Theme Article

The Role of Scientific Communities in Creating Reusable Software: Lessons From Geophysics

Louise H. Kellogg
University of California, Davis

Lorraine J. Hwang
University of California, Davis

Rene Gassmöller
University of California, Davis

Wolfgang Bangerth
Colorado State University

Timo Heister
University of Utah

***Abstract*—The domain of geophysics has historically been a driver of scientific software development due to the size, complexity, and societal importance of the research questions. Geophysical computation complements field observation, laboratory analysis, experiment, and theory. Specialized scientific software is regularly developed by geophysicists in collaboration with computational scientists and applied mathematicians; in this cross-disciplinary environment, reusability is critically important both to preserve the intellectual investment and to ensure the quality of the research and its replicability. The Computational Infrastructure for Geodynamics (CIG) is a "community of practice" that advances Earth science by developing and disseminating software for geophysics and related fields. We discuss CIG's best practices, lessons learned, and community practices, and highlight how development of high-quality, reusable scientific software has accelerated scientific discovery by enabling simulations of the dynamics of**

Digital Object Identifier 10.1109/MCSE.2018.2883326
Date of publication 3 December 2018; date of current version 8 March 2019.

Earth's surface and interior across a wide spectrum of problems using resources from laptops to leadership-class supercomputers.

Established in 2005 with funding from the National Science Foundation, the Computational Infrastructure for Geodynamics (CIG, geodynamics.org) is a partnership between the scientific domains of solid-earth science (the study of the crust, mantle, and core of the Earth and other terrestrial planets) and computational science. CIG's goal is to advance geophysics and related fields of research by developing and disseminating scientific software, using best practices from computational science. CIG grew out of the realization, common to many scientific communities, that a lot of science depends on software that was written for specific research projects by scientists with little formal background in software engineering. This software was handed from scientist to scientist who then continued to modify their own copies before handing them off once again to someone else. Although some scientists shared software and several successful benchmark projects investigated the accuracy and performance of different numerical methods (for example in dynamical modeling of mantle convection),[1–3] the scientific community did not fully benefit from the investments being made in scientific software development and did not have the capability to take full advantage of developments in numerical methods, applied mathematics, and software and hardware advances.

Over the last 13 years, CIG has grown into a "community of practice," identifying and encouraging use of those best practices that are most effective for this community, while also supporting development and dissemination of high quality, free, open-source scientific software for geophysics. CIG has worked with and supported authors of some of the popular and complex packages in the community in an effort to improve its development practices. CIG has also run many workshops and training sessions, teaching early career scientists in particular to work with and extend these software packages using the best practices we have learned. These training efforts prepare our scientific workforce to be expert users and how to contribute to scientific software.

As a result of these activities, software is developed very differently today in the geodynamics community. Software packages disseminated by CIG follow a set of best practices: they are available under an open-source license; are extensively documented; use version control; and have automated test suites. These practices have become the accepted standard in the community, with CIG maintaining a leadership role in the continuous development of software best practices in the geosciences.

BUILDING A COMMUNITY OF SOFTWARE DEVELOPERS

While a large percentage of Ph.D. students in the geosciences takes courses on quantitative and computational methods,[4] these courses generally do not focus on software development, software design, and software management. The geosciences are of course not alone in this challenge: The lack of formal software training is typical across the sciences and engineering.[5–8] The software development education self-reported by members of the CIG community reflects these experiences.[9]

Geophysicists typically come to computational geophysics with strong quantitative backgrounds in geology or physics; their education prepares them in the foundations of their science, but not the practical aspects of developing scientific software. This lack of formal training in software means that students and postdocs often learn software development practices from their advisors, collaborators, and mentors. Mentorship ("apprenticeship relationships"), peer learning, and in-person interactions at conferences and workshops are key to professional development in this community. Recognizing the important role of professional and peer networks, CIG's activities include providing interaction with computational and mathematical scientists, who bring background and skills in scientific software

development to the geophysics community that would otherwise have been missing. The connection to computational scientists allows geophysicists to learn from and hear about how other scientific communities develop and use scientific software. In turn, the real-world scientific questions posed by geodynamics research drive and provide a test-bed for the work of computational scientists and inform their interactions with communities in other scientific domains.

BEST PRACTICES IN USE

The software development community has long developed best practices that are widely known and used among professional software developers. However, until recently, these practices have rarely been used in domain-based scientific software due to a lack of education and differences between production and research software. In addition, the research environment did not recognize or reward the effort involved in developing sustainable software. This is a beginning to change, and CIG has consequently identified a simplified set of best practices in software and training (github.com/geodynamics/best_practices) that is readily accessible to user-developers and takes into account the community's software ecosystem. CIG provides a range from *minimum* to *target* goals for practices, which underpin sustainable and usable software, promote development in an open-source environment, and emphasize inclusive and welcoming developer communities. We next discuss the implementation of some of these practices.

Building on Existing Software

The scientific domain of geodynamics has been defined as the "application of continuum mechanics to geological problems"[10] and draws heavily from the general governing equations in continuum mechanics: mass, momentum, heat flow, and electromagnetics. Similarity of the physics between the disciplines allows us to take advantage of algorithmic and computational advances in other communities. In doing so, our community has learned that building on externally and professionally developed open-source libraries leads to sustainable high-quality software. For example, utilizing the many personal years of development that have gone into PETSc or Trilinos for parallel linear algebra or into deal.II for finite element methods has greatly accelerated the geodynamic software development, extending its abilities to tackle new classes of research questions. External libraries were not widely used in geodynamics before CIG was started, in part due to a reluctance to rely on external projects. This changed when early in its history, CIG participants made personal connections with many key developers of large software packages through workshops and advisory committees, leading, in some cases, to direct involvement by computational scientists in the development of geodynamics software.

Benchmarking

How will domain scientists know that using a particular code is not only useful for their research but will produce reliable results? Often a code gains a reputation within a close scientific community because its effectiveness and correctness are supported by scientific publications, and because it has been shown to reproduce community-established benchmarks.[11] Both peer-reviewed publications and benchmarks are consequently also the requirements for codes donated to and developed by CIG.

As an example, CIG conducted a community benchmark challenge as one step in the development of a next-generation code for modeling planetary dynamos. The scientific problem involves the study of solutions to the equations governing magnetohydrodynamics in a rotating spherical shell, in order to understand the origin and evolution of the magnetic field of the Earth and other planets.[12] This has long been recognized as both a scientific and computational challenge, due to the high resolution required to model real planetary systems.[13] In preparation for developing the next-generation dynamo code, CIG first held a workshop bringing together geodynamo and solar dynamo researchers. Their goal was to identify state-of-the-art methods and grand challenge problems, and to define a common benchmark for geodynamo model codes. In response to the benchmark invitation, 14 research groups worldwide

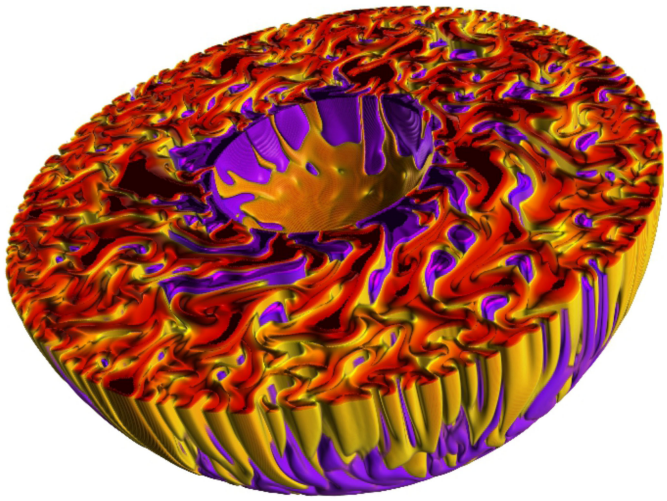

Figure 1. Simulation using the code Rayleigh showing temperature perturbations as realized in rotating convection in an earth-core geometry. Regions of warm flow are depicted in yellow, and regions of cool flow in violet. Run on NASA's Pleiades on about 4096 Sandy Bridge cores. This model uses Rayleigh 0.9.0[15,16] published under the GPL3 license. (Courtesy of Nick Featherstone, University of Colorado, Boulder).

submitted their codes for testing. Performance benchmarks were run by CIG staff using CIG's extreme science and engineering discovery environment allocation on the Texas Advanced Computing Center's Stampede supercomputer.[14] The workshop also outlined a long-term strategy leading to CIG support for developing a new community code, Rayleigh,[15] using the anelastic spherical harmonic method that had previously been developed for and is widely used in solar physics.[16]

The Rayleigh code was designed with insights from community discussions and benchmarks specifically targeting scalability and performance for leadership class computing. A start-up award at Argonne Leadership Computing Facility (ALCF)'s Mira machine, currently the world's seventh fastest computer, led to the definition of a small number of very large high-resolution models needed to make substantial progress on outstanding questions in Earth and planetary sciences. Assisted by the catalyst assigned to the team through the ALCF Innovative and Novel Computational Impact on Theory and Experiment program, the team was successful in their application for up to 493 million core hours over three years that have been used to simulate the creation of magnetic fields of Earth, Jupiter, and the Sun (see Figure 1).

Supporting Reuse and Quality = Supporting Developers

Because scientific software is developed in response to a specific scientific question, it is often developed by individuals who did not anticipate that their software would be used by others or reused in other projects. However, reuse of software avoids duplication of effort, saving time and money. Equally importantly, reusing existing well-tested code also improves software quality and, consequently, trust in the scientific results. The challenge lies in creating a community in which developers feel it is worth the effort needed to create well-designed, well-tested, documented, and reusable software, and users feel it is worth their effort to use.

Supporting an ecosystem of good scientific software for a community not only includes setting up a curated repository, but also establishing a mechanism to make the software findable. Furthermore, those that contributes to software need to receive credit for their efforts. For CIG's research-based core community, software is treated as a scholarly product. CIG hosts software landing pages for its software packages on geodynamics.org, maintains GitHub repositories, and a Zenodo community to increase the visibility of geodynamics software. Through integration with Zenodo, each software release is assigned a persistent identifier—a DOI. This is added to the attribution information for each software package. To assist researchers who are trying to determine what to cite when they use software in their own publications, CIG developed the *attribution builder for citation* ("abc," see geodynamics.org/cig/abc). The attribution builder provides the software citation, citations to appropriate primary and secondary publications about the software, and suggested language for both the body of the publication and the acknowledgment sections of a paper; we used it to construct the citations for the examples used in this paper.

Through these efforts, CIG has been improving the challenging task of computational reproducibility in scientific publications. The attribution builder simplifies the task for the user to cite the specific version of the code that was used. The

Zenodo releases point to an exact version of the software that can be used by a third party to reproduce the exact computations assuming the input files used have been made available by the researcher. The CIG best practices require "Citation of code version" and encourage developers to provide a way to archive a "workflow" for reproducibility. For instance, in the mantle convection software ASPECT,[17] a git repository includes input files, source code, and plotting scripts necessary to reproduce examples.

Supporting Users

Supporting users requires providing good documentation and training. CIG has been supporting and running software tutorials regularly, often colocated with conferences and other meetings. Key elements to successful software tutorials include discussions of the scientific background (the theory), its numerical implementation, software dependencies, and worked end-to-end examples. We have found that longer tutorials (one week) should incorporate "tinker time": scheduled blocks of several hours, during which participants are free to play with software, work on and modify existing cookbooks, try software for their own research problems, and ask questions of the instructors. This self-directed time allows researchers to delve further into their interests, while interacting with trainers and expert users. Training sessions typically begin with a round of brief introductions so that everyone knows which participants are interested in related problems. The immersive character of these tutorials has led to numerous new collaborations between participants that often began during tinker time.

A significant hurdle for participants is often just to get the software installed on the diverse hardware they bring to these tutorials. If the number of participants is small, this can be addressed by scheduling time early in or prior to the tutorial (via online help sessions) to troubleshoot software installation problems. However, this does not scale up to larger numbers of participants, and we have found that the usage of virtual machines and Docker images has vastly decreased installation issues, allowing tutorials to start using the software right away, on a system that looks identical on all participants' machines.

When HPC resources are required, we have installed software at a host facility or an allocation on a national computing resource. Although this sounds straightforward, in practice providing access to computing resources during a tutorial comes with difficulties, including batch systems that do not prioritize access for many jobs submitted at the same time, bandwidth limitations when moving large files in and out of limited access systems, and other real-world problems that are difficult to anticipate ahead of and resolve during a workshop.

An example of a successful HPC software tutorial is one CIG recently organized in collaboration with and at Lawrence Livermore National Laboratory's (LLNL) Livermore Valley Open Campus.[18] LLNL provided a meeting space with robust high-speed Internet access and dedicated resources. LLNL staff provided user account support, monitored queues and jobs in real time. Most importantly, LLNL provided access to 7200 dedicated cores of the Quartz HPC cluster for the last three days of the week-long workshop, enabling the 55 participants to simultaneously run simulations on 100s of processors. Participants successfully ran a simulation of the September 19, 2017 M7.1 Mexico earthquake that occurred during this workshop and were presenting the resulting research at conferences within six months of the tutorial.

The SW4 development team recently enhanced the code for efficient simulation of earthquake ground motions for hazard and risk analysis[19] under DOE's Exascale Computing Program. Figure 2 shows results from a recent study in which direct access to support staff operating the HPC systems being used was a key to success.[20]

Building Communities Through Hackathons

Most developers do not begin a software project with the intent of launching an open-source community. Rather, they develop the software for themselves or their research group. Thinking about expanding a project to be community software is a significant step in the life of the project and its developers. It involves a willingness by the original authors to be responsible for their

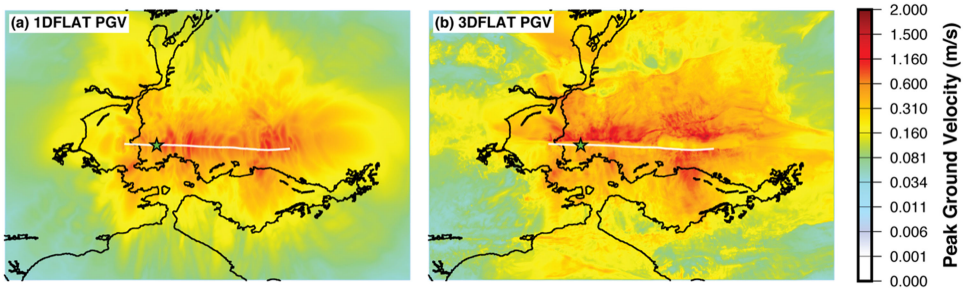

Figure 2. Access to HPC is essential to successfully compute many geophysics models. Here, the wave propagation code SW4 2.01[20] on LLNL's Quartz is used to simulate ground motions from a hypothetical large earthquake on the Hayward fault in the San Francisco Bay region.[21] Sufficient computing power is required to incorporate realistic frequencies and earth properties. The figures show peak ground velocity for (a) a simple plane-layered model with flat free surface and (b) a 3-D model with the effects of topography. The differences between these models can influence policy and business decisions for Northern California.

software, to give up some degree of control over it, and to move from a few personal interactions to larger Internet-based groups. Most importantly, it involves seeing the software's users as potential future developers that can be mentored to grow into roles supporting the project itself.

CIG makes use of and supports "hackathons" to build and bring these user, developer, and software manager communities together. In our practice, we think of hackathons as longer events associated with a single software project, in which participants focus on learning and extending the software. We have found that these hackathons provide an excellent structure for a community to discuss software and community development directions and practices, as well as to build personal connections that reduce the barrier to entry for new community members to contribute to the project.

As one example, CIG has run an annual hackathon since 2014 for current and would-be developers of one of its flagship projects, the ASPECT software for modeling global and regional scale deformation in the solid earth through geologic time (see aspect.geodynamics.org). These hackathons are typically 10 day long immersive events, in which 20 or more scientists convene at a remote location, living and working together. The focus is on improving ASPECT, and in the process, on building a community of expert user-developers who are able to use the code effectively and contribute to the development in the future. These hackathons have significantly contributed to both the code base as well the number and skill level of user-developers. Novice participants have risen to become code maintainers helping to reduce its "bus factor" (a measure of how dependent a project is on a small number of key developers.)

Successive hackathons have steadily increased the level of active contributors to the repository. Figure 3 shows both the history of code

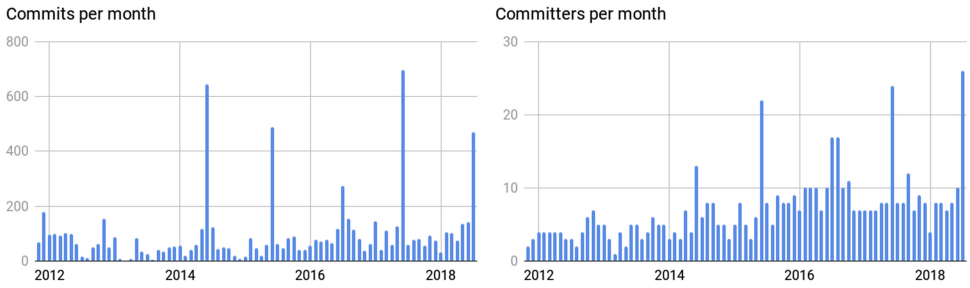

Figure 3. (Left) Number of commits per month to the GitHub repository of the ASPECT code. (Right) Number of people making contributions each month. Peaks reflect activity and new developers at CIG's annual 10-day hackathons, which started in 2014.

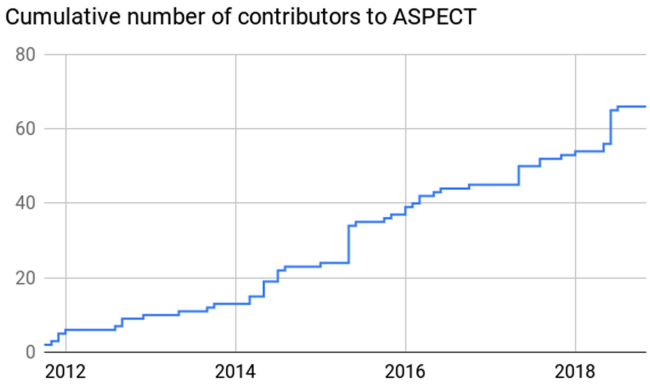

Figure 4. Cumulative number of people who have contributed to the ASPECT code.

development and the number of individuals who contribute code or documentation per month over time; Figure 4 shows the cumulative number of developers of ASPECT over time. The annual hackathons are clearly visible as contributing significantly to the growth of the code as well as to the growth of the developer community. All participants at the hackathons must generate at least one "pull request" to the GitHub repository, however small or inconsequential, to get them used to the process of contributing. In practice, most participants develop substantial contributions over the course of the hackathon, or in case of bigger projects, over the following weeks and months. Senior developers' role is thus to provide feedback, peer review of code contributions, and guidance during and after the hackathon to integrate the new user-developers into the community.

While we have found that hackathons are great tools to build communities of contributors, there are also challenges. For example, as seen from the right graph in Figure 3, at least half of the hackathon participants do not continue to contribute regularly (at least once a month) to the code base after the end of the hackathon, although many do continue to contribute more sporadically. On the other hand, the drop-off in the number of commits after the hackathon (left graph) must be expected given the intense, 10-day around-the-clock development sprint. Hackathons are challenged by the time limitations of senior developers, who often spend the entirety of the workshop on providing feedback in person or through code review; these individuals' time does not scale well to larger groups. We have found that for optimal efficiency, at least 20% of hackathon participants need to be experienced developers with a global overview of the code base to serve in these roles. Finally, hackathons lend themselves to extending the periphery of a code in many different directions (in the case of a modular code such as ASPECT through plugins) by participants. However, they are not well suited to addressing large-scale rewrites or software infrastructure needs since the developers with sufficient knowledge for these tasks do not have available time during the hackathon; such work therefore needs to take place between hackathons.

LESSONS LEARNED FROM A DECADE AND A HALF OF CIG

In the first stages of CIG, software was largely donated to the repository by their developers from the larger geodynamics community, often individuals or small teams of scientists who wrote software to address a compelling research problem. While such projects continue, we have seen a marked shift toward more collaborative and open development. As the community found its footing, its members have pursued the development of a new generation of codes whose roots range from established research codes to numerical libraries. Some of these were directly supported by CIG, with each of these codes forging different paths to wider adoption; in addition, codes continue to be donated by research groups who want help disseminating their work and bringing it up to CIG's standard best practices. As scientific software development is more recognized as scientific contribution on par with theoretical advances, experimental data, and

field observations, CIG's best practices have been invoked by geoscientists to develop data management plans for their research and teaching programs.

Based on our observation of the paths of the dozens of projects that CIG hosts (of which it currently directly supports the development of seven), we summarize some of the lessons learned from the CIG experience.

Successful Scientific Software Requires Scientific Champions

Although it may sound obvious, to succeed, scientific software must have an engaged group of scientists who have a compelling need for the capabilities it provides. It cannot be built in expectation that a user community will emerge. Codes under active development require attention to develop that community. This is one purpose of the hackathons, which both improve the code and create users who deeply understand it and are confident in their skills to contribute back. Building community is a major objective of workshops, tutorials, and webinars. An important CIG goal for all software under active development is that a software project be sustainable even when founding developers are no longer active project members.

Software can be Sunsetted

Software has a life history and lifetime for different reasons. Software that ceases development can suffer code rot as developers move on in their careers or interests, while other software will naturally be superseded as new methods or models are adopted, new physical capabilities are implemented, or the scientific questions evolve. The decision to retire a code can be simple when it can no longer be built on current platforms; more often, software simply endures a slow decline into disuse as users move on to other packages.

For codes whose development is actively supported, the decision to sunset a code is as complex as the codes themselves. Early in CIG's history, a project was initiated to develop a new code to model fundamental processes involved in the dynamics of Earth's tectonic plates. The computational requirements varied widely depending on the specific scientific question; hence, the resulting code was quite complex. Although it was used to solve important scientific problems, it proved overly challenging to use, and never found the dedicated user-developer champions needed for long-term sustainability. As a result, after several years, the decision was made to cease further active development. This decision relied on CIG's governance practices, and involved careful discussions among CIG's elected governing committees, staff, leadership, and the user community. To ensure the provenance of scientific work done with this code, stable versions remain available and CIG continues to run automated testing. Since then, several other codes both within and outside CIG have expanded capabilities to cover this area.

Focus on Building Diverse User and Developer Communities

In practice, most of the codes CIG hosts are neither entirely CIG-supported nor entirely donated; CIG-developed codes involve significant community contributions, while donated code generally requires some CIG contribution, including support of automated testing, assistance with documentation, and the like. Successful community-based software requires technical and scientific expertise, but equally requires significant attention and investment in the culture of the community involved. CIG's role in building self-sustaining communities is to provide examples of success and to build structures that enable such communities to form and thrive, for example, by running workshops, hackathons, and tutorials, hosting mailing lists, establishing wiki-based knowledge repositories, and promoting user-developer groups.

Provide Support and Credit for Code Development

Because producing high quality, reusable scientific software takes significant effort, providing mechanisms to credit that effort is essential to sustain software growth and usage. Very few scientists are fortunate enough to develop code full time; code development is more often pursued as a means to address a research problem. Researchers who develop code worry that by sharing code, their research may be "scooped" or worse, their code misused; they will not receive credit, lose control of their project, and/or be overwhelmed with requests for user

support and feature development. Tools like *abc* can help developers get credit, while CIG's community-building efforts and support can alleviate some of this burden.

CHALLENGES FOR THE FUTURE

Over nearly a decade and a half of existence, CIG has both witnessed and incubated significant changes in how the geodynamics community develops software. As the community moves toward diverse collaborative teams, more adoption of software best practices; and increasingly writes software that can be used for larger, more complex problems, what challenges and opportunities might the future hold?

Preparing the Next Generation of Scientist Developers

Software development strategies are still not part of typical graduate curricula in the sciences. CIG's workshops and training for scientists in the geoscience domain create an educational infrastructure that is not otherwise generally available and meets a real need. Software tutorials, available online, have become a steadily used resource by the community. The hackathons and best practices serve both an operational role—improving and advancing code—and an educational role—scientists learn from experts at the hackathons and from the best practices. By acting as a laboratory and clearinghouse, CIG is able to extend the reach of these projects; for example, CIG has begun holding hackathons for other projects in a style similar to ASPECT involving experienced hackathon leaders from the ASPECT team to share practical insights and assist with the training components of the hackathons. The result will be a workforce with expertise in their primary scientific domains and proficiency in code development.

Balancing Large and Small Projects

Organizational structures that work for a project of a few developers and a dozen users do not necessarily work for tens of developers and hundreds of users, a scale that is now reached by some of the CIG-supported projects. CIG therefore gathers feedback from its community and is experimenting with new methods to scale projects, including developing online tutorials and holding regular online user meetings that engage larger geographically dispersed communities. Our expectation is that these activities will also increase access to CIG activities by scientists who may find it difficult to travel to events. At the same time, it is critically important to be open to new, small, imaginative projects on the horizon.

International Collaborations

The scientific community of computational geophysicists is vast and global. Fostering international collaboration brings diverse approaches and practices to scientific problems, resulting in better research. Establishing and sustaining collaborations is challenging, due to the different ways that science is funded in different countries and differences in scientific culture. However, scientific collaboration across international boundaries has a long and successful history in the geosciences and will continue to benefit computational geophysics.

Computational Capacity

A great deal—perhaps the majority—of scientific research in geophysics is carried out using the small to mesoscale computer clusters available to most practicing scientists. Yet as our knowledge of the physical systems increases, scientific software increases in capability, new observations become available, and several scientific problems in geodynamics have grown to require leadership-class computation. The frontiers of computational geodynamics require identifying sufficient computational capacity to match the computational capabilities provided by the high-quality scientific software. Some of the requirements are specific to geophysics, while others are not: geodynamics models often need very high resolutions, and in some cases require very long duration computations to allow the model to progress through geologic time. This need is likely to prove to be a sustained challenge, even as hardware capabilities and software developments evolve.

Concluding Remarks

Each project and each user community are made up of individuals. What is usually cast as a

technical challenge—the development of sustainable, reusable, high-quality scientific software—is as much a social challenge as it is a technological one. What works for one project may not work for another because of the personalities of the principal developers and the communities they engage. We also need to consider the importance of retaining talent in our communities, and that involves ensuring that those who spend significant parts of their work life developing software for whole communities have career paths in research and academia.

ACKNOWLEDGMENTS

The Computational Infrastructure for Geodynamics is supported by the National Science Foundation under Award EAR-0949446 and Award EAR-1550901. The Attribution Builder for Citation Tool was developed under National Science Foundation under Award SMA-1448633.

REFERENCES

1. S. D. King, A. Raefsky, and B. H. Hager, "ConMan: Vectorizing a finite element code for incompressible two-dimensional convection in the Earth's mantle," *Phys. Earth Planetary Interiors*, vol. 59, pp. 195–207, 1990.
2. B. Blankenbach *et al.*, "A benchmark comparison for mantle convection codes," *Geophys. J. Int.*, vol. 98, pp. 23–38, 1989,
3. P. Van Keken, S. D. King, H. Schmeling, U. R. Christensen, D. Neumeister, and M. P. Doin, "A comparison of methods for the modeling of thermochemical convection," *J. Geophys. Res., Solid Earth*, vol. 102, no. B10, pp. 22477–22495, 1997.
4. C. Wilson, *Status of Recent Geoscience Graduates*. Alexandria, VA, USA: Amer. Geosciences Inst., 2014.
5. V. Basili *et al.*, "Understanding the high-performance-computing community: A software engineer's perspective," *IEEE Softw.*, vol. 25, no. 4, pp. 29–36, Jul./Aug. 2008, doi:10.1109/ms.2008.103.
6. J. Hannay, C. MacLeod, J. Singer, H. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?" in *Proc. ICSE Workshop Softw. Eng. Comput. Sci. Eng.*, 2009, pp. 1–8, doi:10.1109/secse.2009.5069155.
7. Z. Merali, "Computational science: ...Error... why scientific programming does not compute," *Nature*, vol. 467, no. 7317, pp. 775–777, 2010, doi:10.1038/467775a.
8. P. Prabhu *et al.*, "A survey of the practice of computational science," in *Proc. State of the Practice Rep.*, 2011, Paper 19, doi:10.1145/2063348.2063374.
9. L. J. Hwang, A. Fish, L. Soito, M. Smith, and L. H. Kellogg, "Software and the scientist: Coding and citation practices in geodynamics," *Earth Space Sci.*, vol. 4, pp. 670–680, 2017. [Online]. Available: https://doi.org/10.1002/2016EA000225
10. D. L. Turcotte and G. Schubert, *Geodynamics*, 1st ed. Hoboken, NJ, USA: Wiley, 1982, p. 464.
11. W. L. Oberkampf and C. J. Roy, *Verification and Validation in Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
12. G. A. Glatzmaier and P. H. Roberts, "A three-dimensional convective dynamo solution with rotating and finitely conducting inner core and mantle," *Phys. Earth Planetary Interiors*, vol. 91, no. 1–3, pp. 63–75, 1995, doi: 10.1016/0031-9201(95)03049-3.
13. National Research Council, *Getting Up to Speed: The Future of Supercomputing*. Washington, DC, USA: National Academies, 2005. [Online]. Available: https://doi.org/10.17226/11148
14. H. Matsui *et al.*, "Performance benchmarks for a next generation numerical dynamo model," *Geochem. Geophys. Geosyst.*, vol. 17, pp. 1586–1607, 2016, doi: 10.1002/2015GC006159.
15. N. Featherstone, "Rayleigh version 0.9.0 [software]," Computational Infrastructure in Geodynamics, 2018, doi: 10.5281/zenodo.1158290
16. N. A. Featherstone and B. W. Hindman, "The spectral amplitude of stellar convection and its scaling in the high-Rayleigh-number regime," *Astrophys. J.*, vol. 818, no. 1, p. 32, 2016, doi: 10.3847/0004-637X/818/1/32.
17. T. Heister, J. Dannberg, R. Gassmöller, and W. Bangerth, "High accuracy mantle convection simulation through modern numerical methods—II: Realistic models and problems," *Geophys. J. Int.*, vol. 210, no. 2, pp. 833–851, 2017, doi:10.1093/gji/ggx195.
18. A. J. Rodgers, L. J. Hwang, and L. H. Kellogg, "Computational seismology workshop trains early-career scientists," *EoS*, vol. 99, 2018, doi: 10.1029/2018EO090991.
19. H. Johansen, A. Rodgers, N. A. Petersson, D. McCallen, B. Sjogreen, and M. Miah, "Toward exascale earthquake ground motion simulations for

near-fault engineering analysis," *Comput. Sci. Eng.*, vol. 19, no. 5, pp. 27–37, 2017.
20. N. A. Petersson and B. Sjogreen, "SW4, version 2.01 [software]," Computational Infrastructure in Geodynamics, 2017, doi: 10.5281/zenodo.1063644.
21. A. J. Rodgers, A. Pitarka, N. A. Petersson, B. Sjögreen, and D. B. McCallen, "Broadband (0–4 Hz) ground motions for a magnitude 7.0 Hayward fault earthquake with three-dimensional structure and topography," *Geophys. Res. Lett.*, vol. 45, pp. 739–747, 2018. [Online]. Available: https://doi.org/10.1002/2017GL076505

Louise H. Kellogg received the Ph.D. degree in geological sciences from Cornell University, Ithaca, NY, USA, in 1988. She is a distinguished Professor of Earth and Planetary Sciences and the Director of the Computational Infrastructure for Geodynamics at the University of California, Davis. Her research interests include mantle convection, crustal deformation, and scientific visualization. Contact her at kellogg@ucdavis.edu.

Lorraine J. Hwang received the Ph.D. degree in seismology from the California Institute of Technology in 1990. She is the Associate Director with the Computational Infrastructure for Geodynamics, University of California, Davis. Her research interests include sustainable software and preservation of historical seismogram data. Contact her at ljhwang@ucdavis.edu.

Rene Gassmöller received the Ph.D. degree in geophysics from the University of Potsdam, Germany in 2015, in cooperation with the German Research Centre for Geosciences. He is a project scientist with the Computational Infrastructure for Geodynamics, University of California, Davis. His research focuses on the interaction between mantle convection and plate tectonic processes, numerical methods of geodynamic modeling, and sustainable software development in the earth sciences. Contact him at rgassmoeller@ucdavis.edu.

Wolfgang Bangerth is a Professor of Mathematics with Colorado State University, Fort Collins, CO, USA. He received the Ph.D. degree in mathematics from Heidelberg University, Germany, in 2002. He is one of the founders and a principal developer of both the deal.II (www.dealii.org) and ASPECT (aspect.geodynamics.org) projects, and is a computational scientist with interests in finite element methods, scientific software, and parallel computing. He has collaborated with geoscientists, biomedical imaging experts, nuclear engineers, physicists, and scientists from other areas. Contact him at bangerth@colostate.edu.

Timo Heister is an Assistant Professor of Mathematics with the University of Utah and the Scientific Computing and Imaging Institute and his research interests include numerical methods for PDEs, parallel finite element methods, efficient discretizations for fluid flow, and computational science in general. He received the Ph.D. degree in mathematics from the University of Goettingen, Germany, in 2011. Contact him at heister@sci.utah.edu.