Reproducibility and variable precision computing

David H. Bailey*

September 30, 2019

1 Variable precision computing

For several decades, the IEEE-754 standard for floating-point arithmetic has ably served mathematicians, computer scientists, physicists, engineers and others. Even today, the vast majority of numerical computations in research and engineering employ either IEEE single (32-bit) or IEEE double (64-bit). However, recent developments have demonstrated the need for a broader range of precision levels, and also for a level of precision that varies dynamically within a single application. There are certainly performance advantages to variable precision, including faster processing, better cache utilization, lower run-time memory usage and lower offline data storage. But effective usage of variable precision also requires appropriate software tools, a more sophisticated mathematical framework, as well as some fundamental rethinking what reproducibility means in a variable precision context.

At the low end, many graphics, artificial intelligence and machine learning applications in recent years successfully utilized some form of 16-bit floating-point — usually either the IEEE 16-bit "half" precision standard (five exponent bits and ten mantissa bits) or else the emerging "bfloat16" format (eight exponent bits and seven mantissa bits).

At the same time, many high-performance computing (HPC) researchers, in a drive to achieve exascale computing, are also reconsidering their usage of numerical precision, since as mentioned above there are clear performance advantages to employing reduced precision where possible. This has led to new mixed-precision approaches for common linear algebra operations [1, 2] and renewed interest in iterative refinement, where initial iterations are performed using half- or single-precision [7]. Researchers are exploring the use of floating point compression, not only for I/O, but also for storing solution state variables during run time [9, 5].

Exascale computing has also exposed the need for even greater precision than IEEE 64-bit double in some cases, because greatly enlarged problem sizes of often result in greatly magnified numerical sensitivities, so that one can no longer be certain that results are numerically reliable. One remedy is to use IEEE 128-bit quad precision in selected portions of the computation. Unfortunately, as of this date the major microprocessor and accelerator vendors have not yet implemented the 128-bit IEEE in their hardware systems. It is, however, now available via software in some compilers, notably the gfortran compiler (by declaring quad-precision variables with type `real(16)`). Even though the software implementation is quite slow (up to 50X slower, depending on the operation mix), the IEEE quad format is now being used in a growing number of research applications. As a single example, researchers at Stanford have had remarkable success in using quad precision in computational biology applications that involve heavy-duty multiscale linear programming [11].

*Lawrence Berkeley National Laboratory (retired), Berkeley, CA 94720, and University of California, Davis, Department of Computer Science, Davis, CA 95616 dhbailey@lbl.gov.

There has also been a rise in the usage of very high precision (hundreds or even thousands of digits). For example, numerous new results have been discovered by computing mathematical expressions to very high precision, and then using integer relation algorithms such as the "PSLQ" algorithm to recognize these numerical values in terms of simple mathematical formulas. Among the results that have been discovered in this fashion are new formulas connecting mathematical constants [4] and the elucidation of polynomials connected to the Poisson potential function of mathematical physics (the latter requiring up to 64,000-digit precision) [3].

One common thread of these computational applications is to employ a level of numeric precision that varies dynamically over the course of the computation, performing as much as possible using relatively low precision (16-bit or 32-bit), and only switching to higher precision (64-bit, 128-bit or higher) when necessary.

Along this line, while many in the application community are experimenting with variable precision using present-day system facilities, some are suggesting that we fundamentally rethink the concept of floating-point computation, replacing the current set of fixed precision floating-point formats with a more flexible system. One proposed system is Gustafson's "Unum" system [6]. This and some other alternatives to present-day floating-point formats are summarized in a recent article by Lindstrom, Lloyd and Hittinger [10].

2 Reproducibility issues

It is clear that the emerging paradigm of variable precision computing raises some fundamental questions for reproducibility:

1. What does reproducibility mean with, say, 16-bit floating-point arithmetic? What community-standard metrics can be used, say, for machine learning applications, to certify that the final resulting solution is numerically meaningful?

2. How can reproducibility be measured and verified in more general types of variable precision applications?

3. Can techniques for bit-for-bit reproducibility be incorporated in a variable precision environment? Is there a danger that a bit-for-bit reproducibility facility will lull applications programmers into employing fundamentally unstable algorithms and applications?

4. What new opportunities are there for using iterative refinement with variable precision?

5. Almost all published research in numerical mathematics and numerical analysis has presumed either 32-bit or 64-bit hardware. What new techniques should be considered in a variable precision environment (algorithms, error bounds, error estimates, etc.) [8]?

6. What software tools are available to automatically or semi-automatically analyze an application code, identify numerically sensitive spots and propose or implement remedies [12]? What other tools are needed?

3 Summary

In summary, although the IEEE 754 floating-point standard has served the mathematical, scientific and engineering world very well for over 30 years, we now are seeing rapidly growing demand for

reduced precision (machine learning, neural nets, graphics, etc.), a growing need for mixed 32-64-bit precision, and also a need for greater than 64-bit, all typically varying within a given application.

A workshop discussing these types of issues for variable precision computing has been scheduled for May 6–8, 2020 at the Institute for Computational and Experimental Research in Mathematics (ICERM) in Providence, Rhode Island, USA. For details, see `https://icerm.brown.edu/events/htw-20-vp/`.

References

[1] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek and S. Tomov, "Accelerating scientific computations with mixed precision algorithms", *Comp. Phys. Comm.*, vol. 180 (2009), 2526–2533, `http://dx.doi.org/10.1016/j.cpc.2008.11.005`.

[2] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek and S. Tomov, "Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy," *ACM TOMS*, vol. 34 (2008), art. 17, `http://doi.acm.org/10.1145/1377596.1377597`.

[3] D. H. Bailey, J. M. Borwein, J. Kimberley and W. Ladd, "Computer discovery and analysis of large Poisson polynomials," *Experimental Mathematics*, vol. 26 (2016), 349–363, `https://www.tandfonline.com/doi/full/10.1080/10586458.2016.1180565`.

[4] D. H. Bailey, J. M. Borwein, A. Mattingly and G. Wightwick, "The computation of previously inaccessible digits of π^2 and Catalan's constant," *Notices of the AMS*, (2013) 60(7), 844–854, `http://www.ams.org/notices/201307/rnoti-p844.pdf`.

[5] J. Diffenderfer, A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom, "Error analysis of ZFP compression for floating-point data," *arXiv e-prints*, 2018, `https://arxiv.org/abs/1805.00546`.

[6] J. Gustafson and I. Yonemoto, "Beating floating point at its own game: Posit arithmetic," 2017, `http://www.johngustafson.net/pdfs/BeatingFloatingPoint.pdf`.

[7] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, "Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers," in *Proc. of SC18* (2018), IEEE Press, Piscataway, NJ, USA, Article 47, `https://doi.org/10.1145/3148226.3148237`.

[8] N. J. Higham and T. Mary, "A new approach to probabilistic rounding error analysis," *MIMS Preprint 2018.33*, 12 Nov 2018, `http://eprints.maths.manchester.ac.uk/2673/`.

[9] P. Lindstrom, "Fixed-Rate Compressed Floating-Point Arrays," *IEEE Trans. Vis. Comp. Graphics*, vol. 20 (2014), 2674–2683, Dec 2014, `https://doi.org/10.1109/TVCG.2014.2346458`.

[10] P. Lindstrom, S. Lloyd and J. Hittinger, "Universal coding of the reals: Alternatives to IEEE floating point," in *Proc. Next Gen. Arithmetic (CoNGA '18)*, (2018) ACM, New York, NY, USA, Article 5, `https://doi.org/10.1145/3190339.3190344`.

[11] D. Ma and M. Saunders, "Solving multiscale linear programs using the simplex method in quadruple precision," in M. Al-Baali, L. Grandinetti and A. Purnama, ed., *Recent Developments in Numerical Analysis and Optimization*, Springer, 2017, `http://web.stanford.edu/group/SOL/reports/quadLP3.pdf`.

[12] C. Rubio-Gonzalez, C. Nguyen, H. D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey, C. Iancu and D. Hough, "Precimonious: Tuning assistant for floating-point precision," *Proceedings of SC13*, 26 Apr 2013, `https://www.davidhbailey.com/dhbpapers/precimonious.pdf`.