Ian Valiante

Part 1-PSS 2

For this portion of the project, I chose MATLAB as the tool for implementation. Through initial research, I found that Octave offers a free alternative to MATLAB, though with some syntax differences. Using the MATLAB tutorial and its extensive documentation, I learned the essential syntax and plotting techniques. Another reason for selecting MATLAB was its resemblance to concepts I encountered in my “Knowledge Discovery and Data Mining” course this semester, where we wrote Python scripts with libraries like “matplotlib” and “NumPy” for data visualization and analysis. These share similarities with MATLAB’s syntax and functionality.

**Original Graph**

The first part of the project involved plotting the quadratic function  f(x) = x^2 + 2x + 1  over a range of  x -values from -10 to 10. Using a MATLAB script, I assigned the variable  y  to the output of the anonymous function  f , with  x  incremented by 0.1 across the range. The resulting graph displayed the smooth, parabolic curve expected of the quadratic function, serving as the reference for subsequent comparisons.

A graph of a function

Description automatically generated

**Salted Graph**

The second part of the project focused on creating a salted graph, which introduced random fluctuations to the quadratic function f(x) = x^2 + 2x + 1. These fluctuations simulated real-world noise, resulting in scattered points that exhibited erratic behavior, including skips and jumps. This graph visualizes the variability and unpredictability of noisy data.

A graph of a graph

Description automatically generated

**Combined Graph**

The third part of the project involved creating a combined graph that overlaid the original, salted, and smoothed data. The original function remained as the parabolic curve, serving as a baseline. The salted data appeared as scattered circles, representing the noisy fluctuations. The smoothed data, shown as a yellow line, was derived from the noisy data and closely followed the original function by reducing the noise. This combined visualization demonstrated how smoothing techniques can restore data to a more consistent and predictable form while preserving its original shape.

A graph of a function

Description automatically generated

**Conclusion**

MATLAB proved to be a powerful and user-friendly tool for these visualizations. Its comprehensive documentation and intuitive syntax enabled me to quickly learn and apply its plotting techniques. By leveraging MATLAB, I created clear and accurate graphs that showcased the distinctions between the clean original function, the erratic salted data, and the refined smoothed results. This project enhanced my understanding of MATLAB’s capabilities while demonstrating the importance of visualization in making data more meaningful and accessible.

Among the 3 tools we used to Plot original data, salt it, smooth it then Graph the results I found MATLAB to be the most user-friendly. I believe anyone could learn how to use this tool, with no prior coding experience. Personally, I do not like using web-apps; but I know for most users this is a nice feature. If I were to personally do any data analysis out of a class context I would use python. Since we are scripting, it is a good choice; and has extensive amounts of libraries to use.