FOUR WAYS TO GROW SCIENTIFIC-SOFTWARE

SOU-CHENG TERRYA CHOI, YUHAN DING, CLAUDE HALL JR., FRED J. HICKERNELL, AND ALEKSEI SOROKIN

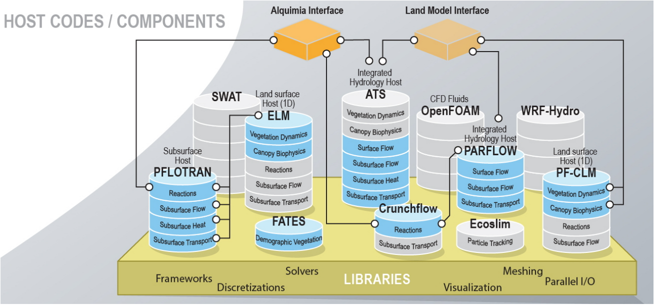
The opportunities for discovery, prediction, and optimization provided by advances in computer hardware and software are immense, but the challenges to taking full advantage of these opportunities are substantial. This position paper focuses on four challenges: i) connecting software libraries, ii) accurately estimating discretization errors from the computational data across multiple algorithms, iii) preparing developers and users to employ collections of libraries, and iv) educating developers and users to exploit large-scale computational environments.

1. Challenge

1.1. **Connectivity.** Developers wishing to contribute new (additions to) libraries and the practitioners wishing to utilize multiple libraries together find that basic mathematical objects may be encoded differently in different libraries. Must a function routine include its domain and the number of scalar inputs? Does an algorithm generating data sites provide a fixed number or an extensible sequence? What standards must developers of library enhancements meet so that their pull requests will succeed? The (quasi-)Monte Carlo and probabilistic numerics libraries that the authors know, such as [5, 13, 15, 16], lack adequate connectivity and consistency.

1.2. **Data-driven error estimation.** Complex computations may require the chaining of algorithms, e.g., the output of a (P)DE solver becomes the input of an optimizer. Since the next algorithm in the chain accepts inexact input from the previous algorithm, we need error bounds on algorithm output, as noted in [10]. A priori algorithm error bounds based on unknown norms of input functions are unhelpful. Reliable data-driven error bounds or estimates are needed. Moreover, error analysis for algorithms needs to factor in inexact inputs, and diagnostics should also be developed to ensure that necessary conditions for error bound validity are still met.

1.3. **Multiple Algorithms.** First computational mathematics courses [3] introduce students to numerical algorithms. Computational science courses [8, 14] combine numerical methods with modeling. Rarely are students taught to combine multiple algorithms to solve a problem. This leaves them unprepared for large-scale computational science, which requires not only multiple algorithms, but multiple libraries. The IDEAS Watershed project, whose codebase is depicted on the right [9], is an example.



1.4. **Large-scale computational environments.** Since the mid-1980s, computational scientists have become proficient at carrying out computations on their personal machines. However, many do not know how to take advantage of the variety of advanced architectures that are now common-place: GPUs, cloud computing, open science grid, and high-performance computing. Even with resources being made available through projects such as The COVID-19 High Performance Computing Consortium, many users and software library developers do not know how to take full advantage of them.

2. Opportunity

2.1. **Connectivity.** Code across multiple languages and platforms adheres to standards for certain data types, such as IEEE 754 for floating-point arithmetic. We propose that interested parties connected with larger scientific libraries develop shared standards for key objects that appear in algorithms, such as functions, sampling sites, and (partial) differential equations ((P)DEs). This would promote connectivity among software libraries and allow smaller libraries to be merged into larger ones. Developers could more easily contribute to existing libraries, and practitioners would find it easier to use multiple libraries.

2.2. **Data-driven error estimation.** Several of the authors and their collaborators have developed data-driven error bounds for multi-dimensional integration as well as univariate function approximation and optimization using numerical analysis and probabilistic numerics approaches [4, 6, 2, 1, 11, 12]. The key behind these error bounds is to identify reasonable cones of input functions that are not too spiky or peaky. This facilitates theoretical guarantees on the data-driven error bounds. The ideas referenced here could be extended to other algorithms such as differential equation solvers. There is also a need for the error bounds derived to allow for inexact input, which comes from chaining multiple algorithms together.

2.3. **Multiple Algorithms.** Computational science curricula need to be re-written to stress the combination of multiple algorithms, or even multiple libraries to solve problems, during introductory courses. To make room, we may need to eliminate detailed explanations of some numerical algorithms in the same way that we often avoid the details of the algorithm for computing the sine function. This curriculum revision may start by way of supplementary materials that will hopefully work their way into standard texts.

2.4. **Large-scale computational environments.** Educating the next generation to embrace cloud or cluster computing the same way that this generation views personal computing can be attained through partnerships involving academic institutions, national labs, and corporations offering test sites for students to learn and for developers to experiment. Software tools are also needed to help developers migrate their libraries from personal machines to environments taking advantage of multiple cores, GPUs, and other advanced architectures.

3. Timeliness

The Extreme-scale Scientific Software Development Kit (xSDK) is an effort to "address challenges in interoperability and sustainability of software developed by diverse groups at different institutions." The push to identify standards for objects such as functions and (P)DEs, would fit this program. The existence of a good number of large libraries makes it possible for healthy discussion among these communities to arrive at reasonable standards.

The rise in popularity of probabilistic numerics research provides one promising avenue for data-driven error bounds. So does the authors' work on adaptive algorithms.

The permeation of computing into multiple disciplines and the availability of multiple packages make it possible to bridge the gap between what is taught in the classroom and what is needed at the frontiers of research. Training grants may be used to develop and pilot programs that address what is lacking in our present curricula.

Just as large-scale computational environments banded together for COVID-19 research, they can do the same to provide basic competency to computational scientists in advanced architectures. This might look like Google Colaboratory but with access to more advanced architectures. Given the wider availability of these architectures, the time is ripe to provide developer tools to migrate libraries to exploit them.

References

[1] F. J. Hickernell and Ll. A. Jiménez Rugama. "Reliable Adaptive Cubature Using Digital Sequences". In: *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*. Ed. by R. Cools and D. Nuyens. Vol. 163. Springer Proceedings in Mathematics and Statistics. arXiv:1410.8615 [math.NA]. Springer-Verlag, Berlin, 2016, pp. 367–383.

[2] F. J. Hickernell et al. "Guaranteed Conservative Fixed Width Confidence Intervals Via Monte Carlo Sampling". In: *Monte Carlo and Quasi-Monte Carlo Methods 2012*. Ed. by J. Dick et al. Vol. 65. Springer Proceedings in Mathematics and Statistics. Springer-Verlag, Berlin, 2013, pp. 105–128. doi: 10.1007/978-3-642-41095-6.

[3] R. L. Burden, J. D. Faires, and A. M. Burden. *Numerical Analysis*. Tenth. Brooks/Cole, 2016.

[4] S.-C. T. Choi et al. "Local Adaption for Approximation and Minimization of Univariate Functions". In: *J. Complexity* 40 (2017), pp. 17–33. doi: 10.1016/j.jco.2016.11.005.

[5] S.-C. T. Choi et al. *QMCPy: A quasi-Monte Carlo Python Library*. 2020. doi: 10.5281/zenodo.3964489. url: https://qmcsoftware.github.io/QMCSoftware/.

[6] N. Clancy et al. "The Cost of Deterministic, Adaptive, Automatic Algorithms: Cones, Not Balls". In: *J. Complexity* 30 (2014), pp. 21–45. doi: 10.1016/j.jco.2013.09.002.

[7] R. Cools and D. Nuyens, eds. *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*. Vol. 163. Springer Proceedings in Mathematics and Statistics. Springer-Verlag, Berlin, 2016.

[8] A. Holder and J. Eichholz. *An Introduction to Computational Science*. Springer, 2019.

[9] *IDEAS Watersheds Code Diagram*. 2021. url: https://ess.science.energy.gov/wp-content/uploads/2021/01/IDEAS-Watersheds-software-ecosystem-road.jpg.

[10] I. Ipsen. *BayesCG: A Probabilistic Numeric Linear Solver*. Presentation at Dagstuhl Seminar on Proabilistic Numerical Methods – From Theory to Implementation. 2021.

[11] R. Jagadeeswaran and F. J. Hickernell. "Fast Automatic Bayesian Cubature Using Lattice Sampling". In: *Stat. Comput.* 29 (2019), pp. 1215–1229. doi: 10.1007/s11222-019-09895-9.

[12] Ll. A. Jiménez Rugama and F. J. Hickernell. "Adaptive Multidimensional Integration Based on Rank-1 Lattices". In: *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*. Ed. by R. Cools and D. Nuyens. Vol. 163. Springer Proceedings in Mathematics and Statistics. arXiv:1411.1966. Springer-Verlag, Berlin, 2016, pp. 407–422.

[13] P. Roy. *SciPy Quasi-Monte Carlo submodule*. 2021. url: https://docs.scipy.org/doc/scipy/reference/stats.qmc.html.

[14] A. B. Shiflet and G. W. Shiflet. *Introduction to Computational Science*. Princeton University Press, 2014. isbn: 9780691160719.

[15] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual, version 2.28*. 2021. url: http://mc-stan.org.

[16] J. Wenger, N. Krämer, and N. Bosch. *ProbNum*. 2021. url: https://github.com/probabilistic-numerics/probnum.

(Choi, Ding, Hickernell, Sorokin) Department of Applied Mathematics, RE 220, 10 W. 32nd St., Chicago, IL 60616
Email address: schoi32@iit.edu, yding2@iit.edu, hickernell@iit.edu, asorokin@hawk.iit.edu

(Hall) Birmingham Southern College, 900 Arkadelpha Rd, Birmingham, AL, 35254
Email address: cdhall1@bsc.edu