Improving Student Takeaway in Introductory Numerical Analysis/Scientific Computing Courses: A Threshold Concepts Approach

Jonathan Calver*, Tom Fairgrieve†, Paul Muir‡

Abstract

In this report, we describe a project whose goal is to address issues associated with student takeaway, i.e., issues with *enduring learning*, in introductory courses in Numerical Analysis/Scientific Computing (NA/SC) commonly taught in undergraduate degrees in Computer Science, Mathematics, and Engineering. The fundamental point we begin with is the observation that if we want our students to take away important concepts from our courses, then our course material, the way that the course is taught, and the corresponding evaluation instruments must *focus* on those important concepts. Deeper student engagement with these concepts implies that they will be better retained by the students after the course is completed.

It is therefore essential that a careful analysis of the course content be undertaken in order to identify the concepts, the essential "gems" of the curriculum, that students should take away. In this project, we have employed the well-known framework of *Threshold Concepts* (TCs) in order to identify essential "takeaway" concepts for introductory NA/SC courses. We report on the TCs we have identified for introductory NA/SC courses and show how components of a typical/traditional NA/SC curriculum map onto the TCs. An initial effort to better incorporate these TCs into a recent offering of an NA/SC course is described. We also report on the results of two types of surveys that we have developed and given to students in order to further investigate the impact of the threshold concepts that we have identified.

Focusing a course on the TCs allows for a more extensive treatment of these featured concepts through the use of active learning activities in the classroom coupled with authentic assessment instruments that require deeper student engagement with course material at higher levels within Bloom's taxonomy of learning.

Keywords: scientific computing, numerical analysis, threshold concepts, curriculum design, active learning, authentic assessment

*University of Toronto
†University of Toronto
‡Saint Mary's University

1 Introduction

A fundamental assumption that we make in every course we teach is that the goal of the course is to enable our students to attain a deep understanding of the most important pillars of the course material. This type of deeper understanding and engagement with the course material leads to improved student retention of these fundamental pillars. This is referred to as *enduring learning*, i.e., learning that lasts.

For our students, this deep understanding happens when we challenge them to engage with the fundamental course concepts at a significant level of detail, in different ways, over an extended period of time. Since this type of engagement with the course material takes more time, it is essential that we challenge the "over stuffed" curriculum. Quite simply, deeper understanding requires a deeper engagement with the course material. This, of course, takes time, and when there is too much course material, deeper understanding cannot happen. An over stuffed curriculum inevitably forces students into superficial learning. When the learning is superficial, i.e., largely memorized and not well understood, the student will retain little knowledge of the course content within a short period of time after the course has ended; see, e.g., [2].

An essential implication from the above, then, is that the course curriculum must be carefully analyzed to determine the key concepts within the course that should be focused upon. These become the centerpieces or "gems" of the curriculum [5]. Furthermore, the course topics that are more incidental or tangential must be removed from the curriculum. This latter step is essential because it then allows the teacher and the students to focus on the course material that matters. Every nonessential topic that is included in the curriculum diminishes the time and focus that is available for the essential course material.

At the centre of the above discussion is the crucial task of identification of the "gems" of the curriculum. In this paper, we employ the well-known framework of *Threshold Concepts* (TCs), first introduced in [6] and now widely used in curriculum development; see, e.g., [9] and [8]. To our knowledge, the question of identification of TCs for an introductory course in NA/SC has not been undertaken before. However, the investigation of TCs for various other areas within computer science has received considerable attention; see, e.g, the survey on TCs in computer science, [8], and references within.

TCs are characterized as concepts that are:

- *troublesome*, i.e., difficult and challenging for students,
- *transformative*, i.e., requiring a fundamental change in the student's understanding of the concept; a transformation must take place within the student in order for the student to move from a naïve understanding of the concept to a deep level of understanding of the concept; it requires a fundamental paradigm shift,
- *irreversible*, i.e., once the student obtains a deep and transformative understanding of the concept, this new perspective cannot be forgotten, and,

- *integrative*, i.e., the deep understanding of the TC allows the student to see how different parts of the course are connected by the concept.

It is our opinion that the characterizations of TCs as transformative and troublesome are the most important of these. Once a TC has been learned deeply it will clearly be irreversible knowledge. As well, the deep understanding of the TC will naturally allow the student to make connections across the course material through the lens of the TC.

It is nontrivial to analyze the contents of the given course in order to identify the TCs. The process requires careful reflection on the traditional course material in order to identify the fundamental concepts that rise to the level of TCs as characterized above.

Once these TCs have been identified, the next step is the hard work of curriculum redevelopment in order to reorganize the course material so that it serves the teaching and learning of the TCs. Course material that supports the TCs must be featured and course material that is peripheral to the TCs must be removed.

The great strength of this approach is that it opens up time within the course for the instructor and the students to engage deeply with the course material that remains. Three important aspects of this deeper engagement with the course material are as follows. Firstly, each of the TCs must be revisited (in slightly different ways) at spaced intervals throughout the course. Secondly, each time a TC is revisited, the level of student engagement should be increased. (An important road map for determining these increased levels of student engagement is available through the well-known Bloom's taxonomy [3] or the revised Bloom's taxonomy [1]. Briefly, the taxonomy identifies six increasing more challenging levels at which a student can engage with course material: remembering, understanding, applying, analyzing, evaluating, and creating.) Thirdly, and this point is essential, the evaluation instruments that are employed by the instructor must be consistent with the deeper levels of understanding that the students are expected to achieve. The evaluation instruments must require the students to engage with the course material at higher levels within the revised Bloom's taxonomy. Good evaluation instruments will recognize the deeper level of understanding that has been attained by the students. Evaluation instruments of this type fall under the umbrella of what is known as authentic assessment — see, e.g., [7] and references within. See Figure 6 in the Appendix of this report for an example of a question that we have developed of this type.

Since the TCs are covered from day one and returned to many times over the duration of the course, there is a gradual buildup in the understanding of the TCs by the students. This improves the "robustness" of the course with respect to the learning of the TCs because no specific class, tutorial, assignment, etc., is responsible for the entire "coverage" of a given TC. We return to this point later in the report.

Another important point is that the focus of the curriculum on the topics that are in support of the TCs allows time for deeper student engagement with these topics through the use of active learning approaches; see, e.g., [11]. Active

learning strategies feature the student as an active participant in the learning process; often active learning exercises involve students working in small groups on problem-solving activities focused on a topic rather than listening to an instructor lecture on that topic. The group work requires communication of a position, listening and understanding the arguments put forth by others, defending a position, etc., all activities that lead to deeper engagement with the course material. Some related work on the use of active learning strategies in Numerical Analysis/Scientific Computing (NA/SC) courses has recently been considered in [4] and [12].

While the name "threshold concept" suggests that a student might pass from the non-expert understanding of a concept to some level of expert understanding of the concept simply by passing through a threshold, as if walking from one room to another, the literature emphasizes that the experience of moving from the initial non-expert state to a deeper state of understanding is usually quite a complicated process, in which a student works hard to go through a transformative experience that can require substantial time and effort. We acknowledge that in a given course it may not be possible for a student to make it all the way to an expert level of understanding of a given concept; the idea is that there is a level of understanding, deeper than the initial non-expert understanding, that the instructor has set as a goal for the students.

An important concept related to TCs is the notion of *liminality*. A student who is in the process of moving from the non-expert understanding of a concept to deeper level of understanding of the concept, but has not yet arrived at the latter, is referred to as being in a *liminal state*. In this state the student understands that the naïve, non-expert viewpoint is incorrect, but has not yet reached the level of understanding that is expected by the instructor. Being comfortable with being within the liminal state is an essential part of a successful student experience that leads eventually to a transition out of the liminal state to the state where the TC is understood sufficiently deeply for the purposes of the course.

The central point in the use of a threshold concept framework for curriculum development is that, while a smaller number of topics are covered, these topics that remain are treated in such a way that students have the opportunity to engage deeply with them, coming away with a transformative understanding of these topics that will stay with them long after the course has been completed.

In this report, we begin by briefly describing a typical introductory NA/SC course. We then consider the application of a threshold concept analysis to the body of material that represents the content of the typical introductory course in NA/SC. Such courses are commonly taught in Computer Science programs as well as in Mathematics, Engineering, and Physics programs. A major component of this report is this identification of a set of TCs for introductory courses in NA/SC. For these TCs, we describe the naive or non-expert understanding of the concept and then compare that description with the expert understanding of the concept. We also discuss how the standard NA/SC course content maps onto the TCs and briefly describe our initial efforts to better highlight these TCs within the course. We then briefly review some preliminary work involving

two surveys that we have developed and administered to students in order to further investigate the impact of the TCs that we have identified. The report closes with our summary and conclusions and the identification of topics for future work.

2 Description of a Typical NA/SC Course and Associated Issues

Introductory courses in NA/SC are often offered as part of a computer science, mathematics, engineering, or physics program — at both the undergraduate and graduate level. We are most interested in courses offered in the context of undergraduate computer science programs, but most of what we will discuss is relevant to other programs.

In this section, we consider the introductory NA/SC course offered in the Department of Computer Science at the University of Toronto. This course is not required for the core computer science program, but is required for students specializing in scientific computing. A typical offering of the course will have an enrollment of around 80-100 students. While the primary audience for the course is computer science majors, it often attracts students taking programs including subjects such as mathematics, economics, or physics. The course is officially listed as a third year course, but it has minimal prerequisites (linear algebra, calculus, and only first year programming); as a result, the course typically has a mix of students from second, third, and fourth year.

The content and structure of the course is fairly similar for each offering of the course, with the content largely dictated by the official course calendar description, which reads:

> The study of computational methods for solving problems in linear algebra, non-linear equations, and approximation. The aim is to give students a basic understanding of both floating-point arithmetic and the implementation of algorithms used to solve numerical problems, as well as a familiarity with current numerical computing environments.

There can be slight variations in the course content depending on the instructor. For example, some instructors devote more time to covering floating point arithmetic and solving linear systems, which leaves less time for other topics such as non-linear equations and interpolation (a common type of approximation).

For many years, the course has been consistently taught using a traditional lecture style with notes either written on a whiteboard or presented using prepared slides. Each week typically consists of two hours of lecture time and one hour of tutorial time, which is used essentially as an extra lecture hour and often includes explanations of worked examples. Assessments typically consist of a midterm, a final exam, and 3-4 assignments. The assignments contain a

mix of theory and programming, requiring students to not only write programs, but also interpret the results of numerical experiments that they run. Unfortunately, it has been observed that this last task is usually poorly done, with many students either only handing in rather superficial discussion of any numerical results or simply skipping entire parts of assignment questions.

In discussions with the instructors of this course and similar courses, there was a consensus that the course has rather poor student uptake and even students taking the course typically do not seem particularly engaged in the course. We also have the sense that students are not taking away from the course what we really want them to be taking away. Taking a look at the curriculum and the structure of the course, we have posited that this lack of student engagement stems from an overly full curriculum, with too many side topics, a lack of clear overarching themes, and a subject matter that students simply find uninteresting. The combination of the above issues and relatively infrequent assessment of student learning results in many students cramming for the tests and not putting sufficient time and effort into the assignments.

As discussed in the introduction, these kinds of concerns with the course have led us to make an initial attempt to identify TCs for introductory NA/SC courses, with the goal of leveraging these TCs to improve student engagement during the course leading to improved student takeaway afterwards.

3 Identification of Threshold Concepts for Introductory NA/SC courses

The literature on TCs suggests that there are several approaches that can be used to identify the TCs within a given body of course material. Our approach was to conduct informal interviews with faculty who have taught an introductory NA/SC course for several decades. Based on the TCs proposed in these interviews, we conducted an intersection analysis to identify the common concepts. Subsequent discussions led to a list of four proposed TCs. These TCs are listed in Table 1, along with related course topics that they encompass. These related topics are not meant to be exhaustive, but are notable ones that were mentioned during the interviews.

3.1 Pre-liminal and post-liminal states

Recall that TCs must be transformative. As explained earlier, this means that the learner typically has a preconceived notion of the TC which is much different from that of the expert. Thus, the transformation that must take place involves a process in which the learner gives up on their naïve notion of the TC (the pre-liminal state) and does the hard work of replacing their original understanding of the concept with that of the expert (the post-liminal state). This work involves both coming to understand the TC from the expert's point of view (at least to the extent that is expected by the instructor) as well as coming to accept that point of view as a replacement for the learner's own original viewpoint. As

Error	Efficiency/Accuracy	Numerical Software	Performance Analysis
Conditioning,	Iterative methods and refinement,	Problem solving environments,	Numerical experiments,
Stability,	Error Control,	Vectorization,	Interpreting results,
Truncation,	Cost-accuracy trade-offs,	Software libraries,	Convergence rates,
Approximation,	Computational and space complexity,	Low level optimized solvers,	Tables and plots
Discretization,	Exploiting structure		
Rounding	Sparsity		

Table 1: The proposed threshold concepts with related topics.

mentioned earlier in this report, while the learner is in the process of undertaking the above transformation they are said to be in a liminal state; they have not yet grasped enough of the expert viewpoint and they have not yet completely given up on the naive viewpoint.

We now describe the pre-liminal and post-liminal states for the TCs that we have identified above.

3.1.1 Error

The initial viewpoint of the learner is that an error is the result of a mistake. The student is going to perform a computation on the computer that, if it had been implemented properly, would have yielded the correct (i.e., the exact) answer, to the available precision. However, a mistake of some kind was made and an incorrect answer was obtained. The error is the difference between the computed result and the exact answer.

The expert viewpoint of error is that it is central to the computation. That is, there is no such thing as a computation that is going to lead to the exact answer. The fact that there will be errors in the computation is a certainty. Numerical computation is about attempting to control these errors when possible, and at least attempting to estimate the error otherwise. The expert knows that, in some cases, there is no hope of getting a numerical result that has a reasonably small error. Because the computations are performed using floating-point arithmetic, at the very least, round off error is inevitable. More fundamentally, many of the algorithms employed in numerical computations are approximation algorithms. This means that even if the arithmetic were exact, the computation would still not give the exact answer. And, equally fundamentally, there are some problems, ones that are poorly conditioned, where no algorithm can be expected to provide an approximation with a small error.

There is also the issue of algorithms that make sense mathematically, that is, if the arithmetic was exact, they would give a good approximate answer, but because the arithmetic is not exact, and these algorithms do not control error growth in a reasonable way, the result is a numerical result with a large

error. Such algorithms are said to be unstable. (Poorly conditioned problems are problems that exhibit sensitivity; small changes in the definition of problem result in large changes in the solution to the problem.)

3.1.2 Efficiency/Accuracy

Regarding accuracy, the initial viewpoint is that performing a computation on a computer leads to a highly accurate result. That is, whatever algorithm one implements or whatever piece of software one uses and no matter what problem one is trying to solve, the computation is going to deliver an accurate result. The initial viewpoint regarding efficiency is that there are, in some cases, multiple ways to perform the same task, and some algorithms are more efficient than others. A non-numerical example would be Selection Sort vs. Quicksort for sorting a list. The initial viewpoint would therefore be to always try to pick the most efficient algorithm. Regarding efficiency, the initial viewpoint assumes that the computation will be so quick that efficiency is not an issue. If the learner is implementing the algorithm themselves, they might choose a simpler algorithm because it will be easier to implement. The initial viewpoint assumes that numerical computations are fast and that efficiency will not be an issue.

The expert viewpoint is that the cost versus accuracy trade-off is central to numerical computation. Accuracy and efficiency are the two pillars of numerical computation. Accuracy must be framed in terms of desired accuracy or requested accuracy. That is, a given problem will come with some limits that defined the accuracy with which the problem itself is known. These would typically be the parameters that appear within the problem. And then what follows regarding the desired numerical accuracy, is the observation that the numerical solution to the problem should have an error that is less than the error associated with the parameters which define the problem. So accuracy is never thought of as absolute accuracy; that is, the desire is never to obtain the exact answer or an answer that is as close as possible to the exact answer; rather it is always about obtaining a sufficiently accurate answer; that is, attempting to perform a computation so that the error in the numerical solution is below a given tolerance. Furthermore, the error of the numerical solution should not be much smaller than the tolerance since additional accuracy is typically coupled with additional computational cost. This is a fundamental idea. Experts will use the term error control to describe a numerical algorithm that operates in this way.

Fundamental to such algorithms, is the idea of iteration based on reducing the error to an acceptable level, involving repeated and adaptive steps to solve the problem. Some form of error estimation is of course central to this type of algorithm. The iterative algorithm is repeated with adaptations until a numerical result is obtained for which the estimated error is less than the given tolerance. Efficiency is about designing or choosing an algorithm that can deliver sufficient accuracy in the least amount of time. Sometimes use of computer memory, that is, space, is also considered as a quantity to be optimized. In algorithms that are performed on parallel or distributed computing frameworks,

efficiency is also discussed in terms of minimizing communication time among processors. And finally, sometimes energy consumption is considered to be a quantity that should be optimized. Sometimes, an important part of improving the efficiency of an algorithm involves taking advantage of structure that the problem may have. For example, exploiting the sparsity structure in a problem is a common approach to improving efficiency.

3.1.3 Numerical Software

The initial view is that numerical computations are straightforward and that a simple algorithm that one might implement oneself will be fine. Alternatively, whatever software one can find within the environment one is working in will be fine. By "fine", we mean the software that we either write or use will deliver an answer that is almost exact and that it will be efficient in delivering that answer. The convenience of problem-solving environments trumps whatever modest efficiency might be gained by using software written in a language that can be compiled.

The expert viewpoint is that numerical software must be considered carefully and with, perhaps, suspicion. Naïve implementations will typically be such that they deliver or at least can deliver poor accuracy while at the same time being inefficient. A software package found within a problem-solving environment or on a website should be viewed with some suspicion. The development of high quality numerical software is an ongoing endeavour that has been underway within the numerical analysis/scientific computing community for many decades. A high quality software package will involve the use of multiple algorithms working together to give an efficient, adaptive, error-controlled result. Development of such software has typically required decades of testing and evaluation.

There is a trade-off between problem-solving environments such as Matlab or scripting languages such as Python, and the kind of efficiency that can be obtained by using software written in a compiled language. Typically the latter is more challenging to use but provides a substantial gain in efficiency.

3.1.4 Performance Analysis

Performance analysis involves the investigation of algorithms and their implementations in software to study performance with respect to cost and accuracy. The results of the performance analysis studies must be interpreted from tables and plots in order to allow the numerical software expert to assess the behaviour of the algorithm or software with respect to measures of cost, accuracy, and efficiency.

The initial view is that performance analysis is unnecessary. Since, from the naïve viewpoint, the software delivers as much accuracy as could possibly be desired and with such speed that efficiency is not a concern, there is no need to consider performance analysis of an algorithm or software.

From the expert viewpoint, every algorithm comes with a cost versus accuracy trade-off. The idea of understanding, for a given algorithm, how this

trade-off works, is central to numerical computing. A key idea here is the notion of convergence. Typically, a computation will have one or more algorithm parameters (e.g., the time step size for an algorithm that approximates the solution of a differential equation) that characterize a trade-off between the computational time required and the accuracy of the computed solution that is returned. (In the context of an algorithm for approximating the solution of a differential equation, a small time-step generally leads to a more accurate result but decreases the speed of the computation.) Since most algorithms are iterative, there is also the question of how fast the iteration converges to a sufficiently accurate solution.

4 Mapping of Traditional Course Material onto the Threshold Concepts

With the TCs in hand — see Table 1 — we then mapped the course content from a typical offering of the NA/SC course onto the TCs. This process was similar to the curriculum mapping exercises widely used in higher education [10]. Figure 1 shows the coverage of topics in the course lectures (not including tutorial time) over the term and indicates which TCs each topic maps onto. We observe that the topics related to error are heavily concentrated in the first week of the course and that error shows up fairly consistently in the following weeks, while the topic of performance analysis is touched on in the first overview topic, but does not resurface until much later in the course, and ends up only being mapped to by less than a quarter of the lecture topics. Numerical software and efficiency/accuracy are present quite consistently over the term. Lastly, we note that around 20% of the lecture topics don't map directly onto any of the TCs. These are topics that act as fundamental background mathematical knowledge that is necessary in order for the subsequent study of the mathematical problem from a numerical perspective.

This mapping exercise was also repeated for the topics covered during the tutorials in a typical offering of the course - see Figure 2. We found that performance analysis and numerical software were significantly underrepresented in tutorials. This can be attributed to the fact that tutorials are typically held in a lecture theatre setting, with the examples covered typically being pen and paper questions rather than examples intended to be done on a computer. The other two proposed TCs have good coverage over the term. There are also a significant number of topics (40%) that are not mapped. These topics again include mathematical background and relevant proofs, but also concrete examples like performing Gaussian elimination on a small system or constructing a low degree polynomial from a set of data points.

An argument could be made that small, concrete examples, while clearly helpful for students basic understanding, may be better left for students to do on their own time and instead use tutorial time for deeper engagement with the course material.

Figure 3 shows the threshold concept mapping for the weekly active learning sessions. Most notable is that the performance analysis and numerical software are both much more present than they were in the typical tutorial setting. This was largely a result of the move to online learning, which prompted the shift to using Juypter notebooks for the active learning worksheets. Online synchronous active learning sessions were held weekly, where students worked through the worksheets together in breakout rooms, with the option of asking the instructor questions when necessary. The sessions were not mandatory and many students opted to work independently, but a small core of the class consistently attended and worked collaboratively. There are still some unmapped topics present, as these topics were necessary to familiarize students with the underlying mathematics. We also went through the mapping exercise for the weekly homework - see Figure 4 - and the results look very similar to the mapping for the active learning sessions, as the worksheets were designed to be directly related to the homework the students would be handing in each week.

One of the central goals of moving away from the over stuffed curriculum did not get fully realized with these changes - partly due to the constraint of the official course description, which essentially dictates that the course must cover floating point arithmetic, numerical linear algebra, numerical solution of non-linear systems, and interpolation. Several students mentioned that they felt the topic of interpolation, which was covered last, did not receive an appropriate amount of attention. Some small efforts were made in an attempt to move away from the over stuffed curriculum, with some side topics and textbook readings omitted or left as optional readings for interested students. If we had been more aggressive in removing tangential topics from the curriculum, there would have been more time to include a more in depth treatment of interpolation.

5 Applying the Threshold Concept Framework to an NA/SC course

After we had settled on the four TCs and courses had moved online during the pandemic, one of the authors had the opportunity to apply some aspects of the threshold concept approach to an introductory NA/SC course in a summer 2020 offering. The course structure underwent the following changes to attempt to address the lack of student engagement, better highlight the TCs, and simultaneously adapt to the online learning setting:

- Lectures essentially followed the same structure, but were somewhat condensed and presented as a weekly one to one and a half hour recorded lecture, with slides posted.
- Weekly active learning worksheets were introduced; these took the form of jupyter notebooks and incorporated a mix of programming and theory, to complement the recorded lectures.

- The course made heavy use of piazza (see piazza.com) to handle student questions.
- The course was reframed in terms of emphasizing four overarching themes, which were revisited throughout the term; there was one introductory lecture that explicitly discussed the TCs and another short lecture on them halfway through the term, following the mid summer exam break.
- Online, open book exams, with generous time constraints were employed.
- The course was put on a 3 week cycle:
  - covering one textbook chapter in each cycle
  - two small weekly homework assignments each cycle and one larger assignment at the end of the 3 weeks (weekly homework and assignments included some auto-tested coding questions)
- As the term went on, we incorporated more explicit connections between the active learning exercises and what the students were required to hand in each week (in an attempt to balance their workload and in response to student feedback)
- Introduced an optional group project:
  - This did not work out too well, as it was just too much extra work for most students, given the rest of the course workload. However, almost a quarter of the class did choose to do a project and doing so helped some students connect with each other during the pandemic.

5.1 Results of the course changes

The following highlights some of the effects that these changes had on students and their engagement in the course.

5.1.1 Course evaluations

Official course evaluations from the university give some insight into how students felt about the Summer 2020 offering of the course - see Table 2. Compared with the average course evaluations for other recent offerings of the course, students found the workload to be substantially higher, but they also reported that they gained a deeper understanding of the course material and they found that the assessments were very important in helping them to improve their understanding of the course material.

5.1.2 Student Quotes

Lastly, the following quotes from students also highlight that the restructuring of the course made a positive impact on their learning experience in the course.

Table 2: Course evaluations for the Summer 2020 offering of the course, along with the average from recent offerings of the course on the main campus and a suburban campus. The scale is 1-5.

	Summer 2020	Campus	Suburban
The course provided me with a deeper understanding of the subject matter.	**4.8**	4.3	3.6
Course projects, assignments, tests, and / or exams improved my understanding of the course material.	**4.9**	4.4	3.7
Course Workload.	**3.9**	3.1	3.0
I found the course intellectually stimulating.	**4.4**	4.0	3.5
The instructor created a course atmosphere that was conducive to my learning.	**4.7**	4.2	3.5
Course projects, assignments, tests and/or exams improved my understanding of the course material.	**4.9**	4.4	3.7
Instructor generated enthusiasm.	**4.7**	4.1	3.4
Overall quality of my learning experience.	**4.6**	3.9	3.2
I would recommend this course.	**4.2**	3.8	3.1

> "The assignments and homework is designed to be tightly connected to what we learned in class and the tutorial helps me deeper understand the content and practice using what I learned to solve real issues."

Another student highlighted the emphasis on understanding over memorization in the course:

> "What I like most about your course is, it really focuses on understanding. In most courses, they emphasize memorization and speed of solving problems. But I believe understanding is more meaningful in the long term. I also like the way you split coding to homework and explanation questions to tests."

And one student highlighted that the weekly active learning and homework really helped them to succeed in the course:

> "I had tried an equivalent course before, on a different campus, but just couldn't wrap my head around a lot of the concepts until this course. I especially liked the weekly homework and worksheets that helped solidify each new concept."

6 Student Feedback on the Threshold Concepts

Since the four TCs were settled upon purely through discussions with experts in the field, we subsequently sought out how students perceived these concepts and where they were in the liminal space before and after the course. We undertook two small preliminary studies in this direction.

6.1 Study I

Our first informal study involved administering an ungraded test, consisting of five short answer questions, to students in two introductory NA/SC courses, which we will refer to as C1 and C2. Each course used slightly different wording for their versions of the test and the test was administered at two points in the course — once at the beginning and once later in the term (end of term in C1 and midterm in C2). The exact wording of each question is listed in Appendix A. Note that C1 students only earned marks for completion and C2 students earned no credit for completing these ungraded tests in their course.

We scored student answers in terms of quality from 0-4 and looked at how the average score changed over the course of the term. The results are summarized in Table 3. For C1, we had 91 observations and ran a paired t-test to test whether or not the average scores increased between the start and end of the term. The first, second, and fifth question, as well as the total scores showed significant increases in the average score. C2 had fewer students complete the surveys (39 and 16). In this case, we didn't have paired data, so independent t-tests were used and indicated only a significant increase in the second question and the total scores (although the first and fifth questions also had p-values close to .01).

Table 3: Average scores for the five test questions. C1 is only for students who completed both, while C2 is for all students who completed either test.

	Q1	Q2	Q3	Q4	Q5	total
C1 start	1.4	1.4	2.3	1.9	1.5	8.4
C1 end	2.1	2.6	2.5	2.1	2.2	11.3
C2 start	1.2	1.1	1.9	1.0	1.3	6.5
C2 midterm	1.8	3.2	2.2	1.4	1.8	10.3

Q1 was about how much accuracy one can expect from floating point operations and was the first of two questions targeting the TC of error. Initial answers were quite poor across both courses. Not as much progress was made as one might hope for, but there was some progress nonetheless. Part of the lack of progress might be explained by a possible misunderstanding by students: they may have thought the question was asking about a single floating point operation rather than an entire floating point computation.

Q2 stood out as the question that had a lot of poor answers initially but

where the most progress was made by the second time the students answered the questions. This question was about getting at the notion of numerical computing being fundamentally about approximation but it was also targeting the TC of error. The question requires students to demonstrate an understanding that one must consider the conditioning of a problem when determining whether or not one should expect the result of the computation to be accurate. Many students initially answered this question very poorly, as they clearly weren't positioned to reason about the notion of "similar" problems — with some students asserting that if the problem solved is not exactly the original problem, then the solution is meaningless.

Q3 was somewhat reasonably well done initially, possibly due to how it was phrased. The question aimed to get students to discuss their understanding of how to reason about a cost-accuracy trade-off, which directly ties into the accuracy and efficiency TC. Little progress was made and this may somewhat be explained by both courses not putting sufficient emphasis on topics such as error control during the term.

Q4 was about software for mathematical applications and was meant to gauge student understanding of the numerical software TC. The question was framed differently across the two courses. C1's version was more clearly focused on selecting mathematical software, whereas C2's version left it open for the student to more generally decide how to go about solving a mathematical problem on a computer. This distinction can be seen in the difference in scores for this question, as many C2 students scored poorly due to falling into the pre-liminal mindset where they would implement an algorithm from scratch rather than rely on existing numerical software.

Q5 targeted the final TC of performance analysis by asking students to discuss how they would go about choosing between two software packages for solving a mathematical problem. There was some progress in Q5, as more students had some understanding of what to look for when considering the performance of numerical software — in particular, the idea of checking for accuracy and not just speed.

A general observation we had regarding the student responses was that some students were clearly trying to incorporate terminology they had learned in the course when they answered the questions for the second time; however they often used the terms incorrectly or at least imprecisely. Some examples from C1 of terminology that only appeared in student end of term answers included "machine epsilon", "well-conditioned", "catastrophic cancellation", "numerically stable", and "absolute error".

6.2 Study II

In our second small study, we invited students from three previous offerings of a introductory NA/SC course at our university (two offerings at our main campus and one offering at our suburban campus) to complete a short survey. Eighteen students completed the survey (eight of which were from our summer 2020 offering of the coruse) and five of them (four from our summer 2020 offering)

volunteered to participate in a thirty minute followup interview. In the survey, students were asked a series of questions probing what they considered to be the TCs for the course. Only a very brief description of what constitutes a TC was presented to the students, so that must be taken into consideration when considering student responses.

6.2.1 Student identification of threshold concepts

One of the last questions on the survey asked students to rate how strongly (on a 5 point Likert-type scale) they felt that each concept from a list of course topics was a TC. The results for this question are summarized in Figure 5. To better facilitate discussion of our results, we have mapped each listed course concept onto the TC that it is most closely related to.

The top three topics are consistent with the Error TC. Topics related to accuracy and efficiency are somewhat positively viewed as potential TCs by students, although we see that topics like efficiency and computational complexity have students on both ends of the scale. This may be due to differences in programs of study. For example, computational complexity may be more likely to be identified as a TC in a numerical methods course by a computer science major than by a mathematics or physics major. In the interviews, students provided some further insight into this — mentioning that the notion of computational efficiency is prominent in computer science theory courses, but that the new piece is that the accuracy of the computation needs to also be considered.

Topics related to performance analysis show up across the board. Students largely felt that visualizations of results and interpreting numerical results were both TCs. Producing and formatting tables was somewhat low on the list, but again there were students on both sides. Discussions in the interviews somewhat clarified that while performing experiments, generating visualizations, and interpreting results is important to succeeding in the course, it is not really a new idea to students, but rather something that simply is not required in most of their other computer science courses. Some students did say that they had done similar work with interpreting results in courses like machine learning and in the interviews one student drew the parallel to applied science courses, where experiments are run and lab reports are written.

Topics related to numerical software appear fairly low on the list. This may be somewhat due to current limitations on the course largely due to the minimal prerequisites for the course we highlighted previously. A result of only requiring minimal programming experience in Python is that we can not assume knowledge of languages like C or Fortran and as a result we do not tend to explore the low level details that arise when working with detailed implementations of numerical methods. In our summer offering of the course, we made an effort, through several timing experiment exercises, to emphasize the importance of using built-ins for operations like matrix multiplication rather than implementing them from scratch. In the survey and interview, one student confirmed that this really stood out as something they took away from the course and would

make them a better programmer.

In several of the interviews, students discussed their thoughts about numerical software and their experience with it in the course. One student, who had a substantial programming background coming into the course, said that he didn't have any trouble picking up what was needed for the course and was happy not to have to code too much himself. For that student, Scipy (a Python library for scientific computing that was used in the course) was "just another software package" and he felt that upper year CS students already prefer to not code things themselves. Another student who only had the minimum required programming background offered a somewhat different perspective. He found learning Scipy very intimidating at first and was happy that a lot of the code was provided, especially early on in the course. In terms of the idea of using existing numerical software instead of implementing an algorithm from scratch, he said that he already had the notion that "people have done things way better than you possibly ever could have", but that the course "reinforced [his] understanding of that perspective".

The last survey question asked students to provide a list of up to five TCs. The results are similar to those from the previously discussed question, but also included course topics like root finding, LU factorization, and interpolation. It is our opinion that these topics themselves do not constitute being elevated to the status of being TCs, but rather are viewed as something new from a student's perspective. Such topics are covered in the course but in service to the more fundamental TCs.

One student made an important observation regarding the fact that the TCs for a course could be different, depending on the backgrounds of the students:

> "... some of the above concepts might be TCs for a generic learner, but are either ones that students taking [this course] have likely encountered already or aren't TCs in the context of a numerical methods course."

6.2.2 Student identification of transformative concepts

When prompted to write about what they found most transformative in the course, one student wrote:

> "Machine representable floating-point precision, and how it affects many computations we take for granted in mathematics. Asymptotic runtime and rate of convergence (if applicable) of mathematical methods like Gaussian Elimination for solving linear equations, precision/runtime investigation of evaluating a polynomial, Newton's method, etc. [...] In math we mostly just learn why these methods work mathematically, but not how well they work runtime-wise or numerical precision-wise."

When prompted to write about something in the course that challenged their previous understanding, students provided answers highly consistent with our previous description of the pre-liminal state of someone learning about error:

"The concept of stable and unstable algorithms challenged my previous understanding of computing. I always thought that the computer could calculate the result stably and quickly, and would always output the correct answer until I learned the concept of Taylor expansion, floating-point system and saw a video about the halting problem."

"I found that the section on solving linear systems to be most transformative because it took something that we were familiar with and elaborated on it, giving insight on how code for solving it would work and what problems might occur and how it can be made more efficient."

"To me, the most transformative concepts were the algorithms for LU factorization and Gaussian Elimination (such as with partial pivoting), and finding roots/fixed points of a functions. Although I was expecting to just see a rehash of Linear Algebra concepts when the course reached the topic of Gaussian Elimination and LU factorization, I was surprised to see that much of what is discussed about Gaussian Elimination is not even about how to carry it out from a mathematical point of view, but instead how to optimize it for properties like precision. Furthermore, I did not know that LU factorization could be used to lessen the amount of operations required to solve multiple systems of linear equations."

"Discretization (realizing that you can't ever model a math problem perfectly (you can't even hope to hold countably infinite items)."

6.3 Concluding remarks on the two studies

The studies represent preliminary attempts to investigate how students perceive the TCs. It is clear that students are in various liminal states with respect to the TC and in some cases are not able to distinguish "interesting topics" from TCs. This is not surprising for at least two reasons: (i) the students were not given time to develop much of an understanding of the definition of a threshold concept, and (ii) it can take many years of working with and teaching a given body of material in order to be able to discern the fundamental, transformative, wide-reaching concepts that characterize TCs.

7 Summary and Conclusions

We have reported on an investigation into an approach for improving student engagement and take away in an introductory NA/SC course. A starting point for our work has been that an over stuffed curriculum necessarily implies superficial learning. We have proposed the use of the well-known threshold concept

framework in order to select from the standard curriculum a smaller set of essential concepts that are then featured within the course. The threshold concepts we identified are Error, Efficiency/Accuracy, Numerical Software, and Performance Analysis. For each of these, we described a learner's pre-liminal and post-liminal state.

We have described an offering of our introductory NA/SC course where we incorporated active learning and a threshold concepts based curriculum. We provided preliminary evidence that students found that the course provided them with a deeper understanding of the course material.

We also discussed the results of two surveys conducted in several of our introductory NA/SC courses where we deployed a preliminary threshold concepts based curriculum. In the first study, we asked students to answer questions that were meant to gauge their understanding of the threshold concepts. We found that the students exhibited an intermediate level of success in transitioning through some of the threshold concepts, while little progress was made in others. *As instructors, we understand that more deliberate attention to the featuring of threshold concepts in our courses will be necessary in order for us to observe a stronger imprint of the threshold concepts in our student populations. Table 3 is diagnostic; it tells us which threshold concepts are being learned best and which require more attention.* It also provides some insight into where students are situated within the liminal space coming into the course.

The second survey directly asked students about what they thought were the threshold concepts in their introductory NA/SC course. Student responses were generally consistent with the threshold concepts we have proposed, but, in some cases, topics that students identified as threshold concepts, are not concepts that we feel satisfy the criteria for a threshold concept. We feel that this is not surprising; the analysis to determine the threshold concepts for a course requires considerable depth of understanding and perspective and this is not something that most students will have acquired even by the end of a typical course.

The value of working from a threshold concepts informed curriculum is that it allows time for improved focus on the topics that will reinforce that threshold concepts. This time can be used to introduce the important tool of active learning into the deliver of the course content. As well, authentic assessment instruments can be employed to further emphasize the threshold concepts.

8 Future Work

There are several directions for future work. The identification of threshold concepts within the standard course curriculum for a given course is not a straightforward process. We feel that it should be viewed as an iterative process where one identifies a candidate set of threshold concepts, implements them in a course designed to feature these threshold concepts, and then observes how these concepts are received and learned by the students. It is quite conceivable that a modification of the set of threshold concepts might be necessary in order

to appropriately tune them to the student population. Therefore, one future project involves revisiting our analysis of the threshold concepts to investigate whether some changes or refinements may be necessary based on what we learn from our students.

We suggested earlier in this paper that one of the major advantages of the use of a threshold concept based curriculum is that it allows more time to feature the threshold concepts within the course. An important aspect of this, mentioned in the Introduction, is that the powerful tool of active learning can be used to provide rich learning experiences for students. We plan to further develop active learning exercises that will invoke deeper learning experiences for our students, based on the threshold concepts we have identified.

We are challenging our students to acquire a deeper understanding of the concepts we have identified as being threshold concepts. In order for this to happen, *it is essential that the evaluation instruments, e.g. assignments, tests, exams, etc., be reflective of the deeper level of understanding that we are asking of our students.* While we have made some preliminary efforts to move our assessment instruments in this direction, further work is required to develop high-quality, authentic evaluation instruments that are consistent with the deeper levels of understanding that we are expecting our students to attain.

As described earlier in this report, the analysis that was used to discern the TCs for an introdcutory NA/SC course was based on discussion between experts whose perspectives are based on years of university level teaching experience. What is missing, and what we hope to better incorporate, is the student perspective. We are therefore interested in answering questions like:

- How do undergraduate students perceive the threshold concepts taught in an introductory NA/SC course?
- Coming into a course like this, where in the liminal space are the students? Have they already been exposed to the concepts or are they purely in the pre-liminal state?
- What is the most effective way to guide students through the liminal space and how can we best identify when a student has sufficiently grasped a threshold concept?

References

[1] Lorin W Anderson and David R Krathwohl. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, 2001.

[2] Lee Averell and Andrew Heathcote. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psycholoogy*, 55(1):25–35, 2011.

[3] Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, 20(24), 1956.

[4] Marta Caligaris, Georgina Rodríguez, and Lorena Laugero. A first experience of flipped classroom in numerical analysis. *Procedia-Social and Behavioral Sciences*, 217:838–845, 2016.

[5] Glynis Cousin. An introduction to threshold concepts. *Planet*, 17(1):4–5, 2006.

[6] J.H.F. Meyer and Ray Land. Threshold concepts and troublesome knowledge: Linkages to ways of thinking and practising within the disciplines. *Improving Student Learning - Ten Years on*, pages 412–424, 01 2003.

[7] Fred M Newmann, M Bruce King, and Dana L Carmichael. Authentic instruction and assessment. *Lowa: Departement of Education*, 2007.

[8] Kate Sanders and Robert McCartney. Threshold concepts in computing: past, present, and future. In *Proceedings of the 16th Koli Calling international conference on computing education research*, pages 91–100, 2016.

[9] Julie A Timmermans and Ray Land. *Threshold Concepts on the Edge*. Koninklijke Brill NV, Leiden, The Netherlands, 2020.

[10] Kay Pippin Uchiyama and Jean L Radin. Curriculum mapping in higher education: A vehicle for collaboration. *Innovative Higher Education*, 33(4):271–280, 2009.

[11] Maryellen Weimer. *Active Learning: A Practical Guide for College Faculty*. Magna Publications Incorporated, 2017.

[12] Francisco José Correa Zabala, Heidi E Parker, and Camilo Vieira. Implementing an active learning platform to support student learning in a numerical analysis course. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE, IEEE, 2017.

A Questions from Survey I

A.1 Q1

A.1.1 Course 1 version

Suppose you performed some computations to obtain an answer to a mathematical problem using a computer that performs arithmetic using 8 decimal digits. Assuming you don't make data entry or programming mistakes during the computation: How many correct digits would you expect to have in the result of the computation? What would you call the difference, if any, between the exact answer to the problem and the computed result that you obtained? By 'exact answer', I mean the answer you would get if all the arithmetic was done exactly instead of using 8 decimal digits.

A.1.2 Course 2 version

If you were to perform a computation to obtain an answer to a mathematical problem on a computer that performs arithmetic using 8 decimal digits, and assuming you don't make mistakes during the computation, how many correct digits would you expect to see in the result of the computation? What would you call the difference, if any, between the exact answer to the problem (i.e., the answer you would get if all the arithmetic was done exactly) and the computed result that you obtained?

A.2 Q2

Suppose that you asked someone to solve a mathematical problem using a computer and, after performing their computations, they came back to you with a result. They told you that the result they obtained is not the exact solution to the problem you gave them but rather that their result is the exact answer for a problem that is very close to the problem you gave them. Would you be OK with the result they gave you?

A.2.1 Course 1 version

You know that the result they have given you is the exact answer for a problem that is really close to your original problem. Can you conclude that the result they gave you is really close to the exact answer to your original problem?

A.2.2 Course 2 version

Since the result they have given you is the exact answer for a problem that is really close to your original problem, is the result they gave you really close to the exact answer to your original problem? Discuss.

A.3 Q3

Suppose you have an algorithm that has a parameter that controls the amount of accuracy that the algorithm is to deliver when it is used to solve a problem. The algorithm takes a longer time to solve a problem when more accuracy is required.

A.3.1 Course 1 version

What factors might you take into account when choosing the accuracy parameter for solving a given problem?

A.3.2 Course 2 version

Discuss how you would choose that parameter in order to solve a given problem.

A.4 Q4

A.4.1 Course 1 version

Suppose that you are working on the development of mission critical software for a given task. Within that task there is a subproblem that requires the solution of a mathematical problem. What things will you take into consideration when choosing software for the solution of the mathematical problem?

A.4.2 Course 2 version

Suppose you are working as a software developer and it turns out that in order to perform a certain task, you find that you need to solve a linear system of equations. It is easy enough to find out from a quick search that there is a common algorithm, called an elimination algorithm, that can be used to solve a linear system. In your role as a software developer, how would you go about dealing with the problem of having to solve linear systems within the larger project that you are working on?

A.5 Q5

Suppose that you are considering two software packages, both of which claim to be able to solve a given mathematical problem. Suppose further that the software will be embedded within a larger package and that different versions of the mathematical problem will have to be solved many times when the larger package is running.

What tests would you perform on the packages that claim to be able to solve the mathematical problem in order to help you decide which one to use?

B Sample Test Question

The question shown in Figure 6 was delivered as part of an online, 2 hour, open book midterm. It requires students to express a thorough understanding of the concepts involved.

In retrospect, it would have been better if the question had indicated which function corresponded to each plot, as that would have further solidified that the emphasis was to be on the quality and completeness of their explanations, rather than on simply matching the functions to the plots.

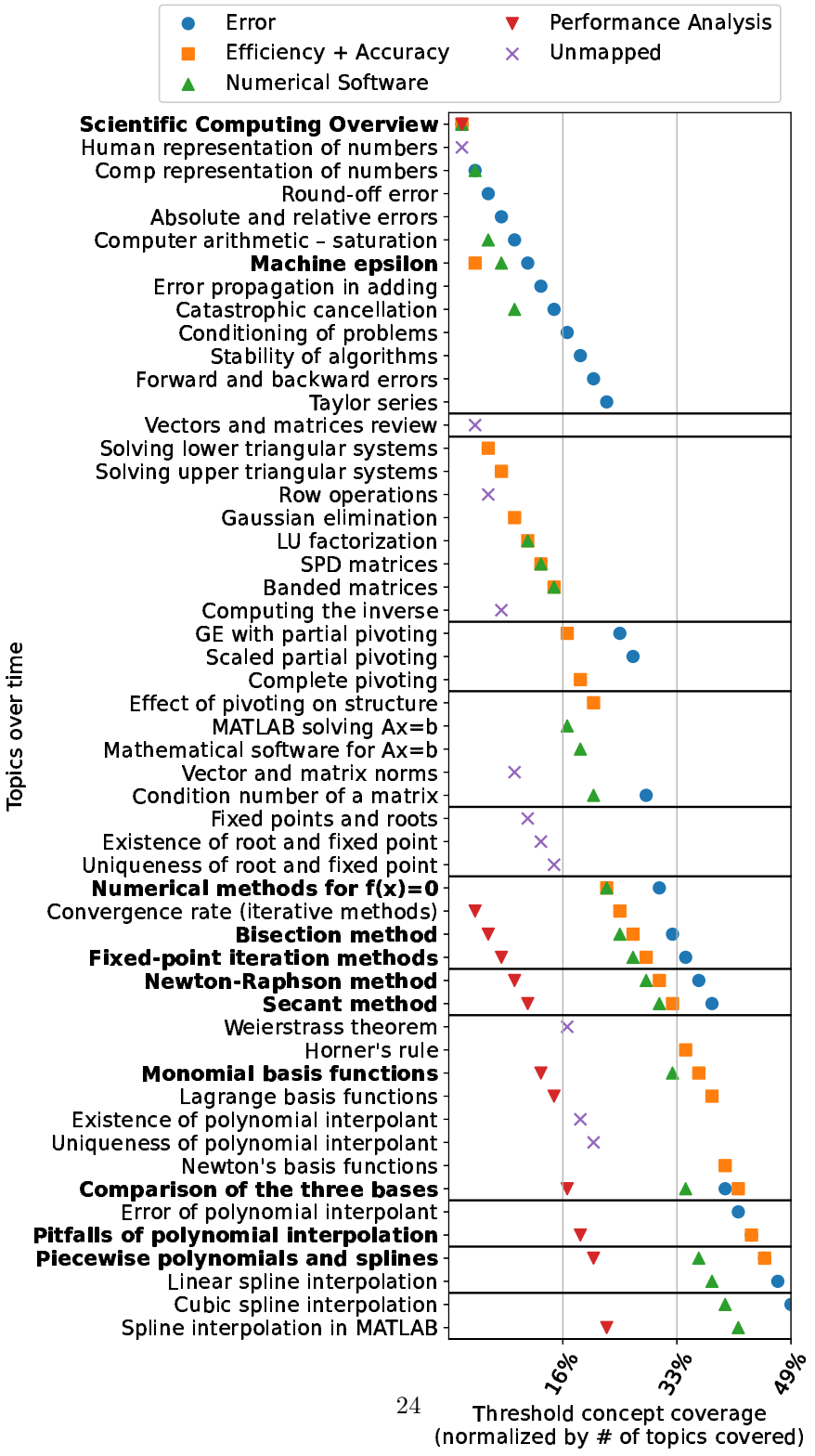


Figure 1: Coverage of threshold concepts during traditional lecture time in a typical offering of the introductory numerical methods course. Horizontal lines delimit weeks in the course. Bolded topics incorporate at least three of the threshold concepts.

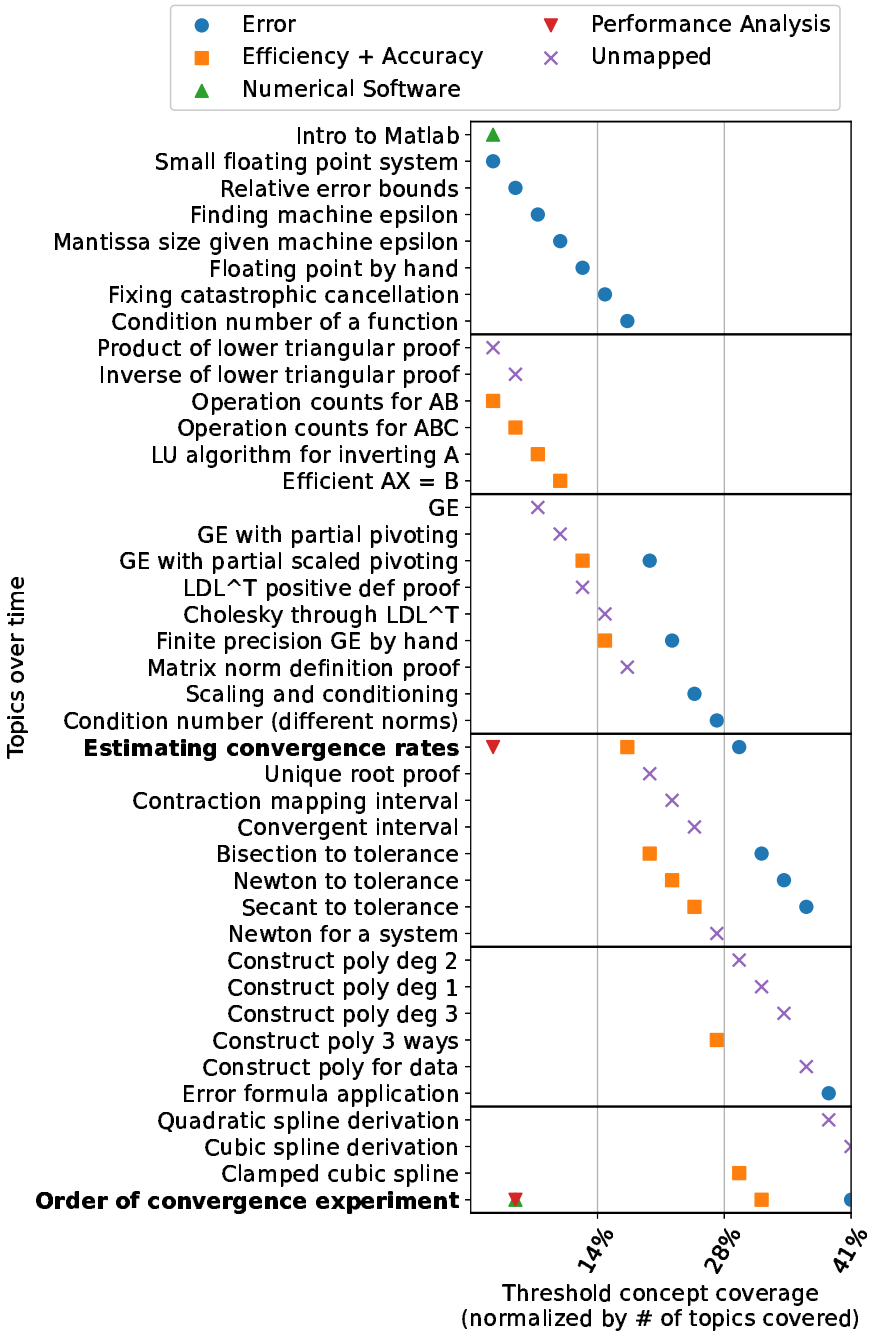


Figure 2: Coverage of threshold concepts during tutorials in a typical offering of the introductory numerical methods course. Horizontal lines delimit weeks in the course. Bolded topics incorporate at least three of the threshold concepts.

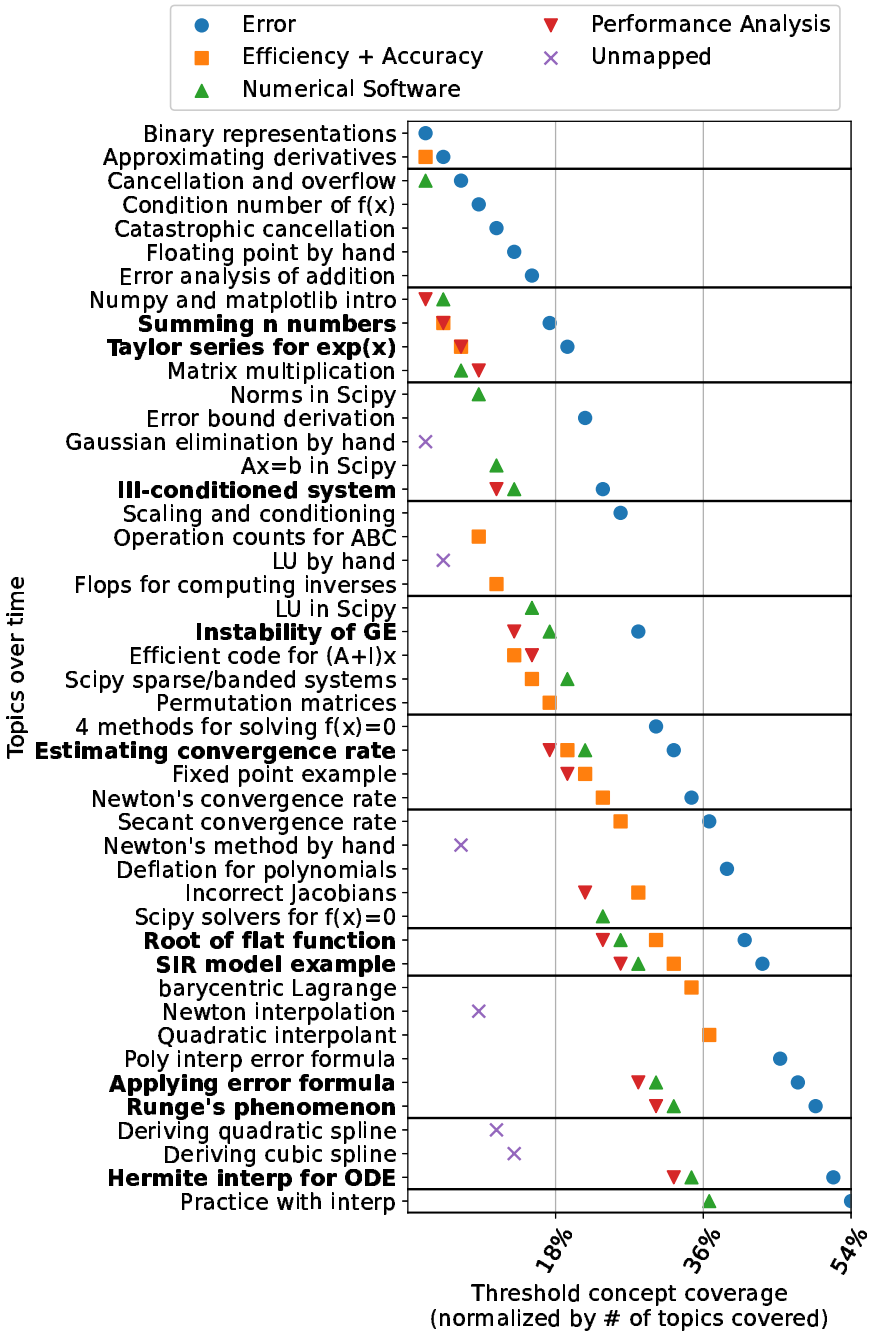


Figure 3: Coverage of threshold concepts during active learning sessions in the summer offering of the introductory numerical methods course. See Figure 1 for additional explanation

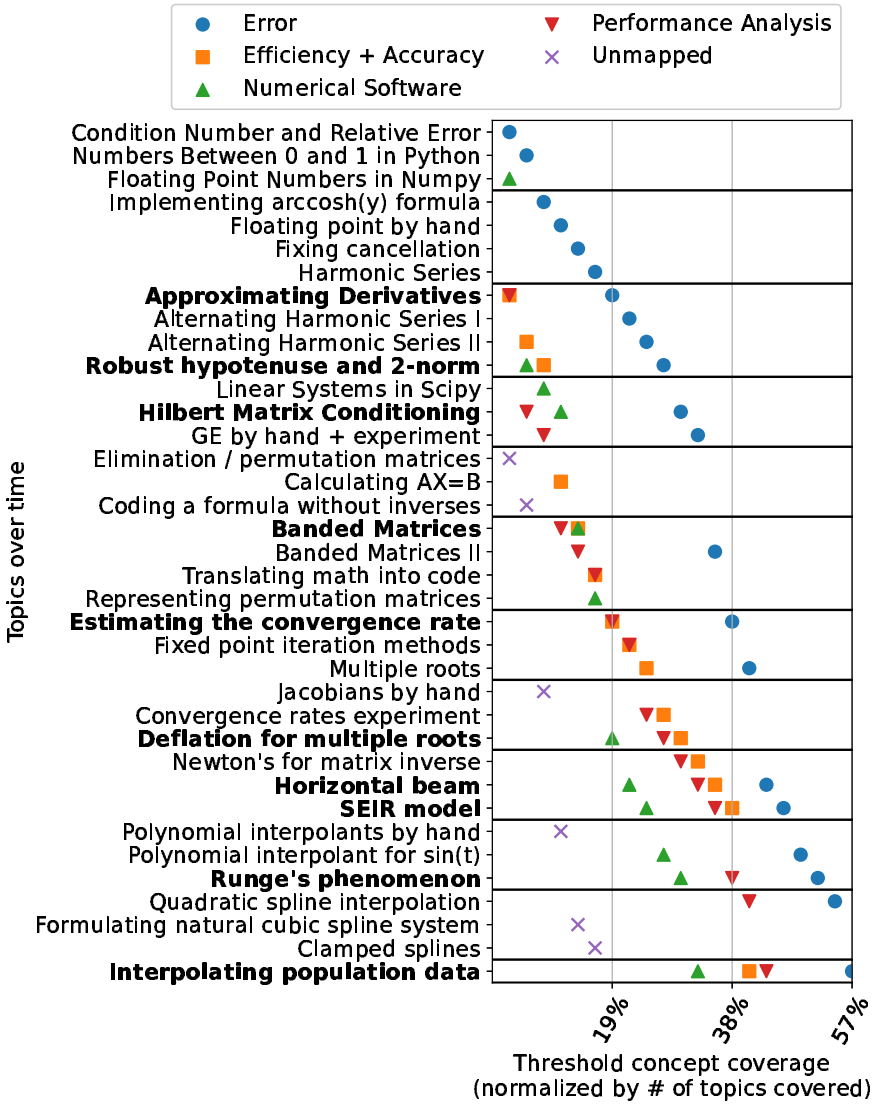


Figure 4: Coverage of threshold concepts in weekly homework during Summer 2020 offering of an introductory NA/SC course. Horizontal lines indicate weeks in the course. Bolded topics incorporate at least three of the threshold concepts.

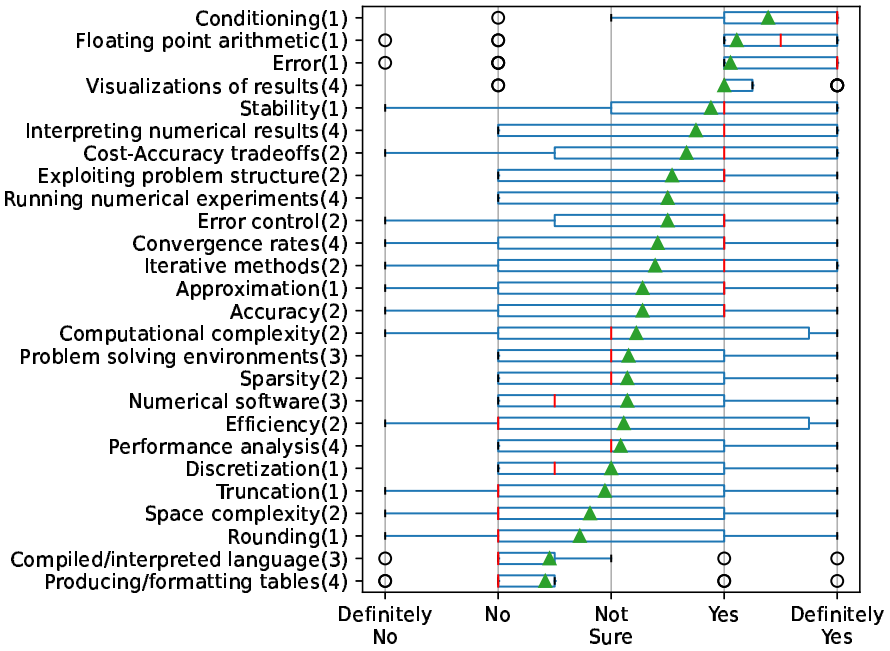


Figure 5: Boxplot of potential threshold concepts ordered by how strongly students indicated they felt they were threshold concepts. The number in brackets indicates which threshold concept the topic or concept is most closely associated with. Triangles denote the average response (based on linear mapping from -2 – 2)

Your friend decided to experiment with your code from Assignment 1, where we investigated the two finite differences formulas and the complex step method for approximating the derivative of f(x). He was curious to see what would happen for several other test functions:

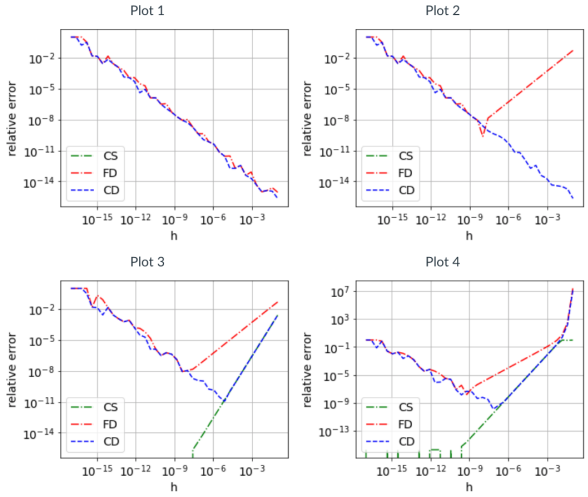
(a) $f(x) = x$, at $x = 1$

(b) $f(x) = e^{200x}$, at $x = 1$

(c) $f(x) = \frac{1}{1+x}$, at $x = 1$

(d) $f(x) = x^2$, at $x = 1$

For your reference, these are the four plots your friend produced:



In the plots, FD means forward differences, CD means centred differences, and CS means complex step method.

1. Match each function to its plot. **Justify your answer and convince us why your answer is correct**. Be as specific as you can and use terminology from the course as appropriate. Please be clear and concise - unnecessary or incorrect details hurt an otherwise perfectly good answer.

Figure 6: Sample Exam Question