Barriers to Entry in Physics Computing: A Snowmass Letter of Interest for the Computational Frontier

R. Schmitz[1], T. Eichlersmith[2], and A. Huebl[3]

[1]University of California, Santa Barbara, California 93106 USA
[2]University of Minnesota, Twin Cities, Minnesota 55455 USA
[3]Lawrence Berkeley National Laboratory, Berkeley, California 94720 USA

August 2020

1 Introduction

Nearly every field in contemporary physics relies heavily on computation, playing key roles in theoretical calculations, experimental design, and data analysis. Therefore, the future success of the field will depend strongly on how accessible physics computing is to new researchers entering the physics community. However, computational physics has historically had a high knowledge barrier to entry, resulting in a disparate set of computing practices and frameworks that marginalizes underrepresented minorities (URMs), undermines adoption of new software, and inhibits transference of knowledge. Solving this issue will require a systematic effort from developers across physics disciplines.

2 Barrier to Entry

There is a culture in physics that computational skills are to be acquired by the students independently, rather than in formal coursework. A recent study on instruction in undergraduate physics programs concluded that computational instruction is consistently taught in introductory courses just 24% of the time [1]. This computational instruction deficit is especially harsh for students with no formal computational training prior to undergraduate coursework, a trait that is disproportionately shared by URMs and students from poor socioeconomic backgrounds [2]. Further, within the context of research, knowledge of computational frameworks often builds on a series of specialized prerequisites and is distributed frequently through local mentor-ships. This provides another structural bias against students with no formal programming background, leaving access to physics only available to students who already have the knowledge necessary to gain these mentorships.

3 Why Focus on Barriers to Entry?

The ultimate goal of computational physics is to research interesting questions and intriguing physics. Programming languages, implementations and software packages are the essentials on which we build our models and analysis workflows. However, this software requires a user- and developer-base who have a strong foundational knowledge of the underlying language and structures. Therefore, the best way to do better physics is to improve the expertise, diversity, and size of the user-base entering the computational physics community. Additionally, by focusing on practices which remove the systematic disadvantages for potential users with little pre-existing background, we have direct access to improving URM representation in computational physics.

4 Methods of Improvement

The essential improvement to computational frameworks should come from compartmentalization of resources. Reproducible, quickly deployable workflows can decrease the start-up costs of installing/configuring dependencies on a system. Good examples include user-level package managers — such as conda, Spack.io, pip – or something broader such as containers [4]. More generally, workflows should be targeted at the new user; veteran users are skilled enough to adapt to a workflow that helps on-board new users. Second, all developer-user interactions should be public and searchable for the reference of current and future users alike [3]. Some frameworks log their interactions in public forums, but many use private forms of communication such as email lists or slack channels for which questions are closed to new users. We should emphasize the automatic documentation, transparency, and accountability that arises from public user-developer interactions that occur on sites such as StackOverflow, ROOT forums, or even HyperNews threads. A regularly updated encyclopedia/FAQ can serve as both a useful resource and to make access to a new framework less intimidating. This can be true for both users and potential contributors. Finally, each framework should create a tutorial, which is informative and easily accessible. Such a tutorial which answers not only the questions of "how" but also "why" helps inform the user about the software and the system they are using. In addition to increasing both the accessibility of the software and the user-base, a good tutorial can also serve as a development resource as it can reveal holes in the software itself. All these factors — an accessible workflow with manageable dependencies; a public database of developer interactions and frequent questions; and a strong, accessible tutorial — are crucial to removing the barrier to entry for users and contributors alike.

References

[1] Marcos D. Caballero and Laura Merner. Prevalence and nature of computational instruction in undergraduate physics programs across the united states. *Phys. Rev. Phys. Educ. Res.*, 14:020129, Dec 2018.

[2] Code.org Advocacy Coalition. 2019 state of computer science education: Equity and diversity. *advocacy.code.org*.

[3] Rémi Lehe, Axel Huebl, and Jean-Luc Vay. Embracing modern software tools and user-friendly practices, when distributing scientific codes. *Snowmass21 LOI*, 2020.

[4] Microsoft. Containers as the foundation for devops collaboration. *docs.microsoft.com*.