Computational practices in introductory science courses

Claudia Fracchiolla and Maria Meehan
School of Mathematics and Statistics. University College Dublin. Ireland

Nowadays computation is considered to be one of the pillars of modern science. This is reflected in the fact that much scientific research and industry work relies heavily on technology and computation. As university educators we need to equip graduates with the tools to help them succeed in their discipline. The study reported here is part of a larger project which aims to identify computational practices used by faculty from across the College of Science in an R1 Irish university in their disciplines and research, in order to inform the design of a computational science course intended for first-year undergraduate students. This paper reports on findings from one-to-one interviews with fourteen faculty from across six schools in the college. Their understanding of the nature of computational thinking and their views on the role that computation does/can play in the undergraduate science curriculum, and specifically in teaching and learning within their disciplinary program are presented. Concerns and challenges related to the embedding of computation in undergraduate science programs voiced by participants are presented here. Finally, implications of the findings for the design of the first year computational science course are discussed.

2021 PERC Proceedings edited by Bennett, Frank, and Vieyra; Peer-reviewed, doi.org/10.1119/perc.2021.pr.Fracchiolla

I. INTRODUCTION

The increased importance of computational practice in science disciplines has led to the proliferation of computationally intensive undergraduate modules with the title 'Computational X', where X includes areas traditionally dependent on computation such as Physics and more recently, including areas such as Biology. Employers value the ability of graduates to integrate programming and computation with the underlying discipline. Yet in conversations with undergraduates, faculty, and international colleagues, the authors are starting to recognise and understand the challenge of teaching "computing" in a manner that enables this integration.

The study reported in this paper is part of a larger project which consists of two parts. The first part aims to identify the computational practices used by faculty in research and teaching in the College of Science at an R1 Irish university. To this end, in Summer 2020 a survey was sent to faculty in the schools[8] in the college. The survey had 93 respondents and findings are reported elsewhere [1]. The survey was followed up with one-to-one interviews with fourteen faculty from across the six schools in late Summer 2020. The findings from these interviews is the focus of this paper.

The second part of the larger project consists of the design and development of a first year computational science course in which students will engage in computational practices to solve problems from across different disciplines in science. This course is intended as a first step in the development of students' computational skills, which will become discipline specific as they progress through the last three years of their chosen science major. The findings from the survey [**?**] and the study reported here, are intended to inform the design of this course and help curriculum developers articulate program-specific computational outcomes.

In the analysis of both the survey and interview data, we use the Weintrop et al. (2016) taxonomy [2] of computational thinking as a base to contextualize the computational skills and practices for each of the science disciplines represented in the study. Weintrop et al. carried out a systematic literature review on computational thinking, interviewed scientists and instructors from multiple disciplines, and examined three different high-school classes, in order to identify, define, and classify computational practices. From their study, they created a taxonomy of computational thinking and practices divided into four main categories: Data Practices; Modeling and Simulation Practices; Computational Problem Solving Practices; and, Systems Thinking Practices. This taxonomy has been used to code STEM professional' interviews about how they use computational thinking in their work [3], [4], [5], [6] . While we acknowledge that this taxonomy might not be comprehensive [9], for the purpose of this study it suited our needs as a guide for determining what counts as computational thinking and practices within the College of Science.

While coding the fourteen interview responses using the Weintrop et al. [2] taxonomy of computational thinking, we identified parts of the responses that did not fit any of the taxonomy categories, yet revealed relevant information about the participants' views on what constituted computational thinking and practices, issues relating to the teaching and learning these practices, as well as their incorporation into the undergraduate curriculum. It is these findings that we present in Section III after discussing the Methodology, including our data collection and analysis, in Section II. Implications of the findings to the design of the first year computational science course are discussed in the final section.

II. METHODS

The study outlined in this paper follows a case study approach. The purpose of using this approach is that it allows one to understand the use of computational practices within the context of a specific College of Science at an R1 Irish University. We are aware that the computational practices might be representative of the specific research that is conducted at the university and not necessarily reflect the wide variety of computational practices that could be applied in other contexts within the disciplines.

Convenience sampling was used to select participants for the interviews. Participants were recruited through the initial faculty survey. In total, fourteen interviews were conducted. Of these, there were two participants from each of the School of Biomolecular and Biomedical Science and the School of Chemistry; three each from the School of Biology and Environmental Science, the School of Physics, and the School of Mathematics and Statistics; and one from the School of Earth Sciences. There was also a good spread of career level, with three interviewees being full Professors, seven Associate Professors, three Assistant Professors/Postdoctoral Researchers, and one PhD student. There were five female and nine male participants.

The semi-structured interviews incorporated the language of computational thinking from Weintrop et al. [2] and followed a protocol designed to capture participants' understanding of the term computational thinking, the computational practices used in their research field, when and where participants developed these practices, what practices they feel graduates from their associated degree program should develop, and, if/how they use computation in their teaching. Interviews were conducted and audio recorded through zoom and lasted approximately 20-40 minutes.

The analysis of the interview data was conducted in two parts. The first part consisted of coding the interview responses using the Weintrop et al. [2] taxonomy of computational thinking, within the context of each participant's discipline. We coded the interviews on MaxQDA using a codebook developed from Weintrop et al. taxonomy where the four main categories corresponded to top level codes and the sub-categories as the corresponding subcodes. However, the taxonomy was only the basis for our codes. While coding the interviews we identified parts of the responses that did not fit any of the taxonomy categories [2], yet revealed relevant

information about the participants' views on computational thinking and practices, as well as the incorporation of those into the undergraduate curriculum.

To analyze these parts of the responses we used emergent coding. We marked them as they were appearing and assigned a name that described what the interviewee was communicating. Through this process we identified eighteen initial emerging themes. To verify the cohesiveness within and across the emergent themes and provide clear definitions of what each theme represented, we reviewed each one individually. We used MaxQDA to retrieve all the coded segments for each individual and reviewed each coded segment for homogeneity. A refined description of the theme was written. During this process, a number of themes were merged, since the distinction between them was not clear enough to have separate categories. It is important to note that existing themes could be refined further, but we believe that the current categorization is sufficient to give a good account of the interviewees' beliefs about computational practices, the use of such practices in their respective fields, and the incorporation of these practices in their undergraduate teaching.

III. FINDINGS

Through Weintrop et al. taxonomy [2] we identified the computational practices that participants expressed are relevant to their discipline. In this paper, we focus only on the final emergent themes and subthemes, which are related to when and how much computational practices is or should be embedded in the corresponding disciplines.

First we want to provide some background on incorporation of computational practices in the College of Science at this R1 University. Currently, first year students at this University can choose to take their twelve courses from across the scientific disciplines, or specialise in one to two areas. In their first year, there is no core course where the basics of programming and computation are introduced. The three core courses are Calculus, Algebra, and Scientific Enquiry - a course where students work in small groups with a faculty member to read a research paper in a area that interests them. Students pursuing path ways in the Mathematical or Physical Sciences take a core course in computation in second year where Python is introduced. In these path ways, students' computational practices are further developed in laboratories, homework assignments, course projects, and final capstone projects. Most faculty in these disciplines stated that they would like students to engage with computation from first year.

In the case of the Biological sciences, there seems to be variation in relation to the prevalence of computational practices within the programs. Some specializations within the biological sciences rely heavily on large data analysis, while others only use visualization tools. Therefore, opinions on when students should start learning computational practices ranged from being when they specialise in their third year, to all students learning some minimal skills (conducting online searches, downloading and installing software, and visualization) from the start of their program.

What is computational thinking?

Participants were asked to explain what they understood by *computational thinking*. Responses varied, with the variation a reflection of the computational practices used within the participant's discipline, and more specifically, their research area. Those who do not make extensive use of computational practices in their research, for example, only use software to collect or visualise different representations of data, described computational thinking as the explicit use of computers: "How the keyboard interacts with the CPU and then spits something out on the screen. That's what computational thinking I'm thinking about would be like - a hardware thing." Or they described it as the action of writing a sequence of computer codes to solve a problem or automatize a process: "being able to use a computer and coding to be able to solve a specific problem." On the other hand, participants whose research relied more on computational practices such as developing computational models or processing data, described computational thinking as the action of breaking a problem down into pieces in a sequential mode: "it is understanding the problem enough to lay down an algorithm and set of rules that a computer is going to be able to use independently to get you to some endpoint - the goal you want to reach." Some participants in the field of Mathematics and Statistics offered that computational thinking is one of the "pillars for building knowledge" as it supports the understanding of a concept: "in order to write a code the individual needs to develop an understanding of the concept and be able to model the problem." The differences in the perceptions of what computational thinking is, seemed to be grounded in whether interviewees believed that computational thinking overlaps with an analytical process or not, with some expressing that the difference between computational thinking and analytical thinking lays in the explicit use of computers.

Students engaging in computational practices

Participants' described how they perceive the relationship between students' learning of content and the use of computational practices. Again the range of beliefs exhibited was generally associated with the prevalence of computational practices in participants' respective disciplines and research. The majority of participants believed that computational practices can be used as tools to support the learning of the content, to better address complex problems, or to transform abstract concepts into something more relatable. For example, a lecturer from the School of physics says "It's a way that you can see. Well, what's the effect of this particular component of the equation. And it's much quicker for the computers to do it, then for the students to work out. Or what if I change the math or the speed or whatever the orbit changes like x or y." Some participants believed that the learning of content and computational practices cannot be disaggregated. A deeper understanding of concepts is developed when engaging in computational practices, because it can be seen as "the student

teaching the computer about the specific concept," as one of the interviewee stated. These types of responses were generally from interviewees from the mathematical and physical sciences. They believed that the use of computational practices supports sense making, such as checking if computer-generated answer reflects the results expected. Another train of thought which was more common amongst participants in the Biological Sciences, was that computation can be used as a tool for learning. For example, simulations can be used for addressing misconceptions or to visualise concepts and results. As stated by a participant "students can see that simulating these things is [spread of virus] interesting and could help for [m]anagement of a health crisis or management of endangered population... I think can be seen [computational thinking] directly is relevant to ecology and biology." However it is not a critical part of the learning process in their discipline. They noted that nowadays there is a myriad of software that can be used directly to do any of the analysis needed and what matters is students' understanding the content.

Baseline knowledge of computational practices

Following the above discussions, all the interviewees attempted to describe a baseline knowledge that students should have on graduating. Once again, the nature of this knowledge depended on the relevance of computational practices to their respective fields. Most expressed that having some minimal knowledge of computational practices was critical for success in the current job market, independent of the discipline. However, the range of what those minimal skills might be, ranged from knowing how to critically search on the internet and download software, to programming and big data analysis. There was a common interest in students feeling comfortable with a software user interface and having a minimum understanding of the underlying processes involved in order to be able to input variables and interpret results. For disciplines like Mathematics, Statistics, and Physics there was a consensus that students must have a basic level of knowledge and confidence to use Python or a similar programming language on graduation. However, in Biology, Chemistry, and Earth Sciences there was no consensus on what the basic level of knowledge should be, and once again, this varied within the research area. For example, those working in ecology or evolutionary biology believe that students must develop at least a basic understanding of how to consider big data analysis, through either coding directly or using some software for this purpose, while others are satisfied with students being able to download, install and use software packages.

Challenges and concerns - incorporating computational practices in the curriculum

Finally, we identify six subthemes that highlight the challenges and concerns that interviewees expressed about incorporating more computational practices into their undergraduate programs, and about the design of the new computational science course that is intended for all first years. Yet again, their views are related to their perception of computational practices and the importance of these practises in their corresponding disciplines. Another issue that underpins their comments is the curricular design of the science programs in this university.

The first subtheme relates to the broad range of students' prior knowledge of computational practices. Dealing with the wide range of coding skills possessed by incoming first year students will always be a challenge for first courses in computation. However the interviewees were concerned with students' prior experiences of computation within the program due to its modular nature. Courses can be quite self-contained especially in the first two years, which in turn can result in students enrolling in courses with varying levels of familiarity with computational practices. This variability means that lecturers may find it easier to assume students have no prior knowledge to avoid having students drop the course, but this then limits what they can do: "there is the limit to how much we can embed computation ... we're in a modular system now where modules are almost meant to be self contained - go pick and choose and it's almost like restarting again."

The second subtheme also relates to the curriculum design. The science undergraduate program does not require students to specialise in first year, therefore a first year course that is intended to be core for all students must appeal to students with interests across the range of scientific disciplines. Interviewees were concerned about how one might design a first year computational science course that is meaningful for the interests of students "the science program has so many different degrees and so many different types of students, but to find something that is useful for all of them is probably very challenging." One interviewee was concerned that if students did not see the relevance of computation to their area of scientific interest, they might dismiss it "if the students don't feel at the end of the module that *Oh, wow, I didn't realize that I needed this stuff. Thank goodness I did that*, then we've kind of lost them forever." Another interviewee also spoke about the need to make explicit the link between computation and the student's area of interest in order to address "students saying, why should I learn programming if I just want to do math or biology or whatever?."

This leads to another subtheme, that of inclusivity. There were different perspectives on this point. In terms of gender and racial inclusiveness the majority of responses expressed that incorporating computational practices within the courses would not further affect the issues of under-representation: "It [under-representation of certain groups in STEM disciplines] is such a big problem already that I do not think it will make a difference." However, some interviewees were concerned that by embedding computational practices in courses, some form of self-selection may occur: "there might be quite a lot of self selectivity here with the people that are going to do physics might already be kind of attuned to, keen on coding things and I suspect there is quite a strong overlap there as well." Students with no experience in computation may avoid these courses, therefore exacerbating the problem.

Time constraints appear in two different ways. One way relates to the time faculty would need to invest to adapt their

courses to incorporate (more) computational practices. Designing discipline specific exercises that incorporate computation as a means for students to learn the content can be a large initial investment of time for lecturers. This can be further exacerbated if the lecturer's knowledge of how to do this is lacking. As one interviewee explains: "staff themselves wouldn't necessarily be familiar enough to deliver those modules. There are some of my colleagues that are really good at coding, but most wouldn't be so it'd be very challenging for them to try and introduce some computation aspects into their current modules."

The second way that time constraints play out for faculty relates to finding a compromise between teaching content and teaching computational practices. As an interviewee questioned: "where would you fit [computational practices] in the schedule? It's very, very tight." Given the variation in students' prior computational knowledge, if a lecturer plans to embed computation in their course, then they would likely have to exclude some content due to time constraints imposed by refreshing computational practices: "Going back to the basics of computer practices, how to write small python scripts, takes a lot of time away from content learning."

Directly related to this last point is the challenge of *embedding, and building on, computational practices in the curriculum* in a coherent manner. If students only meet computation sporadically in different courses, and it is not built upon, then it is likely lecturers will have to spend time reviewing these practices before they use them. As an interviewee noted: "There is a bit of a disconnect - [there] is very little computation, other than in the [physics] labs. If we could really build on that then you can start giving them computation exercises from second year on perhaps maybe the first year."

Finally, scalability of the first year computational science course is another challenge that interviewees foresee. The university can have up to 500 first year science students, which raises the issue of how to incorporate computational practices in courses in a manner that is affordable. There is also the issue of accessibility. Students need access to computers, not only during class but for completing assignments outside of class. One interviewee noted "we do have a decent number of students who don't own laptops. It's in a classroom... So they have to bring a computer and have their own."

IV. CONCLUSIONS

The role that each interviewee's discipline and research area plays in forming their views of what computational thinking is, and what computational skills a graduate should possess, is very clear. Therefore, while a first course in computational science may expose students to a range of ways in which computation plays a role across the sciences, a subsequent one-size-fits-all approach to engaging students in computational practices will likely not be effective. For students to appreciate the role of computation in their discipline and meaningfully engage with computational practices during their program, they should meet the disciplinary relevant practices in authentic disciplinary contexts [3] [5]. In addition, if students only engage with computational practices in special laboratories or homework projects, they may perceive that computational practices are not practices of their disciplinary repertoire, and will not place sufficient value on them [6].

Indeed one may question the wisdom of designing a first year computational science course that caters to all students irrespective of their interests. However, a key feature of the science program in this university is the flexibility it affords first year students to experience a number of scientific disciplines before specialising if they so wish. What is clear is that a collaborative approach from faculty who use computation in their research or discipline is required in order to design materials and problems that are authentic and meaningful. Therefore, getting an understanding of what this baseline is across the different disciplines would be important for the design of the introductory courses and how computational practice could be sprinkled down the program's modules, as students progress in their respective courses. However, this baseline should be defined per discipline and not justly dependent on individual research specific practices.

Some suggestions provided for these changes are: i) Explicitly incorporate computational practices in existing modules. This can make the learning process of some concepts more interactive (i.e. simulations, broken code) and relatable (real live examples of using programming to identify solutions for problems); ii) Use Jupyter notebooks (or similar) early on in the programs, so that when students get to the labs or other modules that require the use of python (or programming) students do not feel like they are starting from scratch; iii) Have introductory science courses that have modeling examples for each of the core disciplines in science to introduce students to what it means to code; iv) Reorganize the core modules and programs' pathway so that there is more consistency in students' level of familiarity with computational practices as they progress through their respective programs; v) Create more projects that require computational practice and use group peer learning to support students scaffolding of the practices and their perception of those practices within their disciplines; vi) Explicitly explain the logical thinking process of what it means to code; and finally vii) Have appropriate physical spaces for students to learn and feel supported (e.g., n a computational science course requires group work, then the learning environment has to support this).

ACKNOWLEDGMENTS

This project was supported by by the National Forum for the Enhancement of Teaching and Learning in Higher Education in Ireland.

[1] Fracchiolla, C., Mullen, C., and Meehan, M. (2021). Computational Practices in Science Disciplines. Conference proceedings presented at 94th Annual International NARST meeting. In press.
[2] Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of science education and technology, 25(1), 127-147. doi:10.1007/s10956-015-9581-5.
[3] Beheshti, E., Weintrop, D., Swanson, H., Orton, K., Horn, M., Jona, K., Wilensky, U. (2017). Computational thinking in practice: How STEM professionals use CT in their work. Paper presented at the American Education Research Association Annual Meeting 2017.
[4] Caballero, M. D. (2015). Computation across the curriculum: What skills are needed?. preprint arXiv:1507.00533.
[5] Caballero, M. D., & Merner, L. (2018). Prevalence and nature of computational instruction in undergraduate physics programs across the United States. Physical Review Physics Education Research, 14(2), 020129.
[6] Pawlak, A., Irving, P. W., & Caballero, M. D. (2020). Learning assistant approaches to teaching computational physics problems in a problem-based learning course. Physical Review Physics Education Research, 16(1), 010139.
[7] Amanda PEEL, Sugat DABHOLKAR, Sally WU, Michael HORN, & Uri WILENSKY (2020). An Evolving Definition of Computational Thinking in Science and Mathematics Classrooms. Paper presented at 2020 International Conference on Computational Thinking and STEM Education.
[8] Faculty in the School of Computer Science was not included as project pertains to integration of discipline-based computation.
[9] An extended version of the Taxonomy, contextualized to Science and Mathematics it is being develop [7]