UNIVERSITY OF BOHOL

Tagbilaran City, Bohol, Philippines

COLLEGE OF ENGINEERING, TECHNOLOGIES,

ARCHITECTURE AND FINE ARTS

COMPUTER ENGINEERING

NUMERICAL METHODS

CPEP221

NUMERICAL METHODS APPLICATION

Submitted by:

Jayzee Joel A. Teves

INTRODUCTION

A **Numerical Methods Application** is an application designed to find **Roots** using the **Search Methods**. This project is a Python-based GUI application designed to assist users in finding the roots of mathematical equations using several classical numerical methods. Built using Tkinter and SymPy, the tool provides a user-friendly interface for educational and analytical purposes.

SCOPES AND LIMITATION

Scope:

- Supports six root-finding algorithms: Graphical, Incremental, Bisection, Regula Falsi, Newton-Raphson, and Secant.
- Symbolic parsing and evaluation of user-input equations.
- Visual graph output with root labeling.
- Tabulated iteration data per method.

Limitations:

- Accepts only single-variable equations with variable x.
- No support for equations with discontinuities or complex roots.
- Performance may degrade for extremely small step sizes or very high iteration limits.
- Requires manual validation of method applicability (e.g., initial guesses).

Problem Requirements

PURPOSE

This project serves as a comprehensive learning and exploration tool for numerical methods in root finding. Numerical methods are fundamental in scientific and engineering disciplines where analytical solutions to equations are often unattainable or impractical. The primary purpose of this application is to provide users with an interactive and educational environment to explore and understand the step-by-step behavior of classic root-finding algorithms.

Through a graphical interface, users are encouraged to engage with different algorithms—such as Bisection, Regula Falsi, Newton-Raphson, and Secant—by inputting equations, adjusting parameters, and observing iterative progress and convergence behaviors. This hands-on experimentation helps users develop intuition about convergence criteria, method suitability, and the mathematical underpinnings of each algorithm. Additionally, the inclusion of visual graphing and tabulated iterations makes abstract mathematical concepts more concrete and approachable.

Ultimately, the application aims to bridge theoretical learning with practical computation, making it a valuable tool for students, educators, and enthusiasts seeking deeper insight into numerical analysis.

OVERALL DESCRIPTION

This application is a desktop-based graphical tool developed in Python using the Tkinter framework for the GUI and SymPy for symbolic computation. It provides an integrated environment for performing numerical root-finding operations through a clean and intuitive interface. Users can input mathematical expressions, specify parameters such as interval bounds, tolerance, and iteration limits, and then select from a list of classical numerical methods to compute the roots of equations.

The interface guides the user through each step, making it suitable for both educational and practical use. The underlying logic parses the input function symbolically, converts it into a numerical representation, and processes it through the chosen method—whether that's the iterative Newton-Raphson approach or the robust Bisection algorithm. The results are then presented in two formats: a tabulated list of iteration steps and a graph that plots the function along with the estimated root(s).

Built entirely in Python 3, the application makes use of libraries such as numpy for numerical operations and matplotlib for rendering graphical outputs. Its design supports reusability and extensibility, making it simple to integrate additional methods or features in the future. Whether used as a study companion, a teaching tool, or a lightweight numerical analysis application, this program offers clarity, functionality, and educational value in exploring the behavior of root-finding algorithms.

SYSTEM REFERENCES

https://www.mediafire.com/file/uu9j7cyrcmgs5n1/AN_INTRODUCTION_TO_NUMERICAL_METHODS_AND_ANALYSIS_2nd_EDITION_by_James_F_Epperson.pdf/file?authuser=0

https://www.mediafire.com/file/ncc622eerv62jdy/Numerical_Methods_FOR_eNGINEERS_BY_Steven_C._Chapra_%2526_Raymond_P._Canale.pdf/file?authuser=0

https://www.mediafire.com/file/m912ae0gv0cfumz/NUMERICAL_METHODS_BY__Jeffrey_R._Chasnov.pdf/file?authuser=0

Analysis

INPUT REQUIREMENTS

Field	**Type**	**Description**
Function(f(x))	str	Algebraic Expression
Method	select	One of 6 Numerical Methods
a(lower/x0)	float	Starting/Interval Point(depends on Method)
b(upper/x1)	float	Ending/Interval Point(depends on Method)
delta X(step size)	float	Incremental Method step-size
Tolerance	float	Acceptable Error Margin
Max Iterations	int	Capacity on Loop Count

OUTPUT REQUIREMENTS

- **Table** of iteration steps with computed values.
- **Graph** showing the function and identified roots.
- **Root Estimate** clearly labeled on the graph.
- **Error Handling** for invalid inputs or math domain issues.

NECESSARY FORMULA AND THEIR DESCRIPTION

Numerical Methods Application

Method	Formula	Description
Bisection	c=(a+b)/2	Midpoint between a and b used iteratively
Regula-Falsi	c=b-f(b)(a-b)/f(a)-f(b)	Uses Secant between points to find next guess
Newton-Raphson	x1 = x0-f(x0) / f ‘ (x0)	Uses Tangent Line to find better approximation
Incremental Method	Repeatedly evaluate f(x) until sign change is found	Detects Interval Containing Roots
Secant Method	x2=x1-f(x1)(x1-x0)/(f(x1)-f(x0))	Like Newton-Raphson but without derivatives
Graphical	Plot f(x) over a range	Visual Identifications of Root Regions

Design

The application follows a modular design strategy inspired by the Model-View-Controller (MVC) pattern. The separation of logic, interface, and computation ensures maintainability and scalability.

- **User-Interface Design:** The interface is built using Tkinter widgets arranged logically into input and output sections. Input parameters such as the function, interval bounds, tolerance, and method are captured through labeled `Entry`, `Combobox`, and `Button` widgets. The layout is responsive and organized with `LabelFrame` containers. Graphical output is embedded directly into the GUI via `matplotlib` using `FigureCanvasTkAgg`, and iteration results are tabulated in a `Treeview` table for clarity.
- **MVC-Inspired:**
  - UI layer with input widgets (Tkinter)
  - Logic layer with mathematical computation per method
  - View layer with matplotlib plotting and table rendering
- **Reusable method structure for easy expansion.**
- **Validation and Exception Handling integrated.**
  - UI layer with input widgets (Tkinter)

Numerical Methods Application

- Logic layer with mathematical computation per method
- View layer with matplotlib plotting and table rendering

- **Reusable method structure for easy expansion.**
- **Validation and Exception Handling integrated.**

Security and Audit Considerations

This application is intended for local, educational, and analytical use only, and as such, security considerations are minimal but relevant in certain contexts.

- **Input Validation**: The application includes basic error handling for invalid mathematical expressions, including syntactic checks and zero-division protections. However, since the application uses symbolic evaluation, unexpected inputs could still lead to runtime exceptions if not properly sanitized.
- **Data Privacy**: The application does not store or transmit any user input or output. All operations are performed in-memory within the local runtime environment, ensuring privacy of user equations and results.
- **Code Auditing**: As an open-source educational tool, the codebase can be reviewed by peers or instructors. Key functions such as parsing, evaluating, and iterating are encapsulated and clearly labeled to support transparency.
- **External Libraries**: The libraries used (`sympy`, `numpy`, `matplotlib`, `tkinter`) are trusted, widely adopted in the academic and scientific Python community, and sourced from official channels.
- **Execution Scope**: The app should not be modified to execute arbitrary system-level operations or external commands. It assumes a safe and secure Python runtime without sandboxing or network permissions.

IMPLEMENTATION

☂ Url of the saved source code:

https://github.com/Jesb405/NumericalMethods

File Names and Their Description

File	**Description**
Numerical_Methods_gui.py	Main Application file containing GUI logic and root-finding method logic
requirements.txt	List of required python Packages(e.g. numpy, matplotlib, sympy)
README.md	Overview of the Project,Setup Instructions, and Usage
doc/ folder	Contains Project Documentation, Diagrams, and Design Notes

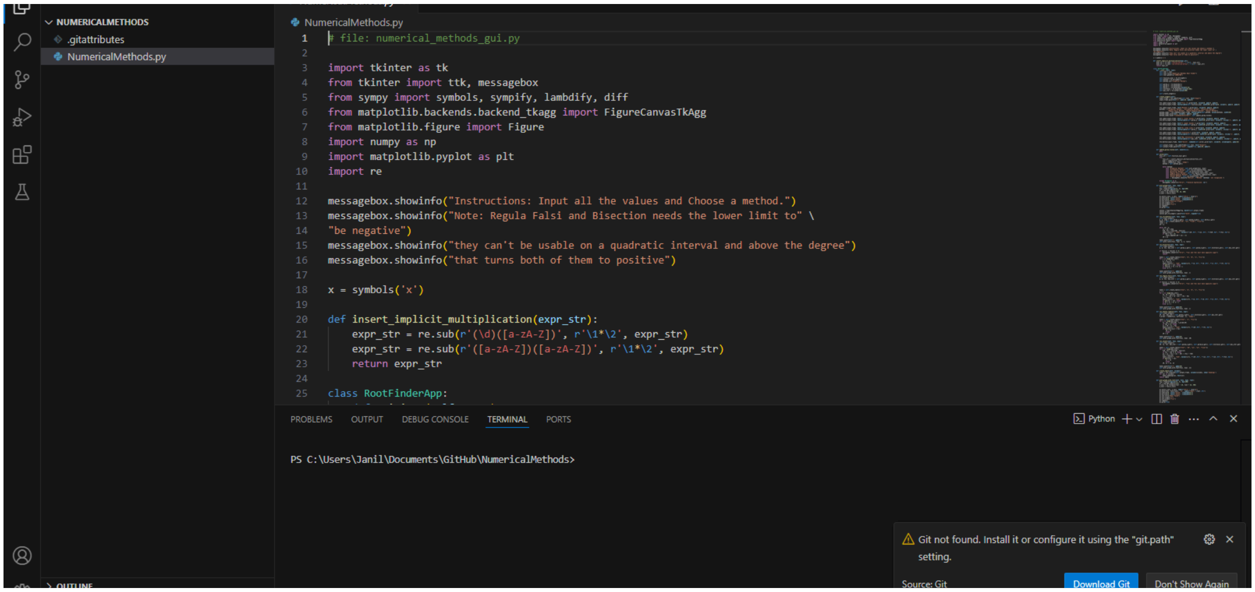
FUNCTION DECLARTIONS AND THEIR DESCRIPTITVE PURPOSES

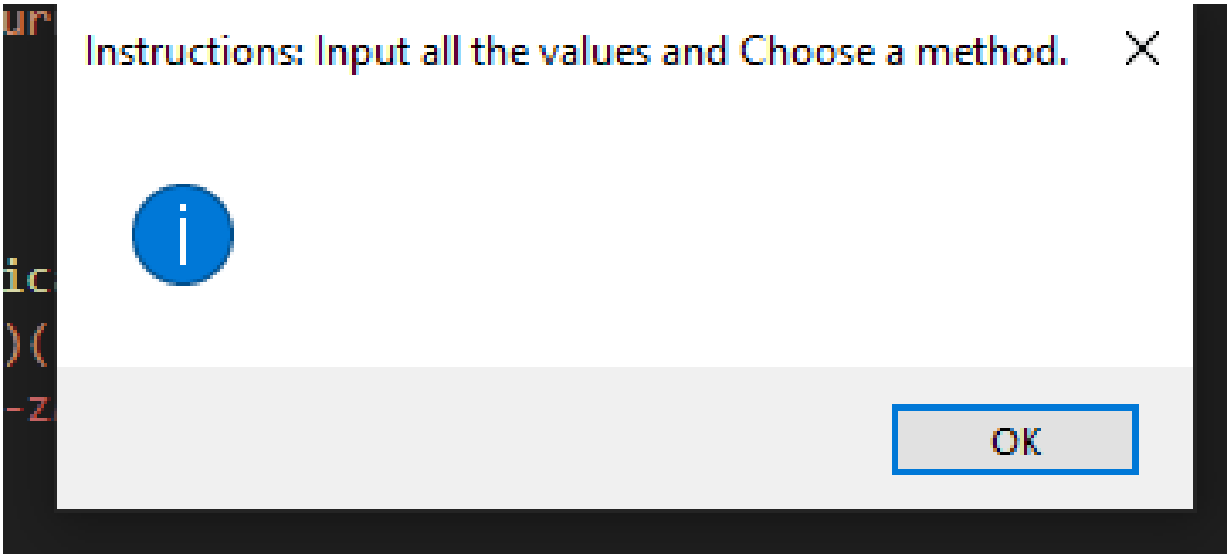
Function Name	Description
__init__	Initializes the main GUI, variables, and interface layout.

create_widgets	Builds and arranges all input fields, labels, buttons, and output sections.
update_param_fields	Placeholder for dynamic field adjustments based on method (future use).
insert_implicit_multiplication	Preprocesses the input equation by inserting multiplication symbols.
solve	Dispatches the selected root-finding method and handles expression parsing.
run_incremental	Implements the Incremental Search method and outputs interval roots.
run_bisection	Applies the Bisection method and shows the iterative table and graph.
run_regula_falsi	Executes the Regula Falsi method with validation and visual output.
run_newton_raphson	Performs Newton-Raphson iterations using symbolic derivatives.
run_secant	Calculates roots via the Secant method iteratively.
plot_graph	Plots the mathematical function over a default range.
plot_graph_with_root	Plots the function and highlights a specific root value.

TESTING AND DEBUGGING

Sample Run





The Team

Names and Tasks

Teves, Jayzee Joel A.

- Project Developer − Overseeing project development
- Writer − Designing the user interface and user experience
- Debugger - Fixing Bugs and other Unintended uses
- Documentation - creates a documentation about this Project

CURRICULUM VITAE

NAME :TEVES, JAYZEE JOEL A.

COURSE : COMPUTER ENGINEERING

BIRTHDATE : NOVEMBER 14, 2003

BIRTHPLACE : TAGBILARAN CITY, BOHOL

NATIONALITY : FILIPINO

RELIGION : APOSTOLIC

GENDER : MALE

ADDRESS : TALOTO DISTRICT, TAGBILARAN CITY, BOHOL

EMAIL : JJATEVES@UNIVERSITYOFBOHOL.EDU.PH

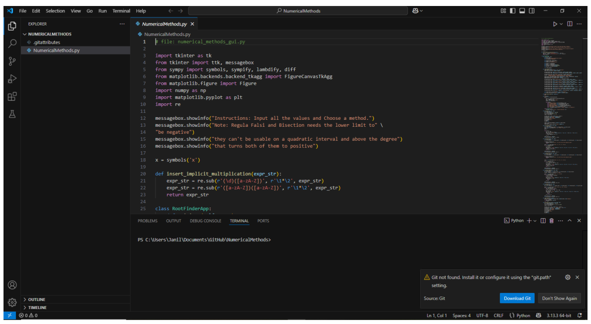
II. EDUCATIONAL BACKGROUND

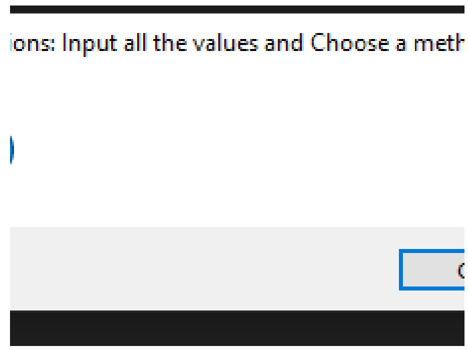
TERITARY: UNIVERSITY OF BOHOL, COLLEGE OF ENGINEERING,
TECHNOLOGY, ARCHITECTURE AND FINE ARTS
DR. CECILIO PUTONG STREET, COGON TAGBILARAN CITY, BOHOL
2023 – PRESENT

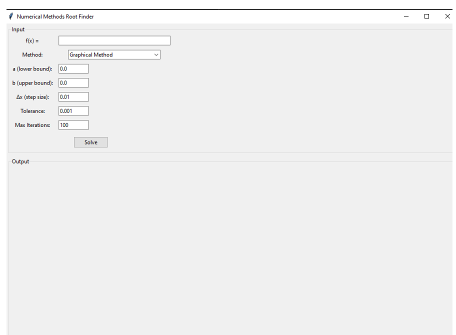
SECONDARY: TAGBILARAN GRACE CHRISTIAN SCHOOL
S.Y. 2016 – 2022

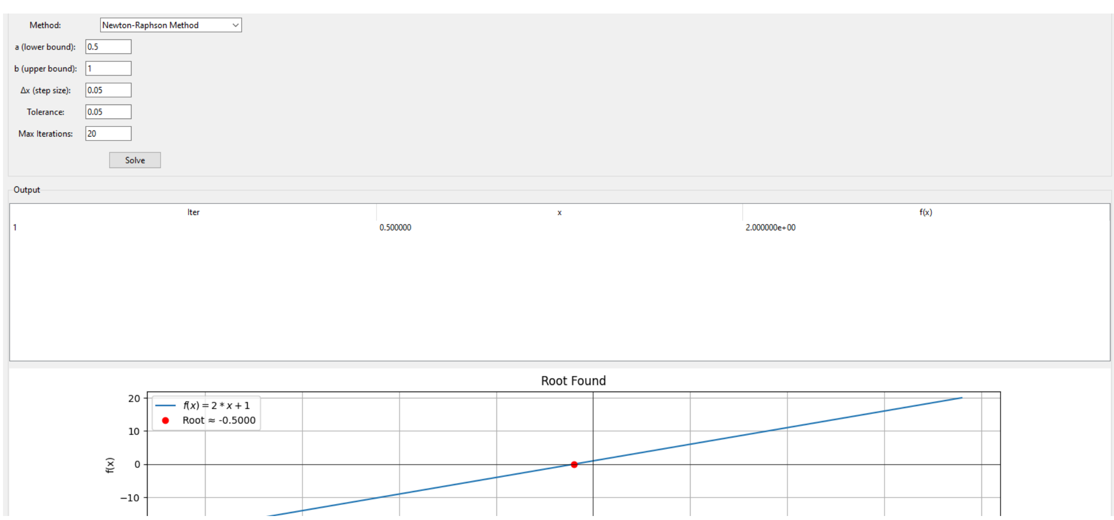
ELEMENTARY: TAGBILARAN GRACE CHRISTIAN SCHOOL
S.Y. 2010 – 2016

Documentation









FUTURE DEVELOPMENT

Looking forward, several enhancements and extensions can be implemented to improve the functionality and usability of the Numerical Methods Root Finder application:

- **Multiple Variable Support**: Enable the solution of multivariable equations or systems of nonlinear equations.
- **Additional Numerical Methods**: Add support for methods like Fixed Point Iteration, Muller's Method, and Brent's Method.
- **Step-by-Step Walkthrough Mode**: Allow users to proceed through iterations manually for instructional purposes.
- **LaTeX Equation Rendering**: Provide real-time LaTeX-style rendering of the entered function to help users verify their input.
- **Export Features**: Enable saving of iteration tables to CSV, and graphs to image files or PDFs for documentation.
- **Theming and Accessibility**: Improve the GUI with light/dark themes, keyboard navigation, and screen reader support.
- **Performance Optimization**: Introduce multiprocessing for large-scale equation analysis or real-time iteration feedback.
- **Cross-Platform Packaging**: Package the app into standalone executables for Windows, macOS, and Linux.

These enhancements will further align the tool with professional standards while preserving its educational accessibility.

PROJECT COST

This project cost was as follow;

Glossary

Term	Definition
Root	A solution to the equation f(x) = 0.
GUI	Graphical User Interface; the visual interface for user interaction.
Tkinter	Python's standard GUI library used to build the interface.
SymPy	A Python library for symbolic mathematics.
`lambdify()`	Converts SymPy symbolic expressions to numeric functions.
`sympify()`	Converts a string expression into a SymPy-compatible symbolic form.
Tolerance	The acceptable margin of error for root approximation.
Max Iterations	The maximum number of iterations an algorithm will perform.
Incremental Method	Finds sign changes in small steps over an interval.
Bisection Method	Root-finding method using interval halving.
Regula Falsi Method	Uses a secant line between two points to find root approximations.

Newton-Raphson Method	Uses derivatives to iteratively improve root estimates.
Secant Method	Similar to Newton-Raphson but uses finite differences.
Treeview	Tkinter widget used for displaying iteration tables.
`FigureCanvasTkAgg`	Embeds matplotlib plots into a Tkinter window.
`DoubleVar`, `StringVar`, etc.	Tkinter variable types that dynamically bind widget values.

Bibliography

1. Burden, R. L., & Faires, J. D. (2011). *Numerical Analysis* (9th ed.). Brooks/Cole Cengage Learning.
2. Chapra, S. C., & Canale, R. P. (2015). *Numerical Methods for Engineers* (7th ed.). McGraw-Hill Education.
3. SymPy Development Team. (n.d.). *SymPy Documentation*. https://docs.sympy.org
4. Python Software Foundation. (n.d.). *Tkinter – Python interface to Tcl/Tk*. https://docs.python.org/3/library/tkinter.html
5. Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. *Computing in Science & Engineering*, 9(3), 90–95.
6. NumPy Developers. (n.d.). *NumPy Documentation*. https://numpy.org/doc/
7. Matplotlib Developers. (n.d.). *Matplotlib Documentation*. https://matplotlib.org/stable/contents.html