Jiang, B. et al. (Eds.) (2025). Proceedings of the 33rd International Conference on Computers in Education.
AsiaPacific Society for Computers in Education

# Impact of LLM Feedback on Learner Persistence in Programming

**Yiqiu ZHOU[a*], Maciej PANKIEWICZ[b], Luc PAQUETTE[a] & Ryan S. BAKER[b]**
[a]*University of Illinois Urbana-Champaign, USA*
[b]*University of Pennsylvania, USA*
*yiqiuz3@illinois.edu

**Abstract:** This study examines how Large Language Model (LLM) feedback generated for compiler errors impacts learners' persistence in programming tasks within a system for automated assessment of programming assignments. Persistence, the ability to maintain effort in the face of challenges, is crucial for academic success but can sometimes lead to unproductive "wheel spinning" when students struggle without progress. We investigated how additional LLM feedback based on the GPT-4 model, provided for compiler errors affects learners' persistence within a CS1 course. Specifically, we examined whether its impacts differ based on task difficulty, and if the effects persist after the feedback is removed. A randomized controlled trial involving 257 students across various programming tasks was conducted. Our findings reveal that LLM feedback improved some aspects of students' performance and persistence, such as increased scores, a higher likelihood of solving problems, and a lower tendency to demonstrate unproductive "wheel spinning" behavior. Notably, this positive impact was also observed in challenging tasks. However, its benefits did not sustain once the feedback was removed. The results highlight both the potential and limitations of LLM feedback, pointing out the need to promote long-term skill development and learning independent of immediate AI assistance.

**Keywords:** Autograding, Automated feedback, Programming, Persistence, GPT, LLM

## 1. Introduction

Persistence, the ability to maintain effort in the face of challenges, is a critical factor in student success and learning outcomes (Cloninger et al., 1993). This concept is closely linked to grit, productive struggle, and growth mindset that encourage persistent efforts (Cutts et al., 2010; Duckworth et al., 2007). Previous studies have demonstrated that higher persistence predicts course completion (Wang & Baker, 2018), college enrollment (Adjei et al., 2021) and academic success (McClendon et al., 2017). Persistence can be particularly relevant to computer science education, as programming often involves debugging. This process can be challenging due to various factors such as misconceptions, fragile knowledge, and the 'superbug' (the incorrect assumption that the programming language has intelligent interpretive powers) (McCauley et al., 2008). Debugging often requires students to engage in iterative testing to identify and resolve issues, demanding persistence as students must maintain their efforts despite the frustration and setbacks.

However, not all persistence in programming tasks is productive. Students may exhibit unproductive persistence, or "wheel spinning." This occurs when students make repeated unsuccessful attempts without making meaningful progress or achieving mastery within a reasonable timeframe (Beck & Gong, 2013). Wheel spinning is particularly prevalent among novice programmers who struggle with syntax and find it difficult to interpret compiler error messages (Hartmann et al., 2010). Without appropriate guidance, students may misinterpret error messages, apply ineffective debugging strategies, or repeatedly attempt incorrect solutions, hindering their learning (Nienaltowski et al., 2008).

Recent advancements in artificial intelligence, particularly Large Language Models (LLMs) like GPT, offer new opportunities for personalized feedback in CS education. Examples include Codehelp (Liffiton et al., 2023) and Codeaid (Kazemitabaar et al., 2024). LLMs have also been used to generate code explanations (Leinonen, Denny, et al., 2023), and debugging hints (Leinonen, Hellas, et al., 2023; McBroom et al., 2021). However, recent studies suggest that novices may struggle to interpret AI-generated feedback, leading to cognitive overload and ineffective problem-solving strategies (Leinonen, Hellas, et al., 2023). Concerns have also emerged about students' potential over-reliance, where students merely copy AI-suggested solutions without fully understanding them, hindering their ability to internalize programming concepts and independently solve problems (Becker et al., 2023; Finnie-Ansley et al., 2022). This highlights the need for further investigation into whether LLM feedback fosters genuine problem-solving or reinforces unproductive behaviors.

This study contributes to this discussion by examining whether LLM-generated feedback enhances learners' productive persistence in programming tasks and whether its benefits persist after AI assistance is withdrawn. We hypothesize that the personalized LLM feedback could help students engage in more effective debugging and problem-solving strategies while mitigating unproductive persistence (i.e., repeated unsuccessful attempts). To rigorously test this hypothesis, we conducted a randomized controlled experiment with 257 CS1 students using an automated assessment platform. Students were randomly assigned to receive either standard compiler error messages or standard compiler error messages plus LLM-enhanced feedback with error explanation, allowing us to compare their persistence behaviors across programming tasks of varying difficulty. Beyond assessing the immediate effects of AI-generated feedback, our study uniquely examines its long-term impact by analyzing student performance after LLM feedback is removed. Specifically, we examine:

**RQ1**: How does LLM feedback affect learners' persistence in programming tasks?

**RQ2**: Does LLM feedback show a significant effect on learners' persistence, specifically in challenging programming tasks?

**RQ3**: How does the removal of LLM feedback affect the performance of students who have previously received it? Are there lasting effects on their persistence?

## 2. Literature Review

### *2.1 Operationalization of Persistence*

Persistence research has recently expanded to computer science domain (Flores & Rodrigo, 2020; Pinto et al., 2021), but the iterative nature of programming poses challenges for traditional models. Programming involves cycles of coding, testing, and debugging (McCauley et al., 2008). Common approaches like correct-in-a-row count (Gong & Beck, 2015) or Bayesian Knowledge Tracing (BKT) (Käser et al., 2016) often assume binary correctness and clear skill mapping, which do not reflect the multifaceted progress typical in programming tasks. In programming tasks, students often make incremental progress in various aspects of the code (e.g., syntax, logic, efficiency) simultaneously. Another challenge is the issue of credit assignment. While intelligent tutoring systems (ITSs) use well-defined skill structure to track learning (Fang et al., 2017; Gong & Beck, 2015), programming tasks often involve overlapping concepts, making it hard to pinpoint which specific skill or concept underlies errors. As a result, the criteria traditionally used to define mastery and differentiate between productive and unproductive persistence in ITS may not directly translate to the programming context. Thus, this work leverages the metrics defined in Pinto et al. (2021), which are tailored to measure persistence in programming tasks. This approach considers the characteristics of programming tasks and provides a more nuanced view of persistence in programming.

### *2.2 LLM in CS education and Research Motivation*

LLM have shown promise in CS education by providing explanations for programming concepts and errors, and personalized hints (Kazemitabaar et al., 2024; Leinonen, Denny, et al., 2023; Liffiton et al., 2023; McBroom et al., 2021; Sit et al., 2024). In introductory programming courses, ChatGPT has been found to deliver consistent formative feedback (Kiesler et al., 2023) and improve task performance with positive student ratings (Pankiewicz

& Baker, 2023). However, limitations remain: GPT-based tools may struggle with complex queries and software testing questions (Qureshi, 2023) and can yield mixed results, especially when deeper understanding of programming concepts is required (Pankiewicz & Baker, 2024). Moreover, concerns around over-reliance have emerged (Xue et al., 2024). One risk is the illusion of understanding, where students believe they have grasped a concept simply because they can produce working solutions with AI (Becker et al., 2023). Tools like OpenAI Codex may enable students to bypass the problem-solving process instead of engaging in debugging and critical thinking (Finnie-Ansley et al., 2022). Research on Copilot has similarly shown that some students develop a habit of outsourcing their cognitive effort to AI (Prather et al., 2023). These concerns highlight that unregulated use may also undermine the acquisition of problem solving skills. Recent research has shown that students' intentions to use these tools are shaped not only by immediate needs such as academic stress and risk-taking tendencies (Samson et al., 2024) but also by broader psychological factors such as personal innovativeness, trust, and self-efficacy (Elinzano & Ching, 2024).

This study aims to bridge the gap between the need for persistence in programming and the potential of AI-generated feedback. Most existing studies on AI-generated feedback have focused primarily on immediate learning outcomes, with limited attention to whether these benefits continue after the AI support is withdrawn. This study examines how LLM feedback impacts student persistence both immediately and over the long term. Specifically, we investigate whether LLM feedback reduces unproductive persistence and whether this benefit persists once removed. By examining the interaction between LLM feedback and task difficulty, this study also aims to inform best practices regarding when and how AI assistance should be used. Ultimately, this research provides evidence-based insights into the conditions under which AI-generated feedback supports meaningful learning in programming.

## 3. Method

### *3.1 Participant and Learning Environment*

The study involved 257 computer science students, enrolled in a mandatory first semester CS1 course at a large European university. Topics include types and variables, conditional statement, recursion, loops, and loops with arrays. Students were randomly assigned to the Experimental group (N=129) or the Control group (N=128). Participants comprised 182 males and 74 females, with gender information missing for one participant. Consent was obtained from students. Tasks were assigned on an online platform for automated assessment, through 14 modules containing between 5 to 14 problems each, with modules typically assigned on a weekly basis. The platform's assessment process involved compiling and testing the submitted code written in C# language, followed by providing students with feedback that included: compiler messages (line, error message, and id), results for each unit test (with input values and expected outcome), and overall score (the percentage of successful unit tests 0-100%).

### *3.2 Experiment Setting and Prompt Design*

Both the control and experimental groups had access to 141 programming tasks distributed across 14 modules (as shown in Figure 1). For the first half of each module (72 tasks), the experimental group received additional feedback on compiler errors from OpenAI's GPT-4 model (gpt-4-0613), while the control group received only standard messages. For the remaining 69 tasks, the GPT-4 feedback was withdrawn for the experimental group to assess any lasting effects. Feedback was created via a request to the OpenAI Chat Completion API using the approach to prompt generation that consisted of the following parts: 1) general instructions for the assistant, 2) assignment text, 3) student code, and 4) results of the code evaluation. The prompt was extended by examples of an ideal response containing three elements (in English): an explanation of the error (example from one of generated hints: "The compiler message *;* expected means that a semicolon is missing in the line *int a = 2*b*"), a solution strategy (example: "To fix this error, you need to add a semicolon *;* at the end of the line int *a = 2*b*"), and an educational element regarding the underlying concept (example: "Remember that in C#, every statement must be terminated with a semicolon"). This structure was designed to provide scaffolded support that guided students toward a solution without

offering explicit code corrections. Hints were generated in approximately 20 seconds as a re-viewable pop-up and offered once per submission, with no limit on attempts. The researchers also conducted a qualitative review of a sample of hints to ensure their general accuracy and pedagogical value.

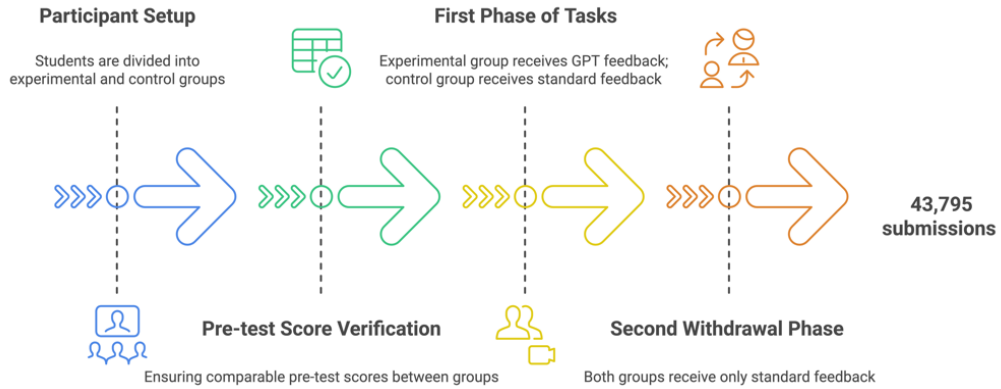


*Figure 1*. Experiment Setting

### *3.3 Persistence Elements*

This study utilizes log data from 43,795 submissions to extract eight features, adapted from (Pinto et al., 2021), that quantify various aspects of persistence. The choice of thresholds—such as the 75th percentile—was informed by prior work (Pinto et al., 2021). For time-related features, we subtracted the actual LLM feedback generation time or used an average of 20 seconds when this record was unavailable. These features were computed per task and then aggregated to the student level by averaging across tasks. Prior to aggregation, all features were standardized within each task (i.e., z-scored across all students) to account for variation in task difficulty and baseline behavior. The final feature list consisted of:

***Number of submissions***: The total number of submission attempts made by the student for a specific task.

***Beyond attempt threshold***: The proportion of tasks for which a student exceeded the 75th percentile of submission attempts (calculated across all students for each task).

***Solved***: The proportion of attempted tasks that a student successfully solved, where a task is considered solved if the final submission received a perfect score.

***Last submission score***: The average score of a student's final submissions across all attempted tasks.

***Time since first submission***: The average time a student spent on tasks, calculated from their first submission to their last for each task. For tasks with LLM assistance, time is adjusted to account for LLM feedback generation.

***Beyond time threshold***: The proportion of tasks for which a student's total time spent exceeded the 75th percentile of time spent (calculated across all students for each task).

***Taking a break***: The proportion of tasks where a student had at least one gap of two hours or more between consecutive submissions. This two-hour threshold was determined based on an inspection of the distribution of submission intervals across all students, where breaks of this length represented a noticeable departure from typical short pauses.

***Wheel spinning***: The proportion of tasks where a student made 7 or more submissions and didn't solve the problem (final submission score was below 1). The 7-submission threshold corresponds to the mean plus one standard deviation of attempt counts in our dataset, aligning with prior work to identify patterns of unproductive persistence (Gong & Beck, 2015).

These features were designed to capture different aspects of student persistence, including effort (number of submissions), efficiency (time spent), success rate (solved tasks), and potential struggles (wheel spinning).

### *3.4 Analysis*

We employed rank-based linear regression from the Rift package in R due to non-normality in the data (Kloke & McKean, 2012), which is robust to non-normal data distributions. To further assess the stability and reliability of the model estimates, we conducted Monte Carlo simulations with 1000 iterations for each model. These simulations allowed us to evaluate the

sensitivity of our results to sampling variability and ensure inference robustness. Additionally, we verified that there were no significant differences in the pre-test scores between the control and experimental groups using the Mann-Whitney U test ($U$ = 4441.0, $p$ = 0.121). The persistence metrics, outlined in Section 3.4, served as dependent variables. Each regression model included two independent variables: feedback type (additional LLM feedback vs. control) and pre-test scores, the latter serving as a covariate to control for prior knowledge differences. Our analysis addressed three specific research questions. **Impact on All Tasks (RQ1)**: The impact of LLM feedback was evaluated across all tasks. **Impact on Challenging Tasks (RQ2)**: Challenging tasks were defined by three criteria: average score below 100%, total time spent above the 75th percentile of all tasks, and total submission attempts above the 75th percentile. By analyzing these tasks separately, we aimed to gain nuanced insights into the impacts of LLM feedback on learners' persistence when faced with more difficult programming tasks. **Post-feedback Removal Effects (RQ3)**: The persistence of observed benefits was examined after LLM feedback was removed.

## 4. Results

### *4.1 RQ1: Impact on All Tasks*

To investigate the impact of additional LLM feedback on learners' persistence, we employed rank-based regression to analyze each persistence metric, with pre-test scores as a covariate. The results are presented in Table 1, which reports regression coefficients  as the estimated effect of group membership on persistence metrics, Monte Carlo standard errors as the measure of estimation precision, and p-values to indicate statistical significance. We also calculated Kendall's Tau $\tau$ as the rank-based measure of association strength and direction. This ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation), with values closer to 0 indicating weaker relationships. Tau values near ±0.10 suggest weak associations, ±0.30 moderate associations, and ±0.50 or higher strong associations. The experimental group, for tasks with GPT feedback enabled, showed significantly higher last submission scores ($\beta$ = 0.029, $p$ = 0.041, $\tau$ = 0.137) and a marginally higher likelihood of solving the problems ($\beta$ = 0.028, $p$ = 0.061, $\tau$ = 0.095) compared to the control group. Furthermore, the experimental group exhibited significantly lower rates of going beyond the time threshold ($\beta$ = -0.098, $p$ =0.00, $\tau$ = -0.143), taking breaks ($\beta$ = -0.035, $p$ = 0.050, $\tau$ = -0.089), and wheel spinning ($\beta$ = -0.025, $p$ = 0.032, $\tau$ = -0.134) compared to the control group. These findings indicate that learners who received GPT feedback were less likely to spend excessive time on tasks, take breaks, or engage in unproductive persistence (i.e., wheel spinning) than those who received only standard compiler error messages.

Table 1. *Combined Regression Analysis for All Tasks*

| **Variable** | **GPT Feedback Enabled (Phase 1)** | | | **GPT Feedback Disabled (Phase 2)** | | |
|---|---|---|---|---|---|---|
| | Coefficient (Std.E) | MC Mean (Std.D) | p-value | Coefficient (Std.E) | MC Mean (Std.D) | p-value |
| ***Number of submissions (count)*** | | | | | | |
| Intercept | 7.192 (0.479) | 7.256 (0.487) | < 0.001*** | 7.101 (0.446) | 7.082 (0.518) | < 0.001*** |
| Experiment | 0.381 (0.417) | -0.439 (0.412) | 0.363 | 0.460 (0.391) | 0.441 (0.411) | 0.242 |
| Pre-test | -0.556 (0.090) | -0.550 (0.075) | < 0.001*** | -0.438 (0.084) | -0.433 (0.085) | < 0.001*** |
| ***Beyond Attempt Threshold (proportion)*** | | | | | | |
| Intercept | 0.141 (0.015) | 0.141 (0.016) | < 0.001*** | 0.120 (0.019) | 0.123 (0.020) | < 0.001*** |
| Experiment | -0.014 (0.014) | -0.016 (0.014) | 0.307 | 0.029 (0.018) | 0.028 (0.020) | 0.113 |
| Pre-test | -0.019 (0.003) | -0.019 (0.003) | < 0.001*** | -0.014 (0.004) | -0.014 (0.003) | < 0.001*** |
| ***Solved (proportion)*** | | | | | | |
| Intercept | 0.866 (0.019) | 0.868 (0.021) | < 0.001*** | 0.840 (0.018) | 0.838 (0.018) | < 0.001*** |
| Experiment | 0.028 (0.015) | 0.029 (0.016) | 0.061 | 0.016 (0.016) | 0.015 (0.016) | 0.324 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Pre-test | 0.011 (0.003) | 0.011 (0.003) | 0.0005*** | 0.017 (0.003) | 0.017 (0.003) | < 0.001*** |
| ***Last Submission Score (0–1 scale)*** | | | | | | |
| Intercept | 0.913 (0.015) | 0.911 (0.015) | < 0.001*** | 0.894 (0.020) | 0.892 (0.022) | < 0.001*** |
| Experiment | 0.029 (0.014) | 0.029 (0.014) | 0.041* | 0.016 (0.019) | 0.015 (0.019) | 0.402 |
| Pre-test | 0.009 (0.003) | 0.009 (0.003) | 0.003** | 0.012 (0.004) | 0.012 (0.004) | 0.003** |
| ***Time Since First Submission (seconds)*** | | | | | | |
| Intercept | 459.217 (29.760) | 464.509 (45.946) | < 0.001*** | 497.061 (39.856) | 501.579 (44.586) | < 0.001*** |
| Experiment | -32.823 (26.028) | -35.152 (29.838) | 0.209 | 43.394 (34.758) | 39.889 (35.614) | 0.214 |
| Pre-test | -43.127 (5.602) | -42.788 (7.031) | < 0.001*** | -41.959 (7.440) | -42.536 (7.016) | < 0.001*** |
| ***Beyond Time Threshold (proportion)*** | | | | | | |
| Intercept | 0.886 (0.034) | 0.882 (0.028) | < 0.001*** | 0.852 (0.032) | 0.871 (0.034) | < 0.001*** |
| Experiment | -0.098 (0.031) | -0.101 (0.034) | 0.002** | 0.013 (0.028) | 0.006 (0.032) | 0.645 |
| Pre-test | -0.047 (0.007) | -0.047 (0.007) | < 0.001*** | -0.036 (0.006) | -0.042 (0.007) | < 0.001*** |
| ***Taking a Break (proportion)*** | | | | | | |
| Intercept | 0.207 (0.023) | 0.214 (0.035) | < 0.001*** | 0.190 (0.023) | 0.198 (0.037) | < 0.001*** |
| Experiment | -0.035 (0.018) | -0.036 (0.020) | 0.050* | -0.012 (0.020) | -0.012 (0.022) | 0.543 |
| Pre-test | -0.025 (0.004) | -0.025 (0.005) | < 0.001*** | -0.024 (0.004) | -0.024 (0.005) | < 0.001*** |
| ***Wheel Spinning (proportion)*** | | | | | | |
| Intercept | 0.086 (0.012) | 0.084 (0.015) | < 0.001*** | 0.088 (0.016) | 0.084 (0.021) | < 0.001*** |
| Experiment | -0.025 (0.012) | -0.028 (0.014) | 0.032* | 0.019 (0.015) | 0.020 (0.019) | 0.196 |
| Pre-test | -0.011 (0.003) | -0.010 (0.003) | < 0.001*** | -0.013 (0.003) | -0.012 (0.003) | < 0.001*** |

** Note: Coefficient (Std.E): The coefficient value and its standard error for the regression analysis. MC Mean (Std.D): The mean value and its standard deviation obtained from Monte Carlo simulations.*

### *4.2 RQ2: Impact on Challenging Tasks*

The results for the challenging tasks with LLM feedback enabled for the experimental group are presented in Table 2. The analysis revealed that the experimental group had significantly higher last submission scores ($\beta$ = 0.059, $p$ = 0.013, $\tau$ = 0.183) and a marginally higher likelihood of solving the challenging problems ($\beta$ = 0.097, $p$ = 0.059, $\tau$ = 0.148) compared to the control group. These findings suggest that the addition of LLM feedback may have helped learners achieve better final outcomes and solve rates in challenging programming tasks. However, no significant differences were found between the two groups in terms of the number of submissions, time spent on the task, going beyond the time or attempt threshold, or taking breaks. Overall, these results suggest that the impact of LLM feedback on learners' persistence can be observed in challenging tasks, particularly in terms of achieving better final outcomes and solving the problems.

Table 2. *Combined Regression Analysis for Challenging Tasks*

| **Variable** | **GPT Feedback Enabled (Phase 1)** | | | **GPT Feedback Disabled (Phase 2)** | | |
|---|---|---|---|---|---|---|
| | Coefficient (Std.E) | MC Mean (Std.D) | p-value | Coefficient (Std.E) | MC Mean (Std.D) | p-value |
| ***Number of submissions (count)*** | | | | | | |
| Intercept | 11.716 (0.887) | 11.689 (1.138) | < 0.001*** | 11.092 (1.144) | 11.092 (1.144) | < 0.001*** |
| Experiment | 0.681 (0.823) | 0.696 (0.879) | 0.409 | 0.925 (1.061) | 0.873 (1.190) | 0.385 |
| Pre-test | -1.043 (0.177) | -1.032 (0.177) | < 0.001*** | -0.802 (0.230) | -0.802 (0.230) | 0.001*** |
| ***Beyond Attempt Threshold (proportion)*** | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Intercept | 0.251 (0.032) | 0.247 (0.044) | < 0.001*** | 0.257 (0.038) | 0.244 (0.042) | < 0.001*** |
| Experiment | 0.030 (0.028) | 0.030 (0.030) | 0.277 | 0.019 (0.033) | 0.020 (0.035) | 0.570 |
| Pre-test | -0.030 (0.006) | -0.029 (0.007) | < 0.001*** | 0.023 (0.007) | -0.022 (0.007) | 0.002** |
| ***Solved (proportion)*** | | | | | | |
| Intercept | 0.638 (0.058) | 0.650 (0.054) | < 0.001*** | 0.615 (0.049) | 0.621 (0.047) | < 0.001*** |
| Experiment | 0.097 (0.051) | 0.097 (0.050) | 0.059 | 0.034 (0.043) | 0.030 (0.045) | 0.436 |
| Pre-test | 0.024 (0.011) | 0.023 (0.010) | 0.028* | 0.043 (0.009) | 0.042 (0.008) | < 0.001*** |
| ***Last Submission Score (0–1 scale)*** | | | | | | |
| Intercept | 0.824 (0.029) | 0.822 (0.032) | < 0.001*** | 0.842 (0.022) | 0.841 (0.030) | < 0.001*** |
| Experiment | 0.059 (0.023) | 0.062 (0.025) | 0.013* | 0.003 (0.020) | 0.002 (0.024) | 0.892 |
| Pre-test | 0.014 (0.005) | 0.014 (0.005) | 0.005** | 0.019 (0.004) | 0.018 (0.005) | < 0.001*** |
| ***Time Since First Submission (seconds)*** | | | | | | |
| Intercept | 964.237 (78.881) | 951.294 (101.410) | < 0.001*** | 1095.265 (103.008) | 1107.270 (130.428) | < 0.001*** |
| Experiment | 56.962 (77.068) | 47.408 (91.108) | 0.461 | 28.874 (92.823) | 22.494 (92.560) | 0.756 |
| Pre-test | -104.134 (16.567) | -99.793 (17.451) | < 0.001*** | -97.333 (20.135) | -95.936 (22.104) | < 0.001*** |
| ***Beyond Time Threshold (proportion)*** | | | | | | |
| Intercept | 1.034 (0.037) | 1.034 (0.029) | < 0.001*** | 1.035 (0.039) | 1.031 (0.027) | < 0.001*** |
| Experiment | -0.007 (0.036) | -0.009 (0.042) | 0.847 | -0.084 (0.038) | -0.084 (0.041) | 0.028* |
| Pre-test | -0.040 (0.008) | -0.040 (0.009) | < 0.001*** | -0.029 (0.008) | -0.030 (0.008) | < 0.001*** |
| ***Taking a Break (proportion)*** | | | | | | |
| Intercept | 0.382 (0.059) | 0.381 (0.060) | < 0.001*** | 0.371 (0.069) | 0.361 (0.075) | < 0.001*** |
| Experiment | 0.018 (0.055) | 0.020 (0.060) | 0.739 | -0.034 (0.063) | -0.029 (0.061) | 0.591 |
| Pre-test | -0.055 (0.012) | -0.053 (0.011) | < 0.001*** | -0.043 (0.014) | -0.042 (0.012) | 0.002** |
| ***Wheel Spinning (proportion)*** | | | | | | |
| Intercept | 0.175 (0.048) | 0.170 (0.050) | < 0.001*** | 0.209 (0.046) | 0.210 (0.059) | < 0.001*** |
| Experiment | -0.080 (0.045) | -0.078 (0.049) | 0.080 | -0.014 (0.045) | -0.013 (0.052) | 0.752 |
| Pre-test | -0.016 (0.010) | -0.014 (0.010) | 0.105 | -0.029 (0.010) | -0.028 (0.009) | 0.003** |

### *4.3 RQ3: Post-feedback Removal Impact*

The analysis revealed that almost all the benefits observed in the experimental group were no longer present when the LLM feedback was disabled. The only statistically significant difference we found is that, for challenging tasks, the experimental group exhibited a lower tendency to go beyond the time threshold ($\beta$ = -0.084, $p$ = 0.028, $\tau$ = -0.091).

## 5. Discussion

Our findings extend previous work on persistence in programming (Flores & Rodrigo, 2020; Pinto et al., 2021) by demonstrating the potential of LLM feedback to reduce wheel spinning. Two key themes emerge from the findings: **the promises of LLM feedback in supporting students**, and the **challenges of sustaining the effects of AI-powered feedback**.

Empirical findings from RQ1 and RQ2 suggest that the addition of LLM feedback can promote learners' persistence and reduce unproductive behaviors. Specifically, we have observed improvements in last submission scores, higher likelihood of solving problems, lower rates of going beyond the time threshold (after adjusting for the time required to generate LLM feedback), and reduced wheel spinning. These findings align with previous studies on intelligent tutoring systems, such as (Maniktala et al., 2020), who found that hints in the format

of unsolicited help can encourage productive persistence. Similarly, our results echo the work of (Marwan et al., 2020) in CS education, where personalized feedback led to greater engagement and increased intention to persist in CS. However, the benefits of LLM feedback for compiler errors appear limited in certain scenarios, particularly for challenging tasks. While the experimental group demonstrated significantly higher last submission scores and a marginally higher likelihood of solving challenging problems, LLM feedback did not significantly reduce the probability of wheel spinning on challenging tasks compared to the control group. This limitation in complex problem-solving scenarios could be attributed to several factors: *(1) Inability to diagnose underlying misconceptions*: In complex tasks, students' difficulties often stem from fundamental misconceptions that are not easily identifiable from code alone (Qian & Lehman, 2017). LLM models, operating solely on submitted code, may struggle to effectively diagnose and address these underlying issues. *(2) Overgeneralization of feedback*: A major issue identified with ChatGPT is its tendency to provide overly general responses (Rahman et al., 2023; Ray, 2023). For highly complex tasks, LLM feedback might lack the specificity needed to overcome challenges, failing to disrupt wheel-spinning. *(3) Cognitive overload*: Challenging tasks already impose a high cognitive load on students. Additional feedback, even if accurate, might exacerbate this cognitive burden, making it difficult for students to effectively process and apply the suggestions. These observations align with previous research (Qureshi, 2023), which found that ChatGPT is effective primarily with simpler data structure problems but struggles with complex queries. This often requires multiple prompts and can potentially increase students' cognitive load. These results highlight the need for a nuanced approach to implementing LLM feedback in CS education. While it shows promise in supporting learners, particularly with simpler tasks, educators and researchers must consider strategies to leverage LLMs in more challenging programming tasks.

When disabled, the previously observed benefits of LLM feedback no longer existed, suggesting that the effects may rely on continuous availability. This contrasts with findings from Pankiewicz and Baker (2023), who reported that students exposed to LLM feedback continued to solve tasks more quickly even after its removal, possibly due to a potential learning effect. In our study, however, students were enrolled in an introductory CS1 course, where many lacked prior programming experience, which may have limited their ability to internalize the feedback. Moreover, LLM feedback targeted only compiler errors and was available for just the first half of each module, a design intended to mitigate over-reliance. Yet this limited exposure, combined with the relatively low frequency of compiler errors (under 10% of submissions; Pankiewicz & Baker, 2024), may have reduced opportunities for meaningful learning transfer. These contrasting findings emphasize the need for further research into the sustainability of the benefits of LLM feedback.

Several implications emerge. First, AI-generated feedback has the potential to support programming tasks by helping students overcome initial barriers to understand compiler error messages and achieve more efficient debugging. However, careful implementation is necessary to avoid over-reliance on AI assistance. Second, future research should explore feedback systems that adapt to the learner's current performance, potentially through hybrid systems that combine LLM feedback with data-driven insights about student's current knowledge. Lastly, there is a need to design feedback systems that explicitly foster the development of transferable problem-solving skills, given that the benefits of LLM feedback were not sustained when removed. This could involve gradually reducing the level of support provided, encouraging them to develop independent problem-solving strategies.

Our study has several limitations. First, the effectiveness of LLM feedback may vary depending on the training, prompts and specific models. Our study used GPT-4, and results may differ with other versions or similar language models. Second, these tasks are optional problems and participants were free to engage as little or as much as they wanted with the problems. Third, existing methods for detecting productive persistence and wheel spinning do not account for quality of code revisions or debugging strategies, which could provide valuable insights and should be considered in future research. Lastly, our quantitative metrics can be complemented by qualitative data, such as interviews or surveys, to better understand how students perceive LLM-generated feedback and to contextualize the modest effect sizes.

## 6. Conclusion

This study provides empirical evidence on the impact of LLM feedback on programming tasks. LLM feedback for compiler errors has shown to enhance certain aspects of student

performance and persistence, especially in challenging tasks. However, its benefits appear to diminish once the feedback is removed in our study context. The transient nature of the benefits emphasizes the need for careful integration of AI tools in educational settings. It emphasizes the importance of developing strategies that not only leverage immediate LLM assistance but also foster students' independent problem-solving skills. Future research should focus on developing LLM feedback systems that can balance immediate learning support with the promotion of long-term skill development and learning. While LLM feedback presents exciting opportunities for CS education, the ultimate goal is to leverage the strengths of LLM feedback while cultivating transferable debugging and programming skills that persist beyond the immediate context of AI assistance.

## References


Adjei, S. A., Baker, R. S., & Bahel, V. (2021). Seven-year longitudinal implications of wheel spinning and productive persistence. *International Conference on Artificial Intelligence in Education*, 16–28.

Beck, J. E., & Gong, Y. (2013). Wheel-spinning: Students who fail to master a skill. *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 913, 2013. Proceedings 16*, 431–440.

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 500–506.

Cloninger, C. R., Svrakic, D. M., & Przybeck, T. R. (1993). A psychobiological model of temperament and character. *Archives of General Psychiatry*, *50*(12), 975–990.

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010). Manipulating mindset to positively influence introductory programming performance. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 431–435.

Duckworth, A. L., Peterson, C., Matthews, M. D., & Kelly, D. R. (2007). Grit: Perseverance and passion for long-term goals. *Journal of Personality and Social Psychology*, *92*(6), 1087.

Elinzano, G. B. O., & Ching, M. R. (2024). Determinants of ChatGPT Adoption in Academe & Other Fields – A Review on Theoretical Perspective. *International Conference on Computers in Education*. https://doi.org/10.58459/icce.2024.4969

Fang, Y., Nye, B., Pavlik, P., Xu, Y. J., Graesser, A., & Hu, X. (2017). Online Learning Persistence and Academic Achievement. *Internatonal Educational Data Mining Society*.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of openai codex on introductory programming. *Proceedings of the 24th Australasian Computing Education Conference*, 10–19.

Flores, R. M., & Rodrigo, M. M. T. (2020). Wheel-spinning models in a novice programming context. *Journal of Educational Computing Research*, *58*(6), 1101–1120.

Gong, Y., & Beck, J. E. (2015). Towards detecting wheel-spinning: Future failure in mastery learning. *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*, 67–74.

Hartmann, B., MacDougall, D., Brandt, J., & Klemmer, S. R. (2010). What would other programmers do: Suggesting solutions to error messages. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1019–1028.

Käser, T., Klingler, S., & Gross, M. (2016). When to stop? Towards universal instructional policies. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, 289–298.

Kazemitabaar, M., Ye, R., Wang, X., Henley, A. Z., Denny, P., Craig, M., & Grossman, T. (2024). Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. *Proceedings of the 2024 Chi Conference on Human Factors in Computing Systems*, 1–20.

Kiesler, N., Lohr, D., & Keuning, H. (2023). Exploring the potential of large language models to generate formative programming feedback. *2023 IEEE Frontiers in Education Conference (FIE)*, 1–5.

Kloke, J. D., & McKean, J. W. (2012). Rfit: Rank-based Estimation for Linear Models. *The R Journal*, *4*(2), 57–64. https://doi.org/10.32614/RJ-2012-014

Leinonen, J., Denny, P., MacNeil, S., Sarsa, S., Bernstein, S., Kim, J., Tran, A., & Hellas, A. (2023). Comparing code explanations created by students and large language models. *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 124–130.

Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. (2023). Using large language models to enhance programming error messages. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 563–569.

Liffiton, M., Sheese, B. E., Savelka, J., & Denny, P. (2023). Codehelp: Using large language models with guardrails for scalable support in programming classes. *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, 1–11.

Maniktala, M., Cody, C., Barnes, T., & Chi, M. (2020). Avoiding help avoidance: Using interface design changes to promote unsolicited hint usage in an intelligent tutor. *International Journal of Artificial Intelligence in Education*, *30*, 637–667.

Marwan, S., Gao, G., Fisk, S., Price, T. W., & Barnes, T. (2020). Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. *Proceedings of the 2020 ACM Conference on International Computing Education Research*, 194–203.

McBroom, J., Koprinska, I., & Yacef, K. (2021). A survey of automated programming hint generation: The hints framework. *ACM Computing Surveys (CSUR)*, *54*(8), 1–27.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67–92.

McClendon, C., Neugebauer, R. M., & King, A. (2017). Grit, growth mindset, and deliberate practice in online learning. *Journal of Instructional Research*, *8*, 8–17.

Nienaltowski, M.-H., Pedroni, M., & Meyer, B. (2008). Compiler error messages: What can help novices? *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 168–172.

Pankiewicz, M., & Baker, R. S. (2023). Large Language Models (GPT) for automating feedback on programming assignments. *Proceedings of the 31st International Conference on Computers in Education. Asia-Pacific Society for Computers in Education*.

Pankiewicz, M., & Baker, R. S. (2024). Navigating Compiler Errors with AI Assistance-A Study of GPT Hints in an Introductory Programming Course. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (pp. 94–100).

Pinto, J. D., Zhang, Y., Paquette, L., & Fan, A. X. (2021). Investigating elements of student persistence in an introductory computer science course. *5th Educational Data Mining in Computer Science Education (CSEDM) Workshop*.

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's weird that it knows what i want": Usability and interactions with copilot for novice programmers. *ACM Transactions on Computer-Human Interaction*, *31*(1), 1–31.

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 1–24.

Qureshi, B. (2023). Exploring the use of chatgpt as a tool for learning and assessment in undergraduate computer science curriculum: Opportunities and challenges. *arXiv Preprint arXiv:2304.11214*.

Rahman, M. M., Terano, H. J., Rahman, M. N., Salamzadeh, A., & Rahaman, M. S. (2023). ChatGPT and academic research: A review and recommendations based on practical examples. *Rahman, M., Terano, HJR, Rahman, N., Salamzadeh, A., Rahaman, S.(2023). ChatGPT and Academic Research: A Review and Recommendations Based on Practical Examples. Journal of Education, Management and Development Studies*, *3*(1), 1–12.

Ray, P. P. (2023). ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*.

Samson, B., Luriaga, R., & Ebardo, R. (2024). Do Academic Stress and Risk Propensity Affect Behavioral Intention to Use ChatGPT among University Students? *International Conference on Computers in Education*. https://doi.org/10.58459/icce.2024.4938

Sit, C. Y. E., Yang, Y., Yeung, W. K., & Kong, S. C. (2024). Developing a LLMs-Driven System Based on Human-AI Progressive Code Generation Framework to Assist Mathematics Learning. *International Conference on Computers in Education*. https://doi.org/10.58459/icce.2024.4815

Wang, Y., & Baker, R. (2018). Grit and intention: Why do learners complete MOOCs? *International Review of Research in Open and Distributed Learning*, *19*(3).

Xue, Y., Chen, H., Bai, G. R., Tairas, R., & Huang, Y. (2024). Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1. *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, 331–341.