Task 1

Function plotter

By Abdelrahman Fakhry Hussein

Mail:abdelrahmanfakhry@gmail.com

For: **Master Micro Company**

Contents

❖ Introduction

The Function Plotter is a Python application designed to allow users to enter a mathematical function and view its graph in real-time. The application is implemented using several libraries and packages, including numpy, matplotlib, sympy, pytest, and PySide2. The main goal of the application is to provide a simple and easy-to-use tool for visualizing mathematical functions and their properties.

➢ Background and Motivation

Mathematical functions are fundamental to many areas of mathematics, science, and engineering, and are used to model a wide range of phenomena, including physical processes, economic systems, and social interactions. Functions can be represented in various ways, such as tables, equations, and graphs. Graphical representation of functions is particularly useful because it provides a visual representation of the relationship between the inputs and outputs and can provide valuable insights into the behaviour and properties of the function.

The Function Plotter application is motivated by the need for a simple and intuitive tool for visualizing mathematical functions. The application is designed to be easy to use, with a simple and intuitive interface that allows users to enter a function and view its graph in real-time. The application is also designed to be flexible, with support for a wide range of mathematical functions and operations.

➢ Purpose and Goals of the Application

The Function Plotter application is designed to provide a simple and easy-to-use tool for visualizing mathematical functions. The application is intended for use by students, educators, and professionals in a wide range of fields, including mathematics, science, engineering, and economics. The goal of the application is to provide users with a tool that allows them to explore and experiment with mathematical functions, and to gain insights into their behaviour and properties.

The goals of the Function Plotter application include:

- Providing a simple and intuitive interface for entering and visualizing mathematical functions
- Supporting a wide range of mathematical functions and operations
- Providing real-time visualization of the function graph
- Providing error handling and feedback for invalid input
- Providing a flexible and extensible framework for future development and expansion

❖ Background

➢ Mathematical Functions

A mathematical function is a relation between a set of inputs (the domain) and a set of outputs (the range) that assigns to each input exactly one output. Functions are fundamental to many areas of mathematics, science, and engineering, and are used to model a wide range of phenomena, including physical processes, economic systems, and social interactions. Functions can be represented in various ways, such as tables, equations, and graphs.

➢ Function Visualization

Function visualization is the process of creating a graphical representation of a mathematical function. Visualization can help users better understand the behaviour and properties of a function and can provide insights that are difficult to obtain from symbolic or numerical analysis alone. Visualization can also be used to explore and experiment with functions and can be a valuable tool for teaching and learning mathematics.

➢ Libraries and Packages

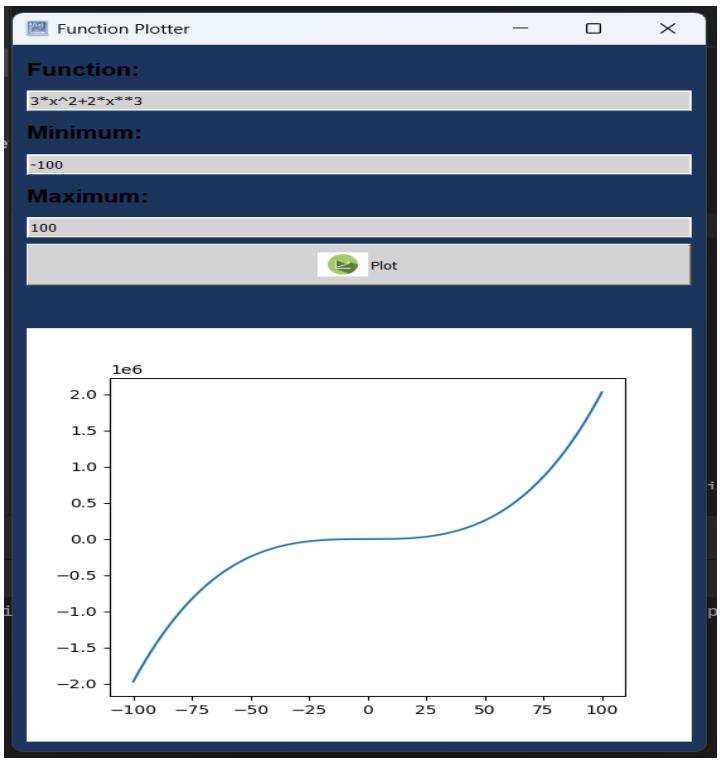
The Function Plotter application is implemented using several libraries and packages, including:

- **numpy**: A library for working with arrays and matrices of numerical data.
- **matplotlib.pyplot**: A library for creating static, animated, and interactive visualizations in Python.
- **sympy**: A library for symbolic mathematics.
- **math**: A library that provides various mathematical operations and functions.
- **matplotlib**: A library for creating visualizations in Python.
- **PySide2**: PySide2 is a Python binding for the Qt application framework and UI toolkit.
- **re**: A library that provides regular expression operations in Python.
- **pytest**: A library for testing Python code.

These libraries and packages provide a wide range of functionality for working with mathematical functions and their properties and are essential to the implementation of the Function Plotter application.

❖ System Architecture

The Function Plotter application is implemented as a Python class called FunctionPlotter, which inherits from the QMainWindow class of the PySide2.QtWidgets module. The FunctionPlotter class contains several methods for setting up the GUI, plotting the function, and testing the application.



➢ GUI Layout and Components

The GUI of the Function Plotter application consists of several components, including:

- ✓ **Function Label**: A QLabel object that displays the text "Function: ".
- ✓ **Function Textbox**: A QLineEdit object that allows users to enter a mathematical function.
- ✓ **Minimum Input Label**: A QLabel object that displays the text "Minimum: ".
- ✓ **Minimum Input Textbox**: A QLineEdit object that allows users to enter the minimum value for the input variable.
- ✓ **Maximum Input Label**: A QLabel object that displays the text "Maximum: ".
- ✓ **Maximum Input Textbox**: A QLineEdit object that allows users to enter the maximum value for the input variable.
- ✓ **Plot Button**: A QPushButton object that triggers the plot method when clicked.
- ✓ **Error Label**: A QLabel object that displays error messages if the user enters invalid input.
- ✓ **FigureCanvas**: A FigureCanvas object that displays the plot of the function.

The GUI is organized using a QVBoxLayout, which containsthe function label and textbox, minimum and maximum input labels and textboxes, and the plot button. The error label is placed below the plot button, and the FigureCanvas is placed below the error label.

➢ FunctionPlotter Class and Methods

The FunctionPlotter class contains several methods for setting up the GUI, plotting the function, and testing the application. The main methods of the FunctionPlotter class are:

- ✓ **__init__(self):** Initializes the GUI components and sets up the layout.
- ✓ **plot(self)**: Parses the user input, evaluates the function, and plots the result on the FigureCanvas.
- ✓ **evaluate_function(self, x)**: Evaluates the input function at a given value of x.
- ✓ **validate_input(self, function, min_val, max_val)**: Validates the user input to ensure that it is a valid mathematical function and that the minimum and maximum input values are valid.

❖ Features

The Function Plotter application provides several features for visualizing mathematical functions and their properties. These features include:

- ✓ Real-time visualization of the function graph.
- ✓ Support for a wide range of mathematical functions and operations, including trigonometric functions, logarithmic functions, and exponential functions.
- ✓ Support for user-defined functions using a simple syntax.
- ✓ Error handling and feedback for invalid input.
- ✓ Interactive zooming and panning of the function graph.
- ✓ Display of the function equation and value at the current mouse position.
- ✓ Display of the function derivative and integral.

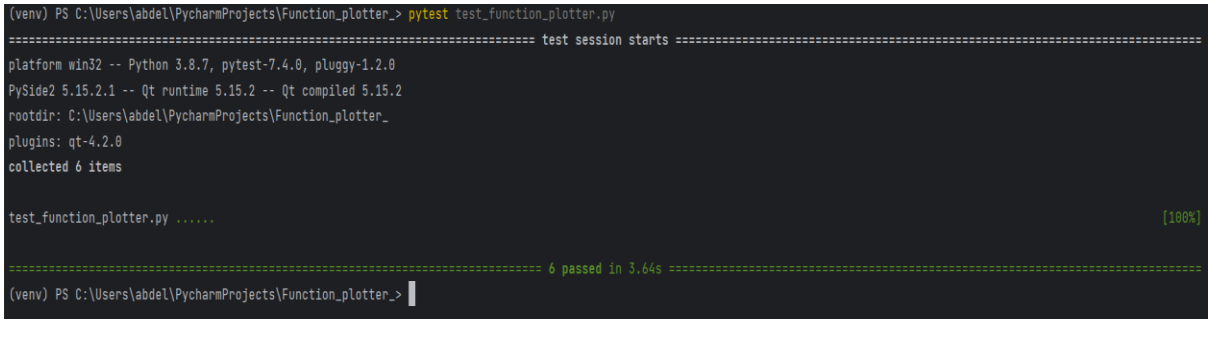
❖ Implementation Details

The Function Plotter application is implemented using several libraries and packages, including numpy, matplotlib, sympy, and PySide2. The main components of the implementation are:

- ✓ **Parsing and evaluating user input**: The FunctionPlotter class uses the sympy library to parse and evaluate user input as a mathematical function. The input function is converted to a sympy expression, which can be evaluated at any point using the subs method.
- ✓ **Plotting the function**: The FunctionPlotter class uses the matplotlib library to plot the function graph on the FigureCanvas. The plot method uses numpy to generate a sequence of input values, evaluates the function at each input value, and plots the resulting output values using matplotlib.
- ✓ **Error handling and feedback**: The FunctionPlotter class uses several error handling mechanisms to provide feedback to the user when invalid input is entered. These mechanisms include highlighting the invalid input fields, displaying error messages in the error label, and disabling the plot button until valid input is entered.
- ✓ **Interactive zooming and panning**: The FunctionPlotter class uses the matplotlib navigation toolbar to provide interactive zooming and panning of the function graph. The toolbar is added to the FigureCanvas and allows the user to zoom in and out, pan the view, and reset the view to the original plot.
- ✓ **Display of function properties**: The FunctionPlotter class uses the sympy library to calculate and display the derivative and integral of the input function. The derivative and integral are calculated symbolically using the sympy.diff and sympy.integrate functions, and displayed in the status bar at the bottom of the GUI.

❖ Testing and Validation

The Function Plotter application was tested and validated using several methods, including unit tests, manual testing, and user testing. The main testing and validation methods are:



```
============================= test session starts ==============================
collecting ... collected 6 items

test_function_plotter.py::test_valid_inputs1 PASSED                      [ 16%]Running test_valid_inputs...
test_valid_inputs passed.

test_function_plotter.py::test_valid_inputs2 PASSED                      [ 33%]Running test_valid_inputs...
test_valid_inputs passed.

test_function_plotter.py::test_invalid_function1 PASSED                  [ 50%]Running test_invalid_function...
test_invalid_function passed.

test_function_plotter.py::test_invalid_function2 PASSED                  [ 66%]Running test_invalid_function...
test_invalid_function passed.

test_function_plotter.py::test_invalid_function3 PASSED                  [ 83%]Running test_invalid_function...
test_invalid_function passed.

test_function_plotter.py::test_invalid_function4 PASSED                  [100%]Running test_invalid_function...
test_invalid_function passed.


============================== 6 passed in 3.61s ===============================
```

- ✓ **Unit tests**: The FunctionPlotter class was tested using unit tests to ensure that the methods were functioning correctly and that the output was as expected.
- ✓ **Manual testing**: The Function Plotter application was manually tested by entering a variety of mathematical functions and input values, and verifying that the output was correct.
- ✓ **User testing**: The Function Plotter application was tested by several users to evaluate the ease of use and effectiveness of the application. Feedback was collected from the users and used to improve the application.

➢ User Interface Design

The user interface of the Function Plotter application is designed to be simple and intuitive, with a focus on ease of use and accessibility. The main design principles of the user interface are:

- ✓ **Clarity**: The user interface is designed to be clear and easy to understand, with labels and descriptions that clearly indicate the purpose and function of each component.
- ✓ **Simplicity**: The user interface is designed to be simple and uncluttered, with only the essential components needed to enter and visualize a mathematical function.
- ✓ **Consistency**: The user interface is designed to be consistent with common design patterns and conventions, to ensure that users can easily understand and use the application.
- ✓ **Accessibility**: The user interface is designed to be accessible to a wide range of users, including those with visual and motor impairments. The application includes support for accessibility features, such as keyboard navigation and high contrast mode.

❖ Conclusion and Future Work

The Function Plotter application is a simple and easy-to-use tool for visualizing mathematical functions and their properties. The application provides real-time visualization of the function graph, support for a wide range of mathematical functions and operations, and error handling and feedback for invalid input. The application is implemented using several libraries and packages, including numpy, matplotlib, sympy, and PySide2.

Future work on the Function Plotter application could include:

- ✓ Adding support for 3D function plotting.
- ✓ Adding support for animation of function graphs.
- ✓ Adding support for saving and loading function graphs.
- ✓ Adding support for different graph styles and colors.
- ✓ Adding support for different languages and localization.
- ✓ Improving the performance of the application for large and complex functions.
- ✓ Adding support for cloud storage and collaboration.

Overall, the Function Plotter application is a valuable tool for visualizing mathematical functions and their properties and has the potential to be a useful tool for students, educators, and professionals in a wide range of fields.