Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XLII, pp. 1209-1215
A. Caggiano, G. Etse, P. Folino, M. Goldschmit, M. Pucheta, M. Storti (Eds.)
C. Pairetti, L. Ponzellini (Issue eds.)
Buenos Aires, November 11-14, 2025

USO DE INTELIGENCIA ARTIFICIAL EN CURSOS DE MÉTODOS NUMÉRICOS: RIESGOS Y OPORTUNIDADES

USING ARTIFICIAL INTELLIGENCE IN NUMERICAL METHODS COURSES: RISKS AND OPPORTUNITIES

César I. Pairetti[a]

[a]*Universidad Nacional de Rosario, Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Escuela de Ingeniería Mecánica, Rosario, Argentina*

Keywords: Programación, CAE, LLM, Taller

Abstract.

En muchos cursos de ingeniería, los estudiantes utilizan software basados en el Método de Elementos Finitos (FEM) sin conocer sus fundamentos, asumiendo los resultados como válidos aun cuando carecen de sentido físico. De modo análogo, los Grandes Modelos de Lenguaje (LLM) como GPT o Deepseek permiten generar código para Dinámica de Fluidos Computacional (CFD), pero sin garantizar la comprensión de la lógica algorítmica ni de los fundamentos matemáticos. Si bien estas herramientas facilitan la obtención de fragmentos básicos de código, su uso acrítico incrementa el riesgo de errores de modelado e implementación. Este trabajo propone un enfoque pedagógico en el cual los estudiantes producen, traducen y optimizan código con apoyo de LLM, a partir de referencias bibliográficas tradicionales. La revisión línea a línea favorece la comprensión de la estructura de los programas, de la lógica de implementación y de los criterios de validación de resultados. La metodología se evaluó en dos ediciones de la asignatura electiva de CFD en la carrera de Ingeniería Mecánica. Los estudiantes mostraron una comprensión más sólida de los métodos numéricos respecto a cohortes previas, y en entrevistas finales manifestaron una mirada más crítica sobre el uso de herramientas de Inteligencia Artificial en el aprendizaje.
Keywords: Coding, CAE, LLM, Workshop.

Abstract. In many engineering courses, students use Finite Element Method (FEM) software without knowing its fundamentals, assuming results as valid even when they lack physical meaning. Similarly, Large Language Models (LLM) such as GPT or Deepseek allow code generation for Computational Fluid Dynamics (CFD), but without ensuring understanding of the algorithmic logic or mathematical foundations. While these tools facilitate obtaining basic code fragments, their uncritical use increases the risk of modeling and implementation errors. This work proposes a pedagogical approach in which students produce, translate, and optimize code with LLM support, based on traditional bibliographic references. Line-by-line review favors understanding of program structure, implementation logic, and criteria for validating results. The methodology was evaluated in two editions of the CFD course for the Mechanical Engineering degree. Students showed a stronger understanding of numerical methods compared to previous cohorts, and in final interviews they expressed a more critical view regarding the use of Artificial Intelligence tools in learning.

ISSN: 2591-3522 DOI: 10.70567/mc.v42.ocsid8396

1 INTRODUCCIÓN

En la actualidad, los profesionales y estudiantes de carreras STEAM (Ciencia, Tecnología, Ingeniería, Arte y Matemática) enfrentan el desafío de integrar rápidamente un conjunto multidisciplinar complejo que incluye matemáticas avanzadas (como Cálculo y Álgebra Lineal), principios fundamentales de física, y conocimientos de informática para modelar y resolver problemas tecnológicos reales. Esta integración es especialmente crítica en áreas de ingeniería donde la resolución numérica y computacional de problemas es fundamental para diseñar, analizar y optimizar sistemas y procesos industriales cada vez más sofisticados Logan (2012); Felippa (2015).

Sin embargo, la creciente complejidad de los problemas técnicos y la aceleración en los tiempos de formación académica -que se traduce en currículas más comprimidas- exigen que los estudiantes adquieran competencias avanzadas en menor tiempo. Esto implica una presión creciente para dominar simultáneamente fundamentos teóricos rigurosos, algoritmos numéricos y su implementación práctica mediante programación computacional. En muchos casos, el aprendizaje formal -en contextos universitarios- de conceptos matemáticos, físicos y computacionales se desarrolla en cursos separados, lo que dificulta la adecuada integración de estos saberes en entornos aplicados y genera brechas entre teoría y práctica (Dyke and Friswell, 2018; Chen et al., 2023).

Paralelamente, el surgimiento y rápida adopción de Grandes Modelos de Lenguaje (*Large Language Models*, LLM), como GPT y similares, ha transformado el ecosistema profesional y educativo. Estas tecnologías, accesibles mediante interfaces de chat intuitivas usualmente llamadas IA (Inteligencia Artificial), permiten generar texto, código y soluciones a problemas complejos a partir de instrucciones en lenguaje natural. En particular, su capacidad para producir fragmentos de código para la implementación de algoritmos ha sido rápidamente incorporada por estudiantes y docentes como herramienta para la resolución de trabajos prácticos en asignaturas relacionadas con métodos numéricos, simulación y modelado computacional (Wang et al., 2023; Pérez et al., 2023).

Si bien estas herramientas de inteligencia artificial generativa ofrecen enormes ventajas en términos de accesibilidad y rapidez para obtener soluciones iniciales, también plantean una problemática pedagógica crítica: el uso acrítico de código generado automáticamente puede fomentar una comprensión superficial, y en ocasiones errónea, de los fundamentos matemáticos y físicos que sustentan los métodos numéricos implementados. Esto incrementa el riesgo de aceptar resultados numéricos sin sentido físico o de aplicar modelos inapropiados para la naturaleza del problema, lo cual puede afectar la formación profesional y la capacidad de resolver problemas reales con rigor y confianza (Ng et al., 2023; Mitchell, 2019).

En este contexto, es necesario replantear las estrategias pedagógicas para integrar el uso de LLM no como simples generadores automáticos de código, sino como asistentes que acompañen un proceso activo y crítico de aprendizaje. Esto implica que los estudiantes sean los protagonistas en la traducción, optimización y validación del código generado, apoyándose en su conocimiento disciplinar para interpretar y corregir resultados, fomentar el pensamiento crítico y desarrollar habilidades de auditoría y supervisión que serán fundamentales en su ejercicio profesional futuro (Luckin et al., 2016; Holmes et al., 2019; Mitchell, 2019, 2025).

El presente trabajo propone y evalúa una metodología didáctica centrada en esta perspectiva, implementada en la asignatura electiva "Dinámica de Fluidos Computacional" de la carrera Ingeniería Mecánica de la Universidad Nacional de Rosario (UNR). Se explora el impacto del uso de un “tutor personalizado basado en IA” para asistir a los estudiantes en la generación,

revisión y validación iterativa de código para resolver problemas de métodos numéricos, con el objetivo de fortalecer su comprensión de los fundamentos teóricos y promover una apropiación crítica de las tecnologías de inteligencia artificial emergentes.

2 USOS PEDAGÓGICOS DE GRANDES MODELOS DE LENGUAJE

La compresión de los planes de estudio y la creciente demanda de innovación tecnológica obligan a los estudiantes a abordar problemas multidisciplinarios en plazos cada vez más reducidos. En este contexto, la irrupción de los LLM -modelos de aprendizaje profundo entrenados sobre enormes volúmenes de datos textuales para capturar patrones del lenguaje y razonamiento general- representa un cambio de paradigma tanto en la práctica profesional como en la educación en ciencia y tecnología. Estos modelos, basados en arquitecturas tipo Transformer, pueden generar, resumir, traducir y analizar textos, así como producir código computacional a partir de descripciones en lenguaje natural, como describe Fleuret (2024).

La facilidad de acceso a Moodelos de Lenguaje a través de interfaces de chat ha provocado una rápida adopción por parte de estudiantes y profesionales en tareas de generación de código, resolución de trabajos prácticos y automatización de procesos en asignaturas de métodos numéricos y simulación computacional. Sin embargo, su uso acrítico puede reforzar tendencias ya observada con herramientas basadas en FEM (Plevris and Markeset, 2018): una aplicación automática de la herramienta sin comprensión profunda de los resultados obtenidos, debido a un conocimiento superficial de los fundamentos matemáticos y la lógica algorítmica, ignorando los criterios básicos de validación.

López-Belmonte et al. (2024) mencionan el potencial de los LLM como asistentes cognitivos, así como también destacan los riesgos asociados a su integración en entornos de aprendizaje. En particular, se ha señalado la importancia de diseñar estrategias pedagógicas que promuevan la reflexión crítica sobre los resultados obtenidos y el entendimiento de los métodos subyacentes. Así, se abre una nueva línea de investigación y experimentación sobre cómo articular el potencial de la IA generativa con la formación en métodos numéricos en carreras STEAM, promoviendo tanto la autonomía como la capacidad de validación de los futuros profesionales.

2.1 Fundamentos de los Generative Pre-trained Transformers (GPT)

Los *Generative Pre-trained Transformers* (GPT) son una arquitectura de *deep learning* particularmente eficiente para desarrollar modelos de lenguaje. Un GPT es entrenado en dos fases: preentrenamiento (*pre-training*) y ajuste fino (*fine-tuning*).

En la primera etapa, el modelo se expone a un corpus masivo de texto, aprendiendo a predecir la siguiente palabra en secuencias de lenguaje natural. Esta etapa de aprendizaje automático, sin supervisión humana, permite que el modelo capte dependencias estadísticas complejas entre palabras, frases y párrafos, incorporando patrones gramaticales, semánticos y de razonamiento distribuidos a lo largo de los datos. Fleuret (2024) describe en detalle este proceso, que puede realizarse con diversos grados de profundidad como reportan Shean et al. (2025), comparando LLM de diversas compañías, o Zhong et al. (2024), evaluando el modelo de *pensamiento profundo* de OpenAI.

La arquitectura *Transformer* emplea mecanismos de *self-attention*, como explican Vaswani et al. (2017), que permiten ponderar el peso relativo de cada palabra respecto a todas las demás de una secuencia. Esta capacidad de atención global favorece la captación de relaciones de largo alcance y contextos complejos en el texto. Una vez completado el preentrenamiento, el modelo

puede ser ajustado mediante *fine-tuning* sobre tareas específicas (como generación de código, resolución de preguntas, ó tutoría educativa) usando conjuntos de datos anotados o técnicas de aprendizaje por refuerzo con retroalimentación humana.

Al recibir un *prompt* o instrucción, el GPT genera texto de manera autoregresiva: predice la palabra más probable dado el contexto anterior, la añade a la secuencia y repite el proceso hasta alcanzar el largo deseado o una señal de detención. Este mecanismo le permite producir respuestas coherentes, redactar explicaciones, traducir o generar código en función del contexto y la tarea planteada por el usuario.

Si bien los GPT han demostrado resultados notables en comprensión y generación de texto, su razonamiento se basa en las correlaciones lingüísticas aprendidas y no en comprensión semántica profunda, lo que puede dar lugar a errores sutiles, respuestas verosímiles pero incorrectas o dificultades en la interpretación de información novedosa.

3 APLICACIONES EN ENSEÑANZA DE MÉTODOS NUMÉRICOS

En trabajos previos sobre el uso de modelos de lenguaje en la enseñanza de métodos numéricos, se observó una tendencia entre los estudiantes a interactuar con sistemas como ChatGPT empleando un enfoque pasivo: simplemente formulaban un *prompt* solicitando la generación de un código base para resolver un problema asignado, esperando obtener una solución funcional de manera automática Tramallino et al. (2024). Si bien esta modalidad facilitaba la obtención de scripts iniciales y reducía la barrera de entrada para tareas de programación, también presentaba limitaciones claras en términos de comprensión conceptual y validación crítica. Los estudiantes tendían a aceptar el código generado sin analizar en detalle su lógica interna, estructura algorítmica ni la adecuación a los fundamentos matemáticos del problema tratado.

En contraste, la experiencia implementada y documentada en Tramallino et al. (2024) propuso una dinámica activa centrada en la **traducción y optimización de código** asistida por GPT. En esta actividad, los estudiantes debían tomar un fragmento de código existente –por ejemplo, desarrollado originalmente en Octave (octave.org) para la resolución de ecuaciones por el Método de Volúmenes Finitos– y traducirlo a otro lenguaje (como Python (python.org), recurriendo al apoyo de GPT sólo para cuestiones puntuales de sintaxis, depuración y comprensión de funciones específicas.

Este abordaje obligó a los alumnos a analizar línea por línea la implementación, a descomponer la lógica algorítmica y a reflexionar sobre la correspondencia entre el código y los conceptos numéricos subyacentes. El modelo de lenguaje pasó a desempeñar el rol de tutor interactivo, guiando la traducción y validación pero sin reemplazar la actividad intelectual central del estudiante.

Como resultado, se observó un aumento en la apropiación crítica de los métodos numéricos y una mejora en la capacidad de identificar errores o inconsistencias, tanto en la traducción como en la interpretación de los resultados. Las entrevistas finales reflejaron una actitud más reflexiva respecto al uso de herramientas de IA generativa, destacando la importancia de la validación y la comprensión conceptual por sobre la mera automatización del código.

4 ANÁLISIS DE LA IMPLEMENTACIÓN DE ACTIVIDADES DE PROGRAMACIÓN EN LA ELECTIVA CFD

La función principal de un curso de métodos numéricos, como la electiva de CFD para la carrera de Ingeniería Mecánica de FCEIA-UNR, es que los estudiantes aprendan a emplear técnicas para la solución aproximada de modelos matemáticos relevantes para la ingeniería Versteeg and

Malalasekera (2007); Moukalled et al. (2016). Sin embargo, si bien los programas formativos suelen incluir cursos previos de programación, en la práctica es común que muchos estudiantes no hayan desarrollado experiencia en la escritura de código propio para resolver problemas numéricos Pairetti et al. (2023). Esto representa un desafío importante, ya que la programación es el puente esencial entre el planteo matemático y la resolución computacional.

En las ediciones 2022 y 2023 de la asignatura, se buscó abordar esta dificultad incorporando actividades específicas que profundizan las habilidades de programación, pero sin requerir tiempo adicional de exposición teórica. Estas actividades se desarrollaron en modalidad taller, promoviendo el trabajo autónomo pero acompañado por docentes y pares.

En la edición 2022, se propuso a los estudiantes emplear un *prompt* en una interfaz de LLM (por ejemplo, ChatGPT) para obtener un código base que resolviera un problema académico concreto de métodos numéricos. Esta modalidad facilitó el acceso a scripts funcionales y redujo las barreras iniciales para la programación. Sin embargo, se observó que este enfoque promovía una actitud pasiva: los estudiantes tendían a aceptar el código generado como una caja negra, prestando poca atención a la lógica interna del programa, la estructura algorítmica o la correcta implementación de conceptos como esquemas de interpolación o integración numérica.

Con el objetivo de fomentar una comprensión más profunda, en la edición 2023 se modificó la estrategia: en lugar de solicitar un código base directamente al LLM, los estudiantes recibieron un código escrito en Octave (generalmente de fuentes bibliográficas clásicas o desarrollado por los docentes) y debieron traducirlo a Python, recurriendo a la IA solo para resolver dudas puntuales de sintaxis o depuración. De esta forma, se forzó a que cada estudiante analizara y reconstruyera el flujo lógico del algoritmo, debiendo implementar explícitamente los conceptos básicos de los métodos numéricos (manejo de arrays, ciclos, interpolación, integración, etc.).

Esta dinámica demostró ser más efectiva para la apropiación conceptual: los estudiantes tuvieron que enfrentarse con la estructura del código, entender la función de cada sección y adaptar las estrategias algorítmicas a un lenguaje diferente, todo ello bajo la supervisión de la IA en rol de asistente y no de generador principal (Tramallino et al., 2024).

A partir de la experiencia acumulada, se identificó una potencial profundización de esta estrategia para próximas ediciones: pedir a los estudiantes que propongan primero el algoritmo en pseudocódigo, utilizando la IA sólo para traducir y optimizar esa solución conceptual en un lenguaje de programación concreto. Este enfoque implica una interacción más rica y compleja con el LLM, donde el estudiante ejercita no sólo la comprensión de métodos numéricos sino también el diseño algorítmico, la capacidad de abstracción y la validación del código final. Además, promueve el desarrollo de competencias clave para el uso crítico y eficiente de herramientas de IA en la ingeniería.

5 CONCLUSIONES

En este artículo delineamos cómo la aparición de los LLM modifican el ecosistema dentro del cual debemos llevar adelante nuestras tareas docentes y propusimos una metodología particular para explotar la IA como una herramienta pedagógica, a fin de evitar su uso acrítico y promover prácticas de aprendizaje activo (Alomá Bello et al., 2022).

El transcurso de las experiencias docentes escritas es consistente con lo planteado por Luckin et al. (2016) y Holmes et al. (2019), quienes destacan que la incorporación de IA en la educación sólo resulta formativa cuando promueve la metacognición y la autonomía del alumno. En este sentido, las dinámicas basadas en traducción y análisis de código fomentan un razonamiento más profundo sobre los algoritmos y sobre los límites de los modelos computacionales, tal como

advierten Ng et al. (2023) y Mitchell (2025) al señalar los riesgos de dependencia cognitiva en el uso acrítico de sistemas generativos.

Asimismo, la comparación entre cohortes permitió verificar que los estudiantes que interactuaron con LLM en un rol de tutor percibieron la herramienta como un apoyo didáctico, y no como una fuente de soluciones automáticas. Esta transición en la percepción del recurso tecnológico constituye un indicio positivo de apropiación crítica, alineado con los metaestudios sobre IA educativa revisados en la introducción.

De cara a futuras ediciones del curso, se proyecta implementar actividades donde los estudiantes formulen algoritmos en pseudocódigo y utilicen LLM para su implementación y optimización en lenguajes concretos. Esta propuesta busca ampliar el grado de reflexión y abstracción en el diseño algorítmico, promoviendo competencias de pensamiento computacional y validación crítica que trasciendan la mera escritura de código.

En conjunto, los resultados refuerzan la idea de que los LLM pueden integrarse de manera productiva en la enseñanza de métodos numéricos, siempre que su uso esté mediado por una orientación pedagógica que priorice la comprensión, la verificación y el pensamiento crítico por sobre la automatización de respuestas.

En síntesis, la integración de la IA en la enseñanza de métodos numéricos debe orientarse a fortalecer el rol activo y reflexivo del estudiante. Utilizar estos modelos como asistentes para revisar, optimizar o traducir código propio, en lugar de como generadores automáticos, facilita la apropiación de los fundamentos teóricos y el desarrollo de habilidades de programación y validación esenciales para el ejercicio profesional.

REFERENCES

Alomá Bello M., Crespo Díaz L.M., González Hernández K., and Estévez Pérez N. Fundamentos cognitivos y pedagógicos del aprendizaje activo. *Mendive. Revista de Educación*, 20(4):1353–1368, 2022.

Chen Z., Kumar A., and Singh L. On the use of large language models in programming education. *Journal of Educational Computing Research*, 61(4):754–775, 2023. http://doi.org/10.1177/07356331231102587.

Dyke S.J. and Friswell M.I. Teaching computational mechanics: Challenges and methods. *Computers Structures*, 199:1–12, 2018. http://doi.org/10.1016/j.compstruc.2017.11.006.

Felippa C.A. Teaching finite element method: Reflections and perspectives. *Engineering Computations*, 32(7):2141–2158, 2015. http://doi.org/10.1108/EC-01-2015-0024.

Fleuret F. *The little book of deep learning*. University of Geneva, , 2024.

Holmes W., Bialik M., and Fadel C. *Artificial Intelligence in Education: Promises and Implications for Teaching and Learning*. Center for Curriculum Redesign, Boston, MA, 2019.

Logan J.D. *Applied Mathematics*. Wiley, Hoboken, NJ, 2012.

López-Belmonte J., Dúo-Terrón P., Marín-Marín J.A., and Moreno-Guerrero A.J. Machine learning as a methodological resource in the classroom. In *Artificial Intelligence of Things for Achieving Sustainable Development Goals*, pages 233–253. Springer, 2024.

Luckin R., Holmes W., Griffiths M., and Forcier L.B. *Intelligence Unleashed: An Argument for AI in Education*. Pearson, 2016. https://www.pearson.com/uk/educators/higher-education-educators/programs-and-services/intelligence-unleashed.html.

Mitchell M. *Artificial Intelligence: A Guide for Thinking Humans*. Farrar, Straus and Giroux, New York, NY, 2019.

Mitchell M. Artificial intelligence learns to reason. 2025.

Moukalled F., Mangani L., and Darwish M. *The Finite Volume Method in Computational Fluid*

Dynamics: An Advanced Introduction with OpenFOAM and Matlab, volume 113 of *Fluid Mechanics and Its Applications*. Springer, 2016. ISBN 978-3-319-16874-6.

Ng J.K.K., Lee S.T., and Tan P.M. Ai-assisted learning: Opportunities and challenges. *Educational Technology Research and Development*, 71:1–20, 2023. http://doi.org/10.1007/s11423-023-10175-0.

Pairetti C.I., Trivisonno N., Godino D., and Venier C.M. Diseño de un taller de cfd para la enseñanza de fundamentos del método de volúmenes finitos. In *Mecánica Computacional*, volume XL, pages 1549–1558. Asociación Argentina de Mecánica Computacional, Concordia, Argentina, 2023. ISBN 978-987-781-123-2.

Plevris V. and Markeset G. Educational challenges in computer-based finite element analysis and design of structures. 2018.

Pérez J., Garcia M., and Lopez L. Chatgpt in stem education: A double-edged sword. *International Journal of STEM Education*, 10(20), 2023. http://doi.org/10.1186/s40594-023-00401-2.

Shean R., Shah T., Sobhani S., Tang A., Setayesh A., Bolo K., Nguyen V., and Xu B. Openai o1 large language model outperforms gpt-4o, gemini 1.5 flash, and human test takers on ophthalmology board-style questions. *Ophthalmology Science*, page 100844, 2025.

Tramallino C., Pairetti C., and Rodríguez G.L. Reflexiones en torno al uso de ia en educación superior: dos casos comparados. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, (37):e9–e9, 2024.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., and Polosukhin I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Versteeg H.K. and Malalasekera W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, Harlow, England, 2nd edition, 2007. ISBN 9780131274983.

Wang Y., Guo D., and Zhang Y. Chatgpt as a tutoring assistant in higher education programming courses. *Computers Education: Artificial Intelligence*, 4:100117, 2023. http://doi.org/10.1016/j.caeai.2023.100117.

Zhong T., Liu Z., Pan Y., Zhang Y., Zhou Y., Liang S., Wu Z., Lyu Y., Shu P., Yu X., et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.