





# Revisiting Computation for Research: Practices and Trends

Jeremiah Giordani<sup>\*</sup> ,

Ziyang Xu<sup>\*</sup> ,

Ella Colby , August Ning ,

Bhargav Reddy Godala ,

Ishita Chaturvedi ,

Shaowei Zhu , Yebin Chon ,


Greg Chan , Zujun Tan 

*Computer Science Department*

*Princeton University*

Princeton, NJ, USA

Galen Collier ,


Jonathan D. Halverson 


*Research Computing*

*Princeton University*

Princeton, NJ, USA

Enrico Armenio Deiana ,

Jasper Liang ,

Federico Sossai ,

Yian Su , Atmn Patel ,

Bangyen Pham ,

Nathan Greiner ,

Simone Campanoni 

*Computer Science Department*

*Northwestern University*

Evanston, IL, USA

David I. August 

*Computer Science Department*

*Princeton University*

Princeton, NJ, USA

**Abstract**—In the field of computational science, effectively supporting researchers necessitates a deep understanding of how they utilize computational resources. Building upon a decade-old survey that explored the practices and challenges of research computation, this study aims to bridge the understanding gap between providers of computational resources and researchers who rely on them. This study revisits key survey questions and gathers feedback on open-ended topics from over a hundred interviews. Quantitative analyses of present and past results illuminate the landscape of research computation. Qualitative analyses, including careful use of large language models, highlight trends and challenges with concrete evidence. Given the rapid evolution of computational science, this paper offers a *toolkit* with methodologies and insights to simplify future research and ensure ongoing examination of the landscape. This study, with its findings and toolkit, guides enhancements to computational systems, deepens understanding of user needs, and streamlines reassessment of the computational landscape.

## I. INTRODUCTION

Computation has become increasingly essential in various research disciplines. It enables researchers to simulate complex phenomena, analyze datasets, and develop new theories. However, the true potential of computational research can only be harnessed through a seamless collaboration between researchers and the providers of computational resources. This symbiosis requires not only the continuous advancement of computational technologies but also a deep, nuanced understanding of the researchers' evolving needs and challenges. A survey of scientists from various disciplines who use computation in their research was conducted in 2011 at a doctoral-granting institution [1]. The survey provides insights into scientists' needs, beliefs, and challenges regarding computation and highlights the importance of expanding computational accessibility in research. It has been more than a decade since its publication, and there have been significant improvements

in computing hardware and software. This study aims to re-examine the state of computation in research, offering fresh insights into the current state of computational science and laying the groundwork for future enhancements in research support systems.

The study starts with a survey targeting researchers from the same institution as in the previous study, Princeton University. The survey participants are drawn from two sources: randomly sampled researchers and top computation users. The interviews were conducted via online or in-person meetings, featuring diverse open-ended and multiple-choice questions on topics such as research projects, computation experience, and desired improvements. The results are compared to the previous study to highlight trends that have surfaced over the past years.

The survey results uncover three categories of computational analysis. First, it reveals the computation landscape at the target institution including computation usage by disciplines, computation hardware, and common software stack including a list of computation-intensive software and their typical running time and environment. These results help us focus on the important research needs and the typical use patterns. Second, the study investigates the computation usage experience of the researchers, such as the programming language choices and trends, time spent on programming, debugging, and actively waiting for runs. These results help us understand the researchers' interaction with the computation. Finally, to understand the needs of researchers more directly, this study explores how hypothetical improvements in computational power could impact their research, along with their desired enhancements to their development environment or broader support from the computer science community.

Upon examining our findings, it becomes evident that, while computational resources and practices have evolved significantly, there remains a substantial gap between available technologies and their effective utilization by researchers. With an analysis framework that includes careful usage of large

<sup>\*</sup>Equal contribution.

language models, we identified four major trends and five challenges. The trends include more time spent in programming, more diverse needs for computation, programming language shift, and increased machine learning usage. The challenges include making performance more accessible, making development tools more user-friendly, managing duplicated and legacy projects, supporting fast-evolving machine learning applications, and providing adequate and diverse training. Focusing on these challenges can help us unlock the full potential of computational research across disciplines, ultimately fostering innovation and expanding the boundaries of scientific inquiry.

During the study, we went through many hurdles from initiating and conducting the surveys to data collection and analysis and realized that documenting and sharing the methodology of the survey itself can be useful. In this paper, we present a “toolkit” designed to assist researchers in navigating the complexities throughout the process of a survey study like this. It provides guidelines for survey execution and our reflective insights. It offers a strategic framework for researchers to conduct future surveys that can extend to multiple institutions and compare them with past studies as this paper does.

The contributions of this paper are:

- Conducting a comprehensive survey on researchers’ engagement with computation, encompassing 106 interviews;
- Presenting the results and analyses, including identified trends and challenges;
- Reflecting on the survey study process and providing a “toolkit” for conducting similar research in the future.

## II. BACKGROUND

The landscape of computational science is shaped by rapid technological advancements and the evolving needs of the research community. A decade ago, a survey provided a snapshot of this dynamic field, capturing the practices, preferences, and challenges faced by computational users. While this prior study offers a baseline, recent developments necessitate a revisiting of the landscape of computation for research.

### A. *Prior Study*

The present study builds upon a survey conducted more than a decade ago that focused on very similar questions regarding the practice of computational science. It was carried out at the same doctoral-granting research institution that we investigated for this survey, Princeton University [1]. The prior study aimed to understand how scientists coped with the growing computing demands and to uncover prevalent programming practices, the importance of computational power, performance-enhancing strategies, and the computational environment within the institution. The survey included 58 randomly selected researchers from diverse fields, including natural sciences, engineering, interdisciplinary sciences, and social sciences. Participants were interviewed and asked about various aspects of scientific computing related to their research.

The results of the prior study revealed that programming systems and tools available at the time did not fully meet the needs of computational scientists. Researchers were found to invest significant time and effort in programming, and while they understood the importance of scientific computing, they were generally unsatisfied with the execution time of their programs. It was identified that improvements in performance would not only enhance the accuracy and scale of experiments but also enable fundamentally new research avenues. Several areas of improvement were identified in the study, including the need for non-numerical algorithms, better utilization of computational resources, and shared-memory parallelization techniques. Additionally, the study found that scientists often did not leverage performance analysis tools to identify and address performance bottlenecks in their code [1].

By revisiting the key questions from the prior research and expanding to unexplored areas, the current study aims to provide insights into the evolution of computational practices, identify emerging trends, and underscore potential areas for future advancements in computational science.

### B. *Recent Context*

Since the publication of the previous study in 2011, the field of computational science has seen significant developments. There has been a notable increase in the adoption of cloud computing and high-performance computing (HPC) clusters by scientific communities. Studies have shown that cloud computing can provide significant benefits in terms of scalability, flexibility, and cost-effectiveness for scientific applications [2]. In addition, HPC clusters have become more accessible to scientists due to advances in cluster management tools and increased availability of resources [3]. Recent work introduces the Research Computing and Data Capabilities Model to evaluate the availability of research computing resources. A community report on this is released annually [4]. With many universities participating in this effort, there are abundant data on how the computation resources are provided to the researchers.

Despite these developments, there is still a gap between the needs of scientific researchers and the tools and technologies available to them. It is necessary to understand how researchers interact with computation and their specific needs, a goal this study aims to address.

The advent of Large Language Models (LLMs) has significantly streamlined survey and interview-based research, enabling small teams to efficiently analyze vast transcript datasets. These models excel at deep text comprehension, allowing for the swift identification of themes and insights. This significantly reduces the time and resources previously required for such analyses. These advancements not only speed up qualitative analysis but also expand research possibilities by accommodating larger sample sizes and more complex questions, enabling more comprehensive studies (such as this current one).

### III. SURVEY METHODOLOGY

This section outlines the methodology employed in the design and execution of the survey. The survey methodology involved a combination of open-ended and multiple-choice questions, aiming to delve into researchers’ computational environments, their programming preferences, and the impact of computational capabilities on their work. It details the structured approach taken to select participants, categorizing them into two distinct groups to capture a broad spectrum of computational practices.

#### A. Survey Questions

The survey questionnaire was designed to cover three main categories including computational backgrounds, current research projects, and researchers’ needs and expectations. The design of the survey questionnaire was informed by a multifaceted approach, integrating questions from previous studies to ensure continuity in tracking the evolution of computational practices, alongside new inquiries aimed at clarifying emerging trends and addressing gaps identified in earlier research. A detailed breakdown of the survey questions is presented in Table I.

TABLE I: Summary of survey questions

Questions	Topic
1.1-1.2	Demographic information and research goals
2.1-2.11	Programming background and preferences
3.1-3.14	Description of research project and its software stack
4.1-4.13	Computation performance and use of parallelism
5.1-5.7	Programming expertise, IDE usage, and debugging techniques
6.1-6.3	Use of machine learning in research projects
7.1-7.6	Impact of computation speedup, learning time, preferred languages, desired features, and feedback

#### B. Candidate Selection

This survey was conducted with two groups of participants, each selected for a different purpose.

##### a) Group 1: Randomly selected from the institution:

To select participants for Group 1, we compiled a list of graduate students across all departments. A subsequent stratified random sampling approach was used to ensure that the sample was representative of the student population across departments. To contact the participants, we employed a multi-pronged approach. We started with cold emails, asking subjects to agree to an interview, and then followed up on non-responsive subjects multiple times. Overall, the sampling approach and outreach methods employed aimed to minimize bias and ensure a representative sample of graduate researchers at the institution.

##### b) Group 2: Top-computation users from the institution:

To select participants from Group 2, we obtained a list of the most frequent computer cluster users from the administrators of the institution’s cluster computers. We then contacted these individuals through cold emails and through personal contacts, inviting them to participate in an interview. Including this group aimed to gain insights from individuals who were particularly experienced with computational resources and could provide a unique perspective on the use of computation in academic research.

The breakdown of interviewees by department is listed in Table II. Overall, the candidate selection and outreach methods employed aimed to ensure a representative sample of graduate students and experienced computational researchers.

TABLE II: Subject Population Distribution (G1: “Group 1: Randomly selected”, G2: “Group 2: Top-computation users”)

Field	Topic	Count		
		G1	G2	All
Engineering	Chemical & Biological	2	5	7
	Civil & Environmental	3	1	4
	Computer Science	5	3	8
	Electrical & Computer	8	2	10
	Mechanical & Aerospace	0	7	7
	Operations Research & Financial	4	1	5
Natural Sciences	Atmospheric & Oceanic	1	5	6
	Astrophysics	2	4	6
	Chemistry	6	6	12
	Ecology & Evolutionary Biology	1	2	3
	Geosciences	1	3	4
	Mathematics	2	0	2
	Molecular Biology	0	1	1
	Neuroscience	1	0	1
	Plasma Physics Lab	2	6	8
	Physics	0	2	2
Quantitative Biology	2	0	2	
Social Studies & Humanities	Economics	6	0	6
	Public & International Affairs	4	0	4
	Other Social Studies	4	0	4
	Other Humanities	4	0	4
Total		58	48	106

#### C. Data Collection

Interviews were conducted via virtual meetings or in-person sessions. Each interview lasted between ten minutes to one hour, depending on how many questions were relevant to the interviewee. Interviewers recorded the meetings while asking the survey questions. Research assistants then reviewed the interview recordings and manually completed a formatted questionnaire. The questionnaire aimed to elicit usable, concise, and comparable data. The survey adhered to the data security guidelines of the Institutional Review Board (IRB). To ensure consistency in the data collected, interviewers were trained on how to conduct the interviews, and research assistants were instructed on how to record responses.

#### D. Data Analysis

The data analysis was conducted in two main phases. The first phase focused on quantitative analysis, where data from

multiple-choice questions were plotted and compared with results from the previous study. This comparison allowed us to quantify changes in computational science practices over the past decade, such as shifts in programming language usage, attitudes toward performance optimization, and computational resource utilization. The second phase was qualitative, using an analysis framework (detailed in Section VII-A) to extract actionable insights. This dual approach, combining quantitative and qualitative methods, enabled a comprehensive synthesis of our findings.

#### E. Survey Timeframe

Interviews were primarily conducted from late 2022 to early 2023. Analysis took place from early 2023 to early 2024. Most of the quantitative insights were aggregated and analyzed in early 2023, while qualitative analysis continued until early 2024.

### IV. RESEARCH COMPUTING LANDSCAPE

The first part of the results focuses on the landscape of research computing including the percentage of researchers who use computation in their research, referred to as the “computation usage rate.” It also explores their hardware and software choices.

#### A. Computation Usage Rate

To show the extent of computational usage across disciplines, we analyzed randomly sampled candidates, grouped departments into three broad categories, and determined the proportion of respondents who utilized computation in their research. Not surprisingly, the usage of computation is high across disciplines. Overall, 75% of all researchers use computation in their research. In natural sciences and engineering, almost all researchers use computation methods (98% and 96%, respectively). In social studies and humanities, 42% use computation regularly in their research.

#### B. Computation Hardware Choices

The percentage of scientists using different computing hardware compared to the prior study is presented in Figure 1. As the results indicate, fewer researchers run their code on desktops now, shifting to running code on servers or clusters. GPUs are becoming an emerging platform for computing. A third of all researchers we interviewed are using GPUs to speed up computation. Some researchers are also looking into FPGA as an alternative approach.

#### C. Research Software Stack

Researchers employ a diverse array of applications, software, and libraries to support their work across various fields. Among these tools, simulation software such as Athena++ [5], LAMMPS [10], and COMSOL [25] plays a significant role, as it helps model complex physical systems. Optimization tools like Gurobi [21] and Mosek [22] and solvers such as Z3 [26] and Sundials [27], [28] are also crucial in handling computationally intensive tasks. Verification software, including Seahorn [29] and NV [30], is used to reason about

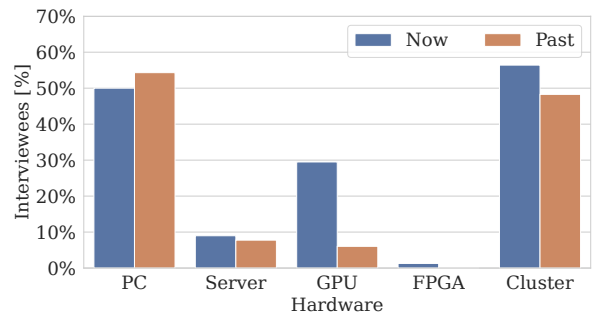


Fig. 1: The percentage of different computing hardware used by scientists, current study (Now) vs previous study (Past).

correctness properties of programs and network protocols. In the realm of machine learning, frameworks like PyTorch [23], TensorFlow [24], and Transformers [31] have gained popularity, while OpenVino [32] serves as an efficient inference toolkit. Libraries such as FFTW [33] and LAPACK [34] facilitate numerical computations. Some numerical computation libraries like PETSc [35] are more specialized to certain areas. HDF5 [36] and Blosc [37] that handle file I/O and compression tasks are also widely used to improve performance. For data analysis, researchers rely on packages like FastQC [38], SAMtools [39], and various R and Python packages like EdgeR [40]–[42], WGCNA [43], [44], and pandas [45]. Image processing and computer vision tools, such as ImageJ [46], mica [47], and OpenCV [48], also play a key role in various research areas.

Table III focuses on computation-intensive applications utilized by the top computational users, as identified in the survey. These diverse applications focus on simulation, optimization, and machine learning.

### V. COMPUTATION USAGE EXPERIENCE

This section focuses on the researchers’ experience with different steps of using computation, from programming to debugging and optimizations.

#### A. Programming Language Choice

By examining the data from both randomly sampled researchers, the top computation users, and from the prior study, we can identify key trends and their potential implications for the future of computational research. Table IV offers a breakdown of language usage.

Python has emerged as the dominant programming language, experiencing a rapid increase in adoption across research disciplines, departments, and experience levels. This surge in popularity can be attributed to Python’s versatility, ease of use, and rich ecosystem of libraries and tools, making it an attractive choice for researchers from various fields. C/C++, Fortran, and MATLAB, once popular choices among researchers, have experienced a significant decline in usage in contrast to the growing popularity of Python and R. The results also show that Fortran and C/C++ maintain a

TABLE III: Computation-intensive applications

Application	Type	Field	Open Source	Typical Runtime	Execution Environment
Athena++ [5]	Simulation	Astrophysics	Y	1 week - 1 month	PC, Cluster, Supercomputer
REBOUND [6]	Simulation	Astrophysics	Y	Hours - days	Cluster
GFDL Models [7]	Simulation	Climate	Y	1 month	Cluster, Supercomputer
Parflow [8]	Simulation	Hydrology	Y	Hours	PC, Supercomputer
Specfem3D globe [9]	Simulation	Seismology	Y	Hours - weeks	Group Server, Cluster
LAMMPS [10]	Simulation	Molecular Dynamics	Y	1 day - months	PC, Cluster
MEEP [11]	Simulation	Electromagnetics	Y	1 month	Cluster
hfss [12]	Simulation	Electromagnetics	N	Hours - days	PC, Personal Server, Cluster
ORCA [13]	Simulation	Quantum Chemistry	N	1 month	PC, Cluster
GAMESS [14]	Simulation	Quantum Chemistry	Upon request	Hours - 1 week	PC, Cluster
DMOL3 [15]	Simulation	Quantum Chemistry	N	Weeks - 2.5 months	PC, Cluster
ONETEP [16]	Simulation	Quantum Chemistry	N	Weeks - 2.5 months	PC, Cluster
VASP [17]	Simulation	Quantum Chemistry	N	Weeks - 2.5 months	PC, Cluster
Tristan v2 [18]	Simulation	Plasma Physics	Y	1 day	Cluster
ISAT [19]	Simulation	Combustion	Y	Several days	Cluster
NGA [20]	Simulation	Computational Fluid Dynamics	Upon request	Several days	Cluster
Gurobi [21]	Optimization	Operations Research	N	Hours	PC, Cluster
MOSEK [22]	Optimization	Operations Research	N	Hours	Cluster
PyTorch [23]	Machine Learning	Multiple Fields	Y	Hours to weeks	Cluster
TensorFlow [24]	Machine Learning	Multiple Fields	Y	Hours to weeks	Cluster

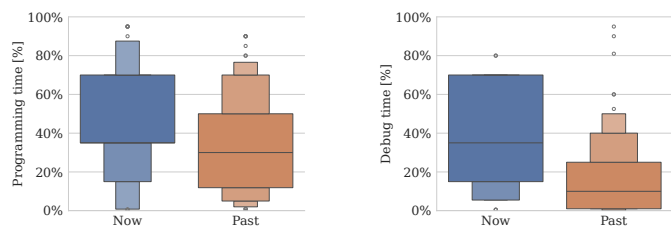
TABLE IV: Programming Language Usage Breakdown

Language	Randomly Sampled	Top Computation Users	Prior Study
Python	69%	83%	26%
R	40%	29%	15%
MATLAB	24%	15%	54%
C/C++	15%	46%	56%
Fortran	4%	27%	27%
Others	35%	21%	20%

strong presence among top computational users, despite Fortran’s near abandonment and C/C++’s sharp decrease among randomly sampled researchers. A similar proportion of top computational users code in C/C++ and Fortran as a decade ago.

### B. Time Spent on Programming and Debugging

Figure 2a shows, unsurprisingly, that programming has taken up an even larger portion of research time overall, compared to the past survey responses, where researchers had already been spending about 35% of their time programming on average. Data on the percentage of time spent in programming is more concentrated than before. Figure 2b suggests that respondents are generally spending a larger portion of time debugging than before. We take a closer look at how researchers debug in Section V-D.



(a) Percentage of research time spent in programming, current study (Now) vs previous study (Past).

(b) Percentage of programming time spent in debugging, current study (Now) vs previous study (Past).

Fig. 2: Time spent on programming and debugging.

### C. Research Time Actively Waiting for Runs

We categorize research time spent actively waiting for runs when the computation tasks are on the critical path of the research. The survey found that this waiting time varied considerably among participants, as shown in Figure 3. While about 40% of respondents indicated that less than 5% of their research time was spent waiting for runs, more than 25% of scientists reported spending more than 20% of their research time waiting. We cannot perform a direct comparison with the previous survey data because the question was formulated in a different way, yet we can still conclude that waiting time remains a problem for many scientists. Only about half of the interviewees optimize performance of their code even though

active waiting has taken up more than 10% of their research time and the benefits from optimizing might be non-trivial. A closer look into the optimization techniques is available in Section V-E.

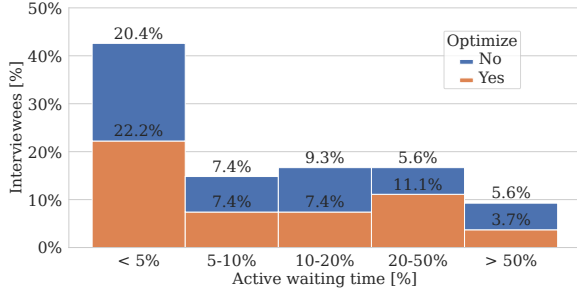


Fig. 3: Research time actively waiting for runs, with the breakdown of whether optimizing for performance.

53 interviewees discussed their understanding of why their computation is blocked. 36 interviewees (68%) attribute the cause to computation speed; 21 (40%) to memory, either memory IO or space; 3 (6%) to storage, either lack of storage space or IO; 4 (8%) to the lack of resources on the cluster; and 1 (2%) to other issues, in this case, lack of license.

#### D. Debugging Methods

Debugging is an essential process in software development that involves identifying and fixing errors in the code. Out of 46 respondents who described their debugging methods, the most popular technique was using printing statement (50%), where developers display variables’ values and track the program’s states. Breakpoints (35%) came in second, allowing developers to pause code execution at specific points for examination. Code inspection (26%) was the third most popular method, which involves manually reviewing the code to identify bugs. Debugger tools (15%) ranked fourth and are used to step through the code, inspect variables, and monitor execution. Lastly, Other techniques (11%) included unit tests or asking other people for help. Compared to the last survey, the use of debugger tools has dropped significantly, while printing and code inspection has gained much popularity. This shift could be attributed to the increased use of Python, a dynamically-typed language, as opposed to other statically compiled programming languages.

#### E. Optimizations

Our study revealed that 37% of respondents reported spending time optimizing their programs, which is less than the prior study (52%). 26 interviewees shared the methods they use to optimize their program. Out of the 26 respondents, algorithmic changes were employed by 4 individuals (15%), while data structure optimizations were used by 5 individuals (19%). Specialized libraries were favored by another 4 participants (15%), whereas loop optimizations had 7 participants (27%) opting for the method. Compiler flags were employed by 2 participants (8%). Lastly, 7 individuals (27%) relied on

configuring their runs to optimize performance, such as the number of nodes used when running one experiment. The distribution is similar to the prior study.

#### F. Parallelism

In our study, 48% of the randomly sampled respondents reported using some form of parallelism in their research, and 92% of top computation respondents use parallelism. The detailed comparison between these two categories and the past interview is presented in Table V. This stark contrast in parallelism usage between randomly-sampled respondents and top computation users underscores the importance of parallel computing in handling computationally intensive research. The widespread adoption among high-end users highlights parallelism’s significance in advancing research efficiency and tackling complex computational challenges.

TABLE V: The comparison of the breakdown of parallelism among the prior survey, the randomly-sampled candidates, and the top-computation candidates of this survey.

Parallelism	Random	Top Computation	Past
Job	28%	60%	51%
Message passing	10%	54%	21%
Threading	10%	48%	7%
GPU based parallelism	28%	42%	9%
Others	5%	6%	14%
None	52%	8%	31%

## VI. PERFORMANCE SATISFACTION AND NEEDS

This section encapsulates direct feedback from researchers on their satisfaction levels with current computational performance and expressing their performance needs for future research.

#### A. Performance Satisfaction

From the interviews, we found that 73% of researchers were satisfied with the computational performance of their projects, a great increase from the number of 29% in the past survey.

There are many possible explanations for the rise in overall satisfaction with computing performance. It may be due to researchers now having access to computing hardware and software packages with significantly better performance, or to research and engineering staff who help streamline the coding and experiment-running processes. Also, it might be the case that researchers are now expecting long-running experiments and plan their schedules and research around it, though this might not be applicable to every researcher since we have shown in Figure 3 that waiting for runs is still non-negligible for many.

Among the researchers who faced computational challenges, 39% resorted to optimization efforts to accommodate computational constraints, 36% utilized resource augmentation, 22% reached out to external support and collaborative resources, and 31% made scope or methodology modification.

Our data also revealed that a minority of researchers, 15%, confronted complete roadblocks in their computation, while 85% reported they did not experience such insurmountable challenges in their work. The latter result could be due to their access to better resources, alternative methodologies, or the adaptability of their research objectives to computational constraints.

TABLE VI: Percentage of researchers indicating the impact of 2x, 10x, and 100x computational speedups on their research.

Impact	Random Candidates			Top Computation		
	2x	10x	100x	2x	10x	100x
No	<b>62%</b>	36%	31%	38%	13%	8%
Minor	24%	<b>40%</b>	<b>33%</b>	<b>52%</b>	<b>58%</b>	38%
Major	0%	10%	22%	8%	27%	<b>52%</b>
Unknown	14%	14%	14%	2%	2%	2%

### B. How much more performance is needed to make an impact?

The survey responses show that 26% of researchers now believe their bigger research plans are unattainable due to the lack of computing power compared to the number of about 70% in the past survey. To understand this deeper, we followed up with a hypothetical question – how a 2x, 10x, or 100x speedup could change their research. The breakdown of four types of responses is presented in Table VI.

- **No Impact:** Some researchers are not sensitive to performance speedup. They are currently not blocked by the computation and do not consider additional computation as a major factor in future research. Or the speedup is not enough to change their research in any meaningful way.
- **Minor Impact:** Some researchers claim that the speedup can improve their workflow but there is no major change in the nature of the research. This might be due to the other part of their research workflow, such as bench work, still requiring a significant amount of research time.
- **Major Impact:** Some are blocked by computation, and the speedup can change the nature of their research or improve their research in a significant way that allows them to explore more important questions.
- **Unknown:** Some are not sure about the impact of the speedup. There might be changes but they cannot predict how significant the impact would be.

The varied responses to potential computational speedups highlight the diverse computational demands and adaptability among researchers, with top computation users particularly sensitive to the benefits of enhanced performance. This sensitivity underscores a critical intersection where computational capability directly influences the scope and ambition of research, enabling more complex inquiries and potentially accelerating scientific discoveries. Furthermore, the distinction between no, minor, and major impacts reflects the multifaceted nature of computational research, suggesting that improvements in computational performance could catalyze a

paradigm shift in how research is conducted across various disciplines, particularly for those currently constrained by computational limitations.

## VII. TRENDS AND CHALLENGES

The end goal of this research is to learn what the computation community should work on and how to better support researchers utilizing computation. The proposed analysis framework enables in-depth analysis, identifying significant trends and challenges. Unlike the results presented in the previous sections, which focus on responses to single questions, these analyses span across questions and often involve revisiting transcripts to gain a deeper understanding of the interviewees.

### A. The Analysis Framework

1) *Data Perspectives: The Past, the Present, and the Future:* The survey, along with the prior study from a decade ago, naturally provides us with three perspectives: the past, the present, and the future.

- **The Past: Historical Baseline.** The foundation of our analysis begins with an examination of the prior study, which offers insight into the methodologies, tools, and analytical approaches prevalent at that time.
- **The Present: Contemporary Results.** Our current results are derived from survey data of current computational researchers. This survey captures a snapshot of today’s practices, tools, and challenges in the field.
- **The Future: Needs and Desires.** Our conversations with the researchers unearth their aspirations, anticipated challenges, and the tools they wish to see developed.

a) *The Past to the Present ⇒ Trends:* Comparing the present results with the historical baseline reveals trends over time, such as shifts in preferred programming languages, adoption of new computational technologies, and evolving research methodologies.

b) *The Present to the Future ⇒ Challenges (Opportunities):* Comparing the present results with the researchers’ needs reveals challenges or opportunities for improvements. These challenges are critical for guiding the development of supportive technologies and methodologies.

2) *Technique: Learn from Individual Cases:* With a lot of data, there is a natural tendency to calculate numbers to represent the general trends and challenges. While the quantitative analysis is important, delving into individual cases can be more helpful to have actionable insights. For example, understanding why one researcher is not using GPU in their research despite clearly expressing a need for additional performance can help us find a better path forward. Understanding both the quantitative trends and qualitative aspirations of researchers enables stakeholders to better allocate resources, develop supportive technologies, and foster an environment conducive to future innovations.

3) *Technique: Get Help from Large Language Models (LLMs):* We employed Large Language Models (LLMs) for analysis on interview transcripts. GPT-4, developed by OpenAI [49], and Claude, by Anthropic [50] were invoked through

their APIs, which guarantees that the data is not used for training based on their privacy policies. Additionally, prior to using any transcripts as input for LLMs, identifiers such as names and emails were removed from the transcript. Figure 4 shows a simple demonstration on how LLMs can help. We ask the LLMs to determine whether a user is “performance aware”, based on mentioning of using high performance hardware like GPUs or doing performance optimizations in their transcript. There is no direct question like this in the survey questionnaire, making the LLM’s ability to search for contextual hints particularly useful. The classification from the LLMs is then compared against “the percentage of research time waiting for runs” which are extracted directly from existing structured survey responses. Human researchers will dive deep into the cases where users are spending significant time waiting for runs but are not “performance aware” and understand the full context.

Throughout the study, we did not use statistics based on LLMs’ responses. Instead, we used LLMs to filter and highlight responses that required further investigation, effectively streamlining the process of identifying critical feedback.

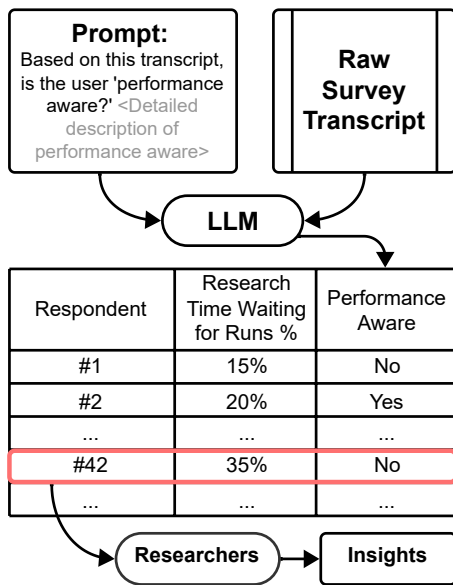


Fig. 4: LLM-driven classification process identifies users for targeted review based on performance awareness and high wait times

4) *The Workflow*: To identify trends, we compare the present data with the past; for challenges, we compare the present with the future. We examine correlated questions within each time perspective—the past, the present, or the future. We then compare multiple entries from either existing structured results or results from LLMs, looking for responses that warrant deeper investigation. These responses either show practices and needs that conflict with each other or provide a detailed account of a problem the researchers face. Actionable insights are summarized from these responses.

## B. Major Trends

a) *More Time Spent In Programming*: Researchers now spend more of their research time programming, in which a larger share of programming time is dedicated to debugging (§V-B). While researchers are generally more satisfied with the performance (§VI-A), the research time actively waiting for runs is still significant (§V-C). These results confirm the importance of programming and computational methods for researchers, motivating further research into enhancing the computational experience to support research endeavors.

b) *More Diverse Needs for Computation*: The needs for computation have become more diverse in many aspects. Computation is now used by researchers from all different disciplines (§IV-A). Thus, these researchers have very different backgrounds and different usage scenarios. In terms of software applications, computation is widely used for simulation, verification, machine learning tasks, and data analysis. Different types of applications exhibit varying performance patterns, including differences in hardware requirements and typical runtime (§IV-C). From the hardware aspect, there is increased usage of clusters, servers, and significantly more reliance on GPUs” (§IV-B). Moreover, improved performance has various impacts on researchers (§VI-B). All these imply that there is no *one-size-fits-all* solution. We must consider the specific needs of target users when developing new techniques for computational researchers.

c) *Programming Languages Shift*: As discussed in Section V-A, the increase in popularity of Python and R, especially with their breadth of packages, has made computing more accessible to more researchers. Meanwhile, other languages including C/C++, Fortran, and MATLAB, have declined significantly. However, for computation-intensive tasks, C/C++ and Fortran are still widely used, highlighting the importance of offering support and resources for these specialized, high-performance computing languages, even as more versatile options like Python and R gain widespread adoption.

d) *Increased Machine Learning Usage*: Our survey results indicate that 30% of respondents are using machine learning techniques in their research in various fields. Machine learning algorithms can uncover hidden patterns, make predictions, and optimize complex systems, making them invaluable assets in various research contexts. Many interviewees expressed interest in incorporating machine learning into their future research. Several researchers noted that machine learning has fundamentally changed their research, citing examples like AlphaFold, which drastically reduced the time needed to predict protein structures. These findings emphasize the importance of supporting machine learning education and infrastructure to facilitate further innovation and adoption across the research community.

## C. Challenge: Making performance more accessible

Compared to a decade ago, there is no doubt that more computing resources are available, both in computing clusters and individual devices such as GPUs. However, effectively



harnessing these resources is a different challenge. Our findings underscore a gap between the potential of these advanced computing resources and the ability of researchers to fully capitalize on them. This gap highlights the critical need to make performance more accessible.

---

*“We don’t have a full understanding of the hardware needed for parallelism. Providing tutorials and training on programming techniques for those unfamiliar with the hardware would be great.”*

— *Chemical and Biological Engineering Researcher*<sup>1</sup>

As computational needs diversify, many researchers lack familiarity with performance optimization techniques. We examined the intersection between computational users that would benefit from different hypothetical speedups and the hardware that they employ in their research. Our analysis, discussed in Sections VI-B and IV-B reveals that around 20% of researchers believe a 10× speedup could significantly impact their work, yet fewer than one-third use GPUs, with similar findings for cluster usage.

At least one interviewee in the engineering discipline explicitly mentioned that, while they spend more than 50% of their research time waiting for runs to complete, they do not optimize code. Most researchers in this category did not have a dedicated software or performance engineer. They tend to work on projects with fewer people involved. These cases highlight a potential area where focused support could make a meaningful impact, especially for researchers who currently lack the resources or expertise to optimize their code. We believe one major barrier to adopting performance enhancing techniques, as cited by several researchers in this category, is a lack of knowledge. This knowledge gap sometimes manifests as specific programming skills, while other times it is as simple as knowing how to access clusters to obtain more compute power.

This observation indicates that we should continue to offer more training on leveraging performance, targeting both existing and potential computation-heavy users. Institutions should establish metrics to avoid “blind spots,” ensuring that users who could significantly benefit from improved performance are aware of the available resources.

#### *D. Challenge: Making development tools more user-friendly*

As research increasingly relies on computational tasks, the demand for reliable, efficient, and portable software has grown. However, the development of tools and practices to meet these needs has lagged. We have compiled feature requests for development tools to address these gaps. These requests include improved auto-completion, enhanced continuous integration and testing, code auto-generation from natural language, comprehensive documentation with practical examples, interactive development environments akin to Jupyter Notebook for non-Python languages, advanced package management and APIs, increased support for code portability and compatibility,

streamlined project setup processes, and more specialized needs such as support for scientific coding in Rust, efficient matrix processing methods, and IDEs tailored to specific research fields.

---

*“I’d really like a more user-friendly UI. When the interface looks like it’s from the 90s, it’s difficult to use. Providing more useful information and tools in the UI would make the learning process much faster.”*

— *Chemistry Researcher*<sup>1</sup>

Despite the advances in programming environments and computational power, the task of debugging remains a time-consuming and often daunting aspect of research (§V-B). We found that the use of debugger has dropped significantly compared to last study. This may be due to the increased use of Python. Developers working with Python may find print statements and code inspection techniques more convenient and accessible, leading to their increased popularity in the debugging process. Many interviewees mentioned that they find debuggers clunky and difficult to use, while some directly expressed the need for better interactive debugging tools to reduce the time spent debugging their code, indicating a need for better debugging interfaces. It is worth noting that some requests are partially fulfilled by recently released large language models and tools based on them. This once again highlights the fast-evolving nature of computation, necessitating frequent revisits of the landscape.

#### *E. Challenge: Handling duplicated and legacy projects*

The extensive list of software and packages highlights the existence of multiple tools serving similar purposes (§IV-C). For instance, for machine learning, frameworks like PyTorch and Tensorflow can be used for similar tasks. DMOL3 [15], Onetep [16], and VASP [17] are focused on similar tasks for quantum chemistry and solid-state physics. These software or libraries often have variations tailored to specific research fields or using different methods, and sometimes the differences are very nuanced. However, this duplication can lead to fragmentation within the scientific community, requiring researchers to learn multiple tools for collaboration or field transitions. There are existing efforts to create abstractions to unify different tools, such as OMFIT for magnetically confined thermonuclear fusion experiments [51], CVXPY for convex optimization problems [52], [53], Keras for machine learning [54]. There are also Python/R packages that wrap computations in more efficient languages, such as Numba [55] and pyfftw [56], making it easier for researchers to integrate these tools into their workflows. Extending these efforts to more fields can be helpful.

Another common issue is with legacy codebases. Researchers frequently expressed frustration over the substantial time needed to gain proficiency, especially in unfamiliar programming languages. It can be helpful to have tools to help transition away from legacy codebases or improve their portability to fully utilize new software and hardware.

<sup>1</sup>This quote has been edited for clarity and conciseness.

### F. Challenge: Supporting machine learning applications

Trends show a significant increase in machine learning usage across various academic fields over the last decade, with scholars expressing strong interest in gaining deeper knowledge and training in its applications.

---

*“Something important would be a lecture series or educational program on the current workings of AI, particularly generative programming and ChatGPT, which are rapidly barreling into our field.”*

— *Comparative Literature Researcher*<sup>2</sup>

---

An economics graduate student expressed a desire to dive deeper into machine learning to uncover new research avenues, which are currently hindered by their limited understanding of these techniques. Additionally, a geoscience researcher focused on climate modeling reported a need for more specialized machine learning training relevant to their research areas.

Many researchers have also mentioned a lack of machines or GPU resources for using machine learning. It is clear that the demand for machine learning will continue to grow, underscoring the importance of current research aimed at making machine learning systems and specialized hardware more efficient and easier to use.

### G. Challenge: Providing adequate and diverse training

More than 30% of researchers requested additional training. As identified in previous challenges, training can be an effective way to close knowledge gaps. In fact, 12 respondents expressed a desire for more technical tutorials on how to fully utilize existing research infrastructure, while three respondents sought resources and training on applying machine learning to their research, addressing two challenges we discussed.

One researcher expressed a wish for educational tools that are more readily accessible, emphasizing a preference for “online resources or tutorials that are easy to navigate and utilize.” This sentiment was echoed by another doctoral candidate who underscored the importance of having clear and concise documentation, instructions, and readily executable code to facilitate learning. Another respondent articulated a preference for a more interactive learning format, specifically highlighting the value of workshop-style training sessions that allow for hands-on experience and direct engagement. The quality of existing educational offerings was a point of contention for some researchers. One individual lamented the organizational aspects of classes, suggesting that improvements in structure and delivery could significantly enhance their willingness to participate. Conversely, a researcher eager to dive into computing and programming felt hindered by a lack of information regarding the availability and location of resources. This indicates a broader issue within the academic ecosystem, where potential learners are ready to engage but struggle due to insufficient guidance and transparency about learning opportunities. Additionally, a social sciences researcher called for

<sup>2</sup>This quote has been edited for clarity and conciseness.

the introduction of more beginner-friendly courses, reflecting a wider demand for educational pathways that accommodate individuals at various levels of proficiency and emphasizing the need for foundational courses for those new to the field.

## VIII. TOOLKIT FOR FUTURE SURVEY STUDY

This section presents a generalized toolkit for conducting similar mixed-methods studies on computational practices based on our approach. It includes the procedures and reflective insights from our study and aims to streamline the survey process for future researchers. As part of this toolkit, several artifacts have been open-sourced [57], including the complete survey questionnaire, a collection of Python scripts, and a spreadsheet to produce tables and figures. More information can be found in the artifact description appendix.

### A. A Typical Procedure

- 1) **Define Research Objectives:** Clearly outline the goals of the study. Identify the specific aspects of computational practices the study will explore, such as programming languages, hardware usage, or the impact of computational speedups on research.
- 2) **Survey Design:** Develop a survey that includes both closed-ended (quantitative) and open-ended (qualitative) questions. Ensure questions are designed to address the research objectives and cover areas like computational backgrounds, current research projects, and needs and expectations.
- 3) **Approval Process:** As a survey project involving human subjects, this kind of study is subject to the approval of institutional review board (IRB). Once the survey scope and questions are clear, it is advisable to seek approval as early as possible. The approval process varies by institution.
- 4) **Participant Selection:** Decide on a sampling strategy such as random sampling or stratified sampling that aligns with the study’s needs. Ensure the strategy allows for diverse representation across disciplines and computational usage intensities.
- 5) **Data Collection:** For qualitative insights, conduct and record interviews with participants. Ensure interviews are semi-structured to allow for in-depth exploration of themes that emerge during the conversation.
- 6) **Data Analysis:** Combine quantitative analysis, which focuses on closed-ended questions and usually presents percentages, with qualitative analysis, which delves into open-ended questions and interviews. Use large language models to help with identifying patterns and themes in the data. Section VII-A discusses our analysis framework which can be a good place to start.
- 7) **Integration of Findings:** Synthesize quantitative and qualitative findings to provide a comprehensive understanding of the research objectives. Look for convergences and divergences in the data to draw nuanced insights.

### B. Reflective Insights

1) *Participant Selection and Generalizability:* In our participant selection strategy, which included researchers across

different disciplines and computation backgrounds, we tried to represent and include the community’s wide-ranging computational practices. However, it is difficult to engage a sufficiently large and diverse participant pool to ensure that our findings generalize to the broader research community beyond the target institution. Future studies could benefit from expanding recruitment strategies to include more varied channels and incentives, enhancing the breadth and depth of participant engagement. Addressing these challenges will be crucial for researchers aiming to build upon our work and further advance our understanding of computational practices in scientific research.

2) *Interviews versus Self-submitted Surveys*: The reliance on self-submitted data introduces the possibility of bias, as participants may have perceptions of their computational practices that differ from reality. The limitations of conventional surveys, often constrained by rigid response options, can also lead to misinterpretations of participant input. Interviews provide opportunities to clarify survey questions based on their background, and also ask follow-up questions to gather more details. During the data analysis process, the transcripts also provide the context to realign some answers when necessary.

3) *Anecdotal Evidence versus Statistical Analysis*: When relying on interviews or surveys, biases in some responses can sometimes be obscured by looking at statistical analysis alone. Given the goal of this study is to understand the computation practice well enough to provide better support to researchers, statistics are less valuable than actionable insights. To achieve a comprehensive understanding of researcher needs, we found that delving into “anecdotes” by revisiting the complete discussions in the interview transcripts provides full context. This enables a deeper exploration of the complexities within computational workflows.

## IX. RELATED WORK

Early studies on scientific software development investigated the state of scientific software development practices, focusing on how researchers acquire knowledge, their use of computational resources, and their attitudes towards software engineering concepts [58], [59]. These studies found that scientific software development knowledge was primarily acquired through self-study and peers and highlighted the need for improvement in areas such as tool use, documentation, and testing. Our work builds upon these findings by exploring the evolution of computational practices and resources over time, as well as expanding the scope of our investigation to include a broader range of academic disciplines. The systematic literature review offers valuable insights into the applicability of software engineering practices in scientific software development [60]. The authors found that the majority of claims supported the usefulness of such practices, with testing and version control being particularly important. Our study complements this work by examining researchers’ actual usage of computational resources, as well as their attitudes towards optimization, reusability, and parallelism. Another

work examines the challenges of effective engagement between users and developers in scientific software projects [61]. Our study takes these findings into account when exploring researchers’ needs and expectations regarding computational resources, providing a more holistic understanding of their requirements and preferences. Our research builds on prior work by investigating the evolution of computational practices and requirements across various academic disciplines [1]. We expand the scope of previous studies by interviewing a larger and more diverse group of researchers, as well as by comparing our findings with historical data. This enables us to identify emerging trends and potential areas for future advancements.

## X. CONCLUSION

This study has revisited and expanded upon a prior study, exploring the landscape of computational practices within the same research institution: Princeton University. Through comprehensive interviews with over a hundred researchers and a comparison with previous survey results, the study captures the evolution of computational practices, emerging trends, and areas requiring further attention. Recognizing the rapidly evolving nature of the field, the study also includes a toolkit for repeating this research in the future to streamline the effort from conducting survey to data analysis. The findings of this study, along with the toolkit for encouraging frequent revisit of this topic, serve as a solid foundation for researchers to continue developing new techniques and systems that advance computation science.

## ACKNOWLEDGMENT

We thank members of the Liberty Research Group, the Arcana Lab, and Princeton Research Computing for their support and feedback on this work. Our thanks also go to Isabella Flinchum, Manraj Mondair, Sanjam Mondair, Kyle Ortiz, Prabhnoor Rai, and Samuel Tusick for their assistance in transcribing the respondents’ answers from the interviews. We deeply appreciate the interviewees’ time and willingness to share their perspectives. We are also grateful for the reviewers’ comments and suggestions, which have significantly strengthened this work. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract numbers DE-SC0022138 and DE-SC0022268, and by the National Science Foundation under Grants CCF-2119070, CCF-2118708, CCF-2107257, CCF-2107042, and DGE-2039656. In developing this paper, we utilized Large Language Models (LLMs) for two purposes. First, OpenAI’s ChatGPT was used to enhance the paper’s readability by refining grammar and spelling. Second, API-based usage of GPT-4 by OpenAI and Claude by Anthropic was employed to extract and interpret themes from the interview transcripts, as detailed in Section VII-A3, with this use being limited to Section VII. All AI-generated output for these purposes was verified by the human authors.

## REFERENCES

- [1] P. Prabhu, T. B. Jablin, A. Raman, Y. Zhang, J. Huang, H. Kim, N. P. Johnson, F. Liu, S. Ghosh, S. Beard, T. Oh, M. Zoufaly, D. Walker, and D. I. August, "A survey of the practice of computational science," in *State of the Practice Reports*, ser. SC '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: <https://doi.org/10.1145/2063348.2063374>
- [2] M. N. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: Opportunities and challenges," *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, 2014.
- [3] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *2014 International Conference on High Performance Computing & Simulation (HPCS)*, 2014, pp. 959–967.
- [4] P. Schmitz, "2021 rcd cm community data report," May 2022. [Online]. Available: <https://doi.org/10.5281/zenodo.6502962>
- [5] J. M. Stone, K. Tomida, C. J. White, and K. G. Felker, "The athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers," *The Astrophysical Journal Supplement Series*, vol. 249, no. 1, p. 4, Jun. 2020. [Online]. Available: <https://doi.org/10.3847/1538-4365/2Fab929b>
- [6] H. Rein and S. F. Liu, "REBOUND: an open-source multi-purpose N-body code for collisional dynamics," *Astronomy and Astrophysics*, vol. 537, p. A128, Jan. 2012. [Online]. Available: <https://rebound.readthedocs.io/en/latest/>
- [7] G. F. D. Laboratory, "Gfdl models," 2023. [Online]. Available: <https://www.gfdl.noaa.gov/model-development/>
- [8] S. F. Ashby and R. D. Falgout, "A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations," *Nuclear Science and Engineering*, vol. 124, no. 1, pp. 145–159, 1996. [Online]. Available: <https://github.com/parflow/parflow>
- [9] D. Komatitsch, J.-P. Vilotte, J. Tromp, M. Afanasiev, E. Bozdag, J. Charles, M. Chen, D. Goddeke, V. Hjørleifsdóttir, J. Labarta, N. Le Goff, P. Le Loher, Q. Liu, A. Maggi, R. Martin, D. McRitchie, P. Messmer, D. Michea, T. Nissen-Meyer, D. Peter, M. Rietmann, S. de Andrade, B. Savage, B. Schuberth, A. Siemenski, L. Strand, C. Tape, Z. Xie, and H. Zhu, "Specfem3d globe [software]," Computational Infrastructure for Geodynamics, 2023. [Online]. Available: [https://github.com/SPECFEM/specfem3d\\_globe](https://github.com/SPECFEM/specfem3d_globe)
- [10] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," *Comp. Phys. Comm.*, vol. 271, p. 108171, 2022.
- [11] A. F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J. D. Joannopoulos, and S. G. Johnson, "Meep: A flexible free-software package for electromagnetic simulations by the ftd method," *Computer Physics Communications*, vol. 181, no. 3, pp. 687–702, 2010.
- [12] I. Ansys, "Ansys hfss," 2023. [Online]. Available: <https://www.ansys.com/products/electronics/ansys-hfss>
- [13] FAccTs, "Orca," 2023. [Online]. Available: <https://www.faccts.de/orca/>
- [14] G. M. J. Barca, C. Bertoni, L. Carrington, D. Datta, N. De Silva, J. E. Deustua, D. G. Fedorov, J. R. Gour, A. O. Gunina, E. Guidez, T. Harville, S. Irle, J. Ivanic, K. Kowalski, S. S. Leang, H. Li, W. Li, J. J. Lutz, I. Magoulas, J. Mato, V. Mironov, H. Nakata, B. Q. Pham, P. Piecuch, D. Poole, S. R. Pruitt, A. P. Rendell, L. B. Roskop, K. Ruedenberg, T. Sattasathuchana, M. W. Schmidt, J. Shen, L. Slipchenko, M. Sosonkina, V. Sundriyal, A. Tiwari, J. L. Galvez Vallejio, B. Westheimer, M. Wloch, P. Xu, F. Zahariev, and M. S. Gordon, "Recent developments in the general atomic and molecular electronic structure system," *The Journal of Chemical Physics*, vol. 152, no. 15, p. 154102, Apr. 2020. [Online]. Available: <http://aip.scitation.org/doi/10.1063/5.0005188>
- [15] B. Delley, "An all-electron numerical method for solving the local density functional for polyatomic molecules," *Journal of Chemical Physics*, vol. 92, no. 1, pp. 508–517, Jan. 1990. [Online]. Available: <http://dmol3.web.psi.ch/dmol3.html>
- [16] J. C. A. Prentice, J. Aarons, J. C. Womack, A. E. A. Allen, L. Andrinopoulos, L. Anton, R. A. Bell, A. Bhandari, G. A. Bramley, R. J. Charlton, R. J. Clements, D. J. Cole, G. Constantinescu, F. Corsetti, S. M.-M. Dubois, K. K. B. Duff, J. M. Escartin, A. Greco, Q. Hill, L. P. Lee, E. Linscott, D. D. O'Regan, M. J. S. Phipps, L. E. Ratcliff, A. R. Serrano, E. W. Tait, G. Teobaldi, V. Vitale, N. Yeung, T. J. Zuehlsdorff, J. Dziedzic, P. D. Haynes, N. D. M. Hine, A. A. Mostofi, M. C. Payne, and C.-K. Skylaris, "The onetep linear-scaling density functional theory program," *The Journal of Chemical Physics*, vol. 152, no. 17, p. 174111, 2020. [Online]. Available: <https://doi.org/10.1063/5.0004445>
- [17] G. Kresse and J. Hafner, "Ab initio molecular dynamics for liquid metals," *Phys. Rev. B*, vol. 47, pp. 558–561, Jan 1993. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevB.47.558>
- [18] F. Bacchini, A. Chernoglazov, D. Grosej, H. Hakobyan, J. Mahlmann, and A. Vanthieghem, "Tristan-mp v2," 2023. [Online]. Available: <https://github.com/PrincetonUniversity/tristan-mp-v2>
- [19] S. Pope, "Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation," *Combustion Theory and Modelling*, vol. 1, no. 1, pp. 41–63, 1997. [Online]. Available: <https://tcg.mae.cornell.edu/isat.html>
- [20] C. T.-F. Laboratory, "Nga," 2023. [Online]. Available: <https://ctflab.mae.cornell.edu/nga.html>
- [21] G. Optimization, "Gurobi optimizer," 2023. [Online]. Available: <https://www.gurobi.com/solutions/gurobi-optimizer/>
- [22] Mosek, "Mosek," 2023. [Online]. Available: <https://www.mosek.com/products/mosek/>
- [23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensorflow: Large-scale machine learning on heterogeneous systems," 2015.
- [25] I. COMSOL, "Comsol," 2023. [Online]. Available: <https://www.comsol.com/comsol-multiphysics>
- [26] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, p. 337–340.
- [27] D. J. Gardner, D. R. Reynolds, C. S. Woodward, and C. J. Balos, "Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, 2022.
- [28] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.
- [29] A. Gurfinkel, T. Kahsai, A. Komuravelli, and J. A. Navas, "The seahorn verification framework," in *Computer Aided Verification: 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*. Springer, 2015, pp. 343–361.
- [30] R. Beckett, N. Giannarakis, D. Loher, and D. Walker, "Nv: An intermediate language for network verification," in *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking and Programming Languages*, 2019, pp. 3–4.
- [31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>
- [32] I. Corporation, "Openvino toolkit," 2023. [Online]. Available: [https://github.com/opencv/opencv\\_toolkit/opencvino](https://github.com/opencv/opencv_toolkit/opencvino)
- [33] M. Frigo and S. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.
- [34] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.
- [35] S. Balay, W. Gropp, L. C. McInnes, and B. F. Smith, "Petsc, the portable, extensible toolkit for scientific computation," *Argonne National Laboratory*, vol. 2, no. 17, 1998.
- [36] The HDF Group. (2000-2023) Hierarchical data format version 5. [Online]. Available: <https://github.com/HDFGroup/hdf5>

- [37] The Blosc Developers. (2023) c-blosc. [Online]. Available: <https://github.com/Blosc/c-blosc>
- [38] S. Andrews, F. Krueger, A. Segonds-Pichon, L. Biggins, C. Krueger, and S. Wingett, "FastQC," Babraham Institute, Babraham, UK, Jan. 2012.
- [39] P. Danecek, J. K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M. O. Pollard, A. Whitwham, T. Keane, S. A. McCarthy, R. M. Davies, and H. Li, "Twelve years of SAMtools and BCFtools," *GigaScience*, vol. 10, no. 2, 02 2021, giab008. [Online]. Available: <https://doi.org/10.1093/gigascience/giab008>
- [40] M. D. Robinson, D. J. McCarthy, and G. K. Smyth, "edgeR: a bioconductor package for differential expression analysis of digital gene expression data," *Bioinformatics*, vol. 26, no. 1, pp. 139–140, 2010.
- [41] D. J. McCarthy, Y. Chen, and G. K. Smyth, "Differential expression analysis of multifactor rna-seq experiments with respect to biological variation," *Nucleic acids research*, vol. 40, no. 10, pp. 4288–4297, 2012.
- [42] Y. Chen, A. T. Lun, and G. K. Smyth, "From reads to genes to pathways: differential expression analysis of rna-seq experiments using rsubread and the edgeR quasi-likelihood pipeline," *F1000Research*, vol. 5, 2016.
- [43] P. Langfelder and S. Horvath, "Wgcna: an r package for weighted correlation network analysis," *BMC Bioinformatics*, no. 1, p. 559, 2008. [Online]. Available: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-559>
- [44] —, "Fast R functions for robust correlations and hierarchical clustering," *Journal of Statistical Software*, vol. 46, no. 11, pp. 1–17, 2012. [Online]. Available: <https://www.jstatsoft.org/v46/i11/>
- [45] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.
- [46] M. D. Abramoff, P. J. Magalhães, and S. J. Ram, "Image processing with imagej," *Biophotonics international*, vol. 11, no. 7, pp. 36–42, 2004.
- [47] J. Troscianko and M. Stevens, "Image calibration and analysis toolbox—a free software suite for objectively measuring reflectance, colour and pattern," *Methods in Ecology and Evolution*, vol. 6, no. 11, pp. 1320–1331, 2015.
- [48] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [49] OpenAI, "Gpt-4 technical report," 2023.
- [50] Anthropic, "Model card and evaluations for claude models," 2023.
- [51] O. Meneghini, S. Smith, L. Lao, O. Izacard, Q. Ren, J. Park, J. Candy, Z. Wang, C. Luna, V. Izzo, B. Grierson, P. Snyder, C. Holland, J. Penna, G. Lu, P. Raum, A. McCubbin, D. Orlov, E. Belli, N. Ferraro, R. Prater, T. Osborne, A. Turnbull, and G. Staebler, "Integrated modeling applications for tokamak experiments with omfit," *Nuclear Fusion*, vol. 55, no. 8, p. 083008, 2015. [Online]. Available: <http://iopscience.iop.org/article/10.1088/0029-5515/55/8/083008/meta>
- [52] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
- [53] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.
- [54] F. Chollet, "Keras," <https://github.com/fchollet/keras>, 2015.
- [55] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: <https://doi.org/10.1145/2833157.2833162>
- [56] pyFFTW. (2023) pyfftw. [Online]. Available: <https://github.com/pyFFTW/pyFFTW>
- [57] J. Giordani and Z. Xu, "Revisiting Computation for Research: Practices and Trends," Jun. 2024. [Online]. Available: <https://doi.org/10.5281/zenodo.12587934>
- [58] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?" in *2009 ICSE workshop on software engineering for computational science and engineering*. Ieee, 2009, pp. 1–8.
- [59] L. Nguyen-Hoan, S. Flint, and R. Sankaranarayanan, "A survey of scientific software development," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: <https://doi.org/10.1145/1852786.1852802>
- [60] D. Heaton and J. C. Carver, "Claims about the use of software engineering practices in science: A systematic literature review," *Information and Software Technology*, vol. 67, pp. 207–219, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584915001342>
- [61] J. Segal and C. Morris, "Scientific end-user developers and barriers to user/customer engagement," *JOEUC*, vol. 23, pp. 51–63, 10 2011.

# Appendix: Artifact Description

## Artifact Description (AD)

### I. OVERVIEW OF CONTRIBUTIONS AND ARTIFACTS

#### A. Paper’s Main Contributions

The paper investigates the evolution of computational practices in various academic disciplines, identifies emerging trends, and highlights potential areas for future advancements. Our study relies primarily on data collected through comprehensive interviews with researchers and results of a previous survey carried out more than a decade ago.

The main contributions of the paper are as follows:

- $C_1$  Carrying out a comprehensive survey on researchers’ engagement with computation, encompassing 106 interviews;
- $C_2$  Presenting the results and analyses, including identified trends and challenges;
- $C_3$  Reflecting on the survey study process and providing a “toolkit” for conducting similar research in the future.

#### B. Computational Artifacts

$A_{1,2,3}$  <https://doi.org/10.5281/zenodo.12587934>

Artifact  $A_1$  is the interview questionnaire. Artifact  $A_2$  is the set of Python scripts for generating tables and figures. Artifact  $A_3$  is a spreadsheet that contains synthetic survey data and equations used to compile data in Tables 2, 4, 5, and 6.

Artifact ID	Contributions Supported	Related Paper Elements
$A_1$	$C_1, C_3$	Table 1
$A_2$	$C_1, C_2$	Figure 1-3
$A_3$	$C_1, C_2$	Table 2, 4, 5, 6

### II. ARTIFACT IDENTIFICATION

#### A. Computational Artifact $A_1$

##### Relation To Contributions

Artifact  $A_1$  is the survey questionnaire, which includes all the questions asked to respondents and the multiple-choice options when applicable. Unlike a traditional computational artifact, it does not function as a computational process. Therefore, it does not have “Expected Result”, “Expected Reproduction Time”, “Artifact Setup”, or “Artifact Execution” sections.

The questionnaire is included to help other researchers fully understand the interviews and facilitate future studies.

#### B. Computational Artifact $A_2$

##### Relation To Contributions

Artifact  $A_2$  consists of Python scripts that generate Figures 1-3 in the paper. These scripts require survey data to function properly. Currently, the necessary survey data is not included, meaning the scripts alone cannot reproduce the figures. To create the figures, one must first prepare the specified CSV files filled with the survey data; then, the scripts can be executed to produce the figures as presented in the paper.

##### Expected Results

When run with our survey data, the expected results is an exact replica of Figures 1-3 in the paper. When run with other survey data, the results will reflect the input data, in the style and format of Figures 1-3.

##### Expected Reproduction Time (in Minutes)

Using our data, the expected time to produce Figure 1-3 is around or under one second. Using other data, the time to produce the figures is proportional to the size of the dataset. We do not expect any survey data to be significantly large that would require more than a few seconds to process.

##### Artifact Setup (incl. Inputs)

*Hardware:* No specific hardware is required.

*Software:* Python 3.x

*Datasets / Inputs:* CSV files should match the specific data requirements detailed in the comments in the script. Each CSV file needed for input includes descriptions of the necessary fields and the corresponding data.

*Installation and Deployment:* Run the specified scripts in a Python environment.

##### Artifact Execution

Run the specified scripts in a Python environment, giving a CSV file path where indicated. Run the code in the order provided in the scripts.

##### Artifact Analysis (incl. Outputs)

The scripts should output figures in the style of Figures 1-3.

#### C. Computational Artifact $A_3$

##### Relation To Contributions

Artifact  $A_3$  is a spreadsheet with synthetic data and spreadsheet equations used to compile data in Tables 2, 4, 5 and 6. Owing to data privacy and confidentiality regulations, the authentic data cannot be made public. This dataset underpins the data compilation procedures used to populate data in tables throughout the paper.

Similar to artifact  $A_1$ ,  $A_3$  is not a computational tool and thus does not require sections such as “Expected Result”, “Expected Reproduction Time”, “Artifact Setup”, or “Artifact Execution” sections.