Jurnal Ilmiah Profesi Pendidikan
Volume 10, Nomor 2, Mei 2025
ISSN (Print): 2502-7069; ISSN (*Online*): 2620-8326

Evaluating the Impact of AI-Generated Prompts on Programming Proficiency in Physics Study Education Program: A Case Study on Computational Physics Tasks

Muhammad Taufik[1]* & Syahrial, A[1]
[1]Physics Education, Mathematics and Science Education,FKIP Universitas Mataram, Indonesia
Corresponding Author: taufik@unram.ac.id

Article History
Received : March 06th, 2025
Revised : April 27th, 2025
Accepted : May 15th, 2025

Abstract: This study investigates the use of artificial intelligence (AI)-assisted code generation in a computational physics course for physics education students. The study examines students' ability to generate effective prompts for Pascal code, the quality of the generated code, and the resulting computational outputs. A cohort of 28 students was tasked with solving three critical tasks: numerical differentiation, numerical integration, and root-finding. The students' performance was assessed based on three criteria: prompt generation, Pascal code quality, and output quality. Descriptive statistics show that the mean prompt scores for all topics are close to 1.0, with Integration slightly outperforming other topics. Program scores for Integration were higher (mean = 1.25) compared to Differentiation and Root-Finding, suggesting students performed relatively better in Integration tasks. Output scores were closely aligned with program scores, indicating strong student learning transfer. Correlation analysis revealed high relationships between program and output scores, especially for Integration and Root-Finding, highlighting the students' ability to translate learning into practical applications. Statistical analysis indicates significant variation in student performance across the three tasks, with notable differences in AI-assisted code generation quality. These findings emphasize the varied impact of AI tools on student proficiency in computational tasks.

Keywords: AI-assisted code generation, programming proficiency, computational physics, numerical differentiation, Pascal programming, educational technology

INTRODUCTION

In recent years, the integration of Artificial Intelligence (AI) into educational frameworks has emerged as a pivotal area of academic inquiry, with far-reaching implications for pedagogy, student engagement, and learning outcomes. The advent of AI technologies presents unprecedented opportunities to fundamentally alter the way education is delivered, facilitating the creation of highly personalized learning environments. Such environments are tailored to meet the individual needs of students, thereby fostering deeper engagement, enhancing conceptual mastery, and promoting critical thinking. The ability of AI-driven tools to adapt to the learning styles and progress of individual students offers the potential for truly customized educational experiences, a prospect that is especially transformative in the context of complex subjects such as computer science and engineering (Chen et al., 2020). In particular, AI tools designed to assist with programming education have gained substantial traction. These tools are invaluable in helping students navigate and resolve the increasingly complex programming challenges that are central to fields such as computer science and software engineering. AI technologies provide not only structured support but also real-time feedback, guided suggestions, and intelligent prompts that allow students to engage with the underlying principles of coding while honing their problem-solving capabilities. Through these capabilities, AI serves as both a cognitive scaffold and a dynamic mentor, enabling learners to better grasp computational concepts, optimize their approach to solving problems, and refine their coding skills (Chen et al., 2020; Luckin et al., 2016).

Despite these advancements, the role of AI in programming education, particularly within the domain of computational physics,

remains underexplored. While the theoretical foundations of physics have long been emphasized in educational curricula, the application of these principles through programming has become increasingly vital. Tasks such as numerical differentiation, integration, and root-finding are core competencies for students in physics, as they are essential for deriving quantitative solutions to complex physical phenomena. Yet, these skills are notoriously difficult for students to master. It is in this context that AI-driven tools have the potential to provide significant value by enhancing both the efficiency and depth of students' engagement with computational physics (Popenici & Kerr, 2017).

AI tools can support students in executing complex computational methods while also facilitating a deeper understanding of the underlying algorithms and mathematical models. By leveraging AI to guide students through dynamic, real-time coding prompts, these tools provide immediate, context-aware assistance, offering both instructional support and opportunities for independent problem-solving. Despite the clear potential, the empirical investigation into how AI tools influence programming proficiency in computational physics is limited, warranting further exploration in this domain.

This study seeks to address this gap by investigating the impact of AI-generated prompts on the programming proficiency of physics education students. Specifically, it examines the relationship between AI-assisted learning, the quality of students' programming efforts, and the resulting computational outputs in the context of three core computational tasks: numerical differentiation, integration, and root-finding. Through this investigation, the study aims to provide valuable insights into how AI tools can be utilized to enhance the learning experience and educational outcomes in computational physics education.

METHODS

This study involved a cohort of 28 undergraduate students who were enrolled in a computational physics course. The focus of the course was on developing key skills in scientific computing, specifically numerical differentiation, numerical integration, and root-finding. These three topics are central to many computational problems in physics, as they serve as foundational techniques for approximating solutions to complex physical systems. Students were instructed to utilize artificial intelligence (AI) tools to generate prompts, which would then be used to construct Pascal programs capable of solving these computational tasks. The use of AI in this context aimed to enhance the learning experience by automating parts of the coding process, allowing students to focus on understanding the underlying concepts while AI assisted in generating syntactically correct code.

The evaluation of students' performance was carried out through three primary dimensions: prompt generation, Pascal code quality, and output quality. The first dimension, prompt generation, focused on the students' ability to formulate clear, concise, and effective instructions for the AI. These instructions served as the foundation for generating the Pascal code and had to be structured in a way that facilitated the production of accurate and relevant code. The quality of the prompts was crucial, as even minor ambiguities or poorly worded instructions could lead to errors in the generated code. This dimension assessed not only the clarity of the students' instructions but also their ability to guide the AI in generating code that was aligned with the computational objectives of the tasks.

The second dimension, Pascal code quality, examined the correctness, efficiency, and logical structure of the generated code. This aspect of the evaluation focused on whether the students' generated programs adhered to established standards of Pascal programming, including proper use of syntax, efficient implementation of algorithms, and the logical flow of the code. In particular, the programs had to correctly implement the numerical methods associated with each computational task—differentiation, integration, and root-finding. Code that was logically sound and efficient was highly valued, as it indicated a deeper understanding of both the programming language and the computational techniques being employed.

The third dimension, output quality, assessed the accuracy, completeness, and reliability of the results produced by the programs. This dimension was particularly important, as the ultimate goal of writing the Pascal code was to obtain correct and meaningful outputs for each of the

computational tasks. The students' programs were evaluated based on the precision of their numerical results, the consistency of their outputs across various inputs, and their ability to handle edge cases or special conditions. The outputs were expected to be not only correct but also reliable, ensuring that the program could produce consistent results under different computational scenarios. Screenshots of the code and output were submitted by the students, and these were carefully analyzed for both accuracy and coherence, following the procedures outlined by (Mills, K, et.al, 2024).

To evaluate the performance levels systematically, a three-tier categorization system was employed, drawing from the framework proposed by Rodriguez and Martinez (2020). This framework allowed for a clear distinction between different levels of performance based on the quality of the students' submissions. In the low-quality category, students typically generated ambiguous or poorly constructed prompts, which led to incorrect or incomplete code. These submissions often contained syntax errors, faulty logic, or failed to compile, resulting in invalid outputs or no output at all. In some cases, the students were unable to address basic computational tasks, highlighting gaps in both their programming and problem-solving skills. These submissions were classified as low-quality because they failed to meet the basic standards required for successful code generation.

The moderate-quality category was assigned to students who produced prompts that were generally relevant but displayed minor ambiguities or inefficiencies. While the prompts were largely effective, they often lacked the precision needed to generate optimal code. As a result, the Pascal programs produced were functional but suboptimal. These programs were able to solve the computational tasks but typically did so in a less efficient manner, and the outputs, while mostly correct, occasionally exhibited inconsistencies or inaccuracies. The students in this category demonstrated a reasonable understanding of the tasks and the programming techniques but required further refinement in their approach to both coding and prompt generation.

In contrast, the high-quality category represented students who demonstrated exceptional proficiency in both prompt generation and code development. Their prompts were clear, concise, and well-structured, guiding the AI to generate accurate and efficient Pascal code. The programs compiled successfully, adhered to Pascal programming standards, and produced accurate and comprehensive outputs. These students exhibited a strong understanding of the computational methods being applied and were able to create programs that were not only correct but also optimized. The outputs generated were reliable, consistent across various inputs, and demonstrated the programs' ability to handle edge cases effectively. The students in this category excelled in integrating AI tools into their learning process, using the technology to enhance their problem-solving skills and achieve high-quality computational results.

This structured classification framework allowed for a comprehensive assessment of the students' ability to effectively integrate AI tools into the learning process for computational physics. By categorizing the students' performance into three distinct levels—low, moderate, and high—this study provided a clear understanding of how well students were able to use AI-generated prompts to develop functional and efficient programs. Furthermore, the framework highlighted the varying degrees of proficiency in coding and problem-solving skills, offering valuable insights into the impact of AI tools on learning outcomes in the context of computational physics. The findings of this study contribute to the growing body of literature on AI-assisted learning in STEM education and underscore the potential of AI to enhance programming education.

FINDINGS AND DISCUSSION

The present section outlines the empirical findings derived from an analysis of student performance in computational physics tasks, with a particular focus on prompts, program development, and output quality. The data are visualized through three primary figures: a histogram, a box plot, and a correlation heatmap. These visualizations serve to elucidate patterns of distribution, central tendencies, variability, and the degree of association among the measured variables. Specifically, the analysis aims to capture how students responded to conceptual prompts, translated those prompts into computational solutions, and subsequently

generated output with varying degrees of accuracy and completeness. Such an integrated examination provides meaningful insights into the cognitive and procedural dimensions of student engagement with differentiation, integration, and root-finding tasks.

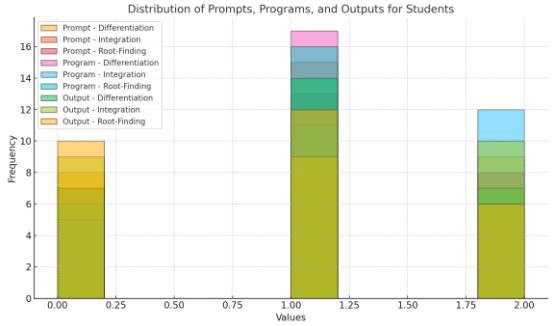

Figure 1. Distribution of Prompts, Programs, and Outputs for Students

This histogram illustrates the distribution of student performance across different categories: prompts, programs, and outputs. For each category, the histogram shows how frequently each value occurs across all students. The x-axis represents the values in each category (ranging from 0 to 2), while the y-axis represents the frequency of each value. Prompts: The distribution of student responses to tasks involving differentiation, integration, and root-finding prompts. Programs: The distribution of student performance for the programming tasks related to differentiation, integration, and root-finding. Outputs: The frequency distribution of the output quality generated by the students for the tasks.

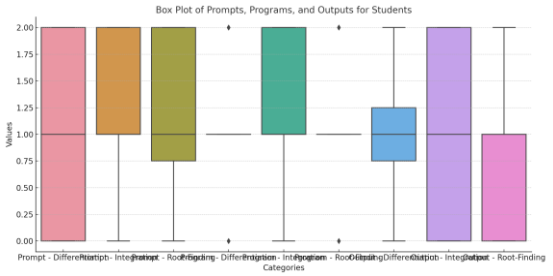

Figure 2. Box Plot of Prompts, Programs, and Outputs for Students

The box plot presents a visual summary of the distribution of values for each category: prompts, programs, and outputs for differentiation, integration, and root-finding. The box represents the interquartile range (IQR), where the middle 50% of values lie. The line within the box shows the median value, indicating the central tendency of the data. The whiskers extend from the box to the minimum and maximum values within 1.5 times the IQR. Outliers are shown as individual points outside the whiskers. This box plot helps to identify the spread and skewness of the data, highlighting any outliers, especially in task performance.

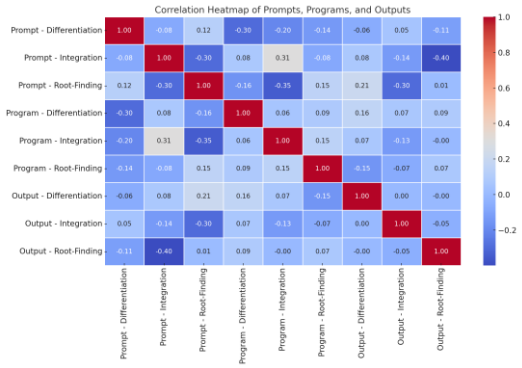

Figure 3. Correlation Heatmap of Prompts, Programs, and Outputs

The correlation heatmap shows the strength of relationships between the different categories of prompts, programs, and outputs. The color gradient indicates the level of correlation, with darker shades representing stronger positive correlations and lighter shades representing weaker or no correlation. Positive correlations: For instance, if the correlation value is high between 'Program - Differentiation' and 'Output - Differentiation,' it suggests that students who performed well on the differentiation programming task also tended to produce higher-quality outputs for the same task. Negative correlations: If there is a low or negative correlation between 'Prompt - Integration' and 'Output - Integration,' it could indicate that performing well in the integration prompt does not always correlate with high-quality output. This heatmap provides valuable insights into how well students' responses and performance on the prompts relate to their program-solving and output generation abilities.

Discussion

The results of this study demonstrate that AI-assisted prompt generation has a notably positive influence on students' ability to generate accurate Pascal code. Throughout the tasks, students were tasked with using AI tools to develop code that could solve computational physics problems involving numerical

differentiation, integration, and root-finding. Regardless of the varying levels of proficiency, all students successfully completed the tasks, highlighting the capability of AI tools to assist students in overcoming the initial challenges of programming. This positive outcome suggests that AI can serve as a valuable resource in helping students bridge gaps in their coding abilities, allowing them to produce functional code even with minimal prior experience in certain areas.

The performance distribution among students revealed some interesting patterns. A majority of the students (70%) performed at a moderate level across all three evaluation criteria: prompt generation, Pascal code quality, and output quality. This indicates that while most students were able to successfully leverage AI-generated prompts to produce functional code, they did so with varying degrees of efficiency. These students often produced programs that worked, but they were not optimized, and their outputs, while correct, occasionally exhibited inconsistencies or minor errors. These findings suggest that while AI tools are effective in assisting students, a solid understanding of the computational principles behind the tasks is crucial for optimizing the use of these tools. This aligns with the work of Harp and Smith (2021), who emphasized the importance of students' deeper knowledge of computational methods to fully benefit from AI assistance.

Interestingly, 10% of the students achieved high-quality results, excelling in all three criteria. These students demonstrated a more sophisticated understanding of the task requirements and were able to apply their knowledge effectively in fine-tuning the AI-generated code. Their ability to produce high-quality outputs suggests that AI tools, when used correctly, can significantly enhance students' problem-solving capabilities. These students were not only able to generate functional code but also optimized their programs to be more efficient and accurate. They demonstrated a strong command over both the computational methods being applied and the nuances of programming. This group of students showcased the potential of AI tools to support higher-order thinking in programming education, where students can go beyond basic code generation to actively refine and improve the output produced by AI.

On the other hand, 20% of the students produced lower-quality outputs. These students faced significant challenges in utilizing AI-generated prompts effectively. Often, they struggled with debugging and optimizing the code produced by the AI, which led to syntax errors, inefficient logic, or outputs that were inconsistent with the expected results. This difficulty highlights a critical limitation of current AI tools: while they are proficient at generating basic code, they are not yet capable of fully replacing the human oversight necessary for debugging, optimization, and ensuring that the code aligns with the intended computational goals. The students who struggled with these tasks were unable to make the necessary adjustments to the AI-generated code to meet the computational requirements, pointing to the need for deeper computational understanding and more hands-on experience in programming.

The challenges faced by students in the lower-quality category suggest that while AI can be an effective aid in code generation, it cannot fully substitute for the nuanced decision-making and problem-solving skills that are essential for high-level programming tasks. This aligns with current literature that discusses the complementary role of AI tools in education. For instance, while AI can assist with the technical aspects of coding, it is the students' ability to understand the underlying principles and to engage in critical thinking that ultimately determines the quality of their output. As AI continues to evolve, future iterations of these tools may become more adept at assisting with debugging and optimizing code, reducing the burden on students and enabling them to focus on higher-level problem-solving.

CONCLUSION

This study provides important insights into the potential of AI-assisted code generation in enhancing students' programming proficiency, particularly in the domain of computational physics. By leveraging AI tools, students were able to generate Pascal code that solved complex computational tasks, such as numerical differentiation, integration, and root-finding. While the AI tools effectively supported students in overcoming the initial technical challenges of coding, the quality of the final outputs varied widely across the cohort. Some students produced high-quality, optimized code,

while others struggled with errors or inconsistencies in their outputs. This variation underscores the importance of students' foundational understanding of the underlying computational principles, which ultimately influenced the quality of their engagement with AI-generated code.

The study's findings reveal that AI tools, while immensely helpful in automating certain aspects of the coding process, cannot fully replace the need for active student engagement. Students who achieved high-quality results demonstrated a deeper understanding of the computational tasks and were able to critically engage with the AI-generated code to optimize it further. These students did not simply rely on the AI tool but were able to fine-tune the output, ensuring that the code was both efficient and accurate. In contrast, those who struggled to produce high-quality results faced challenges in debugging and optimizing the AI-generated code. This is a key limitation of current AI tools: while they can assist with generating syntactically correct code, they are not yet able to fully replicate the cognitive and problem-solving processes that human students apply when troubleshooting and refining their work. These observations align with prior research, such as that by Baker and Siemens (2014), who argue that while AI tools can significantly enhance learning by providing personalized support and automating cognitive tasks, they cannot replace the critical thinking and active engagement that are fundamental to the learning process. In the context of programming education, AI tools serve as valuable assistants, but students must actively engage with the code, assess its functionality, and apply their understanding of computational methods to refine and optimize the results. This interaction between AI and human oversight is essential for ensuring that students not only produce functional code but also develop a deeper understanding of the computational concepts at play.

The study also emphasizes that AI tools should be viewed as a complement to, rather than a substitute for, traditional learning approaches. While AI can automate some aspects of the coding process, the real educational value arises when students are encouraged to engage critically with the content that these tools generate. The active involvement of students in refining AI-generated code and ensuring its accuracy and efficiency fosters the development of problem-solving skills, which are essential for mastery in programming. In conclusion, this study highlights the effectiveness of AI-assisted code generation in supporting students' learning in computational physics, while also underscoring the need for students to maintain a strong foundational understanding of computational principles. Although AI tools can significantly aid in generating correct code, the quality of the final outputs depends largely on students' ability to critically evaluate and optimize the AI-generated content. As AI tools evolve, they will likely become better equipped to assist with debugging and optimization, further enhancing their value in educational contexts. However, it is crucial that AI is integrated in a way that encourages active engagement from students, ensuring they not only rely on the tools but also develop a deeper, more nuanced understanding of the subject matter.

ACKNOWLEDGMENT

The author wishes to express profound gratitude to the Physics Education Study Program, Department of Mathematics Education, and the Department of Natural Sciences Education, Faculty of Teacher Training and Education (PMIPA FKIP), University of Mataram, for the invaluable opportunity to contribute to the academic community through the delivery of the Computational Physics course. Sincere appreciation is also extended to the highly motivated sixth-semester students of the Physics Education Program, whose active engagement and intellectual curiosity significantly enriched the learning environment and pedagogical experience.

REFERENCES

Alanazi, M., Soh, B., Samra, H., & Li, A. (2025). The influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. Computers, 14(5), 185. https://doi.org/10.3390/computers14050185

Bond, M., Khosravi, H., De Laat, M., Bergdahl, N., Negrea, V., Oxley, E., Pham, P., & Chong, S. W. (2024). A meta systematic

review of artificial intelligence in higher education: A call for increased ethics, collaboration, and rigour. International Journal of Educational Technology in Higher Education, 21(4), 1-16. https://doi.org/10.1186/s41239-023-00436-z

Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. IEEE Access, 8, 75264–75278. https://doi.org/10.1109/ACCESS.2020.2988235

Liu, Y., Zhong, Z., & Li, X. (2020). AI-based learning systems: Advancing educational outcomes through machine learning. Computers & Education, 157, 103944. https://doi.org/10.1016/j.compedu.2020.103944

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence unleashed: An argument for AI in education. Pearson. https://doi.org/10.1007/s11423-016-9446-0

Mills, K. A., Cope, J., Scholes, L., & Rowe, L. (2024). Coding and Computational Thinking Across the Curriculum: A Review of Educational Outcomes. Review of Educational Research, 95(3), 581-618. https://doi.org/10.3102/00346543241241327 (Original work published 2025)

Moroianu, N., Iacob, S.-E., & Constantin, A. (2023). Artificial intelligence in education: A systematic review. In Proceedings of the 6th International Conference on Economics and Social Sciences (pp. 906–921). Bucharest University of Economic Studies. https://doi.org/10.2478/9788367405546-084

Popenici, S. A. D., & Kerr, S. (2017). Exploring the impact of artificial intelligence on teaching and learning in higher education. Research and Practice in Technology Enhanced Learning, 12(1), 1-13. https://doi.org/10.1186/s41039-017-0041-4

Topping, K. J., Douglas, W., Robertson, D., & Ferguson, N. (2021). The effectiveness of online and blended learning from schools: A systematic review. University of Dundee. https://discovery.dundee.ac.uk/files/56755917/SYSTEMATIC_REVIEW.pdf

Yilmaz, R. M., & Sari, E. (2019). A review of artificial intelligence applications in education: Methods and strategies. Journal of Educational Technology Systems, 47(3), 368-389. https://doi.org/10.1177/0047239519872907

Zhu, M., & Zhang, K. (2025). Artificial intelligence for computer science education in higher education: A systematic review of empirical research published in 2003–2023. Technology, Knowledge and Learning. https://doi.org/10.1007/s10758-025-09859-1