4. Журавлев А. Л., Нестик Т. А. Социально-психологические последствия внедрения новых технологий: перспективные направления исследований // Психологический журнал. 2019. Т. 40. № 5. С. 35-47. DOI 10.31857/S020595920006074-7

5. Цифровые ресурсы как средство поддержки студентов с ограниченными возможностями здоровья / Л. П. Коннова, В. А. Липатов, К. К. Сирбиладзе, И. К. Степанян // Вестник Нижегородского университета им. Н.И. Лобачевского. Серия: Социальные науки. – 2022. – № 2(66). – С. 59-64. – DOI 10.52452/18115942_2022_2_59.

TEACHING SOLVING MATHEMATICAL PROBLEMS USING MATLAB

Kurbonov E., Rakhimov N., Jurayev S., Kupaysinov S.

Tashkent University of Architecture and Construction, Tashkent city

Abstract. This article explores the effectiveness of using MATLAB as a tool for solving mathematical problems. MATLAB, renowned for its versatility and robust numerical capabilities, serves as a powerful platform for mathematical computation, analysis, and visualization. The article begins by introducing the fundamental concepts of MATLAB programming, including variables, arrays, functions, and control structures. It then delves into problem-solving approaches such as decomposition, algorithm design, and iterative refinement, illustrating their application through practical examples across various mathematical domains.

Key words. MATLAB, mathematical computation, problem-solving, linear algebra, calculus, differential equations, optimization, statistical analysis, programming, debugging, error handling, versatility, efficiency, interactivity, visualization, integration, learning, exploration.

Introduction. MATLAB, short for "MATrix LABoratory," stands as a cornerstone in the realm of computational mathematics and engineering. Developed by MathWorks, MATLAB serves as a robust platform for numerical computation, data analysis, visualization, and algorithm development. Its versatility and wide-ranging applications make it indispensable in academic research, industrial innovation, and beyond. At its core, MATLAB offers a rich environment where users can perform a myriad of mathematical operations with ease and efficiency. From basic arithmetic calculations to advanced numerical simulations, MATLAB provides a comprehensive suite of tools tailored to meet the demands of modern mathematics and engineering disciplines. What sets MATLAB apart is its intuitive syntax and extensive library of built-in functions, allowing users to express complex mathematical concepts in a concise and readable manner. Whether tackling linear algebra, differential equations, optimization problems, or statistical analysis, MATLAB provides a powerful framework for problem-solving across diverse domains. Furthermore, MATLAB's interactive environment facilitates rapid prototyping and experimentation, enabling users to iterate on ideas seamlessly. Its integration with Simulink, a graphical environment for modeling and simulating dynamic systems, further expands its utility in engineering design and simulation tasks.

In this article, we will explore the fundamentals of MATLAB and delve into its application in solving various mathematical problems. Through examples, tutorials, and practical insights, readers will gain a deeper understanding of how MATLAB can revolutionize their approach to mathematical computation and analysis. Whether you're a seasoned researcher, an aspiring engineer, or a student delving into the intricacies of mathematics, MATLAB offers a powerful toolkit to unlock new possibilities and drive innovation forward.

Problem-Solving Approaches**:** In MATLAB, problem-solving involves translating mathematical concepts into executable code to analyze data, simulate systems, or derive solutions to complex problems. Effective problem-solving in MATLAB relies on employing structured approaches to tackle mathematical challenges efficiently. Here are some key problem-solving approaches:

1. Decomposition: Break down complex problems into smaller, more manageable tasks or sub-problems. Identify the key components or steps required to solve each sub-problem. Implement and test solutions for each sub-problem independently before integrating them into a comprehensive solution.

2. Algorithm Design: Design algorithms that outline the steps necessary to solve a particular problem. Specify the input parameters, processing steps, and output requirements for the algorithm. Choose appropriate data structures and algorithms to optimize performance and accuracy.

3. Iterative Refinement: Start with a simple solution or initial implementation of the problem-solving approach. Test the solution using sample data or known solutions to identify errors or areas for improvement. Refine the solution iteratively by addressing identified issues and optimizing performance.

4. Modularization: Encapsulate reusable code segments into functions or scripts to promote code modularity and reusability. Divide the problem-solving process into smaller, self-contained modules with well-defined interfaces. Modularization facilitates code organization, maintenance, and collaboration among multiple developers.

5. Vectorization: Leverage MATLAB's vectorized operations to perform computations efficiently on arrays and matrices. Avoid unnecessary loops by expressing operations as matrix operations whenever possible. Vectorization improves code readability and performance by taking advantage of MATLAB's optimized numerical libraries.

6. Optimization Techniques: Explore optimization algorithms available in MATLAB to solve mathematical optimization problems. Choose appropriate optimization techniques based on the problem characteristics, such as linear programming, nonlinear optimization, or constrained optimization. Fine-tune algorithm parameters and constraints to achieve desired optimization outcomes.

7. Visualization and Interpretation: Utilize MATLAB's plotting and visualization capabilities to analyze and interpret mathematical results. Generate plots, graphs, and visualizations to illustrate relationships, trends, and patterns in the data. Customize plots with labels, titles, legends, and annotations to enhance clarity and insight. By applying these problem-solving approaches in MATLAB, users can effectively tackle a wide range of mathematical problems, from simple calculations to complex simulations and optimization tasks. In the subsequent sections, we'll illustrate these approaches through practical examples and demonstrate how MATLAB's features can be leveraged to address real-world mathematical challenges.

***Debugging and Error Handling in MATLAB*:** While writing MATLAB code for solving mathematical problems, it's common to encounter errors or unexpected behavior. Understanding how to effectively debug code and handle errors is essential for ensuring the correctness and robustness of your solutions. In this section, we'll discuss debugging techniques and error handling strategies in MATLAB.

Error Handling Strategies: Try-Catch Blocks: Use try-catch blocks to handle exceptions gracefully. Place the code that might generate an error within a try block and specify how to handle the error in the catch block. This prevents the entire program from crashing if an error occurs.

```
try
   % Code that might generate an error
catch ME
   % Handle the error
   disp(['Error occurred: ', ME.message]);
```

```
end
```

Debugging and Profiling Tools: MATLAB Debugger: Use the MATLAB debugger to step through your code line by line, inspect variable values, and identify errors. Set breakpoints at specific lines of code to pause execution and examine the program state.

Profiler: Utilize MATLAB's built-in profiler to identify performance bottlenecks and optimize your code. The profiler provides insights into function execution times, memory usage, and function call hierarchies, helping you optimize code efficiency.

By employing these debugging and error handling techniques in MATLAB, you can effectively identify and resolve issues in your mathematical code, ensuring its correctness and reliability. Debugging tools, error handling strategies, and profiler insights empower you to develop robust and efficient MATLAB solutions for a wide range of mathematical problems.

Conclusion. In this article, we explored the use of MATLAB as a powerful tool for solving mathematical problems across diverse domains. We covered fundamental concepts such as variables, arrays, functions, and control structures, which form the basis of MATLAB programming. We then discussed problem-solving approaches, including decomposition, algorithm design, and iterative refinement, which help tackle complex mathematical challenges effectively. In addition, we highlighted debugging techniques and error handling strategies to ensure the correctness and reliability of MATLAB code. Debugging tools, error handling mechanisms, and profiling utilities empower users to identify and resolve issues efficiently, enhancing the robustness of their mathematical solutions.

Efficiency: MATLAB offers optimized numerical algorithms and vectorized operations, enabling efficient computation of mathematical solutions.

Versatility: MATLAB supports a wide range of mathematical domains, from linear algebra and calculus to differential equations, optimization, and statistics.

Interactivity: MATLAB's interactive environment allows for rapid prototyping, experimentation, and visualization, facilitating an iterative approach to problem-solving.

Ease of Use: MATLAB's intuitive syntax and extensive documentation make it accessible to users at various skill levels, from beginners to experienced professionals.

Integration: MATLAB seamlessly integrates with other tools and languages, such as Simulink for modeling and simulation, further expanding its utility in engineering and scientific applications.

Suggestions for research: As you continue your journey in mathematical exploration and problem-solving, I encourage you to delve deeper into MATLAB's capabilities. Experiment with advanced features, explore additional toolboxes, and tackle more complex mathematical challenges. Leverage online resources, tutorials, and community forums to expand your knowledge and skills in MATLAB.

References

1. Majid, M. A., Huneiti, Z. A., Balachandran, W., & Balarabe, Y. (2013). MATLAB as a teaching and learning tool for mathematics: a literature review. International Journal of Arts & Sciences, 6(3), 23.
2. Ochkov, V. F., & Bogomolova, E. P. (2015). Teaching mathematics with mathematical software. Journal of Humanistic Mathematics, 5(1), 265-285.
3. Attaway, D. C. (2013). Matlab: a practical introduction to programming and problem solving. Butterworth-Heinemann.
4. Larsen, K. F., Hossain, N. M. A., & Weiser, M. W. (2016, June). Teaching an undergraduate introductory MATLAB course: Successful implementation for student learning. In 2016 ASEE Annual Conference & Exposition.
5. Larkins, D. B., & Harvey, W. (2010). Introductory computational science using MATLAB and image processing. Procedia Computer Science, 1(1), 913-919.

6. Kurbonov, E., Rakhimov, N., Juraev, S., & Islamova, F. (2023). Derive the finite difference scheme for the numerical solution of the first-order diffusion equation IBVP using the Crank-Nicolson method. In E3S Web of Conferences (Vol. 402, p. 03029). EDP Sciences.

О МЕТОДИКЕ ОБУЧЕНИЯ СТУДЕНТОВ НЕКОТОРЫМ МЕТОДАМ РЕШЕНИЯ ДИОФАНТОВЫХ УРАВНЕНИЙ

Останов К., Тилавов Р.А.

Самаркандский государственный университет имени Шарофа Рашидова, г. Самарканд

Диофантовым уравнением первого порядка с двумя неизвестными x, y называется уравнение вида $mx+ ny=k$, где $m,n,k,x,y \in Z$. Мы предполагаем, что m и n — взаимно простые числа. Если это не так, всегда можно разделить обе части уравнения на наибольший общий делитель m и n (если в результате в правой части окажется нецелое число, то такое уравнение не будет иметь решение). Кроме того, способ решения зависит от того, насколько велики абсолютные значения чисел m и n. Если хотя бы один из коэффициентов (например, m) имеет малое абсолютное значение, мы перепишем уравнение в виде $mx=k-ny$. Левая часть полученного уравнения (1) делится на m. Следовательно, правую часть этого уравнения тоже можно разделить на m. В результате деления на m находим, что правая часть также делится на m для одного значения из заданного интервала с учетом всех возможных остатков $l=0,1,\dots,m-1$. Поскольку число m мало по абсолютной величине, вариантов тоже немного [1].

Студенты знакомы с методом перебора различных вариантов, при котором им предлагается перебрать все возможные варианты, такие как размещение гостей, поход в кино, способы проведения времени, раздача сладостей среди детей и т. д.

Пример 1. Кролики и фазаны сидят в клетке с 18 ногами. Найдите, сколько особей каждого из них находится в клетке?

Решение: Составим уравнение с двумя неизвестными, где x – количество кроликов, а y – количество фазанов: $4x+2y=18$ или $2x+y=9$. Выразим y через x: $y=9-2x$.

Далее используем метод выбора вариантов: не берем неположительные значения x (количество ног не может быть 0 или отрицательным), а также $x>4$ (в противном случае значение y отрицательно).

Таким образом, задача имеет четыре решения

x	1	2	3	4
y	7	5	3	1

Ответ: (1; 7), (2; 5), (3; 3), (4; 1).

Пример 2. Решите уравнение $5x+8y=39$ в натуральных числах.

Решение: Перепишем уравнение в виде $5x=39-8y$. Чтобы уравнение имело смысл в условиях задачи, y должно быть меньше или равно 4, но больше 0:

$0<y\leq 4$. Делаем выбор по неизвестному y:

Если $y=1$, то $x=(39-8y)/5=(39-8\cdot 1)/5=6{,}2$ не является натуральным числом.

Если $y=2$, то $x=4{,}2$ не является натуральным числом.

Если $y=3$, то $x=3$ — натуральное число.

Если $y=4$, то $x=1{,}42$ не является натуральным числом. Ответ: (3; 3).

Теперь покажем использование метода остатков на примере следующей задачи: