Mathematical Software Development in Academic, Industry, and Government Labs

A personal perspective

Applied Math Seminar
Texas A&M, Feb 29, 2016

Tim Davis

Feb 2016

My background

```
x=sparse(A)\b
```

SuiteSparse: `x=sparse(A)\b`, partial user list

Industry: MATLAB (The MathWorks), Mathematics (Wolfram), Google, Cadence Design Systems, MSC Software, IBM, ANSYS, Berkeley Design Automation, Geomodeling Solutions, Orcina, ATopTech, Tandent Vision, Vision Map, EnerNex, FEAT, Freescale, Geograf, HRL Laboratories, Intex, Lumerical, Mentor Graphics, SIMetrix, COMSOL, NVIDIA, MSI Kenny, Accelogic.

Lab: HSL Mathematical Subroutine Library; Xyce, Amesos, and Trilinos (Sandia); Knolls Atomic Power Lab; PETsc (Argonne), FiPy (NIST).

Academia / Open-Source: Google Ceres (Street View, Photo Tours, and 3D Earth), GEGL (Gimp), Julia, SuperLU, MUMPS, Fedora Linux, Debian Linux, Arch Linux, Ubuntu Linux, OpenSUSE Linux, Scientific Linux, GNU Darwin, DarwinPorts, Fink, Octave, R, ROS, deal.II, scilab, CVX, SDPT3, CVXOPT, MBDyn, Boost, OpenSees, CGAL, Kraken, FEniCS, Eigen, SciPy.sparse, Pysparse, NLPy, SfePy, FreeFem++, Elmer, FLOODS/FLOOPS, MILAMIN, ILUPACK, JADAMILU, Cubica, LAMG, LiveV, M.E.S.S., AMDisS, PDCO, MLD2P4, FEATFLOW, FEAST, OpenSLAM.org (g2o, 2D-I-SLSJF, HOG-Man, RobotVision, SLAM6D, SSA2D, MTK, SLOM, iSAM, TJTF for SLAM), SSBA, libdogleg, NGSPICE, hp.fem/Hermes, PATH solver, MESA.

The Scholarly Work of Reliable and Well-Designed Mathematical Software

Outline:

- What defines a *scholarly work*?
- Does mathematical software fit that definition?
- Why publish prototype-quality mathematical software?
- Why publish robust, reliable, and full-featured mathematical software?
- Troubling trends and a troubling status quo
- My recommendations for authors of scientific software

Definition of “scholarly work”

Scholarly or artistic works, regardless of their form of expression: intended purpose is to disseminate the results of academic research, scholarly study, or artistic expression, ... in any medium.

- When is a work scholarly (software or any other form)?
- Proposed metrics:
  1. peer-reviewed
  2. published in reputable academic journals
  3. published in academic-quality books
  4. topic of discussion at academic conferences
  5. citation for academic society fellow

Peer-reviewed software

- **ACM Transactions on Mathematical Software**
- Founded in 1975
- Purpose:

  > *As a scientific journal, ACM Transactions on Mathematical Software (TOMS) documents the theoretical underpinnings of numeric, symbolic, algebraic, and geometric computing applications. It focuses on analysis and construction of algorithms and programs, and the interaction of programs and architecture. Algorithms documented in TOMS are available as the Collected Algorithms of the ACM in print, on microfiche, on disk, and online.*

- 942 Collected Algorithms of the ACM since 1975: peer-reviewed scholarly software
- I serve as an Associate Editor of this journal

ACM TOMS Editorial Charter

The purpose of the ACM Transactions on Mathematical Software (TOMS) is to communicate important research results addressing the development, evaluation and use of mathematical software. In addition, TOMS publishes machine-readable computer software ... In both research papers and software, TOMS seeks contributions of lasting value in which technical quality, relevance to significant computations, interest, and novelty are all high, and where presentation is effective.

ACM TOMS Algorithms Policy

- code is submitted, peer-reviewed, and published by the journal
- short codes are printed in hard copy journal
- all available electronically
- code novelty and quality is essential
- **toms.acm.org/AlgPolicy.html**
  - *The algorithm paper should give a brief description of what the software does and pertinent information on usage and maintenance.*
  - *An ACM algorithm must be complete, portable, well documented, and well structured.*

Scholarly software in academic books and journals

- Long history of academic software. For example:
- George Forsythe (1917 - 1972), founder of computer science as an academic discipline
  - consider mathematical software as scholarly work
  - founder and head of Stanford's Computer Science Department
  - President of ACM
  - author of *Computer solution of linear algebraic systems*, 1967. Includes lengthy code printed in the book

George Forsythe: software as scholarly work

Upon George Forsythe's death:

- *It is generally agreed that he, more than any other man, is responsible for the rapid development of computer science in the world's colleges and universities. His foresight, combined with his untiring efforts to spread the gospel of computing, have had a significant and lasting impact; one might almost regard him as the Martin Luther of the Computer Reformation!*
- *He inaugurated a new area of* **scholarly work:** *refereeing and editing algorithms.* (emphasis mine)

by Donald Knuth, Communications of the ACM, 1972

George Forsythe: software as scholarly work

Article by Knuth, continued: *As an indication of his behind-the-scenes activities, here are some more excerpts from letters he wrote during January and February 1964:*

- *The program is really in poor style, and I'm peeved with the referee for not saying so. You use a switch and a mass of goto's where straightforward ALGOL would use conditional expressions. You even goto "here" from the line above "here"!! [January 8, 1964]*
- *I am sorry that refereeing increases the time between submittal and publication, but I am confident that the net result of refereeing will be a large gain in the quality of our algorithms. [January 13, 1964]*
- *It is very hard to find matching begins and ends, which should be above each other or on the same line. [January 29, 1964]*

SIAM Fellows w/ software contributions (partial list)

Marsha Berger, Courant: PDEs

Bill Gear, NEC: ODEs, DAEs

Jorge Moré, Argonne: optimization

Nick Trefethen, Oxford: sci. computing

Alan George, Waterloo: sparse matrices

Cleve Moler, MathWorks: MATLAB

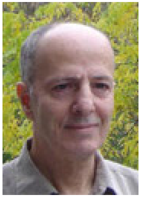
Linda Petzold, UCSB: DASSL (DAEs)

Uri Ascher, UBC: diff.eqns'

SIAM Fellows w/ software contributions (partial list)

Iain Duff,
Rutherford:
sparse matrices

Rande LeVeque,
UWash.:
hyperbolic PDEs


Yousef Saad,
UMinn:
sparse matrices

Danny Sorensen,
Rice:
ARPACK

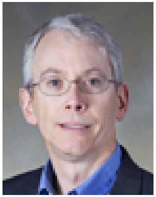
Bill Gropp,
UIUC:
MPI

John Lewis,
Boeing:
sparse matrices

Pavel Bochev,
Sandia:
finite element

Sven Hammarling,
NAG:
LAPACK

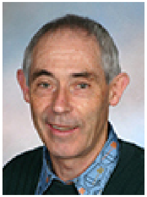
SIAM Fellows w/ software contributions (partial list)


Bruce Hendrickson, Sandia: graph algo.

Barry Smith, Argonne: PETSc

Andrew Conn, IBM: optimization

T.D. Texas A&M: sparse matrices


Michael Saunders, Stanford: optimization

Philip Gill, UCSD: optimization

Jack Dongarra, UTK: matrix algo.

Esmond Ng, LBNL: sparse matrices

Apply the metrics: Software as scholarly work

- **Definition of scholarly work**: purpose must be to disseminate results of academic research, scholarly study, or artistic expression
- Characteristics of scholarly work:
  1. peer-reviewed
     ✓ ACM TOMS, peer-review of mathematical software
  2. published in reputable academic journals
     ✓ ACM TOMS, SIAM Journals (as supplimentary appendices)
  3. published in academic-quality books
     ✓ Many SIAM books include mathematical software or describe mathemaical software
  4. topic of discussion at academic conferences
     ✓ many sessions at SIAM conferences
  5. citation for academic society fellow
     ✓ *"For contributions to ... software, ..."* is not uncommon
- Thus: peer-reviewed software is a scholarly work

So why publish (prototype) mathematical software?

- useful to provide the code even as research prototype quality
- reproducibility of research results
- for a complex algorithm: not all details can fit in a paper
- *Publish your code - it is good enough*, Nick Barnes, Nature 467, 753 (2010).

  > *That the code is a little raw is one of the main reasons scientists give for not sharing it with others. Yet, software in all trades is written to be good enough for the job intended. So if your code is good enough to do the job, then it is good enough to release – and releasing it will help your research and your field.*

Randy LeVeque's thought-experiment

LeVeque's thought-experiment: What if we treated proofs like we now treat code?

- *The proof is too ugly to show anyone.*
- *I didn't work out the details.*
- *Giving out my proof would give away my secrets.*
- *My proof is valuable intellectual property.*
- *Proofs are too hard to review.*
- ...

Top Ten Reasons To Not Share Your Code (and why you should anyway), April 2013, SIAM News

Randy LeVeque's answers (mine added)

- *The code is ugly to show anyone:*
  ugly code is better than no code.
- *I didn't work out the details:*
  the details you did work out are important to the scientific record.
- *Giving out my code would give away my secrets:*
  but as applied mathematicians we want others to rely on our methods.
- *My code is valuable intellectual property.*
  it's protected by copyright.
- *Code is too hard to review.*
  At ACM TOMS we've been doing it for decades.
- ...

Troubling Trends

Troubing Trends in Scientific Software Use, Science, Vol 340, May 2013.

> *A survey of scientists in ecological modelling: ... Many of these scientists rely on the fact that the software has appeared in a peer-reviewed article, recommendations, and personal opinion, as their reason for adopting software. This is scientifically misplaced, as the software code used to conduct the science is not formally peer-reviewed.*

Troubling Trends, continued

Recommendations, Moving Forward

- **Education:** *Universities should produce scientists capable of instantiating science in code such that other scientists are able to peer-review code as they would other aspects of science. Formal training in statistics, computational methods, mathematics, and software engineering should be a core part of the science curriculum at undergraduate and research student levels.*
- **Scientific publication:** *Scientific software code needs to be not only published and made available but also peer-reviewed.*

Cultural gap

Goal	Academia	Industry/Govt
new theory, algo, data structures	*	
publishable papers	*	
meets a theoretical need	*	
advances state-of-the-art (theory)	*	
advances state-of-the-art (practice)	*	*
competitive performance	*	*
meets a practical need	*	*
software prototypes	goal	means
usable in production code	specific	general
software robustness		*
software generality		*
well-thought out API		*

So why publish polished mathematical software?

Typical path towards academic success:

- Create a new algorithm
- Write a paper
- Expect others to implement your algorithm

An alternative path:

- Create a new algorithm and the software that implements it
- Write a paper and submit your code along for peer review
- Enable others use your algorithm and its software

How to publish polished mathematical software

- Code like a theorem/proof: clear, concise, complete, and well-written.
- code is to algorithm as proof is to theorem
- Track down the bugs like you track down a flaw in a proof.
- Test suite: ensure all lines exercised and results verified.
- Publish the test suite and scaffolding-code
- Documentation: not an after-thought.
- Sample result: 3 bugs found in 15 years in 120K lines of code in x=A\b

My experience

10 Collected Algorithms of the ACM

1. *Algorithm 9xx: Sparse QR factorization on GPU architectures*, in revision
2. *Algorithm 933: Reliable calculation of numerical rank, null space bases, pseudoinverse solutions, and basic solutions using SuiteSparseQR*, 2013
3. *Algorithm 930: FACTORIZE, an object-oriented linear system solver for MATLAB*, 2013.
4. *Algorithm 915: SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization*, 2011.
5. *Algorithm 907: KLU, a direct sparse solver for circuit simulation problems*, 2010.
6. *Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*, 2008.
7. *Algorithm 849: A concise sparse Cholesky factorization package*, 2005.
8. *Algorithm 837: An approximate minimum degree ordering algorithm*, 2004.
9. *Algorithm 836: COLAMD, an approximate column minimum degree ordering algorithm*, 2004.
10. *Algorithm 832: UMFPACK - an unsymmetric-pattern multifrontal method with a column pre-ordering strategy*, 2004.

My experience, continued

- My work follows Forsythe's tradition
  - Forsythe and Moler, *Computer solution of linear algebraic systems*, Prentice-Hall, 1967.
  - T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM Series on the Fundamentals of Algorithms, SIAM, Philadelphia, PA, 2006.
- Both books include lengthy printed mathematical software
- Editor of my SIAM Book (Nick Higham, SIAM Fellow, Fellow of the Royal Society), critiqued my software in the book for mathematical clarity.

Peer-review of academic software

- **Anonymous review of my SIAM book**

  > *The algorithms are explained in terms of complete, accessible, and readable code. In the grand tradition of Forsythe/Malcolm/Moler and George/Liu, the code is intended to be both read (for concrete understanding) and used (as a tool) by the student. Code that can fill both roles is extremely rare; Davis's is excellent.*

- **Review by Nick Higham, Editor of my SIAM book**, a Fellow of SIAM and the Royal Society, and a Vice President of SIAM

  > *I see you have put the C listings in a smaller font. I prefer to use the same font size, both for readability and* **because the code is just as important as the text (cf. any of Knuth's books!).** *(emphasis added)*

My academic talks on the general development of mathematical software

- T. A. Davis, *Sparse Matrix Research and Software Development at the University of Florida*, SIAM Annual Meeting, San Diego, 2008.
- mini-symposium at 2014 SIAM Annual Meeting: **Reliable Computational Science.** *Research, development, and operations depend increasingly on extensive computations, which must be reliable. Code and data must be made available so that the results can be reproduced and verified by others. Collaborative development and code reuse require well structured, efficient, tested, and documented code. ...*
- Scientific Software Days 2016, UT Austin

Mathematical software in citation as SIAM, ACM, and IEEE Fellow

- the academic community recognizes my work as scholarly software
- Davis elected as Fellow of the Society for Industrial and Applied Mathematics (SIAM), 2013
- Citation (emphasis mine):

  > *For contributions to* **sparse matrix algorithms and software***, including the University of Florida Sparse Matrix Collection*

- also Davis' ACM (2015) and IEEE (2016) Fellow citations
- of 393 SIAM Fellows, 14 citations explicitly list **software**

Recommedations

For the researcher:

- Create new algorithms.
- Embody them in polished code and submit them for peer-review.
- Clear path towards a career goal of high-impact research.
- For the industrial researcher: if a company allows you to publish your proof, why not your code?

For the academic / lab / industrial research community:

- Goal: enhance reproducibility of research
- Goal: enhance future research by fostering mathematical software tools
- Encourage code submission, even non-peer-reviewed prototype-quality.
- Allow for peer-review of high-quality software.

The Scholarly Work of Reliable and Well-Designed Mathematical Software

Summary:

- **Scholarly work**:
  Purpose is to disseminate results of academic research.
- **Mathematical software**:
  Essential method for disseminating complex algorithms, can be peer-reviewed.
- **Why publish prototype-quality mathematical software?**
  Research reproducibility.
- **Why publish robust, reliable, and full-featured mathematical software?**
  Let others rely on your work and build on it. Like theorem+proof.
- **Troubling trends and a troubling status quo**:
  Too much software hidden from scientific study. Needs to be included with journal publication (both reviewed and unreviewed).
- **My recommendations for you and for the research community:**
  Write and publish high-quality peer-reviewed software.