Facilitating reproducibility in scientific computing: Principles and practice

David H. Bailey[*] Jonathan M. Borwein[†] Victoria Stodden[‡]

July 13, 2014

Abstract

The foundation of scientific research is theory and experiment, carefully documented in open publications, so that other researchers can reproduce and validate the claimed findings. In most disciplines, there are well-established standards for performing and documenting research to facilitate reproducibility. However, standards are only beginning to evolve in computational research, including scientific and mathematical computation. In published computational work, often there is no record of the workflow process that produced the published computational results, and in some cases, even the code is missing or has been changed significantly since the study was completed. In other cases, the computation is subject to numerical error and variability that makes it difficult to reconstruct the results on a different platform or by a different team of researchers.

That tide may be changing, though, in the wake of some recent workshops and papers that recognize the need for explicit and widely implemented standards, and also an opportunity to do computational research work more effectively. This chapter discusses the roots of the reproducibility problem in scientific computing and summarizes some possible solutions that have been suggested in the community. In particular, it presents some new reporting standards and software tools that can enhance the reproducibility of scientific computing.

1 Introduction

A December 2012 workshop on reproducibility in computing, held at the Institute for Computational and Experimental Research in Mathematics (ICERM) at Brown University, USA, noted that

[*]Lawrence Berkeley National Lab (retired), Berkeley, CA 94720, and University of California, Davis, Dept. of Computer Science, Davis, CA 95616, USA. E-mail: `david@davidhbailey.com`.

[†]CARMA, University of Newcastle NSW 2303, Australia, E-mail: `jon.borwein@gmail.com`.

[‡]Columbia University, Department of Statistics, New York, NY 10027, USA. E-mail: `victoria@stodden.net`.

> 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. [57]

Some of the specific issues identified in the ICERM report and other studies include [57]:

- The need to carefully document the full context of computational experiments—system environment, input data, code used, computed results, etc.
- The need to save the code and data in a permanent repository, with version control and appropriate meta-data.
- The need for reviewers, research institutions and funding agencies to recognize the importance of computing and computing professionals, and to allocate funding for after-the-grant support and repositories.
- The increasing importance of numerical reproducibility, and the need for tools to ensure and enhance numerical reliability.
- The need to encourage publication of negative results—other researchers can often learn from them.
- The re-emergence of the need to ensure responsible reporting of performance.

This paper discusses each of these issues in some detail, and then presents some recommended solutions and methodologies that can enhance reproducibility in the field.

2 Reproducibility in other fields

Before continuing, it is worth noting that the issue of reproducibility and reliability of results has recently come to the fore in numerous other fields of scientific research, and for many of the same reasons — perplexing instances of results that cannot be replicated by other researchers, differences between one experimental run and another, questionable reporting practices and more. It is worth briefly examining some of these experiences, not only to better appreciate the risks of not taking reproducibility seriously, but also to learn some specific principles and methodologies to prevent such problems.

2.1 Reproducibility in physics

Recently the issue of reproducibility has arisen on several occasions in the field of experimental physics. For example, in March 2014 a team of researchers from Harvard University made the dramatic announcement that they had discovered an interesting "twisting" pattern in cosmic microwave background data, measured using their BICEP2 experimental system. This pattern fit very well with the hypothesized pattern of the most commonly assumed model of "inflation" in the first tiny fraction of a second after the big bang, and thus has been trumpeted as the first experimental evidence of the inflationary cosmology. MIT physicist Max Tegmark assessed the discovery in these terms: "I think that if this stays true, it will go down as one of the greatest discoveries in the history of science." [48].

Given the premier billing of this discovery, numerous other research teams began to scrutinize the results, which, in a variance to standard procedure with new scientific developments, had not yet passed peer review. Although the original research team had made its raw data available, other researchers had difficulty reconstructing the claimed results. Finally, one research team from the University of California, Berkeley and another team from Princeton University and New York University posted reports directly challenging the BICEP2 findings, saying that the results could more readily be explained by dust in the Milky Way galaxy. Once the dust effect is subtracted, it is not clear that any signal of inflation remains. At the least, the original team will be required to provide additional data and analysis to remove the cloud of doubt now hanging over the results [23].

2.2 Reproducibility in biomedicine

There have been numerous highly publicized instances of reproducibility problems in the biomedical field, where pharmaceutical products or medical procedures look promising based on initial clinical trials, but later disappoint in real-world implementation. For example, the success rates for new drug development projects in Phase II trials have recently dropped from 28% to 18% [51].

Some specific examples include the following:

- In 2004, GlaxoSmithKline acknowledged that while some trials of Paxil found it effective for depression in children, other unpublished studies showed no benefit [31].
- In 2010, irreproducible analysis was discovered in multiple reports in computational biology published by Duke researcher Anil Potti, leading to retractions and the halting of clinical trials [53, 44].
- In 2011, Bayer researchers reported that they were able to reproduce the results of only 17 of 67 published studies they examined [51].
- In 2012, Amgen researchers reported that they were able to reproduce the results of only 6 of 53 published cancer studies [18].

- In 2014, a review of Tamiflu found that while it made flu symptoms disappear a bit sooner, it did not stop serious complications or keep people out of the hospital [33].

- In January 2014, a pair of studies published in *Nature* described a way to make pluripotent stem cells from ordinary skin or blood cells. However, other researchers subsequently were unable to replicate these results. In the wake of these questions, RIKEN, the Japanese institute where most of the work was conducted investigated, investigated the project and ultimately concluded that there were numerous errors of procedure. For example, they found that the lead researcher had manipulated two images of DNA fragments to render the results better than they really were in reality [1]. Finally, in June 2014, the original authors issued a formal retraction of their results in *Nature* [46].

A common lesson from these experiences is that they have exposed a fundamental flaw in biomedical testing methodology: *Only publicizing the results of successful trials introduces a bias into the results.* The AllTrials movement would require all results to be public: `http://www.alltrials.net`. This movement is targeted at large-scale drug trials, but the principle also applies even to individual runs within a single experiment. Even with full reporting of findings, negative and positive, the recommendations from the ICERM report still apply. Researchers need to access the raw data, the code, the computational pipeline to verify outcomes that were obtained using computational methods.

2.3 Reproducibility in social science

The field of social science (in particular, psychology and anthropology) has undergone a fairly large paradigm shift over the past 20 years or so. Prior to about 1990, many studies in the field presumed a *tabula rasa* ("black slate") approach to human nature, namely the notion that heredity and biology play no significant role in human psychology and that all significant personality and behavioral traits are the result of social environment and family upbringing. This philosophy was expressed by Ashley Montagu, who in 1973 wrote, "Man is man because he has no instincts, because everything he is and has become he has learned, acquired, from his culture, from the man-made part of the environment, from other human beings." [49, pg. 24]. A closely connected paradigm, presumed in anthropological studies dating back to Margaret Mead's writings in the 1920s, was that social and psychological problems seen in modern society were largely absent in pre-modern societies, which were, by comparison, relatively serene and hangup-free [49, pg. 25–26].

The blank slate paradigm began to fall in the 1970s and 1980s, and at the present time has little credibility in peer-reviewed literature, although it is often still seen in the public arena. Instead, the consensus of current researchers is that humans at birth possess remarkably sophisticated facilities for social interaction, language acquisition, pattern recognition, navigation and counting, among other things, and that heredity, evolution and biology are major factors in human personality. Some personality traits are now known to be more than 50% heritable. These results certainly do not imply biological determinism, as upbringing and social environment do matter, but the blank slate paradigm that prevailed for so many years has been decisively refuted.

Similarly, with regards to supposedly carefree pre-modern societies, a more careful generation of anthropologists has found rates of crime, war and social problems exceeding those of the present-day western world, in some cases by large margins [49, pg. 435–439].

So how did the earlier researchers get it so wrong? It appears, from after-the-fact analyses, that the main culprits were sloppy experimental methodology and data analysis (e.g., ignoring or omitting data or results that run counter to predisposition [55]), combined with pervasive wishful thinking and inadequate reporting standards. In addition, it is not standard practice in this field to replicate experiments or to publish the results of replication attempts.

2.4 *Science* and *Nature* add statistical checks to publication and review

Along this line, the journal *Science* announced in January 2014 that it will require greater disclosure of statistical details ("whether there was a pre-experimental plan for data handling (such as how to deal with outliers), whether they conducted a sample size estimation to ensure a sufficient signal-to-noise ratio, whether samples were treated randomly, and whether the experimenter was blind to the conduct of the experiment") and an extra round of statistical checks will be added to its peer-review process [43]. Working in conjunction with the American Statistical Association, the journal has appointed several experts to a statistics board of reviewing editors (SBoRE). A manuscript submitted to the journal will first be examined by the journal's internal editors or external reviewers as required. The SBoRE panel will then identity appropriate external statisticians to review the manuscript [45]. In April 2013, *Nature* introduced a checklist for authors which includes statistical reporting standards, such as "Do the data meet the assumptions of the tests (e.g., normal distribution)?" and "Is there an estimate of variation within each group of data? Is the variance similar between the groups that are being statistically compared?" The checklist also species data and code dissemination standards, such as providing retrieval information for data and the deposition of code in a repository [28].

2.5 Reproducibility in mathematical finance

While the field of mathematical finance might not be as well known as physics, biomedicine, psychology or anthropology, it too has been stung with numerous instances of reproducibility problems. A frequently-encountered difficulty is that proposed investment strategies or funds based on these strategies look great on paper, but fall flat when actually fielded.

These problems are most commonly due to *statistical overfitting of "backtest" (historical market) data*, and arise because while modern computer technology has placed very sophisticated software and large datasets at the disposal of financial analysts, these tools have also made it easier than ever before to read too much into that data. In particular, when a computer can analyze thousands or millions of variations of a given investment strategy, it is almost certain that the optimal strategy, measured by backtests, will be overfit and thus of dubious predictive value.

In two 2014 papers, two of the present authors (Bailey and Borwein, with two other collaborators) derived formulas for (a) relating the number of trials to the minimum backtest length, and (b) computing the probability of backtest overfitting. They also demonstrated that under the assumption of memory in markets, overfit strategies are actually somewhat prone to *lose* money [13, 12].

How easy is it to overfit backtest data? Very! If only 2 years of daily backtest data are available, then no more than 7 strategy variations should be tried. If only 5 years of daily backtest data are available, then no more than 45 strategy variations should be tried (see Figure 1). In general, if a financial analyst or researcher does not report the number of trials N explored when developing an investment strategy, then it is impossible for a third party to properly "reconstruct" the results, in the sense of correctly ascertaining the true level of reward and risk. Indeed, it is shown in [13, 12] that for given any desired performance goal, a financial analyst just needs to keep trying alternative parameters for her strategy, and eventually she will find a variation that achieves the desired goal, yet the resulting strategy will have no statistically valid predictive efficacy whatsoever.

In other words, mathematical finance is joining numerous other fields that employ "big data" in recognizing the need to pay careful attention to the rules of statistical inference when drawing conclusions. The days when sloppy data analysis would suffice are long gone.

3 Why the silence?

As a single example, the material highlighted above on financial mathematics raises the question of why so many in the finance field have remained largely silent with the regards to colleagues who, knowingly or not, employ questionable practices in their work. This includes those who fail to disclose the number of models used in developing an investment strategy, make vague market predictions that do not permit rigorous testing, misuse probability theory and statistics, or employ pseudoscientific technical jargon (e.g., "Fibonacci ratios," "cycles," "waves," "pivot points," etc.), charts and graphs. As we wrote in [13], "Our silence is consent, making us accomplices in these abuses."

Similarly, in several instances when we have observed and highlighted instances of reproducibility problems and sloppy data analysis in the field of scientific computing (see the next three sections), a surprisingly typical reaction is to acknowledge that this is a significant problem in the field, but that it is either futile, unnecessary or inappropriate to make much fuss about it.

So what can be done to improve reproducibility in the scientific computing? These are questions that we will ask, and present some thinking regarding answers, in this study.

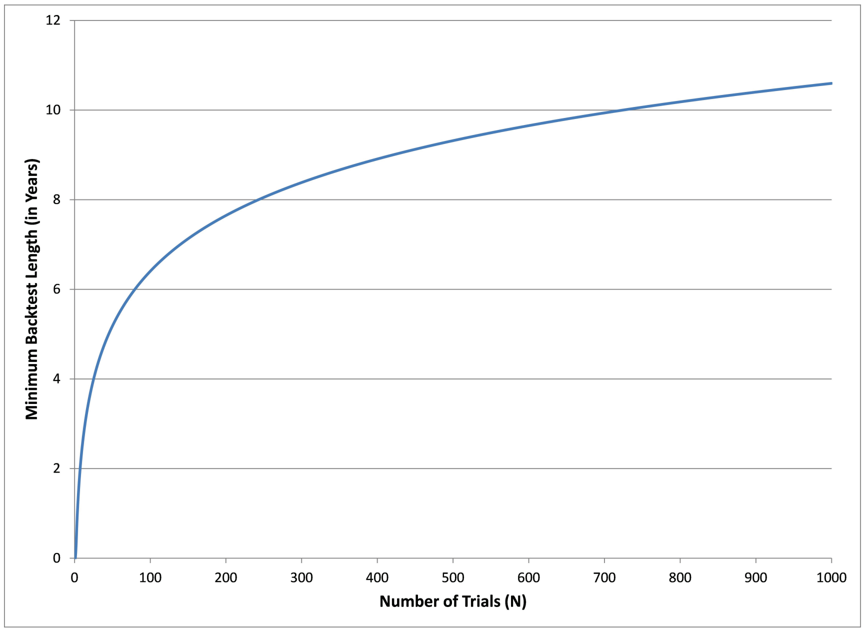

Figure 1: Minimum backtest length versus number of trials

4 Reproducibility in scientific computing

As mentioned above, the emergence of powerful computational hardware, combined with a vast array of computational software, presents unprecedented opportunities for researchers in the field of scientific computing. Indeed, computing is rapidly becoming the backbone of both scientific theory and scientific experiment. Computation is already essential in data analysis, visualization, interpretation and inference. But advanced computer technology also carries with it an increased risk of generating results that do not stand up to rigorous analysis, or cannot reliably be reproduced by independent researchers.

Unfortunately, as we noted in [57], the culture that has developed surrounding scientific computing has evolved in ways that often make it difficult to maintain reproducibility — to verify findings, efficiently build on past research, or even to apply the basic tenets of the scientific method to computational procedures.

Researchers in experimental biology and physics are taught to keep careful lab notebooks documenting all aspects of their procedure. But computational work is often done in a much less careful, transparent, or well-documented manner. Often there is no record of the workflow process used to obtain the published results. As a result, it is not an exaggeration to say that in most published results in the field of scientific computing, reproducing the exact results by an outside team of researchers (or even, after the fact, by the same team!) is often impossible, thus raising the disquieting possibility that numerous published results may be unreliable or even downright invalid [26, 38].

4.1 Documenting the workflow

The first and foremost concern raised in the ICERM report [57] was the need to document the experimental environment and workflow of a computation. Even if all of this information is not included in the published account of the work, a record should be kept while the study is underway, and the information should be preserved. Specific items that were recommended to be noted include the following [57]:

1. A precise statement of the hypothesized assertions to be examined in the study.
2. A statement of the computational approach, and why it constitutes a rigorous test of the hypothesized assertions.
3. Complete statements of, or references to, every algorithm employed.
4. Salient details of auxiliary software (both research and commercial software) used in the computation.
5. Salient details of the test environment, including hardware, system software and the number of processors utilized.
6. Salient details of data reduction and statistical analysis methods.
7. Discussion of the adequacy of parameters such as precision level and grid resolution.
8. Full statement (or at least a valid summary) of experimental results.
9. Verification and validation tests performed by the author(s).
10. Availability of computer code, input data and output data, with some reasonable level of documentation.
11. Curation: Where are code and data available? With what expected persistence and longevity? Is there a site for site for future updates, e.g. a version control repository of the code base?
12. Specific instructions for repeating computational experiments described in the study.
13. The terms of use and licensing should be clearly spelled out. Ideally code and data should "default to open", i.e. a permissive re-use license, unless there are compelling reasons not to (e.g., important privacy or proprietary issues) [56].

14. Avenues of exploration examined throughout development, including information about negative findings.

15. Proper citation of all code and data used, including that generated by the authors.

4.2 Tools to aid in documenting workflow and managing data

A number of the presentations at the ICERM workshop, as summarized in the report [57] described emerging tools to assist researchers in documenting their workflow. Here is a brief summary of some of these tools, condensed and adapted from [57]:

- *Literate programming, authoring, and publishing tools.* These tools enable users to write and publish documents that integrate the text and figures seen in reports with code and data used to generate both text and graphical results. This process is typically not interactive, and requires a separate compilation step. Some examples here are WEB, Sweave, and knitr, as well as programming-language-independent tools such as Dexy, Lepton, and noweb. Other authoring environments include SHARE, Doxygen, Sphinx, CWEB, and the Collage Authoring Environment.

- *Tools that define and execute structured computation and track provenance.* Provenance refers to the tracking of chronology and origin of research objects, such as data, source code, figures, and results. Tools that record provenance of computations include VisTrails, Kepler, Taverna, Sumatra, Pegasus, Galaxy, Workflow4ever, and Madagascar.

- *Integrated tools for version control and collaboration.* Tools that track and manage work as it evolves facilitate reproducibility among a group of collaborators. With the advent of version control systems (e.g., Git, Mercurial, SVN, CVS), it has become easier to track the investigation of new ideas, and collaborative version control sites like Github, Google Code, BitBucket, and Sourceforge enable such ideas to be more easily shared. Furthermore, these web-based systems ease tasks like code review and feature integration, and encourage collaboration.

- *Tools that express computations as notebooks.* These tools represent sequences of commands and calculations as an interactive worksheet. Examples include both closed-source tools such as *MATLAB* (through the publish and app features), *Maple*, and *Mathematica*, as well as open-source tools such as IPython, Sage, RStudio (with knitr), and TeXmacs.

- *Tools that capture and preserve a software environment.* A major challenge in reproducing computations is installing the prerequisite software environment. New tools make it possible to exactly capture the computational environment. For instance, Docker, VirtualBox, VMWare, or Vagrant can be used to construct a virtual machine image containing the environment. Blueprint analyzes the configuration of a machine and outputs its text description. ReproZip captures all the dependencies, files and binaries of the experiment, and also creates a workflow specification for the VisTrails system in order to make the execution and exploration process easier. Application virtualization tools, such as CDE (Code, Data, and Environment), attach themselves to the computational process in order to find and capture software dependencies. Some computational environments can also be constructed and made available in the cloud, and others feature full workflow tracking. Examples include Synapse/clearScience and HUBzero including nanoHUB.

- *Interactive theorem proving systems for verifying mathematics and computation.* "Interactive theorem proving", a method of formal verification, uses computational proof assistants to construct formal axiomatic proofs of mathematical claims. Examples include coq, Mizar, HOL4, HOL Light, ProofPowerHOL, Isabelle, ACL2, Nuprl, Veritas, and PVS. Notable theorems such as the Four Color Theorem have been verified in this way, and Thomas Hales's Flyspeck project, using HOL Light and Isabelle, aims to obtain a formal proof of the Kepler conjecture. Each one of these projects produces machine-readable and exchangeable code that can be integrated in to other programs. For instance, each formula in the web version of NIST's authoritative Digital Library of Mathematical Functions may be downloaded in TeX or MathML (or indeed as a PNG image) and the fragment directly embedded in an article or other code. This dramatically reduces chances of transcription error and other infelicities being introduced.

- *Post-publication tools persistently connecting data, code, workflows, and articles* Although digital scholarly objects may be made available, it is quite possible each object may be hosts by different services and may reside at different locations on the web (such as authors' websites, journal supplemental materials documents, and various data and code repositories). Services such as `http://ResearchCompendia.org`, `http://RunMyCode.org` and `http://ipol.im` attempt to co-locate these objects on the web to enable both reproducibility and the persistent connection of these objects [58].

The development of software tools enabling reproducible research is a new and rapidly growing area of research. We believe that the difficulty of working reproducibly will be significantly reduced as these and other tools continue to be adopted and improved. But prodding the scientific computing community, including researchers, funding agencies, journal editorial boards, lab mangers and promotion committee members, to broadly adopt such tools remains a challenge [41].

5 Performance reporting in high-performance computing

One aspect of reproducibility in scientific computing that unfortunately is coming to the fore once again is the need for researchers to more carefully analyze and report data on the performance of their codes.

5.1 A 1992 perspective

Some background is in order here. One of the present authors (Bailey) was privileged to participate in some of the earliest implementations and analysis of highly parallel scientific computing, back in the late 1980s and early 1990s. In this time period, many new parallel systems had been introduced; each vendor claimed theirs was "best." What's more, many scientific researchers were almost as excited about the potential of highly parallel systems as were the computer vendors themselves. Few standard benchmarks and testing methodologies had been established, so it was hard to reproduce published performance results. Thus much confusion reigned. Overall, the level of rigor and peer review in the field was relatively low.

In response, Bailey published a humorous essay entitled "Twelve ways to fool the masses," which was intended to gently poke some fun at some of these less-than-fully rigorous and responsible practices, hoping that this bit of humor would prod many in the community to tighten their standards and be more careful when analyzing and reporting performance.

The "Twelve ways" article was published in *Supercomputing Review*, which subsequently went defunct. Here is a brief reprise, taken from [6]:

1. Quote 32-bit performance results, not 64-bit results, but don't mention this in paper.
2. Present performance figures for an inner kernel, then represent these figures as the performance of the entire application.
3. Quietly employ assembly code and other low-level language constructs.
4. Scale up the problem size with the number of processors, but omit any mention of this.
5. Quote performance results projected to a full system.
6. Compare your results against scalar, unoptimized code on conventional systems.
7. When run times are compared, compare with an old code on an obsolete system.
8. Base Mflop/s rates on the operation count of the parallel implementation, instead of the best practical serial algorithm.
9. Quote performance as processor utilization, parallel speedups or Mflop/s per dollar.

10. Mutilate the algorithm used in the parallel implementation to match the architecture.

11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.

12. If all else fails, show pretty pictures and animated videos, and don't discuss performance.

Since abuses continued, Bailey subsequently published a paper in which he explicitly called out a number of these practices, citing actual examples from peer-reviewed papers (although the actual authors and titles of these papers were not presented in the paper, out of courtesy) [6]. Here is a brief summary of some of the examples:

- *Scaling performance results to full-sized system.* In some published papers and conference presentations, performance results on small-sized parallel systems were linearly scaled to full-sized systems, without even clearly disclosing this fact. For example, in several cases 8,192-CPU performance results were linearly scaled to 65,536-CPU results, simply by multiplying by eight. Sometimes this fact came to light only in the question-answer period of a technical presentation. A typical rationale was, "We can't afford a full-sized system."

- *Using inefficient algorithms on highly parallel systems.* In other cases, inefficient algorithms were employed for the highly parallel implementation, requiring many more operations, thus producing artificially high Gflop/s rates. For instance, some researchers cited partial differential equation simulation performance based explicit schemes, for applications where implicit schemes were known to be much more efficient. Another paper cited performance for computing a 3-D discrete Fourier transform by direct evaluation of the defining formula, which requires $8n^2$ operations, rather than by using a fast Fourier transform (FFT), which requires $5n \log_2 n$ operations. Obviously, for sufficiently large problems, FFT-based computations can produce desired results with vastly fewer operations.

  This is not to say that alternate algorithms should not be employed, but only that when computing Gflop/s rates, one should base the operation count on the *best practical serial algorithm*, rather than on the actual number of operations performed (or cite both sets of figures).

- *Not actually performing a claimed computation.* One practice that was particularly troubling was that of not actually performing computations that are mentioned in the study. For example, in one paper the authors wrote in the Abstract that their implementation of a certain application on a CM-2 system (a parallel computer available in the 1992 time frame) runs at "300-800 Mflop/s on a full [64K] CM-2, or at the speed of a single processor of a Cray-2 on 1/4 of a CM-2." However, in the actual text of the paper, one reads that "This computation requires 568 iterations (taking 272 seconds) on a 16K Connection Machine." Note that it was actually run on 16,384 processors, not 65,536 processors — the claimed performance figures in the Abstract evidently were merely the 16,384-CPU performance figures multiplied by four. One then reads that "In contrast, a Convex C210 requires 909 seconds to compute this example. Experience indicates that for a wide range of problems, a C210 is about 1/4 the speed of a single processor Cray-2." In other words, the authors did not actually perform the computation on a Cray-2, as clearly suggested in the Abstract; instead they ran the computation on a Convex C210 and employed a rather questionable conversion factor to obtain the Cray-2 figure.

- *Questionable performance plots.* The graphical representation of performance data also left much to be desired in many cases. Figure 2 shows a plot from one study that compares run times of various sizes of a certain computation on a parallel systems, in the lower curve, versus the same computations on a vector system (a widely used scientific computer system at the time), in the upper curve. The raw data for this graph is as follows:

Problem size (x axis)	Parallel system run time	Vector system run time
20	8:18	0:16
40	9:11	0:26
80	11:59	0:57
160	15:07	2:11
990	21:32	19:00
9600	31:36	3:11:50*

In the text of the paper where this plot appears, one reads that in the last entry, the 3:11:50 figure is an "estimate" — this size problem was not actually run on the vector system. The authors also acknowledged that the code for the vector system was not optimized for that system. Note however, by examining the raw data, that the parallel system is actually slower than the vector system for all cases, except for the last (estimated) entry. Also, except for the last entry, all real data in the graph is in the lower left corner — the design of the plot leaves much to be desired; a log-log plot should have been used for such data.

5.2 Fast forward to 2014: New ways to fool the masses

Perhaps given that a full generation has passed since the earlier era mentioned above, present-day researchers are not as fully aware of the potential pitfalls of performance reporting. In any event, several researchers in the high-performance computing field have noted a "resurrection" of some of these questionable practices.

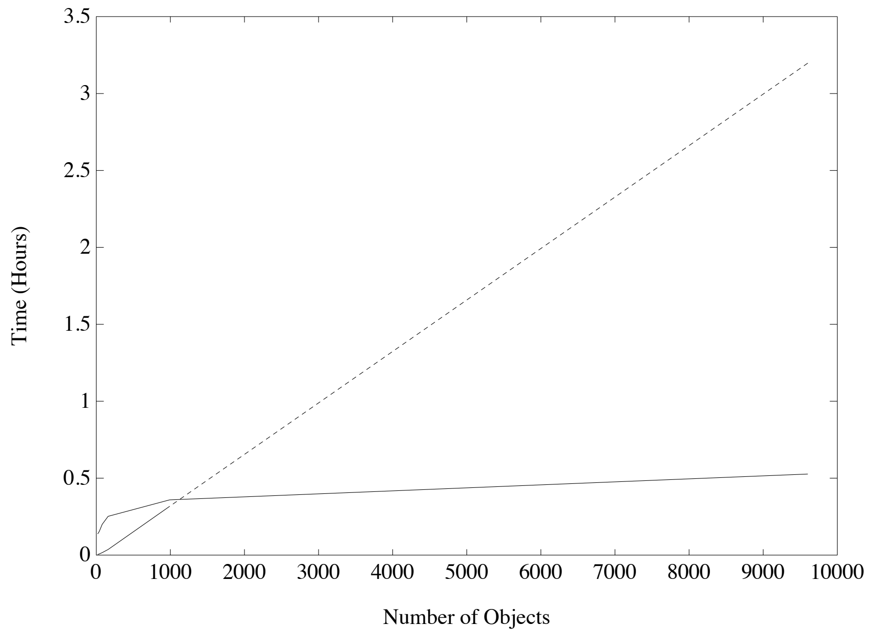

Figure 2: Performance plot [parallel (lower) vs vector (upper)

Even more importantly, the advent of many new computer architectures and designs has resulted in a number of "new" practices unique to these systems. Here are some of the areas that are of the most concern:

- Citing performance rates for a run with only one processor core active in a shared-memory multi-core node, producing artificially inflated performance (since there is no shared memory interference) and wasting resources (since most cores are idle). For example, some studies have cited performance on "1024 cores" of a highly parallel computer system, even though the code was run on 1024 multicore nodes (each node containing 16 cores), using only one core per node, and with 15 out of 16 cores idle on each node.

- Claiming that since one is using a graphics processing unit (GPU) system, that efficient parallel algorithms must be discarded in favor of more basic algorithms. Again, while some algorithms may indeed need to be changed for GPU systems, it is important that when reporting performance, one base the operation count on the *best practical serial algorithm.*

- Citing performance rates only for a core algorithm (such as FFT or linear system solution), even though full-scale applications have been run on the system.

- Running a code numerous times, but only publishing the best performance figure in the paper (recall the experience of pharmaceutical tests in Section 2).
- Basing published performance results on runs using special hardware, operating system or compiler settings that are not appropriate for real-world production usage.
- Defining "scalability" as successful execution on a large number of CPUs, regardless of performance.

In each of these cases, note that in addition to issues of professional ethics and rigor, these practices result make it essentially impossible for other researchers to reconstruct claimed performance results, or to build on past research experiences to move the field forward. After all, if one set of researchers report that a given algorithm is the best known approach for performing a given application on a certain computer system, other researchers may also employ this approach, but then wonder why their performance results seem significantly less than expected.

Reproducible practices are more than just the "right" thing to do; they are essential to making real progress in the field.

6 Numerical reproducibility

The ICERM report also emphasized rapidly growing challenge of 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. [57]

6.1 Floating-point arithmetic

Difficulties with numerical reproducibility have their roots in the inescapable realities of floating-point arithmetic. "Floating-point arithmetic" means the common way of representing non-integer quantities on the computer, by means of a binary sign and exponent, followed by a binary mantissa — the binary equivalent of scientific notation, e.g., 3.14159×10^{13}. Such representations are approximate, and computer arithmetic operations involving floating-point numbers produce similarly approximate results. Thus roundoff error is unavoidable in floating-point computation.

For many scientific and engineering computations, particularly those involving empirical data, IEEE 32-bit floating-point arithmetic (roughly 7-digit accuracy) is sufficiently accurate to produce reliable, reproducible results, as required by the nature of the computation. For more demanding applications, IEEE 64-bit arithmetic (roughly 15-digit accuracy) is required. Both formats are supported on almost all scientific computer systems.

Unfortunately, with the greatly expanded scale of computation now possible on the enormously powerful, highly parallel computer systems now being deployed, even IEEE 64-bit arithmetic is sometimes insufficient. This is because numerical difficulties and sensitivities that are minor and inconsequential on a small-scale calculation may be major and highly consequential once the computation is scaled up to petascale or exascale size. In such computations, numerical roundoff error may accumulate to the point that the course of the computation is altered (i.e., by taking the wrong branch in a conditional branch instruction), or else the final results are no longer reproducible across different computer systems and implementations, thus rendering the computations either useless or misleading for the science under investigation.

6.2 Numerical reproducibility problems in real applications

As a single example, the ATLAS (acronym for "A Toroidal LHC Apparatus") experiment at the Large Hadron Collider must track charged particles to within 10-micron accuracy over a 10-meter run, and, with very high reliability, correctly identify them. The software portion of the experiment consists of over five million lines code (C++ and Python), which has been developed over many years by literally thousands of researchers.

Recently, some ATLAS researchers reported to us that in an attempt to improve performance of the code, they changed the underlying math library. When this was done, they found some collisions were missed and others were misidentified. That such a minor code alteration (which should only affect the lowest-order bits produced by transcendental function evaluations) had such a large high-level effect suggests that their code has significant numerical sensitivities, and results may even be invalid in certain cases. How can they possibly track down where these sensitivities occur, much less correct them?

As another example of this sort, researchers working with an atmospheric simulation code had been annoyed by the difficulty of reproducing benchmark results. Even when their code was merely ported from one system to another, or run with a different number of processors, the computed data typically diverged from a benchmark run after just a few days of simulated time. As a result, it was very difficult even for the developers of this code to ensure that bugs were not introduced into the code when changes are made, and even more problematic for other researchers to reproduce their results. Some divergence is to be expected for computations of this sort, since atmospheric simulations are well-known to be fundamentally "chaotic," but did it really need to be so bad?

After an in-depth analysis of this code, researchers He and Ding found that merely by employing double-double arithmetic (roughly 31-digit arithmetic) in two key summation loops, almost all of this numerical variability disappeared. With this minor change, benchmark results could be accurately reproduced for a significantly longer time, with very little increase in run time [36].

Researchers working with some large computational physics applications reported similar difficulties with reproducibility. As with the atmospheric model code, they were able to dramatically increase the reproducibility of their results by employing a form of custom high-precision arithmetic in several critical summation loops [52].

6.3 High-precision arithmetic and numerical reproducibility

As noted above, using high-precision arithmetic (i.e., higher than the standard IEEE 64-bit arithmetic) is often quite useful in ameliorating numerical difficulties and enhancing reproducibility.

By far the most common form of extra-precision arithmetic is known as "double-double" arithmetic (approximately 31-digit accuracy). This datatype consists of a pair of 64-bit IEEE floats (s, t), where s is the 64-bit floating-point value closest to the desired value, and t is the difference (positive or negative) between the true value and s. "Quad-double" arithmetic operates on strings of four IEEE 64-bit floats, providing roughly 62-digit accuracy. Software to support these datatypes is widely available, e.g., the QD package [37].

Extensions of such schemes can be used to achieve any desired level of precision. Software to perform such computations has been available for quite some time, for example in the commercial packages *Mathematica* and *Maple*. However, until 10 or 15 years ago those with applications written in more conventional languages, such as C++ or Fortran-90, often found it necessary to rewrite their codes. Nowadays there are several freely available high-precision software packages, together with accompanying high-level language interfaces, utilizing operator overloading, that make code conversions relatively painless. A current list of such software packages is given in [8].

Recently a team led by James Demmel of U. C. Berkeley have begun developing software facilities to find and ameliorate numerical anomalies in large-scale computations. These include facilities to: 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; and even navigate through a hierarchy of precision levels (32-bit, 64-bit or higher) as needed. The current version of this tool is known as "Precimonious." Details are presented in [54].

Obviously there is an extra cost for performing high-precision arithmetic, but in many cases high-precision arithmetic is often only needed for a small portion of code, so that the total run time may increase only moderately.

6.4 Computations that require extra precision

While it may come as a surprise to some, there is a growing body of important scientific computations that actually *require* high-precision arithmetic to obtain numerically reliable, reproducible results. Typically such computations involve highly ill-conditioned linear systems, large summations, long-time simulations, highly parallel simulations, high-resolution computations, or experimental mathematics computations.

The following example, condensed from [7], shows how the need for high-precision arithmetic may arise even in very innocent-looking settings. Suppose one suspects that the data $(1, 32771, 262217, 885493, 2101313, 4111751, 7124761)$ are given by an integer polynomial for integer arguments $(0, 1, \ldots, 6)$. Most scientists and engineers will employ a familiar least-squares scheme to recover this polynomial [50, pg. 44].

Performing these computations with IEEE 64-bit arithmetic and the LINPACK routines [2] for LU decomposition, with final results rounded to the nearest integer, finds the correct coefficients $(1, 0, 0, 32769, 0, 0, 1)$ (i.e., $f(x) = 1 + (2^{15} + 1)x^3 + x^6$), but it fails when given the 9-long sequence generated by the function $f(x) = 1 + (2^{20} + 1)x^4 + x^8$. On a MacPro system, for example, the resulting rounded coefficients are $(1, 6, -16, 14, 1048570, 2, 0, 0, 1)$, which differ badly from the correct coefficients $(1, 0, 0, 0, 1048577, 0, 0, 0, 1)$.

From a numerical analysis point of view, a better approach is to employ a Lagrange interpolation scheme or the Demmel-Koev algorithm [25]. An implementation of either scheme with 64-bit IEEE-754 arithmetic finds the correct polynomial in the degree-8 case, but even these schemes both fail in the degree-12 case. By contrast, merely by modifying a simple LINPACK program to employ double-double arithmetic, using the QD software [37], all three problems (degrees 6, 8 and 12) are correctly solved without incident. See [7] for additional details.

There are numerous full-fledged scientific applications that also require higher precision to obtain numerically reliable, reproducible results. Here is a very brief summary of some applications that are known to the present authors. See [7] and [8] for additional details:

1. A computer-assisted solution of Smale's 14th problem (18-digit arithmetic) [59].

2. Obtaining accurate eigenvalues for a class of anharmonic oscillators (18-digit arithmetic) [42].

3. Long-term simulations of the stability of the solar system [40, 30] (18-digit and 31-digit arithmetic).

4. Studies of n-body Coulomb atomic systems (120-digit arithmetic) [32].

5. Computing solutions to the Schrodinger equation (from quantum theory) for the lithium atom[62] and, using some of the same machinery, to compute a more accurate numerical value of the fine structure constant of physics (approx. $7.2973525698 \times 10^{-3}$) (31-digit arithmetic) [63].

6. Computing scattering amplitudes of collisions at the Large Hadron Collider (31-digit arithmetic) [29, 19, 47, 24].

7. Studies of dynamical systems using the Taylor method (up to 500-digit arithmetic) [15, 16, 17].

8. Studies of periodic orbits in dynamical systems using the Lindstedt-Poincaré technique of perturbation theory, together with Newton's method for solving nonlinear systems and Fourier interpolation (1000-digit arithmetic) [60, 61, 5].

7 High-precision arithmetic in experimental mathematics and mathematical physics

Very high-precision floating-point arithmetic is essential to obtain reproducible results in experimental mathematics and some related mathematical physics applications [20, 21].

Many of these computations involve variants of Ferguson's PSLQ integer relation detection algorithm [14]. Given an n-long vector (x_i) of floating-point numbers, the PSLQ algorithm finds the integer coefficients (a_i), not all zero, such that $a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$ (to available precision), or else determines that there is no such relation within a certain bound on the size of the coefficients. Integer relation detection typically requires very high precision, both in the input data and in the operation of the algorithm, to obtain numerically reproducible results.

One of the earliest applications of PSLQ was to numerically discover what is now known as the "BBP" formula for π:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right).$$

This remarkable formula, after a simple manipulation, can be used to calculate binary or base-16 digits of π beginning at the n-th digit, without needing to calculate any of the first $n-1$ digits [20, pp. 135–143].

Very high-precision computations, combined with variants of the PSLQ algorithm, have been remarkably effective in mathematical physics settings. We summarize here just one or two examples of this methodology in action, following [7] and some related references as shown below.

In one study, the ARPREC high-precision software was employed to study the following classes of integrals [10]. The C_n are connected to quantum field theory, the D_n integrals arise in the Ising theory of mathematical physics, while the E_n integrands are derived from

D_n:

$$C_n = \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$D_n = \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{\prod_{i<j}\left(\frac{u_i-u_j}{u_i+u_j}\right)^2}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$E_n = 2\int_0^1 \cdots \int_0^1 \left(\prod_{1\le j<k\le n} \frac{u_k - u_j}{u_k + u_j}\right)^2 \mathrm{d}t_2\, \mathrm{d}t_3 \cdots dt_n.$$

In the last line $u_k = \prod_{i=1}^k t_i$.

In general, it is very difficult to compute high-precision numerical values of n-dimensional integrals such as these. But the C_n integrals can be written as one-dimensional integrals as follows:

$$C_n = \frac{2^n}{n!}\int_0^\infty pK_0^n(p)\,\mathrm{d}p,$$

where K_0 is the *modified Bessel function* [3]. For large n, these numerical values approach the limit

$$\lim_{n\to\infty} C_n = 0.630473503374386796122040192710\ldots.$$

This numerical value was quickly identified, using the *Inverse Symbolic Calculator 2.0* (available at `http://carma-lx1.newcastle.edu.au:8087`), as

$$\lim_{n\to\infty} C_n = 2e^{-2\gamma},$$

where γ is Euler's constant. This identity was then proven [10]. Some results were also obtained for the D_n and E_n integrals, although the numerical calculations involved there were much more expensive, requiring highly parallel computation [10, 9].

One application of very high-precision arithmetic (thousands of digits) arose out of attempts to solve the Poisson equation, which, as our colleague Richard Crandall (deceased December 2012) has shown, arises in various contexts such as the analysis of crystal structures and even the sharpening of photographic images. The following lattice sums arise as values of basis functions in the Poisson solutions [11]:

$$\phi_n(r_1,\ldots,r_n) \quad = \quad \frac{1}{\pi^2} \sum_{m_1,\ldots,m_n \text{ odd}} \frac{e^{i\pi(m_1 r_1 + \cdots + m_n r_n)}}{m_1^2 + \cdots + m_n^2}.$$

By noting some striking connections with Jacobi ϑ-function values, Crandall, Zucker, and the present authors were are able to develop new closed forms for certain values of the arguments r_k [11].

After extensive high-precision numerical experimentation, we discovered, then were able to prove, the remarkable fact that for rational numbers x and y,

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

where A is an *algebraic number*, namely the root of an algebraic equation with integer coefficients.

In this case we computed $\alpha = A^8 = \exp(8\pi\phi_2(x,y))$ (as it turned out, the '8' substantially reduces the degree of polynomials and so computational cost), and then generated the vector $(1, \alpha, \alpha^2, \cdots, \alpha^d)$, which, for various conjectured values of d, was input to the multipair PSLQ program. When successful, the PSLQ program returned the vector of integer coefficients $(a_0, a_1, a_2, \cdots, a_d)$ of a polynomial satisfied by α as output. With some experimentation on the degree d, and after symbolic verification using *Mathematica*, we were able to ensure that the resulting polynomial is in fact the minimal polynomial satisfied by α. Here are some examples of the minimal polynomials discovered by this process [11]:

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	$1 - 216\alpha + 860\alpha^2 - 744\alpha^3 + 454\alpha^4 - 744\alpha^5$ $+860\alpha^6 - 216\alpha^7 + \alpha^8$

Using this data, we were able to conjecture a formula that gives the degree d as a function of k [11].

These computations required prodigiously high precision: up to 20,000-digit floating-point arithmetic in some cases, such as in finding the degree-128 polynomial satisfied by

$\alpha = \exp(8\pi\phi_2(1/32, 1/32))$. Related computations in a follow-up study required up to 50,000-digit arithmetic.

In each of these examples, the usage of very high-precision arithmetic is not optional, but absolutely essential to the computation — the relations in question could not possibly be recovered in a reliable, reproducible fashion except by using these exalted levels of precision. Future research in this field will focus not only on how to perform such computations in the most efficient way possible, but also how to explore and then certify that the precision level used in such computations is sufficient to obtain reproducible results.

8 Reproducibility in symbolic computing

Closely related to the high-precision computations described in the past two sections is the usage of symbolic computing in experimental mathematics and mathematical physics. At the present time, the commercial packages *Maple* and *Mathematica* are most commonly used for symbolic computations, although other packages, such as *Sage* and *Magma* are also used by some.

But like all other classes of computer software, symbolic manipulation software has bugs and limitations. In some cases, the software itself detects that it has encountered difficulties, and outputs messages acknowledging that its results might not be reliable. But not always.

For example, consider the integral

$$W_2 = \int_0^1 \int_0^1 \left| e^{2\pi i x} + e^{2\pi i y} \right| \mathrm{d}x \, \mathrm{d}y, \tag{2}$$

which arose in research into the properties of n-step random walks on the plane [22, 39]. The latest editions of *Maple* (version 18) and *Mathematica* (version 9) both declare that $W_2 = 0$, in spite of the obvious fact that this integral is positive and nonzero. Indeed $W_2 = 4/\pi$ is the expected distance traveled by a uniform random walk in two steps. It is worth pointing out that (2) can easily be rewritten as

$$W2 = \int_0^1 |e^{2\pi i x}| \left(\int_0^1 |1 + e^{2\pi i (y-x)}| \mathrm{dy} \right) \mathrm{dx} \;=\; \int_0^1 |1 + e^{2\pi i y}| \, \mathrm{dy}, \tag{3}$$

which is in a form that both *Maple* and *Mathematica* can evaluate, but this "human" observation evidently is not noted by the commercial packages.

As another example, three researchers in Spain encountered difficulties with *Mathematica* while attempting to check a number theory conjecture. (This problem seems to have been resolved in *Mathematica 10*.) They ultimately found errors that they could exhibit even in fairly modest-sized examples. For example, they presented an example involving a 14×14 matrix of pseudorandom integers between -99 and 99, which was then multiplied by a certain diagonal matrix with large entries, and then added to another matrix of pseudorandom integers between -999 and 999. When they then attempted to compute the determinant of the resulting fixed matrix using *Mathematica*, the results are not consistent — they often obtain different answers for the same problem! They have not yet determined the exact source of this difficulty [27].[1]

Symbolic computing applications often are quite expensive, with jobs running for many hours, days or months. For example, recently three researchers attempted to update an earlier effort to explore Giuga's conjecture, which is that an integer $n \geq 2$ is prime if and only if

$$\sum_{k=1}^{n-1} k^{n-1} \equiv -1 \pmod 6. \tag{4}$$

They computationally verified this conjecture for all composite n with up to 4771 prime factors, which means that any counterexample must have at least 19,907 digits. This computation required 95 hours.

9 Why should we trust the results of computation?

These examples raise the question of why should anyone trust the results of any computation, numeric or symbolic. After all, there are numerous potential sources of error, including user programming bugs, symbolic software bugs, compiler bugs, hardware errors, and I/O errors. None of these can be categorically ruled out. However, substantial confidence can be gained in a result by reproducing it with different software systems, hardware systems, programming systems, or by repeating the computation with detailed checks of intermediate results.

For example, programmers of large computations of digits of π have for many years certified their results by repeating the computation using a completely different algorithm. If all but a few trailing digits agree, then this is impressive evidence that both computations are almost certainly correct. In a similar vein, programmers of large, highly parallel numerical simulations, after being warned by systems administrators that occasional system hardware or software errors are inevitable, are inserting internal checks into their codes. For example, in problems where the computation carries forward both a large matrix and its inverse, programmers have inserted periodic checks that the product of these two matrices is indeed the identity, to within acceptable numerical error.

In other cases, researchers are resorting to formal methods, wherein results are verified by software that performs a computer verification of every logical inference in the proof, back to the fundamental axioms of mathematics. For example, University of Pittsburgh researcher Thomas Hales recently completed a computer-aided proof of the Kepler conjecture, namely the assertion that the most efficient way to pack equal-sized spheres is the

[1]We need to obtain permission from the authors for this example.

same scheme used by grocers to stack oranges [34]. Some objected to this approach, so Hales is now attempt to re-do the proof using formal methods. This project is now 80% complete [35].

9.1 Is 'Free' Software Better?

We conclude by commenting on open-source versus commercial software. For example, *GeoGebra*, which is based on *Cabri*, is now very popular in schools as replacement for *Sketchpad*. But the question is whether such software will be preserved when the core group of founders and developers lose interest or move on to other projects or employment. This is also an issue with large-scale commercial products, but it is more pronounced with open-source project.

One advantage of *Maple* over *Mathematica* is that most of the *Maple* source code is accessible, while *Mathematica* is entirely sealed. As a result, it is often difficult to track down or rectify problems that arise. Similarly, *Cinderella* is very robust, unlike *GeoGebra*, and mathematically sophisticated—using Riemann surfaces to ensure that complicated constructions do not crash. That said, it is the product of two talented and committed mathematicians but only two, and it is only slightly commercial. In general, software vendors and open source producers do not provide the teacher support that has is assumed in the textbook market.

One other word of warning is given by the recent experience of the Heartbleed bug [4], which many cybersecurity observers termed "catastrophic." This is a bug in the OpenSSL cryptography library, which is incorporated in many other software packages, commercial and open-source. A fixed version of the OpenSSL was quickly released, but it will take time to identify all instances.

10 Conclusions

The advent of new extra-high performance computer systems, typically involving tens of thousands, hundreds of thousands or even millions of processing elements, has opened a new vista to the field of scientific computing and computer modeling. It has revolutionized numerous fields, from climate modeling and materials science to protein biology and nuclear physics. In the coming years, this technology will be exploited in numerous industrial and engineering applications as well, such as for virtual product testing and market analysis.

However, like numerous other fields that are embracing "big data," scientific computing must confront issues of reliability and reproducibility, both in computed results and also in auxiliary analysis such as performance measurements. For one thing, these computations are becoming very expensive, both in terms of system acquisition costs but even more so in human time to develop the requisite programs.

Thus it is increasingly essential that reliability and reproducibility be foremost in these activities: designing the workflow process right from the start for reliability and reproducibility; carefully documenting each step of code preparation; data preparation; execution; post-execution processing and analysis and publication of the results. A little effort spent early on will be rewarded with much less confusion and much more productivity at the end.

References

[1] Let the light shine in. *Economist*. Available at http://www.economist.com/news/science-and-technology/21604089-two-big-recent-scientific-results-are-looking-shakyand-it-open-peer-review.

[2] Linpack. Available at http://www.netlib.org/linpack.

[3] Nist digital library of mathematical functions. Available at http://dlmf.nist.gov.

[4] Heartbleed. 2014. Available at http://en.wikipedia.org/wiki/Heartbleed.

[5] A. Abad, R. Barrio, and A. Dena. Computing periodic orbits with arbitrary precision. *Phys. Rev. E*, 84:016701, 2011.

[6] D. H. Bailey. Misleading performance reporting in the supercomputing field. *Scientific Programming*, 1:141–151, 1992.

[7] D. H. Bailey, R. Barrio, and J. M. Borwein. High-precision computation: Mathematical physics and dynamics. *Appl. Math. and Computation*, 218:10106–10121, 2012.

[8] D. H. Bailey and J. M. Borwein. High-precision arithmetic: Progress and challenges. Available at http://www.davidhbailey.com/dhbpapers/hp-arith.pdf.

[9] D. H. Bailey and J. M. Borwein. Hand-to-hand combat with thousand-digit integrals. *J. of Computational Science*, 3:77–86, 2012.

[10] D. H. Bailey, J. M. Borwein, and R. E. Crandall. Integrals of the ising class. *J. Physics A: Math. and Gen.*, 39:12271–12302, 2006.

[11] D. H. Bailey, J. M. Borwein, R. E. Crandall, and J. Zucker. Lattice sums arising from the poisson equation. *J. Physics A: Math. and Theor.*, 46:115201, 2013.

[12] D. H. Bailey, J. M. Borwein, M. L. de Prado, and Q. J. Zhu. The probability of backtest overfitting. Available at http://ssrn.com/abstract=2326253.

[13] D. H. Bailey, J. M. Borwein, M. L. de Prado, and Q. J. Zhu. Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, pages 458–471, May 2014.

[14] D. H. Bailey and D. Broadhurst. Parallel integer relation detection: Techniques and applications. *Math. of Computation*, 70:1719–1736, 2000.

[15] R. Barrio. Performance of the Taylor series method for ODEs/DAEs. *Appl. Math. Comput.*, 163:525–545, 2005.

[16] R. Barrio. Sensitivity analysis of odes/daes using the taylor series method. *SIAM J. Sci. Computing*, 27:1929–1947, 2006.

[17] R. Barrio, F. Blesa, and M. Lara. VSVO formulation of the Taylor method for the numerical solution of ODEs. *Comput. Math. Appl.*, 50:93–111, 2005.

[18] C. G. Begley and L. M. Ellis. Drug development: Raise standards for preclinical cancer research. *Nature*, 483:531–533.

[19] C. F. Berger, Z. Bern, L. J. Dixon, F. F. Cordero, D. Forde, H. Ita, D. A. Kosower, and D. Maitre. An automated implementation of on-shell methods for one-loop amplitudes. *Phys. Rev. D*, 78, 2008.

[20] J. M. Borwein and D. H. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century.* A.K. Peters, Natick, MA, second edition, 2008.

[21] J. M. Borwein, D. H. Bailey, and R. Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery.* A.K. Peters, Natick, MA, 2004.

[22] J. M. Borwein and A. Straub. Mahler measures, short walks and logsine integrals. *Theoretical Computer Science. Special issue on Symbolic and Numeric Computation*, 479:4–21, 2013. DOI: http://link.springer.com/article/10.1016/j.tcs.2012.10.025.

[23] R. Cowen. Doubt grows about gravitational waves detection. *Scientific American.*

[24] M. Czakon. Tops from light quarks: Full mass dependence at two-loops in qcd. *Phys. Lett. B*, 664:307, 2008.

[25] J. Demmel and P. Koev. The accurate and efficient solution of a totally positive generalized vandermonde linear system. *SIAM J. of Matrix Analysis Appl.*, 27:145–152, 2005.

[26] D. Donoho, A. Maleki, M. Shahram, V. Stodden, and I. U. Rahman. Reproducible research in computational harmonic analysis. *Computing in Science and Engineering*, 11, January 2009.

[27] A. J. Duran, M. Perez, and J. L. Varona. Misfortunes of a mathematicians' trio using computer algebra systems: Can we trust? 2014.

[28] N. Editorial. Announcement: Reducing our irreproducibility. *Nature*, April 2013. Available at http://www.nature.com/news/announcement-reducing-our-irreproducibility-1.12852.

[29] R. K. Ellis, W. T. Giele, Z. Kunszt, K. Melnikov, and G. Zanderighi. One-loop amplitudes for w+3 jet production in hadron collisions. Available at http://arXiv.org/abs/0810.2762.

[30] A. Farres, J. Lsaskar, S. Blanes, F. Casas, J. Makazaga, and A. Murua. High precision symplectic integrators for the solar system. *Celestial Mechanics and Dynamical Astronomy*, 116:141–174, 2013.

[31] S. Foley. Glaxosmithkline pays $3bn for illegally marketing depression drug. *U.K. Independent*. Available at http://www.independent.co.uk/news/business/news/glaxosmithkline-pays-3bn-for-illegally-marketing-depression-drug-7904555.html.

[32] A. M. Frolov and D. H. Bailey. Highly accurate evaluation of the few-body auxiliary functions and four-body integrals. *J. Physics B*, 36:1857–1867, 2003.

[33] B. Goldacre. What the tamiflu saga tells us about drug trials and big pharma. *U.K. Guardian*. Available at http://www.theguardian.com/business/2014/apr/10/tamiflu-saga-drug-trials-big-pharma.

[34] T. C. Hales. A proof of the kepler conjecture. *Annals of Mathematics*, 162:10651185, 2005.

[35] T. C. Hales. Lessons learned from the flyspeck project. 2011. Available at http://www-sop.inria.fr/manifestations/MapSpringSchool/contents/ThomasHales.pdf.

[36] Y. He and C. Ding. Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications. *J. Supercomputing*, 18:259–277, 2001.

[37] Y. Hida, X. S. Li, and D. H. Bailey. Algorithms for quad-double precision floating point arithmetic. *Proc. of the 15th IEEE Symposium on Computer Arithmetic*, 2001.

[38] J. Ioannidis. Why most published research findings are false. *PLoS Med*, 2(8), 2005.

[39] M. S. J. M. Borwein and C. Maitland. Computation of a lower bound to giuga's primality conjecture. *Proceedings of Integers13*, 2013. Available at http://www.westga.edu/~integers/cgi-bin/get.cgi.

[40] G. Lake, T. Quinn, and D. C. Richardson. From sir isaac to the sloan survey: Calculating the structure and chaos due to gravity in the universe. *Proc. of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1997.

[41] R. LeVeque, I. Mitchell, and V. Stodden. Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science and Engineering*, pages 13–17, July 2012.

[42] M. H. Macfarlane. A high-precision study of anharmonic-oscillator spectra. *Annals of Physics*, 271:159–202, 1999.

[43] M. McNutt. Reproducibility. *Science*, 343(6168):229, January 2014. Available at http://www.sciencemag.org/content/343/6168/229.summary.

[44] C. M. Micheel, S. J. Nass, and E. C. o. t. R. o. O.-B. T. f. P. P. O. i. C. T. B. o. H. C. S. B. o. H. S. P. I. o. M. Gilbert S. Omenn. *Evolution of Translational Omics: Lessons Learned and the Path Forward*. The National Academies Press, 2012. Available at http://www.iom.edu/Reports/2012/Evolution-of-Translational-Omics.aspx.

[45] R. V. Noorden. Major scientific journal joins push to screen statistics in papers it publishes. *Nature*, July 2014.

[46] H. Obokata, T. Wakayama, Y. Sasai, K. Kojima, M. P. Vacanti, H. Niwa, M. Yamato, and C. A. Vacanti. Retraction: Stimulus-triggered fate conversion of somatic cells into pluripotency. *Nature*, 511:641–647, July 2014.

[47] G. Ossola, C. G. Papadopoulos, and R. Pittau. Cuttools: A program implementing the opp reduction method to compute one-loop amplitudes. *J. High-Energy Phys.*, 0803:04, 2008.

[48] D. Overbye. Space ripples reveal big bang's smoking gun. *New York Times*. Available at http://www.nytimes.com/2014/03/18/science/space/detection-of-waves-in-space-buttresses-landmark-theory-of-big-bang.html.

[49] S. Pinker. *The Blank Slate: The Modern Denial of Human Nature*. Penguin Books, 2003.

[50] W. H. Press, S. A. Eukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

[51] F. Prinz, T. Schlange, and K. Asadullah. Believe it or not: how much can we rely on published data on potential drug targets? *Nature Reviews Drug Discovery*, 10:712.

[52] J. M. R. R. W. Robe and and R. Aulwes. In search of numerical consistency in parallel programming. *Parallel Computing*, 37:217–219, 2011.

[53] E. S. Reich. Cancer trial errors revealed. *Nature*. Available at http://www.nature.com/news/2011/110111/full/469139a.html.

[54] 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. *Proc. of SC13*. Available at http://www.davidhbailey.com/dhbpapers/precimonious.pdf.

[55] U. Simonsohn, L. Nelson, and J. Simmons. P-curve: A key to the file-drawer. *Journal of Experimental Psychology: General*, 143(2). Available at http://psycnet.apa.org/journals/xge/143/2/534/.

[56] V. Stodden. The legal framework for reproducible scientific research. *Computing in Science and Engineering*, 11, January 2009.

[57] 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. Available at http://www.davidhbailey.com/dhbpapers/icerm-report.pdf.

[58] V. Stodden, S. Miguez, and J. Seiler. Researchcompendia: Cyberinfrastructure for reproducibility and collaboration in computational science. *Computing in Science and Engineering*, January 2015.

[59] W. Tucker. A rigorous ode solver and smale's 14th problem. *Foundations of Computational Mathematics*, 2:53–117, 2002.

[60] D. Viswanath. The fractal property of the lorenz attractor. *J. Phys. D*, 190:115–128, 2004.

[61] D. Viswanath and S. Şahutŏglu. Complex singularities and the lorenz attractor. *SIAM Review*, 52:294–314, 2010.

[62] Z.-C. Yan and G. W. F. Drake. Bethe logarithm and qed shift for lithium. *Phys. Rev. Letters*, 81:774–777, 2003.

[63] T. Zhang, Z.-C. Yan, and G. W. F. Drake. Qed corrections of $o(mc^2\alpha^7 \ln \alpha)$ to the fine structure splittings of helium and he-like ions. *Physical Review Letters*, 77:1715–1718, 1994.