Octave Exercise Report

Christian Simpson

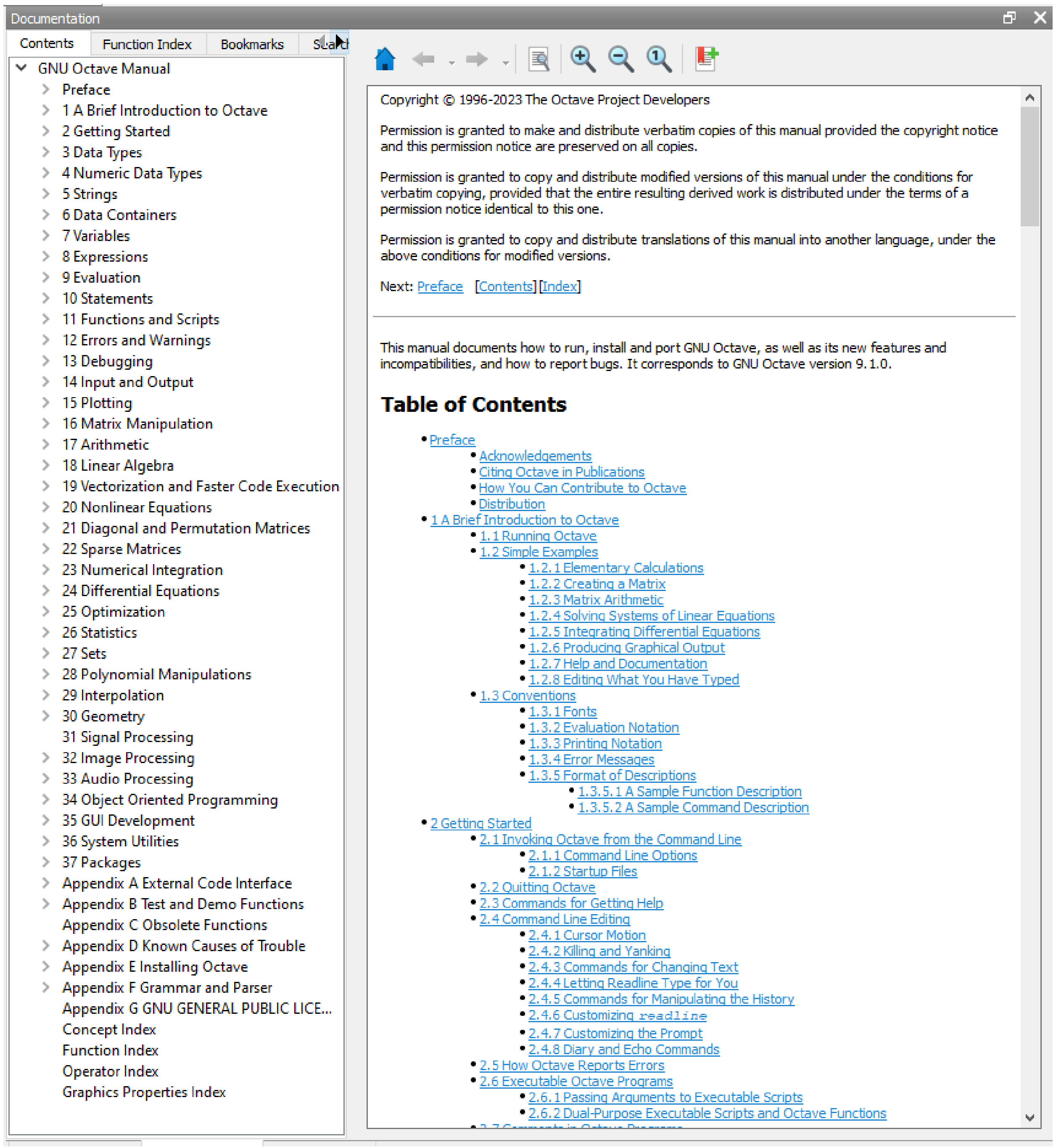
CSCI-3327-001 Probability and Applied Statistics

Byron Hoy

April 29th, 2024

In addition to writing and implementing programs entirely from scratch, we were tasked with another regular skill we'd be expected to employ in a career involving Computer Science: Learning! Sarcasm aside, the goal of this exercise was, when presented with an entirely new program, language, or application, to learn its uses and apply it to another task we were assigned. Specifically, we were tasked with learning MatLab or Octave, and using their supplied methods and functions to again Plot, Salt, and Smooth data. Thanks to the built-in documentation provided with Octave, along with its ability to easily display graphs and charts, this process was intuitive and fairly simple. Follow along to learn my process, and see the results of this foray into a new, unknown program.

Octave is free-to-use software that allows for a multitude of operations in calculating mathematical equations and displaying results. Not only that, though - there are several opportunities for users to implement what they've learned in other programming languages in Octave. Octave allows for user-defined classes and data types, and even allows them to be "hot-loaded" while a program is running. For the basics, I chose to get a handle on Octave's elementary functions using their documentation. This is found in the Help menu (Fig 1.1).



Documentation

Contents | Function Index | Bookmarks | Search

- GNU Octave Manual
  - Preface
  - 1 A Brief Introduction to Octave
  - 2 Getting Started
  - 3 Data Types
  - 4 Numeric Data Types
  - 5 Strings
  - 6 Data Containers
  - 7 Variables
  - 8 Expressions
  - 9 Evaluation
  - 10 Statements
  - 11 Functions and Scripts
  - 12 Errors and Warnings
  - 13 Debugging
  - 14 Input and Output
  - 15 Plotting
  - 16 Matrix Manipulation
  - 17 Arithmetic
  - 18 Linear Algebra
  - 19 Vectorization and Faster Code Execution
  - 20 Nonlinear Equations
  - 21 Diagonal and Permutation Matrices
  - 22 Sparse Matrices
  - 23 Numerical Integration
  - 24 Differential Equations
  - 25 Optimization
  - 26 Statistics
  - 27 Sets
  - 28 Polynomial Manipulations
  - 29 Interpolation
  - 30 Geometry
  - 31 Signal Processing
  - 32 Image Processing
  - 33 Audio Processing
  - 34 Object Oriented Programming
  - 35 GUI Development
  - 36 System Utilities
  - 37 Packages
  - Appendix A External Code Interface
  - Appendix B Test and Demo Functions
  - Appendix C Obsolete Functions
  - Appendix D Known Causes of Trouble
  - Appendix E Installing Octave
  - Appendix F Grammar and Parser
  - Appendix G GNU GENERAL PUBLIC LICE...
  - Concept Index
  - Function Index
  - Operator Index
  - Graphics Properties Index

Next: Preface [Contents][Index]

This manual documents how to run, install and port GNU Octave, as well as its new features and incompatibilities, and how to report bugs. It corresponds to GNU Octave version 9.1.0.

Table of Contents

- Preface
  - Acknowledgements
  - Citing Octave in Publications
  - How You Can Contribute to Octave
  - Distribution
- 1 A Brief Introduction to Octave
  - 1.1 Running Octave
  - 1.2 Simple Examples
    - 1.2.1 Elementary Calculations
    - 1.2.2 Creating a Matrix
    - 1.2.3 Matrix Arithmetic
    - 1.2.4 Solving Systems of Linear Equations
    - 1.2.5 Integrating Differential Equations
    - 1.2.6 Producing Graphical Output
    - 1.2.7 Help and Documentation
    - 1.2.8 Editing What You Have Typed
  - 1.3 Conventions
    - 1.3.1 Fonts
    - 1.3.2 Evaluation Notation
    - 1.3.3 Printing Notation
    - 1.3.4 Error Messages
    - 1.3.5 Format of Descriptions
      - 1.3.5.1 A Sample Function Description
      - 1.3.5.2 A Sample Command Description
- 2 Getting Started
  - 2.1 Invoking Octave from the Command Line
    - 2.1.1 Command Line Options
    - 2.1.2 Startup Files
  - 2.2 Quitting Octave
  - 2.3 Commands for Getting Help
  - 2.4 Command Line Editing
    - 2.4.1 Cursor Motion
    - 2.4.2 Killing and Yanking
    - 2.4.3 Commands for Changing Text
    - 2.4.4 Letting Readline Type for You
    - 2.4.5 Commands for Manipulating the History
    - 2.4.6 Customizing `readline`
    - 2.4.7 Customizing the Prompt
    - 2.4.8 Diary and Echo Commands
  - 2.5 How Octave Reports Errors
  - 2.6 Executable Octave Programs
    - 2.6.1 Passing Arguments to Executable Scripts
    - 2.6.2 Dual-Purpose Executable Scripts and Octave Functions

Fig 1.1: Octave's Built-In Documentation.

Once I was caught up on the essentials, including variable declaration, the "ans" command, and simple plotting of data, I moved on to attempting to import my data points from the Java portion of our Plot, Salt, and Smoother. Figure 1.2 below shows the sample output for 100 data points as input to a parabolic function. See Figure 1.3 for an example of data read from the output of my Java Plotter.

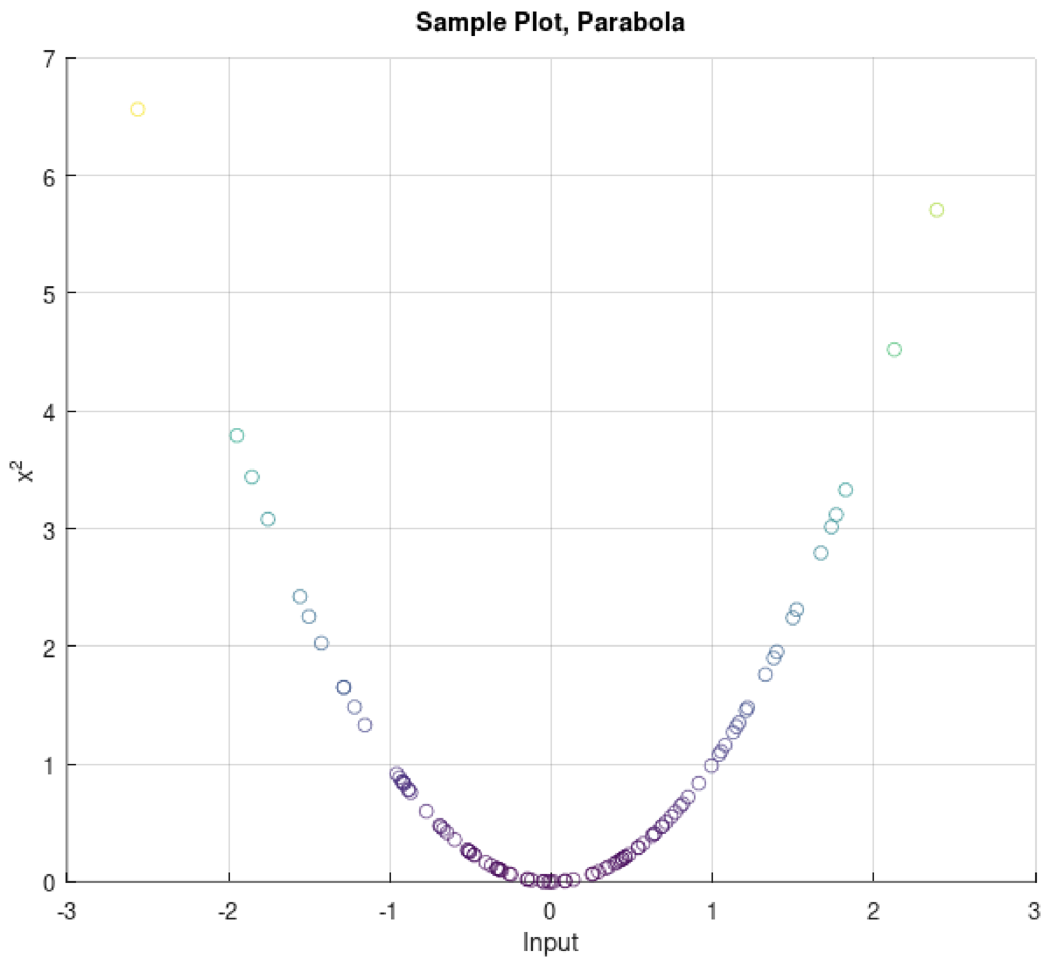


Fig 1.2: Sample Plot using Octave's Scatter Function.

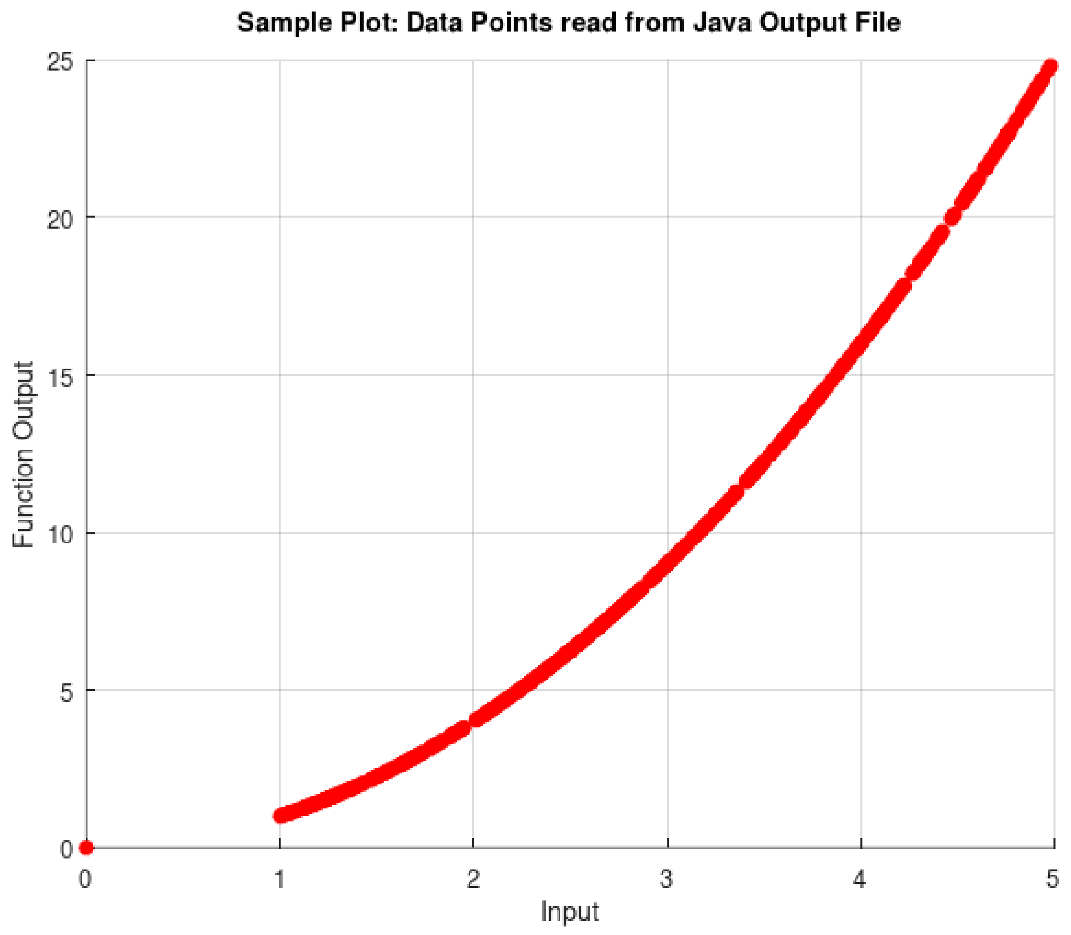


Fig. 1.3: Plotted data from Plotter csv output using Octave Functions.

After learning the basics of reading in files, separating data points, and plotting from a list of data, I next worked to implement a simple salting algorithm. See Figure 1.4 for an example of the salted Plot data points.

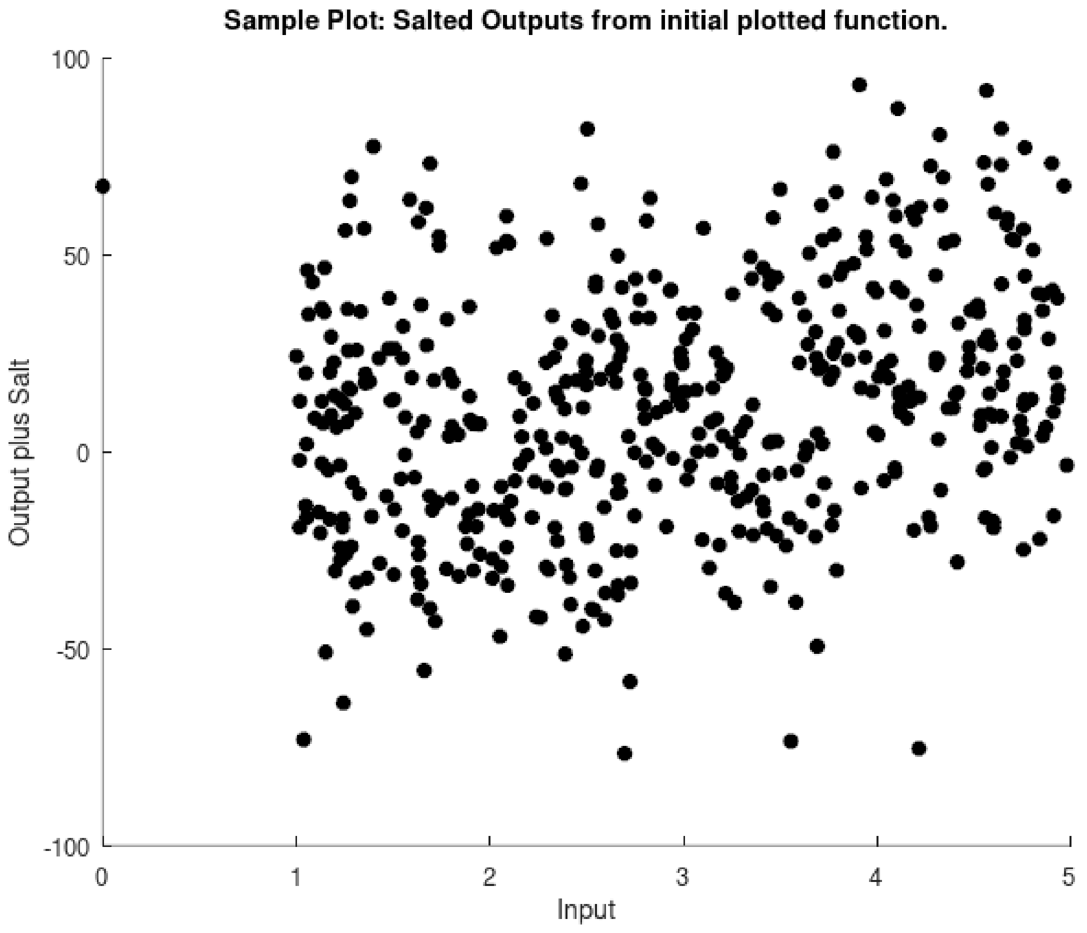


Fig. 1.4: Data with the Salt applied.

This salt was achieved by using a randomly generated value and adding it to each output point of the original list. Then, the same function was called to generate the scatter plot, adding new titles, axis labels, and colors as needed to differentiate the data. Finally, I attempted to implement a smoothing algorithm to the data to reflect the original trend of the graphed plot. See Figure 1.5 for an example of the output.

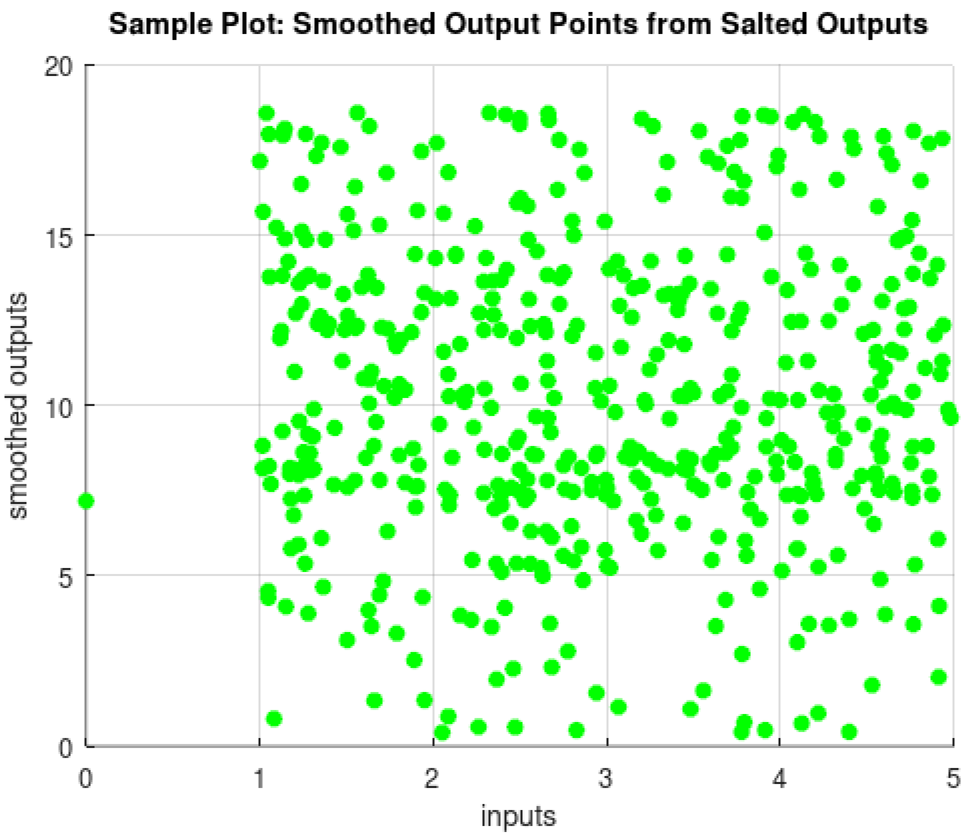


Figure 1.5: An example plot of the salted data with a moving average applied to each point.

After replicating the functions in Octave, it became clear that the errors I had experienced in my previous implementations had everything to do with the averages applied to each point. Instead of repeatedly taking an average of the salt, and then the new outputs, et cetera, I needed to additionally apply the original, clean output data. Also, reducing the window size allowed me to narrow the average taken for each point. I applied this to the smoothing algorithm provided by Octave and produced new results, shown below in Fig 1.6.

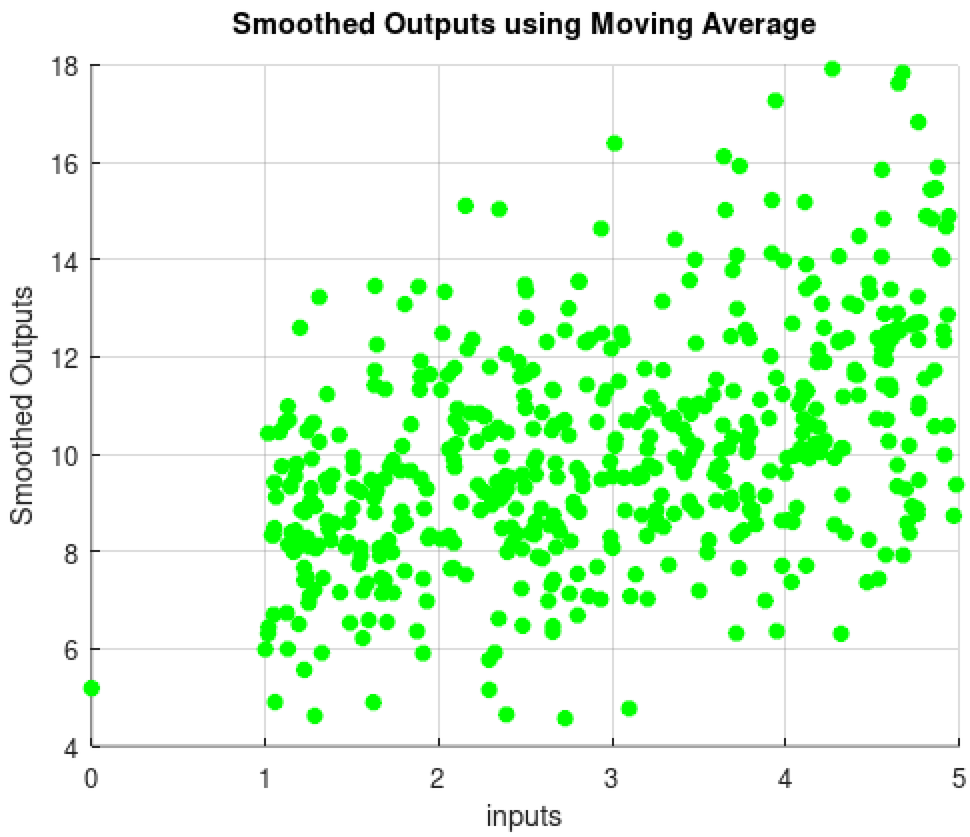


Figure 1.6: Updated Smoothing algorithm including weighted, original output.

As is observed between the original smooth and the updated algorithm, there is a somewhat noticeable trend in the graph, behaving like the original function. While additional smoothing and adjustment of the weighted original outputs parameter is necessary, this output is closer to what I have been attempting to display concerning a smoothed output plot.

By learning basic use, function implementation, and plot display in GNU Octave, I have added a useful and powerful skill to my arsenal. Concerning applied statistics, this program is a wonderful addition to what we have already learned and programmed ourselves. Not only did this exercise allow me to learn another application's functions, but also helped push my troubleshooting and implementation of these algorithms in my own projects that much further.