Research | December 02, 2019

Computational Research Software: Challenges and Community Organizations Working for Culture Change

By Lois Curfman McInnes, Daniel S. Katz, and Scott Lathrop

Do you *use* software in your research? Do you *develop* software, either for your own work or to share with others? Are you a leader, stakeholder, or supporter of a project that uses or develops research software?

Computational science and engineering (CSE) depends on software that integrates advances in mathematics, statistics, computer science, and core disciplines from science and engineering. Modeling, simulation, and data analytics are driving new discoveries and technological innovations that benefit all areas of study and all sectors of society — from the physical, life, and social sciences to business, finance, and government policy.

High-quality software typically emerges from sustained community involvement, requirements gathering, sharing, and collaboration. Improvements in software development and utilization are ensuring that software is reusable and research results are reproducible, thereby reducing time to novel scientific discoveries.

However, we currently face a dramatic increase in software complexity due to a confluence of trends in emerging heterogeneous computer architectures and new frontiers of predictive science. Here we discuss technical and social challenges in research software, and introduce efforts in this area by grassroots organizations as well as projects to improve software quality, productivity, and sustainability. These endeavors ensure the integrity of research results and enable more effective collaboration. We emphasize numerical software while also considering the broader scope of research software.

Computational Software Successes and Strategies

The SIAM community has a long tradition of advancing theory, algorithms, and software for computational mathematics and scientific computing, including the development of popular mathematical libraries that underpin countless CSE advances. For example, high-performance packages like deal.II, DUNE, FeniCS, hypre, p4est, PETSc, Sundials, SuperLU, Trilinos, and waLBerla provide discretizations, solvers, integrators, and related capabilities for large-scale numerical simulations. Software libraries—high-quality, encapsulated, documented, optimized, tested,[1] and multiuse software collections—provide the functionality desired by application developers who do not want to spend time "reinventing the wheel." A library user only needs to know the library interface, and when and how to appropriately utilize library functionality.

Well-designed software libraries encapsulate cutting-edge algorithms and domain-specific expertise, thereby enabling users to reduce coding efforts while also customizing and extending capabilities as needed to exploit application-centric knowledge. A key design principle for high-performance packages is the use of interfaces that are independent of physical processes and separate from algorithm choices and data structures. Employment of abstractions for mathematical objects (such as vectors and matrices) to enable a clean user interface supporting a variety of underlying implementations is also important. For example, a high-level matrix platform can uphold sparse and dense representations, including variants for parallel architectures and heterogeneous central processing unit/graphics processing unit configurations, as well as composite and matrix-free operators. Such approaches help developers manage complexity and change.

Technical and Social Challenges in Research Software

As we work toward predictive science, the use of good design for individual software efforts is not enough; we must be able to combine and leverage diverse codes for the various phases of modeling, simulation, and analysis. Collaboration is essential because the full scope of required functionality is too broad for any single person or team to deeply understand (see Figure 1). At a practical level, collaborative software efforts require high-quality, trusted code that is sustainable, extensible, and portable. Simulations built from external software must be compatible and interoperable. Regular tests for interoperability help to ensure sustainability. To assure confidence in computational science discoveries, teams need to improve transparency and reproducibility of computational results.

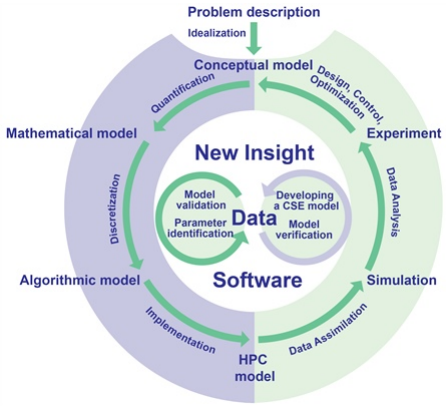

Figure 1. Software is the foundation of sustained computational science and engineering (CSE) collaboration and scientific progress. Figure courtesy of [2].

We must therefore embrace software ecosystem perspectives that explicitly consider relationships among distinct codes and their development communities. Research software is almost never entirely developed by a single project; each project instead depends on, uses, and builds upon software from other projects and developers. Much like research itself, research software is developed and maintained by a community ecosystem of competing and collaborating efforts. The open-source movement and its culture of sharing and collaboration—similar to the culture of open science and open research toward which the community is moving—enhances these efforts.

Consequently, we must address cultural issues in scientific software and social concerns in software communities. Determining value metrics for research software; increasing rewards for developers of open-source, reliable, extensible, and sustainable software; and expanding career paths for expert research software developers all require work. We have to create funding models for software sustainability, as well as models for software citation and credit (including approaches for publication and peer review). Finally, students and researchers need training about best practices in research software.

Community-led Culture Change in Research Software

To address these circumstances and promote research collaboration in emerging software ecosystems, community members have established a variety of grassroots organizations and projects inspired by the growth of the internet, shareable digital resources, and collaborative tools such as GitHub and Slack. Groups that focus on a particular discipline, technology, or functional skill set can help researchers understand relevant parts of the ecosystem, including the various available software packages and how they compare. These community organizations can support ecosystem health by encouraging policies, reuse, and collaboration that urge best practices for software development. Below are a few organizations and ideas that should be of interest to the SIAM community.

- The Software Sustainability Institute (SSI): Cultivating better and more sustainable research software to enable world-class research for the U.K. research community.
- Conceptualization of a U.S. Research Software Sustainability Institute (URSSI): Planning an institute to improve science and engineering research by supporting the development and sustainability of research software in the U.S.
- Interoperable Design of Extreme-scale Application Software (IDEAS) Productivity Project: Catalyzing advances in software productivity and sustainability that are driven by the needs of extreme-scale computational science.
- Better Scientific Software (BSSw): Encouraging the exchange of information on practices, techniques, experiences, and tools to improve developer productivity and software sustainability for CSE and related areas of technical computing.
- Apache Software Foundation: Fostering the growth of open-source software communities and providing the necessary technical infrastructure and support mechanisms.
- Software Carpentry: Teaching foundational coding skills to researchers and empowering them to develop research software, automate research tasks and workflows, and perform reproducible science.
- Working Towards Sustainable Software for Science: Practice and Experiences (WSSSPE): Promoting sustainable research software by addressing challenges related to the full lifecycle of such software via shared learning and community action.

- NumFOCUS: Promoting open code for better science while emphasizing sustainable high-level programming languages, open code development, and reproducible scientific research.
- rOpenSci: Enabling open and reproducible research by creating technical and social infrastructure and advocating for a culture of data sharing and reusable software.
- Extreme-scale Scientific Software Development Kit (xSDK): Promoting collaboration and commitment to independent numerical library efforts through community-based policies for quality improvement, better infrastructure, and the use of diverse libraries for large-scale CSE.
- Research Software Alliance (ReSA): Bringing communities together to collaborate on the advancement of research software.

Further information on best practices and reusable software that spans multiple disciplines and organizations is available in a special issue of IEEE's *Computing in Science and Engineering* on "Accelerating Scientific Discovery with Reusable Software" [1].

Get Involved

The aforementioned organizations act in their respective spheres of influence to nurture communities, change research culture, and promote the growth of software ecosystems. They provide information about effective approaches for creating, sustaining, and collaborating via scientific research software while simultaneously articulating key issues to stakeholders, agencies, and the broader research community. These actions inspire changes in policies, funding, and reward structure, and advance understanding of the importance of high-quality software to the integrity of computational research. We encourage you to explore the resources provided by groups that are relevant to your research, and consider joining them so we can collectively tackle these issues.

---

[1] In an ideal world, open-source software has fewer bugs than closed source software since more people view it. See "Linus's Law."

References

[1] Lathrop, S., Folk, M., Katz, D.S., McInnes, L.C., & Terrel, A. (2019). Introduction to accelerating scientific discovery with reusable software. *Comp. Sci. Eng., 21*, 5-7.

[2] Rüde, U., Willcox, K., McInnes, L.C., & De Sterck, H. (2018). Research and education in computational science and engineering. *SIAM Rev., 60*(3), 707-754.

Lois Curfman McInnes is a senior computational scientist in the Mathematics and Computer Science Division of Argonne National Laboratory. Her research focuses on high-performance numerical software for large-scale computational science. She also coordinates work on mathematical libraries in the U.S. Department of Energy's Exascale Computing Project and oversees the software column in *SIAM News*. Daniel S. Katz is Assistant Director for Scientific Software and Applications at the National Center for Supercomputing Applications. He is also a research associate professor in the Departments of Computer Science and Electrical and Computer Engineering, and in the School of Information Sciences at the University of Illinois at Urbana-Champaign. Katz works on software development and issues related to software sustainability, software citation and credit, and software career paths. Scott Lathrop is the Blue Waters Education, Outreach, & Training Technical Program Manager at the National Center for Supercomputing Applications. He focuses on advancing community knowledge for discovery through the use of resources and services for high-performance computing.