A Survey of Issues in Reliable Computational Science

Fred J. Hickernell
Department of Applied Mathematics, Illinois Institute of Technology
hickernell@iit.edu  mypages.iit.edu/~hickernell
Supported by NSF-DMS-1115392

Many thanks to Sou-Cheng Choi, collaborators, and speakers

My goal is that we continue and contribute to an ongoing discussion.

July 9, 2014
Three Modes of Investigation

**Theory**
- Rigorous argument
- Theorems

**Experiment**
- Carefully documented procedures
- Well designed
- Reproducible

**Computation**
- Efficient and tested code
- Heading towards exascale
- Increasingly important as capability grows

Bailey and Borwein (2013) say this pervades the other two
**Is it reliable?**

**Goal:** Under the conditions ________, the outcome is ________.
Reliable Computational Science and How to Get It

Goal of Computational Science:

Under the conditions ________, the outcome is ________.

› Want the process to be reliable—like for theory or experiment. Want to
  › clearly understand the assumed conditions underlying the computation,
  › view certain details of the computation, and
  › be able to reproduce, mimic, or improve upon the computation.

› So we need
  › accessible, published software together with input & output data,
  › robust, tested software,
  › efficient code that fits the hardware well,
  › justified algorithms with warning labels, and
  › to educate our community.
Software and Data Should Be *Published*

- Need to **re-use** software and build upon what was done before.
- Need to be able to view code and data to be **convinced** that the results are correct.

- Publication lies at the heart of **reproducible** research pioneered by Jon Claerbout and David Donoho (Buckheit and Donoho, 1995).
- LeVeque (2013 April) and Bangerth and Heister (2014 January) elaborate reasons to **publish code and data**, not just the numerical analysis.
- Stodden et al. (2014) describes **tools and practices** for reproducible research.
- Katz et al. (2014) give recommendations for **sustainable** software.
- NSF 14-059 solicits proposals for **standards** for software and data publishing.
Software and Data Should Be Published (cont’d)

- Publication has different magnitudes: short scripts to generate figures, driver programs, software packages, input and output data, . . . . What are the different standards for these?
- Software publication should be recognized as scholarship.
- Published code and data need to be documented so that others can follow.
- ACM-TOMS publishes software. www.runmycode.org is a repository for code and data that accompany articles. How is software reviewed? What about an Annual Review of Scientific Software or a Computational Science Consumer Reports?
- Tim Davis (Wednesday, 4:30 pm) describes how software is scholarly output.
- Barry Smith (Wednesday, 5:00 pm) describes how to automatically generate citations for the algorithms from a library that are used in a computation.
- Tanu Malik (Thursday, 8:30 am) describes how to imbed code and data in research articles.
- Bill Rider (Thursday, 10 am) addresses the explosion in the computational science literature and how software publication may make it more compact.
Software Should Be *Robust*

- Computations are performed under a wide *variety* of input parameters and computing environments. Software should respond well under diverse conditions.

- ACM-TOMS policy on submission of algorithms, toms.acm.org/AlgPolicy.html, provide important *guidelines* for clarity, robustness and efficiency that could be used more broadly.

- Suites of *tests*—short unit tests, realistic workouts—ensure that software works as expected.

- Code should be *portable* across platforms. It should tune itself to machine-dependent constants.
Software Should Be *Robust* (cont’d)

- Code should give *warnings* if the assumptions on which it is based might be violated or if computational cost budgets are exceeded.
- Code should be *resistant* to possible hardware failures or uneven communication speeds.
- Software Carpentry ([software-carpentry.org](http://software-carpentry.org)) teaches lab skills for scientific computing. They maintain a description of *best practices* (Wilson et al., 2013), with several recommendations for robustness.
- Sou-Cheng Choi (Tuesday, 4 pm) explained how GMRES was improved to handle *nearly singular* matrices.
Software Should Be *Efficient*

- The **computational cost** of software should be made known to the user.
- Ideally, the computational cost should be asymptotically **optimal**—track the cost of the best possible algorithm—as the problem size tends to infinity or the error tolerance vanishes.
- The software should adhere to **modern** programming practices that promote efficient use of memory, parallelization, re-use of code, etc.
- Code should be automatically optimized for the **architecture** of the system, including the memory size, the numbers of cores and their connections.
Software Should Be *Guaranteed* to Work

- Complex computations need automatic, adaptive algorithms, i.e., the computations are determined based on the error tolerance and the difficulty of the problem as revealed in the process of computing.
- **Examples** include MATLAB’s (The MathWorks, 2014) integral, fminbnd, and ode45, and the Chebfun toolbox (Hale et al., 2014).
- Existing adaptive algorithms have no guarantees of success, just a hope.
- Equivalently they have no warning labels of when they might fail. Do you know when \( \sin(x) \) gives the wrong answer?
- **Martha Razo** (Friday, 9:10 am) describes how existing adaptive quadrature can be fooled.
**integral(·) Fails for Spiky and Flukey Integrands**

\[
\int_{0}^{1} f_{\text{spiky}}(x) \, dx = \frac{1}{2}
\]

\[
\int_{0}^{1} f_{\text{fluky}}(x) \, dx = 0.278827
\]

```
integral(spikef,0,1, ...)  
'AbsTol',1e-13,'RelTol',1e-13)
ans =
0
```

```
integral(flukef,0,1, ...)  
'AbsTol',1e-13,'RelTol',1e-13)
ans =
0.278799
```

Unavoidable, but should be quantified

Lyness (1983) warned us about this.
Guaranteed Automatic Integration Library (GAIL)

- We are developing guaranteed adaptive algorithms published in GAIL code.google.com/p/gail/.
- Algorithms work for\textit{ not}\ $\mathcal{C}^1$,\ of functions (Hickernell et al., 2014; Clancy et al., 2014). They include upper bounds on computational cost in terms of the error tolerance and the problem difficulty. In some cases these are asymptotically\ textit{ optimal}.
- \textbf{Yuhan Ding} (Wednesday, 5:30 pm) describes a guaranteed adaptive linear spline algorithm.
- \textbf{Yizhi Zhang} (Thursday, 9:30 am) describes a guaranteed trapezoidal rule.
- \textbf{Lan Jiang} (Tuesday, 5:20 pm) described a guaranteed Monte Carlo method for computing means of random variables and multidimensional integrals.
- \textbf{Tony Jiménez Rugama} (Tuesday, 5:40 pm) described a guaranteed quasi-Monte Carlo method for computing multidimensional integrals.
- Ongoing work on relative error, higher order, local adaption, optimization, multi-level Monte Carlo, \ldots
Computational Scientists Need to Learn Good Practices

What we have

- Numerical analysis classes teach the theory of algorithms. Programming and data structures courses teach good coding practices.
- Scientists learn to write clear academic articles, give academic presentations, and perform laboratory experiments through practice and feedback.
- Existing articles written using \LaTeX{} and presentations and posters prepared using Beamer serve as templates.

What we need

- We need courses that combine numerical analysis and software engineering.
- We need standards and templates for publishing code to help our community learn.
- At IIT we have taught a 75 minutes × 10 week course for graduate students that tries to cover the principles described here. Students use Git and publish code that is tested.
- Lorena Barba (Thursday, 9:00 am) describes how to imbed good software practices in our computational science and engineering curriculum.
Reliable Computational Science and How to Get It

Goal of Computational Science:

Under the conditions ________, the outcome is ________.

- Want the process to be reliable—like for theory or experiment. Want to
  - clearly understand the assumed conditions underlying the computation,
  - view certain details of the computation, and
  - be able to reproduce, mimic, or improve upon the computation.
- So we need
  - accessible, published software together with input & output data,
  - robust, tested software,
  - efficient code that fits the hardware well,
  - justified algorithms with warning labels, and
  - to educate our community.

Looking forward to your contributions.
References I


References II


The MathWorks, Inc. 2014. *MATLAB 8.3*, Natick, MA.