Board 134: MATLAB Integration in Sophomore Mathematical Analysis Course

Dr. Djedjiga Belfadel, Fairfield University

Djedjiga Belfadel is an Associate Professor in the Electrical and Bio Engineering department at Fairfield University. She obtained her Ph.D. degree from University of Connecticut in 2015, in electrical engineering. Her interests include embedded system

MATLAB Integration in Sophomore Mathematical Analysis Course

Abstract:

This Evidence-based Practice Paper describes the addition of MATLAB programming language to the Mathematical Analysis course for sophomore engineering students at Fairfield University majoring in electrical, mechanical, and biomedical engineering. Previously, MATLAB was taught in later years of the engineering program. Now, it is introduced in the first-year course, “Fundamentals of Engineering”, and then fully integrated into the sophomore-level “Mathematical Analysis” course. This redesign prepares students for their future courses. MATLAB provides a diverse array of application-specific functions, graphical user interfaces, debugging tools, and facilitates advanced visualization and matrix manipulation. Additionally, it includes App Designer and Simulink providing essential programming tools for developing coding skills and solving complex engineering problems.

The 3-credit Mathematical Analysis course, held twice a week, provides engineering students with essential numerical methods concepts relevant to a wide range of engineering fields. Designed to foster these skills within an interactive learning environment. During the first half of the semester, students are introduced to basic programming concepts including variables, conditional statements, loops, live scripts as well as the creation and execution of user-defined functions, symbolics, and plotting techniques. In the second half, the curriculum progresses into mathematical topics such as numerical solutions for nonlinear equations, analysis of linear systems, and engineering-centric statistics and probability models. This is followed by a focus on numerical differentiation, integration techniques, and data analysis methodologies like curve fitting, linear regression, and interpolation.

The course highlights the significance of coding skills and encourages active student engagement. Through a structured interactive lecture format, each session begins by introducing fundamental mathematical concepts, moves on to numerical exercises, and concludes with a practical MATLAB-programming session. This methodology seamlessly integrates theoretical understanding with hands-on application.

A distinctive feature of the course is its weekly Peer Learning Group (PLG) sessions, designed as programming workshops to offer students extra practice beyond regular lectures. Each week, students engage in preparatory worksheets before class and programming assignments aligned with lectures and PLG sessions. Additionally, the curriculum integrates online self-paced training auto-graded modules from the MathWorks platform.

The course's unique design, combining MATLAB programming with mathematical analysis in a condensed half-semester each presents both advantages and challenges. It offers an efficient way to cover essential topics rapidly, emphasizing the practical application of programming to mathematical concepts. This approach necessitates precise curriculum planning to ensure content is both deep and manageable within the limited time, requiring strategies that maximize learning outcomes and thoughtfully designed assessments to accurately gauge students' proficiency in both areas.

Analytical data from assignment evaluations and student feedback indicate that integrating MATLAB into the mathematical analysis course effectively develops sophomore students' programming skills.

1. Introduction:

The integration of computer programming in engineering education has become increasingly essential, especially in the sophomore year when students are expected to tackle more complex engineering problems. Recognizing this need, most engineering curricula require a computer programming course, often taught using traditional languages like Python, C, or JAVA. While these languages have their merits, their complexity can be a barrier for students who are still developing their engineering problem-solving skills [1,6]. This challenge becomes even more pronounced when students lack a solid understanding of mathematics and engineering principles [2].

The author's redesign of the mathematical analysis course to include MATLAB addresses an educational gap, leveraging MATLAB's strong reputation in engineering and its ranking among the top technical skills sought by employers. The initiative was shaped through consultations with the chairs from the Electrical, Mechanical, and Bioengineering departments and curriculum committees of the school. This updated engineering course supersedes the former mathematical analysis that solely focused on mathematical skills and is distinct from computer science programming offerings.

MATLAB popularity stems from its user-friendly interface and powerful computational abilities, making it approachable for programming novices. In contrast to conventional programming languages, MATLAB offers a wide range of specialized functions, graphical user interfaces, debugging tools, and supports enhanced visualization and matrix manipulation. These capabilities not only make programming easier but also promote the advancement of mathematical skills essential for solving engineering problems [3].

Originally, MATLAB introduction occurred in the later stages of the engineering program at Fairfield University. Acknowledging the value of early engagement with practical coding abilities, the author has now implemented MATLAB teaching in the freshman course, "Fundamentals of Engineering", and subsequently fully integrated it into the sophomore-level Mathematical Analysis course. This adjustment guarantees that students acquire essential computational tools at an early stage in their educational path, thereby enhancing their preparedness for subsequent coursework [4].

The Mathematical Analysis course is designed to be highly interactive and engaging [5]. It aims to introduce students to essential numerical methods and mathematical concepts, which are pivotal across various engineering disciplines. The course begins with fundamental programming concepts like variables, conditionals, loops, user-defined functions, and plotting. It then advances to topics such as numerical solutions for nonlinear equations, linear system analysis, and statistical models pertinent to engineering as the semester unfolds.

This course stands out due to its inclusion of weekly 75-minute Peer Learning Group (PLG) sessions. These workshops, led by a teaching assistant, offer hands-on programming practice beyond lectures, reinforcing core concepts. The PLG is a non-credit corequisite, taught by a proficient former student, with all materials provided by the faculty. There is no direct grade assigned to the PLG because students are completing their Programming assignments during the PLG. The focus is to give students confidence to start writing code from scratch and let them develop their own programming style.

In addition to the regular coursework, the curriculum is enriched with challenges and modules from the MathWorks platform, such as ONRAMP and fundamentals of programming. These resources are tailored to the individual majors of the students, providing them with a personalized learning experience that is both relevant and challenging.

2. **Course Design Challenges:**

Integrating MATLAB into the Mathematical Analysis course creates a unique mix of computational skills and mathematical theory. While this approach offers substantial benefits, it also introduces challenges that require thoughtful attention:

- Increased Learning Curve: Introducing MATLAB concurrently with complex mathematical concepts might overwhelm students, especially those with minimal programming background or who are struggling with the mathematical content. This compounded focus has the potential to impact student confidence and engagement negatively.
- Curriculum Depth vs. Breadth: Incorporating MATLAB sessions into an already dense curriculum necessitates a trade-off between the depth and breadth of mathematical topics covered. This compromise may reduce the mathematical rigor of the course, potentially diminishing students' comprehensive understanding of crucial concepts.
- Standardization and Flexibility: Choosing MATLAB as the primary computational tool ensures a uniform learning experience. However, this decision could limit students' exposure to a wider variety of programming languages and tools, which hold significant value in the broad spectrum of engineering fields. Such a restriction could impair students' ability to approach engineering challenges in various sectors with flexibility or to adapt to new technologies.
- Assessment and Evaluation: Evaluating students' competencies in both programming and mathematical analysis introduces a layer of complexity to the grading process. It may become challenging for educators to discern if students' difficulties arise from programming, understanding mathematical principles, or integrating both. This complexity risks masking the true nature of students' learning challenges, potentially complicating the provision of effective support.

3. **Course Overview:**

The "Mathematical Analysis" course at Fairfield University, designed to teach math and programming skills, benefits from a small class size of around 15 students. This intimate setting ensures personalized instruction and direct interaction between teachers and students.

The course is divided into two main segments, starting with an introduction to essential programming concepts. This phase covers MATLAB basics, vector and matrix operations, control structures, advanced functions, symbolic computing, and plotting techniques.

To maximize classroom time for hands-on programming and problem-solving, pre-class videos and readings are provided. Each topic typically has one or two short videos, totaling about 15 minutes, and an initial weekly assignment worksheet to be completed before the lecture. By employing a 'flipped' classroom approach, the course shifts focus on practical examples and in-class exercises, reducing the emphasis on extensive theoretical lectures.

During lecture sessions, each topic is succinctly introduced with slides that cover its syntax, supported by a series of MATLAB examples. The learning process is structured as follows: the professor demonstrates the first example, the second is a joint effort guided by student contributions, and the third is an individual task for students, with the professor and teaching assistants available for help. A sample example is provided in appendix 1. Advanced students frequently assist their peers, fostering a cooperative and lively classroom environment. Each session wraps up with group discussions and the revelation of solutions for individual challenges.

Each week, students are assigned 2 HomeWorks: an initial weekly assignment - a preparatory worksheet to be completed before the lecture, graded mainly for completion, using pre-class videos and readings provided; and a second main assignment - programming assignments aligned with lectures and PLG sessions, due at the end of the second lecture and PLG session of that week and graded by teaching assistants. Additionally, the curriculum integrates online self-paced training auto-graded modules from the MathWorks platform.

The initial phase of the course is structured to provide students with a comprehensive understanding of MATLAB and its application in engineering, detailed as follows:

- MATLAB Basics: Introducing the interface, commands, and workflow of MATLAB.
- Vectors and Matrices: Exploring the creation, manipulation, and application of vectors and matrices.
- Scripts & Functions: Learning to write live-scripts and functions for task automation and modular programming.
- Control Flow (Conditional Statements): Implementing conditional logic to make decisions within programs.
- Control Flow (Loops): Using loops to repeat operations and process data efficiently.
- Advanced Functions: Delving into more complex functions to solve intricate problems.
- Symbolic Operations & Plotting: Conducting symbolic calculations and visualizing data through various plotting techniques.
- App Design: Creating graphical user interfaces (GUIs) for applications within MATLAB.

Starting right after the midterm, the course shifts into its second phase, delving into more complex topics in mathematical analysis, including Root Finding & Optimization, Linear Algebra, Statistics, Curve Fitting, Interpolation, and Numerical Integration & Differentiation. It's important to note that students are not expected to master these advanced topics on their own outside of class. The course is designed to first lay down the theoretical groundwork and tackle

numerical problem-solving manually in the initial weekly session. The following session then focuses on algorithm development using MATLAB, marking a key transition from theoretical concepts to practical computational applications. This phase encourages students to move from manual problem-solving towards leveraging computational methods, guiding them through the conceptualization and execution of computer algorithms to address problems. A sample final exam is provided in appendix 2.

The second phase of the course is designed to equip students with a good understanding of comprehensive mathematical concepts, as detailed below:

- Measuring Errors: True vs. Approximate and Absolute vs. Relative.
- Root Finding & Optimization: Techniques for identifying solutions to equations and optimizing mathematical models using the bisection method, secant method, Newton-Raphson method, and false position method.
- Linear Algebra: Exploration of vector spaces, matrices, linear transformations, and solving a system of linear equations using the Gauss Elimination method.
- Integration: Fundamental methods for calculating integrals, including the trapezoidal and Simpson's rule.
- Differentiation: Fundamental methods for calculating derivatives, including forward, backward, and central differences.
- Data Analysis & Statistics: Introduction to statistical methods and data analysis techniques for interpreting and understanding data.
- Linear Interpolation: Techniques such as Lagrange and Newton interpolation for estimating values within a set of points, and the introduction of spline methods.
- Linear Regression: Strategies for estimating unknown values and analyzing the relationship between variables, including methods for assessing the goodness of fit.

4. Course Outcome:

[ ] course outcome link to the Blooms Taxonomy levels goal ( ) link to ABET student outcomes

1. Show proficiency in MATLAB including the understanding of the workspace, using m-files, graphics and plotting, and vector manipulation. [I] (1)
2. Demonstrate mastery of mathematical, numerical, and statistical engineering topics such as matrix algebra, data analysis and statistics, data interpolation, curve fitting, integration, and differentiation [II] (2)
3. Identify how programming and mathematical content applies to the field of engineering and understand the impact of engineering solutions in a global economic, environmental, and societal context. [I,II] (4) knowledge

This course supports ABET Student Outcomes: (1, 2, 4)

1. ABET 1 an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics.
2. ABET 2 an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors.

3. ABET 4 an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts.

5. Grade Distribution:

- Participation 5%
- Quizzes 10%
- Mathworks Certificates 10%
- Homework (WS) 10%
- Programing Assignments (PA) 20%
- Midterm Exam 20%
- Final Exam 25%

6. MATLAB Online Training and Its Implementation:

As a supplement to in-class and PLG sessions, students were required to engage with MATLAB's online training platforms. These platforms offer a variety of hands-on, self-paced, and free online training courses, including ONRAMP, MATLAB Fundamentals, App Building, Linear Algebra, and a module specific to each student's major.

- ONRAMP Course: This introductory course, taking approximately two hours, covers eleven main topics, beginning with fundamental MATLAB commands and progressing to data importation and visualization. It culminates in a final project where students apply their skills to analyze astronomical data, learning the practical application of concepts from theory to implementation.
- MATLAB Fundamentals: This course offers a comprehensive dive into MATLAB programming, covering a wide array of topics from data analysis to visualization. It's ideal for students looking to expand their knowledge after completing the ONRAMP course.
- App Building with MATLAB: For those interested in applying MATLAB to mathematics teaching, courses on app building in MATLAB are available. These courses focus on creating graphical user interfaces/apps, serving as practical tools for interactive teaching.
- Additional MATLAB Courses: Following ONRAMP, students are encouraged to explore further courses available on the MATLAB academy training portal. These courses, such as MATLAB for Data Processing and Visualization, MATLAB Programming Techniques, and Simulink ONRAMP, offer more in-depth knowledge and specialized skills.
- Certificates and Incentives: Upon completing these modules, students receive certificates from MathWorks.

Topic Description of the ONRAMP

Topic	Description
Commands	Enter commands in MATLAB to perform calculations and create variables
Vectors and Matrices	Create MATLAB variables that contain multiple elements
Importing Data	Bring data from external files into MATLAB
Indexing into and Modifying Arrays	Use indexing to extract and modify MATLAB arrays
Array Calculations	Perform calculations on entire arrays at once
Calling Functions	Call functions to obtain multiple outputs
Obtaining Help	Use the MATLAB documentation to discover information about MATLAB features
Plotting Data	Visualize variables using MATLAB's plotting functions
MATLAB Scripts	Write programs in script
Logical Arrays	Use logical expressions in MATLAB
Final Project	Bring together the introduced concepts with a project

Table I: Main topics of MATLAB Onramp training

7. Student Feedback

In Spring 2023 data was collected via Blackboard from all students (30). The course survey highlighted several positive aspects and areas for improvement, summarized as follows:

- Positive Aspects:
    - The practical programming assignments and PLG sessions were highly valued by students, indicating the effectiveness of hands-on learning and collaborative environments in enhancing understanding and engagement with the course material.
- Understanding and Participation:
    - 100% of students understand that survey participation won't affect their grade and confirm their voluntary participation, indicating clear communication of survey purposes and voluntary nature.

- PLG Sessions Helpfulness:
    - A significant majority (86.666%) found PLG sessions beneficial for grasping the course's objectives, suggesting the need to maintain these sessions while incorporating more practical examples and guided exercises.

- Impact of MATLAB on Interest:
    - About two-thirds of students (66.666%) reported that working with MATLAB heightened their interest in programming, showcasing the positive impact of integrating software tools into the curriculum.
- Suggestions for Course Improvement:
    - Students recommended more in-class examples, and a better balance of coursework to ensure a manageable workload.
- MathWorks Training Feedback:
    - Feedback on MathWorks training was mixed, with some finding it overly lengthy while others appreciated its content, indicating a need to evaluate and potentially adjust this component to maximize its benefit without causing undue burden.

All students' feedback in addition to the faculty's observations, will be considered for the improvement of next year's course.

8. Conclusion

The integration of MATLAB programming into the Mathematical Analysis curriculum at Fairfield University significantly benefits engineering students by bridging theoretical mathematics with practical application. This method enriches students' comprehension of mathematical concepts via interactive visualization and hands-on practice, fostering essential skills such as problem-solving, computational thinking, and teamwork. The positive outcomes of this approach, supported by student feedback and enhanced programming skills, underscore the importance of combining practical programming abilities with a strong mathematical foundation to improve overall student outcomes and readiness for future challenges. Although this model has proven effective in smaller class settings, adapting it for larger classes involves adopting a flipped classroom model where students engage with introductory materials before class and focus on active MATLAB application during class time. Creating a collaborative environment with structured group work and peer feedback, along with leveraging teaching assistants for additional support and grading, further enhances learning. Integrating MathWorks auto-graded programming workshops and requiring completion certificates for these sessions ensures active participation and solidifies learning, making this comprehensive approach conducive to fostering active learning and supporting individual advancement in larger classroom contexts.

References

[1] P. T. Goeser, W. Johnson, S. L. Bernadin, and D. A. Gajdosik-Nivens, "Work-in-Progress: The Impact of MatLab Marina - A Virtual Learning Environment on Student Learning in a Computing for Engineers Course", ASEE Annual Conference and Exposition, 2013.

[2] R. Talbert, "Learning MATLAB in the Inverted Classroom", ASEE Annual Conference and Exposition, 2012.

[3] K. Larsen, A. Hossain And M. Weiser, “Teaching an Undergraduate Introductory MATLAB Course: Successful Implementation for Students Learning”. ASEE Annual Conference and Exposition, 2016.

[4] D. Belfadel, M. Zabinksi and I. Macwan. *Introduction to MATLAB Programming in a Fundamentals of Engineering Course,* ASEE: Annual Conference and Exposition, Long Beach, California, July 2021.

[5] D. Belfadel, M. Zabinksi and R. Munden. *Walking on Water Term Design Project in Fundamentals of Engineering*. ASEE: Annual Conference and Exposition, Montreal, Canada. June 2020.

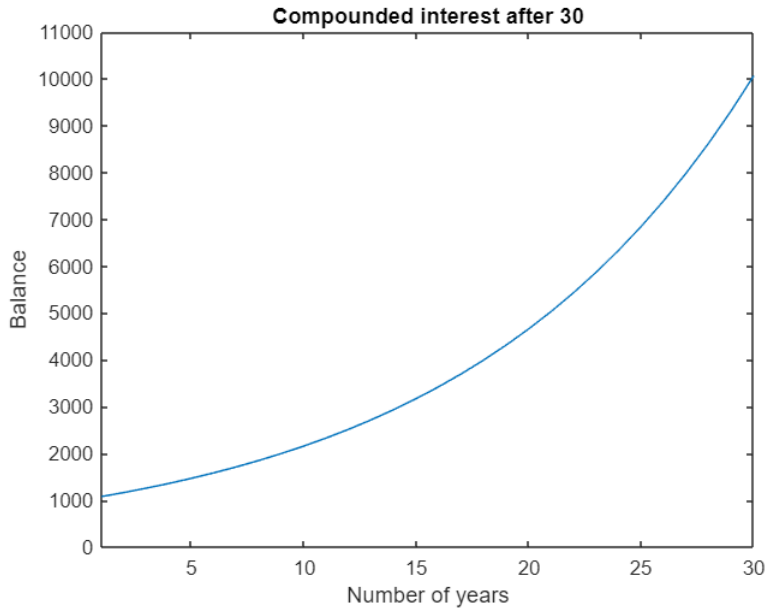
[6] D. Belfadel, M. Arambulo, M. Zabinksi, R. Munden, and James Cavallo. “Use of the Arduino Platform in Fundamentals of Engineering.” ASEE: Annual Conference and Exposition, Tampa, Florida. June 2019.

Appendix 1: Sample in class code "Example of Interest Calculation"

```
format bank
ibalance=1000;
ny=30;
rate=0.08;
for i=1:ny
   ibalance=ibalance+rate*ibalance;
   bal(i)=ibalance;
end
fprintf('Balance after %d years is: %.2f dollars.\n', ny,ibalance)
```

Balance after 30 years is: 10062.66 dollars.

```
plot(1:30,bal)
title('Compounded interest after 30')
xlabel('Number of years')
ylabel('Balance')
axis([1 30 0 11000])
```



Appendix 2: Sample Final Exam (take home)

Problem 01:

The horizontal velocity of a body is given as a function of time by

$$v(t) = t^2 + 4t - 2$$

where t is given in seconds, and v is given in m/s.

Question 1: (5 pts)
Determine the value of the acceleration at $t = 6$ seconds using forward divided difference.

Question 2: (5 pts)
Determine the value of the acceleration at $t = 8$ seconds using backward divided difference.

Question 3: (5 pts)
Determine the value of the acceleration at t=10 seconds using central divided difference

Question 4: (5 pts)
Write your own Matlab function to validate your results (all 3 methods: forward, backward and central).

Question 5: (5 pts)
Compare the results of question 4 to the results of question 1, 2 and 3.
what is your conclusion?

Problem 02:

The following data of the velocity of a body is given as a function of time.

Time (s)	0	15	28	32	54
Velocity (m/s)	0	34	47	85	173

Question 1: (5 pts)
Determine the value of the velocity at $t = 29$ seconds with first order polynomial interpolation using direct polynomial interpolation.
Question 2: (5 pts)
Determine the value of the velocity at $t = 29$ seconds with first order polynomial interpolation using Newton polynomial interpolation.
Question 3: (5 pts)
Determine the value of the velocity at $t = 29$ seconds with first order polynomial interpolation using Lagrange polynomial interpolation.
Question 4: (5 pts)
Use Matlab (built in function) to calculate the velocity at $t = 29$ seconds.
Question 5: (5 pts)
Discuss your results.

Problem 3:

Determine the root (highest positive) of:

F(x)= x^3+5.9*x.^2+10.9*x-2

Note: Remember to compute the error Epsilon-a after each iteration.

Use epsilon_s=0.01%.

Question1: (10 pts)
Perform (hand calculation) 4 iterations of Newton's Raphson method to solve the equation.
Use an initial guess of x0=1.

Question 2: (10 pts)

Write your own Matlab function to validate your results and plot the error.

Question 3: (5pts)

Compare the results of question 1 to the results of question 2, what is your conclusion?

Problem 4:

f(x) = x^3-5

Question 1: (10pts)

Perform 4 iterations of the bisection method, using initial guesses xl=-2, and Xu=4.

Question 2: (10pts)

Write your own Matlab function to validate your results and plot the error.

Question 3: (5pts)

Compare the results of question 1 to the results of question 2, what is your conclusion?

Note: Remember to compute the error Epsilon-a after each iteration (starting from iteration number 2).

Use epsilon_s=0.01%.

Problem 05

Given the system of equations:

x_1+2 x_2-x_3 =10
-3x_1-3 x_2 + 2 x_3=-10
x_1+4x_2 + 5 x_3 =-10

Question 1: (20 pts)

Use 2 different ways (methods) in MATLAB to Solve the system.
Show the code and the outputs.

Question 2: (5pts)

Is the solution unique? Explain why?