Numerical reproducibility in high-performance computing

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

November 19, 2015

Big data science: DANGER AHEAD

Supercomputers operating on big data can generate nonsense faster than ever before!

Key concerns:

- Are the results statistically sound?
- Are the results numerically reliable?
- Have the results been validated using independent rigorous tests?
- Are the algorithms, data sources and processing methods well documented?

Reproducibility crises in physics and cosmology

- In 2011, an international team of researchers at the Gran Sasso Laboratory in Italy announced that neutrinos had exceeded the speed of light, thus directly challenging Einstein's relativity. However, after months of careful checking, a subtle flaw was found in the measurement apparatus (a major embarrassment).
- In 2013, CERN researchers confirmed the discovery of the long-sought Higgs boson. But more recently, scientists have raised questions as to whether the particle discovered is really the Higgs – it might be some other particle or particles masquerading as the Higgs; additional research studies are required.
- In March 2014, researchers announced with considerable fanfare that they had detected the fingerprint of the long-hypothesized inflationary epoch, a tiny fraction after the big bang. Sadly, within a few weeks critics pointed out that their experimental results might well be due to dust in the Milky Way, pending better data.

Reproducibility crises in biomedicine, psychology, economics, finance, autos

- In 2011, Bayer researchers reported that they were able to reproduce only 17 of 67 pharma studies.
- In 2012, Amgen researchers reported that they were able to reproduce only 6 of 53 cancer studies.
- In August 2015, the Reproduciblity Project in Virginia reported that they were able to reproduce only 39 of 100 psychology studies.
- In September 2015, the U.S. Federal Reserve was able to reproduce only 29 of 67 economics studies.
- In 2014-2015, "backtest overfitting" emerged as a major problem in computational finance.

Reproducibility Project staff

Credit: NY Times

- In March 2014, West Virginia researchers reported that they were unable to reproduce Volkswagen's claimed emission figures. This has now exploded into a major international scandal.

ICERM workshop report on numerical reproducibility

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. M. Borwein, R. J. LeVeque, W. Rider and W. Stein, "Setting the default to reproducible: Reproducibility in computational and experimental mathematics," 2012, `http://www.davidhbailey.com/dhbpapers/icerm-report.pdf`.

Numerical reproducibility

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:

- Highly ill-conditioned linear systems.
- Large summations, especially those involving $+/$- terms and cancellations.
- Long-time, iterative simulations (such as molecular dynamics or climate models).
- Computations to resolve small-scale phenomena.
- Studies in computational physics or experimental mathematics often require huge precision levels to produce reliable results.

Large-scale, highly parallel computations greatly magnify numerical sensitivities.

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).

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.

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

Approaches to solve accuracy and numerical reproducibility problems

- 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.
- Carefully analyze every step of the computation to ascertain the level of numerical sensitivity at each step.
- Employ interval arithmetic for large portions of the application (which greatly increases run time and code complexity).
- Employ some “smart” tools to help identify the numerically sensitive parts of an application, then use higher-precision arithmetic arithmetic if needed to fix them.

The last item is the only practical, time-critical 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 rigorous training in numerical analysis.

Selective usage of high-precision arithmetic

High-precision arithmetic is an attractive solution for many present-day reproducibility problems, because it can be implemented on an application with only minor changes to source code.

Free software packages for high-precision computation

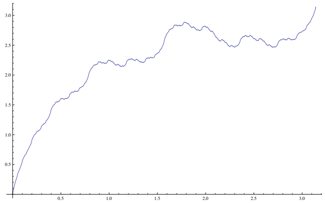
1. QD. Performs "double-double" (31 digits) and "quad-double" (62 digits) arithmetic. High-level interfaces for C++ and Fortran-90. `http://crd-legacy.lbl.gov/~dhbailey/mpdist`.
2. MPFR. C library for arbitrary-precision floating-point computations with exact rounding, based on GMP. `http://www.mpfr.org`.
3. MPFR++. High-level C++ interface to MPFR. `http://www.holoborodko.com/pavel/mpfr/`.
4. MPFUN2015. High-level Fortran interface to MPFR. `http://crd-legacy.lbl.gov/~dhbailey/mpdist`.

High-precision does not guarantee bit-for-bit reproducibility, but it can ensure reproducibility (to a given number of digits) with high probability.

Enhancing reproducibility by selective use of double-double 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 (correct to 15 digits).
- 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 $(0, \pi)$.

U.C. Berkeley's "Precimonious" tool

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

- Test the level of numerical accuracy required for an application.
- Delimit the portions of code that are inaccurate.
- Search the space of possible code modifications.
- Repair numerical difficulties, including usage of high-precision arithmetic.
- 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*, Nov 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.

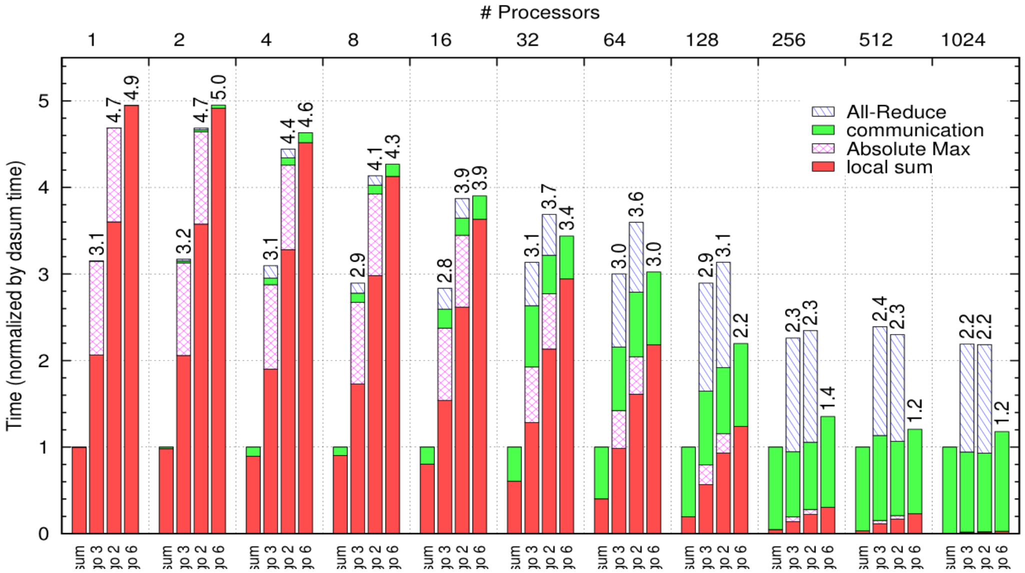
The ReproBLAS project

Researchers who have supported the basic linear algebra subroutine (BLAS) library are developing versions that guarantee reproducibility (to the last bit) for results. Some of these routines are being proposed to be incorporated into the message passing interface (MPI) standard.

Example: A summation routine returns the sum of a vector of 64-bit summands, with the same result regardless of the order of the summands.

For details, see http://bebop.cs.berkeley.edu/reproblas

ReproBLAS summation on 1024-processor Cray XC30 (1M summands)



Bit-for-bit reproducibility: pluses and minuses

Full-application bit-for-bit reproducibilit is nearly feasible at the present time.

Advantages:

- Some applications have legal requirements for bit-for-bit reproducibility.
- Aids parallel debugging and porting to other systems.

Disadvantages:

- Runs several times longer.
- Potentially conflicts with advances in compiler optimization.
- Masks a host of potential numerical accuracy problems, since one never sees the effect of slight perturbations.

Would interval arithmetic be a better solution for those with legal requirements?

We need to discuss this more in the high-performance computing community.

John Gustafson's new "unum" system for floating-point

Gustafson proposes a radical new definition of floating-point arithmetic, based on unums (short for "universal numbers"), a flexible format that contains a "ubit" to designate that the true number lies between two adjacent binary floats.

Example: Avogadro's constant $= 6.022 \times 10^{23}$ is (29 bits total):

sign	exp	frac	ubit	exp size	frac size
0	11001101	111111100001	1	111	1011

Advantages:

- Obeys commutative law, transitive law and other laws of algebra.
- No rounding error, overflow, underflow, negative zero.
- Consistent treatment of positive and negative infinity and NaN.
- Safe to parallelize; portable between systems.
- Often requires fewer bits than conventional IEEE 64-bit arithmetic.

Sample problem (due to Jean-Michel Muller)

Define:

$$
\begin{aligned}
E(z) &= (e^z - 1)/z, \quad E(0) = 1, \\
Q(x) &= |x - \sqrt{x^2+1}| - 1/(x + \sqrt{x^2+1}), \\
H(x) &= E(Q^2(x)).
\end{aligned}
$$

Problem: Compute $H(x)$ for $x = 15, 16, 17, 9999$.

- IEEE 32-bit: $(0, 0, 0, 0)$.
- IEEE 64-bit: $(0, 0, 0, 0)$.
- IEEE 128-bit: $(0, 0, 0, 0)$.
- Unum (correct): $(1, 1, 1, 1)$.

Another sample problem (due to Siegfried Rump)

Define

$$F(x,y) = (333 + 3/4)y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + (5 + 1/2)y^8 + x/(2y)$$

Problem: Compute $F(77617, 33096)$.

- IEEE 32-bit: 1.172603.
- IEEE 64-bit: 1.1726039400531.
- IEEE 128-bit: 1.172603940053178.
- Unum (correct to 23 digits): $-0.827396059946821368141165\ldots$.

Unum: Current status and plans

- Gustafson's book has been published by CRC Press (see below).
- A full implementation is nearly complete in C.
- Much testing and analysis will be done.
- Will require a flexible working precision level.
- Hardware implementation?

For additional details:

- John Gustafson, *The End of Error: Unum Computing*, CRC Press, 2015.

Aren't 64 bits enough?

Some applications actually require more than 64-bit arithmetic:

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.

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:

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

Innocuous example, cont.

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:

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

Lagrange interpolation and Demmel-Koev fail for this problem, but a straightforward Linpack scheme, implemented with double-double arithmetic, works fine.

Summary

- Numerical reproducibility is looming as a major challenge for large-scale scientific computations.
- One practical solution is to apply intellgient tools that judiciously employ higher-precision arithmetic for numerically sensitive portions of the code.
- Numerous other tools are being developed to enhance reproducibility or to ensure bit-for-bit reproducibility.
- Do we want bit-for-bit reproducibility? Will this only mask serious numerical problems? This issue needs to be further discussed in the community.

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