

Article

AI as a Procedural Equalizer: Performance Comparison in Programming-Based Engineering Coursework Following the Emergence of Generative AI

Ghazal Barari ^{1,*} , Jorge Ortega-Moody ¹, Kouroush Jenab ² , Tyler Ward ³  and Karl Siebold ¹ ¹ School of Engineering, Embry-Riddle Aeronautical University, Daytona, FL 32114, USA; ortegamj@erau.edu (J.O.-M.); siebo6ed@erau.edu (K.S.)² Department of Engineering Sciences, Morehead State University, Morehead, KY 40351, USA; k.jenab@moreheadstate.edu³ Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA; tbwa233@uky.edu

* Correspondence: bararig@erau.edu

Abstract

Generative artificial intelligence (AI), especially large language models (LLMs) that can write and debug code, is changing how students approach programming work in engineering education. Unlike more open-ended conceptual or modeling tasks, programming fits closely with what these systems do well: generating syntax, fixing errors, building procedural logic, and completing code structures. Hence, programming coursework may be one of the areas in which AI changes performance patterns in a measurable way. This study examines whether that shift appears in actual student outcomes. Using a retrospective pre/post design, it compares results from a pre-AI period (2021–2022) with results from a post-AI period (2023–2025), when generative AI tools became widely available to students. The focal assessment is a comprehensive programming project graded with the same rubric across multiple sections and terms. Performance is evaluated through descriptive statistics, distributional comparisons, and mastery thresholds ($\geq 80\%$). The post-AI period shows a rise in overall scores, along with strong clustering near the top of the scale. Lower- and middle-range scores become much less common, most students fall in the highest score band, and overall variability declines. These results suggest that generative AI acts as a procedural equalizer in programming contexts, referring to the role of generative AI in reducing performance differences by assisting with rule-based, syntax-driven, and execution-oriented aspects of tasks, thereby raising baseline outcomes while compressing variation among students. It appears to raise lower-end performance and make outcomes more consistent, but it also narrows the spread among stronger students and creates a ceiling effect. That pattern raises questions about assessment validity, skill differentiation, and what “mastery” means when AI can handle much of the procedural work. Using multi-term data from authentic online courses, this study adds empirical evidence to the growing literature on AI in engineering education and identifies programming coursework as a setting where generative AI may have already changed performance dynamics in a structural way.



Academic Editor: Christos Bouras

Received: 20 March 2026

Revised: 9 April 2026

Accepted: 17 April 2026

Published: 14 May 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: generative artificial intelligence; engineering education; large language models (LLMs); AI-assisted learning; online engineering course

1. Introduction

Generative AI, especially code-capable LLMs, has quickly become part of many students' everyday workflow. In programming, MATLAB (R2026a) analysis, and other computer-based engineering tasks, these tools can produce syntactically correct code, fix errors, suggest algorithms, and format outputs in seconds. Unlike earlier educational technologies that were introduced through planned course integration, generative AI has often been adopted informally by students with little instructor direction. That shift makes it important to ask how AI availability is affecting measurable performance in programming and computational engineering coursework (Kasneci et al., 2023 [1]; Tlili et al., 2023 [2]; Lee et al., 2024 [3]; Baek et al., 2024 [4]; Wang et al., 2024 [5]).

Programming and other computer-based assignments fit the strengths of generative AI particularly well. Tasks such as writing code, fixing syntax errors, building loops and functions, generating plots, and visualizing data rely on procedural rules that these models handle well. As a result, programming and MATLAB projects may be settings where AI does more than support learning; it may change the way performance is produced and distributed. In un-proctored online courses, where students work independently, these tools are especially easy to use and potentially influential (Finnie-Ansley et al., 2022 [6]; Nguyen & Nadi, 2022 [7]; Yetistiren et al., 2022 [8]; Prather et al., 2024 [9]; Kuhail et al., 2024 [10]; Mo et al., 2025 [11]).

Understanding that influence matters for instructional decisions. Programming projects and MATLAB-based modeling assignments are often used to assess applied reasoning, algorithmic thinking, and computational competence in engineering curricula. If generative AI raises baseline performance, compresses score distributions, or makes it harder to distinguish among higher-performing students, then common interpretations of mastery may need to be revisited. Without empirical evidence, instructors risk either underestimating AI's procedural impact or responding in ways that do not match what is actually happening in student work (Cotton et al., 2024 [12]; Chen et al., 2024 [13]; Karkoulou et al., 2025 [14]; Ortiz-Bonnin & Blahopoulou, 2025 [15]).

1.1. Literature Review

Early studies show that student use of generative AI is widespread, fast-moving, and often self-directed rather than instructor-assigned. Students' decisions to rely on these tools appear to be shaped by perceived usefulness, technical confidence, institutional rules, and the risk of academic penalties, which helps explain the wide variation in AI-supported study practices (Baek et al., 2024 [4]; Strzelecki, 2024 [16]; Oc et al., 2025 [17]; Fajt & Schiller, 2025 [18]). At the same time, broader reviews of higher education research note that much of the literature still focuses on perceptions, motivations, and reported use cases, even as the empirical base continues to grow (Batista et al., 2024 [19]; Bhullar et al., 2024 [20]; Ogunleye et al., 2024 [21]; Zhang et al., 2024 [22]).

In engineering and computing education, programming has become one of the clearest sites of AI adoption. Generative AI is well suited to procedural and syntax-heavy work such as code generation, debugging, loop construction, function writing, and structured algorithmic output. Classroom studies increasingly describe changes in how students work with ChatGPT and related tools, including shifts in debugging habits, reflection, and perceived efficiency (Ghosh, 2024 [23]; Yilmaz & Karaoglan Yilmaz, 2023 [24]; Sun et al., 2024 [25]; Groothuijsen et al., 2024 [26]; Garcia, 2025 [27]; Güner & Er, 2025 [28]; Suh et al., 2025 [29]; Lee et al., 2025 [30]). Still, much of this work emphasizes usage patterns and perceptions rather than multi-term comparisons based on the same assessment.

Scoping and systematic reviews increasingly describe generative AI as a structural change in how students produce written and computational work, not simply another

add-on tool (Weng et al., 2024 [31]; Belkina et al., 2025 [32]; Manorat et al., 2025 [33]). In programming, AI systems can often generate executable code with high syntactic accuracy, which may raise baseline performance and reduce procedural mistakes. Because of that fit between tool capability and task structure, performance effects may be easier to detect in computational coursework than in assignments that depend more heavily on open-ended conceptual reasoning or physical modeling (Park & Kim, 2025 [34]; Lepp & Kaimre, 2025 [35]).

Research outside programming points in a similar direction. Reviews show that AI can help with ideation, revision, and explanation, which may improve the final quality of submitted work. At the same time, scholars warn that better products do not necessarily reflect deeper understanding when AI substitutes for key reasoning steps (Lo et al., 2024 [36]; Deng et al., 2025 [37]). The effect of AI on performance is therefore likely to vary with task design, especially with the balance between procedural correctness and conceptual interpretation.

Recent engineering education research also emphasizes that emerging technologies are reshaping not only assessment practices, but broader expectations around student engagement, interdisciplinary competency development, and authentic learning environments. Studies on virtual environments and immersive instructional approaches report that technology-enhanced learning experiences can increase engagement and participation in online engineering education, particularly when activities involve applied problem solving and interactive computational tasks (Barari & Sanders, 2024 [38]; Barari & Sanders, 2025 [39]; Ward et al., 2025 [40]; Siebold & Barari, 2026 [41]). Related work on T-shaped competency development and curriculum alignment further argues that modern engineering curricula increasingly require students to integrate disciplinary depth with computational fluency, systems thinking, and adaptive technical skills across domains (Barari et al., 2026a [42]; Barari & Siebold, 2026 [43]; Janke et al., 2025 [44]). Within this context, generative AI may represent not only a new instructional tool, but also a structural shift in how computational competencies are practiced, demonstrated, and evaluated in engineering coursework.

Another important strand of the literature focuses on assessment validity and academic integrity in AI-enabled settings. Institutional guidance and policy analyses warn against relying too heavily on AI-detection tools and instead stress alignment between AI use, learning goals, and disciplinary norms (Cotton et al., 2024 [12]; Wang et al., 2024 [5]; Jin et al., 2025 [45]). Empirical studies also show that AI-enabled misconduct is difficult to detect and is often hard to separate from legitimate AI-supported assistance, especially in coding and writing tasks (Chen et al., 2024 [13]; Fajt & Schiller, 2025 [18]; Karkoulou et al., 2025 [14]; Ortiz-Bonnin & Blahopoulou, 2025 [15]). Together, these findings point toward assessment redesign rather than purely enforcement-based responses.

Parallel developments in artificial intelligence research further reinforce the relevance of adaptive computational support systems in engineering education. Prior work in continual learning, representation learning, and incremental concept formation demonstrates that modern AI systems are increasingly capable of retaining procedural knowledge, adapting to new problem contexts, and generating increasingly robust outputs across iterative tasks (Barari & Barari, 2025 [46]; Barari & Kim, 2021 [47]; Barari et al., 2024 [48]; Barari et al., 2025 [49]). Related multi-course evaluations of large language models in engineering education show that AI performance varies substantially across cognitive levels and task structures, with particularly strong performance on procedural and execution-oriented tasks (Barari et al., 2026b [50]). These findings align with the present study's focus on programming and computational assignments, where generative AI capabilities may interact especially strongly with procedural dimensions of engineering work.

Taken together, the literature suggests that generative AI does not affect all task types in the same way. Programming and computational assignments, because they are rule-based and procedural, may be especially vulnerable to AI-mediated performance shifts. Yet there is still limited multi-term evidence showing whether AI availability is associated with higher baselines, lower score variability, or ceiling effects in authentic programming assessments (Groothuijsen et al., 2024 [26]; Sun et al., 2024 [25]; Garcia, 2025 [27]; Manorat et al., 2025 [33]).

1.2. Gaps in the Existing Literature

Despite the rapid growth of research on generative AI in higher education, several gaps remain, especially for programming and computational engineering coursework (Batista et al., 2024 [19]; Bhullar et al., 2024 [20]; Weng et al., 2024 [31]; Zhang et al., 2024 [22]).

First, much of the empirical work looks at a single term, a single cohort, or a classroom intervention. Those studies are useful for understanding short-term use patterns and student perceptions, but fewer studies compare multiple years of performance using the same assessment artifact. Now that generative AI has moved from novelty to routine availability, longer-term evidence is needed to see whether authentic programming and computational outcomes have shifted over time (Sun et al., 2024 [25]; Groothuijsen et al., 2024 [26]; Güner & Er, 2025 [28]; Suh et al., 2025 [29]).

Second, even when programming courses are discussed directly, many studies still emphasize workflow descriptions, student attitudes, or small pilots rather than systematic analysis of score distributions and mastery rates across repeated offerings of the same assignment. In domains where AI fits the task closely, such as code generation, debugging, and syntax-driven computation, the effect may show up not just as higher averages but as compressed score distributions and reduced separation among high-performing students. Evidence for that pattern is still limited (Ghosh, 2024 [23]; Yilmaz & Karaoglan Yilmaz, 2023 [24]; Park & Kim, 2025 [34]; Lepp & Kaimre, 2025 [35]).

Third, the literature often talks about AI's impact in broad pedagogical terms without clearly separating procedural execution tasks from conceptually demanding computational modeling tasks. Programming projects and MATLAB-based engineering assignments both involve code, but they do not place the same weight on conceptual reasoning, modeling choices, and interpretation. Distinguishing between these task types is important for understanding performance changes and for redesigning assessments responsibly (Lo et al., 2024 [36]; Deng et al., 2025 [37]; Kuhail et al., 2024 [10]).

These gaps point to the need for multi-term analyses of programming and computer-based engineering coursework using stable assessment artifacts in the post-AI period. Without that evidence, educators have limited basis for deciding whether higher scores reflect deeper learning, procedural assistance, or a more fundamental change in how students complete computational work (Jin et al., 2025 [45]; Wang et al., 2024 [5]).

To address this need, the present study examines student performance in two fully online undergraduate engineering courses: a programming final project in ENGR 115 and a MATLAB-based project in a fluid mechanics course in which students visualize streamlines for a prescribed velocity field. Both assignments are computer intensive, un-proctored, and graded with consistent rubrics across multiple sections and terms. By comparing a pre-AI period (2021–2022) with a post-AI period (2023–2025), the study asks whether the broad availability of generative AI corresponds to measurable changes in computational task outcomes.

Rather than assuming that AI use is either good or bad, this study focuses on what can be observed in real course data. If the results show higher baselines, reduced variability, or clear ceiling effects, then programming and MATLAB-based assignments may need to

be redesigned for AI-enabled settings. Possible responses include placing more emphasis on conceptual interpretation, asking students to explain code logic, adding validation steps, and using reflective components that separate procedural execution from underlying understanding (Chen et al., 2024 [13]; Cotton et al., 2024 [12]; Wang et al., 2024 [5]).

By focusing on programming and computational engineering tasks, which align closely with current AI capabilities, this study contributes empirical evidence to ongoing discussions in engineering education. The goal is to inform assignment design that preserves learning objectives while acknowledging how AI-assisted workflows are reshaping student performance (Kasneci et al., 2023 [1]; Belkina et al., 2025 [32]).

2. Methodology

This study uses a retrospective, quasi-experimental design to compare student performance in computational engineering coursework before and after generative AI became widely available. Course sections from the pre-AI period (2021–2022) are compared with sections from the post-AI period (2023–2025). No major instructional changes, AI-specific policies, or required tool use were introduced during this time. The analysis therefore captures naturally occurring performance patterns in authentic online courses as AI tools became readily accessible to students.

The post-2023 period is used as a temporal proxy for the widespread availability of generative AI tools rather than a direct measure of AI use. While no major instructional or curricular changes were introduced during this time, other factors, such as cohort variation, student preparedness, or broader shifts in the learning environment, may also have influenced the observed outcomes. The comparison should therefore be interpreted as reflecting changes over time under evolving conditions rather than the isolated effect of a single variable.

Because this study is based on a retrospective comparison of course performance across time, it does not establish causal relationships. The analysis captures naturally occurring changes in student outcomes as generative AI tools became widely accessible, but other factors, such as cohort characteristics, prior preparation, instructor variation, and grading practices, may also contribute to the observed patterns. The findings should therefore be interpreted as associations rather than direct effects of AI use.

The dataset includes two fully online undergraduate engineering courses:

- Programming Course, culminating in a comprehensive programming project.
- Fluid Mechanics, including a MATLAB-based project requiring visualization of streamlines for a prescribed velocity field.

Both courses are offered in multiple sections throughout the academic year, typically 8–10 sections per course each year. That structure introduces variation across terms and instructors while maintaining a stable course design. The courses were selected because the assignment prompts, grading rubrics, and learning objectives remained consistent across the years analyzed.

One major un-proctored computational assignment from each course was selected as the focal artifact. In the programming course, this was a comprehensive project in which students design, implement, and test a functional software solution that meets specified technical requirements. The project evaluates code structure, logical control flow, functional decomposition, input and output handling, and documentation. Submissions are graded with a standardized rubric used consistently across sections and terms.

In the fluid mechanics course, the focal assignment requires students to write a MATLAB script that visualizes streamlines for a prescribed velocity field. Students must implement the governing equations, generate appropriate plots, apply the correct boundary conditions, and format the graphical output according to the assignment criteria. The task

combines computational execution with engineering interpretation, and the grading rubric remained consistent across the analyzed terms.

Exams, quizzes, and any proctored or time-restricted assessments were excluded from analysis to isolate computer-based assignments completed independently by students.

Student performance data were extracted from the institutional learning management system in de-identified form. For each focal assignment, two outcome measures were analyzed:

- Percentage score (continuous variable);
- Mastery indicator, defined as achieving a score of 80% or higher.

Pre-AI and post-AI sections were compared to evaluate changes in overall performance levels, score distributions, and mastery attainment.

Performance was also broken down by available demographic characteristics, including age group and gender. These breakdowns support an exploratory look at whether performance shifts differ across student groups.

The dataset is drawn from two courses within a single institution and reflects a fully online, un-proctored instructional environment. These conditions may increase the accessibility and potential influence of generative AI tools compared to more controlled or in-person settings. As such, the findings should be interpreted within this specific context rather than as broadly generalizable across all engineering education environments.

Because this is an exploratory study, the analysis emphasizes descriptive comparisons rather than formal modeling of causal relationships. Mean and median scores, standard deviations, mastery rates, and score distributions were calculated for the pre-AI and post-AI cohorts, and graphical comparisons were used to examine changes in central tendency and variability. While these methods provide clear and interpretable insights into distributional shifts, more advanced statistical approaches, such as effect size estimation, confidence intervals, and mixed-effects models that account for clustering across sections and terms, were not applied in the current analysis and represent an important direction for future work.

The aim is to determine whether the emergence of generative AI corresponds to measurable changes in baseline performance, score compression, or ceiling effects in computational engineering assignments.

This study does not include direct measures of student use of generative AI tools. As a result, it is not possible to determine whether individual students used AI, how frequently they used it, or how it influenced their specific submissions. The analysis instead focuses on performance patterns observed across time as AI tools became widely accessible, and any interpretation related to AI use should be understood as indirect and inferential rather than directly measured.

3. Data Analysis and Results

To examine how performance changed after the emergence of generative AI, multiple complementary analyses were conducted using assignment-level data from the programming and fluid mechanics courses. The comparison between pre-AI and post-AI cohorts focuses on central tendency, score distributions, mastery rates, and subgroup patterns. Because these assignments are completed independently in an un-proctored online environment, they offer a useful view of how performance may have shifted as AI became widely available.

Results were also disaggregated by demographic characteristics to see whether the observed changes were widespread or concentrated in particular groups. Age and gender were used as the primary subgroup categories.

3.1. Programming Project

Figure 1 shows the percentage of students in four score bands (20–40, 40–60, 60–80, and 80–100) for ENGR 115 Assignment 5.3 in the pre-AI and post-AI periods. The shift toward the highest band after AI became widely available is clear.

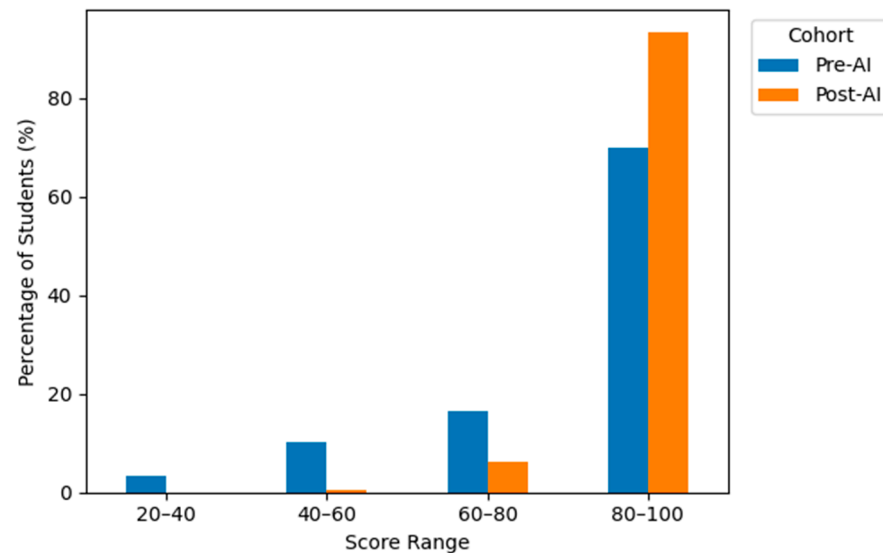


Figure 1. Percentage distribution of students across four score ranges (20–40, 40–60, 60–80, and 80–100) for Programming course project.

Before AI became widespread, scores were distributed across all four ranges, with substantial representation in the 40–60 and 60–80 bands. About 70% of students scored in the 80–100 range. In the post-AI cohort, more than 90% of students fell in the top band, and the lower and middle bands shrank sharply. Scores in the 20–40 and 40–60 ranges were almost absent.

A chi-square test of independence showed that this redistribution was statistically significant ($\chi^2 = 37.85$, $p < 0.001$). In other words, the movement of scores toward the upper range is unlikely to be due to random variation alone. The pattern points to a higher performance floor and a much larger share of students reaching strong outcomes on this programming assignment.

Figure 2 compares score distributions for the same assignment using boxplots. The post-AI cohort has a higher median, a narrower interquartile range, and a shorter lower tail. At the upper end, scores are packed more tightly near the maximum, which creates a visible ceiling effect.

This clustering near the upper bound suggests a reduction in score dispersion and a potential loss of resolution in the upper range of the grading scale. While not formally quantified in this study, the visual compression indicates that distinctions among higher-performing students may be less detectable under these conditions.

Read together, Figures 1 and 2 suggest that post-AI performance is not only higher but also more compressed. High achievement is common in both periods, but the post-AI cohort shows less spread and less separation among the strongest students. That pattern is consistent with the idea that generative AI supports the procedural parts of programming work and makes performance more uniform.

Figure 3 shows average programming project scores by age group for the pre-AI and post-AI cohorts. Scores rose noticeably for Gen X, Millennial, and Gen Z students alike, which suggests that the improvement was not confined to a single age group.

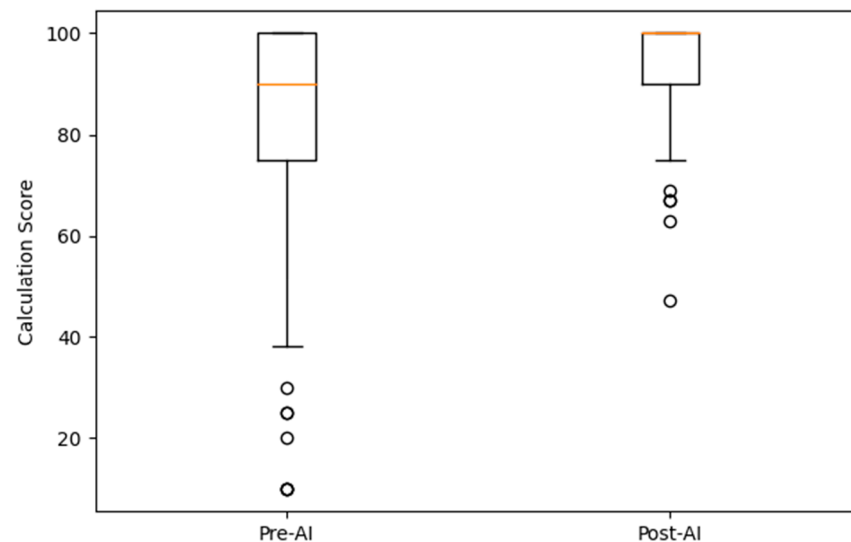


Figure 2. boxplot comparison of calculation scores for Programming course project.

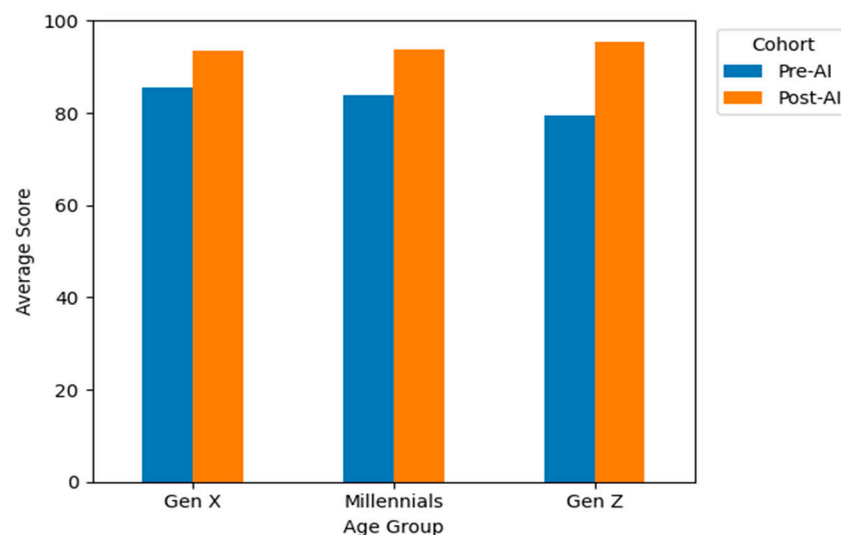


Figure 3. Average Programming Project score by age group (Gen X, Millennials, and Gen Z) for the pre-AI and post-AI cohorts.

In the pre-AI period, Gen X and Millennial students posted similar averages in the mid-80s, while Gen Z students scored slightly lower on average. In the post-AI period, all three groups moved into the mid-90s. The increase appears largest for Gen Z students, which narrows the gap seen before AI became widely available.

Overall, the age-based results suggest a broad rise in performance across groups, although these patterns are descriptive and not tested for statistical significance. Differences between the age groups become smaller in the post-AI period for this programming assignment.

Figure 4 reports average programming project scores by gender for the pre-AI and post-AI cohorts. The same upward pattern appears here: both male and female students score substantially higher in the post-AI period.

Before AI became widespread, average scores for male and female students were very similar, both in the low- to mid-80s. In the post-AI cohort, averages for both groups move into the mid-90s. The size of the increase is comparable, and the relative parity between male and female students remains intact.

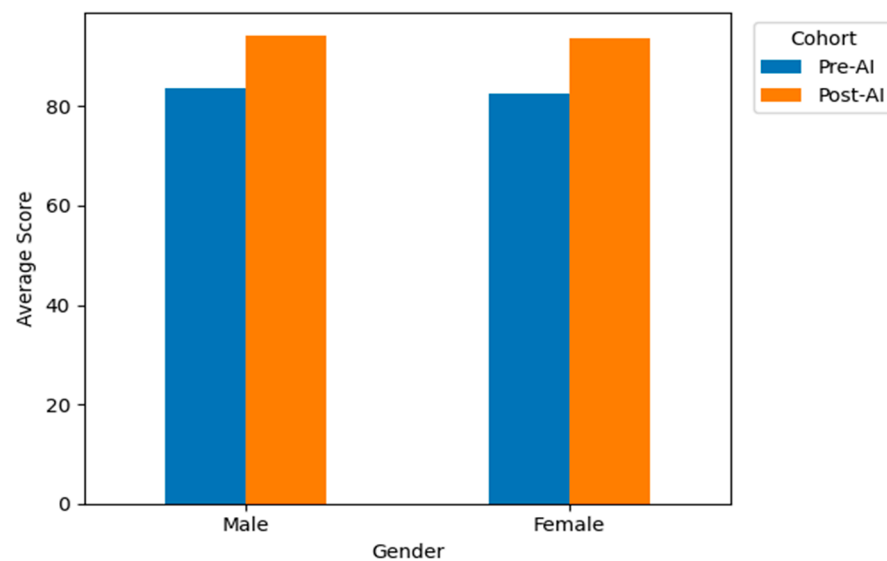


Figure 4. Average Programming Project score by gender for the pre-AI and post-AI cohorts.

This suggests that the performance gains are not concentrated within one gender. Access to generative AI appears to support the procedural demands of the programming task in a way that benefits different student groups similarly, without creating new disparities in average performance.

The gender results therefore echo the age results in a descriptive sense, with performance increasing across groups, although no inferential testing was conducted to determine the statistical significance of these patterns.

Figure 5 shows mastery rates, defined here as scores of 80% or higher, by gender before and after generative AI became widely available. Mastery increased sharply for both groups. For male students, it rose from about 72% to 96%; for female students, it rose from about 70% to 94%. The gains are large and nearly parallel, which points to a broad improvement rather than an advantage for one group.

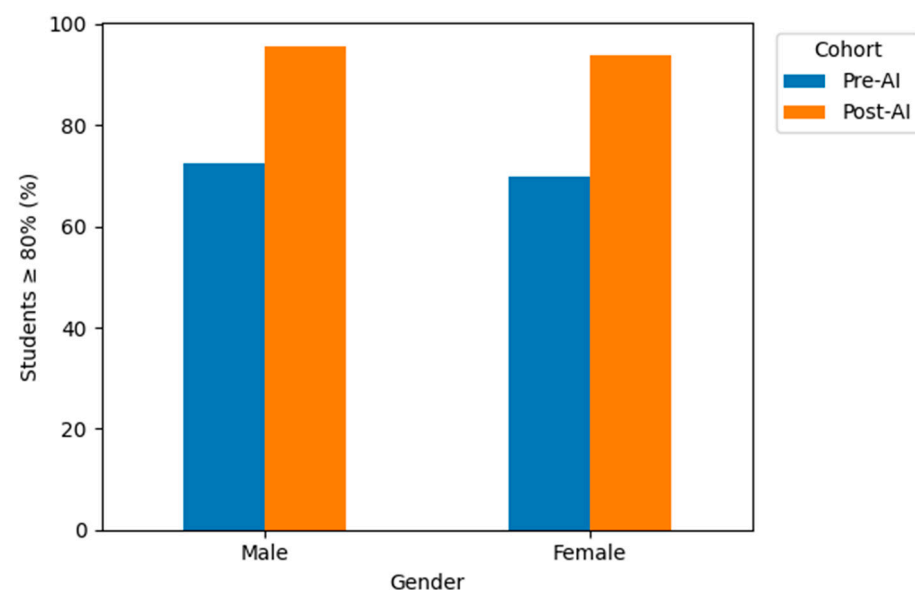


Figure 5. Mastery Rate (≥80%) by Gender for the Programming project comparing pre-AI and post-AI cohorts.

3.2. MATLAB Project

Figure 6 shows score distributions for the ESCI 206 MATLAB project before and after generative AI became widely available. The post-AI cohort contains a larger share of students in the 80–100 range, while mid-range scores become less common and the lowest bands nearly disappear. A chi-square test confirms that the shift is statistically significant ($\chi^2 = 17.53, p < 0.001$). At the same time, some variation remains below the top band, so this modeling task does not move to the ceiling as completely as the programming project.

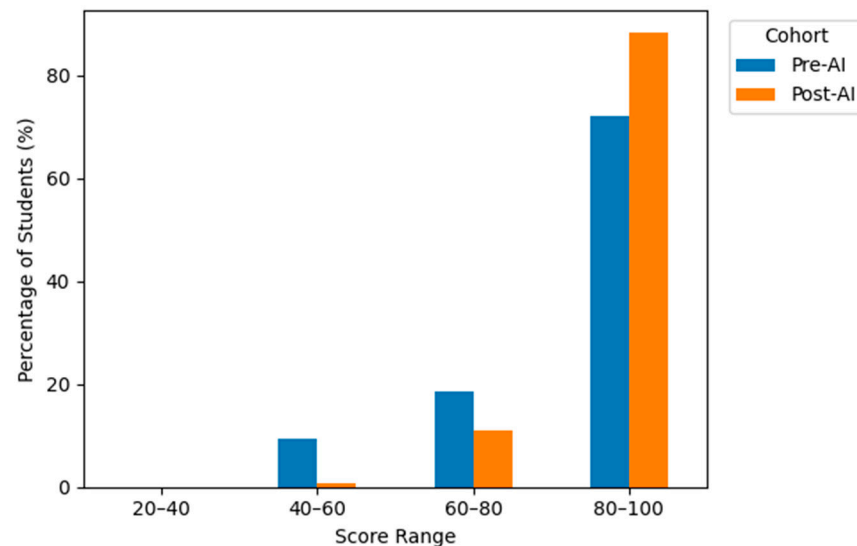


Figure 6. Percentage distribution of students across score ranges for MATLAB project for pre-AI and post AI.

Figure 7 compares MATLAB project scores with boxplots for the pre-AI and post-AI cohorts. Both groups perform well overall, with medians in the upper part of the grading scale, but the distributions are not identical.

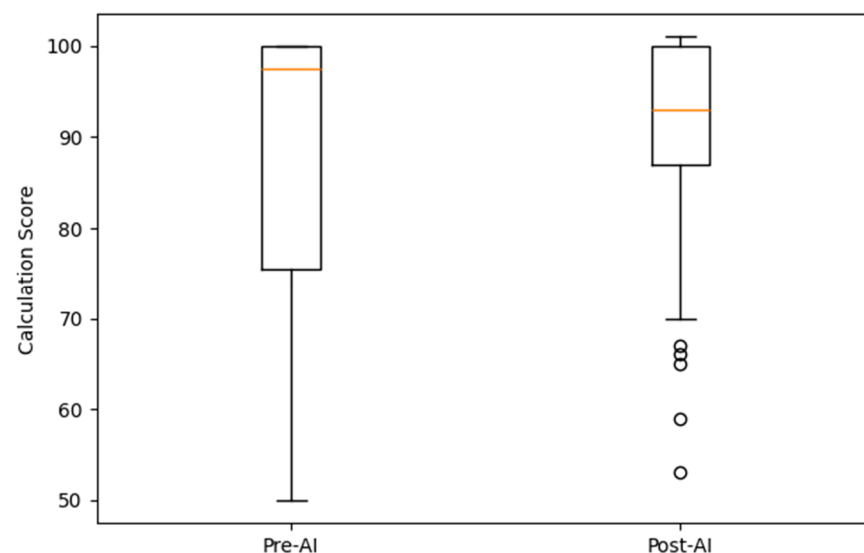


Figure 7. Boxplot Comparison of MATLAB Project Calculation Scores. The yellow lines indicate the median for each cohort, and the circles represent outlier data points.

The pre-AI cohort has a median near the upper 90s and a fairly wide interquartile range that extends downward into the mid-70s. The lower whisker reaches roughly 50,

which shows that some students scored well below the main cluster even though extreme low outliers were limited.

The post-AI cohort has a slightly lower median but a tighter interquartile range concentrated between the upper 80s and near-perfect scores. Several low outliers remain visible, including scores in the 50–70 range, which suggests that although most students performed well, a small subgroup still struggled.

Both cohorts cluster near the maximum score, so some ceiling effect is present in both periods. Unlike the programming assignment, however, the MATLAB project does not show a simple upward shift with uniformly compressed results. The post-AI cohort still includes noticeable variability and several low outliers.

This pattern suggests that AI may help many students on the MATLAB project, but it does not erase differences in performance. Assignments that require conceptual modeling, physical interpretation, and correct application of engineering principles appear to remain more sensitive to underlying understanding, even when AI support is available.

Figure 8 shows average MATLAB project scores by age group for the pre-AI and post-AI cohorts. All three groups improved after AI became widely available. Gen X and Millennial students show the largest gains and the highest post-AI averages, while Gen Z students also improve but remain slightly lower on average. The gains are broad, but some age-group differences remain.

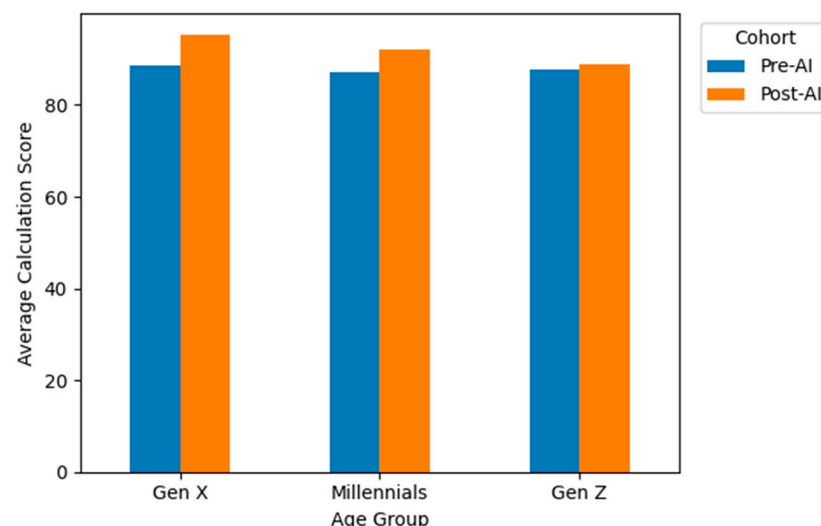


Figure 8. Average calculation score for MATLAB project by age group for the pre-AI and post-AI cohorts.

Figure 9 presents average MATLAB project scores by gender for the pre-AI and post-AI cohorts. Both male and female students score higher in the post-AI period. Male students show a clearer increase, moving from the high 80s into the low 90s, while female students remain strong in both periods with a smaller post-AI gain. Overall, the pattern suggests broad improvement without a meaningful change in the gender gap.

Figure 10 shows mastery rates of 80% or higher on the MATLAB project by gender. Mastery rises for both male and female students after AI becomes widely available. The larger increase appears among male students, rising from roughly seven in ten to about nine in ten, while female students also improve slightly from an already high starting point. The gains are broad, but they are less dramatic than in the programming project.

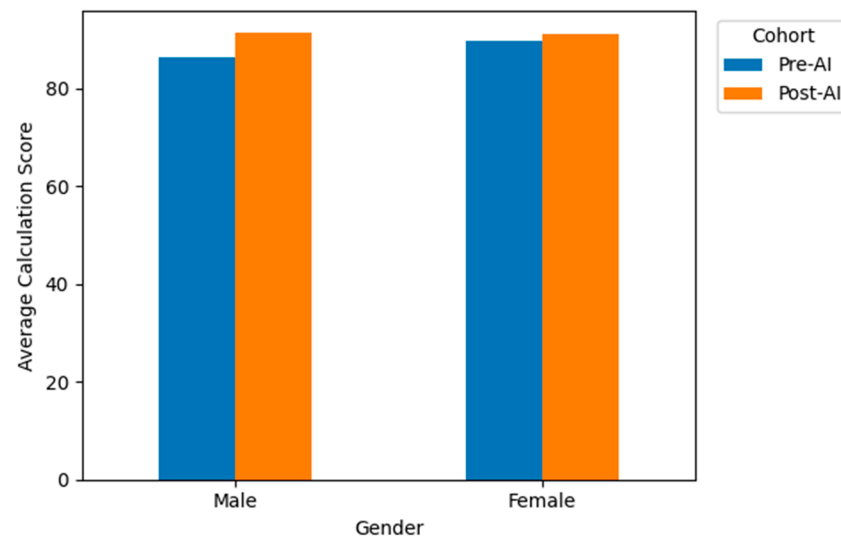


Figure 9. The average calculation score for MATLAB project by gender for the pre-AI and post-AI cohorts.

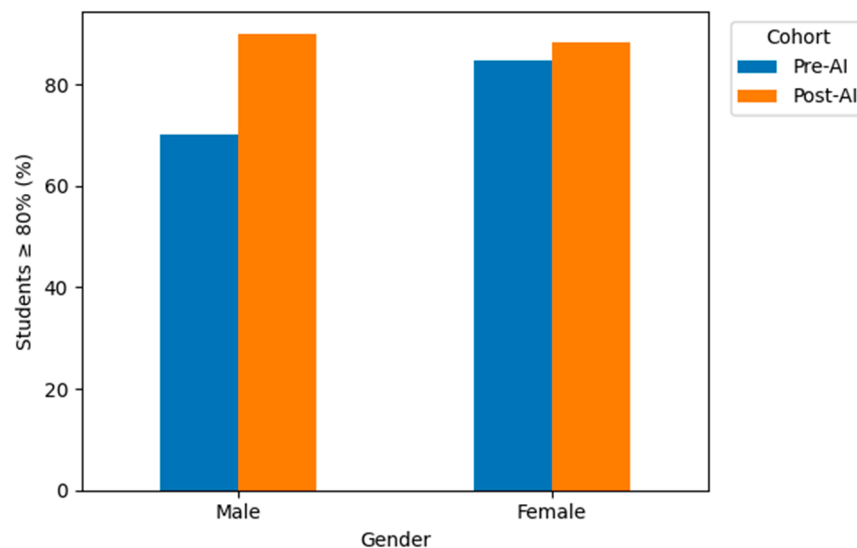


Figure 10. Mastery Rate ($\geq 80\%$) by Gender for the MATLAB project comparing pre-AI and post-AI cohorts.

4. Discussion

The results show that generative AI is associated with measurable changes in unproctored computational coursework in fully online engineering classes. At the same time, the size and shape of those changes depend strongly on the kind of task being assessed.

Importantly, this study does not directly observe or measure student use of generative AI. The interpretation that AI contributes to these patterns is based on the timing of widespread tool availability and the close alignment between AI capabilities and the procedural demands of the assignments. While this provides a plausible explanation, it should be understood as a proposed mechanism rather than a confirmed causal factor.

4.1. Differential Impact Across Computational Tasks

The programming checkpoint in ENGR 115 shows the clearest shift. In the post-AI period, average scores and mastery rates rise substantially, more students fall into the highest score band, and score variability drops. Lower- and mid-range scores become much less common, producing a clear ceiling effect. This is consistent with the close fit between

generative AI and procedural programming work such as syntax generation, debugging, and structured algorithmic output.

The MATLAB modeling project in ESCI 206 changes as well, but less dramatically. Post-AI cohorts score somewhat higher and more students reach the upper range, yet variability persists and low outliers remain. In other words, AI seems to help, but it does not lift this assignment in the same uniform way seen in the programming project.

That contrast matters. The MATLAB task still requires students to understand governing equations, translate physical ideas into code, and interpret the resulting plots. While this suggests that conceptual understanding continues to play a role in performance, this study does not directly measure students' reasoning processes or conceptual knowledge. The interpretation is therefore based on the structure of the task and observed performance patterns rather than direct empirical validation.

In addition, differences in assignment structure and grading criteria may contribute to these patterns. The programming project places greater emphasis on discrete correctness and successful execution, which may facilitate higher clustering of scores once procedural barriers are reduced. In contrast, the MATLAB task involves more open-ended modeling decisions and interpretation, which may preserve variability in performance. As such, the observed differences likely reflect a combination of task demands and assessment structure rather than a single underlying factor. Table 1 shows a comparative summary of observed student performance characteristics on programming and MATLAB-based computational assignments before (pre-AI) and after (post-AI) the widespread availability of generative AI tools in fully online engineering courses.

Table 1. Comparison of student performance characteristics in computational assignments before and after AI emergence.

Perspective	Pre-AI	Post-AI
Overall consistency	Variable outcomes across students and assignments	Markedly higher consistency in programming tasks; moderate improvement in modeling tasks
Baseline competency	Mixed performance with substantial mid-range scores	Elevated baseline performance across cohorts; sharp decline in low scores
Peak performance	High scores achieved by a subset of students	Majority of students achieving high scores in programming; strong but less uniform gains in MATLAB
Score spread	Broad distribution across score ranges	Compressed distribution in programming; narrower but still dispersed distribution in MATLAB

4.2. Implications for Assessment Validity

These results complicate the way traditional performance metrics are interpreted in AI-enabled courses. In programming contexts, higher scores may reflect faster execution and fewer procedural mistakes as much as deeper understanding. When score distributions compress near the top, the assignment becomes less useful for distinguishing levels of proficiency (Cotton et al., 2024 [12]; Jin et al., 2025 [45]).

The MATLAB results point to a different pattern. Because variability persists, assessments that require conceptual reasoning and modeling decisions may retain more discriminative power. That does not mean AI is unimportant in these settings; rather, its effect seems to depend on the nature of the task.

Although the ceiling effect is clearly visible in the distributional patterns, this study does not formally quantify changes in variance, skewness, or rubric-level discrimination. As a result, the implications for assessment validity are inferred from consistent visual and descriptive patterns rather than derived from formal statistical measures. Future work could strengthen this line of inquiry by applying distributional metrics and rubric-level analyses to better capture how AI-related performance compression affects the ability of assessments to differentiate levels of student understanding.

These findings also raise important questions about academic integrity and the role of AI in student work. Because this study does not track how students used generative AI or whether such use aligned with course policies, it is not possible to determine whether higher performance reflects increased understanding, effective tool-supported problem solving, or partial outsourcing of procedural tasks. This ambiguity highlights a broader challenge in AI-enabled learning environments: distinguishing between learning gains and shifts in how work is produced. Rather than focusing solely on detection, these results point toward the need for clearer alignment between instructional goals, assessment design, and acceptable AI use.

4.3. Mastery Rate Comparison

Mastery rates provide another way to see the difference between the two assignments. On the programming checkpoint, the share of students scoring 80% or higher jumps for both male and female students, rising from about seven in ten to well above nine in ten. On the MATLAB project, mastery also increases, but from an already high starting point and by a much smaller amount. This reinforces the task-dependent pattern seen throughout the results. In programming work, generative AI appears capable of moving a large share of students past a conventional competency threshold, which may reduce the assignment's ability to distinguish between levels of understanding. In modeling tasks, mastery improves, but conceptual demands still appear to limit how far AI support can substitute for domain knowledge. That distinction matters when instructors use mastery thresholds to judge learning in AI-enabled settings.

It is also important to note that the 80% threshold is used here as a consistent benchmark for comparison across cohorts rather than as a fixed indicator of cognitive mastery. In AI-enabled contexts, where procedural aspects of tasks may be supported by external tools, achieving this threshold may not reflect the same level of independent understanding as in earlier periods. This suggests that conventional mastery definitions may need to be reconsidered as AI becomes more integrated into student workflows.

4.4. Guidance for Assignment Design

The results suggest that assignment design in AI-enabled engineering courses should be matched carefully to the type of learning being assessed. The key issue is whether the task mainly rewards procedural execution, conceptual understanding, or some combination of both.

For programming assignments, instructors may need to shift some emphasis away from final output alone. Useful options include asking students to explain their code logic, justify algorithmic choices, describe testing strategies, and reflect on implementation decisions. Those additions can help keep the assessment aligned with the intended learning goals (Groothuisen et al., 2024 [26]; Lee et al., 2025 [30]; Shah et al., 2025 [51]).

For computational modeling projects such as the MATLAB assignment, current structures may already capture a wider set of competencies, including interpretation and conceptual reasoning. Even so, instructors may want to add checkpoints that probe understanding of physical principles, assumptions, and model validity so that AI-assisted code

generation does not stand in for subject-matter comprehension (Deng et al., 2025 [37]; Kuhail et al., 2024 [10]).

In addition, instructors may need to make expectations around AI use more explicit, including what forms of assistance are permitted and how students should document or reflect on their use of such tools. Aligning assessment practices with transparent AI policies can help ensure that performance outcomes more accurately reflect intended learning goals.

In AI-enabled settings, rubrics may also need to expand beyond correctness and functional output to better differentiate levels of understanding. Additional criteria could include the quality of reasoning and explanation, justification of algorithmic or modeling choices, evaluation of solution limitations, and the ability to interpret and validate results. These elements can help distinguish between surface-level completion and deeper engagement with the problem.

For courses such as ENGR 115, this does not imply removing programming tasks, but rather rebalancing them. Future designs may combine code generation with activities such as code auditing, debugging AI-generated solutions, explaining underlying logic, and validating outputs against expected behavior. This approach allows students to engage with modern AI-assisted workflows while still demonstrating core computational understanding.

4.5. Broader Educational Significance

The contrast between the two projects shows why task-specific evidence matters when judging AI's role in engineering education. Rather than treating generative AI as uniformly helpful or uniformly harmful, educators and researchers need to examine how its capabilities interact with the cognitive demands of particular assignments (Batista et al., 2024 [19]; Belkina et al., 2025 [32]; Manorat et al., 2025 [33]).

These findings also point to the need for continued adjustment in instructional practice. As AI tools improve, assignments that once measured student learning effectively may need to be redesigned to remain valid and rigorous. Knowing where AI changes performance substantially, and where it leaves important variation intact, can help educators build assessments that use AI productively without abandoning core engineering competencies (Weng et al., 2024 [31]; Jin et al., 2025 [45]).

The demographic analyses in this study are descriptive and do not include inferential statistical testing of subgroup differences or interaction effects. As a result, conclusions about the uniformity of performance changes across age and gender groups should be interpreted cautiously. Future research could apply regression models with interaction terms or hierarchical approaches to more rigorously examine how AI-related performance shifts may vary across student populations.

Another limitation of this study is the use of primarily descriptive statistical methods. Although the observed shifts in score distributions and mastery rates are consistent and statistically supported through chi-square tests, the analysis does not quantify effect sizes or account for potential clustering across course sections and terms. Future research could strengthen these findings by incorporating effect size measures, confidence intervals, and hierarchical or mixed-effects models to better capture variation across instructional contexts and improve the robustness of the results.

Further research is needed to directly examine the role of conceptual understanding in AI-supported computational tasks. Incorporating measures such as written explanations, reasoning-based prompts, or concept inventories would allow future studies to test whether performance differences across task types are driven by underlying understanding or by differences in how AI supports procedural versus conceptual work.

5. Conclusions

This study examined performance on un-proctored computational assignments in fully online engineering courses before and after generative AI became widely available. By comparing multiple years of data from stable assessment artifacts, it offers early empirical evidence of how AI is associated with changes in observed performance under authentic course conditions.

Performance on the programming checkpoint improved substantially: average scores rose, mastery rates increased, variability fell, and scores concentrated near the top of the scale. The MATLAB modeling project also improved, but with a smaller shift and more persistent dispersion. Together, these results show that the impact of generative AI is strongly task-dependent. Assignments that align closely with AI's procedural strengths, such as code generation, debugging, and structured algorithmic work, appear especially susceptible to broad performance elevation. Assignments that require conceptual modeling, interpretation, and domain knowledge remain less fully automated.

The mastery results underscore the same point. In programming, AI availability is associated with a much larger share of students moving above a conventional competency threshold, which may weaken the assignment's ability to separate different levels of understanding. In modeling tasks, mastery also improves, but the gains are smaller, suggesting that conceptual demands still constrain performance even when AI support is available.

These findings matter for instructional design in online engineering education. Instead of treating AI only as an academic integrity problem or only as a learning aid, instructors may need to reconsider how assignments are structured, how they are graded, and what kind of evidence they use to infer learning. Programming assignments may need more emphasis on reasoning, problem formulation, and explanation of implementation choices, while modeling assignments may benefit from even clearer attention to conceptual understanding.

In this context, fundamental engineering competency may need to be reinterpreted to include not only the ability to produce correct solutions, but also the ability to critically evaluate, validate, and refine computational outputs, including those generated with AI assistance. This shift emphasizes judgment, interpretation, and conceptual understanding as central components of engineering practice in AI-enabled environments.

This study offers a starting point for evidence-based adaptation in AI-enabled engineering education. As generative AI continues to evolve, further research should examine whether higher performance translates into durable learning, how students incorporate AI into problem-solving, and how assessment can remain valid while still making productive use of AI. Knowing where AI changes outcomes and where it does not will be essential for designing courses that use new tools without losing core engineering competencies.

Author Contributions: Conceptualization, G.B.; Methodology, G.B.; Validation, G.B.; Formal analysis, J.O.-M., K.J. and T.W.; Investigation, G.B. and K.S.; Resources, K.S.; Data curation, G.B.; Writing—original draft, G.B.; Writing—review and editing, J.O.-M., K.J., T.W. and K.S.; Visualization, G.B.; Supervision, G.B.; Funding acquisition, K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and approved by the Institutional Review Board (protocol code 26-017 and date of approval, 13 August 2025).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kasneci, E.; Sessler, K.; Küchemann, S.; Bannert, M.; Dementieva, D.; Fischer, F.; Gasser, U.; Groh, G.; Günemann, S.; Hüllermeier, E.; et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learn. Individ. Differ.* **2023**, *103*, 102274. [\[CrossRef\]](#)
2. Tlili, A.; Shehata, B.; Adarkwah, M.A.; Bozkurt, A.; Hickey, D.T.; Huang, R.; Agyemang, B. What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. *Smart Learn. Environ.* **2023**, *10*, 15. [\[CrossRef\]](#)
3. Lee, D.; Arnold, M.; Srivastava, A.; Plastow, K.; Strelan, P.; Ploechl, F.; Lekkas, D.; Palmer, E. The impact of generative AI on higher education learning and teaching: A study of educators' perspectives. *Comput. Educ. Artif. Intell.* **2024**, *6*, 100221. [\[CrossRef\]](#)
4. Baek, C.; Tate, T.; Warschauer, M. "ChatGPT seems too good to be true": College students' use and perceptions of generative AI. *Comput. Educ. Artif. Intell.* **2024**, *7*, 100294. [\[CrossRef\]](#)
5. Wang, H.; Dang, A.; Wu, Z.; Mac, S. Generative AI in higher education: Seeing ChatGPT through universities' policies, resources, and guidelines. *Comput. Educ. Artif. Intell.* **2024**, *7*, 100326. [\[CrossRef\]](#)
6. Finnie-Ansley, J.; Denny, P.; Becker, B.A.; Luxton-Reilly, A.; Prather, J. The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In *Australasian Computing Education Conference (ACE '22)*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 10–19.
7. Nguyen, N.; Nadi, S. An empirical evaluation of GitHub Copilot's code suggestions. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1–5.
8. Yetistiren, B.; Ozsoy, I.; Tüzün, E. Assessing the quality of GitHub Copilot's code generation. In *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '22)*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 62–71.
9. Prather, J.; Reeves, B.N.; Denny, P.; Becker, B.A.; Leinonen, J.; Luxton-Reilly, A.; Powell, G.; Finnie-Ansley, J.; Santos, E.A. "It's weird that it knows what I want": Usability and interactions with Copilot for novice programmers. *ACM Trans. Comput.-Hum. Interact.* **2024**, *31*, 1–31. [\[CrossRef\]](#)
10. Kuhail, M.A.; Mathew, S.S.; Khalil, A.; Berengueres, J.; Shah, S.J.H. "Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. *Sci. Comput. Program.* **2024**, *235*, 103111. [\[CrossRef\]](#)
11. Mo, R.; Wang, D.; Zhan, W.; Jiang, Y.; Wang, Y.; Zhao, Y.; Li, Z.; Ma, Y. Assessing and analyzing the correctness of GitHub Copilot's code suggestions. *ACM Trans. Softw. Eng. Methodol.* **2025**, *34*, 1–32. [\[CrossRef\]](#)
12. Cotton, D.R.E.; Cotton, P.A.; Shipway, J.R. Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. *Innov. Educ. Teach. Int.* **2024**, *61*, 228–239. [\[CrossRef\]](#)
13. Chen, B.; Lewis, C.M.; West, M.; Zilles, C.B. Plagiarism in the age of generative AI: Cheating method change and learning loss in an intro to CS course. In *Eleventh ACM Conference on Learning @ Scale (L@S '24)*, Atlanta, GA, USA, 18–20 July 2024; Association for Computing Machinery: New York, NY, USA, 2024; pp. 75–85. [\[CrossRef\]](#)
14. Karkoulou, S.; Sayegh, N.; Sayegh, N. ChatGPT unveiled: Understanding perceptions of academic integrity in higher education: A qualitative approach. *J. Acad. Ethics* **2025**, *23*, 1171–1188. [\[CrossRef\]](#)
15. Ortiz-Bonnin, S.; Blahopoulou, J. Chat or cheat? Academic dishonesty, risk perceptions, and ChatGPT usage in higher education students. *Soc. Psychol. Educ.* **2025**, *28*, 113. [\[CrossRef\]](#)
16. Strzelecki, A. To use or not to use ChatGPT in higher education? A study of students' acceptance and use of technology. *Interact. Learn. Environ.* **2024**, *32*, 5142–5155. [\[CrossRef\]](#)
17. Oc, Y.; Gonsalves, C.; Quamina, L.T. Generative AI in higher education assessments: Examining risk and tech-savviness on student's adoption. *J. Mark. Educ.* **2025**, *47*, 138–155. [\[CrossRef\]](#)
18. Fajt, B.; Schiller, E. ChatGPT in academia: University students' attitudes towards the use of ChatGPT and plagiarism. *J. Acad. Ethics* **2025**, *23*, 1363–1382. [\[CrossRef\]](#)
19. Batista, J.; Mesquita, A.; Carnaz, G. Generative AI and higher education: Trends, challenges, and future directions from a systematic literature review. *Information* **2024**, *15*, 676. [\[CrossRef\]](#)
20. Bhullar, P.S.; Joshi, M.; Chugh, R. ChatGPT in higher education: A synthesis of the literature and a future research agenda. *Educ. Inf. Technol.* **2024**, *29*, 21501–21522. [\[CrossRef\]](#)
21. Ogunleye, B.; Adepoju, O.; Chen, H. A systematic review of generative AI for teaching and learning in higher education. *Educ. Sci.* **2024**, *14*, 636. [\[CrossRef\]](#)

22. Zhang, X.; Zhang, P.; Shen, Y.; Liu, M.; Wang, Q.; Gašević, D.; Fan, Y. A systematic literature review of empirical research on applying generative artificial intelligence in education. *Front. Digit. Educ.* **2024**, *1*, 223–245. [\[CrossRef\]](#)
23. Ghosh, S. ChatGPT as a tool for equitable education in engineering classes. In Proceedings of the 2024 ASEE Annual Conference Exposition, Portland, OR, USA, 23–26 June 2024.
24. Yilmaz, R.; Karaoglan Yilmaz, F.G. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Comput. Hum. Behav. Artif. Hum.* **2023**, *1*, 100005. [\[CrossRef\]](#)
25. Sun, D.; Boudouaia, A.; Zhu, C.; Li, Y. Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *Int. J. Educ. Technol. High. Educ.* **2024**, *21*, 14. [\[CrossRef\]](#)
26. Groothuisen, S.; van den Beemt, A.; Remmers, J.C.; van Meeuwen, L.W. AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Comput. Educ. Artif. Intell.* **2024**, *7*, 100290. [\[CrossRef\]](#)
27. Garcia, M.B. Teaching and learning computer programming using ChatGPT: A rapid review of literature amid the rise of generative AI technologies. *Educ. Inf. Technol.* **2025**, *30*, 16721–16745. [\[CrossRef\]](#)
28. Güner, H.; Er, E. AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Educ. Inf. Technol.* **2025**, *30*, 12681–12707. [\[CrossRef\]](#)
29. Suh, J.; Lee, K.; Lee, J. Programming education with ChatGPT: Outcomes for beginners and intermediate students. *Educ. Inf. Technol.* **2025**, *30*, 19511–19536. [\[CrossRef\]](#)
30. Lee, H.-Y.; Chen, P.-H.; Lin, C.-J.; Huang, Y.-M.; Wu, T.-T. Leveraging ChatGPT for personalized reflective learning in programming education: Effects on self-efficacy, higher-order thinking, and project implementation skills. *Educ. Inf. Technol.* **2025**, *30*, 24815–24854. [\[CrossRef\]](#)
31. Weng, X.; Xia, Q.; Gu, M.; Rajaram, K.; Chiu, T.K.F. Assessment and learning outcomes for generative AI in higher education: A scoping review on current research status and trends. *Australas. J. Educ. Technol.* **2024**, *40*, 37–55. [\[CrossRef\]](#)
32. Belkina, M.; Daniel, S.; Nikolic, S.; Haque, R.; Lyden, S.; Neal, P.; Grundy, S.; Hassan, G.M. Implementing generative AI (GenAI) in higher education: A systematic review of case studies. *Comput. Educ. Artif. Intell.* **2025**, *8*, 100407. [\[CrossRef\]](#)
33. Manorat, P.; Tuarob, S.; Pongpaichet, S. Artificial intelligence in computer programming education: A systematic literature review. *Comput. Educ. Artif. Intell.* **2025**, *8*, 100403. [\[CrossRef\]](#)
34. Park, A.; Kim, T. Code suggestions and explanations in programming learning: Use of ChatGPT and performance. *Int. J. Manag. Educ.* **2025**, *23*, 101119. [\[CrossRef\]](#)
35. Lepp, M.; Kaimre, J. Does generative AI help in learning programming: Students' perceptions, reported use and relation to performance. *Comput. Hum. Behav. Rep.* **2025**, *18*, 100642. [\[CrossRef\]](#)
36. Lo, C.K.; Hew, K.F.; Jong, M.S.-Y. The influence of ChatGPT on student engagement: A systematic review and future research agenda. *Comput. Educ.* **2024**, *219*, 105100. [\[CrossRef\]](#)
37. Deng, R.; Jiang, M.; Yu, X.; Lu, Y.; Liu, S. Does ChatGPT enhance student learning? A systematic review and meta-analysis of experimental studies. *Comput. Educ.* **2025**, *227*, 105224. [\[CrossRef\]](#)
38. Barari, G.; Sanders, B. The effectiveness of virtual environments for increasing engagement in engineering technology courses. In Proceedings of the 2024 Conference for Industry and Education Collaboration, Garden Grove, CA, USA, 18–21 March 2024.
39. Barari, G.; Sanders, B. Application of FLASH to Forecast Student Engagement in Online Engineering Courses. In Proceedings of the 2025 ASEE Annual Conference & Exposition, Montreal, QC, Canada, 22–25 June 2025.
40. Ward, T.; Jenab, K.; Ortega-Moody, J.; Barari, G.; Molina Acosta, L.D.C. Virtual classrooms, real impact: A framework for introducing virtual reality to K–12 STEM learning based on best practices. *Appl. Sci.* **2025**, *15*, 11356. [\[CrossRef\]](#)
41. Siebold, K.; Barari, G. Enhancing Undergraduate Engineering Education through HLA-Based Simulation Exploration Experience (SEE). In Proceedings of the 2026 Computer-Utilized Education Society (CIEC), New Orleans, LA, USA, 4 February 2026.
42. Barari, G.; Janke, C.; Lin, Y. Integrating T-shaped competency models in interdisciplinary aviation and engineering education. *Coll. Aviat. Rev. Int.* **2026**, *44*, 136–159. Available online: <https://ojs.library.okstate.edu/osu/index.php/CARI/article/view/10584/9344> (accessed on 16 April 2026).
43. Barari, G.; Siebold, S. Curriculum Alignment through the Industry Pulse Platform. In Proceedings of the 2026 Computer-Utilized Education Society (CIEC), New Orleans, LA, USA, 4 February 2026.
44. Janke, C.; Lin, Y.; Barari, G. BOARD# 15: From drones to airplanes, Lessons learned from uncrewed aviation for teaching innovative and sustainable electric aviation propulsion. In Proceedings of the 2025 ASEE Annual Conference & Exposition, Montreal, QC, Canada, 22–25 June 2025; American Society for Engineering Education: Washington, DC, USA, 2025.
45. Jin, Y.; Yan, L.; Echeverria, V.; Gašević, D.; Martinez-Maldonado, R. Generative AI in higher education: A global perspective of institutional adoption policies and guidelines. *Comput. Educ. Artif. Intell.* **2025**, *8*, 100348. [\[CrossRef\]](#)
46. Barari, N.; Barari, G. Continual learning models in aviation systems. *Coll. Aviat. Rev. Int.* **2025**, *43*. Available online: <https://ojs.library.okstate.edu/osu/index.php/CARI/article/view/10310> (accessed on 16 April 2026).
47. Barari, N.; Kim, E. Linking sparse coding dictionaries for representation learning. In 2021 International Conference on Rebooting Computing (ICRC); IEEE: New York, NY, USA, 2021; pp. 84–87.

48. Barari, N.; Lian, X.; MacLellan, C.J. Incremental concept formation over visual images without catastrophic forgetting. *arXiv* **2024**, arXiv:2402.16933.
49. Barari, N.; Kim, E.; MacLellan, C. Explaining Robustness to Catastrophic Forgetting Through Incremental Concept Formation. *arXiv* **2025**, arXiv:2510.23756. [[CrossRef](#)]
50. Barari, G.; Basu, D.; Lin, Y.; Ortega-Moody, J. *Evaluating Large Language Models in Engineering Education: A Multi-Course Assessment Across Bloom's Taxonomy*; IEEE Access: New York, NY, USA, 2026. [[CrossRef](#)]
51. Shah, A.; Chernova, A.; Tomson, E.; Porter, L.; Griswold, W.G.; Soosai Raj, A.G. Students' use of GitHub Copilot for working with large code bases. In Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS '25), Pittsburgh, PA, USA, 26 February–1 March 2025; pp. 1050–1056.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.