Computer Applications in Engineering Education

WILEY

RESEARCH ARTICLE

Physics-Informed Pedagogy in AI Engineering: A Case Study on Teaching Fundamental Algorithms via Signal Processing

Tetiana Ivanivna Nosenko

Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine

Correspondence: Tetiana Ivanivna Nosenko (t.nosenko@kubg.edu.ua)

Received: 30 December 2025 | **Revised:** 27 April 2026 | **Accepted:** 6 May 2026

Keywords: algorithmic thinking | CS1 education | engineering education | instructional scaffolding | physics-informed machine learning (PIML)

ABSTRACT

The rapid integration of artificial intelligence tools into engineering education creates the risk of a "black-box" approach, where students utilize high-level ML libraries without a deep understanding of fundamental algorithms. This study proposes and empirically validates the physics-informed scaffolding methodology, which leverages stochastic physical simulations as structured pedagogical support for teaching basic algorithms in introductory programming courses. A case study ($N = 34$) focused on localizing radiation anomalies in noisy data was conducted. Students implemented fundamental signal processing algorithms (smoothing and thresholding) from scratch, replicating the micro-logic of mathematical convolution. This task was framed as a fundamental signal processing problem, providing a bridge between basic algorithmic structures and AI engineering applications. The effectiveness of the approach was measured using a validated instrument assessing three constructs: Engagement, Conceptual Understanding, and Scaffolding and Process Support (SPS). Quantitative analysis confirmed the high reliability of the instrument (Cronbach's $\alpha = 0.924$). An ordinary least squares regression model demonstrated that the proposed pedagogical support and student engagement are statistically significant predictors, explaining over 70% of the variance ($R^2 = 0.718$, $p < 0.001$) in students' perceived conceptual understanding of the algorithms. The research empirically demonstrates the efficacy of transitioning from a traditional "Syntax-First" to a "Data-First" educational model. Engaging with real-world physical noise helps students overcome "stochastic shock" and fosters the engineering intuition necessary for effective problem-solving in AI engineering.

1 | Introduction

Modern AI engineering requires more than just knowledge of syntax from a specialist; it demands deep data intuition. However, traditional CS1 (Introductory Computer Science) curricula often focus on sorting perfectly ordered arrays, leaving students unprepared for the stochastic nature of data derived from real sensors. This paper proposes a methodology to bridge this gap by teaching algorithms through the "AI-First" framew ork presented in [1].

1.1 | Relevance of the Study

The rapid evolution of Artificial Intelligence (AI) has transformed the requirements for basic engineering training. Modern deep learning algorithms require the developer to possess not only programming language syntax knowledge but also a deep understanding of the nature of data—its stochasticity, noisiness, and physical context. Traditional Computer Science teaching methodology in the first year (CS1) often ignores this aspect, focusing on processing idealized data. This creates a "cognitive

gap": students can write code but are unable to apply it to solve real engineering problems where data is unstructured or distorted by physical noise. The proposed approach integrates stochastic physical processes (Poisson noise and Gaussian signal) as a core pedagogical mechanism in CS1 to support the development of algorithmic thinking.

Given the growing demand for specialists in AI Engineering and Physics-Informed Machine Learning, there is an urgent need for new pedagogical approaches. Recent studies highlight the transformative potential of integrating artificial intelligence and Python-based tools in higher education to enhance student engagement and academic performance [2, 3]. Furthermore, the application of specialized Python libraries for structural analysis demonstrates how computational tools can automate complex engineering tasks and bridge the gap between theoretical concepts and practical application [4]. The relevance of this study lies in the necessity to implement a methodology that, from the very first learning steps, connects fundamental algorithmic structures (loops, arrays) with real signal processing tasks, fostering engineering intuition in students even before they begin studying complex neural network architectures.

1.2 | Research Aim and Objectives

The aim of this work is the theoretical substantiation and experimental verification of the effectiveness of the integrated "AI-First" approach in teaching the fundamentals of algorithmization. We aim to demonstrate that teaching fundamental algorithms through the lens of an applied physics problem (detecting a weak signal in noise) facilitates a deeper student understanding of the operating principles of modern AI systems.

To achieve this goal, the following objectives were set:

- To analyze existing methodological gaps in teaching algorithms to future AI engineers.
- To adapt the methodology described in the textbook *"Application of algorithms and data structures in artificial intelligence: A textbook"* [1] for creating an educational case study on radiometric data processing.
- To develop and implement an algorithmic anomaly detection model (radiation signals) using only basic Python tools without high-level ML libraries.
- To assess the impact of the proposed approach on forming students' understanding of convolution and data filtration concepts.

2 | Literature Review

2.1 | The Gap Between Computer Science Fundamentals and AI Engineering

For decades, the gold standard for teaching algorithms has remained the approach systematized in the classic work of [5]. The primary focus of this methodology is on deterministic algorithms, complexity estimation (Big O notation), and working with idealized data structures. However, the modern Deep Learning paradigm, described in the fundamental work by [6], imposes fundamentally new requirements on engineers: the ability to work with stochastic data, noise, and uncertainty.

Despite this technological shift, academic education is transforming slowly. A replication study by [7] shows that the content of introductory courses (CS1) has remained almost unchanged for the last 20 years. Although the language of instruction has shifted to Python, the methodological emphasis is still placed on syntax rather than data engineering. This contradicts the recommendations of the newest *Computer Science Curricula 2023* standard [8], which calls for integrating Artificial Intelligence elements at the early stages of undergraduate studies.

This situation creates what [9] refer to as a "vast chasm" between Computer Science pedagogy and Machine Learning requirements. As noted by [10], students graduate with coding skills but without an understanding of how this code interacts with real physical data, which is critical for developing reliable AI systems. In this context [11], emphasize that introducing AI elements specifically in the first years is critical for overcoming perception barriers in students with diverse backgrounds.

2.2 | The Physics-Informed Machine Learning (PIML) in Education

In response to the limitations of purely "data-centric" models, the field of PIML has emerged in science [12]., pioneers of this direction, demonstrated that integrating physical laws into the neural network training process allows compensating for data scarcity and improving model generalization. This approach is particularly relevant for radiation monitoring tasks. As noted by [13] in the seminal work on radiation detection, radiometric data always obey Poisson statistics and contain significant background noise. Purely algorithmic methods without considering this physical nature often yield erroneous results.

Recent reviews by [14] and [15] demonstrate the explosive growth of PIML in research. However, as noted by [16], PIML principles are still viewed as a topic for graduate students and are rarely used as a pedagogical tool for teaching beginners.

2.3 | Integration of Signal Processing Into CS1 Curriculum

The intersection of educational challenges and PIML capabilities opens space for innovation. Research by [17] proved that active learning methods increase the performance of engineering students by 55%. The fundamental basis for this lies in the principles of meaningful learning formulated by [18], who state that material is best mastered through solving complex, inquiry-based tasks. Building on this [19], propose shifting the learning focus from abstract programming to solving applied problems [20]., in their review, emphasize that contextual anchoring (e.g., to physical signals) is key to engaging engineering students.

Using Python as a tool for this is justified [21]: points out that Python allows students to focus on algorithm logic (e.g., convolution) rather than memory management, as in C++. The

effectiveness of such an approach is explained by Cognitive Load Theory, formulated by [22]. According to it, novices cannot learn effectively from complex tasks without prior structuring.

From Vygotsky's perspective, the proposed guidebook acts as a crucial mediation tool within the Zone of Proximal Development [23]. Students initially possess the capability to write basic loops (actual development level) but lack the intuition for signal processing (potential development level). The physics-informed simulation serves as the 'scaffold' that bridges this gap. By visualizing the invisible logic of convolution through manual array manipulation, we concretize the cognitive process, allowing learners to internalize the concept of 'filtering' before moving to abstract library calls. This approach aligns with Vygotsky's view that higher mental functions appear first on the social/external plane (the guidebook tasks) and then on the individual/internal plane (algorithmic intuition).

Furthermore, the development of "Computational Thinking," as defined by [24], requires specifically abstracting from concrete syntax to the algorithmic essence, which our approach ensures.

The practical implementation of this strategy is presented in the methodological complex [1]: we use the task of searching for radioactive anomalies as "scaffolding" for learning fundamental algorithmic structures.

3 | Methodology

3.1 | Research Design and Pedagogical Framework

The study involved students with basic knowledge of Python but a limited background in physics. The learning process was structured around key chapters of the textbook. Following Chapter 3 ("Arrays and Tensors: The Language of Neural Networks"), students modeled the data stream from the detector not as an abstract list, but as a continuous block of memory representing physical time steps.

To reduce noise, students implemented a moving average filter. This required the application of basic logical structures from Chapter 1 ("Basic Concepts of Algorithm Theory and Data Structures. AI Foundation: From Idea to Python Code"). By manually creating nested loops, students gained a mechanistic understanding of how a "sliding window" operates—a concept usually hidden within high-level functions (such as np.convolve).

The detection of the radioactive source was formulated as a search problem described in Chapter 6 ("Search Algorithms: From Array Search to Pathfinding"), where the target was not a specific value, but a condition (exceeding a threshold signal-to-noise ratio).

The conceptual structure of the proposed educational intervention, illustrating the transition from basic syntax to AI-ready thinking, is shown in Figure 1

To ensure a systematic approach to learning, the case study architecture was developed in alignment with the guidebook's structure. The relationship between the experiment stages and fundamental algorithmic concepts is presented in Table 1.

3.2 | Task Formulation as a Computational Experiment

Students were tasked with detecting a weak Gaussian signal, simulating a radioactive source, embedded within Poisson noise.

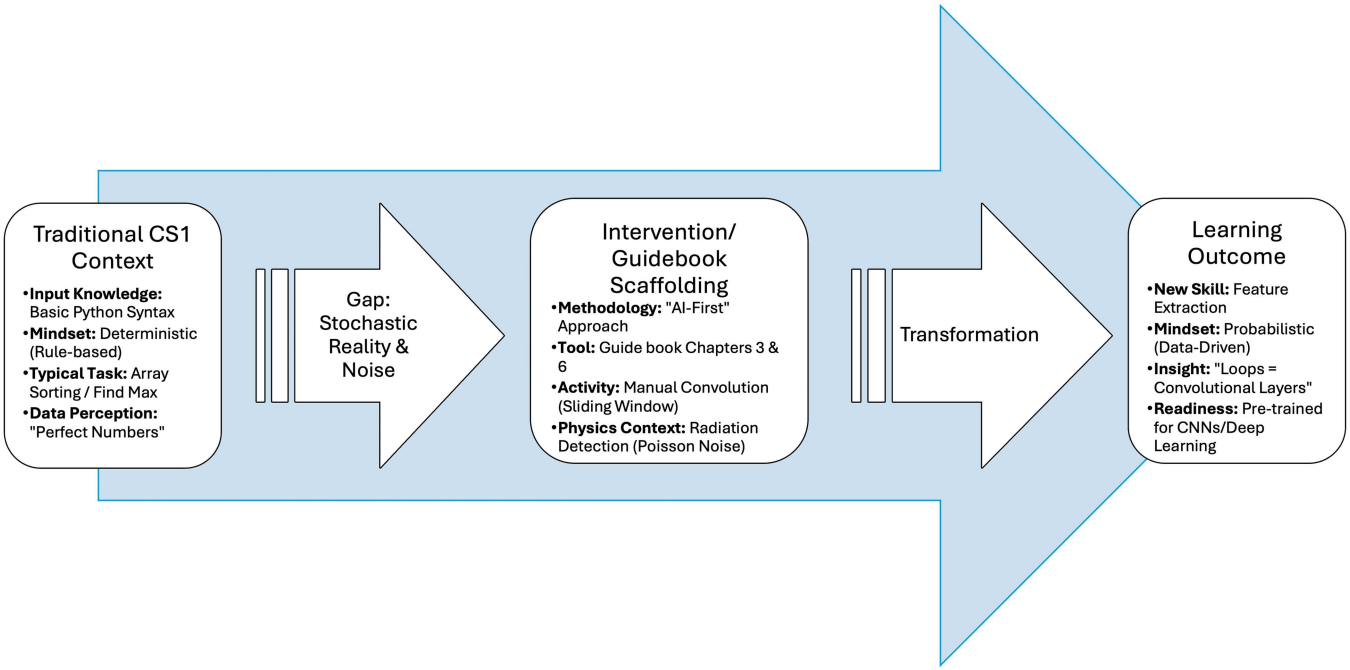

FIGURE 1 | The "physics-informed scaffolding" Framework. The diagram illustrates the pedagogical transition from traditional syntax learning (left) to AI-ready engineering thinking (right). The intervention (center) employs the textbook methodology to bridge the gap by applying fundamental algorithms to physical signal processing tasks.

TABLE 1 | Mapping of the guidebook's fundamental algorithmic concepts to the signal processing experiment stages.

Case study stage	Chapter/Topic	Pedagogical message/Scientific rationale
1. Signal Storage	Chapter 3. Arrays and Tensors – The Language of Neural Networks	Forming an understanding of the array as a discretized physical signal in contiguous memory, as opposed to abstract high-level lists.
2. Filter Implementation	Chapter 1. Basic Concepts of Algorithm Theory and Data Structures. AI Foundation: From Idea to Python Code	Developing algorithmic thinking through the manual implementation of the convolution operation using nested iterations, ensuring a deep understanding of noise filtering mechanics.
3. Anomaly Detection	Chapter 6. Search Algorithms: From Array Search to Pathfinding	Adaptation of classic linear search algorithms for pattern detection tasks, where the success criterion is not an exact value match, but meeting the condition of exceeding the Signal-to-Noise Ratio (SNR).
4. Understanding Efficiency	Chapter 1. Topic 1.5. Estimating Time and Space Complexity: How to Know if Your Algorithm Can Handle Gigabytes of Data	Justification for choosing algorithms with linear complexity O(n) for processing streaming data in real-time systems; analysis of computational costs.
5. Bridge to AI Training	Chapter 7. Algorithm Design Strategies and a Glimpse into the Future	Introduction to hyperparameter optimization principles: iterative tuning of the sensitivity threshold as a propaedeutic step* toward understanding gradient descent and model training.

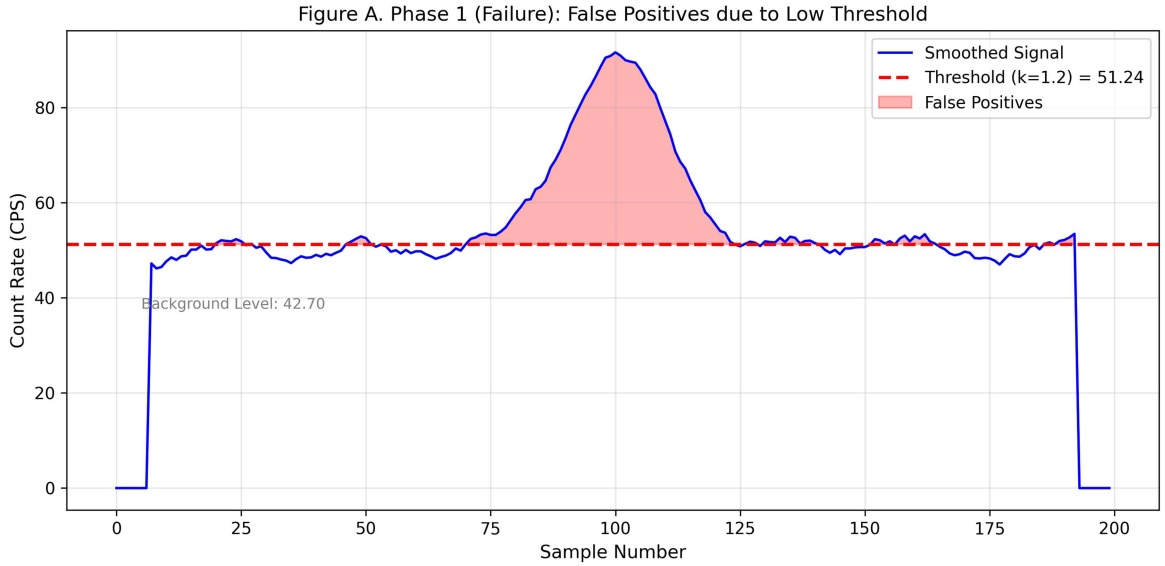

FIGURE 2 | Demonstration of the false positives problem during Phase 1. The red dashed line indicates the initial low threshold ($k = 1.2$), which proved insufficient for filtering stochastic Poisson noise. The red zone illustrates the data range erroneously classified by the algorithm as an anomaly, preventing source localization.

- Input Data: A one-dimensional array of 200 data points representing counts per second (CPS).
- The Challenge: The signal peak was commensurate with noise outliers, making simple thresholding ineffective.

Students implemented signal processing algorithms (smoothing and threshold detection) using core Python without high-level ML libraries.

The pedagogical design incorporated scaffolding (gradual complexity), manual implementation, physical interpretation of variables and iterative parameter tuning.

3.3 | Experimental Procedure

The experiment consisted of two phases.

3.3.1 | Phase 1. Initial Implementation and False Positive Analysis

Students applied a threshold coefficient $k = 1.2$. The algorithm incorrectly classified a large portion of the signal (indices 31–192) as anomalies due to stochastic noise fluctuations. As seen in Figure 2, the red threshold line is set too low, and the red zone covers almost the entire plot, masking the true signal.

This phase demonstrated that formally correct code does not guarantee correct physical interpretation.

3.3.2 | *Phase 2: Iterative Improvement*

Guided by the methodology of the guidebook [1], students applied the method of iterative parameter tuning. The threshold coefficient was increased to $k = 1.6$. The optimized algorithm calculated a background level of 42.70 CPS and set a detection threshold of 68.32 CPS. This adjustment successfully filtered out noise artifacts and isolated the signal within the index range [91, 109], which corresponds precisely to the source center (around the 100th point). As shown in Figure 3, the green dashed threshold line runs above the noise, and the green zone highlights exclusively the central peak.

This phase illustrates the importance of parameter tuning.

Results evaluation was conducted via Code Review and oral defense, during which students were required to explain the physical meaning of every variable in their algorithm (e.g., what the window_sum variable represents in the context of signal processing).

Although students implemented convolution algorithms "from scratch," the standard Python scientific stack was used for visualization and basic array operations. Specifically, the NumPy library [25] served as the standard for tensor representation, and Matplotlib [26] was used for plotting radiometric signals. This choice aligns with the research of [27], who identifies Python as the de facto standard for scientific programming due to its high-level abstraction. However, while Python serves as the primary instrument in this study due to its industry dominance, it is not the sole tool for data analysis. Exposure to established statistical software environments such as SAS, JMP, or SPSS in later curricular stages would further enrich students' analytical arsenal, providing a more holistic view of statistical engineering beyond the Python ecosystem.

The experiment consisted of two phases demonstrating the importance of Chapter 7 ("Algorithm Design Strategies and a Glimpse into the Future").

As a result of the first run, the algorithm classified a significant portion of the data array (indices 31–192) as anomalies. This phenomenon demonstrated the problem of False Positives. Due to the stochastic nature of the Poisson distribution, background noise fluctuations exceeded the low threshold.

The research was conducted as a case study incorporating elements of a quantitative educational experiment, aimed at evaluating the effectiveness of the Physics-Informed Scaffolding approach in teaching fundamental algorithms. The methodology is grounded in a combination of active learning principles [17], cognitive load theory [22], and physics-informed machine learning [12].

Unlike traditional approaches to algorithmic instruction that rely on deterministic data structures [5], the proposed intervention integrates stochastic physical processes as a foundation for cultivating algorithmic thinking. This aligns with contemporary recommendations for the early integration of AI components into academic curricula [8] and addresses the gap between theoretical programming and real-world data processing [9].

Students were assigned the task of localizing a weak signal within a time series simulating radiation monitoring data. The data were interpreted as a sequence of intensity measurements (counts per second) captured by a detector. The primary challenge involved isolating the target signal from the stochastic background without relying on high-level machine learning libraries.

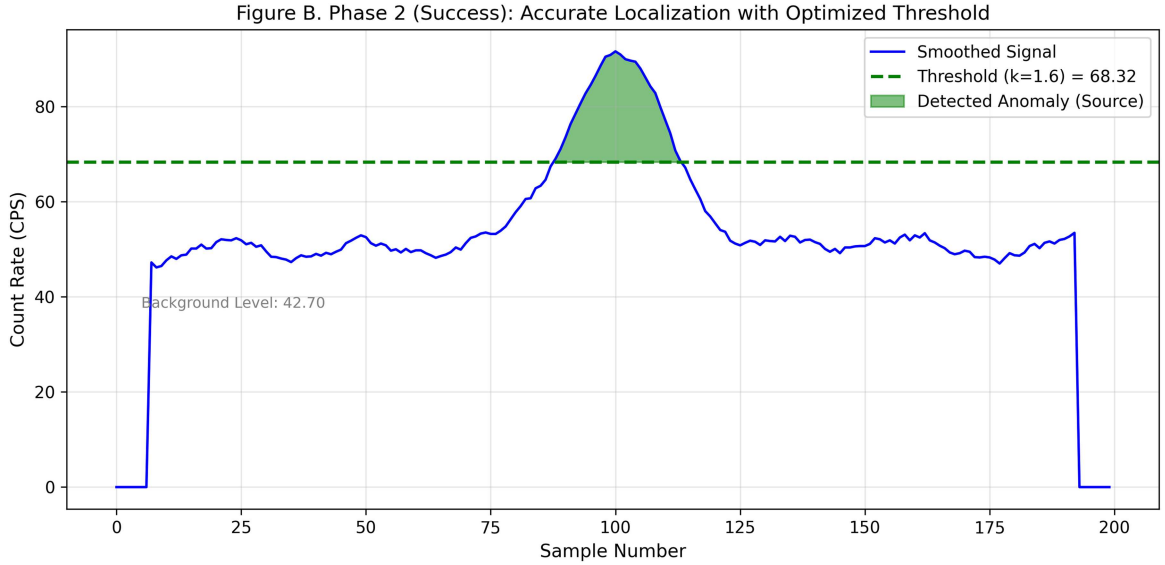

FIGURE 3 | Successful signal localization in Phase 2 following iterative tuning. The green dashed line indicates the optimized threshold ($k = 1.6$), calculated based on the background level. This enabled the algorithm to disregard noise outliers and distinctly isolate (green zone) only the true signal from the source, aligning with the center of the array.

The students implemented fundamental signal processing algorithms—specifically, smoothing and threshold filtering—using core Python, without the aid of high-level ML libraries. The pedagogical framework of the assignment incorporated gradual complexity progression (scaffolding), manual algorithmic implementation, interpretation of the physical context of the data, and iterative parameter adjustment (threshold tuning). The study cohort comprised first-year engineering students (CS1). Following data cleaning (complete-case analysis), the final sample of valid responses was $N = 34$. Data were collected via a structured survey administered upon the completion of the laboratory assignment.

3.4 | Measurement Instrument

The survey was designed using a 5-point Likert scale (1–strongly disagree, 5–strongly agree) and included 9 statements grouped into three latent constructs: ENG (Engagement)–interest and connection to real-world tasks, CU (Conceptual Understanding)–comprehension of algorithmic and physical concepts and SPS (Scaffolding and Process Support)–effectiveness of the step-by-step instructional structure.

4 | Results

Table 2 presents the descriptive statistics for the Likert-scale responses (1–5).

Students demonstrated consistently high performance in all measured categories ($M > 4.0$). The highest average score was obtained by the construct of pedagogical support (SPS, $M = 4.10$), which confirms the effectiveness of the developed step-by-step task structure in processing noisy signals.

The internal consistency (reliability) of the instrument was assessed using Cronbach's α. The instrument demonstrated high reliability across the individual constructs: Engagement ($\alpha = 0.812$), Conceptual Understanding ($\alpha = 0.834$), and Scaffolding and Process Support ($\alpha = 0.819$), as well as for the full 9-item scale ($\alpha = 0.924$). This confirms the high validity of the collected data for subsequent statistical modeling.

Pearson correlation coefficients were computed to examine the interrelationships between the aggregated constructs. As illustrated in the correlation matrix (Figure 4), the results revealed strong and statistically significant positive associations across all three variables.

TABLE 2 | Descriptive statistics of student survey constructs.

Latent construct	Mean (M)	Standard deviation (SD)
Engagement (ENG)	4.04	0.72
Conceptual Understanding (CU)	4.02	0.73
Scaffolding and Process Support (SPS)	4.10	0.79

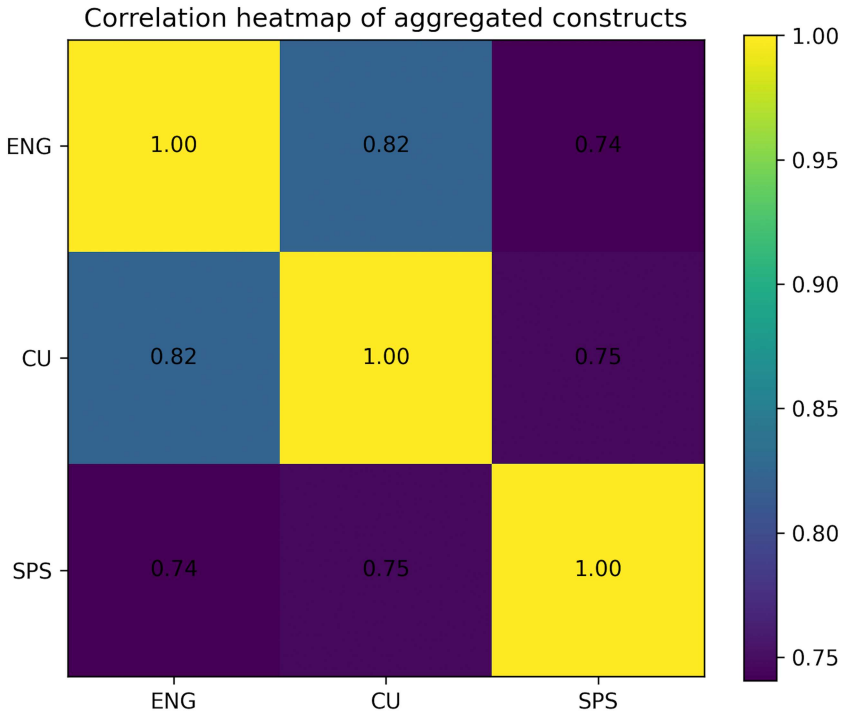

FIGURE 4 | Correlation heatmap of aggregated constructs (ENG, CU, SPS), demonstrating strong positive relationships between engagement, scaffolding, and conceptual understanding.

Engagement (ENG) exhibited a particularly strong correlation with Conceptual Understanding (CU) ($r = 0.822$, $p = 2.54 \times 10^{-9}$), while Scaffolding and Process Support (SPS) showed significant correlations with both Engagement ($r = 0.741$, $p = 5.53 \times 10^{-7}$) and Conceptual Understanding ($r = 0.747$, $p = 3.99 \times 10^{-7}$). These findings suggest that robust procedural scaffolding not only sustains student motivation but also directly facilitates a deeper comprehension of the underlying algorithmic logic.

The internal consistency (reliability) of the instrument was assessed using Cronbach's alpha: Overall Scale $\alpha = 0.924$.

The survey instrument demonstrated high internal consistency, confirming a reliable measurement of student perceptions across all constructs. The key significant correlations are presented in Table 3.

A strong positive correlation was observed between engagement and conceptual understanding ($r = 0.822$), suggesting that context-rich, physically grounded tasks significantly enhance learning outcomes. The strong positive correlation between scaffolding and understanding ($r = 0.747$) indicates that the structured task design effectively facilitates the comprehension of complex algorithms.

TABLE 3 | Correlation matrix of the latent constructs.

Variables	*r*	*p* value
Engagement ↔ Understanding	0.822	< 0.001
Understanding ↔ Scaffolding and Process Support (SPS)	0.747	< 0.001
Engagement ↔ Scaffolding and Process Support (SPS)	0.741	< 0.001

Given the sample size ($N = 34$), statistical significance was interpreted with appropriate caution; however, the observed effect sizes (all $r > 0.70$) indicate substantial and pedagogically meaningful relationships between the constructs, as further visualized in the correlation heatmap (Figure 4).

To examine the impact of pedagogical factors on the level of comprehension, a linear regression model was constructed:

$$CU = \beta_0 + \beta_1 ENG + \beta_2 SPS$$

The results of the analysis yielded the following coefficients: ENG: $B = 0.608$, $p = 0.0002$, SPS: $B = 0.284$, $p = 0.039$. As visualized in Figure 5, the model demonstrates a high explanatory capacity: $R^2 = 0.718$, $R^2_{adj} = 0.699$.

As illustrated in Figure 6, the distribution of scores revealed a high concentration of responses within the 4–5 range, an absence of significant outliers, and stability in ratings across all constructs.

The proposed approach integrates, for the first time, stochastic physical processes (specifically, Poisson noise and Gaussian signals) as a foundational didactic mechanism in a CS1 course to cultivate algorithmic thinking. All supplementary materials, source code, and datasets are available in an open-access repository, ensuring the full reproducibility of the results.

5 | Discussion and Conclusions

The empirical results of this study convincingly demonstrate that integrating physics-informed tasks into introductory programming courses (CS1) significantly enhances the acquisition of fundamental algorithms. Quantitative analysis confirmed

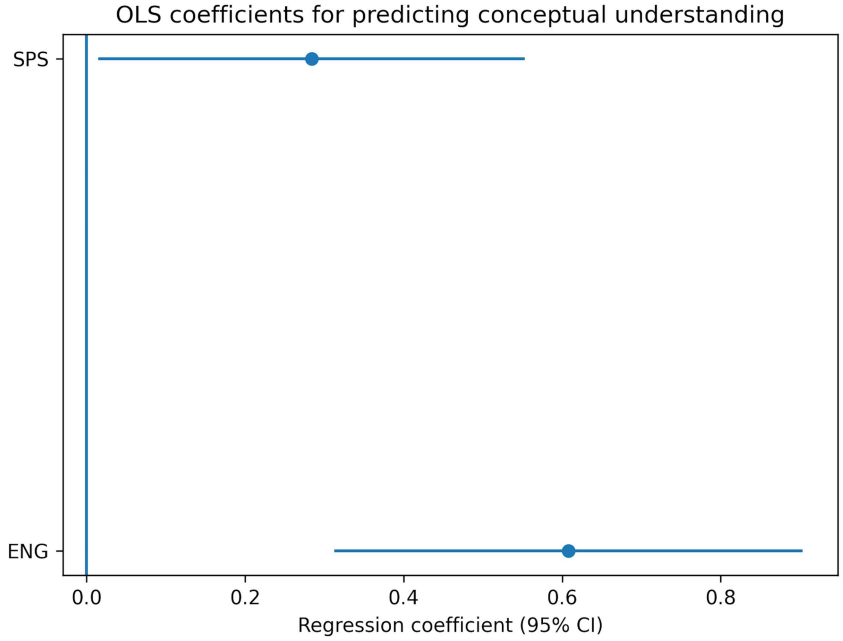

FIGURE 5 | Regression coefficients with 95% confidence intervals for predicting Conceptual Understanding (CU) based on Engagement (ENG) and Scaffolding and Process Support (SPS).

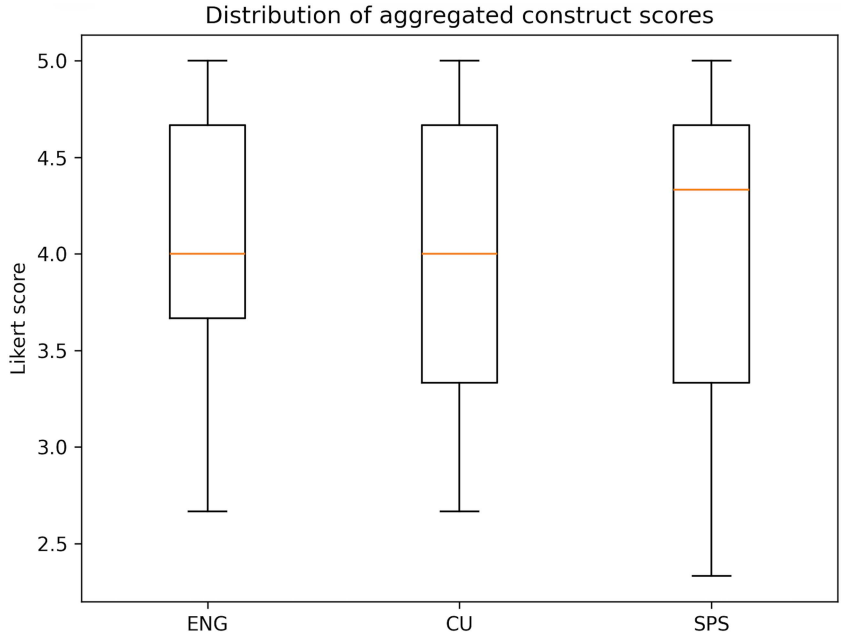

FIGURE 6 | Distribution of student responses across constructs, indicating consistently high perceived engagement and understanding.

that structured pedagogical support has a statistically significant impact on learning outcomes, effectively mitigating cognitive overload when dealing with complex concepts. Student engagement emerged as a key predictor of success, demonstrating a strong correlation with the level of conceptual understanding ($r = 0.822$). The constructed regression model proves that the proposed Physics-Informed Scaffolding methodology explains over 70% of the variance ($R^2 = 0.718$) in students' comprehension of algorithms. For educational research in the fields of engineering and computer science, this metric confirms the robustness and reliability of the proposed instructional design.

The major pedagogical achievement of this study is the substantiation of the need to transition from the traditional "Syntax-First" learning model to a "Data-First" paradigm. Classical computer science education often focuses on the syntax of programming languages and manipulations with abstract arrays that are disconnected from reality. Conversely, the early introduction of stochastic data models—specifically, simulating Poisson noise when searching for radiation anomalies—creates an authentic engineering context for students. Through the independent, manual implementation of noise reduction filters using basic loops, students were able to overcome the so-called "stochastic shock" (the fear of imperfect, noisy real-world data). This transition from a simple search for specific numerical values to the identification of hidden patterns marks a critical shift in mindset: from a novice coder to a professional ready for the challenges of AI engineering.

Particular attention should be given to the impact of this methodology on addressing a fundamental problem in modern artificial intelligence: the "black-box" effect. The rapid development of high-level APIs (such as TensorFlow or PyTorch) creates a temptation to utilize complex algorithms without understanding their internal mechanics. However, as emphasized in international guidelines on AI ethics future engineers are obligated to understand algorithmic decision-making mechanisms to ensure the transparency and safety of systems. By implementing the "moving average" algorithm from scratch, students mastered at a micro-level the mechanics underlying one of the most important operations in deep learning: mathematical convolution. As noted by leading researchers in the field, it was convolution operations that catalyzed breakthroughs in pattern recognition. Thus, learning basic loops through signal processing cultivates the very "engineering intuition" that will empower students in the future not merely to call ready-made neural network functions, but to consciously architect them.

In conclusion, this research demonstrates that grounding algorithmic education in physical contexts creates a powerful and effective methodological framework for training the next generation of artificial intelligence engineers. The proposed approach not only improves academic performance but also serves as a strong motivator, demonstrating to students the real-world impact of their code in solving engineering problems.

5.1 | Limitations and Future Work

Despite the positive results demonstrating high levels of student engagement and improved conceptual understanding, several limitations of this study should be acknowledged.

First, the study was conducted without a control group, which limits the ability to directly compare the proposed approach with traditional teaching methods. Therefore, the findings should be interpreted as indicative of the effectiveness of a specific pedagogical intervention rather than as conclusive evidence of its superiority.

Second, the sample size is relatively small ($N = 34$), which constrains the statistical generalizability of the results. Although the observed internal consistency (Cronbach's α) and regression analysis indicate strong reliability and explanatory

power, further studies with larger cohorts are required to validate the stability of these findings.

Third, the study relies partially on self-reported data, which may introduce subjective bias. This limitation is mitigated by complementary performance-based evaluation, including analysis of students' source code and their ability to correctly localize anomalies in the data.

Fourth, the study represents a pilot implementation and focuses primarily on qualitative assessment of student performance (e.g., code quality and oral defense), rather than a fully controlled quantitative comparison with alternative instructional approaches. This limits the extent to which broad generalizations can be made.

Fifth, the current case study is restricted to one-dimensional signal processing. While this simplification is pedagogically justified to reduce cognitive load at the initial stage of learning, it does not fully reflect the complexity of multidimensional data processing typical of modern deep learning systems.

Future research will address these limitations and further extend the proposed methodology.

First, a controlled experimental design incorporating a comparison group will be implemented to provide statistically grounded evidence of the effectiveness of the Physics-Informed Scaffolding approach relative to traditional instruction.

Second, the case study will be extended to two-dimensional data processing tasks, such as image-based noise reduction. This extension will establish a direct bridge between fundamental algorithmic concepts and applications in computer vision.

Third, future iterations of the course will integrate advanced evaluation metrics, including:

- Receiver operating characteristic (ROC) curves,
- Area under the curve (AUC),
- Confusion matrices.

These methods were intentionally excluded from the current study to manage cognitive load. However, the foundational understanding of signal-to-noise ratio developed here serves as a prerequisite for their effective application.

Finally, future work will explicitly address underlying statistical assumptions of data (e.g., normality and linearity) and examine how violations of these assumptions contribute to noise and impact algorithmic performance.

Author Contributions

Tetiana Nosenko is the sole author of this manuscript and is responsible for all aspects of the research, including conceptualization, methodology, data collection and analysis, and writing of the manuscript.

Acknowledgments

The author has nothing to report.

Conflicts of Interest

The author declares no conflict of interest.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub repository at https://github.com/TetiHAnna/Physics-Informed-CS1.

References

1. T. I. Nosenko, I. V. Mashkina, and V. O. Yaskevych, *Zastosuvannia Algorytmiv ta Struktur Danykh u Shtuchnomu Intelekti: Navchalnyi Posibnyk [Application of Algorithms and Data Structures In Artificial Intelligence: A Textbook]* (Borys Grinchenko Kyiv Metropolitan University, 2025).

2. G. D. O. Lima, J. A. R. Costa, F. A. Dorça, and R. D. Araújo, "An AI-Supported Pedagogical Architecture to Foster Self-Regulated Learning in Virtual Environments," *Computer Applications in Engineering Education* 33 (2025): 1–20, https://doi.org/10.1002/cae.70118.

3. G. T. Rao and N. Suhasini, "Integrating Artificial Intelligence in Higher Education to Enhance Teaching and Learning," *Computer Applications in Engineering Education* 33 (2025): e70085, https://doi.org/10.1002/cae.70085.

4. P. Dumka, D. R. Mishra, R. Chauhan, and N. Pandey, "Python-Powered Structural Analysis: Modeling and Solving 2D Truss Systems With the 'Anastruct' Module," *Computer Applications in Engineering Education* 33 (2025): e70072, https://doi.org/10.1002/cae.70072.

5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, 2022). 4th ed..

6. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).

7. R. Garcia and M. Craig, "20 Years Later: A Replication Study on Teaching CS1 Concepts," *ACM Transactions on Computing Education* 25 (2025): 22.1–22.30.

8. ACM/IEEE/AAAI Joint Task Force*Computer Science Curricula 2023* (ACM, 2023), https://doi.org/10.1145/3664166.

9. S. Perach and G. Alexandron, "Mind the Gap: Confronting the Vast Divide Between CS Teaching and Machine Learning Pedagogy." *European Conference on Technology Enhanced Learning* (Springer, 2024), 344–358, https://doi.org/10.1007/978-3-031-72312-4_24.

10. J. C. Jauregui-Correa and M. Sen, "Revolutionizing Engineering Education: Adapting Curricula to Address Artificial Intelligence Challenges and Opportunities," *ASEAN Journal of Engineering Education* 8, no. 1 (2024): 64–69, https://ajee.utm.my/index.php/ajee/article/view/145.

11. S. I. Flores-Alonso, N. V. M. Diaz, J. Kapphahn, et al., "Introduction to AI in Undergraduate Engineering Education," *IEEE Frontiers in Education Conference (FIE)* 1 (2023): 4.

12. G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-Informed Machine Learning," *Nature Reviews Physics* 3, no. 6 (2021): 422–440.

13. G. F. Knoll, *Radiation Detection and Measurement* (John Wiley & Sons, 2010). 4th ed..

14. J. D. Toscano, V. Oommen, A. J. Varghese, et al., "From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning," *Machine Learning for Computational Science and Engineering* 1, no. 1 (2025): 15.

15. C. Meng, S. Griesemer, D. Cao, S. Seo, and Y. Liu, "When Physics Meets Machine Learning: A Survey of Physics-Informed Machine Learning," *Machine Learning for Computational Science and Engineering* 1, no. 1 (2025): 20.

16. K. Kashinath, M. Mustafa, A. Albert, et al., "Physics-Informed Machine Learning: Case Studies for Weather and Climate Modelling," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379, no. 2194 (2021): 20200093.

17. S. Freeman, S. L. Eddy, M. McDonough, et al., "Active Learning Increases Student Performance in Science, Engineering, and Mathematics," *Proceedings of the National Academy of Sciences* 111, no. 23 (2014): 8410–8415, https://doi.org/10.1073/pnas.1319030111.

18. B. Barron and L. Darling-Hammond, *Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning* (Edutopia, 2008).

19. M. Tedre, T. Toivonen, J. Kahila, et al., "Teaching Machine Learning in K-12 Computing Education: Potential and Pitfalls," *arXiv preprint* arXiv:2106.11034 4 (2021): 1–14.

20. Y. Hao, Y. Liu, B. Liu, G. Amarantidis, and R. Ghannam, "Integrating AI in Engineering Education: A Comprehensive Review and Student-Informed Module Design," *IEEE Transactions on Education* 20 (2025): 1–11.

21. H. Hamadeh, Transforming Engineering Education: Python as the Tool for Modern Engineers *[Unpublished manuscript]* (UCL Discovery, 2023).

22. J. Sweller, "Cognitive Load During Problem Solving: Effects on Learning," *Cognitive science* 12, no. 2 (1988): 257–285.

23. L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes* (Harvard University Press, 1978).

24. J. M. Wing, "Computational Thinking," *Communications of the ACM* 49, no. 3 (2006): 33–35.

25. C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array Programming With Numpy," *Nature* 585, no. 7825 (2020): 357–362, https://doi.org/10.1038/s41586-020-2649-2.

26. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering* 9, no. 3 (2007): 90–95.

27. T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering* 9, no. 3 (2007): 10–20.