ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE

Paper ID #36870

Innovating and modernizing a Linear Algebra class through teaching computational skills

Mariana Silva (Teaching Associate Professor)

Mariana Silva is a Teaching Associate Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Silva is known for her teaching innovations and educational studies in large-scale assessments and collaborative learning. She has participated in two major overhauls of large courses in the College of Engineering: she played a key role in the re-structure of the three Mechanics courses in the Mechanical Science and Engineering Department, and the creation of the new computational-based linear algebra course, which was fully launched in Summer 2021. Silva research focuses on the use of web-tools for class collaborative activities, and on the development of online learning and assessment tools. Silva is passionate about teaching and improving the classroom experience for both students and instructors.

Philipp Hieronymi

Matthew West (Prof.) (University of Illinois at Urbana - Champaign)

Nicolas Nytko

Akshit Deshpande

Akshit is an undergraduate student at the University of Illinois at Urbana-Champaign studying computer science and applied statistics. He has taught as a course assistant for introductory computer science, computational linear algebra, and numerical methods courses. He is a former software engineer intern at Intuit and Google, and is currently a software engineer intern at Meta.

Jer-Chin Chuang

Sascha Hilgenfeldt

Innovating and modernizing a Linear Algebra class through teaching Computational Skills

Introduction

In the curriculum of most leading universities both within the US and abroad, linear algebra is considered one of the pillars of mathematical instruction in the engineering and science disciplines. It is therefore taught very early on, in parallel with elementary calculus, and students are familiar with concepts such as vectors, matrices, and linear spaces from the very beginning. In this paper, we describe the design choices and implementation guidelines adopted during three years of major innovation and overhaul of an existing linear algebra course (called the *Traditional* course in the following) offered to engineering students at the University of Illinois at Urbana-Champaign.

The Traditional linear algebra course was listed as an upper-level course, following the calculus sequence targeted to freshmen and sophomores. This resulted in a significant portion of engineering students only taking linear algebra later in the curriculum, despite the fact that most of its content requires only Calculus 1 as a prerequisite. Consequently, instructors teaching upper-level engineering courses which require a linear algebra background had to provide students with review material, taking time away from the actual course content. Even mid-level classes were hampered by not being able to use elegant linear algebra results, having to resort to more cumbersome methods. Moreover, even though the traditional course was advertised as an applied linear algebra course, it lacked the use of either modern teaching approaches or modern tools to familiarize students with solutions to real-world applications.

Examples from other educational institutions have shown promise for modernizing the engineering linear algebra curriculum, often with a focus on strengthening the use of computational tools. A Python-based linear algebra course was developed at China University of Geosciences, in which Jupyter notebooks were designed to introduce students to physical representations of linear algebra concepts [1]. A qualitative survey reported an overwhelmingly positive opinion of the course by students. At the University of Pittsburgh, chemistry research was becoming increasingly reliant on computational tools, but the chemistry curriculum failed to provide a sufficiently substantial computational background. Thus, scientific computing was integrated with a mathematics course designed for chemistry students via Jupyter notebooks [2]. The modernized class focused on computing tools specifically relevant to chemistry, in place of the plethora of topics covered in traditional computer science courses; this resulted in an overall improvement in student opinion toward programming and Jupyter notebooks. These studies suggest that introducing computational tools can serve as a focal point for an effective modernization of a linear algebra course, while also providing a foundation for students' future computational endeavors.

The present project proposed a new linear algebra course, called the *Computational* course in the following, targeted to freshmen and sophomores and including the use of modern computational tools such as Python. The specific objectives were the following: (1) establish a stronger integration of basic programming knowledge into the general engineering curriculum; (2) introduce computational exercises that solve "real-world" linear algebra problems, emphasizing the importance in connecting fundamental and applied concepts in modern mathematics instruction; and (3) incorporate collaborative learning activities that motivate students to learn mathematics and promote an environment for them to develop social and communication skills. The linear algebra course was redesigned through a Community of Practice consisting of faculty from Engineering and Mathematics Departments, collaborating closely to design the new curriculum and implement the changes in stages, and with a view to sustainability of the project.

This effort was begun in the fall of 2019, and has led to a successful, stable implementation as an approved new course by the fall of 2021. Figure 1 shows that student enrollment numbers have not only been moved from the traditional to the computational class, but that the new course is attracting additional students from departments that had not implemented a comparable linear algebra class in their curricula before.

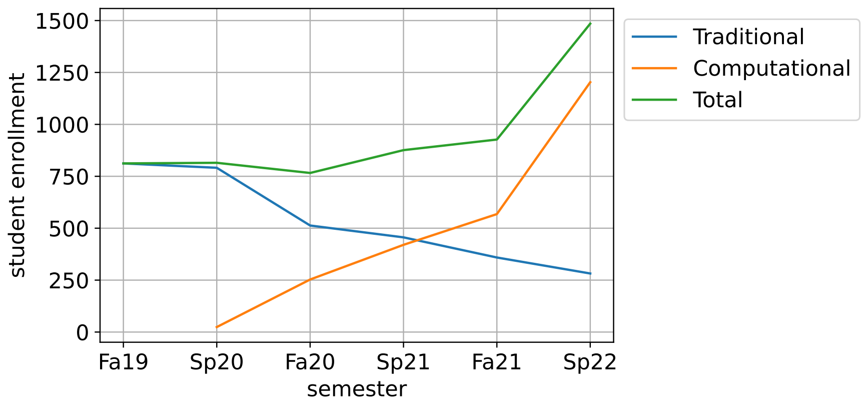


Figure 1: Student enrollment for the Computational and Traditional linear algebra courses, as well as the added total enrollment, demonstrating the growth of the modernized computational class.

Section details the course changes and the means by which they were achieved, while Section describes their implementation and reception so far. Section provides conclusions and a perspective on the broader impact of this project, and others like it, in the future.

Methods

Changes via a Community of Practice

This project was funded by the Grainger College of Engineering Strategic Instructional Innovations Program (SIIP) that promotes the sustainable adoption of evidence-based pedagogies by forming Communities of Practice (CoPs) consisting of instructors who are committed to the reform effort [3, 4, 5, 6]. While these CoPs can be comprised of any number of enthusiastic lecturers and junior faculty, these CoPs are required to also include senior faculty who can broker the departmental buy-in to sustain innovations and reforms. Further, these CoPs are encouraged to meet weekly. During these weekly meetings, representatives from the SIIP executive teams, comprised of faculty experienced in evidence-based educational methods, facilitate the development and implementation of reforms. This model of collaborative development provides avenues to secure faculty buy-in, organically spread effective practices, facilitate evaluation, and provide just-in-time training for faculty. Following these guidelines, the first team committed to this project consisted of two experienced tenured professors in Mechanical Engineering who frequently teach courses that require or benefit from linear algebra as a prerequisite, one teaching professor in Computer Science experienced in integrating collaborative learning activities into lectures, and one junior professor in Mathematics enthusiastic about using modern tools to help students solve real-world applied problems in the course.

Re-structuring the course content

Our first goal was to re-structure the curriculum of the traditional linear algebra course (which had 3 lecture hours per week), such that the theoretical components could be taught over 2 lectures per week (Mondays and Wednesdays), and the examples and application components over 1 lecture per week (Fridays). Instead of presenting the examples and application components using an expository teaching style, we transformed the third lecture hour into a computational lab following a flipped classroom style, where students worked on computer-based activities applying linear algebra concepts to solve real-world examples. These examples were first introduced during the Monday/Wednesday lectures to highlight how applications are strongly connected with theoretical concepts, with the intent to create a two-way flow between lecture and labs. The real-world examples were designed to both motivate students regarding the utility of linear algebra, and to help them integrate linear algebra concepts in practical settings. It should be emphasized that the reduction in lecture hours only very minimally reduced the scope of the course, as the stronger focus on applied aspects allowed for (i) streamlining and excising more purely mathematical elements such as proofs, and (ii) directly teaching many topics with the help of computational tools. Table 1 contains the list of topics for the lecture and computational lab for the new linear algebra course, demonstrating that it retains a comprehensive introduction to important topics while coordinating closely with the content of the computational exercises. In addition to the introduction of the computational lab, we also modified the teaching style of the discussion sections (which continued to be held for 1 hour per week). In the traditional class, the teaching assistants assigned to discussion sections were expected to solve a set of given problems from a worksheet on the blackboard, while students copied the solution. According to feedback, students did not feel they benefited much from these sections, and therefore attendance was low. For the computational course, the worksheets were re-designed to include more guidance, making it easier for students to work in teams to solve the problems, while teaching assistants work as facilitators. Table 2 summarizes the comparison between the Traditional and Computational course structures.

	Lecture 1	Lecture 2	Computational Lab
Week 1	Introduction to linear systems, matrices	Echelon forms, Gaussian Elimination	Python tutorial
Week 2	Vectors, linear combination, spans	Matrices, matrices-vector multiplication	Working with Vectors: color mixing as linear combination
Week 3	Matrix multiplications, matrix transpose	elementary matrices, triangular matrices, LU-decomposition	Matrices as images
Week 4	Inverse of a matrix	Solving linear systems with LU	Midterm 1
Week 5	Vector spaces and subspaces, column spaces and nullspaces	Abstract vector spaces, linear independence	Spring systems: solve using LU
Week 6	Basis and dimension	The four fundamental subspaces, orthogonal complements	Graphs and Algebraic Graph Theory
Week 7	Coordinates, Orthogonal bases	Linear transformations, coordinate matrix	Data Compression
Week 8	Determinants, Cofactor expansion	Eigenvalues and Eigenvectors	Midterm 2
Week 9	Computing eigenvectors and eigenvalues	Properties of eigenvectors, Markov matrices	Markov Chains
Week 10	Diagonalization, powers of matrices	Matrix exponential, orthogonal projection onto lines	Dynamical systems
Week 11	Orthogonal projections onto subspaces, least squares	Linear Regression	Linear Regression
Week 12	Gram-Schmidt process, QR decomposition	Spectral Theorem	Midterm 3
Week 13	SVD, low-rank approximations	pseudo-inverse, least-squares solutions via SVD	SVD and applications
Week 14	Principal Component Analysis	Review complex numbers, complex linear algebra	Principal Component Analysis

Table 1: Curriculum of the modernized Computational linear algebra course.

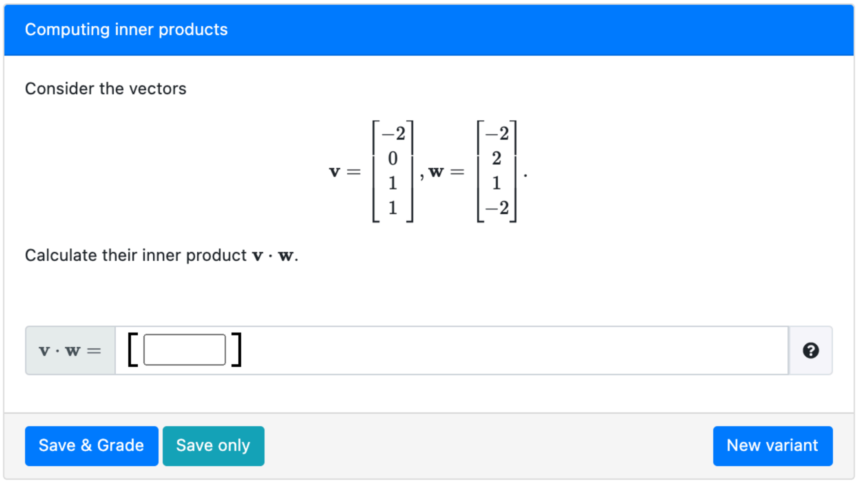
Computer-based assessments

All the homeworks and exams from the Traditional linear algebra course were moved to our homebrew online assessment platform called PrairieLearn [7], designed to facilitate learning to mastery. Within the PrairieLearn platform, students are able to practice solving randomized

	Traditional	*Computational*
Lecture	3 hours/week	2 hours/week
Discussion	1 hour/week (TAs solving examples on white board)	1 hour/week (students working on collaborative activities)
Computational Lab	–	1 hour/week
Credit Hours	3	3
Prerequisites	Calculus 3	Calculus 1 and Introduction to Programming

Table 2: Comparison between the structure of the original linear algebra course (Traditional) and the re-designed course (Computational).

problem variants repeatedly until mastery, receiving immediate feedback about their current mastery level. Figure 2 illustrates how question variants appear on a homework or exam. In this example, the vectors $\mathbf{v}$ and $\mathbf{u}$ are randomized for each student, providing a unique version of the question. In addition, when the question appears in a homework, students can use the "New variant" button to generate a new version of the question for additional practice.


Computing inner products

Consider the vectors

$$\mathbf{v} = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} -2 \\ 2 \\ 1 \\ -2 \end{bmatrix}.$$

Calculate their inner product $\mathbf{v} \cdot \mathbf{w}$.

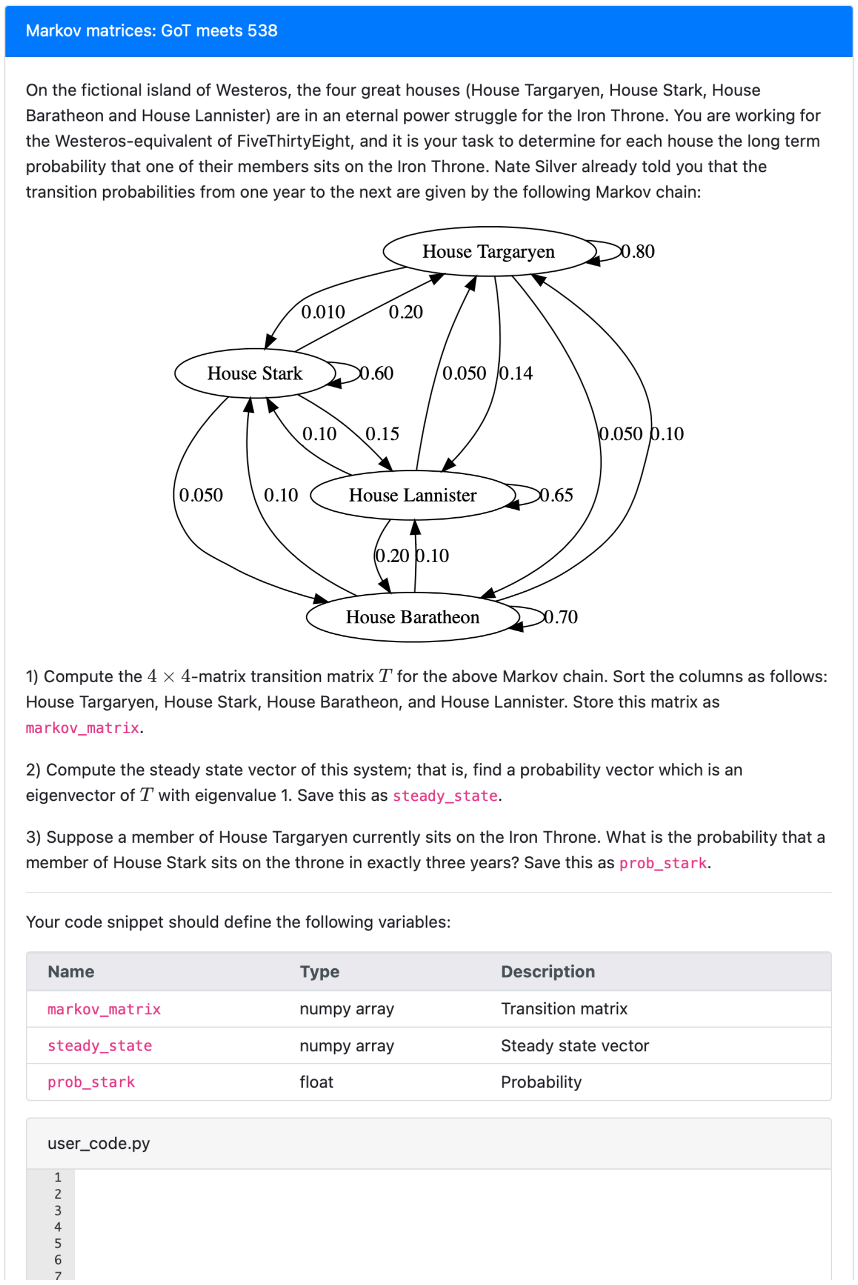
$\mathbf{v} \cdot \mathbf{w} =$ [ ]

Figure 2: Example of question variant from PrairieLearn.

Figure 3 shows an an example for a homework question closely related to content presented during the computational labs. Note that students are presented with a real application of linear algebra here, where the solution can be constructed using simple Python tools introduced in the lab section.

Collaborative learning

Collaborative learning is an evidence-based instructional practice that has been adopted by many instructors in STEM courses in higher education. Research indicates that engaging students in collaborative activities is associated with increased student persistence and improved student



Markov matrices: GoT meets 538

On the fictional island of Westeros, the four great houses (House Targaryen, House Stark, House Baratheon and House Lannister) are in an eternal power struggle for the Iron Throne. You are working for the Westeros-equivalent of FiveThirtyEight, and it is your task to determine for each house the long term probability that one of their members sits on the Iron Throne. Nate Silver already told you that the transition probabilities from one year to the next are given by the following Markov chain:

1) Compute the 4×4-matrix transition matrix T for the above Markov chain. Sort the columns as follows: House Targaryen, House Stark, House Baratheon, and House Lannister. Store this matrix as `markov_matrix`.

2) Compute the steady state vector of this system; that is, find a probability vector which is an eigenvector of T with eigenvalue 1. Save this as `steady_state`.

3) Suppose a member of House Targaryen currently sits on the Iron Throne. What is the probability that a member of House Stark sits on the throne in exactly three years? Save this as `prob_stark`.

Your code snippet should define the following variables:

Name	Type	Description
`markov_matrix`	numpy array	Transition matrix
`steady_state`	numpy array	Steady state vector
`prob_stark`	float	Probability

user_code.py

Figure 3: Example of a coding question from PrairieLearn.

learning outcomes and retention [8, 9]. Successful and productive collaborations are rarely guaranteed, but they can be greatly improved by (a) careful design of the task [10, 11], (b) assignment of team roles [12] and (c) the use of available technologies to both promote collaborations among students and support the instructors implementing these activities [13, 14].

(a) Careful design of tasks. We adopted the POGIL (Process Oriented Guided Inquiry Learning) [15] instructional strategy to build the material of the computational labs, which are based on a student-centered approach where students work in small teams and the instructors act as facilitators. As such, all the computational labs include detailed explanation of the problem, provide guided steps towards the solution, and create opportunities for open discussion among students. We created the lab exercises using Jupyter notebooks [16], because they offer an interactive computational environment which combines code, text, figures, mathematical computation, and plots. The creation of these notebooks for the re-designed course is described in more detail below.

(b) Assignment of team roles. During the POGIL activities, we encourage students to self-assign roles among team members, and alternate these roles on a weekly basis. We proposed three structured roles: the Recorder is assigned to write the team's answers to problems, the Manager is responsible for keeping the team on task, and the Reflector ensures nobody in the group is lagging behind and provides feedback to instructors about the task and team interactions.

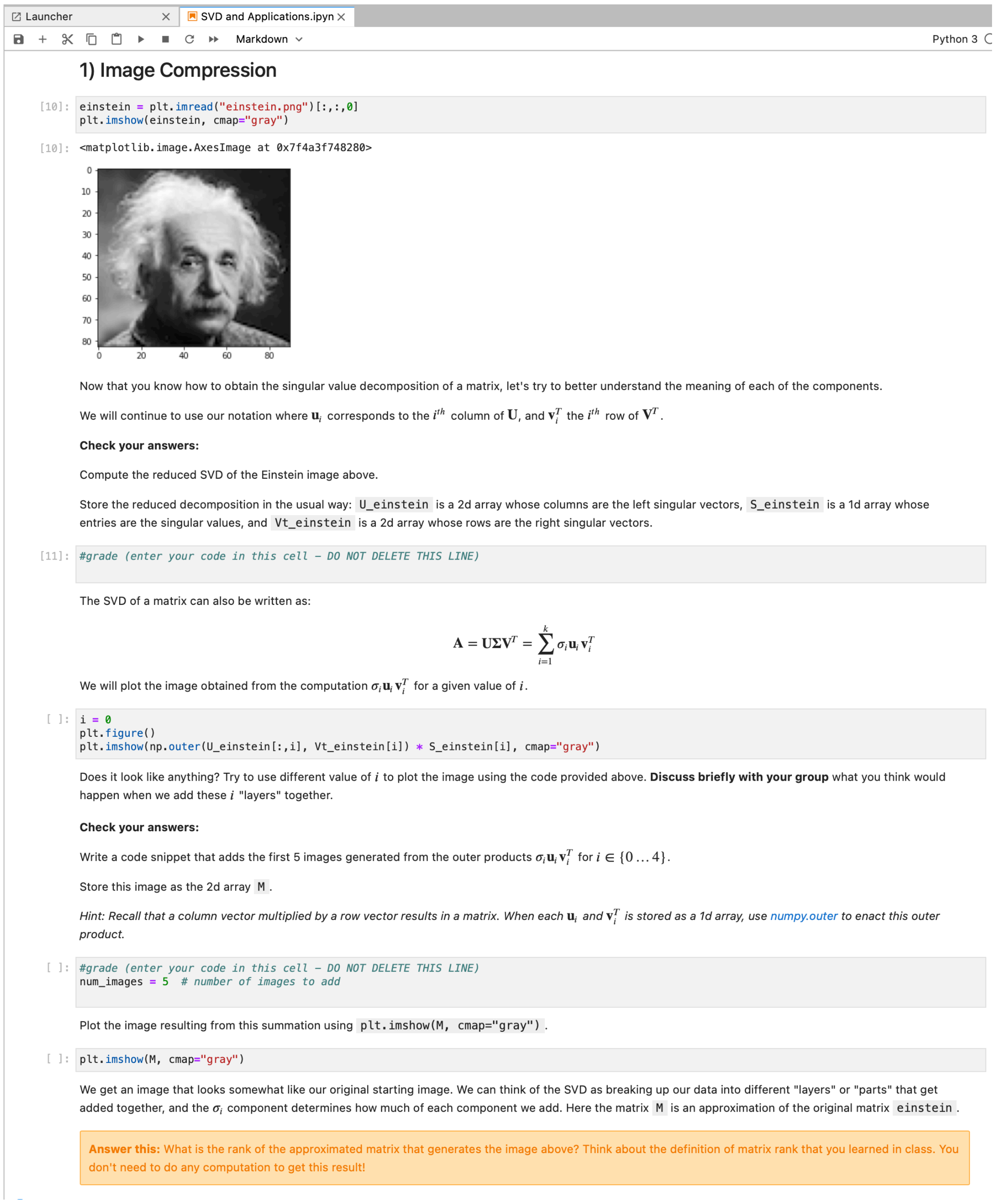
(c) Use of technological tools. When solving a task, group members should build on their individual knowledge and develop coherent contributions that can be aided by the use of synchronized tools. The students should continuously discuss and revise their solution approach, an interactive approach that is facilitated by instant feedback on attempted solutions. To accomplish this, we have developed a collaborative environment within the PrairieLearn platform that integrates synchronized Jupyter notebooks with auto-grading features that provide such immediate feedback.

Jupyter notebooks for interactive work

Computational exercises were developed for collaborative Jupyter notebooks, allowing multiple students in a group to write Python code in parallel. These notebooks contain information in various forms, including code snippets, equations, figures, videos, and links. Exercises were developed with a focus on student collaboration and real-world applications. For example, singular value decomposition (SVD) and principal component analysis (PCA) were presented as abstract mathematical techniques in the Traditional course. In the Computational course, however, SVD was used to perform image compression and PCA was utilized for facial recognition. Students were thus able to see visual results of these abstract concepts in scenarios which they encounter in everyday life. Figure 4 illustrates part of the notebook used in week 13 of the Computational course, where students apply SVD concepts to image compression. Note that the notebook provides template code for plotting, as well as text explanation and instructions for code that students need to create. It also provides prompts to encourage the students to discuss solutions and interesting features of the problem. The notebooks are synchronized in real time to facilitate the collaborative aspect of the activity.

Classroom setting

Traditional classrooms, typically with slanted-arm chairs arranged auditorium style, are not suitable for collaborative learning activities: students cannot face each other for fruitful conversations, and course staff cannot easily reach every group of students [17]. Since large



1) Image Compression

```
[10]: einstein = plt.imread("einstein.png")[:,:,0]
      plt.imshow(einstein, cmap="gray")
```

```
[10]: <matplotlib.image.AxesImage at 0x7f4a3f748280>
```

Now that you know how to obtain the singular value decomposition of a matrix, let's try to better understand the meaning of each of the components.

We will continue to use our notation where $\mathbf{u}_i$ corresponds to the i^{th} column of $\mathbf{U}$, and $\mathbf{v}_i^T$ the i^{th} row of $\mathbf{V}^T$.

Check your answers:

Compute the reduced SVD of the Einstein image above.

Store the reduced decomposition in the usual way: `U_einstein` is a 2d array whose columns are the left singular vectors, `S_einstein` is a 1d array whose entries are the singular values, and `Vt_einstein` is a 2d array whose rows are the right singular vectors.

```
[11]: #grade (enter your code in this cell - DO NOT DELETE THIS LINE)
```

The SVD of a matrix can also be written as:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

We will plot the image obtained from the computation $\sigma_i \mathbf{u}_i \mathbf{v}_i^T$ for a given value of i.

```
[ ]: i = 0
     plt.figure()
     plt.imshow(np.outer(U_einstein[:,i], Vt_einstein[i]) * S_einstein[i], cmap="gray")
```

Does it look like anything? Try to use different value of i to plot the image using the code provided above. **Discuss briefly with your group** what you think would happen when we add these i "layers" together.

Check your answers:

Write a code snippet that adds the first 5 images generated from the outer products $\sigma_i \mathbf{u}_i \mathbf{v}_i^T$ for $i \in \{0 \ldots 4\}$.

Store this image as the 2d array `M`.

Hint: Recall that a column vector multiplied by a row vector results in a matrix. When each $\mathbf{u}_i$ and $\mathbf{v}_i^T$ is stored as a 1d array, use numpy.outer to enact this outer product.

```
[ ]: #grade (enter your code in this cell - DO NOT DELETE THIS LINE)
     num_images = 5  # number of images to add
```

Plot the image resulting from this summation using `plt.imshow(M, cmap="gray")`.

```
[ ]: plt.imshow(M, cmap="gray")
```

We get an image that looks somewhat like our original starting image. We can think of the SVD as breaking up our data into different "layers" or "parts" that get added together, and the σ_i component determines how much of each component we add. Here the matrix `M` is an approximation of the original matrix `einstein`.

Answer this: What is the rank of the approximated matrix that generates the image above? Think about the definition of matrix rank that you learned in class. You don't need to do any computation to get this result!

Figure 4: Example of an interactive Jupyter notebook, providing insight into image compression through singular value decoposition.

classrooms with flexible furniture are not always available, the computational lab was split into smaller sections offered on the same day, but at different times, depending on the enrollment numbers. Due to the pandemic-induced change to online learning in Fall 2020, we conducted the computational labs via Zoom using breakout rooms for individual student groups (for more detailed information refer to the timeline in Section).

Introduction of undergraduate course staff

Embedding collaborative problem solving activities within the computational lab classes required a large number of teaching staff to be involved in the courses as group work facilitators. Based on our experience teaching engineering courses that include collaborative learning components, we adopted a ratio of 1 course staff per 20 students for the computational lab sections. Historically, the Mathematics department assigned TAs for the Traditional linear algebra course following a ratio of 1 TA per 4 discussion section, which roughly corresponds to 1 TA per 100 students. In order to increase the number of staff without a substantial increase in the budget, we introduced the hiring of undergraduate course staff to assist with the flipped-classroom lectures and office hours, following a similar approach that is widely used by courses in the Grainger College of Engineering. Hence, each section of the computational lab is staffed by two graduate teaching assistants (TA) and multiple undergraduate course assistants (CA), depending on the number of students assigned for that section. The CAs are selected from the pool of students that took the class in prior semesters and showed outstanding potential, not only based on grades, but also classroom participation and peer interactions. This model has been shown to be extremely successful, benefiting all involved–the instructors, TAs, CAs, and the students.

Results

Project milestones

Table 3 summarizes the timeline and milestones achieved in this project. This section provides the most relevant aspects of course development each semester, and how we adapted the course based on feedback and the shift to online learning due to COVID-19. The Computational course needed to first be introduced as an alternative to the Traditional course to allow for a smooth transition as the course position in the curriculum changed from upper-level (Traditional) to early instruction (Computational). As a general strategy, we aimed to introduce the Computational class on a small scale first and then grow it, opening larger enrollment numbers to it semester to semester.

This project started in Fall 2019, with the re-structuring of the linear algebra curriculum as depicted in Tables 1-2 and the development of the first five Jupyter notebooks. The Spring 2020 semester saw the creation of a pilot course offered to honors students only. This pilot course was taken in parallel with the Traditional course for an additional 1 credit hour, meeting for one hour per week and offering exclusively the additional computational components. Enrollment was 24 students, split into groups of 4-5 for collaborative learning. The class was assigned to a flexible classroom with round tables that could accommodate up to 8 students, and each table had a dedicated computer monitor. For the collaborative notebooks, which were initially hosted on CoCalc [18], students were encouraged to only have one or two members actually writing code, one projecting the content to the table monitor, while the entire group discussed how to solve the problem. This helped prevent groups from simply "splitting" the notebook amongst themselves, defeating the point of a collaborative environment. Of the 14 sessions of this pilot course, 9 were reserved for the collaborative notebooks while the rest were dedicated to a project, where students would propose their own application of linear algebra and present it to their peers. Examples of final projects included image classifiers and stock market predictors. Due to the restrictions imposed by the COVID-19 pandemic, the course moved to a fully online format midway through

Semester	Milestones	Trad #s	Comp #s
Fall 2019	Curriculum re-design and development of IPython notebooks for 5 computational lab assignments	812	–
Spring 2020	Pilot Honors section; introduction of group work in discussion sections of the Traditional course	791	24
Fall 2020	First offering of the Computational course as a special section of the Traditional course; development of homework and exam questions associated with lab content	513	253
Spring 2021	Second offering of the Computational course as a special section of the Traditional course; added auto-grading feature to IPython notebooks; continue development of homework and exam questions; added new instructor	456	420
Fall 2021	First offering of the Computational course under the new rubric; development of training material for sustainability	359	568
Spring 2022	New course staff using training material	282	1203

Table 3: Project timeline and milestones of implementation, and enrollment numbers for the Traditional and Computational courses.

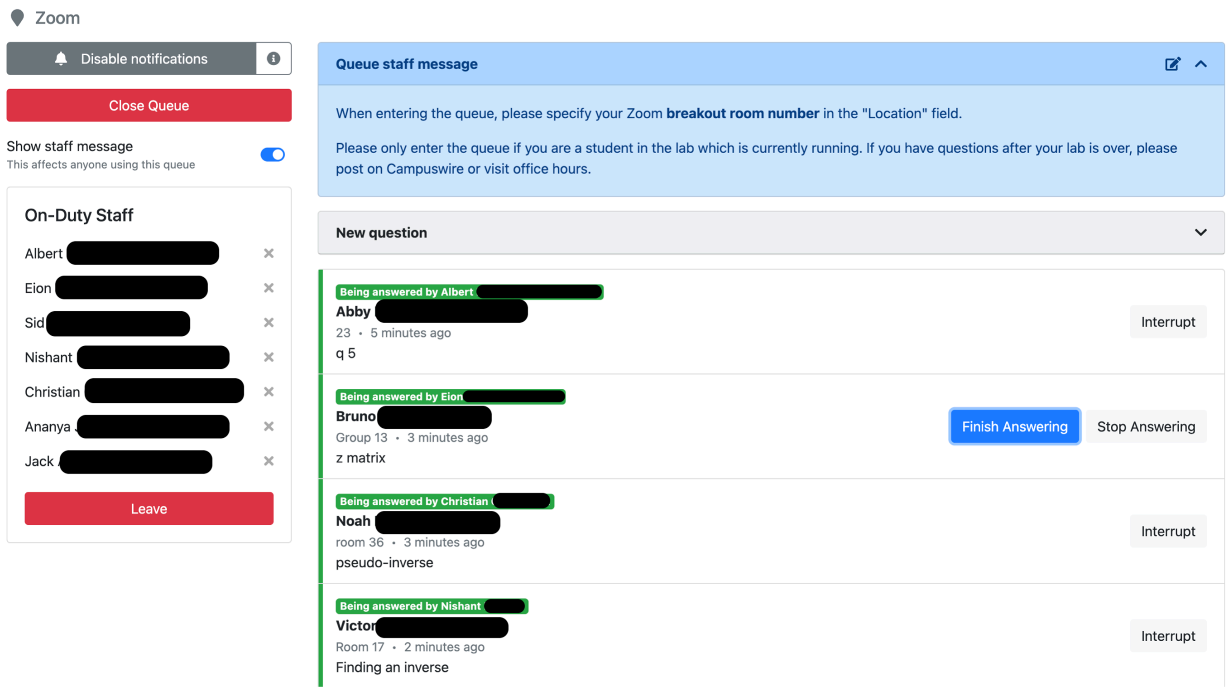
the semester. Students attended class through Zoom and worked with their teams using the breakout room feature. In parallel with the introduction of this pilot course, the discussion sections of the Traditional class were re-structured to follow a group format, as explained in Section .

At the end of Spring 2020, we surveyed the students in the Traditional linear algebra course, to collect information about how they felt regarding the use of computational tools in the class. Out of the 335 students that submitted the survey, 27% reported they were not familiar with the Python programming language. The results indicated that over half the students would like a computational lab component associated with the class, including stronger emphasis on real-world applications. Indeed, 70% of the students indicated that a closer integration of computational tools with the course content would make the class more useful to students, with another 20% reporting neutral with respect to that statement. These results were very encouraging and supported the course changes we implemented in the following semester.

In the Fall 2020 semester, we created a special section of the Traditional linear algebra course introducing the full content of the Computational course, after using the summer of 2020 to develop additional exercises to hold computational labs for the entire semester. This special section already followed the newly developed curriculum described in Tables 1-2, although it still used the old course rubric. This approach was necessary since the process of creating a new course can take several semesters, but creating a new section of an existing course did not require any special approval. While the newly approved Computational course would have an Introduction to Programming class as a prerequisite, such a requirement was not in place for the

Traditional class. Therefore, we also made sure to develop a Python tutorial as the computational lab topic of week 1 (see Table 1).

In Fall 2020, the registration increased tenfold to 250 students, and several students from the Fall 2019 pilot course were recruited to act as course assistants. The existing notebooks were improved using feedback from the previous semester and two additional notebooks were created in place of the final project. The class also remained fully online, delivered synchronously via Zoom. Student groups were assigned to breakout rooms, where they were encouraged to share their screen and work collaboratively with their peers. In addition to Zoom, an online queue system was used to moderate in-class questions (Fig. 5). Students were able to post a question indicating the breakout room number they were meeting in, allowing the course staff to move between these virtual rooms very efficiently.

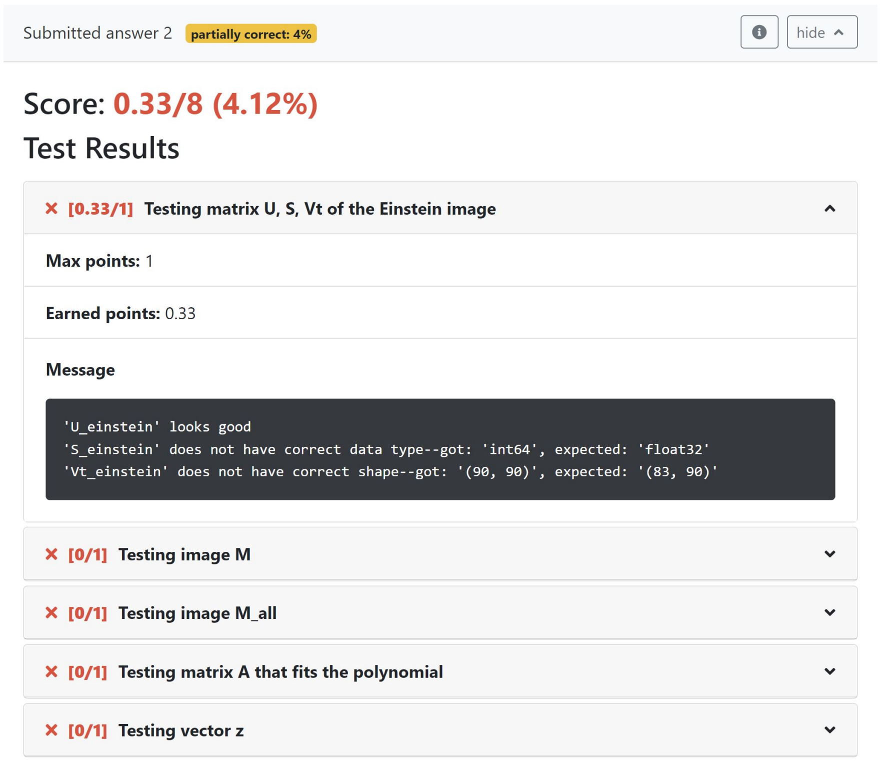


When entering the queue, please specify your Zoom **breakout room number** in the "Location" field.

Please only enter the queue if you are a student in the lab which is currently running. If you have questions after your lab is over, please post on Campuswire or visit office hours.

Figure 5: Screenshot of the queue online application that allows students to post questions and specify their location (in a physical or virtual room) such that course staff can provide help more efficiently during class activities and office hours.

In Spring 2021, the new course became fully integrated with PrairieLearn, hosting all assessments such as computational labs, homeworks, and exams. This transition was possible after the implementation of new features to PrairieLearn that allowed for shared group assignments and the use of Jupyter notebooks. In addition, we added auto-grading to all Jupyter notebooks, providing groups with immediate feedback during the computational labs. Figure 4 illustrates an example of a notebook cell marked with *#grade (enter your code in this cell - DO NOT DELETE THIS LINE)*, indicating to students that the auto-grader will evaluate the contents of that cell, and check against the correct answer. Students instantly receive feedback from the auto-grader, indicating if


Submitted answer 2 partially correct: 4%

Score: 0.33/8 (4.12%)

Test Results

✕ [0.33/1] Testing matrix U, S, Vt of the Einstein image

Max points: 1

Earned points: 0.33

Message

```
'U_einstein' looks good
'S_einstein' does not have correct data type--got: 'int64', expected: 'float32'
'Vt_einstein' does not have correct shape--got: '(90, 90)', expected: '(83, 90)'
```

✕ [0/1] Testing image M

✕ [0/1] Testing image M_all

✕ [0/1] Testing matrix A that fits the polynomial

✕ [0/1] Testing vector z

Figure 6: Screenshot of auto-grader evaluation and feedback regarding a submitted solution.

their answer is correct or not. Figure 6 shows an example of the feedback after a student submission. The switch to PrairieLearn also allowed groups to “officially” assign roles to each member within the system. In the previous semester, the roles of Recorder, Manager and Reflector were self-assigned within a group, but these assignments were not tracked by course staff, and hence many groups did not adhere to the roles.

In Fall 2021, we had the first offering of the Computational linear algebra course under a separate approved rubric at large scale with over 560 students enrolled across two lecture sections and 3 computational labs. The experienced teaching assistants developed a documentation of class operations for ease of training of new CAs and TAs, and to ensure the sustainability of the course in future offerings. This documentation was successfully used for the first time by new teaching staff taking over in Spring 2022.

Impact on engineering and science programs

Over the last three semesters, all departments in the Grainger College of Engineering, with the exception of one small program, have switched from requiring the Traditional course to the new Computational one. Moreover, some programs that did not require the Traditional linear algebra as part of the curriculum are now in the process of adding the Computational linear algebra course to their curriculum (for example, Physics, Electrical and Computer Engineering). The success of this project relies on the effective use of Communities of Practice, and the continuous involvement of this team with each departmental administration, to make sure the implemented changes would positively impact the students from different programs.

The Computational course was approved as a new class together with the prerequisite of an Introduction to Programming course. This provision anchors the new class in an important slot within the overall engineering and science education of students in many departments. In particular, it now serves as a first introduction to real-world applications of computational tools, reinforcing the usefulness of elementary programming skills.

Enrollment data

The success of the Computational course is illustrated by its rapid growth in student enrollment over the last three years. Table 3 as well as Figure 1 show the enrollment numbers for both Traditional and Computational courses, as well as the total number of students taking either class. Note that the increase of the enrollment in the Computational course is not solely a result of students moving from one course to the other, but also the absorption of new students from programs that are now either requiring or suggesting the new course to students (there is no overall increase in undergraduate admissions that would explain the increase of the total). From Fall 2021 to Spring 2022, course enrollment jumped by over 100%, supported by an increase of course staff, especially by a matched increase in the number of course assistants, crucially important to uphold effective group learning in the computational lab sections. The newly developed class is already the default linear algebra class for the majority of departments in the Grainger College of Engineering, and no linear algebra class has attracted a larger proportion of students before.

Survey data

(a) Feedback from students in the Computational course. Survey data were collected from students at the end of each semester's course offering. For the Computational linear algebra course, the survey was adapted to include specific questions regarding the computational assignments and group interactions. Figure 7 displays a sample of the survey results. The survey questions used a Likert scale with ratings from 1 to 5, where 1 indicates that a student Strongly Disagrees with the statement, and a 5 indicates that a student Strongly Agrees with the statement.

Almost 50% of the students think that the computational activities help with the overall understanding of the course content presented in the lectures (survey answers of rating 4 or above). We believe that some students still find it difficult to correlate theory with application, possibly explaining why this number is not higher. In future studies, we will use log data from PrairieLearn to find if there are correlations between student performance in the theoretical and computational components of the course. A substantial proportion of students believe the course helped them develop Python skills, and that linear algebra is an important tool for practical applications (61% and 80%, respectively). These results are evidence that the newly developed class is perceived as helpful, useful, and practically important by the student population. They also highlight the importance of the use of computational tools in introductory engineering courses.

When combining the data from the three semesters, we find that 16% of the students think their team did not work well together in the computational labs (survey answers rating 2 or below). If

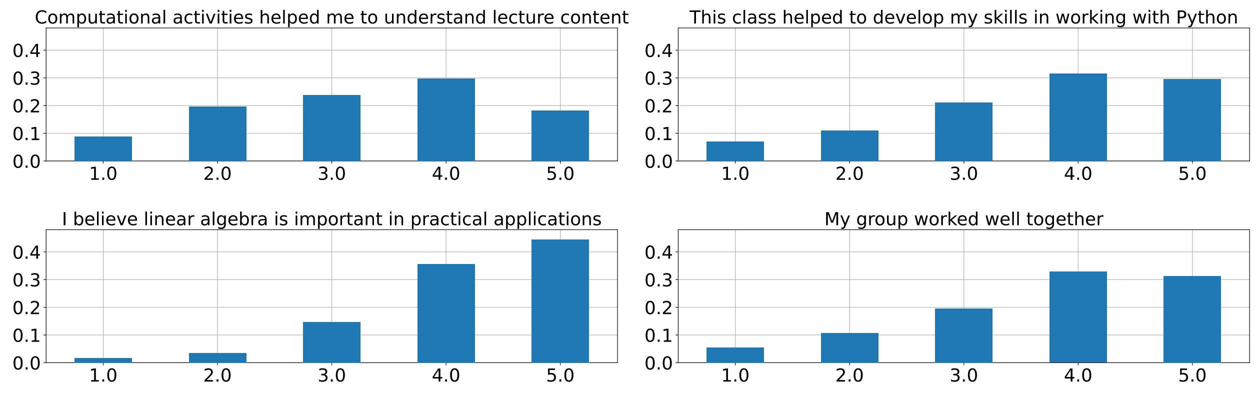


Figure 7: Combined survey results from students in the Computational courses of Fall 2020 up to and including Fall 2021 (total number of responses: 475). The survey uses a Likert scale with ratings from 1 (Strongly disagree) to 5 (Strongly agree). The individual trends for all three semesters were very similar, so that the combined results provide a more statistically sound representation of the data.

we look at the data from each semester, the percentage of students that believe their groups did not work well together was 22% in Fall 2020, 16% in Spring 2021 and 12% in Fall 2021. This decrease can be potentially explained by the adoption of PrairieLearn for the group assessments, providing more guidance to define roles and the ability to check answers with immediate feedback. Another factor contributing to the improvement could be the increased experience of teaching staff in facilitating group work.

(b) Comparison of Traditional and Computational courses regarding group work. One of the survey questions asked students to "describe their attitude towards collaborative learning". This question appears in the surveys for both Traditional and Computational courses, since both of them included group activities in the discussion sections. In Spring 2020, 20% of the students in the Traditional course indicated they did not like working in groups. Starting from Fall 2020, the same question was asked to students in the Computational course, and the percentage of students that did not like working in groups was 12% in Fall 2020, 11% in Spring 2021 and 12% in Fall 2021. We believe this increase in satisfaction from students in the Computational course can be explained by these students being exposed, through the computational labs, to the use of facilitation tools such as group roles, synchronized online tools, and auto-grading systems. This illustrates that the introduction of computational components in an organic collaborative learning scenario benefits the general comfort level of students with working in a group.

Conclusions and Outlook

The ultimate goal of the project described here has been to ensure that students taking the Computational course are more knowledgeable, more confident in their mastery of linear algebra and computational tools, and better prepared for subsequent courses in their curricula. In this paper, we present our approach to re-design a linear algebra course with a focus on incorporating computational exercises that demonstrate the solution of “real-world” applied problems. The use of modern collaborative and interactive learning techniques is an integral part of the approach. A qualitative analysis based on survey results reveals a positive impact of the course re-design on the ability of students to work together in teams, as well as on their aptitude to use Python to solve linear algebra problems. Importantly, through taking the modernized class, students have overwhelmingly understood the usefulness of linear algebra in practical applications.

As this project draws to its completion with outstanding success in enrollment and adoption of the new class by many departments, our team has also worked to ensure a sustainable trajectory of the remodeled course. The Department of Mathematics has enthusiastically adopted the new course structure and teaching techniques, and will continue to offer the class with excellence and appropriate staffing. We have developed training material to guide and support new instructors, TAs and CAs that join the course staff.

A future goal is to assess the impact of the modernized class on student performance quantitatively. To this end, we have access to log data from PrairieLearn indicating the degree to which students are achieving specific learning goals for a linear algebra course. An analysis of this data set comparing performance in the Computational course with the Traditional course will be documented in a future publication to provide more fine-grained insight into specific benefits of the new course organization. Furthermore, a future longitudinal study will analyze the impact of incorporating computational tools in linear algebra on student performance in courses downstream from linear algebra. Overall, such data analysis is a rare opportunity to draw meaningful comparisons between student learning in classes that differ only by well-defined teaching components, and provide substantial statistics at the same time.

Due to the success of this project, other programs within the Grainger College of Engineering and beyond became interested in implementing similar activities and pedagogical changes in some of their classes. In particular, the adoption of PrairieLearn for collaborative learning activities and the use of computational tools can be flexibly applied to various courses, although care must be taken to consider the specific needs of each individual class. A subset of the professors initially involved in this project have received another SIIP grant to incorporate computational tools in the Differential Equations course, an on-going project currently in its first year. Such initiatives are aligned with general educational directions in the College of Engineering, emphasizing computational literacy in undergraduate students as well as the students’ ability to work productively in groups. The approach presented here provides a blueprint for larger-scale changes that benefit math and engineering education as a whole.

Acknowledgments

This work was supported by the Department of Mechanical Science and Engineering and the Department of Computer Science as well as the Grainger College of Engineering at the University of Illinois at Urbana-Champaign as part of the SIIP program.

References

[1] C. Tang, “Computer-aided linear algebra course on jupyter-python notebook for engineering undergraduates,” *Journal of Physics: Conference Series*.

[2] G. Hutchison, “Integrating python into an undergraduate mathematics for chemists course,” in *Teaching Programming across the Chemistry Curriculum*. ACS Symposium Series, 2021, ch. 9, pp. 123–134.

[3] G. Herman, L. Hahn, and W. West, “Coordinating college-wide instructional change through faculty communities,” in *Proceedings of the ASME 2015 International Mechanical Engineering Congress and Exposition*, 2015.

[4] M. West, M. Silva, and G. Herman, “Sustainable reform of an introductory mechanics course sequence driven by a community of practice,” in *Proceedings of the ASME 2015 International Mechanical Engineering Congress and Exposition*, 2015.

[5] W. Fagen, C. Heeren, G. Herman, and W. West, “Re-engineering an "introduction to computing" course within a college-wide community of practice,” in *Proceedings of the 122nd American Society for Engineering Education Annual Conference and Exposition*, 2015.

[6] G. Herman, I. Mena, W. West, J. Mestre, and J. Tomkin, “Creating institution-level change in instructional practices through faculty communities of practice,” in *Proceedings of the 122nd American Society for Engineering Education Annual Conference and Exposition*, 2015.

[7] M. West, C. Zilles, and G. Herman, “Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning,” in *Proceedings of the American Society for Engineering Education (ASEE) 2015 Annual Conference*, 2015.

[8] S. Freeman, S. Eddy, M. McDonough, M. Smith, N. Okoroafor, H. Jordt, and M. Wenderoth, “Active learning increases student performance in science, engineering, and mathematics,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 23, pp. 8410–8415, 2014.

[9] J. Gasiewski, M. Eagan, G. Garcia, S. Hurtado, and M. Chang, “From gatekeeping to engagement: A multicontextual, mixed method study of student academic engagement in introductory stem courses,” *Research in Higher Education*, vol. 53, p. 229–261, 2012.

[10] M. Kapur and C. K. Kinzer, “Examining the effect of problem type in a synchronous computer-supported collaborative learning (cscl) environment,” *Educational Technology Research and Development*, vol. 55, pp. 439–459, 2007.

[11] T. Tucker, S. Shehab, E. Mercier, and M. Silva, “Board 50: Wip: Evidence-based analysis of the design of collaborative problemsolving engineering tasks,” *Proceedings of American Society for Engineering Education*, 2019.

[12] H. H. Hu, C. Kussmaul, B. Knaeble, C. Mayfield, and A. Yadav, “Results from a survey of faculty adoption of process oriented guided inquiry learning (pogil) in computer science,” in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 2016, pp. 186–191.

[13] T. Nokes-Malach, J. Richey, and S. Gadgil, “When is it better to learn together? insights from research on collaborative learning,” *Educational Psychology Review*, vol. 27, p. 645–656, 2015.

[14] S. Shehab, L. Lawrence, E. Mercier, A. Margotta, E.Livingston, M. Silva, and T. Tucker, “Towards the effective implementation of collaborative problem solving in undergraduate engineering classrooms: co-designing guidelines for teaching assistants,” *Proceedings of American Society for Engineering Education*, 2020.

[15] “Pogil: Process oriented guided inquiry learning,” https://pogil.org, Last accessed on 2022-02-04.

[16] “Jupyter notebooks,” https://jupyter.org, Last accessed on 2022-02-04.

[17] R. Essick, W. West, M. Silva, G. Herman, and E. Mercier, “Scaling-up collaborative learning for large introductory courses using active learning spaces, ta training, and computerized team management,” in *Proceedings of the 123rd American Society for Engineering Education Annual Conference and Exposition*, 2016.

[18] “Cocalc,” http://www.cocalc.com, Last accessed on 2022-02-05.