CAD Laboratory: Module II

PROJECT REPORT

GRAPH PLOTTER

Submitted By
Deepu George Kurian
121701006

Objective:

To code and implement in python, a graph plotting application from user inputs (using the Shunting-Yard Algorithm).

Python Tools Required:

1. Spyder3 IDE for python3 programming.
2. Basic python libraries such as Matplotlib, NumPy and SciPy.
3. Libraries such as PyQt5 for the interactive user interface.
4. Basic plot functions from matplotlib.pyplot.
5. mpl_toolkits.mplot3d for plotting 3d graphs.

Introduction:

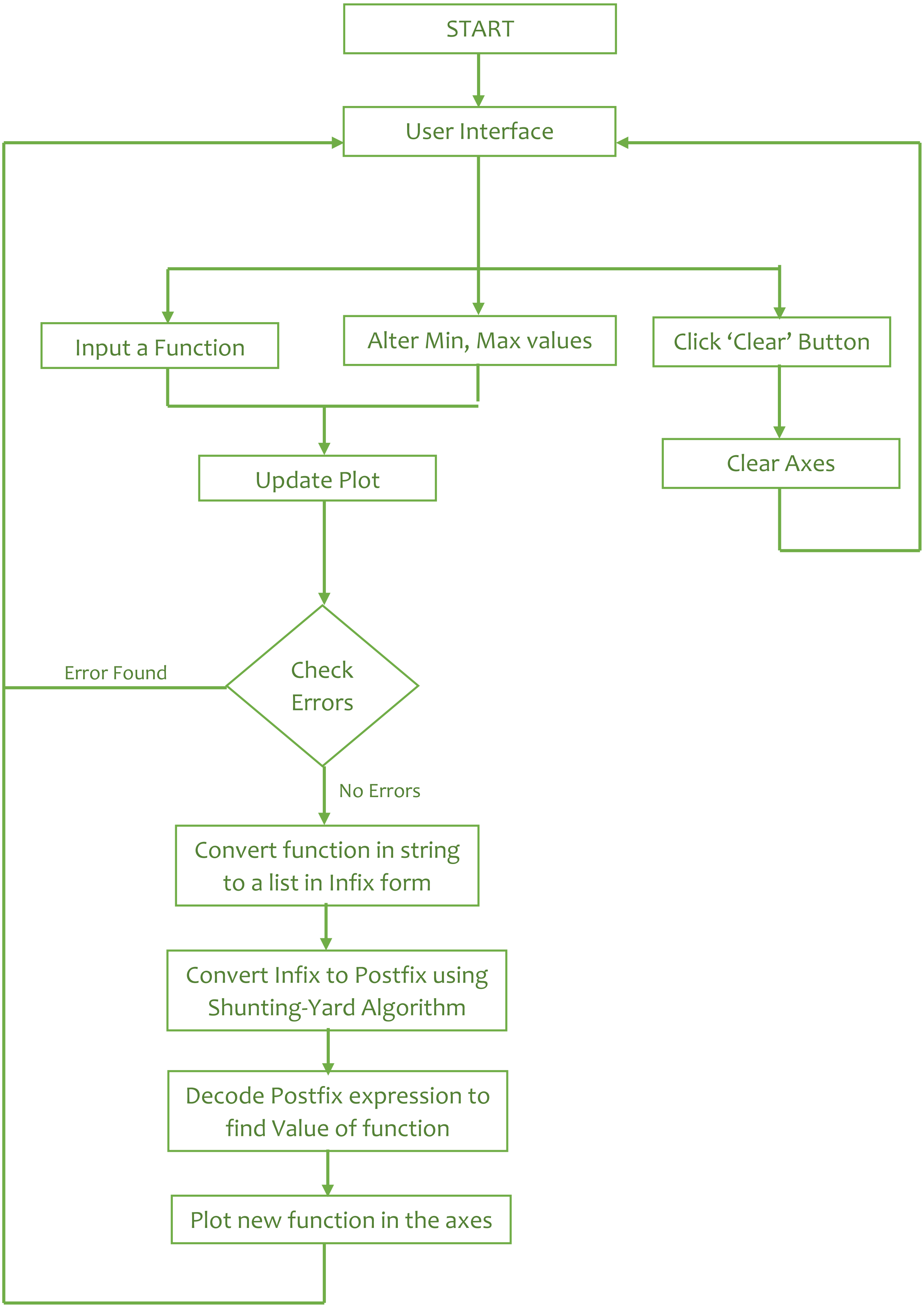
The above-proposed program allows us to visualise any function in one / two variables in a graphical 2D/3D domain. Several graphs can be plotted on the same axes for comparing between different functions. The limits of the domain can also be determined dynamically by the user.

The user inputted expressions can have a wide variety of functions such as exponential, logarithmic, differential, trigonometric, inverse trigonometric, hyperbolic, inverse hyperbolic etc.

There's also an efficient method to avoid run time errors due to illegal user inputs. An end-user without any programming knowledge can easily utilise this work to plot any arbitrary graph using a user-friendly interface. This can be used for the development of a graph plotting application.

Program Logic:

Block Diagram


START
User Interface
Input a Function
Alter Min, Max values
Click 'Clear' Button
Update Plot
Clear Axes
Error Found
Check Errors
No Errors
Convert function in string to a list in Infix form
Convert Infix to Postfix using Shunting-Yard Algorithm
Decode Postfix expression to find Value of function
Plot new function in the axes

Upon running the code, the user will be prompted to give a function in one / two variables. But the inputted function is read as a string and can't be processed as such. The string data is converted to a list where each element is either a numerical value, a variable (x or y), parenthesis or an operator.

The list thus obtained is in infix format which is most commonly used. But a computer system can't process from an Infix expression and thus it has to be converted to its postfix equivalent expression. In Infix expression, the binary operator comes in between the operands. While in Postfix expression operands are written first, then the operator. This Infix to Postfix conversion is done by the Shunting Yard Algorithm. This algorithm takes care of the precedence of different operators.

A different algorithm is used to find the value of the function for various values of x, y. Stack operations are used to decode the postfix expression and hence find the value of the expression.

Source Code:

The whole program is divided into four modules which work together to give a user-friendly output window for plotting and comparing functions.

1. Module: Code

In this module, the inputted string is checked for correctness (eg: if the parenthesis is balanced) and processed to return the postfix expression of the function.

2. Module: Decode

Here the processed Postfix expression, and range of values of x, y is taken as input and the corresponding value range of the function is returned.

3. Module GUI_2D

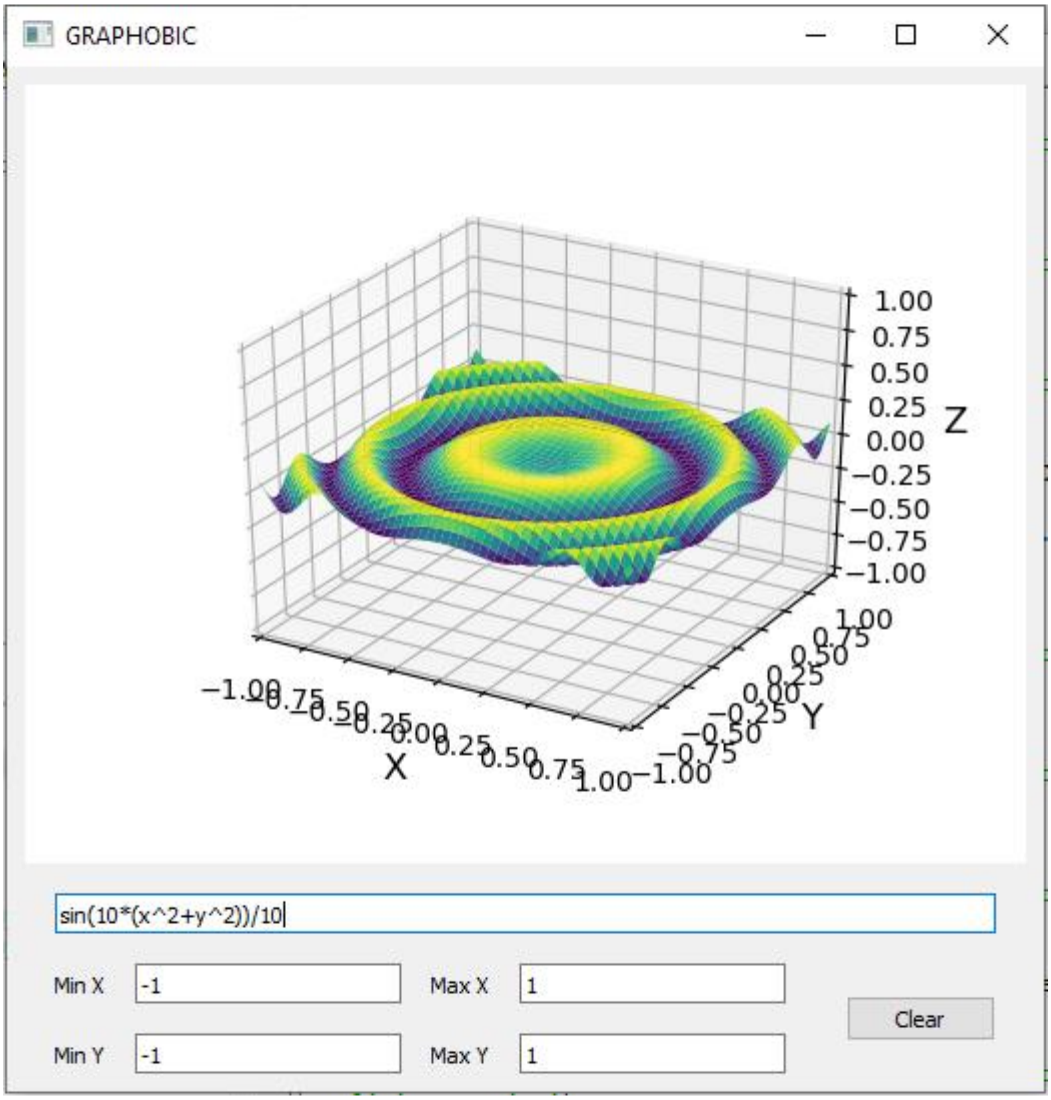
The run file for 2D plot requirements that creates a graphical interface with two-dimensional axes and incorporates other modules for execution.

4. Module GUI_3D

The run file for 3D plot requirements that creates a graphical interface with three-dimensional axes and incorporates other modules for execution.

Results:

A user-friendly output window is designed.



sin(10*(x^2+y^2))/10

Min X -1

Max X 1

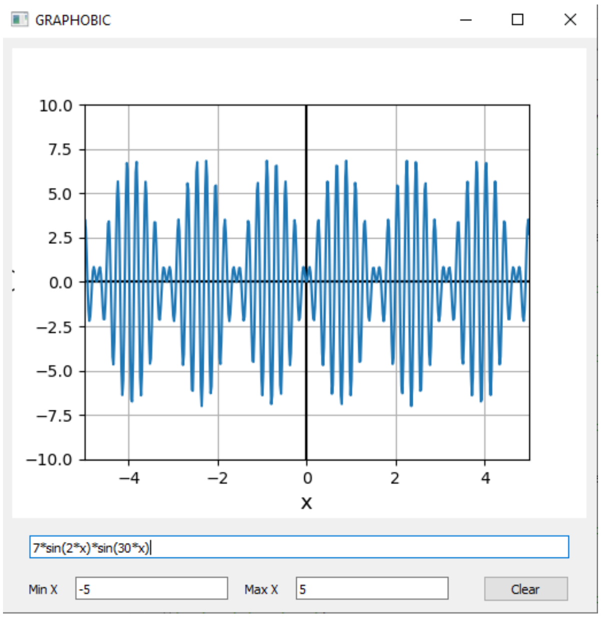
Min Y -1

Max Y 1

1. The x and y limits of the graph plotted can be altered dynamically in the output window by the user.
2. Multiple graphs can be plotted on the same axes to compare different functions.

3. There's a "clear" button in the window to allow the user to clear the axes and plot a new set of functions.
4. Both 2D and 3D graphs could be plotted.
5. The code also has error detection and management system.

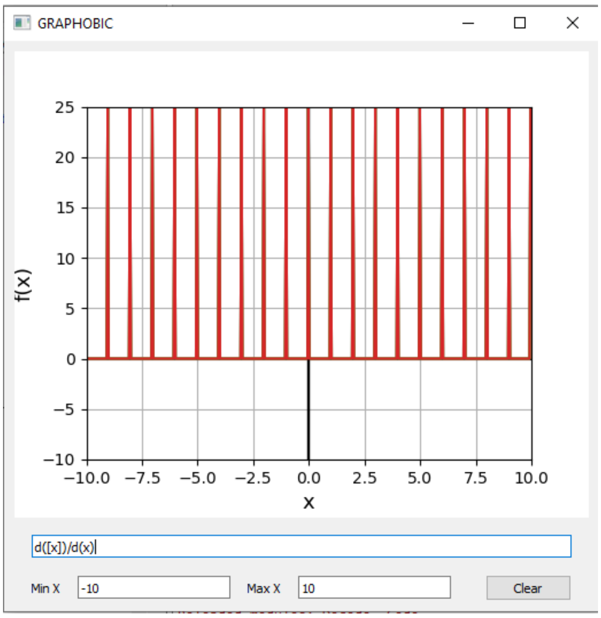
A few more examples of graphs of some functions are given here.



7*sin(2*x)*sin(30*x)

Min X -5

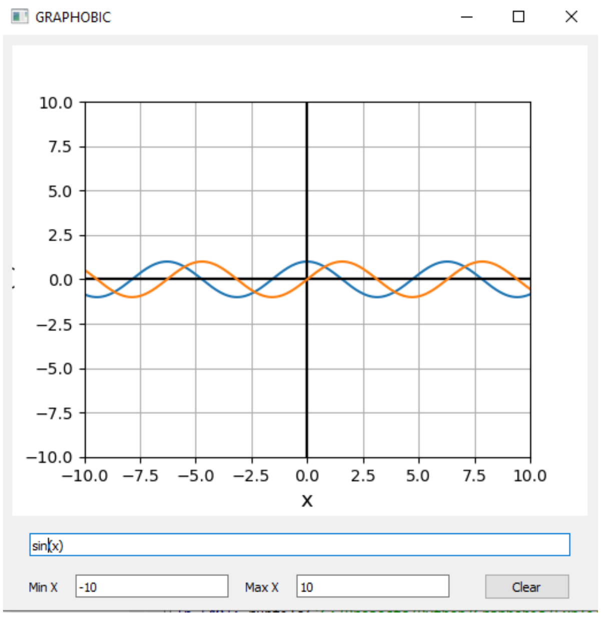
Max X 5



d([x])/d(x)

Min X -10

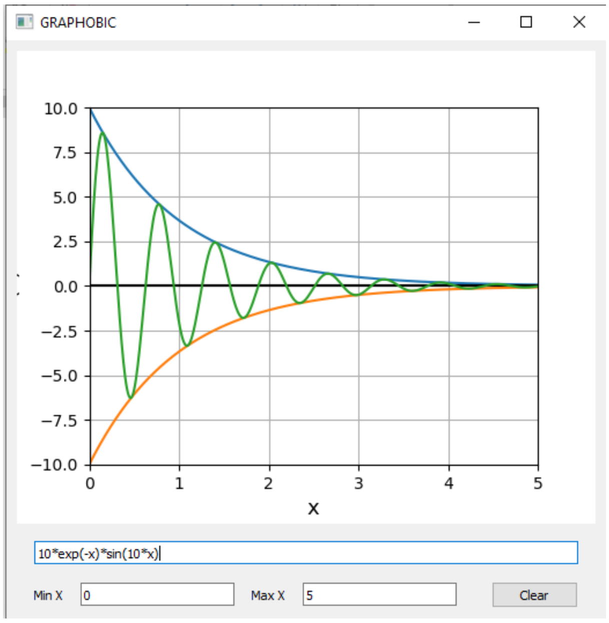
Max X 10



sin(x)

Min X -10

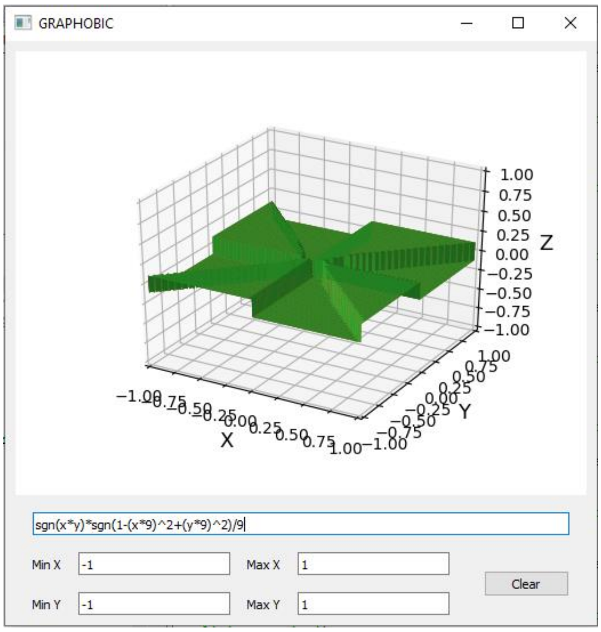
Max X 10



10*exp(-x)*sin(10*x)

Min X 0

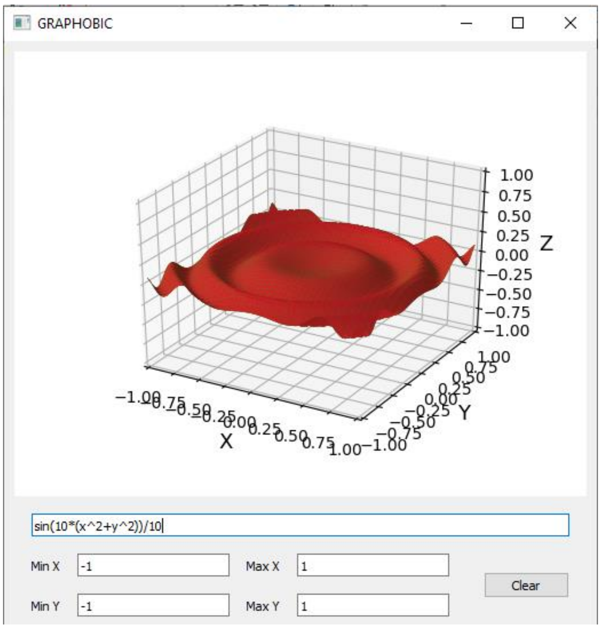
Max X 5



sgn(x*y)*sgn(1-(x*9)^2+(y*9)^2)/9

Min X -1 Max X 1

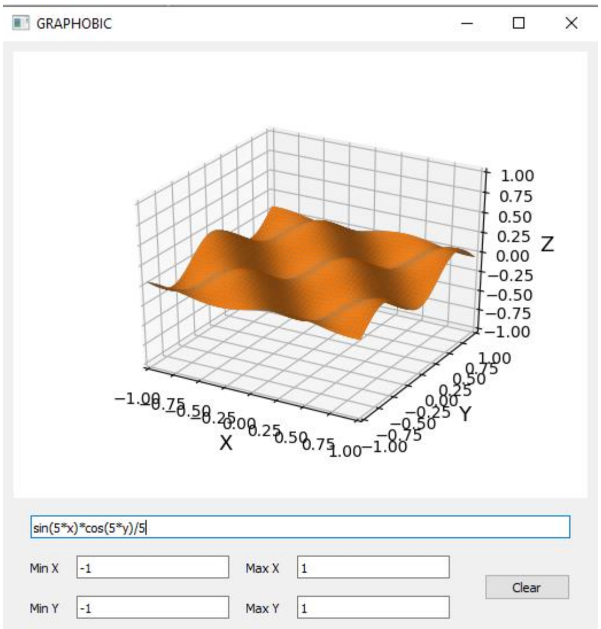
Min Y -1 Max Y 1



sin(10*(x^2+y^2))/10

Min X -1 Max X 1

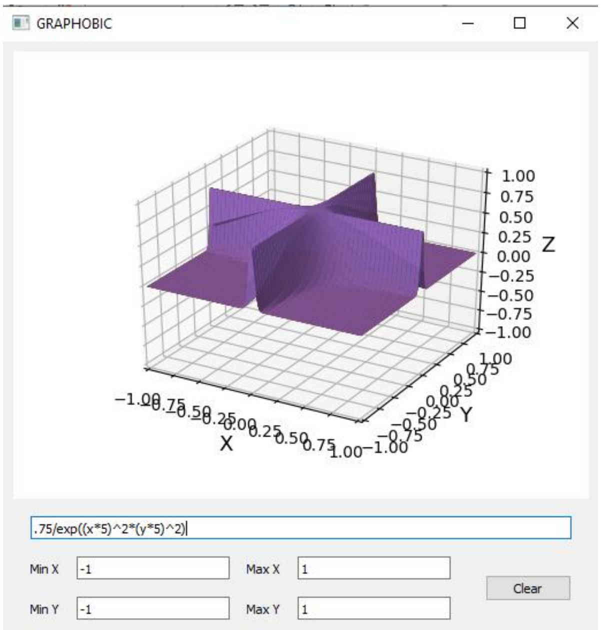
Min Y -1 Max Y 1



sin(5*x)*cos(5*y)/5

Min X -1 Max X 1

Min Y -1 Max Y 1



.75/exp((x*5)^2*(y*5)^2)

Min X -1 Max X 1

Min Y -1 Max Y 1

Conclusions:

1. Comparison of different functions is made easier and available for users without any programming knowledge.
2. This program can be modified further to a graph plotting application.