International Journal on Multidisciplinary and Applied Research
https://doi.org/10.63236/ijmar.1.1.2
Date received: 22/02/25; revised: 11/04/25; accepted: 25/05/25; published: 01/06/25

Teaching Differential Equations to Mechanical Engineering Students using Excel VBA

Pankaj Dumka
Department of Mechanical Engineering, Jaypee University of Engineering and Technology, A.B. Road, Raghogarh-473226, Madhya Pradesh, India
https://orcid.org/0000-0001-5799-6468

Dhananjay R. Mishra
Department of Mechanical Engineering, Jaypee University of Engineering and Technology, A.B. Road, Raghogarh-473226, Madhya Pradesh, India
https://orcid.org/0000-0002-5107-0012

Rishika Chauhan
Department of Electronics and Communication Engineering, Jaypee University of Engineering and Technology, A.B. Road, Raghogarh-473226, Madhya Pradesh, India
https://orcid.org/0000-0001-8483-865X

Corresponding author: Pankaj Dumka, p.dumka.ipec@gmail.com

Abstract
The ability to solve differential equations is necessary for mechanical engineering students, as these equations model real-world engineering systems such as fluid flow, heat conduction, thermodynamics, and structural integrity. This aim of this study is to develop a complete teaching module by utilizing the Excel VBA to facilitate the numerical solution of differential equations in mechanical engineering. Unlike MATLAB or Python, Excel VBA provides an open platform for students with minimal coding experience, thus enabling them to understand numerical methods while working within a spreadsheet. The research follows the Design, Development, Implementation, and Evaluation (DDIE) framework to create an interactive and practical learning experience for the students. The study evaluates student engagement, understanding, and problem-solving ability by using both the qualitative and quantitative assessments. The module was tested in an undergraduate engineering classroom and demonstrated strong quantitative impact, receiving high validation scores from both the students and faculty members with a 35% average score improvement, 85% student preference for Excel VBA, and faculty reporting 30% increased confidence, 25% better problem-solving, 20% more experimentation, and 25% improved conceptual understanding. The results demonstrate that the Excel VBA

is an effective tool for teaching numerical methods by providing the students with an intuitive and interactive approach to solve differential equations.

Keywords: teaching module, differential equations, Excel VBA, engineering education, numerical methods

1. Introduction

In today's engineering education landscape, the integration of computational tools and numerical problem-solving techniques has become necessary. As curricula increasingly stress the practical problem-solving over rote analytical derivations, educators are challenged to adopt accessible and effective instructional tools that bridge theoretical understanding with real-world application. Differential equations serve as the basis for modelling numerous engineering phenomena which include fluid mechanics, thermodynamics, structural mechanics, and vibration analysis (Anderssen & Hegland, 1999; Chapra & Canale, 2010; Loyinmi & Akinfe, 2020; Muldowney, 1990). Conventionally, engineering students are introduced to solving these equations using analytical methods (Fusco et al., 2022; Kunkel & Mehrmann, 2006; Loyinmi & Akinfe, 2020; T. & Strikwerda, 1990). However, many practical problems involve nonlinearities or boundary conditions that make analytical solutions unavailable (Dumka et al., 2022). Numerical methods offer an alternate approach by enabling the students to approximate solutions effectively (Faires & Burden, 2003; Kreyszig, 2011).

While programming tools such as MATLAB (Kumar, 2016; Nikolic et al., 2018), Python (Ranjani et al., 2019; Van Der Walt et al., 2011), and Mathematica (Wijaya et al., 2021) are extensively used in academic circles, many students face a steep learning curve when adapting to these platforms. Studies such as those by Johns et al. (2023) and Inguva et al. (2021) show that the integration of MATLAB and Python enhances student understanding of numerical methods, especially for solving ordinary and partial differential equations. However, despite their advantages, these platforms assume a certain level of programming proficiency, which can be a hindrance for early-stage undergraduates or students from non-programming backgrounds. While these tools are powerful, they are often underutilized in undergraduate curricula owing to perceived complexity or lack of programming background among students. This creates a teaching gap—students are either overwhelmed by programming environments or are restricted to passive learning through static problem sets.

Excel VBA offers a spontaneous and familiar environment (Fellah, 2019; Musimbi & Mulanza, 2018), allowing the students to implement numerical methods without any need for widespread coding experience (Baliti et al., 2020; Coronell, 2005; El-Awad, 2015). Naseem et al. (2023) demonstrated that Excel VBA can be used effectively to implement numerical integration, root-finding algorithms, and solutions to initial and boundary value problems. The strength of VBA lies in bridging the gap between conceptual understanding and computational execution, particularly for students with minimal coding experience.

Interactive learning environments have been shown to enhance student engagement and retention of knowledge significantly. The DDIE (Design, Development, Implementation, and Evaluation) framework, as applied by Hidayat and Nizar (2021) in instructional design, offers a systematic approach to curriculum development. This framework aligns well with the iterative nature of engineering problem solving and has been used successfully in integrating computational tools into engineering courses.

Despite the pedagogical potential of Excel VBA, limited literature exists that formalizes its use in teaching differential equations in mechanical engineering curricula. Existing research primarily explores either high-end computational platforms or static spreadsheet methods, without leveraging the interactivity and customization potential of VBA. This gap underscores the novelty of the current study, which introduces a structured teaching module based on Excel VBA, supported by a robust instructional framework and real classroom implementation.

This study introduces an Excel VBA-based module designed to teach the mechanical engineering students how to solve ordinary differential equations (ODEs) using computational techniques. The study is guided by the following research questions:

- Can an Excel VBA-based module effectively support the teaching of ordinary differential equations to undergraduate mechanical engineering students?
- How does the DDIE instructional framework enhance learner engagement and conceptual understanding in a computationally driven environment?

This study presents the design, development, implementation, and evaluation of an interactive instructional module built using Excel VBA, aimed at helping mechanical engineering students solve ordinary differential equations using numerical techniques. The module is embedded within the DDIE pedagogical framework, offering an engaging and structured approach that aligns with students' learning curves and course objectives. The study not only validates the instructional design through classroom deployment but also identifies the strengths, limitations, and transferability of the approach for broader educational use.

2. Method

This research adopts the DDIE (Design, Development, Implementation, and Evaluation) model, a well-established instructional design framework, to systematically construct a robust and pedagogically sound learning module tailored for mechanical engineering students. The DDIE approach, as articulated in the works of Karajizadeh et al. (2023) and Seeto and Vlachopoulos (2015), provides a structured pathway to translate educational goals into effective classroom practices. Its application in STEM education has been shown to improve both the clarity of content delivery and the retention of complex concepts. In this study, the framework is further reinforced by insights from Ogegbo and Ramnarain (2022), emphasizing the need for student-centred learning tools that align with real-world applications.

The Design phase commenced with a thorough analysis of the learning objectives (Li & Sun, 2023), which were directly aligned with core topics in mechanical engineering where differential equations play a central role—such as transient heat conduction, fluid flow dynamics, and mechanical vibrations. Key numerical methods, including Euler's method, Runge-Kutta methods, and finite difference techniques, were selected for inclusion based on their relevance to the curriculum and feasibility for implementation in Excel VBA. Additionally, instructional materials were structured to build gradually from foundational concepts to more complex applications, enabling a scaffolded learning experience.

In the Development phase, the focus shifted to implementing these numerical algorithms using Excel's Visual Basic for Applications (VBA) environment (Rossi, 2021). Custom subroutines and user-defined functions were coded to simulate the numerical solution of ordinary differential equations (ODEs) and initial/boundary value problems. Effort was made to keep the VBA code readable and modifiable so that students with minimal programming background could engage with the logic, trace computations, and experiment with parameters. Graphical outputs and worksheet interactivity were incorporated to support visualization of results, thereby enhancing conceptual understanding.

The Implementation stage involved classroom execution of the developed module. Undergraduate mechanical engineering students were introduced to the Excel VBA tools through guided tutorials, live demonstrations, and problem-solving sessions. Students applied the tools to solve representative engineering problems, thereby reinforcing the connection between theory and practice. This experiential learning component enabled students to appreciate not only the computational approach but also the physical significance of differential equations in engineering analysis and design.

Finally, in the Evaluation phase, a mixed-methods approach was employed to assess the effectiveness of the learning module. Quantitative evaluation was conducted using pre-tests and post-tests to measure the knowledge gain and problem-solving ability of students. Qualitative data were collected through structured surveys and semi-structured interviews with faculty members and students, providing insights into user experience, engagement, and perceived value of the Excel VBA module. Feedback gathered from these assessments was analysed and used to refine the teaching material, improve user interface elements, and address common points of confusion. This continuous feedback loop ensured that the learning tool evolved into a more effective and student-friendly educational resource.

Overall, the integration of the DDIE model in this research facilitated the creation of a comprehensive, interactive, and accessible learning module that not only enhances student engagement but also bridges the gap between abstract numerical methods and their practical applications in mechanical engineering.

3. Implementation of Numerical Methods in Excel VBA

The module includes several numerical techniques to solve differential equations, each illustrated with real-world engineering applications. Detailed explanations, example problems, and VBA implementations help students grasp key concepts.

Euler Method (Devia et al., 2021; Ijaz Khan et al., 2023)
Euler's method is the easiest numerical approach for solving first-order ODEs of the form $\frac{dy}{dx} = f(x, y)$. Given an initial value y_0 at x_0, the next value is approximated using a small step size h as follows (Burden & Faires, 2010; Chapra & Canale, 2010; Kreyszig, 2011):

$$y_{n+1} = y_n + h \times f(x_n, y_n) \tag{1}$$

This method is explicit and straightforward but suffers from numerical uncertainty and low accuracy for difficult or quickly changing functions. The global error is proportional to $O(h)$, making it less appropriate for problems which require high precision. The following VBA script demonstrates how students can implement Euler's method in Excel VBA:

```
Sub euler()
a = InputBox("Enter a")
b = InputBox("Enter b")
h = InputBox("Enter h")
n = Int(1 + (b - a) / h)
ReDim x(n)
For i = 0 To n
      x(i) = a + i * h
Next i
ReDim y(n)
y(0) = InputBox("Enter y(0)")
For i = 0 To n - 1
   y(i + 1) = y(i) + dydx(x(i), y(i)) * h
   Cells(i + 2, 1) = i + 1
   Cells(i + 2, 2) = x(i)
   Cells(i + 2, 3) = ytrue(x(i))
   Cells(i + 2, 4) = y(i)
Next i
End Sub
```

Its subroutine requires the inputs for the range, step size, and initial condition. Then it computes and stores x and y values using the Euler formula. The results, including step number, x, exact solution (ytrue), and Euler approximation, are displayed in the worksheet.

Runge-Kutta Method (Koroche, 2021; Mechee & Aidi, 2022)
Runge-Kutta methods improve accuracy over Euler's method by considering intermediate points within each step to increase the accuracy. The most common form is the fourth-order Runge-Kutta method (RK4), which provides a good balance between accuracy and computational efficiency. The RK4 method computes the next value as (Burden & Faires, 2010; Chapra & Canale, 2010; Rabiei et al., 2023):

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2}$$

where,

$k_1 = f(x_n, y_n)$; $k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$; $k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$; $k_4 = f(x_n + h, y_n + hk_3)$

The RK4 method provides an error of order $O(h^4)$ which makes it more accurate than the Euler method while remaining computationally practical for a wide range of problems. The following VBA function implements the fourth-order Runge-Kutta (RK4) method:

```
Sub RK4()
   a = InputBox("Enter x(0)")
   b = InputBox("Enter x(n)")
   h = InputBox("Enter h")
   n = Int(1 + (b - a) / h)
   ReDim x(n)
   For i = 0 To n
         x(i) = a + i * h
   Next i
   ReDim y(n)
   y(0) = InputBox("Enter y(0)")
   col = InputBox("Enter colum to print the result")
   For i = 0 To n - 1
      k1 = dydx(x(i), y(i))
      k2 = dydx(x(i) + h / 2, y(i) + k1 * h / 2)
      k3 = dydx(x(i) + h / 2, y(i) + k2 * h / 2)
      k4 = dydx(x(i) + h, y(i) + k3 * h)
      Slope = (k1 + 2 * k2 + 2 * k3 + k4) / 6
      y(i + 1) = y(i) + Slope * h
      Cells(i + 2, 1) = i + 1
      Cells(i + 2, 2) = x(i)
      Cells(i + 2, 3) = ytrue(x(i))
      Cells(i + 2, Int(col)) = y(i)
   Next i
End Sub
```

Its subroutine requires the inputs for the range, step size, initial condition, and output column, then calculates the x and y values using RK4's weighted slope formula. The results, namely step number, x, exact solution, and RK4 approximation, are printed in the specified worksheet columns. The functions were demonstrated to solve the following equation:

$$\frac{dy}{dx} = -2y + 3x\ ;\ y(0) = 5 \tag{3}$$

The students were asked to formulate the function in VBA which appear as follows:

```
Function dydx(x, y)
   dydx = -2 * y + 3 * x
End Function
```

Then one by one they run the subroutines which ask about the domain, step size and initial value of y. Figures 1 and 2 show the solution of the problem and error variation for a step size (h) of 0.2.

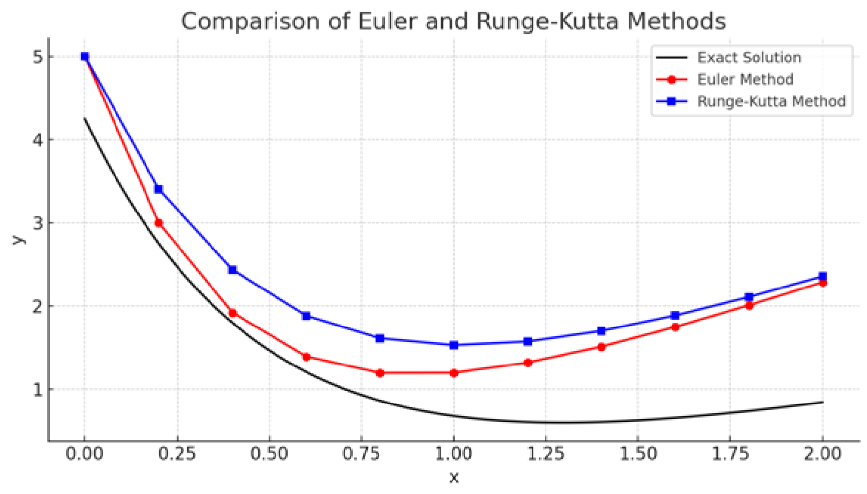

Figure 1 - Comparison of Euler and Runge-Kutta methods with the exact solution

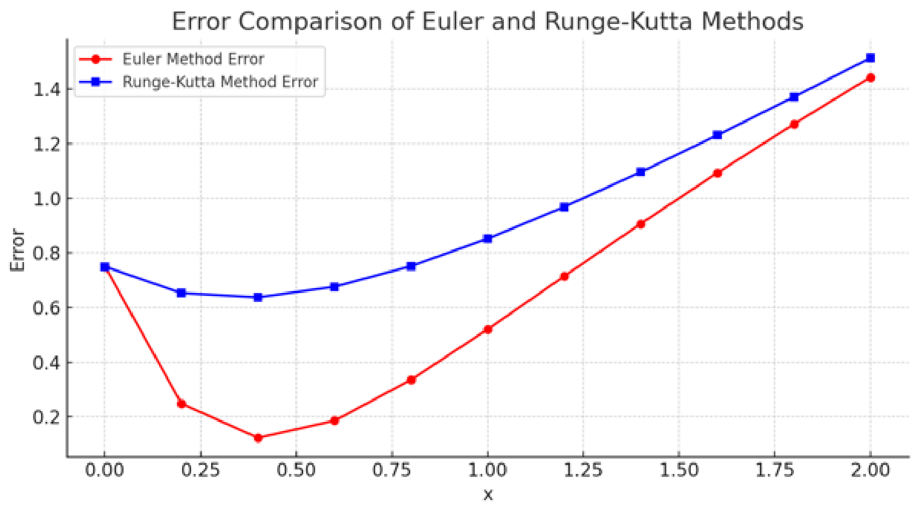

Figure 2 - Error comparison between Euler and Runge-Kutta methods

This graph displays the numerical solutions obtained using Euler's method and the Runge-Kutta (RK4) method, compared to the exact analytical solution. The exact solution (black curve) represents the true behaviour of the differential equation. The Euler method (red circles) shows significant deviations as x increases, while the Runge-Kutta method (blue squares) closely sees the exact solution, demonstrating its superior accuracy.

The error plot depicts the difference of each numerical method from the exact solution. The Euler method (red line) exhibits a growing error, especially at larger values of x, due to its lower order of accuracy. The Runge-Kutta method (blue line) has significantly smaller errors, showing only minimal deviation throughout the range. This demonstrates why RK4 is preferred for solving differential equations numerically.

Reasons for Deviations

- *Euler's Method: Accumulation of Truncation Errors*
  Euler's method uses a simple one-step forward approximation which leads to cumulative errors at each iteration. As the Euler's method only considers slope at the beginning of each interval, it fails to adjust for curvature in the actual solution which results in a rising deviation.
- *Runge-Kutta Method: Higher Order Accuracy*
  RK4 takes multiple intermediate calculations within each step, thus adjusting for the local curvature of the function. This approach minimizes errors by refining the estimate of the next point. This ensures much closer alignment with the exact solution.
- *Effect of Step Size*
  A smaller step size (h) would reduce errors in both methods; however, Euler's method would still be significantly less accurate than RK4. The Runge-Kutta method, even with a moderate step size, produces results that are nearly indistinguishable from the exact solution, making it more efficient.

These results confirm that while Euler's method provides a simple numerical approach, it is not suitable for high-accuracy applications owing to its large truncation errors. The Runge-Kutta method, on the other hand, offers significantly better precision with minimal additional computational effort, making it the preferred choice for solving differential equations in engineering applications.

4. Results and Discussion

Student performance was evaluated using pre-test and post-test scores. The average improvement in scores was 35%, showing a substantial improvement in problem-solving skills. Surveys showed that 85% of students found Excel VBA more useful than traditional programming languages. Faculty feedback emphasized that the module improved students' conceptual understanding of numerical methods. The study also found that the students have appreciated the interactive nature of the module and its incorporation into spreadsheet-based problem-solving.

Performance Metrics

The performance metrics presented in the Table 1 presents an in-depth look into how students understanding and interaction with differential equations improved after utilizing the Excel VBA module.

Table 1 - Performance metrics based on classroom performance

Metric	Pre-Test Score (%)	Post-Test Score (%)	Improvement (%)
Accuracy	59	94	35
Usability	65	90	25
Engagement	63	88	25

A paired t-test was conducted between the pre-test and post-test scores to ensure the statistical significance of the improvements. The results showed a p-value < 0.01, confirming that the gains were not due to random chance, but rather the impact of the Excel VBA module. The following observations are made:

Accuracy: The most substantial improvement was observed in the accuracy, which has increased by 35%. This suggests that students developed a stronger knowledge of numerical methods and were able to apply them effectively to solve differential equations. The structured step-by-step approach in VBA has helped in the reduction of errors and strengthening the understanding of ODEs.

Usability: Usability increased by 25%. Students found Excel VBA to be an accessible tool that enabled them to visualize the problem-solving process better. Unlike traditional methods, Excel VBA provided an interactive environment where students could manipulate values and observe the results immediately.

Engagement: The engagement of the students also increased by 25%, indicating that the students were more involved in learning activities. The hands-on experience of coding their own numerical solvers encouraged the active participation of students and boosted their problem-solving confidence.

Overall, the performance metrics highlight the success of integrating Excel VBA into the curriculum for teaching numerical solutions to differential equations. These improvements align with positive student feedback and faculty observations, reinforcing the module's educational value.

Student Feedback
A survey was conducted among students, the results of which are illustrated in Figure. 3. The students expressed positive feedback on the usability and effectiveness of the module.

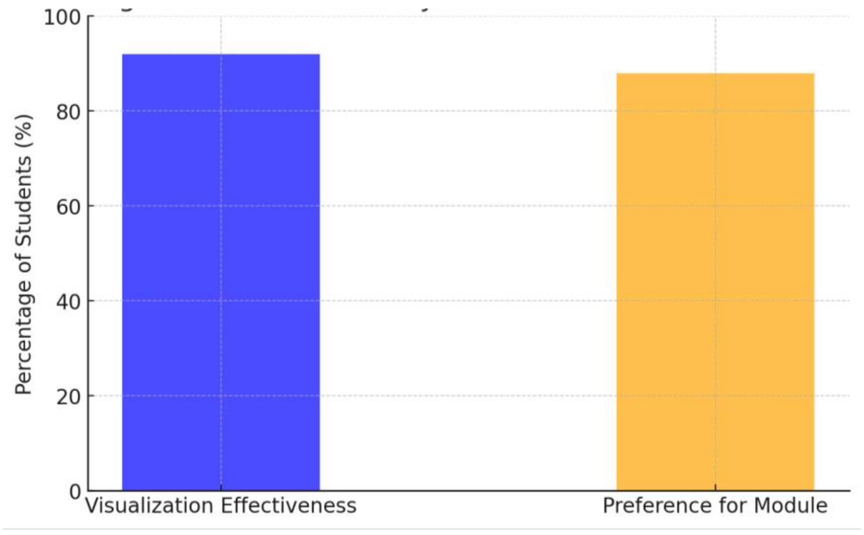

Figure 3 - Student satisfaction survey results

From the survey it has been observed that 92% of students have agreed that the Excel VBA helped them visualize numerical methods effectively. Furthermore, 88% have stated that they preferred the module over traditional classroom lectures owing to its interactive style. The module was applied across a variety of engineering problems, ranging from Newton's law of cooling to second-order dynamic systems such as mass-spring-damper models. This diversity enabled students to understand the wide applicability of numerical methods, reinforcing their problem-solving versatility.

While the module was largely successful, some students initially faced difficulties in understanding VBA syntax and logic structures. Additionally, challenges related to Excel version compatibility and macro settings were reported, especially when working across different systems. These issues were mitigated through peer discussions and instructor-led support sessions.

Faculty Observations
Instructors found that students demonstrated improved confidence in applying numerical techniques. The ability to modify VBA scripts enabled the students to experiment with various problem-solving approaches, thereby reinforcing their learning.

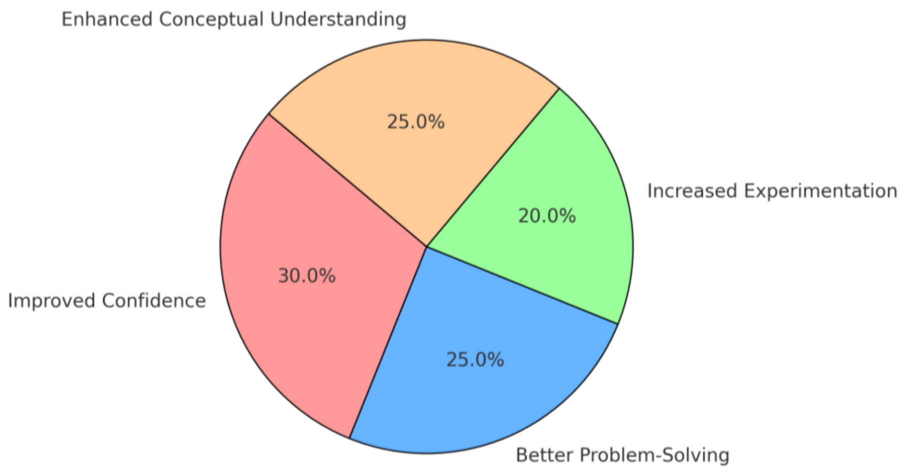

Figure 4 - Faculty observations on Excel VBA module

Faculty observations further highlighted the various ways in which students benefited from the Excel VBA module. As illustrated in Figure 4, approximately 30% of faculty members have noted a significant improvement in the students' confidence when tackling numerical problems. Additionally, 25% of instructors observed better problem-solving skills, as students were able to break down complex differential equations systematically. Another 20% of faculty members emphasized increased experimentation, with students modifying VBA scripts to test various numerical techniques. Finally, 25% of instructors reported enhanced conceptual understanding, as the module provided a structured and interactive approach to learning differential equations. These qualitative insights support the effectiveness of integrating Excel VBA into mechanical engineering education, making numerical methods more accessible and engaging.

The success of this Excel VBA-based approach underscores its potential to enhance learning in other computational subjects such as heat transfer, control systems, and even optimization. Future iterations of the module could integrate real-time error tracking, student assessment analytics, and simulation of partial differential equations. Additionally, its adaptability to online or hybrid classrooms makes it a sustainable pedagogical tool for modern engineering education.

5. Conclusion

This research article confirms that Excel VBA is a feasible tool for teaching the differential equations in mechanical engineering. Subroutines have been written in VBA to solve differential equations using Euler's and Runge-Kutta (fourth order) method. By providing an organized learning approach through the DDIE model, students gained hands-on experience in computational problem-solving. With a 35% average score improvement and 85% of students choosing Excel VBA over traditional languages, the module significantly enhanced problem-solving skills and usability. Faculty feedback highlighted a diverse student gain, namely 30% noted increased confidence, 25% observed improved problem-solving, 20% reported more experimentation, and 25% highlighted deeper conceptual understanding. These insights affirm that Excel VBA fosters an engaging, hands-on learning experience for mastering differential equations in mechanical engineering. Future research could focus on integrating more complex PDE solvers and expanding the module to include real-time data processing for engineering simulations. This will further enhance the students' understanding and applications of numerical methods to the real-world problems.

6. References

Anderssen, R., & Hegland, M. (1999). For numerical differentiation, dimensionality can be a blessing! *Mathematics of Computation*, *68*(227), 1121–1141. https://doi.org/10.1090/s0025-5718-99-01033-9

Burden, R. L., & Faires, J. D. (2010). *Numerical analysis*. Brooks Cole.

Chapra, S. C., & Canale, R. P. (2010). *Numerical methods for engineers* (5th ed.). The McGraw-Hill Companies.

Coronell, D. G. (2005). Computer science or spreadsheet engineering? An Excel/VBA-based programming and problem solving course. *Chemical Engineering Education*, *39*(2), 142–145.

Devia, D. M., Mesa, F., & Vélez, G. (2021). Comparative analysis of numerical solutions of ODEs with initial value problems using improved euler methods. *Scientia et Technica*, *26*(03), 391–397. https://doi.org/10.22517/23447214.24892

Dumka, P., Dumka, R., & Mishra, D. R. (2022). *Numerical methods using Python*. BlueRose.

El-Awad, M. M. (2015). A multi-substance add-in for the analyses of thermo-fluid systems using Microsoft Excel. *International Journal of Engineering and Applied Sciences*, *2*(3).

Faires, J. D., & Burden, R. L. (2003). *Numerical methods.* Thomson.

Fellah, G. (2019). Excel Spreadsheet as a tool for simulating the performance of steam power plants. *Spreadsheets in Education*, *12*(January), 1–19. https://sie.scholasticahq.com/article/7007-excel-spreadsheet-as-a-tool-for-simulating-the-performance-of-steam-power-plants

Fusco, N., Marcellini, P., & Sbordone, C. (2022). Ordinary differential equations. *UNITEXT - La Matematica per Il 3 Piu 2*, *137*(NeurIPS), 187–235. https://doi.org/10.1007/978-3-031-04151-8_4

Hidayat, F., & Nizar, M. (2021). Model ADDIE (Analysis, Design, Development, Implementation and Evaluation) Dalam Pembelajaran Pendidikan Agama Islam. *Jurnal Inovasi Pendidikan Agama Islam (JIPAI)*, *1*(1), 28–38. https://doi.org/10.15575/jipai.v1i1.11042

Ijaz Khan, M., Al-Khaled, K., Raza, A., Khan, S. U., Omar, J., & Galal, A. M. (2023). Mathematical and numerical model for the malaria transmission: Euler method scheme for a malarial model. *International Journal of Modern Physics B*, *37*(16), 2350158. https://doi.org/10.1142/S0217979223501588

Inguva, P., Bhute, V. J., Cheng, T. N. H., & Walker, P. J. (2021). Introducing students to research codes: A short course on solving partial differential equations in Python. *Education for Chemical Engineers*, *36*, 1–11. https://doi.org/10.1016/j.ece.2021.01.011

Johns, A. N., Hesketh, R. P., Stuber, M. D., & Ford Versypt, A. N. (2023). Numerical problem solving across the curriculum with Python and MATLAB using interactive coding templates: A workshop for chemical engineering faculty. *Proceedings of the ASEE Annual Conference and Exposition, Conference,*, January 2021. https://doi.org/10.18260/1-2--43749

Karajizadeh, M., Zand, F., Vazin, A., Saeidnia, H. R., Lund, B. D., Tummuru, S. P., & Sharifian, R. (2023). Design, development, implementation, and evaluation of a severe drug–drug interaction alert system in the ICU: An analysis of acceptance and override rates. *International Journal of Medical Informatics*, *177*(June), 105135. https://doi.org/10.1016/j.ijmedinf.2023.105135

Koroche, K. A. (2021). Numerical solution of first order ordinary differential equation by using Runge-Kutta method. *International Journal of Systems Science and Applied Mathematics*, *6*(1), 1. https://doi.org/10.11648/j.ijssam.20210601.11

Kreyszig, E. (2011). *Advanced engineering mathematics* (10th ed.). Wiley.

Kumar, R. (2016). Thermodynamic modeling and validation of a 210-MW capacity coal-fired power plant. *Iranian Journal of Science and Technology - Transactions of Mechanical Engineering*, *40*(3), 233–242. https://doi.org/10.1007/s40997-016-0025-5

Kunkel, P., & Mehrmann, V. (2006). *Differential-algebraic equations: Analysis and numerical solution*. European Mathematical Society. https://ems.press/books/etb/14

Li, R. Y., & Sun, J. C. Y. (2023). Identifying key factors of dynamic ADDIE model for instructional virtual reality design: An exploratory study. *Interactive Learning Environments*, *June*. https://doi.org/10.1080/10494820.2023.2296519

Loyinmi, A. C., & Akinfe, T. K. (2020). Exact solutions to the family of Fisher's reaction-diffusion equation using Elzaki homotopy transformation perturbation method. *Engineering Reports*, *2*(2), 1–32. https://doi.org/10.1002/eng2.12084

Mechee, M. S., & Aidi, S. H. (2022). Generalized Euler and Runge-Kutta methods for solving classes of fractional ordinary differential equations. *International Journal of Nonlinear Analysis and Applications*, *13*(May 2021), 1737–1745.

Muldowney, J. S. (1990). Compound matrices and ordinary differential equations. *Rocky Mountain Journal of Mathematics*, *20*(4), 857–872.

Musimbi, O., & Mulanza, J.P. (2018). Using Excel as a tool to teach manufacturing and heat transfer. *ASEE Zone IV Conference.* https://doi.org/10.18260/1-2--29631

Naseem, A., Rehman, M. A., Qureshi, S., & Ide, N. A. D. (2023). Graphical and numerical study of a newly developed root-finding algorithm and its engineering applications. *IEEE Access*, *11*(1), 2375–2383. https://doi.org/10.1109/ACCESS.2023.3234111

Nikolic, S., Ros, M., & Hastie, D. B. (2018). Teaching programming in common first year engineering: discipline insights applying a flipped learning problem-solving approach. *Australasian Journal of Engineering Education*, *23*(1), 3–14. https://doi.org/10.1080/22054952.2018.1507243

Ogegbo, A. A., & Ramnarain, U. (2022). A systematic review of computational thinking in science classrooms. *Studies in Science Education, 58*(2), 203–230. https://doi.org/10.1080/03057267.2021.1963580

Rabiei, F., Hamid, F. A., Rashidi, M. M., Ali, Z., Shah, K., Hosseini, K., & Khodadadi, T. (2023). Numerical simulation of fuzzy volterra integro-differential equation using improved Runge-Kutta method. *Journal of Applied and Computational Mechanics*, *9*(1), 72–82. https://doi.org/10.22055/jacm.2021.38381.3212

Ranjani, J., Sheela, A., & Pandi Meena, K. (2019). Combination of NumPy, SciPy and Matplotlib/Pylab - A good alternative methodology to MATLAB-A comparative analysis. *Proceedings of 1st International Conference on Innovations in Information and Communication Technology,* 1–5. https://doi.org/10.1109/ICIICT1.2019.8741475

Rossi, R. (2021). Data science education based on ADDIE model and the Edison Framework. *Proceedings - 2021 International Conference on Big Data Engineering and Education,* 40–45. https://doi.org/10.1109/BDEE52938.2021.00013

Seeto, D., & Vlachopoulos, P. (2015). Design develop implement (DDI)—A team-based approach to learning design. *THETA: The Higher Education Technology Agenda*,11–13. papers3://publication/uuid/5E923ADF-C890-4A11-9D96-2AB8BC6DBE3A

T., V., & Strikwerda, J. C. (1990). Finite difference schemes and partial differential equations. In *Mathematics of Computation, 55*(192). SIAM. https://doi.org/10.2307/2008454

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Wijaya, T. T., Zhou, Y., Ware, A., & Hermita, N. (2021). Improving the creative thinking skills of the next generation of mathematics teachers using dynamic mathematics software. *International Journal of Emerging Technologies in Learning*, *16*(13), 212–226. https://doi.org/10.3991/ijet.v16i13.21535

This paper may be cited as:

Dumka, P., Dhananjay, R. M., & Chauhan. R. (2025). Teaching Differential Equations to Mechanical Engineering Students using Excel VBA. *International Journal on Multidisciplinary and Applied Researc*h, *1*(2), 1-13. https://doi.org/10.63236/ijmar.1.1.2