

INTEGRATING NUMERICAL METHODS AND PROGRAMMING IN ENGINEERING EDUCATION

Simina Maris¹ Liliana Dorneanu² Ștefan Alfred Mariș³ Iulia Știrb⁴ Carmen Holotescu⁵

Abstract: This article presents an innovative approach to engineering education by integrating numerical methods with Python programming in a proposed course titled "**Numerical Methods for Linear Systems with Applications in Python.**" The course focuses on foundational techniques such as Gaussian elimination, LU factorization, and the Jacobi method, emphasizing their implementation and application in solving real-world problems. By utilizing Python and its libraries (NumPy, SciPy), students gain hands-on experience with numerical computation and matrix manipulation, fostering critical thinking and practical skills. The curriculum bridges the gap between theoretical knowledge and professional demands, equipping students with the tools to tackle complex engineering challenges. This model highlights the importance of applied education in preparing future engineers for a rapidly evolving technological landscape.

Key words: Numerical Methods, Python Programming, Engineering Education, Applied Learning

1. INTRODUCTION

In the contemporary landscape of engineering education, the integration of numerical methods and programming is paramount. This fusion equips students with the computational proficiency necessary to tackle complex engineering problems effectively. Numerical methods, such as Gaussian elimination, LU factorization, and iterative approaches like the Jacobi method, form the backbone of problem-solving in fields such as structural analysis, optimization, and computational physics. Similarly, programming tools like Python and its associated libraries offer a robust platform for implementing these methods, making them accessible and applicable in real-world scenarios.

Computational engineering, which involves developing and applying computational models to solve engineering design challenges, has become indispensable across various industries.

The significance of embedding numerical modelling into engineering curricula is underscored by recent studies. For instance, Lonsky et al. [1] demonstrated that incorporating simulation-based problem-solving enhances students' computational skills, thereby better preparing them for real-world applications. Similarly, Sachmpazidi et al. [2] highlighted the benefits of integrating numerical modelling into introductory physics laboratories, noting improvements in students' understanding and application of computational tools.

Moreover, the evolution of computational tools and methods has significantly influenced engineering practices. Reddy's extensive work on finite element methods [3] provides a foundational framework for understanding complex structural behaviours, emphasizing the necessity of computational approaches in modern engineering analysis.

Additionally, Ober-Blöbaum's research into discrete mechanics and optimal control [4] offers insights into the application of numerical methods for optimizing engineering systems. Furthermore, the development of computational models, as discussed by Laurière [5] in the context of mean field games, showcases the interdisciplinary applications of numerical methods and programming, extending beyond traditional engineering fields.

In summary, the integration of numerical methods and programming within engineering education is not merely beneficial but essential. It fosters a generation of engineers adept at utilizing computational tools to devise innovative solutions to complex challenges, aligning academic preparation with the evolving demands of the engineering profession.

¹Associate Professor, Ioan Slavici University, email: simina.maris@islavici.ro

²Associate Professor, Ioan Slavici University, email: liliana.dorneanu@islavici.ro

³Lecturer, Ioan Slavici University, email: stefan.maris@islavici.ro

⁴Lecturer, Ioan Slavici University, email: iulia.stirb@islavici.ro

⁵Professor, Ioan Slavici University, email: carmen.holotescu@islavici.ro

The aim of this paper is to propose a comprehensive framework for integrating numerical methods and programming into engineering curricula. By designing a specialized course, we demonstrate how theoretical knowledge can be seamlessly combined with computational practice to enhance student learning. The paper discusses the importance of numerical methods, their implementation using Python, and the pedagogical strategies employed to foster active and applied learning. This approach not only bridges the gap between theoretical understanding and real-world application but also prepares students for interdisciplinary challenges in their professional careers. Through this contribution, we aim to provide a scalable and effective model for modernizing engineering education.

2. COURSE DESIGN AND IMPLEMENTATION FRAMEWORK

In engineering education, the integration of numerical methods with modern programming tools is essential for developing students' problem-solving capabilities. This section explores three fundamental numerical methods—Gaussian elimination, LU decomposition, and the Jacobi method—and discusses the application of Python and its associated libraries in facilitating these methods.

2.1. Numerical Methods

Gaussian elimination [6], [7] is a direct method for solving systems of linear equations by transforming the coefficient matrix into an upper triangular form, enabling straightforward back-substitution to find solutions. It provides a systematic approach to solving linear systems, foundational in various engineering analyses, and is particularly useful in structural analysis and circuit theory, where multiple linear equations are prevalent.

LU decomposition [6], [7] transforms a matrix into the product of a lower triangular matrix (L) and an upper triangular matrix (U), simplifying the solution of linear systems, especially when multiple right-hand side vectors are involved. LU decomposition enhances computational efficiency by allowing for the reuse of the decomposed matrices, reducing the computational cost in solving linear systems and is widely used in numerical simulations and modelling, where repeated solutions of linear systems with the same coefficient matrix but different right-hand sides are required.

Jacobi method [6], [7] is an iterative technique for solving diagonally dominant systems of linear equations, where each variable is solved in sequence and iteratively refined. This method offers a parallelizable approach suitable for large-scale problems, providing flexibility in handling sparse and structured matrices and is effective in scenarios where direct methods are computationally expensive or impractical, such as in large-scale simulations and real-time systems.

2.2. Digital Instruments

The implementation of these numerical methods is greatly facilitated by programming languages like Python, which, coupled with specialized packages and libraries, offers a robust environment for numerical computation.

Python [8] is a high-level, interpreted programming language known for its readability and simplicity, making it accessible for educational purposes. Using it encourages the development of clear and concise code, allowing students to focus on understanding numerical methods without the overhead of complex syntax. Python serves as a platform for implementing algorithms and visualizing results, enhancing the learning experience.

Python offers several distinct advantages over tools like MATLAB, making it a superior choice for teaching numerical methods in engineering education. Firstly, Python is an open-source, free-to-use platform, which significantly reduces barriers to access compared to MATLAB's expensive licensing fees. This accessibility ensures that students and institutions can adopt Python without financial constraints, making it an equitable choice for diverse educational settings. Secondly, Python's ecosystem includes powerful libraries such as NumPy, SciPy, and Matplotlib, which provide

functionality comparable to MATLAB for numerical computation, matrix manipulation, and data visualization, while also offering greater flexibility for integration with other tools and applications. Additionally, Python's syntax is intuitive and easy to learn, making it ideal for beginners while still being robust enough for advanced applications.

Unlike MATLAB, which is specialized for numerical tasks, Python is a general-purpose programming language that allows students to explore a broader range of applications beyond numerical methods, including machine learning, web development, and data analysis. This versatility prepares students for a wider array of professional roles. Furthermore, Python has a large, active community that continuously contributes to its development, ensuring access to extensive resources, tutorials, and support. These advantages make Python a practical and future-proof tool for engineering education.

Together with Python, there are some dedicated packages for implementing numerical methods, such as NumPy [9] and SciPy [10]. These packages provide support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays, offering efficient implementations of numerical routines, enabling the handling of large datasets and complex computations. Through these packages is facilitated the implementation of numerical methods such as matrix factorizations and iterative solvers, providing built-in functions that simplify coding and improve performance.

Additionally, a useful package for data visualization in Python is Matplotlib [11]. It enables the creation of static, animated, and interactive visualizations, assisting in the interpretation of numerical results through graphical representations, aiding in the comprehension of complex data. Matplotlib allows students to visualize convergence behaviour of iterative methods and the effects of different parameters on solutions.

3. CASE STUDY: A NUMERICAL METHODS COURSE FOR ENGINEERS

The proposed course, titled "**Numerical Methods for Linear Systems with Applications in Python,**" is designed to equip engineering students with both theoretical knowledge and practical skills in solving linear systems using numerical techniques. This innovative curriculum integrates mathematical methods with programming, providing a comprehensive learning experience that bridges the gap between theory and application.

3.1. Course Objectives

The primary objective of the course is to familiarize students with numerical methods for solving linear systems, such as Gaussian elimination, LU factorization, and iterative methods like the Jacobi method.

By incorporating programming in Python, the course also aims to enhance computational skills, enabling students to implement these methods effectively and analyse their applications in real-world scenarios.

This dual focus ensures that students not only understand the theoretical foundations but also develop the ability to apply these methods practically.

3.2. Structure and Content

The course is structured to provide a gradual progression from fundamental concepts to advanced applications. It begins with a theoretical overview of linear systems and numerical methods, followed by hands-on sessions focused on implementing these methods in Python. The course content includes the following modules:

- **Programming Skills:** A brief introduction to Python basics, focusing on its syntax, data structures, and fundamental programming concepts, followed by a training in Python and its libraries (NumPy, SciPy) for numerical computation and matrix manipulation.
- **Introduction to Linear Systems:** Recapitulation of analytical methods like substitution and Cramer's rule.
- **Numerical Methods:** A detailed exploration of Gaussian elimination, LU factorization, and iterative methods, including their theoretical foundations and practical relevance.
- **Applications:** Real-world problems, such as structural analysis, resource optimization, and electrical circuit modelling, to demonstrate the relevance of numerical methods.

3.3. Learning Activities

The course employs an experiential learning approach, encouraging active participation and collaboration. Key activities include:

1. **Individual Assignments:** Students implement algorithms for Gaussian elimination and LU factorization in Python, testing their accuracy and efficiency.
2. **Group Projects:** Teams develop Python programs to solve complex systems of equations using iterative methods like the Jacobi method, analysing convergence and stability.
3. **Case Studies:** Real-world problems, such as traffic flow simulation or resource allocation, are modelled and solved using numerical techniques, highlighting their practical applications.
4. **Interactive Sessions:** Discussions and feedback on coding challenges and numerical errors foster deeper understanding.

3.4. Evaluation Metrics for the Course

The evaluation metrics for the proposed course are designed to assess both the theoretical understanding and practical skills of the students. These metrics emphasize continuous assessment, project-based learning, and reflective practices to ensure comprehensive student development.

The course employs regular evaluations to monitor student progress and reinforce learning outcomes. Key elements include **periodic quizzes** (short quizzes administered at the end of each module), and **coding exercises** (weekly coding tasks focusing on the implementation of the discussed numerical methods in Python).

A significant component of the evaluation involves collaborative projects that simulate real-world engineering challenges. Examples include **team projects** (students working in small groups to develop Python programs that address specific problems) and **project reports**. Alongside the implementation, students submit detailed reports explaining their methodology, challenges encountered, and how these were addressed. These reports assess their ability to document their work and reflect on the learning process.

Students are encouraged to maintain a reflective journal throughout the course, documenting their learning journey. These journals serve to highlight personal progress in understanding numerical methods and programming, reflect on challenges faced during coding exercises and projects and how they were overcome and assess the practical relevance of the course content in preparing for real-world engineering problems.

The course culminates in a comprehensive evaluation, including a **written exam** to test theoretical knowledge of numerical methods, a **coding test** and a **capstone project** in which the students are required to address a large-scale, interdisciplinary problem, integrating all the methods learned throughout the course.

The evaluation criteria are weighted to balance theoretical knowledge and practical application: Quizzes and coding exercises: 30%, Group projects and reports: 30%, Final assessment (exam, coding test, and capstone project): 40%.

This structured evaluation ensures a holistic assessment of students' understanding and skills, equipping them with the competence and confidence required for professional engineering challenges.

3.5. Importance of Python Integration and Relevance for the Students

Python is central to the course, serving as an accessible and versatile platform for implementing numerical methods. Its user-friendly syntax allows students to focus on understanding the mathematical concepts rather than grappling with overly complex programming details. The integration of Python's powerful libraries, such as NumPy and SciPy, significantly enhances the learning experience. These libraries enable efficient manipulation of matrices, solving linear systems, and executing numerical algorithms with ease, all while maintaining high computational performance. Furthermore, Python's plotting and visualization capabilities, supported by libraries like Matplotlib, allow students to graphically explore the behaviour of numerical methods. This visual feedback transforms abstract concepts into tangible insights, enabling learners to better understand algorithmic processes, error propagation, and convergence behaviours. Python's open-source nature also ensures that students can continue to use these tools beyond the academic setting, making it a lifelong skill.

The course equips students to meet the challenges of modern engineering by fostering analytical thinking, technical proficiency, and computational literacy. Mastering numerical methods and their Python implementation prepares students to solve complex problems in diverse fields such as structural engineering, fluid dynamics, optimization, and beyond. In addition to technical competence, the course enhances employability by building sought-after skills in programming, data analysis, and problem-solving. These competencies align closely with industry demands, ensuring students are well-prepared for roles in engineering, data science, and research. Python's relevance in cutting-edge fields like artificial intelligence and machine learning further amplifies the value of these skills, providing students with a competitive edge in a rapidly evolving technological landscape. By embedding these capabilities into the curriculum, the course not only supports academic growth but also ensures long-term professional relevance.

The adoption of this course in resource-constrained institutions can be significantly facilitated by leveraging Python's open-source nature and compatibility with free development environments. Tools such as Google Colab, Jupyter Notebooks, or Visual Studio Code provide accessible platforms that require minimal hardware resources, as they can operate on low-cost computers or even entirely in the cloud. This eliminates the need for expensive software licenses or high-performance infrastructure. Additionally, Python's extensive documentation and active global community offer abundant free resources for both students and educators, enabling self-paced learning and troubleshooting. Institutions could further address resource limitations by fostering partnerships with industry or academic organizations to support faculty training in Python and numerical methods. Collaborative initiatives, such as sharing digital teaching materials or providing remote guest lectures, can also reduce costs while enhancing the educational experience. These strategies ensure that even institutions with limited budgets can effectively implement the course and equip their students with modern computational skills.

3.6. Benefits and Challenges of Applied Education

Integrating numerical methods and digital tools into the engineering curriculum offers several educational benefits. First, it enhances students' understanding of theoretical concepts by allowing them to implement algorithms and observe their behaviour through simulations. Second, it develops practical skills in programming and numerical computation, preparing students for real-world engineering challenges where these abilities are indispensable. Third, it promotes critical thinking and problem-solving, encouraging students to analyse and optimize computational solutions. Fourth, it fosters collaborative learning through group projects and coding exercises, which mirror professional

engineering environments. Finally, it bridges the gap between academia and industry, equipping students with skills that align with modern workforce demands.

However, there are also challenges to this approach. One disadvantage is the steep learning curve associated with programming, particularly for students with little or no prior experience in coding, which may initially hinder their progress. Another potential drawback is the additional time and effort required to design and implement such a curriculum, including training educators to effectively teach both numerical methods and programming. Despite these limitations, the long-term benefits of this integrated approach outweigh its initial challenges, making it a valuable model for preparing engineering students for the complexities of modern technological landscapes.

This course can serve as a foundational pillar in computational and applied mathematics, seamlessly connecting with advanced subjects such as fluid dynamics, structural analysis, control systems, and optimization. By introducing students to both the theoretical and computational aspects of numerical methods, the course lays a strong groundwork for specialized courses that rely heavily on computational problem-solving. Furthermore, the integration of programming in Python bridges traditional mathematical concepts with modern technological applications, fostering interdisciplinary collaboration. Students can apply the skills gained in this course to capstone projects, research initiatives, or internships, where they encounter real-world problems requiring numerical modelling and computational efficiency. Embedding this course also supports the development of soft skills, such as teamwork and communication, through collaborative coding exercises and project-based learning. Ultimately, integrating this course into a broader curriculum not only enriches the academic experience but also equips students with the technical expertise and adaptability required to thrive in an evolving and technology-driven engineering landscape.

4. CONCLUSION

The proposed course, "Numerical Methods for Linear Systems with Applications in Python," exemplifies an innovative and integrative approach to engineering education. By combining mathematical theory with computational practice, the course bridges a critical gap, providing students with both the conceptual foundation and practical tools needed to solve complex engineering problems.

One of the key strengths of this approach lies in its dual focus. Numerical methods, such as Gaussian elimination, LU factorization, and the Jacobi method, are essential for understanding and solving linear systems encountered in various engineering fields. Integrating Python programming further enhances this understanding by offering hands-on experience with implementation, debugging, and optimization. Python's user-friendly nature and powerful libraries (NumPy, SciPy) make it an ideal tool for simplifying complex numerical computations and visualizing results.

This method fosters critical thinking, problem-solving, and collaboration, aligning with the interdisciplinary demands of modern engineering professions. Students not only learn to tackle theoretical challenges but also develop practical skills in coding and analysis, preparing them for industry roles and advanced research opportunities.

Overall, this course structure serves as a model for modern engineering education, promoting an active, applied learning environment. It equips students with skills that are highly relevant to both academic growth and the professional challenges of a digital, data-driven world.

Future development of the course can explore integrating artificial intelligence (AI) with numerical methods to enhance both learning outcomes and practical applications. AI, particularly machine learning, offers powerful tools for optimizing numerical algorithms. By incorporating AI techniques, students could explore hybrid approaches where traditional numerical methods are combined with neural networks to solve large-scale systems or approximate solutions to problems with high computational complexity. Further, students could engage in advanced projects, such as

developing AI algorithms to refine numerical solutions in fields like computational fluid dynamics or structural health monitoring. Expanding the course to include these topics not only reflects the evolving demands of the engineering profession but also prepares students to leverage cutting-edge technologies, positioning them as leaders in the interdisciplinary landscape of AI-enhanced engineering solutions.

5. ACKNOWLEDGEMENT

This paper was written using data and results obtained through the project "Performance and excellence in the field of environment and renewable energy through modern cluster-type entities" with the acronym PEDMEREMC, SMIS code 138692. The project was co-financed by the European Regional Development Fund through the Operational Program Competitiveness 2014 -2020.

6. REFERENCES

- [1] Lonsky, M., Lang, M., Holt, S., Pathak, S.A., Klause, R., Lo, T.H., Beg, M., Hoffmann, A. & Fangohr, H.: *Numerical simulation projects in micromagnetics with Jupyter*. American Journal of Physics, 92(10), pp.794-800, 2024.
- [2] Sachmpazidi, D., Bautista, M., Chajeki, Z., Mendoza, C., & Henderson, C.: *Integrating numerical modeling into an introductory physics laboratory*. American Journal of Physics, 89(7), 713-720, 2021
- [3] Reddy, J. N.: Numerical Methods, Finite Element. In *Encyclopedia of Solid Earth Geophysics* pp. 1145-1149. Cham: Springer International Publishing, 2021.
- [4] Ober-Blöbaum, S., Junge, O. & Marsden, J.E.: Discrete mechanics and optimal control: An analysis, *ESAIM: Control, Optimisation and Calculus of Variations*, 17(2), pp. 322–352, 2011.
- [5] Laurière, M., Perrin, S., Geist, M. & Pietquin: Learning mean field games: A survey. *arXiv preprint arXiv:2205.12944*, pp.19-49, 2022.
- [6] Chapman, S. J.: *MATLAB programming for engineers*. Nelson Education. 2015.
- [7] Mariș, S., Brăescu, L.: *Metode numerice – probleme de seminar și lucrări de laborator*, Editura UVT, Timișoara, 2008.
- [8] Reitz, K.: *Python guide documentation*. Release 0.01. Wilmington (DE): Python Software Foundation. 2017
- [9] NumPy Documentation: <https://numpy.org/doc/> August 2024.
- [10] SciPy Documentation: <https://docs.scipy.org/doc/scipy/> August 2024.
- [11] Python for Data Analysis by Wes McKinney: <https://www.oreilly.com/library/view/python-for/9781449323592/> August 2024.