Chris Ricchi

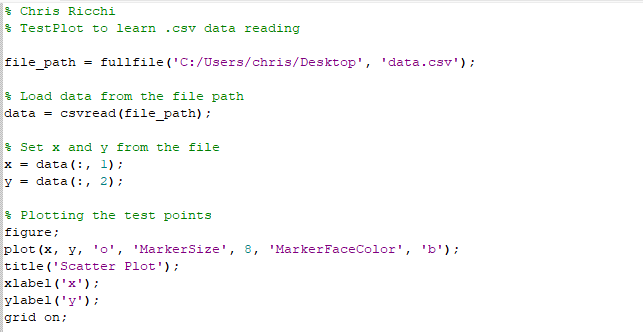
12-9-2023

Professor Hoy

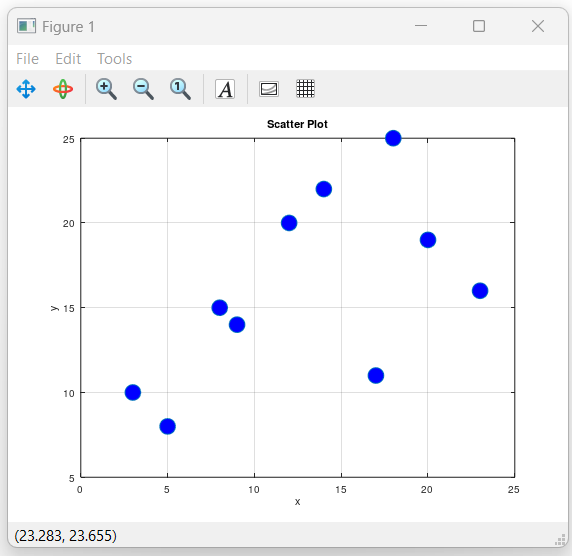
**Learning Octave**

Octave is a powerful open-source computing tool that allows users to use numerical analysis and statistical computation seamlessly without the need for extensive programming. As someone who is already familiar with Octave, I focused on a specific point for my tutorial which is fundamental for statistics; reading a dataset and graphing it. Despite the simplicity of graphing, learning how it works in Octave is pivotal in creating accurate visual representations of real world information. This paper focuses on my step-by-step process, from beginning to end, detailing the specifics of graphing points works in Octave; from loading the data from a comma-separated list to visualizing it and performing calculations on the set itself.

Before working on my final program to handle salting and smoothing data, I began scanning the official Octave documentation for a function to process comma-separated data, as I already have extensive experience plotting in Octave and MATLAB. According to the official documentation, the function *csvread* can be used to process comma-separated values and store them into matrix X. Before attempting to read any external data, I created a sample dataset to have a handful of variables to begin test plotting, called *data.csv.* Then, I wrote a very simple script called *TestPlot.m* that reads the .csv from a local path, sets the correct columns to local variables x and y, and plotting them in a simple scatter plot.

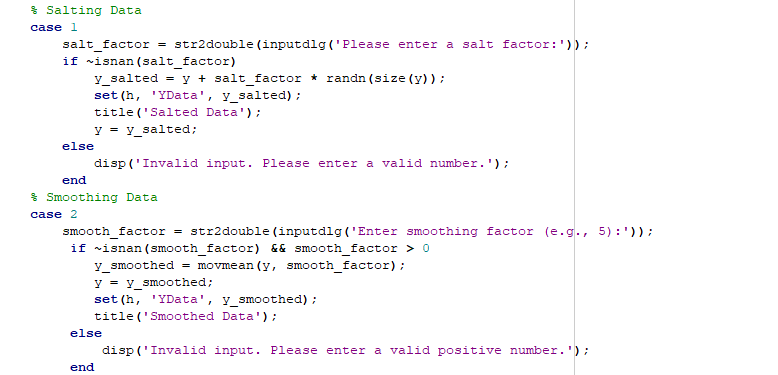


In the TestPlot.m, I started by initializing file\_path to the desktop of my local machine and tell it to search for *data.csv*. After the .csv is loaded, I used the *csvread* function from the official documentation and set x and y from the data by using the parameter data(:, 1) and data(:, 2) which separates each column into x and y respectively. Finally, I created a simple plot by initializing a figure and plotting x and y using the plot function and enabling the grid. Below is the outputted plot from the example data:

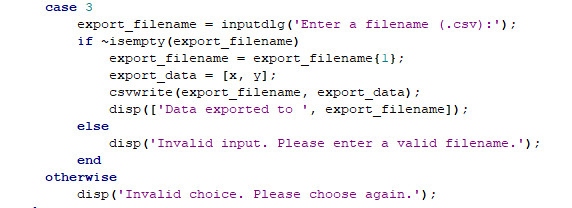


Now that the data is correctly being read from the .csv file and plotted, the script *TestPlot.m* has served its full purpose and I can move on to the salting and smoothing program, which we will call *DataSmoothing.m*. The first thing I wanted to implement was a way for the user to choose from a list of options to alter the graph: salting, smoothing, exporting, and exiting. I found out that Octave has a *menu* function on their documentation. This menu displays user options and waits for input. If we tie this with a simple while loop, then we would be able to allow users to continuously edit the graph through the menu until they want to exit.

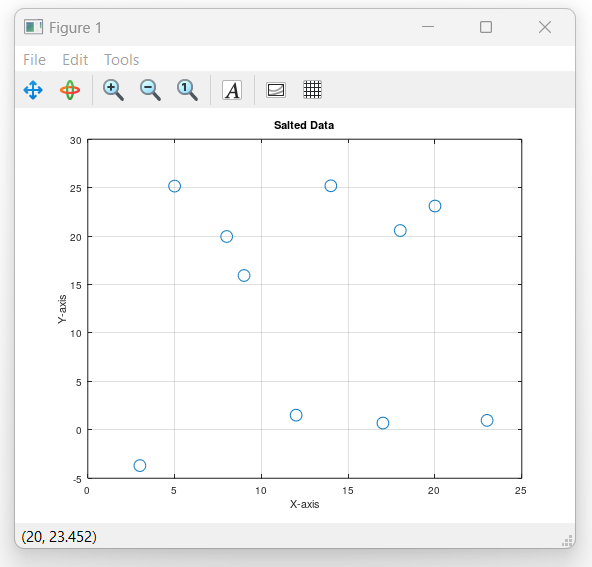
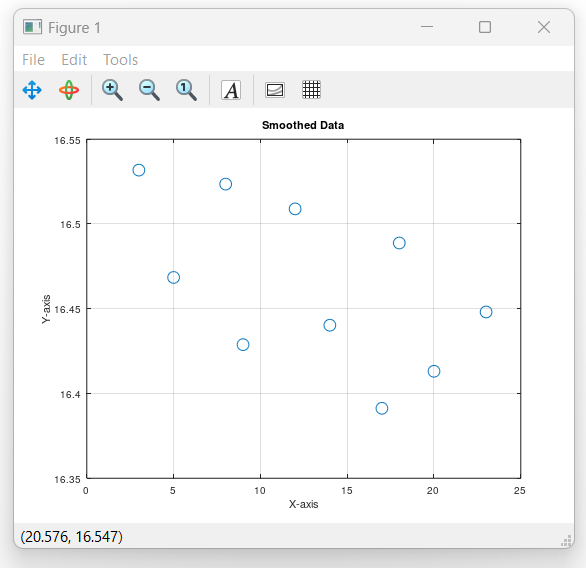
Octave has a function called *inputdlg* which, according to the documentation, returns user input from a dialog box. To perform both salting and smoothing operations, we first have to ask the user by what factor they want to either expand or reduce the points on the graph. For both salting and smoothing, we will have to ask the user to enter a salting or smoothing factor to multiply by. When taking the factor we first need to check it is a real number by running the *~isnan* function, which is the result when the conversion to a double fails. Assuming the factor is valid, we can multiply the salt factor by a random number of size X for the x coordinate point, and by a random number of size Y for the y coordinate point. To smooth the points, we can use the function *movemean* on the coordinates, which calculates the moving mean of a sequence. By using the *movemean* function between the original point and the entered smooth factor, we will see a very simple smoothed result for each point.



The last function we need is an export function which allows us to save our new points to a .csv so we can externally graph it in Excel. According to the documentation, Octave’s *csvwrite* function will do exactly this. We will use an *inputdlg* to ask the user to input a filename and set our points x and y to an array named *export\_data.* Lastly, we will call the *csvwrite* function to write *export\_data* to the local drive.



Using Octave’s official documentation, our program is now a functional salter and smoother, and can read data from a .csv file as well as export the altered data to a new .csv file.



Works Cited

<https://octave.sourceforge.io/octave/function/csvread.html>

<https://octave.sourceforge.io/octave/function/menu.html>

<https://octave.sourceforge.io/octave/function/inputdlg.html>

<https://octave.sourceforge.io/octave/function/csvwrite.html>