UNIVERSITY OF BOHOL

Tagbilaran City, Bohol, Philippines

COLLEGE OF ENGINEERING, TECHNOLOGIES,
ARCHITECTURE AND FINE ARTS

COMPUTER ENGINEERING NUMERICAL METHODS

CPEP 221

NUMERICAL PLOTTING USING MATLAB

Submitted by:

ORIG, JEFF AXEL P.

Introduction

This project is a MATLAB-based graphical application designed to assist in the study and application of numerical methods, particularly focusing on root-finding and function visualization. It provides students and users with an interactive environment to input mathematical functions, view their graphical representations, and apply numerical algorithms to approximate roots.

Built using MATLAB's GUI capabilities, the application enables users to enter a single-variable function in natural mathematical syntax, specify the plotting range, and select from several classical root-finding techniques. The system then plots the function on a graph and, if applicable, displays root locations with visual markers. For each method used, the application generates a detailed iteration table summarizing the numerical steps toward convergence, giving insight into how each method performs.

This project leverages MATLAB's powerful numerical computing environment and intuitive GUI tools to create a compact, educational, and functional tool for exploring how numerical root-finding methods work. It is especially valuable in academic settings where visual learning enhances conceptual understanding.

Scope and Limitation

Scope

This project focuses on delivering an interactive MATLAB-based environment for plotting mathematical functions and performing numerical root-finding methods. The application is designed with usability and educational value in mind, providing the following core features:

- **Function Input:** Users can input single-variable functions in standard MATLAB syntax (e.g., x^2 - 4, sin(x), exp(-x) - x).
- **Graph Plotting:** The application plots the input function over a user-defined range, allowing for visual inspection of root locations.
- **Root-Finding Methods:** The following numerical methods are implemented:
    - Incremental Search
    - Bisection Method
    - False Position (Regula Falsi)
    - Newton-Raphson Method
    - Secant Method
- **Method-Specific Iteration Tables:** For each method, the application displays a structured table showing iteration steps, function values, errors, and remarks.
- **Graphical Root Indicators:** Roots (if found) are visually marked on the plot to enhance understanding of the numerical solution.
- **User-Friendly Interface:** Built with MATLAB GUI components such as panels, buttons, text fields, and dropdowns, the interface is designed for ease of use without requiring code-level interaction.

Limitations

While the MATLAB application is functional and educational, it has the following limitations:

- **Single Function Input:** Only one function can be analyzed and plotted at a time. The application does not support systems of equations or multivariable inputs.
- **Initial Guess Sensitivity:** Methods like Newton-Raphson and Secant require appropriate initial guesses; poor inputs may lead to divergence or invalid results.
- **Numerical Derivative Approximation:** The derivative in the Newton-Raphson method is computed using finite differences, which may introduce minor numerical inaccuracies.

- **No Symbolic Parsing:** Unlike some symbolic toolkits, this application does not automatically simplify or validate complex expressions.
- **Static Iteration Visualization:** The process is not animated; only final plots and iteration tables are provided.
- **Basic Input Validation:** While common errors are handled, some invalid expressions or syntax errors might not be caught gracefully.
- **Performance Constraints:** The tool is designed for educational use and moderate function complexity. Performance may degrade with highly complex expressions or very tight tolerances.

PROBLEM REQUIREMENTS

Purpose

The main purpose of this project is to help users gain familiarity with MATLAB and build a foundational understanding of its programming language and environment. Through the development and use of an interactive GUI-based function plotter, users are introduced to essential MATLAB concepts such as function handling, plotting, GUI design, and numerical computation.

This project also serves as a learning tool for numerical methods, particularly root-finding techniques. By allowing users to input equations, define intervals, and visualize function behavior, it reinforces theoretical knowledge with practical application. The combination of visual output and method-specific iteration tables makes it easier to grasp how each method works and how MATLAB can be used to solve mathematical problems efficiently.

Overall Description

This project is a standalone MATLAB application that allows users to plot mathematical functions and find their roots using classical numerical methods. Designed with a graphical user interface (GUI), the tool provides an interactive way to input equations, set plotting parameters, and select a root-finding method to apply.

Users can enter any valid single-variable function, specify the x-range for plotting, and choose among five numerical methods: Incremental Search, Bisection, False Position, Newton-Raphson, and Secant. Once the method is executed, the program displays both the function graph and a table of iteration steps, showing how the solution is reached.

The interface is built using MATLAB's GUI components such as panels, buttons, dropdown menus, and tables, all integrated into a clean, two-column layout. The left panel contains all user inputs and controls, while the right panel displays the plot and the results, including graphical markers for the roots and method-specific iteration data.

This project not only demonstrates practical numerical computation techniques but also showcases the use of MATLAB for developing interactive applications. It serves as an effective learning platform for both programming in MATLAB and understanding the behavior of numerical algorithms.

Analysis

Input requirements

1. **Function to Plot:**
   - Accepts mathematical functions as strings (e.g., x^2 - 4, sin(x), e^-x - x).
   - Must follow MATLAB-compatible syntax for mathematical operations.

2. **Range:**
   - X Min: The starting value of the x-axis.
   - X Max: The ending value of the x-axis.
   - Both should be numeric and X Min must be less than X Max.

3. **Numerical Method Selection:**
   - Choose one of the following methods:
     - Incremental
     - Bisection
     - False Position
     - Newton-Raphson
     - Secant

4. **Tolerance:**
   - Defines the convergence threshold for root-finding methods.
   - Must be a positive numeric value.

5. **Maximum Iterations:**
   - Sets the limit for iterations during root calculations.
   - Must be a positive integer.

6. **Method-Specific Parameters:**
   - Incremental Method: Step size (positive numeric value).
   - Newton-Raphson Method: Initial guess x0 (numeric).
   - Secant Method: Initial guesses x0 and x1 (numeric).

Output requirements

1. **Graphical Output:**
   - Plots the function over the specified range.
   - Highlights root locations (if found) with markers and annotations.

2. **Numerical Results:**
   - Displays roots of the function with details in a list and tabular format.
   - Each root is accompanied by iteration-specific data depending on the method:
     - *Incremental:* Ranges, function values, and remarks on sign changes.
     - *Bisection:* Interval midpoints, errors, and function evaluations.
     - *False Position:* Interval bounds, estimated roots, and convergence errors.
     - *Newton-Raphson:* Iteration steps, errors, and derivative values.
     - *Secant:* Iterative approximations and relative errors.

3. **Tabs for Iteration Details:**
   - Iteration-by-iteration data specific to the selected root-finding method.

4. **Status Messages:**
   - Real-time feedback on errors or warnings (e.g., invalid function syntax, no roots found, convergence warnings).

Necessary Formula and their Description

1. **Incremental Search Method**

Formula:

Check for sign change:

$$f(x_l) \cdot f(x_u) \leq 0$$

where x_l and x_u are the lower and upper bounds of the interval.

Description:

This method scans the function over the range with a fixed step size to detect sign changes, indicating the presence of a root.

2. **Bisection Method**

Formulas:

Midpoint:

$$x_r = \frac{x_l + x_u}{2}$$

where x_l and x_u are the lower and upper bounds.

Error:

$$\epsilon_a = \left| \frac{x_r^{(new)} - x_r^{(old)}}{x_r^{(new)}} \right| \times 100\%$$

Description:

This method iteratively halves the interval $[x_l, x_u]$ containing a root by selecting the midpoint x_r and determining which subinterval to keep based on the sign of the function.

3. **False Position (Regula Falsi) Method**

Formulas:

Root Approximation:

$$x_r = \frac{x_l f(x_u) - x_u f(x_l)}{f(x_l) - f(x_u)}$$

Error:

$$\epsilon_a = \left| \frac{x_r^{(new)} - x_r^{(old)}}{x_r^{(new)}} \right| \times 100\%$$

Description:

This method calculates a root approximation using a linear interpolation of the function over the interval $[x_l, x_u]$. The interval is updated based on the sign of the function.

4. **Newton-Raphson Method**

Formula:

Next Approximation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Error:

$$\varepsilon_a = \left|\frac{x_{n+1} - x_n}{x_{n+1}}\right| \times 100\%$$

Description:

This method uses the derivative $f'(x)$ to refine the root approximation x_n. It requires a good initial guess x_0 to converge quickly.

5. **Secant Method**

Formulas:

Next Approximation:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Error:

$$\varepsilon_a = \left|\frac{x_{n+1} - x_n}{x_{n+1}}\right| \times 100\%$$

Description:

This method approximates the derivative by using two nearby points (x_n, x_{n-1}) to compute the next approximation. It does not require explicit calculation of $f'(x)$.

Design

Files and Their Descriptions

Filename: Program.m

A standalone MATLAB script that combines GUI-based function plotting with numerical root-finding methods. Users can input functions, configure parameters, and visualize results interactively.

Key Features:

1. **Graphical User Interface (GUI)**:
   - Left Panel: Inputs (function, range, method, parameters).
   - Right Panel: Graphs and results.

2. **Numerical Methods**:
   - Incremental Search, Bisection, False Position, Newton-Raphson, Secant.

3. **Dynamic Visualization**:
   - Plots functions and highlights roots with markers and annotations
   .

4. **Interactive Inputs**:
   - Function entry, range settings, and method-specific parameters.

5. **Results Display**:
   - Root summaries and iteration details in tabs

User Interface Design

The MATLAB GUI application features a user-friendly interface with two primary panels:

Left Panel: Controls

- *Function Input:* Enter functions or select from pre-defined examples.
- *Range and Method:* Define X Min, X Max, tolerance, and iterations. Select methods like Incremental, Bisection, False Position, Newton-Raphson, or Secant. Dynamic inputs appear for method-specific parameters.
- *Actions:* Buttons for plotting or resetting inputs.
- *Results Summary:* Display roots and details in a text area and list.

Right Panel: Visualization

- A plot shows the function and annotated roots.
- Tabs below the graph provide step-by-step iteration details and calculations.

Workflow

Input a function, set parameters, and click "Plot" to visualize and compute roots. Results are shown graphically and in detailed tabs.

The design ensures simplicity, logical navigation, and flexibility for diverse computations.

Features of the Project

1. User-Friendly GUI:
   - Intuitive two-panel layout for input controls and visualization.
   - Organized sections for easy navigation and input management.

2. Dynamic Function Plotting:
   - Supports plotting of mathematical functions over a user-defined range.
   - Highlights roots on the graph with markers and annotations.

3. Numerical Root-Finding Methods:
   - Implements five different methods for root-finding:
     - Incremental Search
     - Bisection
     - False Position
     - Newton-Raphson
     - Secant
   - Adapts input fields dynamically based on the selected method.

4. Real-Time Input Validation:
   - Ensures valid function syntax and parameters before processing.
   - Displays error messages for invalid or incomplete inputs.

5. Configurable Parameters:
   - Allows users to specify:
     - Function and range (X Min and X Max).
     - Tolerance level for convergence.
     - Maximum iterations for numerical methods.
     - Step size or initial guesses for method-specific configurations.

6. Pre-Defined Function Examples:

- Includes a dropdown menu with common mathematical functions for quick selection.

7. Detailed Results Presentation:
   - Tabbed interface for method-specific iteration data.
   - Summary of roots with computed values and error details.

8. Interactive Controls:
   - Plot Button: Generates the graph and computes roots.
   - Clear Button: Resets all inputs, plots, and results.

9. Comprehensive Visualization:
   - Real-time function plotting with dynamic updates.
   - Annotated markers for roots on the graph.

10. Robust Error Handling:
    - Alerts users about convergence issues or invalid configurations.
    - Handles edge cases like singularities or undefined function intervals.

11. Self-Contained Implementation:
    - All features integrated into a single MATLAB script, making it easy to run and manage.

Implementation

URL of the Saved Source Code

https://github.com/046289/NumericalMethodMatlab

Built-In Features of MATLAB Utilized

1. Graphical User Interface (GUI):
   - Functions like `uipanel`, `uicontrol`, `uitabgroup`, and `uitable` are used to build the interface.
   - These are standard MATLAB tools for creating GUIs.

2. Mathematical Operations:
   - The script uses MATLAB's native capabilities for function evaluation, string manipulation (`regexprep`, etc.), and numerical methods (str2func for function conversion).

3. Plotting Tools:
   - Functions like `plot`, `cla`, and `axes` are used to handle visualizations.

4. Root-Finding Algorithms:
   - All numerical methods (Incremental Search, Bisection, False Position, Newton-Raphson, and Secant) are implemented directly in the script without relying on MATLAB's numerical solvers like `fzero`.

5. Error Handling:
   - MATLAB's native error-handling mechanisms (e.g., `try-catch`) are used to manage invalid inputs and runtime issues.

Function Declarations and their Descriptive Purposes

GUI Callback Functions

These functions handle user interactions with the GUI elements.

1. method_changed
   Purpose:
   - Updates the visibility of method-specific input fields (e.g., step size or initial guesses) based on the selected root-finding method.

2. select_example
   Purpose:
   - Updates the function input field with a pre-defined example selected from the dropdown menu.

3. clear_all
   Purpose:
   - Resets all inputs, plots, and results to their default states, effectively clearing the GUI.

4. plot_function
   Purpose:
   - Handles the main functionality of the script. It validates inputs, converts the function string to MATLAB syntax, applies the selected root-finding method, plots the function, and displays results in the GUI.

Root-Finding Methods

These functions implement various numerical algorithms for finding roots of equations.

1. incremental_search_method
   Purpose:
   - Scans the function over the defined range with a fixed step size to detect sign changes, indicating potential roots. Includes refinement for improved accuracy.

2. bisection_method
   Purpose:
   - Implements the Bisection method by iteratively halving the interval containing a root until the tolerance or maximum iterations are met.

3. false_position_method
   Purpose:
   - Uses linear interpolation over an interval to approximate a root and iteratively refines the approximation.

4. newton_raphson_method
   Purpose:
   - Applies the Newton-Raphson method to refine root approximations using the derivative of the function. It requires an initial guess for convergence.

5. secant_method
   Purpose:
   - Uses two initial approximations to iteratively compute roots without requiring the derivative of the function.

Utility Functions

These provide auxiliary functionalities to support the main processes.

1. initialize_plot
   Purpose:
   - Prepares the plotting area by clearing previous plots and resetting axes to default settings.

2. remove_function_prefix
   Purpose:
   - Cleans the input function string by removing prefixes like f(x) = or similar.

3. convert_to_matlab_syntax
   Purpose:
   - Converts user-input mathematical functions into MATLAB-compatible syntax, handling implicit multiplication, constants like e, and operations like ^.

4. find_roots_with_method
   Purpose:
   - Selects and applies the appropriate root-finding method based on user input.

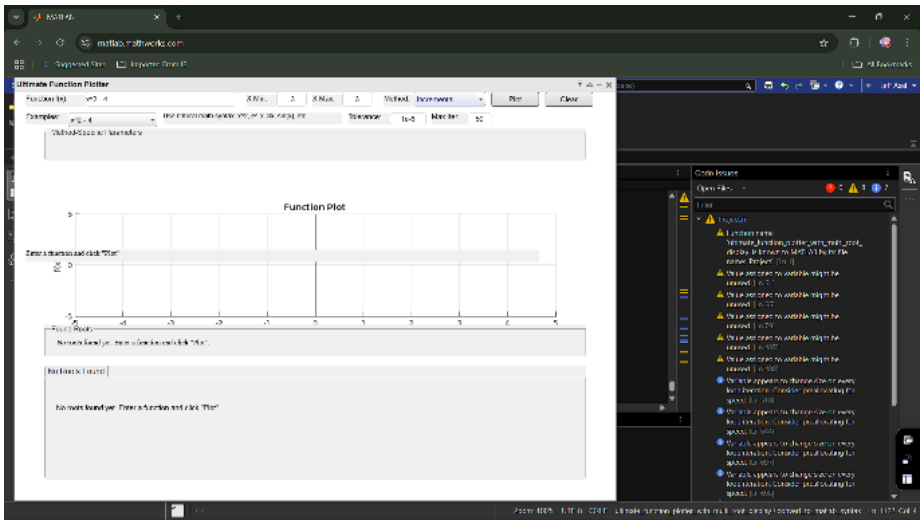
5. refine_root_bisection
   Purpose:
   - Refines roots detected in the Incremental Search method using the Bisection method for higher accuracy.

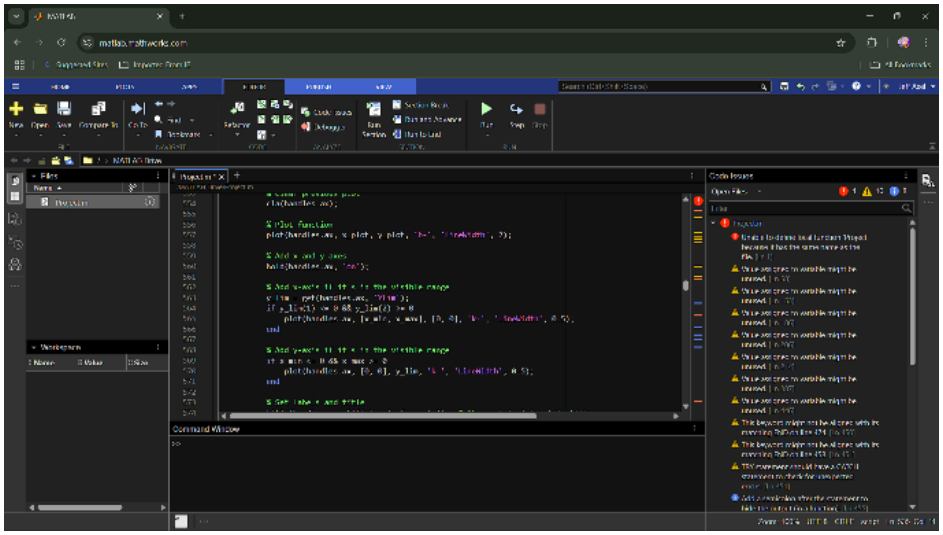
6. find_roots_newton
   Purpose:
   - Customizes the Newton-Raphson method for specific root intervals or initial guesses.

7. find_roots_secant
   Purpose:
   - Applies the Secant method for finding roots using user-defined initial guesses.

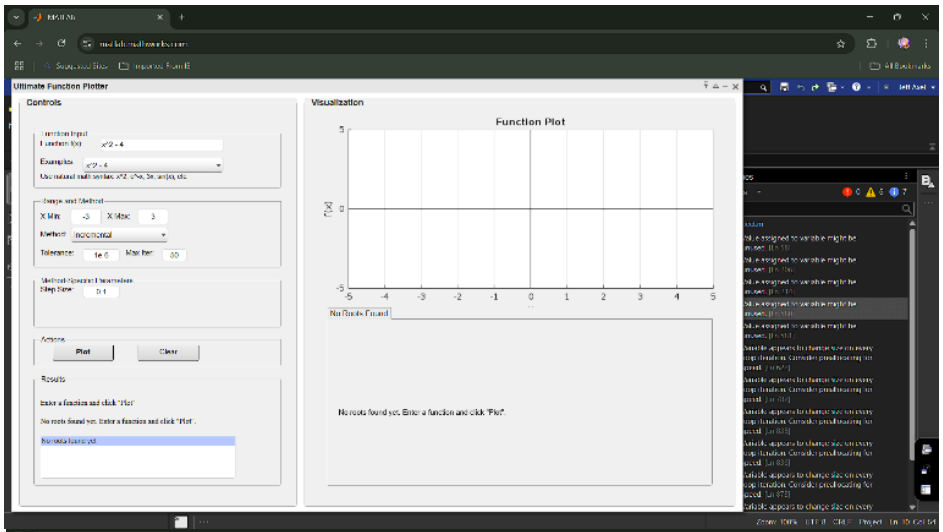
Testing and debugging

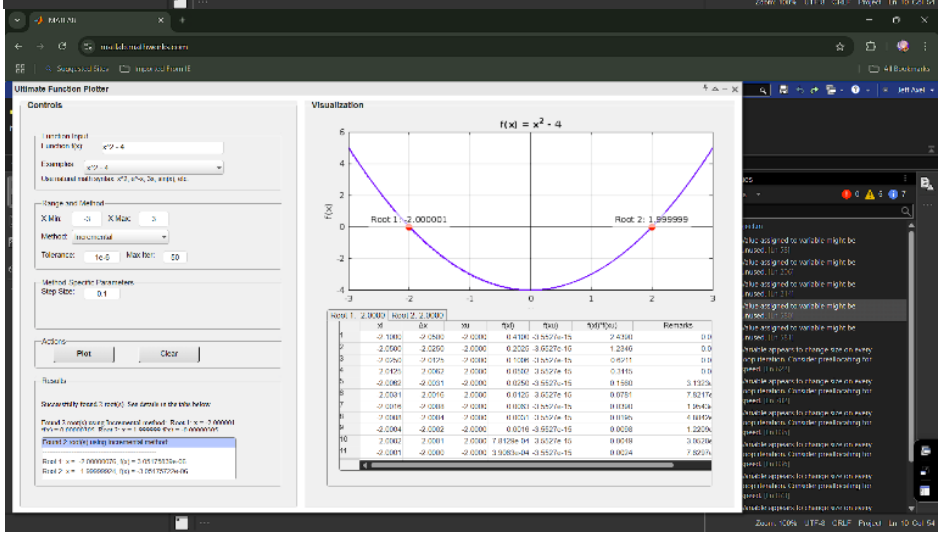
Initial output.





Final output.





UNIVERSITY OF BOHOL

CITY OF TAGBILARAN

CURRICULUM VITAE

I. PERSONAL INFORMATIONS

Name	:	Jeff Axel P. Orig
Address	:	Pamaong Stanai Rd., brg. Cogon, Tagbilaran City
Birthdate	:	June 26, 2005
Nationality	:	Filipino
Religion	:	Roman Catholic
Civil Status	:	Single
Parents	:	Eleazar Wilson Orig Juliet Orig

II. EDUCATIONAL BACKGROUND

Tertiary	:	University of Bohol, College of Engineering, Technology, Architecture and Fine Arts Dr. Cecilio Putong Street, Cogon Tagbilaran City, Bohol 2023-Present
Secondary	:	Cor Jesu Institute of Mabini Inc. Mabini, Davao de Oro S.Y. 2017-2023
Elementary	:	Maabini Central Elementary School Mabini, Davao de Oro S.Y. 2011-2017

Future Development

Recommendations

1. Expand Methods:
   - Add advanced algorithms like Brent's method or Fixed-Point Iteration.
   - Support systems of nonlinear equations.

2. Enhanced Visualization:
   - Include zoom, pan, and 3D plotting.
   - Allow plot customization (styles, colors, annotations).

3. UI Improvements:
   - Add tooltips and a live preview.
   - Provide method performance comparisons (iterations, accuracy).

4. Data Export:
   - Export plots as images and results as CSV/Excel files.

5. Robust Input Handling:
   - Improve syntax parsing and error messages.
   - Validate parameter ranges more comprehensively.

6. Integration and Optimization:
   - Leverage MATLAB toolboxes for symbolic math and optimization.
   - Optimize for faster performance with large datasets.

7. Documentation:
   - Add built-in tutorials and detailed help sections.

8. Cross-Platform Support:
   - Package as a standalone app for use without MATLAB.

Project Cost

Expenses	Costs
Printing	110
Bookbinding	150
Total	260

Glossary

1. **Function Plot**
   - A graphical representation of a mathematical function $f(x)$, showing the relationship between input x and output $f(x)$.

2. **Root**
   - A value x_r for which $f(x_r) = 0$. Roots are points where the graph intersects the x-axis.

3. **Incremental Search**
   - A numerical method to find roots by stepping through an interval and checking for a change in the sign of $f(x)$.

4. **Bisection Method**
   - A root-finding method that iteratively halves an interval containing a root, refining the approximation.

5. **False Position (Regula Falsi) Method**
   - A method similar to bisection but uses a linear interpolation to estimate the root within an interval.

6. **Newton-Raphson Method**
   - An iterative method that refines root approximations using both the function $f(x)$ and its derivative $f'(x)$.

7. **Secant Method**
   - A root-finding method that approximates the derivative using two previous function values instead of requiring $f'(x)$.

8. **Tolerance**
   - A user-defined threshold determining the stopping condition for numerical methods when approximations are close enough to the actual root.

9. **Iteration**
   - A single computational step in a numerical method used to refine the approximation of a root.

10. **Error**
    - The difference between the current approximation and the exact value. Expressed as absolute or relative error.

11. **GUI (Graphical User Interface)**
    - A user-friendly interface that allows interaction through visual components like buttons, dropdowns, and text fields.

12. **Derivative**
    - A measure of how a function changes at a point, defined mathematically as $f'(x) = \frac{d}{dx} f(x)$.

13. **Interval**
    - A range of xxx-values, typically defined by X Min and X Max, where root-finding methods search for roots.

14. **Finite Difference**
    - A numerical technique for approximating derivatives by evaluating the function at nearby points.

15. **MATLAB**
    - A high-level programming environment used for numerical computing, visualization, and algorithm development.

16. **Convergence**
    - The process by which a numerical method approaches the actual root of the function as iterations proceed.

17. **Divergence**
    - A failure of a numerical method to approach a root, often caused by poor initial guesses or inappropriate methods.

18. **Function Handle**
    - A MATLAB construct for defining and passing mathematical functions in the form @(x) expression.

19. **Sign Change**
    - A shift in the sign of $f(x)$ between two points, indicating the presence of a root in the interval.

20. **Symbolic Math**
    - Mathematical computation involving symbolic expressions rather than numerical values, often used for exact solutions.

Bibliography

MathWorks. (n.d.). *Creating a GUI with MATLAB*. Retrieved May 17, 2025, from https://www.mathworks.com/help/matlab/creating_guis.html

MathWorks. (n.d.). *MATLAB function syntax and use*. Retrieved May 17, 2025, from https://www.mathworks.com/help/matlab/function-basics.html

MathWorks. (n.d.). *Creating user interfaces*. Retrieved May 17, 2025, from https://www.mathworks.com/help/matlab/creating-user-interfaces.html

MathWorks. (n.d.). *MATLAB graphics and visualization*. Retrieved May 17, 2025, from https://www.mathworks.com/help/matlab/graphics.html