Speedup: 12.2X

Overall speedup: 156X

MPFUN2015: A thread-safe arbitrary precision package

DHB has written a package (approx. 50,000 lines of code) for arbitrary precision floating-point computation. It is available in two versions:

- MPFUN-Fort: Completely self-contained, all-Fortran version. Compilation is a simple one-line command, which completes in a few seconds.
- MPFUN-MPFR: Calls the MPFR package for lower-level operations. Installation is significantly more complicated (since GMP and MPFR must first be installed), but performance is roughly 3X faster. We used MPFUN-MPFR in this study.

Both versions include a high-level language interface, using Fortran custom datatypes and operator overloading — for most applications, only a few minor changes to existing double precision code are required.

A C++ version is planned.

Full details of design, algorithms, installation and usage are given in

- D. H. Bailey, "MPFUN2015: A thread-safe arbitrary precision computation package," manuscript, 1 Oct 2015. `http://www.davidhbailey.com/dhbpapers/mpfun2015.pdf`. The software is available at `http://www.davidhbailey.com/dhbsoftware`.

New three-level multipair PSLQ program

Employs three levels of numeric precision:

- Ordinary double precision.
- Medium precision, typically 100–2000 digits.
- Full precision, typically many thousands of digits.

When an entry of the double precision reduced vector is smaller than 10^{-14}, or when an entry of one of the integer-valued double precision arrays exceeds $2^{53} \approx 9.007 \cdot 10^{15}$, the medium precision arrays are updated by matrix multiplication.

Similarly, when an entry of the medium precision reduced vector is smaller than the medium precision "epsilon," the full-precision arrays are updated.

Substantial care must be taken to manage this three-level hierarchy, and to correctly handle numerous atypical scenarios.

A fast algorithm to compute the Poisson potential function $\phi_2(x, y)$

$$\phi_2(x, y) = \frac{1}{2\pi} \log \left| \frac{\theta_2(z, q)\theta_4(z, q)}{\theta_1(z, q)\theta_3(z, q)} \right|,$$

where $q = e^{-\pi}$ and $z = \frac{\pi}{2}(y + ix)$. Compute the four theta functions using the following rapidly convergent formulas from Jon and Peter Borwein's 1987 book *Pi and the AGM*:

$$\theta_1(z, q) = 2 \sum_{k=1}^{\infty} (-1)^{k-1} q^{(2k-1)^2/4} \sin((2k-1)z),$$

$$\theta_2(z, q) = 2 \sum_{k=1}^{\infty} q^{(2k-1)^2/4} \cos((2k-1)z),$$

$$\theta_3(z, q) = 1 + 2 \sum_{k=1}^{\infty} q^{k^2} \cos(2kz),$$

$$\theta_4(z, q) = 1 + 2 \sum_{k=1}^{\infty} (-1)^k q^{k^2} \cos(2kz).$$

High-level computational algorithm

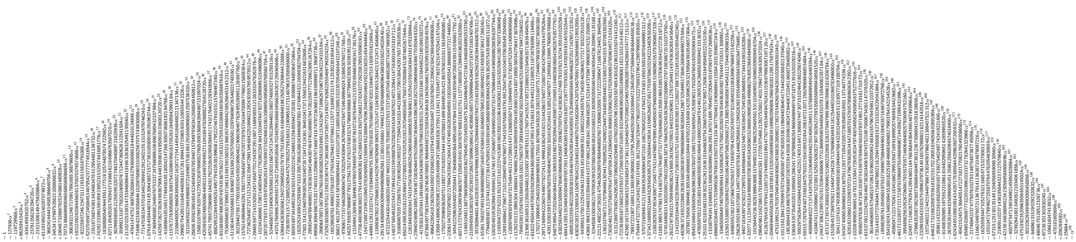
1. Given rationals $x = p/s$ and $y = q/s$, select a conjectured minimal polynomial degree $m(s)$ (using Kimberley's formula) and other parameters for the run.
2. Calculate $\phi_2(x, y)$ to P_2-digit precision using the formulas from two viewgraphs above. When done, calculate $\alpha = \exp(8\pi\phi_2(x, y))$ and generate the $(m + 1)$-long vector $X = (1, \alpha, \alpha^2, \cdots, \alpha^m)$, to P_2-digit precision.
3. Apply the three-level multipair PSLQ algorithm to X. For larger problems, employ a parallel version of the three-level multipair PSLQ code, using the OpenMP construct `DO PARALLEL` to perform certain time-intensive loops in parallel.
4. If no numerically significant relation is found, try again with a larger degree m or higher precision P_2. If a relation is found, employ the polynomial factorization facilities in *Mathematica* and *Maple* to ensure that the polynomial is irreducible.

Application program and libraries for the Poisson calculations

Description	Language	Lines of code
Poisson polynomial program*	Fortran	2,000
MPFUN-MPFR package	Fortran	12,000
MPFR package	C	93,000
GMP package	C	83,000
Total		190,000

*This includes the computation of $\phi_2(x, y)$ and the 3-level multipair PSLQ program.

192-degree minimal polynomial found by multipair PSLQ for $x = y = 1/35$



This polynomial has degree 192, with coefficients as large as 10^{85}. This computation required 18,000-digit arithmetic and 34 CPU-hours.

The case $(1/37, 1/37)$ required 51,000-digit arithmetic and 90 CPU-days (5.6 days on a 16-core parallel system).

Kimberley's formula was upheld for $(1/s, 1/s)$, for all s up to 52 (except for $s = 41, 43, 47, 49, 51$, which were too expensive), and also for $s = 60$ and $s = 64$.

Selected runs (degrees, precision, timings, etc.) for $x = y = 1/s$

s	m	$\log_{10}(D)$	P_1	P_2	N	M	C	T (sec.)	$C \cdot T$ (sec.)
20	32	-463.84	160	700	1967	0.81	1	$2.19 \cdot 10^0$	$2.19 \cdot 10^0$
24	64	-1883.78	320	2200	9297	11.33	1	$7.73 \cdot 10^1$	$7.73 \cdot 10^1$
30	64	-1868.01	350	2300	9064	11.33	1	$1.02 \cdot 10^2$	$1.02 \cdot 10^2$
32	128	-7577.07	650	8200	45893	168.20	1	$5.13 \cdot 10^3$	$5.13 \cdot 10^3$
34	128	-7574.93	650	8200	45914	168.20	1	$5.16 \cdot 10^3$	$5.16 \cdot 10^3$
36	144	-9570.86	750	10300	62282	267.10	1	$9.54 \cdot 10^3$	$9.54 \cdot 10^3$
37	324	-48431.32	1650	51000	931254	6579.66	16	$4.84 \cdot 10^5$	$7.74 \cdot 10^6$
38	180	-14951.64	900	16000	120984	642.98	1	$3.88 \cdot 10^4$	$3.88 \cdot 10^4$
39	288	-38330.14	1450	40000	667153	4124.24	16	$2.68 \cdot 10^5$	$4.29 \cdot 10^6$
40	128	-7580.00	650	8200	45655	168.20	1	$5.02 \cdot 10^3$	$5.02 \cdot 10^3$
42	192	-16993.99	1000	18000	150364	829.41	8	$1.57 \cdot 10^4$	$1.26 \cdot 10^5$
44	240	-26604.14	1200	28000	323762	2003.33	8	$7.43 \cdot 10^4$	$5.94 \cdot 10^5$
45	288	-38315.08	1450	40000	660001	4124.24	16	$2.09 \cdot 10^5$	$3.35 \cdot 10^6$
46	264	-32036.34	1350	34000	476902	2921.57	16	$1.06 \cdot 10^5$	$1.70 \cdot 10^6$
48	256	-30248.55	1350	32000	415316	2586.39	16	$8.98 \cdot 10^4$	$1.44 \cdot 10^6$
50	200	-18421.18	1000	20000	168947	974.44	8	$2.12 \cdot 10^4$	$1.69 \cdot 10^5$
52	288	-38414.49	1550	41000	655291	4124.24	16	$2.12 \cdot 10^5$	$3.40 \cdot 10^6$
*60	256	-14477.99	800	16000	90371	336.41	1	$5.28 \cdot 10^3$	$5.28 \cdot 10^3$
*64	512	-57816.90	1600	64000	802361	5172.79	16	$3.78 \cdot 10^5$	$2.42 \cdot 10^6$

Palindromic polynomials

From our results, in the case $(1/s, 1/s)$ where s is even, the resulting polynomial is always palindromic ($a_k = a_{m-k}$). For instance, when $s = 16$,

$$\begin{aligned} p_{16}(\alpha) = 1 &- 1376\alpha^1 - 12560\alpha^2 - 3550496\alpha^3 + 81241720\alpha^4 - 169589984\alpha^5 \\ &+ 1334964944\alpha^6 - 24307725984\alpha^7 + 238934926108\alpha^8 - 1043027124704\alpha^9 \\ &+ 2328675366384\alpha^{10} - 3219896325280\alpha^{11} + 4238551472456\alpha^{12} \\ &- 10247414430048\alpha^{13} + 28552105805904\alpha^{14} - 55832851687968\alpha^{15} \\ &+ 70020268309062\alpha^{16} \\ &- 55832851687968\alpha^{17} + 28552105805904\alpha^{18} - 10247414430048\alpha^{19} \\ &+ 4238551472456\alpha^{20} - 3219896325280\alpha^{21} + 2328675366384\alpha^{22} \\ &- 1043027124704\alpha^{23} + 238934926108\alpha^{24} - 24307725984\alpha^{25} + 1334964944\alpha^{26} \\ &- 169589984\alpha^{27} + 81241720\alpha^{28} - 3550496\alpha^{29} - 12560\alpha^{30} - 1376\alpha^{31} + \alpha^{32} \end{aligned}$$

Nitya Mani, an undergraduate student at Stanford University, observed that if α is a root of a palindromic polynomial such as this, then $\alpha + 1/\alpha$ is a root of a transformed polynomial of half the degree. This fact can be used to significantly accelerate the computation of Poisson polynomials in the even case (runs denoted by * in the table).

Observations for the case $(1/s, 1/s)$

After doing some Google searches on the coefficients of the polynomials p_{11} and p_{13}, we found the coefficient 387221579866 in p_{11} appears in a 2010 preprint by Savin and Quarfoot, and the coefficient 221753896032 in p_{13} appears in a manuscript, also dated 2010, by Bostan, Boukraa, Hassani, Maillard, Weil, Zenine and Abarenkova.

Savin and Quarfoot define a sequence ψ_n of polynomials in x and y, based on the curve $y^2 = x^3 + x$, as follows:

$$\begin{aligned}\psi_1 &= 1\\ \psi_2 &= 2y\\ \psi_3 &= 3x^4 + 6x^2 - 1\\ \psi_4 &= 2y(2x^6 + 10x^4 - 10x^2 - 2),\end{aligned} \tag{1}$$

and, recursively,

$$\begin{aligned}\psi_{2n+1} &= \psi_{n+2}\cdot\psi_n^3 - \psi_{n-1}\cdot\psi_{n+1}^3 \quad \text{for } n \geq 2\\ \psi_{2n} &= 1/(2y)\cdot\psi_n(\psi_{n+2}\cdot\psi_{n-1}^2 - \psi_{n-2}\cdot\psi_{n+1}^2) \quad \text{for } n \geq 3.\end{aligned}$$

Proofs of Kimberley's formula and the palindromic property

- On March 16, DHB presented our results at a seminar at the University of California, Berkeley.
- Following the presentation, Watson Ladd, a graduate student in mathematics, brought to our attention the fact that some of our conjectures should follow from results in the theory of elliptic curves, Gaussian integers and ideals.
- After some effort, Ladd produced proofs of Kimberley's formula and the palindromic property, which proofs were then incuded in our paper and returned to the journal.
- The paper has now appeared:
  David H. Bailey, Jonathan M. Borwein, Jason Kimberley and Watson Ladd, "Computer discovery and analysis of large Poisson polynomials," *Experimental Mathematics*, 27 Aug 2016.
- A preprint is available here:
  `http://www.davidhbailey.com/dhbpapers/poisson-res.pdf`.

Conclusions

- Jon Borwein, seeing a great opportunity for high-powered experimental mathematics, led a team of researchers over a several-year period to pose, computationally explore and ultimately resolve a very interesting question.
- These computations, which employed up to 64,000-digit precision, producing polynomials with degrees up to 512 and integer coefficients up to 10^{229}, constitute the largest successful integer relation computations to date.
- These efforts ultimately led to a proof of Kimberley's formula and the palindromic property, employing techniques of elliptic curves, Gaussian integers and ideals.
- Additional research is needed to understand many other combinations, e.g., $x = p/s$ and $y = q/s$, for different values of p, q and s, which will require even more sophisticated analysis and extreme computation.

Thanks!

- This talk is available at http://www.davidhbailey.com/dhbtalks/dhb-paris-2017.pdf.
- A preprint with full details, results and analysis is available at http://www.davidhbailey.com/dhbpapers/poisson-res.pdf.