The Community is the Infrastructure: A Short Discussion of the PETSc Community

Mark Adams[1], Jed Brown[2], Satish Balay[3], Victor Eijkhout[4], Jacob Faibussowitsch[5], Fande Kong[6], Matthew Knepley[7], Scott Kruger[8], Oana Marin[3], Richard Tran Mills[3], Todd Munson[3], Patrick Sanan[9], Barry F. Smith[10], Hong Zhang[11], Hong Zhang[3], and Junchao Zhang[3]

[1]Lawrence Berkeley National Laboratory
[2]University of Colorado at Boulder
[3]Argonne National Laboratory
[4]The University of Texas at Austin
[5]University of Illinois at Urbana-Champaign
[6]Idaho National Laboratory
[7]University of New York at Buffalo
[8]Tech-X Corporation
[9]ETH Zurich
[10]Argonne Associate, Argonne National Laboratory
[11]Illinois Institute of Technology

July 5, 2021

Abstract

Hard and soft infrastructure work in tandem to accomplish a particular enterprise. Soft infrastructure – the formal and informal culture, institutions, standards, practices, and procedures that support an enterprise – is often more important to human endeavors than hard infrastructure. For example, a highway system's hard infrastructure (e.g., the roads) could not satisfy its enterprise – effective transportation — without the corresponding soft infrastructure of regulations, policing, driver habits, maintenance crews, and so forth. The relative cost and importance of the different aspects of infrastructure evolves, particularly during the development stages, from planning, to construction, to commissioning, operation, maintenance, and upgrading.

The design, development, support, and dissemination of computer software is an archetypal example of soft infrastructure, regularly being the tail that wags the dog of computer hardware (prototypical hard infrastructure). Quantifying soft infrastructure is more difficult than hard infrastructure. Thus, it is often ignored or under-emphasized when analyzing or proposing changes to human-developed systems. In particular, government understanding and funding of scientific soft infrastructure lags well behind that of hard infrastructure (for example, experimental devices such as accelerators). In fact, the funding for soft infrastructure often only flows as a by-product of the funding for tangentially-related hard infrastructure.

This paper explores the soft infrastructure of the Portable Extensible Toolkit for Scientific computing (PETSc), which we dub the PETSc community. To address the needs of next-generation science the PETSc community is simultaneously engaged in developing additional features and algorithms (construction and commissioning), supporting a vibrant set of users and developers (operation), and maintaining and upgrading existing software. The informal PETSc community agglomeration across these activities is the most valuable asset of PETSc. While we briefly discuss some of the technology applied to PETSc's development and support, the dominant story is that of PETSc community building and strengthening, while considering the lessons we have learned that may apply to other software development projects.

To meet the technological challenges of the 21st century, the ongoing revolution in data management needs to be mirrored by a revolution in scientific simulation. Advances in manufacturing, energy infrastructure and generation, and so forth must be underpinned by flexible, scalable, multiphysics multiscale simulation capabilities. In turn, this simulation technology rests on a foundation of numerical algorithms and software. This foundation rises to the level of hard infrastructure, such as an energy grid, semiconductor foundry, or particle accelerators, in importance, but realizing its funding and organizational models will probably be different. It must stand the test of time, support diverse interests, and incorporate cutting-edge research while running on the most advanced hardware. This will require significant investment for development and upgrades, but more importantly, it will require human investment and new ways of organizing the software development, support, and maintenance effort.

Academia is a natural home for leading-edge research on algorithms and software, but faculty incentives are geared toward short-term priorities, which make supporting such decade-long efforts unmanageable. Likewise, industrial efforts benefit from a close association of research with practical application but have often foundered due to a closed development model and narrow customer focus. National laboratories, on the other hand, have a forward-thinking policy toward funding long-term software programs, but can be insular, focusing almost exclusively on internal customers. Sustainable infrastructure requires combining the strengths of each type of institution while mitigating the weaknesses, creating a value proposition for a diverse set of international stakeholders to engage and contribute to the community effort.

Though we think of numerical software emerging as part of soft infrastructure, some of the early, influential numerical software packages, including EISPACK, LINPACK, and LAPACK, were based on the plan-propose-fund-build-use model, much like traditional hard infrastructure [7]. The packages are more exemplars of the hard infrastructure development model than soft. Today's numerical software libraries (in fact, more generally, most software today) are no longer plan-propose-fund-build-use. They require comprehensive maintenance, extension, and support systems as hardware, software, user, and community needs evolve rapidly. Scientific software libraries are soft infrastructure. There is a large collection of writings on software communities, the following list only scratches the surface: [6], [4], [2], [8], [3].

The PETSc community as infrastructure PETSc, apart from being a popular library for scientific computing, is a group of people dedicated to developing, extending, maintaining, and using the library to solve scientific and engineering problems. The PETSc community has strong representation in laboratories, academia, and industry. The PETSc community boosts broader priorities for members in organizations with more limited goals, while also providing continuity of support and vision for members in institutions with shorter time horizons.

PETSc was not purposefully designed to support long-term, community software infrastructure. Rather, work on the software inspired the creation of a set of practices to enable a small development team with large ambitions and a long event horizon to develop and support software capable of solving problems of interest to the original developers and their collaborators. However, these practices, reviewed below, appear to have wider benefits, and certain community properties could serve as a template for long-term software infrastructure.

- Swift, in-depth engagement on the communication channels,[1] especially for new users.
- Encouragement and opportunities for anyone to contribute to the software and documentation.
- Providing a virtual institution for collaboration.
- Extensible interfaces that enable people interested in mathematics and algorithms to experiment and deploy research.

[1]Mailing lists, issue boards, and so forth.

- Developer autonomy to pursue topics about which individuals are passionate.
- Strong ties to academia, industry, and laboratories worldwide. Being spread throughout the world the community allows a real-time transfer of knowledge across institutions and application fields. This assures algorithmic development, preserves its state of the art status, and benefits the scientific community at large. This may be difficult to achieve in a moderate sized dedicated commercial software group.

Engagement PETSc places a high premium on engagement and support of users, in no small part because most developers are also users of the library. Moreover, to maintain the vitality of the library, new algorithmic developments must be rapidly integrated, bugs promptly fixed, and awkward constructions removed. This requires PETSc to establish a high level of user trust, communicating that, even in the face of rapid evolution, the library will be well-supported, and user code will continue to run with help from the community. A precondition of establishing trust is to create a welcoming environment for new users and developers. The PETSc community includes a wide range of professions, backgrounds, and levels of involvement, with individual members often participating in several ways over the years. Engagement is key to disseminating tacit knowledge and developing the skills and social support for users to become contributors as developers and mentors. The PETSc community has developed a broad base of people with expertise and kindness to reduce and report bugs, to mentor newcomers, and contribute in other ways. However, the community lags in racial and gender diversity relative to broader computational science. We attribute this to structural deficiencies, inertia, and insufficient outreach. The community recognizes this weakness and has become more deliberate in recent years, establishing norms and proactive outreach to build more diverse representation, to be responsive to historically-unrecognized stakeholders, and to benefit from social and technical contributions.

Collaboration PETSc's community helps members find funding, access expertise, and transition between roles in the project. One of the greatest difficulties in maintaining a coherent software project over decades is providing a career path for contributors. PETSc has given academic contributors a solid foundation for advancement via awards, the highly-cited users manual, professional recognition, and many productive collaborations born from PETSc development, maintenance, and user support. PETSc provides resource sharing from collaborative grants and collaboration opportunities that extend beyond the development group. In some cases, PETSc affiliation may be more important than departmental affiliation, especially since modern academic departments are often atomized, with little internal collaboration. The community provides a strong academic connection for industrial and laboratory members, tangible outputs recognized by future employers, and active participation in the wider computational science and engineering community.

Autonomy PETSc has no official organizational hierarchy, in that there is no appointed leader or elected offices. Decisions are made collectively by affirmation, although not always with unanimity. Membership is determined by a willingness to contribute to the codebase, including documentation, examples, and tutorials. The individual developers largely set development priorities. There is deference to members who have made deeper contributions to a particular component when making decisions about its development, but this kind of conservatorship changes over time, depending on interests, available time, and new ideas. Tabulated voting, elections, and enforcement of a schedule may make a product more predictable for users, and stricter controls on the introduction of new features or large changes may make for a simpler and more maintainable codebase. However, these practices also tend to stifle creativity, impede evolution, and discourage reworking inconvenient historical interfaces. PETSc errs on the side of empowering the developer, while still maintaining a development guide [1] and rules of development. It replaces the certainty of a strict development plan with the certainty of user support. This, along with the wide variety of contributors, allows for a level of agility that might not otherwise occur. It does, however limit the ability to organize a large centralized response to change when it may be needed.

Roles of PETSc PETSc has a broad overlapping community because it serves many roles. The roles include but are not limited to (see Figure 1): A *research platform* targeting cutting-edge algorithmic development. An *HPC library*. A repository of template applications via the wealth of examples that accompany the library. A *compendium of algorithms*. A *pedagogical tool* for training numerical analysts on HPC platforms [5]. A *promoter of best-choice numerical methods* in its role as an interface between academic algorithmic development and user necessities. This role is essential to both application codes in need of adapting and accelerating their solvers, as well as academic method developers seeking feedback and a promotion platform. Finally PETSc provides an *algorithm management system* to provide state-of-the-art partitioning, preconditioning, and other capabilities via a flexible PETSc configuration code.

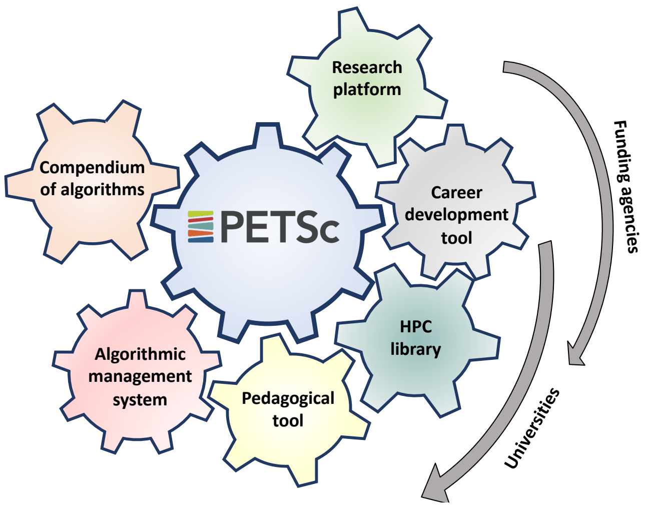

Figure 1: The multiple interconnected roles of PETSc driven by the scientific ecosystem.

Roles of members of the PETSc community Virtually all active PETSc community members are PETSc users, while a smaller subset of these, who almost always began as PETSc users, are also PETSc developers. PETSc users who do not work on its development contribute to PETSc with suggestions, bug reports, bug fixes, new features, and improved documentation. In no small part due to the emphasis on autonomy, engagement, and collaboration, it is possible and indeed common for individuals to move between different roles in the PETSc community. There is a "long tail" of members who contribute with lower frequency than the most active developers, yet collectively contribute a great deal. There are well over 100 contributors to the PETSc Git repository, with no single institution providing the majority of the contributors. The community structure is crucial in providing a pathway to increased involvement for interested users.

PETSc users can be categorized in a variety of ways, e.g. along institutional lines.

- **Academic users:** These are students, faculty members, research and development staff in universities and government agencies. Students might use PETSc to do their homework or develop a serious code for their paper or thesis. Student users have incentives to contribute their code back to PETSc to make their paper or thesis more credible. As they graduate, they likely bring PETSc to fields in industry or academia. Some have become PETSc developers.
- **Industrial users:** These use PETSc in their company's research or commercial product. PETSc's 2-clause BSD license eases its commercial uses. They may request support that is unlikely to be funded by government agencies, for example, support on Microsoft Windows. But fortunately, there are PETSc community members with the expertise to help with these types of requests. These users also require discretion and confidentiality in what they provide to members of the PETSc community who are helping them. In addition, they cannot always share their use cases, so the PETSc community needs to provide general solutions without understanding the specifics of what is needed. Developing the trust needed for industrial users

is a slow, gradual process that recognizes its importance from both sides.

Another way to categorize PETSc community members is by the focus and goals of their work.

- **Algorithm developers** are generally members of academia, they focus on the development and analysis of algorithms, and as such may be less concerned about generality and usability. They use PETSc because it allows them to use the large library infrastructure to prevent unnecessary coding. They face the challenge of writing scalable implementations in C, which, even in PETSc, can be time-consuming with a steep learning curve.
- **Application developers** focus on producing a code addressing one type of simulation well at a low cost. These are often discipline scientists or engineers for whom simulations are an important component of their job.
- **Scientific toolkit developers** develop numerical simulation toolkits that target a high-level solution of the class of problems they are interested in by leveraging PETSc capabilities and introducing additional infrastructure. These include Firedrake, MOOSE, Deal.II, and so on.

The PETSc community develops, maintains, and supports the PETSc library. User support is a crucial benchmark of a healthy community. Members of the community usually get back to user requests in hours, if not minutes. This engagement helps new users to feel welcomed and valued, make rapid progress, and hence gain confidence that using PETSc is the right choice. Through users' feedback, experienced PETSc community members get to know what works and doesn't and where to improve the software. From feature requests and discussions, PETSc community members often discover new research topics for funding.

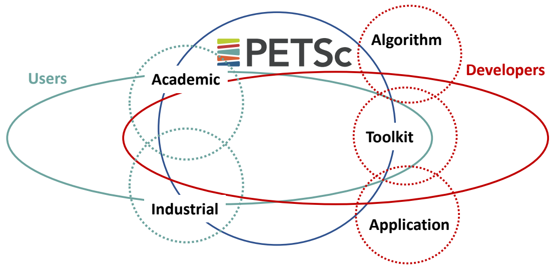

Figure 2: PETSc is a scientific junction; the boundary between types of users and developers is fluid; roles shift and change.

However, providing good user support is a substantial effort, consuming a large amount of PETSc developers' time, particularly when users encounter difficult bugs or performance issues at scale. User-developer communication on a particular topic could last for weeks or even months. PETSc developers need to be patient and consistently engaged with users. One may question whether this practice is sustainable, but so far, it works reasonably well.

Community to community The PETSc software calls many other third-party libraries, including MPI, MUMPS, SuperLU, Hypre, etc., and is used in many higher-level toolkits mentioned above, each of which has its own community. PETSc developers and users can also be developers or users of other packages; some of these people serve as liaisons that connect the communities. These individuals speak the languages of both communities and understand PETSc capabilities and the needs of their communities. They can appropriately explore, explain, and introduce PETSc features into their communities. This helps expand the reach of PETSc community. They also reduce the PETSc maintenance burden by addressing many PETSc questions in their communities while contributing PETSc patches, providing feature requirements, and even serving as final testers of PETSc releases. This connection helps both communities. For example, we explained PETSc's MPI communication requirement to MPICH developers and, they, in turn, use PETSc tests to drive their optimization. Another example is the liaison from the MOOSE team who understands the requirements of varied modeling and simulation in the language of nuclear energy and concisely translates the requirements to the PETSc language. Though the requirements of the MOOSE community drive these additions, they turn out to be useful for the wider PETSc community. These interactions increase the strengths of both communities.

Acknowledgments

This material was partially based upon work funded by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. This work was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0016140. Matt Knepley and Jed Brown were partially supported by U.S. DOE Contract DE-AC02-0000011838.

Bibliography

[1] PETSc developers guide, 2010. https://petsc.org/release/developers/.

[2] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. On the abandonment and survival of open source projects: An empirical investigation. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–12. IEEE, 2019.

[3] Wolfgang Bangerth and Timo Heister. What makes computational open source software libraries successful? *Computational Science & Discovery*, 6(1):015010, 2013.

[4] C. Titus Brown. Sustaining open source: thinking about communities of effort, 2019. http://ivory.idyll.org/blog/2019-communities-of-effort.html.

[5] Ed Bueler. PETSc for partial differential equations: Numerical solutions in C and Python, 2020.

[6] Nadia Eghbal. Roads and Bridges: The unseen labor behind our digital infrastructure. https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/.

[7] P.A. Freeman, W.R. Adrion, and W. Aspray. *Computing and the National Science Foundation, 1950-2016: Building a Foundation for Modern Computing*. ACM Books. Association for Computing Machinery and Morgan & Claypool Publishers, 2019.

[8] Matthew J Turk. How to scale a code in the human dimension. *arXiv preprint arXiv:1301.7064*, 2013.