Numerical Reproducibility in High-Performance Computing

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

17 Jun 2013

Reproducibility in scientific computing

A December 2012 workshop on reproducibility in computing, held at Brown University in Rhode Island, USA, noted that

> *Science is built upon the foundations of theory and experiment validated and improved through open, transparent communication. With the increasingly central role of computation in scientific discovery this means communicating all details of the computations needed for others to replicate the experiment. ... The "reproducible research" movement recognizes that traditional scientific research and publication practices now fall short of this ideal, and encourages all those involved in the production of computational science ... to facilitate and practice really reproducible research.*

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`.

Numerical reproducibility

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

The growing problem of numerical reliability

Many applications routinely use either 32-bit or 64-bit IEEE arithmetic, and employ fairly simple algorithms, assuming that all is well. But problems can arise.

Particularly vulnerable are:

1. Large-scale, highly parallel simulations, running on systems with hundreds of thousands or millions of processors — numerical sensitivities are greatly magnified.
2. Certain applications with highly ill-conditioned linear systems.
3. Large summations, especially those involving cancellations.
4. Long-time, iterative simulations (such as molecular dynamics or climate models).
5. Computations to resolve small-scale phenomena.
6. Studies in computational physics or experimental mathematics often require huge precision levels.

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.

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?

Ways to enhance numerical reproducibility and solve accuracy problems

1. Employ an expert numerical analyst to re-examine every algorithm employed in the computation to ensure that only the most stable known schemes are being used.
2. Carefully analyze every step of the computation to ascertain the level of numerical sensitivity at each step.
3. Employ interval arithmetic for large portions of the application (which greatly increases run time and code complexity).
4. Employ higher-precision arithmetic arithmetic, assisted with some “smart” tools to help determine where extra precision is needed and where it is not.

Item #4 is the only practical solution.

Numerical analysis expertise among U.C. Berkeley graduates

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

- Division of Mathematical and Physical Sciences (Math, Physics, Statistics).
- College of Chemistry.
- College of Engineering (including Computer Science).

Other fields (not counted) that will likely involve significant computing:

- Biology, geology, medicine, economics, psychology, sociology.

Enrollment in numerical analysis courses:

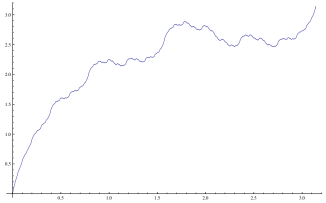
- Math 128A (Introductory numerical analysis required of math majors): 219.
- Math 128B (A more advanced course, required to do serious work): 24.

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

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 (31-digit accuracy), 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 $(0, \pi)$.

Aren't 64 bits enough?

There has been considerable resistance in the scientific computing community to the notion that more than 64-bit arithmetic is not only useful, but may even be essential in some scientific computations. Why?

- Many are persuaded that physical reality fundamentally does not require high precision, beyond, say, the limits of 64-bit IEEE arithmetic.
- Many believe that numerical sensitivity problems are always due to the usage of inferior algorithms.
- Some researchers regard the usage of high-precision arithmetic as "cheating" or "sinful" — because it is too easy?
- Until a few years ago, easy-to-use high-precision arithmetic software facilities were not widely available to experiment with, or the computational cost was too great.

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 64-bit 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:

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

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:

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

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.

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.

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 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," manuscript, May 2013.

Precimonious in action for the arc length computation

Original code:

```
int main() {
  int i, n = 1000000;
  long double h, t1, t2, dppi;
  long double s1;
  ...
  for (i = 1; i <= n; i++) {
    t2 = fun (i * h);
    s1 = s1 + sqrt (h*h + (t1-t2)*(t1-t2));
    t1 = t2;
  }
  return 0;
}
```

Automatically tuned code:

```
int main() {
  int i, n = 1000000;
  double h, t1, t2, dppi;
  long double s1;
  ...
  for (i = 1; i <= n; i++) {
    t2 = fun (i * h);
    s1 = s1 + sqrt (h*h + (t1-t2)*(t1-t2));
    t1 = t2;
  }
  return 0;
}
```

The tuned code produces the same answers, to 15-digit accuracy, yet runs faster.

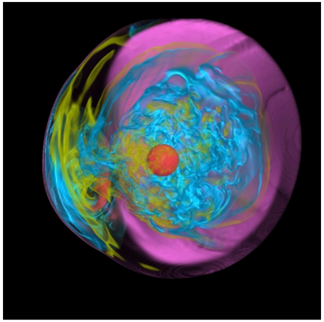
Some applications where high precision is useful or essential

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. 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.

Supernova simulations

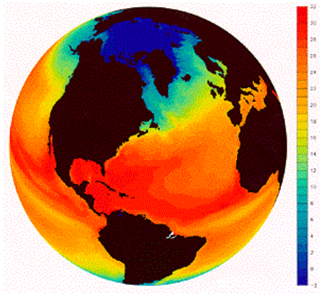
- Researchers at LBNL have used quad-double arithmetic to solve for non-local thermodynamic equilibrium populations of iron and other atoms in the atmospheres of supernovas.
- Iron may exist in several species, so it is necessary to solve for all species simultaneously.
- Since the relative population 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, and cancellations may occur.
- Quad-double arithmetic (62 digits) was employed to reduce these errors.



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.

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

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:

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

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

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

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

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 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 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.

Summary

- Reproducibility in general, and numerical reproducibility in particular, is looming as a major challenge for future exascale computations.
- The most practical solution to such difficulties is to judiciously employ higher-precision arithmetic, combined with some "smart" tools to identify sensitivities and make the requisite code modifications.
- In most cases, only a handful of particularly troublesome loops need to be computed using higher precision — thus, the total run time is only slightly increased.
- While 32-digit or 64-digit arithmetic suffices for almost all conventional scientific computations, a growing body of scientifically interesting computations require hundreds or even thousands of digits.
- Fortunately, such computations are now feasible using modern high-performance computing technology, together with relatively easy-to-use software facilities.

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