I wrote code to display the process of Euler’s Method in an animation involving a scroll bar. The concept of Euler’s method is quite simple and involves a basic understanding of calculus. Essentially it is a way to approximate the particular solution to any Ordinary Differential Equation, or ODE for short. Essentially you use the equation dy/dx, which gives the slope of the unknown equation y at any position. Every ‘step’ of Euler’s Method uses the slope dy/dx and uses it to approximate the next point on the graph. With very few step sizes, this approximation is a bad one, but improves when you change it. The variable that you change is essentially the number of steps in between each point on the graph. Between x =0 to x =3 in the user interface, the first step size is just one whole number, which is not a good approximation. When you evaluate the slope over .01 steps per whole number of x, it becomes an incredible accurate approximation.

The main function just simply establishes the user interface and fills it with all the objects that will be used. There is a plot, a scroll bar, and two different buttons with different purposes. The plot takes up the majority of the space in the pop up window and displays a red graph of the function y = e^x from x=0 to x=3. If you move the scroll bar manually, it shows a blue graph under the red, which shows the graph according to Euler’s Method. The different step sizes are 1, .5, .25, .1, .01, which is also displayed in the command window. As you can see, the .01 step size is remarkably close to the real shape. I also implemented a feature which runs through the steps automatically, with the button reading ‘Play Animation’. Some other functions I included was the learnEuler function, which outputs a message box further explaining Euler’s Method. The scroll bar function itself is very important to the code. It uses logical statements that read values from a for loop, which tells it to display a certain graph.

I had a few struggles with getting this code to work properly. At first, I wanted to write this code without needing an external data sheet. As I began writing it, I realized that it would make the code overly complicated and the whole file would take up 100+ lines of code. In a separate MATLAB file, I used code to automatically compute the values from Euler’s Method, which is a lot less time consuming than computing Euler’s Method by hand, especially when the smallest step size would have needed around 60 complete computations of Euler’s Method, which is made incredibly trivial with for loops. My initial plan to have all of the computations in one file was made a lot more difficult because I originally planned to have two Euler’s Method problems that the user could switch between, but that turned out to be too tedious. I planned to include a button group in the figure, but since I reduced to only one problem it wasn’t fully needed. I tried desperately to get the code to run after removing the button group and it just wasn’t working. After a while I realized that I didn’t need a .m file with 6 functions in it, and I reduced it to 4. The problem was me having too many functions.

I am not entirely sure what I might use GUI’s for in the future, but I know I will continue to use MATLAB as it relates to my major of Mechanical Engineering. It makes tedious problems that require immense brain power simple with the power of computers. I haven’t worked on it yet, but I am considering writing some sort of function that solves Ordinary Differential Equations automatically. This is just an idea, but hopefully I use code for a future job or the rest of my college major. I can even include code knowledge to any job resumes that I write, as it is a valuable skill to understand the basic concepts of coding languages.

This project was approved by Quinn Lanik.