AI-Co-Piloted Engineering of an Interactive Educational Suite for Automated Line Integral Visualization and Verification in Vector Calculus*

Halyna Tuluchenko[1,*,†] and Artem Kovalskyi[1,†]

[1] National Technical University "Kharkiv Polytechnic Institute", 2 Kyrpychova St, 61002, Kharkiv, Ukraine

Abstract

This paper introduces an AI-co-piloted engineering workflow for developing specialized educational software in vector calculus. The developed interactive suite integrates symbolic computation via SymPy, procedural 3D animation using the Manim engine, and a PyQt6 user interface that automates the verification and visualization of line integrals. The key methodological contribution lies in the transition from generic demonstration examples to the large-scale automated generation of individualized procedural video content. The implemented scaffolding-lock mechanism ensures a step-by-step analytical solution process by restricting access to the visualization until symbolic verification of the student's input is completed. This is complemented by the Cognitive Reward interface, where pre-rendered 3D simulations provide a geometric interpretation of the specific problem variant, reinforcing the link between symbolic and spatial representations. The system's offline-first architecture, managed through a CSV-based registry, ensures high scalability and operational resilience in environments with unstable network infrastructure. Functional validation confirms that the suite effectively bridges the gap between abstract symbolic manipulation and intuitive spatial perception of vector fields, offering a scalable model for personalized engineering education..

Keywords

AI-co-piloted engineering, Vector field visualization, Automated verification, Scaffolding lock, Cognitive reward, Manim, SymPy, Line integrals, Procedural animation, STEM Education

1. Introduction

The ongoing digital transformation of higher education reveals a persistent gap between advanced technological tools and their effective implementation in highly specialized academic disciplines. A major barrier is the technological gap faced by domain experts who possess deep pedagogical knowledge but lack professional programming skills. This gap hinders the conversion of innovative didactic ideas into modern interactive educational software. As a result, innovative teaching methods remain limited to traditional formats, such as static textbooks or passive video lectures, which cannot deliver the immediate feedback crucial for deep learning.

The rise of Artificial Intelligence (AI) as a co-pilot in software ngineering provides a promising way to bridge these challenges. By using Large Language Models (LLMs) as architectural consultants, academics can bypass the lack of programming expertise and rapidly prototype sophisticated educational tools using open-source stacks. This paradigm shift redefines the educator's role—from writing low-level code to focusing on strategic architectural design and high-level interaction logic. This enables educators to independently develop professional-grade educational software without relying on external developers. Specifically, in Vector Calculus problem-solving requires strong abstract symbolic reasoning combined with well-developed spatial visualization skills. Learning outcomes improve substantially when students work with procedurally generated visualizations customized to their individual problem variants. The ability

* The Third International Conference of Young Scientists on Artificial Intelligence for Sustainable Development: Intelligent Systems and Artificial Intelligence (YAISD-WS 2026:ISAI) May 7-8, 2026, Ternopil, Ukraine

[1,*] Corresponding author.

[†] These authors contributed equally.

Halyna.Tuluchenko@khpi.edu.ua (H. Tuluchenko);

0000–0002–6196–540X (H. Tuluchenko); 0009–0006–6924–1285 (A. Kovalskyi)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-11

CEUR-WS.org/Vol-4326/paper5.pdf

to immediately see the geometric trajectory of a line integral or the local behavior of a vector field —right after performing the analytical calculation—delivers powerful feedback. This visual confirmation serves as a potent motivating factor, where a dynamic 3D scene becomes a well-earned cognitive reward for successfully

navigating the analytical stage. The resulting visual sequence does more than merely illustrate the solved problem; it transforms abstract formalism into a verified conceptual experience, offering a deeper understanding of the geometric nature of calculated symbolic constructions. Although modern educational media feature high visual quality, most solutions provide only deterministic content, which prevents educators from dynamically adapting visual scenarios to match specific problem parameters. The proposed AI-co-piloted approach overcomes this limitation by enabling flexible adaptation of mathematical models and visualization scenarios to specific edagogical goals. This frees instructors to concentrate on pedagogical design, while the AI co-pilot automatically resolves low-level graphical and algorithmic issues that would otherwise require significant time and expertise.

2. Research Aim and Objectives

The aim of this study is to develop and validate an AI-co-piloted rapid engineering methodology for creating a dual-level interactive ducational suite that supports the full cycle of automated rification
and procedural 3D visualization of line integrals. To achieve this aim, the study addresses the following objectives:

1. 1. Define the stages of a human–AI collaborative workflow for rapid prototyping of professional-grade educational software and for overcoming technical limitations of graphical engines.
2. 2. Justify the architectural synergy among the symbolic (SymPy), visual (Manim), and interface (PyQt6) layers to enable invariant verification of mathematical expressions.
3. 3. Develop an Instructor–Student dual-module paradigm based on he principle of distributed content generation and autonomous offline knowledge validation.
4. 4. Implement the scaffolding-lock mechanism and the cognitive reward system to enforce step-by-step control over calculation accuracy and deliver personalized visual feedback.
5. 5. Evaluate the effectiveness of LLMs in architectural synchronization and rapid translation of pedagogical algorithms into executable code, thereby enabling educators to independently develop scalable digital learning environments.

The structure of the paper follows the logical flow of the research. ection 2 reviews existing visualization methods and cognitive support tools in mathematics education. Section 3 describes the proposed AI-assisted rapid engineering methodology and the system's technology stack. Section 4 details the Instructor– Student architectural paradigm and the implementation of the scaffolding-lock mechanism. Section 5 presents the results of the pilot testing along with an evaluation of the engineering efficiency of the approach. Finally, Sections 6 and 7 discuss the system's limitations and draw overall conclusions.

3. Literature Review

Traditional Vector Calculus instruction primarily focuses on mastering symbolic differentiation and integration techniques to compute key field characteristics, such as curl, divergence, circulation, work, and flux. Classic textbooks [5,11, 19] lay the theoretical foundation for field theory, emphasizing symbolic manipulation and conventional integration methods. The need to modernize these pedagogical approaches to align with the digital demands of technical universities is well justified in the analytical review [17]. An analysis of the evolution of Vector Calculus

textbooks [4] shows ongoing efforts to balance abstract symbolism with geometric intuition. The use of nteractive tools in this context is theoretically grounded in Paivio's dual-coding theory [10], which asserts that learning is more effective
when verbal (symbolic) and non-verbal (visual) channels are engaged simultaneously. Meta-analyses of visual interventions in mathematics education [13] confirm that spatial tools significantly improve learning outcomes, particularly in complex domains such as field theory [9].
Central to the proposed methodology is the concept of scaffolding [3, 6]. Recent studies stress that effective AI-based educational tools must incorporate metacognitive support [14, 20] to prevent students from bypassing the analytical reasoning process. This justifies our implementation of the scaffolding-lock mechanism, which restricts access to visual feedback until the symbolic solution is verified as correct. The development of mathematical visualization tools is closely linked to the use of the Manim animation engine [2,8]. Although several educational resources on vector field theory are available [1], most provide only deterministic, predesigned content. This approach does not allow real-time modification of mathematical parameters, which significantly limits their adaptability for tasks requiring variable student input.
Current trends in AI-assisted engineering [12] point toward a transition to automating the creation of such animations using Large Language Models (LLMs). In prior works, the authors developed vector field visualization algorithms based on the Manim library [15,16,18]. The present study advances this line of research by integrating AI methods directly into the engineering workflow for developing educational applications. To clarify the positioning of the proposed suite, it is compared with leading educational platforms such as Khan Academy, MIT OpenCourseWare, and 3Blue1Brown. lthough these resources provide high-quality deterministic content
and scoring-based assessment systems, they often separate the nalytical task from the visual result. In contrast, our approach utilizes the "visual-as-areward" paradigm.
Unlike standard testing systems that focus on quantitative metrics (grades), the proposed suite implements a psychological incentive: the 3D visualization is not just a demonstration, but an earned cognitive reward that delivers additional spatial insights only after successful symbolic verification.

4. Methodology: AI-Co-Piloted Rapid Engineering and

The development of the interactive educational suite is based on an I-co-piloted rapid engineering methodology, in which the Gemini Large Language Model (LLM) serves as a key tool for architectural analysis and rapid prototyping. For domain experts accustomed to closed mathematical systems (e.g., Maplets Builder in Maple), LLMs significantly ease the transition to the open-source Python cosystem. The methodology integrates symbolic computation, rocedural animation, and event-driven interfaces through iterative Human–AI design cycles.

4.1. Human–AI Collaborative Development Workflow

The development process is formalized as a collaborative dialog between the subject matter expert and the AI agent, consisting of the following stages:
• Prompt-Based Architecture Design. The expert formulates pecifications for a unified interface between the Instructor and Student Modules. Since the modules do not communicate directly, the AI assists in designing a shared CSV-registry that ensures consistent interpretation of mathematical expressions and trajectory parameters across both applications.
• Code Synthesis and Algorithmic Integration. The Gemini LLM is sed to generate precise Manim code for dynamic 3D scenes requiring high geometric accuracy. Heuristic Debugging and Functional Refactoring. The AI supports optimization of interface components, bidirectional parsing of mathematical expressions, management of relational problem variant databases, and implementation of robust input validation algorithms.

4.2. Multi-Layer Functional Integration: SymPy, Manim, and PyQt6

The system architecture is built upon the tight integration of three core layers:

1. Analytical Layer (SymPy). This layer performs symbolic equivalence esting by analytically simplifying the difference between the eference solution and the user's input. This allows for the robust verification of mathematically identical but syntactically distinct answers. Furthermore, a unified analytical solution procedure is integrated into both modules after input parsing:
   In the Instructor Module: The procedure is used for the preemptive detection of task design errors (e.g., if an integral lacks an analytical solution or exceeds the intended educational complexity). This nables indirect complexity control and prevents logical (rather than merely syntactic) errors during variant generation.
   In the Student Module: The same logic is applied to automatically generate the reference answer, against which the user's input is compared through expression difference simplification.
2. Visualization Layer (Manim). This layer implements the concept of Automated Mass Video Generation. Unlike traditional educational content on platforms like YouTube, which focuses on passive heoretical explanation, this approach ensures the procedural generation of unique dynamic scenes for each individual task. This transforms visualization from a mere illustrative supplement into a cognitive reward for successful analytical problem-solving.
3. Interface and Validation Layer (PyQt6). This layer manages the step-by-step validation logic based on a Finite State Machine (FSM). Access to the final visualization and subsequent calculation steps is hierarchically locked until the analytical layer confirms the validity of the intermediate stages.

4.3. Overcoming Manim Engine Limitations

A critical aspect of the methodology is the utilization of AI to overcome documented limitations of the Manim graphical engine.

1. • Spatial Consistency Fix. The research addresses the 3D Text Plane Collapse issue, where MathTex objects lose their viewer-oriented alignment during camera rotations (set_camera_orientation) [7]. By implementing the add_fixed_in_frame_mobjects() method, the system ensures the stability of UI elements within a 2D layer superimposed on the dynamic 3D space. This approach guarantees consistent visual parameters and text readability throughout the scene, regardless of subsequent camera angle manipulations.ю
2. • Final Frame Preservation. To prevent the automatic closure of media content upon completion, a short pause (10 ms) is implemented. By introducing it before the rendering process concludes, the system preserves the final frame (the result exposure) for detailed student analysis.

4.4. Evaluation of AI-Driven Development Efficiency

The engineering process was evaluated through two distinct metrics: development velocity for the research team and content-generation efficiency for the instructor.

- Development Velocity and Technology Migration. The research team leveraged its extensive experience in developing educational software within the established Maple ecosystem to design the system's core logic. However, the transition to a Python-based stack (PyQt6, Manim, SymPy) was strategically chosen to integrate Manim, which is currently recognized as the international standard for high-fidelity procedural mathematical animation. This migration was facilitated by the LLM acting as a high-speed syntactic

bridge. For a developer already proficient in algorithmic logic but new to Python-specific libraries, the time required for syntax mastery and rendering troubleshooting was reduced from ~150 to 40 hours. This 3.75× increase in development velocity was primarily driven by the AI's ability to generate approximately 65% of the functional boilerplate code (UI signalhandling and standard object configurations), allowing the authors to focus on high-level architectural integration.

- Instructor-Side Productivity. The automated "Verify-then-Record" workflow was benchmarked against traditional manual methods of generating individual problem sets. For a standard set of 30 unique task variants (including symbolic verification, coordinate consistency checks, and static visual aids), the system demonstrated a 45% reduction in total preparation time.
- Procedural Content Advantage. A critical efficiency gain was bserved in the production of procedural 3D animations. While manual creation of such assets is typically bypassed by faculty due to its prohibitive time-to-output ratio, the integrated Manim-based module makes dynamic visualization a standard component of the task. This represents a qualitative shift in pedagogical capability that is effectively non-comparable to traditional static text-processing methods.

5. System Architecture: The Instructor–Student

The proposed system architecture is based on the principles of istributed content generation and autonomous knowledge verification (Fig. 1). This design decouples the computationally intensive video rendering process (assigned to the instructor side) from the student module, which remains a lightweight application that does not require high-performance hardware.

5.1. Instructor Module: Content Authoring and Validation

The Instructor Module acts as a CAD-like system for mathematical content creation. It supports the full lifecycle of educational content preparation—from AI-Co-Piloted Engineering of an Interactive Educational Suite.

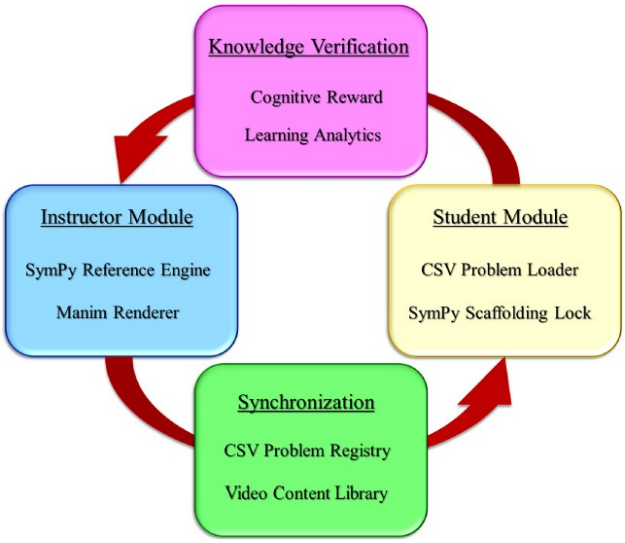


Figure 1: System architecture of the Instructor–Student dual-module paradigm for line integral visualization and verification.

analytical model verification to the automated generation of personalized visual materials. The module's internal structure is organized into the following functional components:

• Analytical Pre-verification and Parsing. The module ensures task validity through a robust verification pipeline powered by the SymPy engine. Before a variant is accepted, the system performs bidirectional parsing of the input strings, converting them into symbolic objects. A specialized is_integral_

calculable procedure then executes a multi-stage heuristic check:

◦ Divergence Analysis: Detecting discontinuities or infinite results (nan, zoo);

◦ Domain Validation: Filtering out complex results (I) that may arise

when a path crosses a singularity;

◦ Elementary Function Filtering: Ensuring the result consists only of a

predefined set of allowed mathematical functions (e.g., sin, exp, log, sqrt) to maintain pedagogical alignment with the engineering curriculum;

◦ Analytical Solvability Check: Verifying that the integral is not too complex for a closed-form solution. This rigorous filtering ensures that only mathematically sound and educationally appropriate tasks are advanced to the rendering stage, while the mandatory validation sequence (verification before recording) guarantees the integrity of the CSV-registry.

• Parameterization Engine. Given the vector field F(P, Q,R) and the parametric curve r(t) defined over the interval $t \in [a, b]$, the module automatically synthesizes the corresponding Manim script. This engine performs symbolicto-numerical transformation by parsing expressions via SymPy and computing the scalar product F·dr to derive the integrand. To optimize performance, symbolic objects are converted into numerical functions using the lambdify method, ensuring that vector directions and curve coordinates are computed rapidly during the rendering process.

• Procedural Rendering Loop. The system automates the generation of unique 3D animations by treating visualization as a dynamic output of the mathematical model. The loop manages the high-precision mapping of calculated values to visual mobjects, where a symbolic primitive (antiderivative) is evaluated at each frame to drive the synchronized ValueTracker and Decimal-Number components. This ensures that the real-time update of the integral value and the movement of the red indicator arrow (representing the field's force) strictly correspond to the underlying analytical solution of the specific problem variant.

• Database Management and Audit. To maintain the integrity of the educational repository, the module includes a View Table interface that opens the CSV-registry in a separate modal window. This component provides a centralized view of all generated tasks, allowing the instructor to perform a visual audit of the stored variants. The management is streamlined through a Delete Variant function, which facilitates the synchronized removal of the symbolic record from the CSV-file and the corresponding MP4 video file.

This manual override capability ensures that the instructor can quickly refine the active problem set by removing redundant or experimental variants, maintaining a lean and error-free data schema.

This multi-stage architecture ensures that the instructor maintains full control over the pedagogical content quality. The system's backend, engineered through an AI-co-piloted workflow, automates the transition from symbolic mathematics to visual content by autonomously managing rendering tasks and data records. This enables the rapid production of professional-grade educational assets without the need for the instructor to perform low-level programming or manual video editing for each individual task.

5.2. Student Module: Interactive Scaffolding and Verification

The Student Module is designed as an intelligent support environment that implements a scaffolding-lock mechanism, ensuring a rigorous step-by-step validation of the analytical solution. The technical implementation of the module is based on the following components:

• Hierarchical Step Validation. To prevent error propagation, the system enforces a mandatory sequence of intermediate symbolic milestones. This includes the verification of differential forms derived from the parametric path components (e.g., dx, dy, dz) and the subsequent construction of

the integrand. Access to the visual simulation is programmatically restricted until each stage is confirmed, ensuring that the student masters the calculus workflow
before receiving visual feedback.

• Dynamic CAS-based Verification Pipeline. Unlike traditional testing systems, the module does not store reference answers in the CSV-registry. Instead, it employs a Just-In-Time (JIT) verification strategy. When a student submits a result, the system invokes a symbolic evaluation procedure identical to the calculate routine in the Instructor Module. By performing real-time integration of the variant's parameters using the SymPy engine, the system generates a reference value programmatically. This architectural decision eliminates the possibility of students extracting answers via registry
inspection, thereby increasing the pedagogical integrity of the ssessment.

• Equivalence Testing. Invariant Equivalence Testing (SymPy Core): The validation of the student's input is performed via an invariant equivalence test. The module evaluates the symbolic difference between the student's expression and the dynamically computed reference. By applying the simplify() and expand() methods within the SymPy framework, the system confirms mathematical identity (e.g., recognizing that $2\sin(x)\cos(x) \equiv \sin(2x)$) only if the difference reduces to identically zero. Since the parameters have already
passed the pre-verification stage in the Instructor Module, the student-side evaluator operates with a streamlined set of constraints, focusing on realtime identity verification.

• State-Triggered Modal Interface. Upon successful verification, the module transitions to a congratulatory modal interface (Success Window). This interface programmatically activates an integrated media player that loads the corresponding Manim-generated MP4 file from the local repository.

• Iterative Visual Analysis. The playback interface is equipped with functional controls for deep result auditing, including a Play Again mechanism for repeated observation of the vector field dynamics and a status label confirming the correlation between the simulation and the specific problem variant.

This dual-layer verification architecture effectively transforms the software rom a passive visualization tool into an active pedagogical agent. By combining real-time symbolic computation with conditional content delivery, the system ensures that the 3D simulation is perceived not merely as an illustration, but as a verified cognitive reward for analytical accuracy.

5.3. Data Interoperability and Synchronization

The modules are connected through a unified data schema consisting of a CSV registry and a shared media library. In the CSV file, each record represents an individual problem variant and defines its specific mathematical features. According to the internal file structure, these fields include: the variant ID (variant), vector field components (P, Q,R), parametric trajectory equations (x(t), y(t), z(t)), and integration limits [a, b].

Such organization ensures seamless synchronization between the SymPy analytical engine and the Manim visualization module, enabling the system to interpret the mathematical context consistently. This architectural approach provides:

• Offline Scalability. The system requires no persistent server onnection, making it particularly suitable for educational use in regions with unstable network infrastructure.

• Modular Independence. Instructors can easily expand the problem database by adding new CSV records and corresponding video files without modifying the student application. The Student Module automatically loads the updated registry on startup. The resulting dual-module architecture ensures a clear separation between content authoring and interactive learning. While the system perates as a standalone offline application, its internal logic—engineered through an AI-co-piloted development workflow—effectively utomates the complex transition from abstract symbolic mathematics to dynamic 3D simulations based on the specific problem parameters.

6. Implementation Results and Functional Validation

The pilot testing of the developed suite was carried out within the “Vector Calculus” module of a higher mathematics course for engineering students. The main goal was to assess the system’s functional reliability in generating large numbers of individualized tasks and to verify seamless data exchange between the instructor and student modules.

6.1. Performance and Stability Metrics

Thirty unique problem variants were prepared for the pilot student group. The technical performance metrics of the system were recorded as follows:

1. Parsing Reliability. During the pilot testing, no user reports were received regarding symbolic expression interpretation errors or analytical core failures. This indicates sufficient robustness in the processing algorithms for trigonometric and polynomial functions integrated via SymPy.
2. Rendering Throughput. On a standard laptop, generating a 60-second dynamic 3D scene takes approximately 4 minutes (25 s for static context construction and 3 min 35 s for dynamic elements and camera movements). During this period, the interface remains in a wait state until the media file is fully formed, ensuring operational sequence and data integrity.
3. Data Integrity. The implementation of the unified data schema facilitated error-free synchronization between the CSV-registry records and the corresponding high-definition MP4 files.

6.2. UI/UX Verification and Visual Logic

A detailed analysis of the interfaces and the generated content confirms the correct functional logic of the system:

• Instructor Module (Fig. 2). Using variant 10 as an example, the system demonstrates operations with a conical helix r(t) = (t cos(2t), t sin(2t), t) within the field F = (y, x,–z). The Calculate function provides an exact analytical result (I = 0), allowing the instructor to assess the methodological value of the task before adding it to the variant library.

• Student Module (Fig. 3). The screenshot shows the scaffolding-lock mechanism in action. When an incorrect integrand f(t) is entered, the system displays the message “Incorrect answer. Try again!” and blocks access to the visualization.

• Cognitive Reward Interface (Fig. 4). The final rendered scene for complex spatial curves includes real-time numerical updates of the integral value. A red arrow indicates the vector field direction at the current point on the trajectory, visually linking motion direction with the sign of the integral increment.

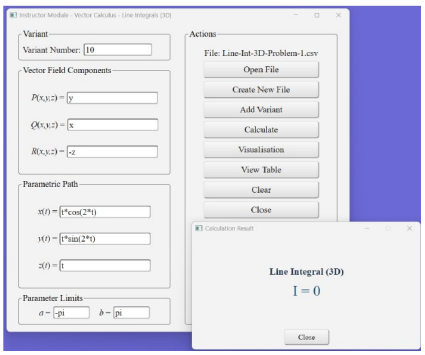


Figure 2: System architecture of the Instructor–Student dual-module paradigm for line integral visualization and verification.

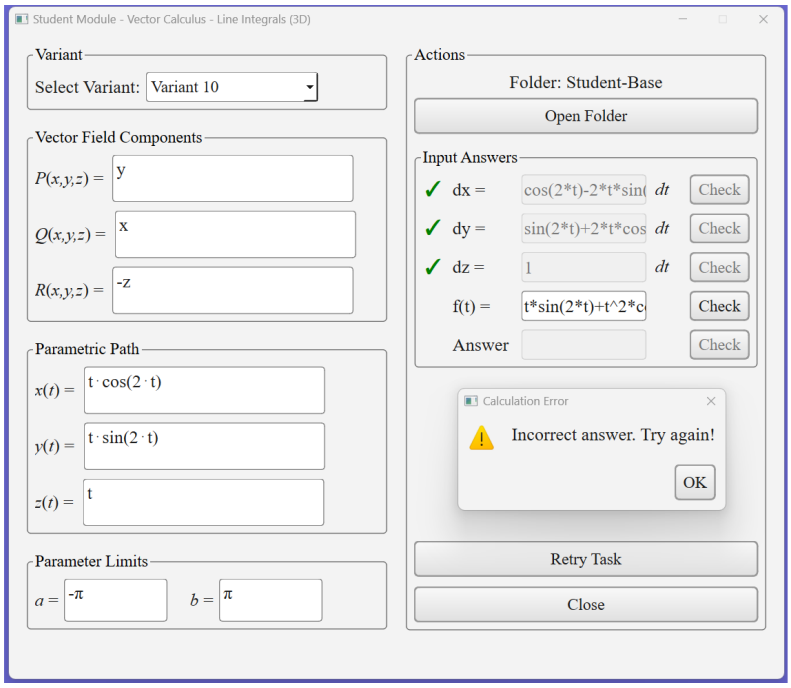


Figure 3: Student's interface demonstrating the scaffolding-lock mechanism.



Figure 4: Cognitive reward window showing the final procedural 3D simulation.

The tight synchronization between the analytical engine and the rendering module demonstrates the robustness of the proposed architecture when handling complex 3D Vector Calculus problems across varied educational scenarios.

7. Discussion

The implementation results position the developed suite within the broader context of digital transformation in higher education. The proposed architecture effectively bridges mathematical formalism and dynamic visualization. The development process demonstrates a shift from traditional programming toward iterative, AI-co-piloted design. Using LLMs as an intelligent intermediary significantly reduces the barriers associated with integrating heterogeneous libraries such as SymPy and Manim. This confirms that LLMs serve not merely as code generators, but as effective tools for high-level architectural design and synchronization of heterogeneous components. This shift is particularly important as it enables educators and researchers to independently develop specialized educational software without external development teams. Furthermore, the scaffolding-lock mechanism redefines the role of visualization in mathematics learning. Unlike static illustrations, the procedural animations in this system serve as a cognitive reward, creating a clear link between analytical accuracy and visual confirmation. Real-time visualization of integral values (Fig. 4) bridges the gap between symbolic computation and the geometric meaning of line integrals, promoting deeper conceptual understanding. The system achieves scalability through a simple unified CSV-based registry, which avoids unnecessary architectural complexity. By relying on local file-based assets rather than external databases, the system maintains full offline-first operation—a critical advantage in regions with unstable network infrastructure. The framework's high adaptability allows instructors to expand the task database by simply modifying metadata, with minimal risk of introducing software regressions.

Despite its demonstrated stability, the system has several limitations. First, the SymPy analytical core currently cannot handle integrals that lack elementary antiderivatives but could be solved using symmetry or properties of odd/even functions. Second, rendering time is constrained by the current architecture of the Manim engine and cannot be significantly reduced at the application level. Future work will focus on developing an online version of the system with integrated learning analytics capabilities (Fig. 1). This would enable instructors to identify common student errors in real time and dynamically adjust task difficulty based on collective learning patterns.

Conclusions

This study confirms the viability of AI-co-piloted engineering for developing highly specialized interactive educational tools. This research resulted in the development and testing of an interactive suite designed for the automated verification of line integral omputations. The system ensures a comprehensive procedural visualization of the entire mathematical context, mapping all lements of the problem statement—from vector field parameters and the specific integration path to the final numerical result—into a dynamic 3D scene. The main findings can be summarized as follows. First, AI acting as an architectural assistant enables seamless integration of symbolic computation (SymPy) and procedural animation (Manim) libraries, substantially reducing development time while maintaining mathematical accuracy. Second, the scaffoldinglock mechanism combined with the cognitive reward interface strengthens the connection between symbolic computation and geometric nderstanding of line integrals. Finally, the offline-first architectural solution, based on a unified data schema (CSV-registry), ensures high operational resilience in environments with unstable digital infrastructure. The resulting suite offers a complete, scalable solution for personalized learning in Vector Calculus. Future work will focus on extending the system with online learning analytics to enable real-time monitoring of student errors and dynamic adaptation of task difficulty.

Declaration on Generative AI

During the preparation of this work, the authors utilized Gemini 3 lash as an AI-co-pilot in order to:

• Software Engineering and Rapid Prototyping. Accelerate the development of the system architecture by optimizing the integration logic between SymPy, Manim, and PyQt6. The AI tool acted as a technical collaborator in generating core functional modules, troubleshooting complex rendering loops, and enhancing the efficiency of the procedural visualization engine.

• Scientific Narrative and Academic Refinement. Support the authors, who are non-native English speakers, in maintaining style consistency and refining the academic tone. The AI tool assisted in transforming technical descriptions into a formal scientific narrative and ensuring the precise use of terminology across all sections.

Following the collaborative sessions with the AI tool, the authors conducted a comprehensive manual review, verification, and editing of all outputs. The authors take full responsibility for the ethodology, the integrity of the generated software, and the final content of the publication.

References

[1] 3Blue1Brown, Divergence and curl: The language of Maxwell's equations, fluid flow, and more, YouTube, 2017. URL: https://www.youtube.com/watch?v=rB83DpBJQsE.

[2] 3Blue1Brown, Site of 3Blue1Brown: Explaining math and physics with animation, 2025. URL: https://www.3blue1brown.com/.

[3] B. Belland et al., Synthesizing results from empirical research on computer-based scaffolding in STEM education: A meta-analysis, Review of Educational Research 87 (2016) 309–344. doi:10.3102/0034654316670999.

[4] R. Godard, M. Lewis, Two textbooks on vector analysis: Gibbs and Runge, in: Research in History and Philosophy of Mathematics, Springer, 2024, pp. 111–126. doi:10.1007/978-3-031-46193-4_7.

[5] O. Heliarovska, L. Liubchyk, M. Aslandukov, Practicum on the course "Mathematical Analysis". Line and Surface Integrals: Study Guide, NTU "KhPI", Kharkiv, 2022 [in Ukrainian].

[6] N. Kim, B. Belland, A. Walker, Effectiveness of computer-based scaffolding in STEM education, Educational Psychology Review 30 (2018) 397–429. doi:10.1007/s10648-017-9419-1.

[7] Manim Community Developers, Mobjects not remaining fixed in frame during camera rotation, Manim issue tracker, 2024. URL: https://github.com/ManimCommunity/manim/issues.

[8] Manim Community Developers, Manim Community v0.20.0 documentation, 2026. URL: https://docs.manim.community/en/stable/.

[9] L. Medina Herrera et al., Enhancing mathematical education with spatial visualization tools, Frontiers in Education 9 (2024) 1229126. doi:10.3389/feduc.2024.1229126.

[10] A. Paivio, Mental Representations: A Dual Coding Approach, Oxford University Press, Oxford, 1990.

[11] N. Romanchuk, N. Shapoval, O. Maiboroda, Multiple, Line, Surface Integrals. Elements of Field Theory: Study Guide, NUK, Mykolaiv, 2025 [in Ukrainian].

[12] F. Schneider, P. Libbrecht, V. Mas, Creating animations for maths teaching: A praxis-oriented discussion paper about the use chatbots for the creation of mathematical animations with Manim, IU Discussion Papers, 2025. doi:10.56250/4092.

[13] J. Schoenherr, A. Strohmaier, S. Schukajlow, Learning with visualizations helps: A meta-analysis of visualization interventions in mathematics education, Educational Research Review (2024) 100639. doi:10.1016/j.edurev.2024.100639.

[14] M. Tsakeni et al., Mapping the scaffolding of metacognition and learning by AI tools in STEM classrooms (2005–2025), Journal of Intelligence 13 (2025) 148. doi:10.3390/jintelligence13110148.
[15] H. Tuluchenko, Interactive visualization of vector calculus: Development of a gamified educational application using the Manim library, in: Proceedings of the VII International Scientific-Practical Conference "Innovative Technologies in the Development of Modern Society", Institute of Spatial Planning and Advanced Technologies (IPPT), Lviv Polytechnic National University, Lviv Polytechnic Publishing House, Lviv, Ukraine, 2025, pp. 50–54. URL: https://ippt-lpnu.herokuapp.com/about?section=1518b7f9385ed0004c55d7b.
[16] H. Y. Tuluchenko, T. P. Bilousova, Verification potential of custom-built software modules in the mathematical training of engineers amid digitalization, Modern Information Technologies and Innovation Methodologies of Education in Professional Training Methodology Theory Experience Problems 79 (2026) 144–155. doi:10.31652/2412-1142-2026-79-144-156.
[17] H. Tuluchenko, Analysis of methodical support for teaching vector analysis and approaches to its improvement, in: H. D. Berehova (Ed.), Theory and Methods of Teaching in Technical Higher Education Institutions, Knyzhkove vydavnytstvo FOP Vyshemyrskyi V. S., Kherson, 2025, pp. 253–277 [in Ukrainian].
[18] H. Tuluchenko, A. Kovalskyi, Manim library as a tool for animating vector fields in the educational process, in: Proceedings of the II International Scientific-Practical Student Conference "IT Space of Today: Trends, Innovations and Development Prospects", V. N. Karazin Kharkiv National University, Kharkiv, 2025, pp. 383–384 [in Ukrainian].
[19] H. Tuluchenko, T. Nemchenko, Methodological instructions for practical classes on "Calculation of line integrals of the I and II kind", NTU "KhPI", Kharkiv, 2025 [in Ukrainian].
[20] T. Wang et al., Computer-based scaffoldings influence students' metacognitive monitoring and problem-solving efficiency in an intelligent tutoring system, Journal of Computer Assisted Learning 39 (2023) 1652–1665. doi:10.1111/jcal.12824.