Comments on “Opportunities and Challenges in 21st Century Experimental Mathematical Computation: ICERM Workshop Report”

Nelson H. F. Beebe
University of Utah
Department of Mathematics, 110 LCB
155 S 1400 E RM 233
Salt Lake City, UT 84112-0090
USA
E-mail: beebe@math.utah.edu
Web: http://www.math.utah.edu/~beebe
Telephone: +1 801 581 5254
FAX: +1 801 581 4148

5 September 2014
Version 2.00

Preface

The workshop report in the title of this document can be found at

- http://www.thecarma.net/jon/ICERM-2014.pdf;
- http://www.davidhbailey.com/dhbpapers/ICERM-2014.pdf
- http://www.math.utah.edu/~beebe/icerm/ICERM-2014.pdf

This document is an almost-verbatim conversion to LaTeX and PDF of plain-text remarks distributed via e-mail to the ICERM report authors. However, to improve readability, I have dressed it up with color, live hyperlinks, and multiple fonts, and fixed my fumble-fingered typos. It is available at http://www.math.utah.edu/~beebe/icerm/ICERM-2014-NOTES.pdf. I have also given it a version number, starting at 2.00, in the expectation that further minor tweaks may be applied in the future.

During the conversion, I added many hyperlinks, and a couple of additional remarks that are set off as indented paragraphs in a sans-serif font, with marginal version numbers.

General Remarks

First, bravo! I agree strongly with most of that report.

I have several nits, however. Here they are, in no particular order.

1. While commercial packages like MATLAB, MAPLE, MATHEMATICA, and S-PLUS are well-designed, and generally well-documented, their use has several drawbacks:

   - They run on only relatively few platform classes (even though class members may be numerous), often just MICROSOFT WINDOWS, APPLE MAC OS X, and selected distributions of GNU/LINUX, all of those now only on a single CPU family — the INTEL x86/x86-64, also implemented by AMD and several other now-historical chip vendors. If your chosen (or only available) platform is, say, an IBM z-SERIES system, or a POWERPC system, or a SPARC system, or an ARM system, or ..., then you are out of luck. The cheapness, and low-power requirements, of ARM processors has made them the CPU of choice in mobile devices, and recently, they can support multiple virtual machines (see `http://www.math.utah.edu/pub/tex/bib/sigarch.html#Dall:2014:KAD`).

     Experts in floating-point arithmetic are keenly aware of the peculiarities of the x86 architecture, where the INTEL 8087 floating-point coprocessor chip was introduced in 1981, *after* a draft of the IEEE binary floating-point standard had been published, but *before* it was finally adopted in 1985. As a result, there are serious issues with double rounding, precision control, and having only a single kind of NaN, instead of both the *quiet* and *signaling* NaNs required by the final standard. However, novices, and even experienced programmers, are often surprised by the floating-point misfeatures in the x86 family, which can operate with 32-bit, 64-bit, or 80-bit formats. Version 2.00.

     The x86-64 extension, introduced by AMD and copied by INTEL, *could* eliminate the double-rounding issues with its new 32-bit and 64-bit registers, but because the x86 instruction set remains accessible, compilers may use a mixture of both sets, making it hard to predict, without examining generated assembly code, exactly how arithmetic is being done, and making it hard to guarantee uniform behavior across compilers, and even compiler optimization levels. Regrettably, the NaN deficiency remains, and as a result, it has contaminated the virtual machines that underlie the popular JAVA and C# programming languages, and all other languages that run on top of the MICROSOFT .NET environment, which further cripples the otherwise outstanding IEEE arithmetic system. Version 2.00.

   - Their source code is closed and inaccessible to inspection.

- They tend to support only what their vendors view as 'popular' platforms (will any still run on an x86 processor a decade from now?), and as a result, computations with them on a current platform are unlikely to be reproducible in the future. Also, a customer who wants to experiment with different hardware or a different operating system is strongly discouraged from doing so, because of the danger of loss of access to those commercial packages. That tends to increase monoculture, which, in both computers and biology, is extremely dangerous for long-term survival.

- Several commercial vendors have recently changed their license-manager practices, so that only the two or three most-recent versions can be run. That regrettable decision far too quickly makes it impossible to reproduce computations made even as recently as two or three years ago. Work is simply not scientific when it is not reproducible. Version 2.00.

- Vendors come and go, and any of them could be swallowed up by a larger company, perhaps with the only purpose being to either monopolize the market, or destroy it. Consider MICROSOFT's destruction of numerous smaller companies (e.g., WORDPERFECT), and ORACLE's acquisition of PEOPLESOFT and SUN MICROSYSTEMS, with threatened destruction of PEOPLESOFT support, and pricing of SUN-developed hardware and software out of the budgets of SUN's former academic, and small business, customers (my department is in that situation, after being a loyal SUN customer for 27 years). Once again, computations may become unreproducible.

- Students in particular cannot afford the high prices (often, thousands of US dollars per machine) once they leave school, even if they enjoyed lower-cost access while still on campus. We therefore have an academic and moral duty, in my strongly-held view, to teach them about free alternatives (more below on that point).

- The AXIOM computer-algebra team is engaged in a complete rewrite of that venerable system, which began as the IBM SCRATCHPAD system more than 40 years ago. They are rewriting Axiom as a literate program on the grounds that that is the *only* way in which the code can be maintained by generations of expert mathematicians, and the *only* way in which non-computer-using mathematicians just *may* be willing to trust results from a computer-algebra system.

2. SAGE is an agglomeration of existing packages, including OCTAVE, GP/PARI, and MAXIMA, using `python` code to attempt to achieve tighter integration between disparate, and radically-different, tools. A successful build of SAGE requires success for each of its dependent packages, and that too, in my extensive experience, means that Sage is almost a monoculture product. That could be fixed, of course, if only we could

convince programmers to strive for the cleanest most-portable code in everything they write. History so far says that is impossible:

- The portability of the GNU `gcc` compiler family went down the flush in 2005 with the 4.x series. The family is widely used, and has backends for at least Ada, C, C++, Fortran, Java, Objective C, and Pascal, so in practice, it supplies much of the compiler technology on many systems today.
- The situation with LLVM + CLANG is far worse (see `http://www.math.utah.edu/pub/llvm/`).
- The OPEN64 compiler project for X86 and X86-64 CPUs is nearly dormant.
- The beautifully small and literately-programmed AT&T/PRINCETON `lcc` compiler, which needs only about 600 lines of code for any architecture-dependent backend code-generator, is effectively dead, stuck at language level C89, with no support for 64-bit integers or other C99 features (see `http://www.math.utah.edu/pub/tex/bib/lcc.html`).
- The fast and streamlined `tcc` compiler for C89 and parts of C99, which runs only on a small number of O/Ses for the X86 platform (and not at all on any of {DRAGONFLY,FREE,KFREE,MIR,NET,OPEN}BSD. or GNU HURD, or SUN/ORACLE SOLARIS), is also in a code freeze after its sole author ran out of development energy.
- Despite hundreds, or even thousands, of computer-science student compiler projects, no other robust, and freely-distributable, C/C++ compiler family has arisen from all of that work.
- Computer scientists seem bent on inventing new programming languages much more rapidly than any user community can ever develop to write in them, much less document them in textbooks.

  Most serious large scientific software projects have components that are decades old; fly-by-night programming languages are useless for implementation of such software.

  Version 2.00.
- Developers of `gcc` and the LINUX kernel seem to be obsessed with extending `gcc` compilers with new syntax, and using it in at least kernel code, and then spattering it all through standard C and C++ header files, making it difficult for other compilers to handle such files without also duplicating `gcc` extensions. For that reason, `lcc` supplies its own standard header files, but often cannot parse other widely-used UNIX header files, such as `<sys/types.h>` and `<sys/time.h>`. We need strict conformance to ISO language standards, and it should be possible to parse any header file named in those standards with a compiler that implements no more syntax than the standards specify.

- No serious free-software developer should ever consider using a programming language for long-term software development when that compiler runs only on a single O/S–CPU platform, or has only a single compiler implementation. Thus, with the exception of `awk`, JAVA, and `sh`, we must reject virtually all scripting languages, including `icon`, `javascript`, `perl`, `php`, `python`, `ruby`, ... [`awk` and `sh` are defined by several IEEE POSIX standards, and there are five or more completely independent implementations of each of them.]

3. **Reference 6** has a run-together word:

```
theontology    ->    the ontology
```

4. **Reference 24** to the `FlySpeck Project` can now be updated to show the successful completion of the project in early August 2014, after the ICERM meeting was held [I'll forward a copy of my posting about that significant event after posting these comments]. See `http://www.math.utah.edu/pub/tex/bib/kepler.html` for publications on the history and solution of the famous *Kepler Conjecture* on sphere packing, which dates back to 1661.

5. "Data" is plural, and "datum" is singular, calling for these changes:

```
data itself is         ->    data themselves are
access to this data    ->    access to these data
```

6. There is a small group of logicians, one of whom recently retired from my University, but is still accessible via e-mail, who are interested in computer proofs of mathematical theorems. A significant problem that they face is that proofs are often done in radically different software systems, so rather than develop $N(N-1)/2$ pairs of language translators, they are developing a common lingua franca that allows proof programs to be translated to and from a core language with just N translators. So far, their work has mostly been with fundamental axioms of logic and mathematics, but with more people involved, it could grow to a much larger proof repository such as is mentioned in the ICERM report. I can probably dig out details from my retired colleague for anyone interested in learning more.

7. As a future small contribution to the family of computer arithmetic packages, earlier this year I designed, and built from scratch, a new multiple-precision integer package, and wrote a 256-page book about it. Before it is released on the Web, there are a few more algorithms that I want to add to it, notably for efficient modular arithmetic, and then I plan to build a flexible multiple-precision floating-point arithmetic package on top of it.

The main goal for me is demonstration of extreme code portability and flexibility: there is no platform-dependent configuration needed for it, and coefficient arrays can be represented with compile-time choices of 8-bit, 16-bit, 18-bit, 32-bit, 36-bit, and 64-bit integers, with and without sanity checking of arguments, giving several different link libraries to choose from, while needing zero changes in user code to switch between those libraries.

Version 2.00.

The reason for offering a choice of coefficient representations, rather than hard-coding the size as all other such packages do, is that computer architectures differ substantially in their efficiency of handling of integers of various sizes, so for numerically-intensive work, it may be beneficial for users to benchmark their own programs with various coefficient sizes, and then choose the fastest representation.

8. **Page 2**, near the middle of the prose, says

```
Note, strikingly, that the aggregate performance of
the 500 systems in the 1994 list was surpassed by
the lowest-ranked machine a decade later!
```

If I look at **Figure 1**, the intersection of vertical axis for year Y with the orange line, it appears to reach the 1 Tflop/s 1994 aggregate only at $Y = 2006$, not 2004. The exponential performance improvement is nevertheless impressive.

Of course, few research mathematicians have access to any of the TOP-500 systems, and most likely have to limit themselves to a multicore desktop or laptop, where the performance gains have been much less impressive. The biggest improvement has been instead in the cost of DRAM (my department now has a 1TB 64-core server, and 8 servers with 128GB or 256GB DRAM on 32 to 128 cores), and storage (1 TB of cheap USB-3 shirt-pocket disk today costs US$75; by contrast, our 600MB washing-machine-sized DEC RP06 cost $20,000 in 1984, which scales to $35M/TB, or a cost reduction factor of 466,000. Even better, when inflation is taken into account, with US$1 (1984) ≡ US$2.29 (2014) [see `http://www.bls.gov/data/inflation_calculator.htm`], the cost reduction is more than 1,000,000 times.).

9. **Page 8**, about 2/3 down:

```
an habit    ->    a habit
```

[I don't support the sloppy American dialect that drops leading *h*'s, talking about *'yoouge yoomans eating erbs'* instead of *'huge humans eating herbs'*]

10. I applaud the ICERM report's urging of the creation of scientific data repositories, but I would go even further. I would **mandate** that both software and data, and documentation thereof, developed under federal grant support be both *deposited* in public copyright-free (or open-source-licensed) replicated repositories, and also be *reviewed* as part of the normal satisfactory-completion process at the end of the grant period.

    We need to get beyond the idea that all that matters for grant completion is a summary report of research publications in peer-reviewed commercial journals. Grant agencies could even forbid publication in commercial journals of results from grant-supported research, and instead, permit only open-source journals with replicated, and worldwide distributed, archives at sites that are expected to have significant longevity, such as venerable academic institutions and national libraries.

11. **Section 3.3, page 9**, lists several software packages. I suggest tossing in a reference to Cowell's 1984 book (see `http://www.math.utah.edu/pub/tex/bib/master.html#Cowell:1984:SDM`), because many of those packages still exist and are in use three decades later.

    - To the first item, I would add at least AXIOM, MACAULAY2, MAGMA, MAXIMA (included in SAGE), MUPAD, and REDUCE. MACSYMA (MAXIMA's grandparent) and REDUCE were the first computer-algebra systems that have survived, the first from MIT, and the second from UTAH. Both are now open source, and both have small development teams. MAXIMA can be built on top of any of 8 different COMMON LISP systems, and REDUCE on at least two COMMON LISPs, and on PSL (PORTABLE STANDARD LISP). I have built both from source many times on many systems.

        Here are Web addresses for some of the mentioned computer-algebra systems: Version 2.00.

        – `http://magma.maths.usyd.edu.au/magma`
        – `http://maxima.sf.net/`
        – `http://pari.math.u-bordeaux.fr`
        – `http://reduce-algebra.sourceforge.net/`
        – `http://www.axiom-developer.org`
        – `http://www.math.uiuc.edu/Macaulay2`
        – `http://www.sagemath.org/`

        Here are corresponding extensive bibliographic resources for most of them: Version 2.00.

        – `http://www.math.utah.edu/pub/tex/bib/axiom.html`
        – `http://www.math.utah.edu/pub/tex/bib/common-lisp.html`
        – `http://www.math.utah.edu/pub/tex/bib/macsyma.html`

  - http://www.math.utah.edu/pub/tex/bib/magma.html
  - http://www.math.utah.edu/pub/tex/bib/maple-extract.html
  - http://www.math.utah.edu/pub/tex/bib/maple-tech.html
  - http://www.math.utah.edu/pub/tex/bib/mathematica.html
  - http://www.math.utah.edu/pub/tex/bib/mathematicaj.html
  - http://www.math.utah.edu/pub/tex/bib/matlab.html
  - http://www.math.utah.edu/pub/tex/bib/mupad.html
  - http://www.math.utah.edu/pub/tex/bib/red-a-f.html
  - http://www.math.utah.edu/pub/tex/bib/red-g-l.html
  - http://www.math.utah.edu/pub/tex/bib/red-m-z.html
  - http://www.math.utah.edu/pub/tex/bib/redbooks.html
  - http://www.math.utah.edu/pub/tex/bib/redextra.html
  - http://www.math.utah.edu/pub/tex/bib/reduce.html

  In each case, changing the suffix from `.html` to `.bib` refers to the BIBTEX version of that file.

- To the second, I would add the commercial, and widely-used, SAS and S-PLUS systems, and possibly also BMDP, NAG, and IMSL (aka PRECISION NUMERICS), and STATSOFT (now owned by DELL).
- There should be a bulleted item for fixed-precision (hardware) floating-point systems, including MATLAB and OCTAVE (included in SAGE) and SIMULINK (for which no free alternative exists). MATLAB could fit in the third point, but it now does so much more, so a separate point is preferred.

12. On **pages 5 and 9**, there is mention of Padé-approximant fits. In practice, however, minimax polynomial fits are generally much better, because they fit over the entire region of approximation, and their fits are even better when rational minimax polynomial approximations are used, despite the need for a single final division. My MATHCW library (see `http://www.math.utah.edu/pub/mathcw`) makes extensive use of such fits, and its forthcoming book describes how to generate them, and discusses the difficulty of computing them for high-precision approximations.

    The extensive experience documented in my book strongly suggests that polynomial fitting is not a fully-solved problem, and further research and software improvements are needed to provide robust and portable algorithms that can be implemented in many different programming languages. Current algorithms require arbitrary-precision arithmetic with digit counts that are often many times higher than that of the final fit. Version 2.00.

    My book also discusses, and sometimes implements in software, continued-fraction algorithms for computation of certain functions. Version 2.00.

It introduces relatively-recently discovered computational recipes that remove most of the premature overflow and underflow problems of traditional continued-fraction work, and reduce the number of divisions to just *one*. The beauty of such algorithms is that they are often fairly compact, rapidly convergent, precision independent, and sometimes have a wider range of convergence of arguments, compared to series expansions. They have largely been ignored in numerical analysis texts of the 20th Century, perhaps because of the long-held mantra that 'division is expensive'. They deserve better treatment, and I look forward to reading the just-appeared book *Neverending Fractions* (ISBN-10 0-521-18649-8, ISBN-13 978-0-521-18649-0), one of whose authors appears on the ICERM report.

13. On **page 13**, in the last bullet, MYSQL appears alone. Although that was long a freely-available database program that grew to industrial strength and worldwide adoption, its acquisition by Oracle has left its status and continued development in significant doubt (I recently could not download the source code anymore, despite having done so many times in the past). The original architect of MYSQL, Michael (aka 'Monty') Widenius, and some members of his MYSQL development team have consequently spun off a new open-source project, MARIADB, that is program-name, and command-name, compatible with MYSQL, but incorporates significant performance improvements. See `https://mariadb.org/`.

    Mention should also be made of POSTGRESQL, which grew out of Michael Stonebraker's INGRES database work at UC/Berkeley, and which, like MYSQL is highly portable and certainly industrial strength. See `http://www.actian.com/products/operational-databases/` and `http://www.postgresql.org/`.

    Thus, INGRES, IBM's DB2, MICROSOFT's SQL SERVER, ORACLE, and SYBASE should also receive mention, with a note that they are all commercial, and complex to setup and manage.

    I run 9 SQL databases here containing the TUG and BIBNET PROJECT bibliography archives (see `http://www.math.utah.edu/pub/tex/bib` and `http://www.math.utah.edu/pub/bibnet`), now with 1.04M entries, and have used all of the above, plus SQLITE3, which is a public-domain, small, and astonishingly-portable database system that can easily handle tens of thousands of records without needing any database administrator, or database server, or access controls, and whose database files are independent of byte-order, CPU, and O/S, and can thus be shared across the entire Internet without change. Thus, SQLITE3 should be mentioned too. It won't handle terabytes of data with satisfactory performance, but it can handle tens to hundreds of megabytes acceptably well. For more on the subject of database choice, see the documentation and papers at `http://www.math.utah.edu/pub/bibsql`.

14. On **page 13**, **third bullet**, I am extremely uneasy with recommendations for use of things like DROPBOX, GOOGLE DOCS, and so on. Yes, they are convenient in the short term, but there is no guarantee that they will be here tomorrow, or next year, or a decade hence. Also, they are free because the data are mined to produce advertising and data-information revenues. Our campus this summer inaugurated a campus-controlled solution at `http://box.utah.edu` that offers partial solutions to my objections.

    I am similarly unhappy with use of social-media sites (FACEBOOK, MYSPACE, TWITTER, ...), blogs, and other forms of electronic communication that do not have mechanisms for long-term archiving and *open access* to content. By contrast, I'm perfectly satisfied with `http://arxiv.org/`, which has now received wide acceptance in the hard sciences and mathematics.

Version 2.00.

15. **Reference 21** has a bogus, but hard-to-spot, trailing digit in an author name:

```
[21] Jean-Baptiste Michel1, \ldots{}
```

Conclusion

Thanks for holding the conference, and writing the report. I would have been interested in attending the ICERM meeting, but it overlapped with another conference on the opposite coast with a group for which I have a 35-year-long devoted association.