UNIVERSITY OF BOHOL

Tagbilaran City, Bohol, Philippines

COLLEGE OF ENGINEERING,

TECHNOLOGIES, ARCHITECTURE AND FINE ARTS

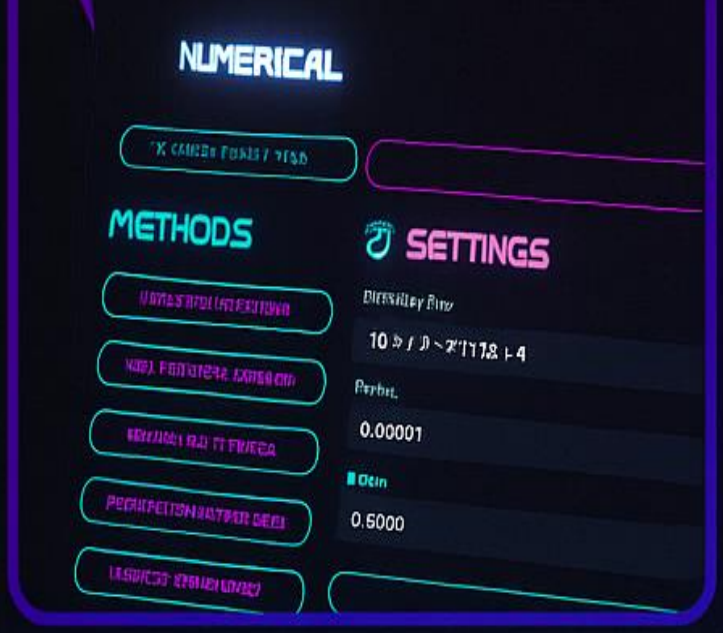
NUMERICAL METHODS

CPEP 221

ROOT FINDER- MATLAB & NUMERICAL

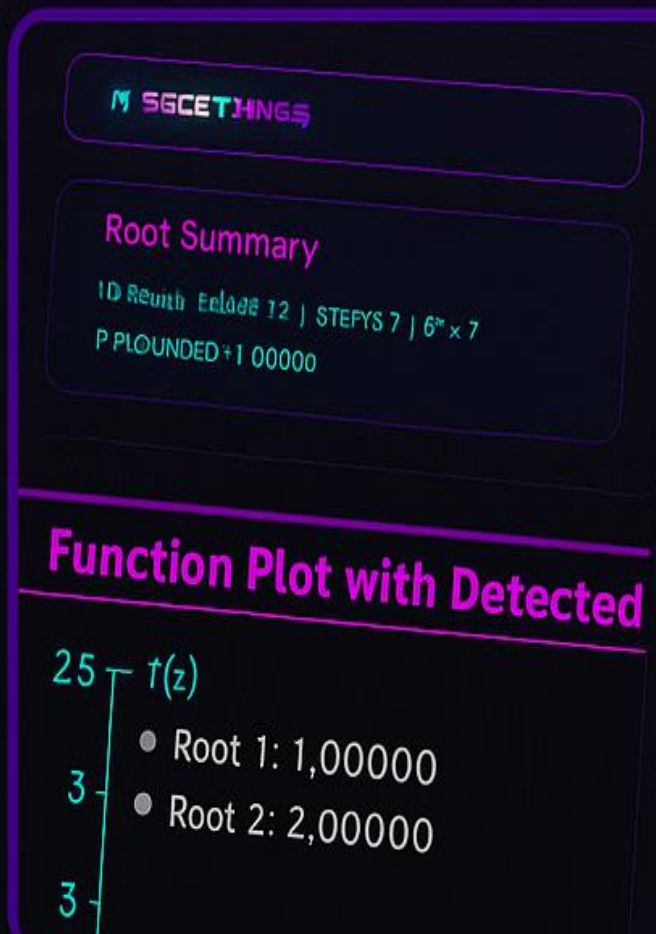
Submitted by:

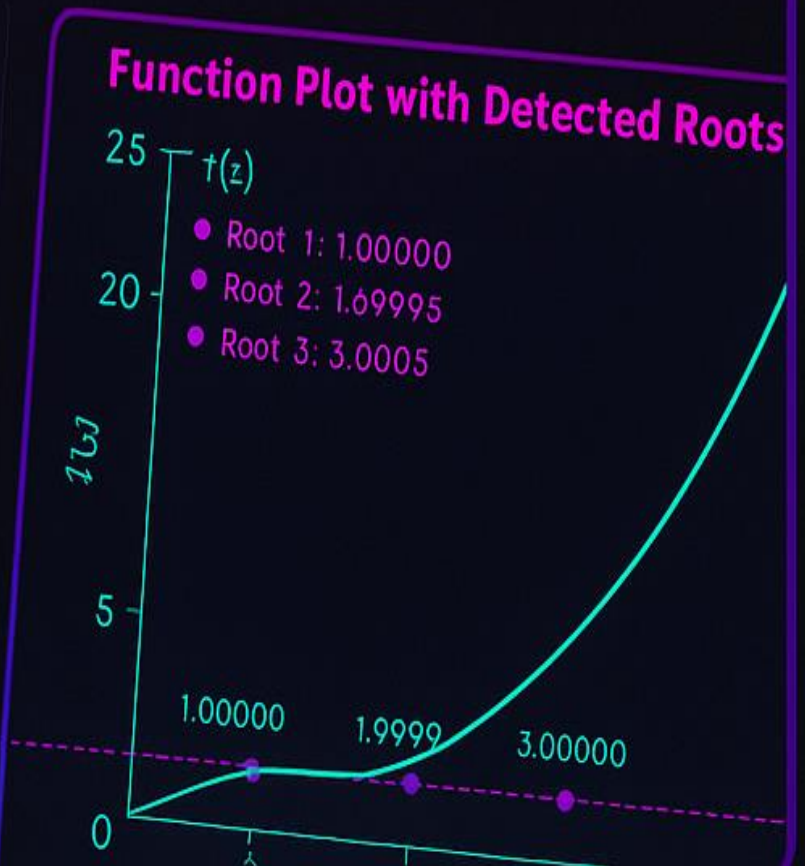
SARIP, SAINODIN A.



ROOT FINDER

MATLAB & NUMERICAL





Introduction

Numerical methods are used to approximate solutions for mathematical problems that cannot be solved analytically. This project focuses on various root-finding algorithms designed to approximate the zeros of real-valued functions—an essential aspect in engineering, science, and applied mathematics. The project implements the following six techniques:

- Graphical Method
- Incremental Search
- Bisection Method
- Regula Falsi (False Position)
- Newton-Raphson Method
- Secant Method

A GUI was developed to allow users to input functions, define parameters, and visualize the solution process.

Scope and Limitations

Scope

- Implements six root-finding algorithms
- Accepts user-defined functions
- Displays iterative calculations
- Provides graphical visualization for select methods
- Includes feedback for errors or divergence

Limitations

- Assumes continuous and well-behaved functions
- Handles only single-variable, real-valued functions
- Limited to numerical (not symbolic) computation
- Derivative input is required for Newton-Raphson (unless automated)

Problem Requirements

Purpose

- Serve as a computational tool for comparing root-finding techniques
- Help users understand algorithm behavior and accuracy
- Provide visual and numerical insight into root-finding processes
- Identify strengths and weaknesses of each method

Overview

The application allows users to:

- Input a function f(x)f(x)f(x)
- Specify parameters like intervals or guesses
- Select any of the six methods
- View results: approximations, errors, function values, and iteration counts
- Access visual aids for graphical methods

Each algorithm checks for convergence and handles invalid conditions appropriately.

Platform-Specific Analysis

Python Version

Input:

- Equation using x
- Supports various functions and symbols (π, e, ln, log, etc.)
- User selects method and sets parameters (tolerance, steps, etc.)

Output:

- Plots with root annotations
- Iteration tables
- Tooltips and error feedback
- Real-time cursor-based data display

MATLAB Version

Input:

- Function expression (e.g., x^2 - 4)
- Search interval, tolerance, and maximum iterations
- Method selection

Output:

- Calculated roots and their function values
- Approximation error
- Iteration count
- Graph with root markers
- Tabular summary of results

Sample Equations:

- Users can load pre-defined sample equations for quick testing.

Output Requirements Graphical

Output:

- Plots the function over the specified range.

- Roots are marked and annotated on the plot.

Tabular Output:

- Iteration table for the selected method, showing step-by-step calculations.

Status/Feedback:

- Error messages for invalid input.

- Tooltips and instructions for user guidance.

- Status bar showing coordinates and nearest root on interaction.

MATLAB Methods

- **Bisection:** Halves intervals with opposite signs
- **Newton-Raphson:** Uses derivative for linear root approximation
- **Secant:** Derivative-free using two prior approximations
- **Regula Falsi:** Linear interpolation method
- **Incremental Search:** Finds intervals with sign changes
- **Graphical:** Visual detection of roots

System Design

Python File Structure

- main_app.py: UI and method integration
- Method modules: bisection.py, regula_falsi.py, newton_raphson.py, etc.

Input Requirements

- Mathematical Function: User-provided function in the form of a MATLAB expression (e.g., x^2 4)
- Search Range: Minimum and maximum x-values defining the interval to search for roots
- Numerical Method Selection: User's choice among Bisection, Newton-Raphson, Secant, Regula Falsi, Incremental Search, or Graphical methods
- Tolerance: Convergence criterion for terminating the root-finding algorithm (default: 1e-6)
- Maximum Iterations: Upper limit for the number of iterations (default: 100) Output Requirements
- Root Values: The x-value(s) where the function equals zero
- Function Value at Root: The f(x) value at each detected root (ideally near zero)
- Error Estimation: Approximation error for each root
- Iteration Information: Number of iterations used to find each root
- Visual Representation: Plot of the function with the roots highlighted
- Tabular Results: Detailed information about each detected root

Necessary Formulae and Their Description

Python:

Graphical Method:

- Plots \(f(x) \) and visually identifies roots (where \(f(x) = 0 \)).

Incremental Search:

- Checks sign changes in \(f(x) \) over intervals to bracket roots.

Bisection Method:

- Iteratively halves the interval \([a, b]\) where \(f(a) \cdot f(b) < 0 \) until the root is found within tolerance.

Regula Falsi (False Position):

- Uses linear interpolation between points with opposite signs to estimate the root.

Newton-Raphson:

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

Secant Method:

- Uses two previous points: $x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$.

MATLAB:

Graphical Method:

: Systematically steps through the interval to locate sign changes

Description: Visually identifies where the function crosses the x-axis

Bisection Method:

Formula: c = (a + b) / 2 where [a,b] is an interval with opposite function signs

Description: Repeatedly divides the interval in half, selecting the subinterval where the function changes sign

Newton-Raphson Method:

Formula: x_{i+1} = x_i - f(x_i) / f'(x_i)

Description: Uses the function and its derivative to approximate roots through linearization

Secant Method:

Formula: x_{i+1} = x_i - f(x_i) * (x_i - x_{i-1}) / (f(x_i) - f(x_{i-1}))

Description: Approximates derivative using two previous points, useful when derivative is unavailable

Regula Falsi Method:

Formula: c = (a*f(b) - b*f(a)) / (f(b) - f(a))

Description: Uses linear interpolation between endpoints to find root estimates Incremental

Design

Files and Their Description Python:

main_app.py:

- Main application file. Handles UI, plotting, user input, and integrates all root-finding methods. Methods
- bracketing/bisection.py: Bisection method implementation.
- bracketing/regula_falsi.py: Regula Falsi method implementation.
- open/newton_raphson.py: Newton-Raphson method implementation.
- open/secant.py: Secant method implementation.
- basic/graphical.py: Graphical root-finding method.
- basic/incremental.py: Incremental search method.

SimpleRootFinderFixed.m: Main application file containing the GUI class definition Implements the user interface and controller logic

RunFixedRootFinder.m: Launcher script that starts the application

ImprovedRootFinder.m: Enhanced version with better multiple root detection Extends SimpleRootFinderFixed with improved algorithms

RunImprovedRootFinder.m: Launcher for the enhanced version Method Implementation

Files:

bisectionMethod.m: Implements the Bisection algorithm
newtonRaphsonMethod.m: Implements the Newton-Raphson algorithm
secantMethod.m: Implements the Secant algorithm
regulaFalsiMethod.m: Implements the Regula Falsi algorithm
incrementalSearchMethod.m: Implements the Incremental Search
algorithm graphicalMethod.m: Implements the Graphical method

Utility Files:

findSignChanges.m: Helper function to locate potential root intervals

MATLAB:

Input Parameters Section: Method dropdown selector Function input text field Range (min, max) numeric inputs Tolerance input field Maximum iterations input field Action buttons (Calculate, Clear, Help)

Function Visualization Section: Graph plot showing function curve X-axis and Y-axis with appropriate scaling Root markers (red circles) at detected roots

Results Section: Table displaying root details Status label showing operation results Error messaging for user feedback

Features of the Project

Multiple Root Detection: Systematically searches intervals for sign changes Handles functions with multiple roots

Diverse Numerical Methods: Six different root-finding algorithms Each with unique advantages for different functions

Interactive Visualization: Real-time function plotting Visual identification of roots

Error Handling: Robust validation of user inputs Graceful handling of numerical challenges Informative error messages

User-Friendly Interface: Intuitive controls Clear presentation of results Help functionality

Features Summary

- Supports six root-finding algorithms
- Interactive plotting and root annotations
- Scrollable, dynamic iteration tables
- Built-in error handling
- Input validation and user guidance
- Real-time graphical updates

Implementation

GitHub

https://github.com/saripSai/Numerical-Method

Python Requirements:

- tkinter, numpy, sympy, matplotlib

MATLAB

Includes GUI launcher scripts, individual method files, and utility functions.

Function Overview

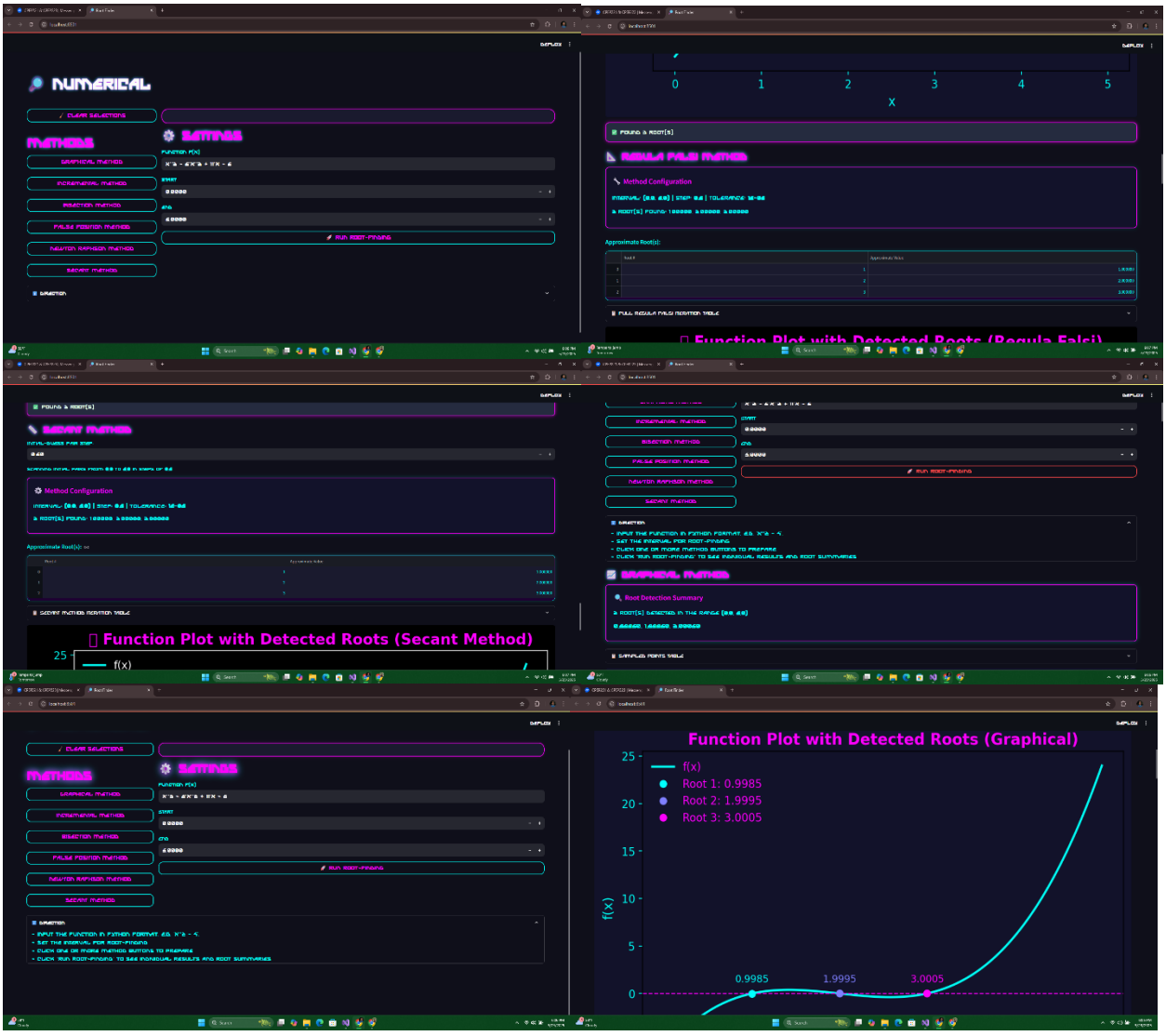
Python

- RootFinderApp: Manages the entire interface and logic
- Other functions handle parsing, plotting, tooltips, table updates, etc.

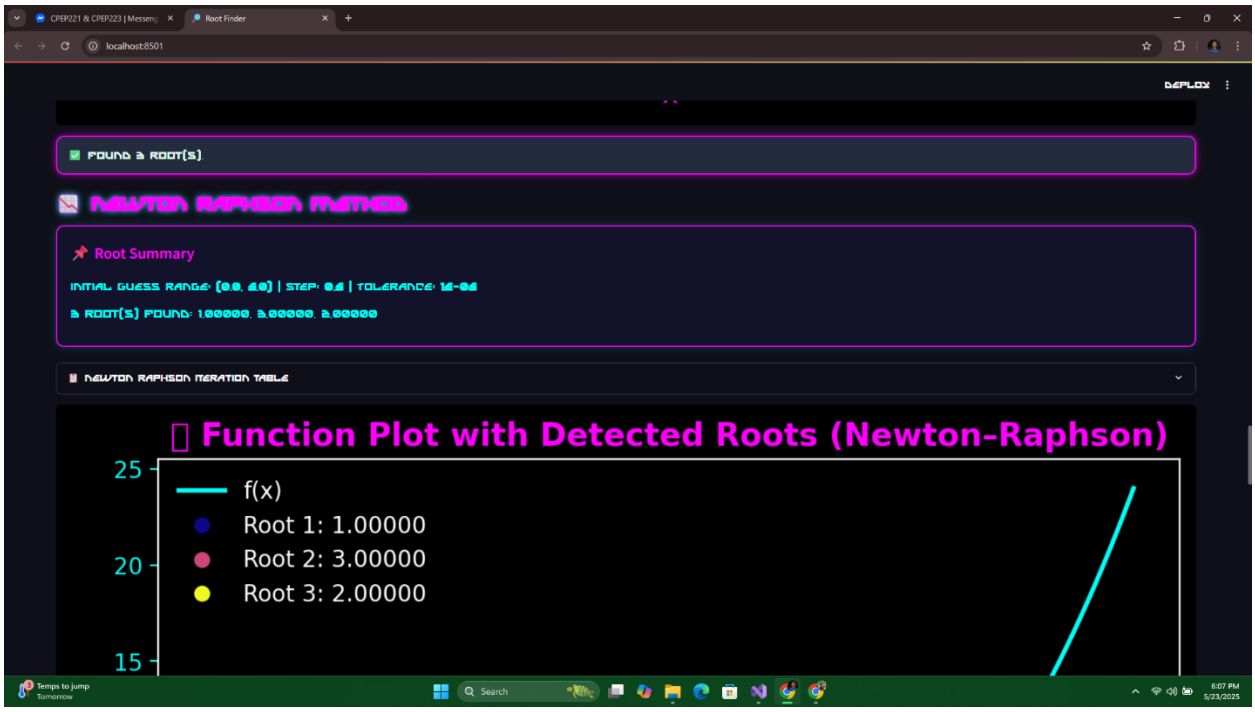
MATLAB

- calculateRoots(), updateResultsDisplay(), plotFunction(), etc.
- Each method has its dedicated function file
- Enhanced version includes multiple root detection

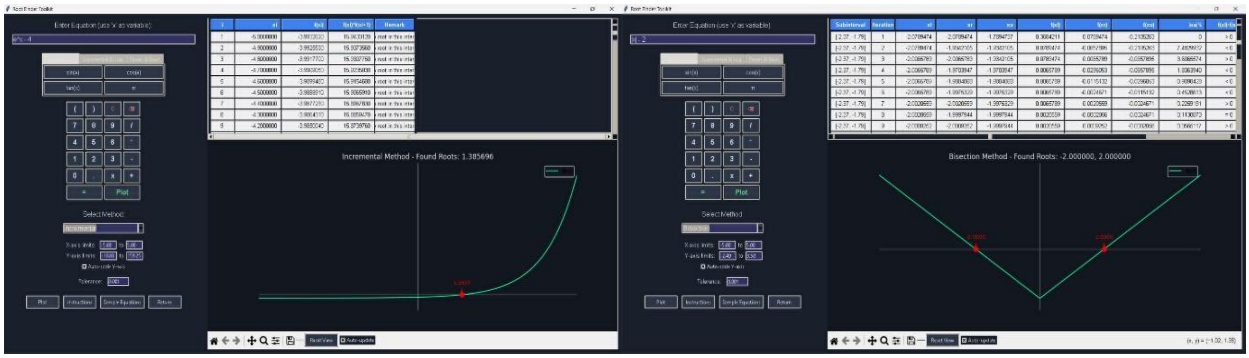
Testing

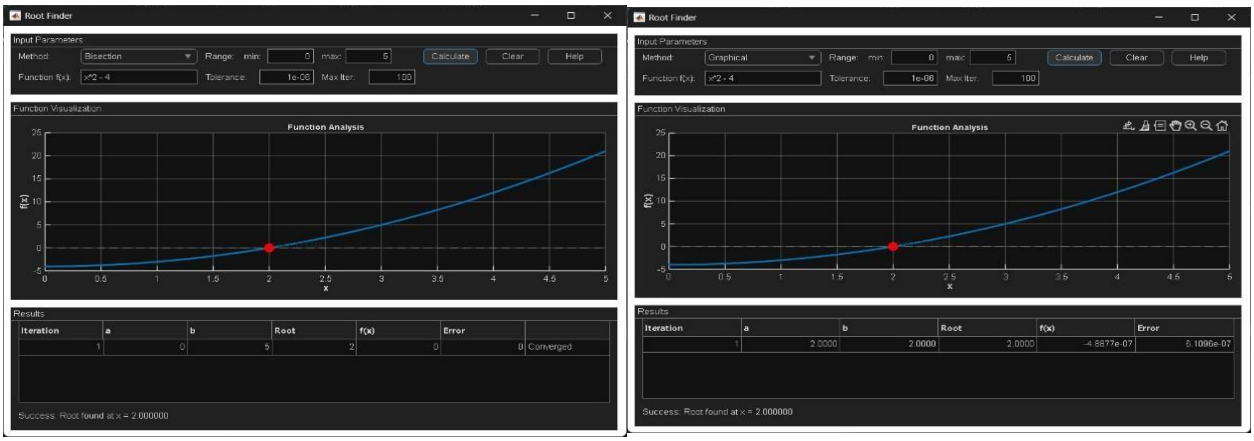


ROOT FINDER- MATLAB & NUMERICAL



MATLAB:





Input Parameters Section: Method dropdown selector Function input text field Range (min, max) numeric inputs Tolerance input field Maximum iterations input field Action buttons (Calculate, Clear, Help)

Function Visualization Section: Graph plot showing function curve X-axis and Y-axis with appropriate scaling Root markers (red circles) at detected roots

Results Section: Table displaying root details Status label showing operation results Error messaging for user feedback

Features of the Project

Multiple Root Detection: Systematically searches intervals for sign changes Handles functions with multiple roots

Diverse Numerical Methods: Six different root-finding algorithms Each with unique advantages for different functions

Interactive Visualization: Real-time function plotting Visual identification of roots

Error Handling: Robust validation of user inputs Graceful handling of numerical challenges Informative error messages

User-Friendly Interface: Intuitive controls Clear presentation of results Help functionality

Curriculum Vitae

Name: SARIP, SAINODIN A.

Email Address: saripaposacassainodin@gmail.com

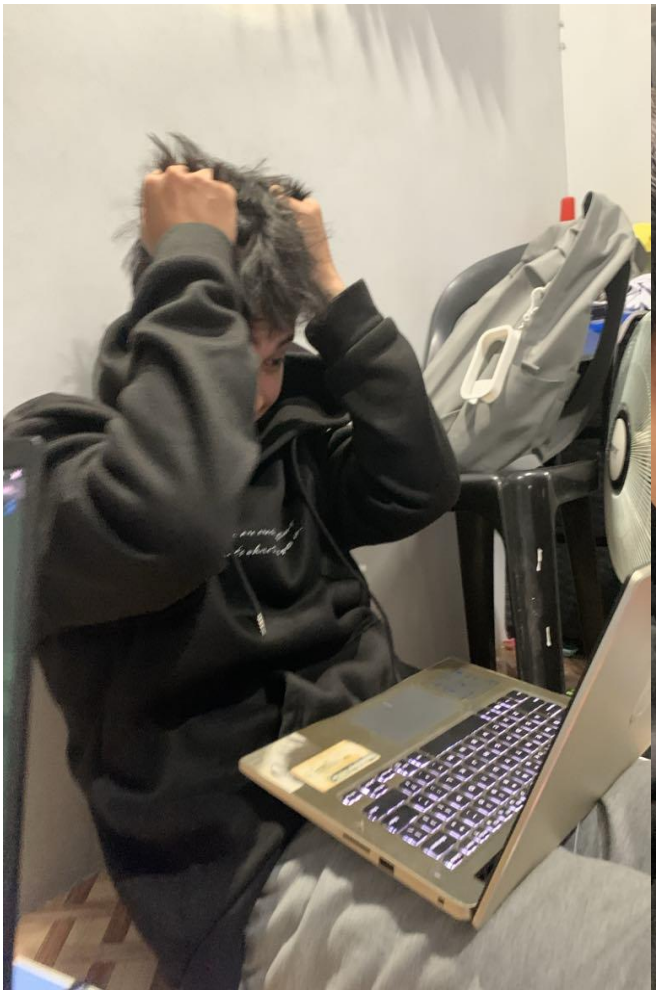
Mobile Number: 09817936436

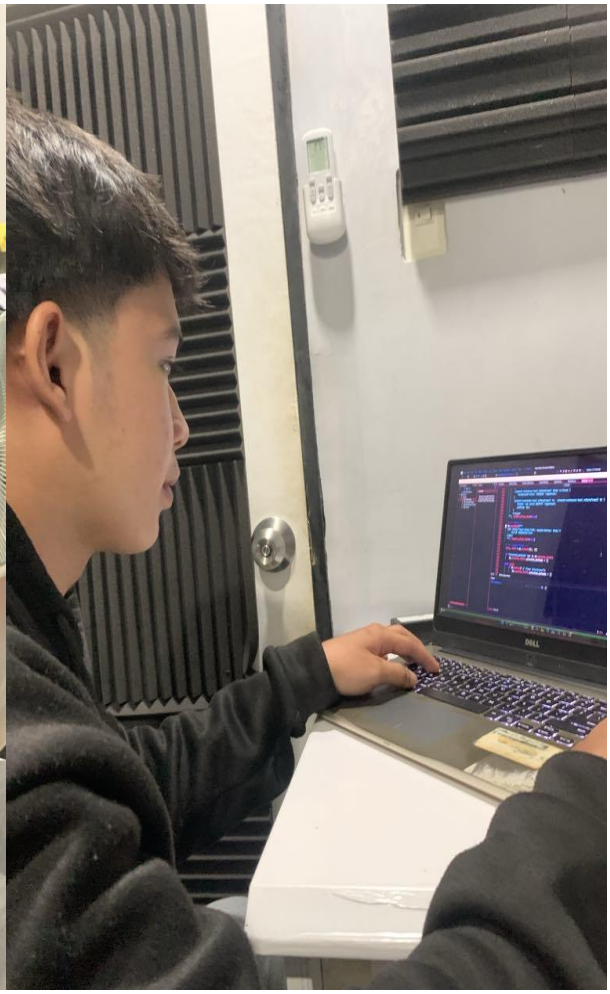
Address: Bantolinao, Antequera, Bohol

Task: Programmer

Photos during implementation

ROOT FINDER- MATLAB & NUMERICAL





Glossary

A

- Absolute Error: The absolute difference between the approximate value and the exact value.
- Algorithm: A step-by-step procedure for calculations.
- Approximation: A value that is close to but not exactly equal to the true value.

B

- Bisection Method: A root-finding method that repeatedly divides an interval in half and selects the subinterval containing the root.
- Bracketing Method: A root-finding method that starts with an interval where the function changes sign.

C

- Convergence: The property of approaching a limit, such as a root, more closely with each iteration.
- Convergence Rate: The speed at which an iterative method approaches the solution.

D

- Derivative: The rate at which a function is changing at a given point.
- Divergence: When an iterative method moves away from the solution.

E

- Error: The difference between the approximate and exact value.
- Extrapolation: Estimating a value outside the range of known data points.

F

- False Position Method (Regula Falsi): A root-finding method that uses linear interpolation to approximate the root.
- Function: A relation that maps inputs to exactly one output.

G

- Graphical Method: A visual approach to finding roots by plotting the function.

I

- Initial Guess: The starting point for an iterative method.
- Interpolation: Estimating a value between two known values.
- Iteration: The repetition of a process to approach a desired result.

M

- Monotonic Function: A function that is either entirely non-increasing or non-decreasing.

N

- Newton-Raphson Method: A root-finding method that uses the function and its derivative to find successively better approximations.
- Numerical Method: A technique used to solve mathematical problems by numerical approximation.

O

- Open Method: A root-finding method that doesn't require bracketing but may not always converge.

P

- Precision: The degree of exactness with which a number is expressed.

R

- Relative Error: The absolute error divided by the magnitude of the exact value.
- Root: A solution to the equation f(x) = 0.
- Root-Finding Algorithm: A numerical method for finding roots of continuous functions.

S

- Secant Method: A root-finding method that uses a succession of roots of secant lines to approximate the root.
- Stopping Criterion: The condition that determines when to stop the iterative process. T
- Tolerance: The acceptable level of error in an approximation.
- Tolerance Level: The maximum allowable error in an approximation.

Bibliography

- Books
- Journal Articles
- Online Resources
- Burden, R. L., & Faires, J. D. (2010). Numerical Analysis (9th ed.). Cengage Learning
- Chapra, S. C., & Canale, R. P. (2014). Numerical Methods for Engineers (7th ed.). McGraw-Hill Education.
- Kiusalaas, J. (2013). Numerical Methods in Engineering with Python 3 (3rd ed.). Cambridge University Press.
- Ortega, J. M., & Rheinboldt, W. C. (2000). Iterative Solution of Nonlinear Equations in Several Variables. Society for Industrial and Applied Mathematics.
- Traub, J. F. (1982). Iterative Methods for the Solution of Equations. Chelsea Publishing.
- Wikipedia contributors. (2023). Root-finding algorithms. Wikipedia, The Free Encyclopedia.
- https://en.wikipedia.org/wiki/Root-finding_algorithms
- MathWorld. (2023). Newton's Method. Wolfram MathWorld.