**Final Project Report**

For my CSCE-155N final project, I have chosen to create a vector field plotter. It consists of a graphical user interface (GUI) that allows the user to input 2-D vector field components and visualize the vector field through the graph. This is especially useful for engineers as explained in the later parts of this report. The report will go over the coding process, design difficulties, and the future applications of this program.

The coding topic chosen was approved by Migule Moreno Tenorio before the coding process started. The first step was creating the text elements of the GUI using the text UI control element. Titles, basic instructions, and examples were added to the interface. After that, edit boxes were added to receive user input using the edit box UI control. The 'u' and 'v' components of the vector field will be the string the user will input into the interface. Translating the user string input into an equation required the use of a call back function. In this call back function, a useful MATLAB built-in function called ‘eval ( )’ takes in a string and evaluates it into an equation. Both x and y were defined as variables using the ‘syms’ command. Two separate call back functions were then created to convert the user u and v inputs into vector field equations. Utilizing the outputs from the first two call back functions, a third call back function which was linked to the pushbutton named ‘plot’ was designed to use the equations to plot the velocity field. The third call back function, ‘graphPlot’, was the main element in this code. To plot the vector field, a mesh grid was first defined and given an appropriate scale to allow the user to see the vectors clearly. The quiver function was then utilized to plot each vector component on the mesh grid, displaying the direction and magnitude of each vector component. Adjustments were later made to the graph size on the figure by using the parent field of the plot function to access its position field.

Through the coding process, three major struggles were encountered which were eventually resolved through research and trial & error. The first main obstacle was finding a way to extract the user input from the edit box. In class, we only have dealt with receiving user input in the form of events such as pushing/toggling a certain button but not typing in a string. To achieve this, two variables need to be called at the input of the call back function which is ‘handles’ and ‘hObject’ which essentially gives you access to the user input in the edit box. The built-in command get( ) was then used to ‘get’ the string from the edit box. The second main struggle was changing the size of the graph in the figure. This could usually be done by accessing the parent field of the plot function and then accessing the position of that parent field. However, this did not seem to work for the quiver function. MATLAB produced an error message that read, “Parent field does not exist for quiver”. The solution to this problem was defining an empty plot figure in the main function and changing the position of that plot figure using the method described above. When the actual quiver function was called with all its variables in the graphPlot call back function, it seems to have populated the properly position empty figure that was defined in the main figure with the vector field graph. This part of the code took a lot of trial and error as the MATLAB documentation did not have any information to do something this specific. The last challenge was identifying incorrect user input and generating an error message for the user. Generating the error message was relatively simple but identifying the error was the main obstacle. When the user inputs either a non-equation format string such as “jowjfs” or an equation without the ‘\*’ symbol as the multiplication symbol, the eval() command on MATLAB crashes. According to Ms. Lanik, this would require writing a parser which is beyond the scope of this course; hence, this criterion in the rubric was omitted in this code.

Vector field visualization is very important in the field of physics and engineering. Electric fields, magnetic fields, and fluid flows are all represented in the form of vector fields. This GUI serves the purpose of helping engineers readily visualize the vector fields to observe their interesting features. For instance, certain patterns shown in a fluid flow field can tell a fluid engineer how the particular fluid is behaving. To further emphasize the importance of fluid field visualization through GUIs, the following given velocity field is plotted.

Chart

Description automatically generated

Figure 1

As seen in figure 1, the resulting vector field exhibits a swirling pattern. In terms of fluid mechanics, this is called rotational flow. Knowing the rotationality of a fluid flow is an important factor to consider in fluid mechanics-related applications. In general, merely having the vector field equation does not tell us much about the vector field. By plotting and visualizing the vector field equation, we are aware of different types of patterns in the vector field as the one shown above. This concept can be applied to electric fields and magnetic fields too.

Conclusively, this project was a challenging and enjoyable experience. The MATLAB documentation found online would be the main source of reference for this project as well as the help of Ms. Lanik.