BOARD # 78: Student Use of ChatGPT and Claude in Introductory Engineering Education: Insights into Metacognition and Problem-Solving Patterns

Dr. Anthony Cortez, Point Loma Nazarene University

Anthony Cortez is currently an Assistant Professor in the department of Physics and Engineering at Point Loma Nazarene University. He received his BS in Physics from University of California San Diego (UCSD). He went on to complete his MS and Ph.D. in Mechanical Engineering from University of California Riverside (UCR). His research interests include technology as a tool in the classroom, high temperature superconductivity, superconducting detectors, nanofabrication, and space THz technology.

Dr. Paul Schmelzenbach, Point Loma Nazarene University

Dr. Paul Schmelzenbach is currently a Professor of Physics and Engineering at Point Loma Nazarene University. He received his BS in Physics and a BS in Chemistry from Northwest Nazarene University in 1998. He then went on to complete his MS and Ph.D. in Nuclear Physics from Oregon State University in 2003. His research interests include gamma-ray spectroscopy, analysis of large data sets, computational physics, and physics and engineering education.

Student Use of ChatGPT and Claude in Introductory Engineering Education: Insights into Metacognition and Problem-Solving Patterns

1 Introduction

The widespread and rapid emergence of large language models (LLMs) such as ChatGPT, Claude, and Gemini may be fundamentally transforming how students approach their academic work. This transformation is clear in higher education, where AI tools can now successfully generate solutions to many problems that undergraduates previously struggled with for hours [1]. Rather than simply supplying answers, educators have experimented with using LLMs in the classroom in various ways such as brainstorming, proofreading, providing feedback, giving encouragement, and checking for understanding [2–4]. By their widespread availability and increasing capabilities, LLMs appear to be reshaping how students access, interpret, apply, and potentially learn information.

In engineering education, where problem-solving skills and technical understanding are foundational, the role of LLMs creates complex pedagogical considerations. The capabilities of widely available models, such as ChatGPT, have evolved dramatically over the last two years. The present and future role of LLMs in programming for engineers appears quite complicated, serving dual roles as both coding assistance (or code creation) and learning tools. These two roles, which are sometimes at odds with each other, raise important questions about how students develop programming skills. The rapid advancement in quality and capability of these tools has compressed the timeline for engineering educators to understand and adapt their pedagogical strategies to help students best learn in this new environment.

While LLMs may change how students read and understand material in some contexts, the ability to read and understand technical documentation remains a key skill for engineers. Documentation serves not only as a reference but also as a tool for understanding a system's constraints and capabilities. This study examined how students navigated the MATLAB documentation in an exam that included functions they had not previously used in class. These results were then compared with student interactions with an LLM as they revisited this same MATLAB task. Through this analysis, we identified different problem-solving approaches, observed common pitfalls in AI usage in basic programming tasks, and identified patterns that can inform our pedagogy.

2 Methods

A. Classroom Information and Experimental Overview

An introductory engineering course consisting of 21 students completed a series of assignments that involved the use of ChatGPT and Claude in an effort to gain insight into student metacognition and problem-solving patterns. Specifically, we examined metacognitive aspects including planning behaviors (time spent reviewing problems before coding), self-monitoring (recognition and correction of errors), and resource utilization strategies (documentation usage patterns and LLM interaction methods).

This study used a two-part experimental design to examine student problem-solving behaviors. In the first part, students completed a traditional MATLAB exam where scripting required the use of documentation to use previously unfamiliar functions. Screen recordings were captured through Honorlock for each student while working on the exam. Screen recordings were chosen as they allow direct observation of students' problem-solving processes, including their planning, resource utilization, and self-monitoring behaviors. The first 20 minutes were analyzed as this period captured the critical initial problem-solving approaches while maintaining a consistent comparison across students. In the second part, conducted during a subsequent lab session, without the pressure of an exam, students used LLMs to review and correct their exam solutions. Screen recordings and transcripts of the LLM interactions were analyzed for their prompting patterns and timing between actions. Because these interactions were in a reduced stress environment, the AI lab session was utilized to create problem-solving strategy groups. These groups were then used retrospectively to analyze their exam behaviors, allowing us to identify potential relationships between traditional and AI-assisted problem-solving approaches.

Information on course structure and learning objectives is described in a previous paper by the authors [5]. The learning outcome this study focuses on is the students' ability to "become proficient at using MATLAB, including writing .m files and correcting or modifying existing code." Before the AI section of the course, students were introduced to MATLAB in a traditional lecture format. The lesson plans emphasized the plotting capabilities of MATLAB and the efficient utilization of the built-in MATLAB documentation. Assessment of the course learning outcome consisted of a practical MATLAB exam, where students were required to use MATLAB to achieve a series of tasks that led to a final plot.

B. MATLAB Exam Question

Students were given 1 hour and 30 minutes to complete an in-person MATLAB exam. The only resource available to students was the MATLAB built-in documentation. Screen recordings were captured through Honorlock for each student while working on the exam. The following text is the exam question.

Create a plot of overlaying sine waves with randomized amplitude and frequency following these specifications:

(a) Use randi() to generate a random integer between 1 and 10, and store it in the variable A (amplitude).

(b) Use rand() to generate a random decimal value between 0.1 and 1, and store it in the

variable f (frequency).

(c) *Create a range of 200 equally spaced values stored in the variable t (time) between 0 and 10 seconds.*

(d) *Create the array **y** of the sine wave using the formula*

$$y = A\sin(2\pi ft)$$

(e) *Make a plot of **t** and **y**, where **t** is the horizontal axis and **y** is the vertical axis. Use a solid line if the frequency is less than 0.5 Hz, and dashed line if it is greater than or equal to 0.5 Hz.*

(f) *Repeat steps a-e, overlaying each plot on top of the others, as long as the random amplitude is less than 9. (Once it is 9 or larger, stop)*

(g) *Create a title that displays the number of sine waves that were generated in your plot. (Note: This number can change with each running of your script file. Therefore, create a variable to keep track of this count.)*

(h) *Label the x and y axes with "Time (s)" and "Amplitude (m)", respectively.*

(i) *Add a legend that shows the frequency of each plotted sine wave. Each label in your legend should include the appropriate variable, number, and units. For example: f = 0.56192 Hz.*

The exam was designed to test different levels of critical thinking and coding skills that required students to use documentation. For example, a solution to part A is

```
A = randi(10);
```

This required students to use the command randi(), which they had not previously seen in class. A random integer between 1 and 10 is returned using randi(10), which is clear from the documentation. In contrast, part B required students to both shift and scale rand(), thus requiring a more sophisticated approach. A solution takes the form

```
a = 0.1;
b = 1;
f=a+(b-a)*rand(1);
```

This required students to use the command rand(), which they had also not previously seen in class. In addition to understanding the output of this function, students needed to consider how to effectively shift and scale the output to match the bound requirements of the exam. Ideally, students should be comfortable referencing the documentation to gain an understanding of the rand() function and troubleshoot the problem such that it meets the exam requirements. However, if a student has difficulty arriving at a proper solution, then careful reading of the help documentation will provide the answer for them in an example. At the time of this paper, one can find a formula that provides the solution to part B of the exam by visiting the examples in the rand() documentation.

C. ChatGPT and Claude Lab

Following the MATLAB exam, the course covered LLMs such as ChatGPT and Claude. These lesson plans included background information on LLMs, the importance of effective prompting [6–8], and examples of misinformation [5]. After the lectures on LLMs, the MATLAB exams were returned to students with feedback. A three-hour lab period was dedicated to revisiting the MATLAB exam question. The first part of the lab required students to review their MATLAB exam feedback, and work towards improving their code to satisfy all of the exam requirements. During this portion of the lab, students were allowed to use any resource other than LLMs. The second part of the lab required the students to work through the exam problem with the use of ChatGPT or Claude. The specific instructions given to students were "spend 30 minutes using ChatGPT or Claude to help you solve the exam problem". The third part of the lab provided time for the students to answer questions regarding their experience with the AI used.

D. Screen Recordings of Student Work

Each student in the class had their screen recorded on two occasions, the MATLAB exam and the ChatGPT/Claude lab. We focused our analysis on the first twenty minutes of the exam recordings and the full thirty-minute videos of the AI recordings. An initial screening of the exam recordings revealed that the twenty minutes was sufficient in analyzing student work on parts A-C of the question. During this initial 20-minute period, we tracked how students utilized their time. Specifically, we noted the times when students had the documentation window as their focus, when students had the question window as their focus (which was given electronically), or when they were working in MATLAB. Additionally, we identified common initial MATLAB coding techniques as follows:

- immediate from memory
- immediate from documentation
- delayed with searches in documentation
- multiple attempts before correct
- not solved

for each of these three parts.

The thirty-minute AI interaction recordings were analyzed to identify patterns in student prompting. Based on these observations, students were classified into one of the following five engagement groups:

- *Systematic Problem Solvers*: Students who demonstrated organized approaches, clarified requirements before prompting (showing planning), and asked targeted questions.
- *Iterative Approach Solvers*: Students who broke the problem into smaller chunks and asked clarifying questions to build understanding.
- *Verification Focused Solvers*: Students who primarily used the prompts to fix very specific issues.

- *Distracted Solvers*: Students who often seemed to show curiosity about the capabilities of AI (often unrelated to the task), used very casual language in their prompts, and seemed to jump from topic to topic with partially formed questions.
- *Struggling Partial Solvers*: Students who consistently misunderstood parts of the problems, lacked technical vocabulary, and inadvertently misdirected AI responses due to their misunderstandings.

The student interaction transcripts with the LLMs were used to identify patterns in their prompting strategies and problem-solving approach. Students were categorized based on key indicators such as prompting organization, use of technical vocabulary in prompts, frequency of clarifying questions, and overall approach to the problem. For example, systematic solvers typically prefaced their prompts with context and requirements, while iterative solvers broke down complex parts into sequential questions to build understanding.

The creation of these engagement groups provides insight into how students interact with the AI in a non-exam setting. Therefore, this provided a means to characterize each student's problem-solving approach in a reduced stress environment. We compare these engagement groups to the exam recordings to identify any meaningful patterns.

Several of these categories represent different but important valid problem-solving strategies. While systematic planners demonstrated strong upfront organization, iterative solvers showed effective decomposition of complex problems, and verification-focused solvers displayed targeted use of AI assistance.

3 Results

A. MATLAB Exam Screen Recordings

The initial coding strategies for each student to work through parts A, B, and C of the exam are shown in Figure 1. The majority of students struggled with part B, as indicated by the number of students who solved the problem correctly compared to those who solved the problem incorrectly. Part C of the exam appears to be the simplest question for the students, since the majority of those who solved it correctly did so from memory immediately. A visualization of student time utilization in the twenty-minute window is shown in Figure 2. All but one student spent fewer than one hundred seconds reading the exam question before working on the problems. Student A spends nearly two hundred seconds reading the exam question before attempting part A. This student is one of the four who solved part B correctly.

Student time utilization during part B is shown in Figure 3 to provide visual clarity. Notice how some students take a break and come back at a later time to work on the problem. The three students who answered correctly in the 20-minute period accessed the documentation to help lead them to the correct solution, demonstrating good utilization of the resource. A fourth student did come back after the 20-minute period and answered the question correctly. Review of the final answer submitted indicates this student also used the documentation. All but one who answered the problem incorrectly attempted to use the documentation. Despite the solution being readily available in the documentation, students were unable to solve this problem.

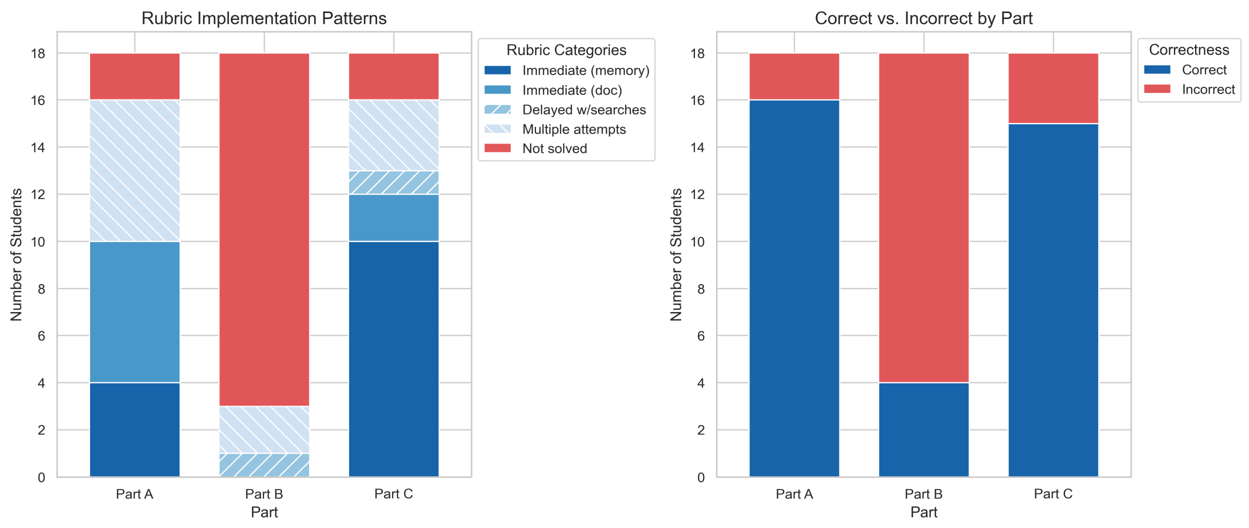

Figure 1: (Left) Categorization of student work on parts A, B and C on the traditional MATLAB programming exam. Notice students struggled significantly more with part B. Also it can be seen students typically were more able to answer part C from memory. (Right) The number of students who got each problem correct versus incorrect by the end of the exam.

B. Problem-solving Strategy Groups

The students were placed in one of the five groups: systematic, iterative, verification-focused, distracted, and struggling solvers as described in the methods section. It was interesting to see how students in these various categories approached aspects of the MATLAB exam. One of the most noticeable patterns was that systematic solvers tended to spend more time reading the question before working on the problem as shown in Figure 2.

Since the solution to part B was available in an example problem within the MATLAB documentation, the systematic solvers' careful reading behavior suggests they should have been able to find the example solution in the documentation. However, only student A of this group answered part B correctly. Reviewing the video recordings for each of these students revealed that they all found and read the example that contained the solution and formula, however none of them used it in their final answer.

Student A arrived at a correct answer using a while statement to force the minimum value required for part B. Student Q quickly skims through the examples, including the one with the formula, but their final answer did not attempt to specify bounds and simply used $f = rand()$. Student J found the example with the formula early in their documentation use and took a considerable amount of time to read through the example. However, this student continued to swap back and forth between the problem and the documentation as shown in Figure 3. During this period, the student searched the following terms "random decimal", "random", "rand", "random decimal value", "rng", and "random decimal generate". The final answer submitted by student J did not account for the bounds, but simply used $f = rand()$. Although the systematic solvers spend more time reading, there seems to be a disconnect from what they read in the documentation in relation to part B of the exam. In contrast, students N and M of the iterative

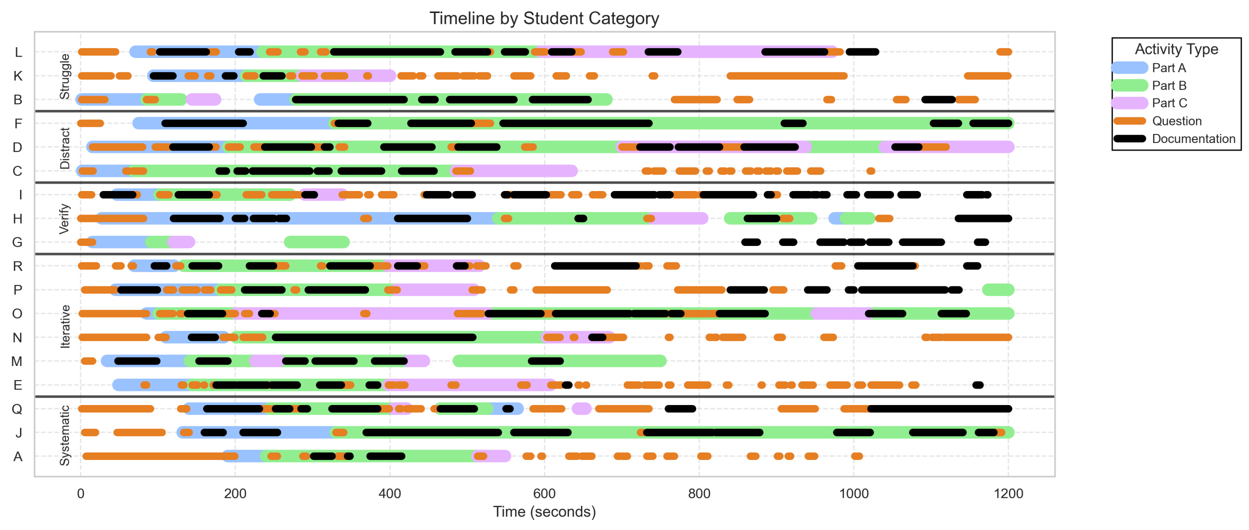

Figure 2: Timeline of students' activities during the first 20 minutes of the MATLAB exam. This plot shows how students utilized their time during the exam period. We specifically show when students were reading the exam question (orange), viewing the help documentation (black), and actively working on either parts A, B, or C (blue, green, and purple) of the exam. The students are grouped by their problem-solving strategies, as determined from the AI videos, to identify patterns in time and resource utilization.

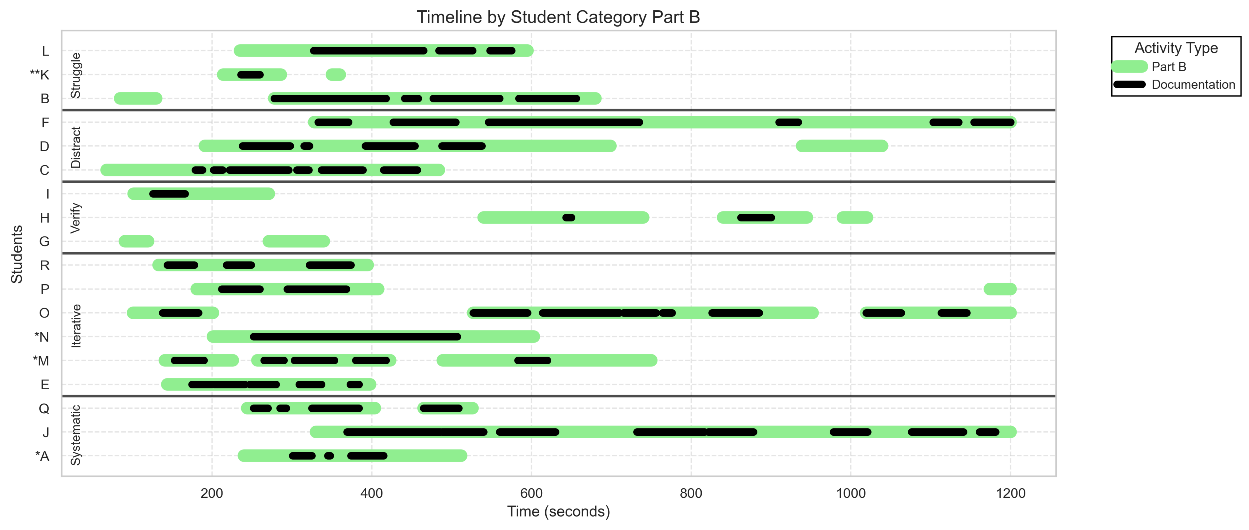

Figure 3: Timeline of students' access to documentation while working on part B. Students marked with a "*" answered part B correctly within this twenty-minute period. The student marked with a "**" answered part B correctly after the twenty-minute period.

solving group utilized the example in their final solution. Student K of the struggling group, utilized the correct formula after the 20-minute period.

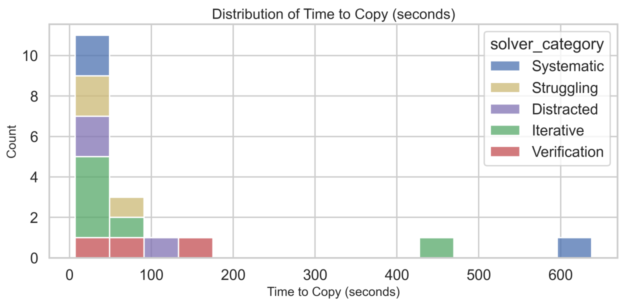

Figure 4: Time interval between AI-prompt and copying code. Notice that all problem solving categories can be found in short time intervals. Only two students displayed longer engagement times before copying the code.

C. Screen Recordings of Student Interaction with LLMs

Time analysis of the AI recordings revealed that students spend a short amount of time reading through the LLM's output. The time between a student's initial prompt to the LLM to when they copied the code is shown in Figure 4. The majority of students spent less than one hundred seconds before they copied the code into MATLAB. This is interesting considering that the code generation time of the LLM ranged from approximately 5-30 seconds. Therefore, it suggests the 11 students on the shorter time intervals were simply waiting for the LLM to finish generating the code and disregarded any reasoning or other output provided by the LLM.

Although the students did not concern themselves with how the code generated by the LLM was structured, they briefly verified the plot generated seemed to satisfy all of the exam requirements. For example, to ensure the maximum and minimum bounds were met on part B of the exam, students ran the same code multiple times to visually inspect the frequency generated each time. This method of verification ensures a generated plot meets all of the requirements, but does not verify that the source code will *always* meet the requirements because of its use of random numbers.

In most cases, the AI solved the problem correctly on the first attempt. In the case where the AI was incorrect, students prompted the AI such that it led to a correct answer. An example included ChatGPT producing a plot that did not properly display the axes labels. The student recognized that the axes were missing, but incorrectly prompted the AI by stating: "The legend does not work". This is a concern because ChatGPT did not correct the students' mistake on terminology, but rather agreed and adjusted the axes labels to match the exam requirements. Another student commented in the lab assignment "It just gave me the formula that is used to calculate a random number given a range. [...] It didn't give much beyond that." This does not agree with the transcript of the students' interaction with the AI, which provides a detailed explanation behind the formula listed. This suggests the student did not carefully read through the AI's response or possibly does not understand the explanation.

4 Discussion

A. Student Metacognition before AI and with AI

The screen recording provided a unique perspective for the instructor on the students' approach in solving the exam problem. It was evident when a student recognized their own mistakes and their method in fixing those mistakes to arrive at a correct solution. For example, in part B, students were required to generate a random number from 0.1 to 1 using the rand() function. The solution requires critical thinking beyond the ability to call the rand() function due to the fact that one must scale and offset the value such that the minimum number is non-zero. Although a majority of the students initially approached this problem by using the rand() function without any shifting or scaling, 77% of students analyzed their code by verifying its output in the command window. Students recognized this output did not meet the exam requirements, which led them to access the documentation for help in using the rand() function. Students were able to locate an example problem in the help documentation that provides a formula to answer part B of the exam, but only three students (N, M, and K) used this formula in their final code. In addition, there were some cases in which the student appeared to misread the question or quickly skimmed the problem. For example, one student used the random function although the exam required the use of rand(). This highlights the necessity for students to read through the problem carefully.

Analysis of the screen recordings of students interacting with AI revealed that students did not fully read the output of the LLM, but tended to simply copy and paste the resulting code into MATLAB. These students partially verified the code worked by running the script multiple times and visually inspected the plot against the requirements of the exam. Although these students found the correct answer, they did not spend time thinking about the problem themselves and tended to omit critical thinking.

When working with the LLMs, we observed students to prioritize rapid code generation over their understanding. The near immediate availability of AI-generated answers appeared to reduce students' metacognitive engagement. They were less likely to evaluate their understanding of the problem, and tended not to reflect on the code they were developing when working with the AI. In many cases, the LLMs explanations of code appeared to be unread and ignored.

B. Recommendations for Teaching Strategies

Throughout this study, we observed students making mistakes due to insufficient attention to detail, particularly in not spending enough time reading, both during the exam and when working with AI. Large Language Models such as ChatGPT and Claude can generate extensive text rapidly, which often leads students to skim or skip reading the content entirely. These observations suggest a need for specific prompt engineering strategies to improve student learning and engagement.

The first strategy is managing the length of output when using an LLM to better learn and understand new material. We recommend establishing output constraints before asking questions. For example, an initial prompt might state: "I am going to ask you some questions regarding MATLAB code. I am a novice coder and would like to learn more. Please only respond in short bullet points (no more than 5 at a given time)." This approach ensures responses are concise and promotes thorough reading of the material. The number of bullet points can be changed and the phrasing can be adapted depending on the student's preference.

A second recommendation is to reduce the number of questions asked in a single prompt. Rather than copying and pasting entire problems all at once, questions should be divided into manageable pieces. This approach allows students to focus on individual parts while building understanding.

After breaking problems into smaller parts, we recommend implementing a check of understanding for the student. A sample prompt could include: "Can you give me a sample problem to verify that I understand how to properly use this new function? Do not provide the solution." This strategy encourages active engagement with the material and helps identify gaps before continuing with the problem.

The combination of these prompting strategies can help students practice careful reading and build a deeper understanding of the material. As LLMs continue to evolve in capabilities, specific strategies for working with them will need to adapt. As we have been learning over the past few years, what works effectively one semester may need adjustment the next. As engineering educators, we should remember that though specific techniques change, students' active engagement with the material remains key to learning.

References

[1] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, "Solving olympiad geometry without human demonstrations," *Nature*, vol. 625, no. 7995, pp. 476–482, 2024.

[2] M.-L. Tsai, C. W. Ong, and C.-L. Chen, "Exploring the use of large language models (llms) in chemical engineering education: Building core course problem models with chat-gpt," *Education for Chemical Engineers*, vol. 44, pp. 71–95, 2023.

[3] C. Liu and S. Yang, "Application of large language models in engineering education: A case study of system modeling and simulation courses," *International Journal of Mechanical Engineering Education*, p. 03064190241272728, 2024.

[4] L. M. Sánchez-Ruiz, S. Moll-López, A. Nuñez-Pérez, J. A. Moraño-Fernández, and E. Vega-Fleitas, "Chatgpt challenges blended learning methodologies in engineering education: a case study in mathematics," *Applied Sciences*, vol. 13, no. 10, p. 6039, 2023.

[5] A. Cortez and P. D. Schmelzenbach, "Integrating chatgpt in an introductory engineering undergraduate course as a tool for feedback," in *2024 ASEE Annual Conference & Exposition*, 2024.

[6] G. Marvin, N. Hellen, D. Jjingo, and J. Nakatumba-Nabende, "Prompt engineering in large language models," in *International conference on data intelligence and cognitive informatics*, pp. 387–402, Springer, 2023.

[7] B. Meskó, "Prompt engineering as an important emerging skill for medical professionals: tutorial," *Journal of medical Internet research*, vol. 25, p. e50638, 2023.

[8] L. Wang, X. Chen, X. Deng, H. Wen, M. You, W. Liu, Q. Li, and J. Li, "Prompt engineering in consistency and reliability with the evidence-based guideline for llms," *npj Digital Medicine*, vol. 7, no. 1, p. 41, 2024.