Matlab Octave Documentation

May 6, 2025

Introduction

This document provides detailed documentation for a MATLAB script designed to process and visualize polynomial data based on the function $y = x^3$. The script generates original data, adds random noise (salting), applies a moving average for smoothing, exports the results to CSV files, and creates visualizations (scatter plots with lines and a bar chart). The script is structured in four steps: data generation, salting, smoothing, and plotting. This documentation covers the script's purpose, functionality, parameters, outputs, and example usage.

1 Script Overview

The MATLAB script performs the following tasks:

- Generates discrete points for $y = x^3$ over a specified range and exports them to a CSV file.
- Adds random noise to the y-values (salting) and exports the salted data to a CSV file.
- Smooths the salted data using a moving average and exports the smoothed data to a CSV file.
- Creates two plots: a scatter plot with lines comparing original, salted, and smoothed data, and a bar chart of the original data.

Purpose

- Demonstrate data manipulation (generation, noise addition, smoothing) and visualization for a polynomial function.
- Export data to CSV files for further analysis or use in other programs.
- Visualize the effects of noise and smoothing on polynomial data.

Dependencies

- **MATLAB**: Core MATLAB environment for numerical computations and plotting.
- No external toolboxes required.

2 Functionality

The script is divided into four main steps, executed sequentially.

Step 1: Generate Original Data

- **Description**: Generates discrete points for the polynomial $y = x^3$ over the range [`startingX`, `endX`] with step size `step`.
- **Parameters**:

  - `startingX`: Starting x-value (default: -5).
  - `endX`: Ending x-value (default: 5).
  - `step`: Step size between x-values (default: 1).
- **Process**:
  - Creates an array `x` using `startingX:step:endX`.
  - Computes `y = x.`$^3(element-wisecubicfunction).Storesdataasamatrix$`original_data = [x; y]'`$(row$ $[x, y]pairs).$
  - Exports to `PolynomialGraph.csv` with headers `x,y`.
- **Output**: CSV file `PolynomialGraph.csv`.

Step 2: Salt the Data

- **Description**: Adds random noise to the y-values of the original data.
- **Parameters**:
  - `salt_min`: Minimum noise value (default: -100).
  - `salt_max`: Maximum noise value (default: 100).
- **Process**:
  - Generates random noise using `rand` scaled to [`salt_min`, `salt_max`].
  - Computes `y_salted = y + noise`, rounding to integers.
  - Stores data as `salted_data = [x; y_salted]'`).
  - Exports to `SaltedData.csv` with headers `x,y`.
- **Output**: CSV file `SaltedData.csv`.

Step 3: Smooth the Salted Data

- **Description**: Applies a moving average to the salted y-values to smooth the data.
- **Parameters**:
  - `window_size`: Number of points in the moving average window (default: 5, i.e., 2 points before, current, 2 points after).
- **Process**:
  - Iterates over each y-value in `y_salted`.
  - For each point, defines a window from `max(1, i - half_window)` to `min(length(y_salted), i + half_window)`.
  - Computes the mean of the window and rounds to an integer.
  - Stores smoothed data as `smoothed_data = [x; y_smoothed]'`).
  - Exports to `SmoothedData.csv` with headers `x,y`.
- **Output**: CSV file `SmoothedData.csv`.

Step 4: Plotting

- **Description**: Creates two figures to visualize the data.
- **Plot 1: Scatter Plot with Lines**
  - **Content**:

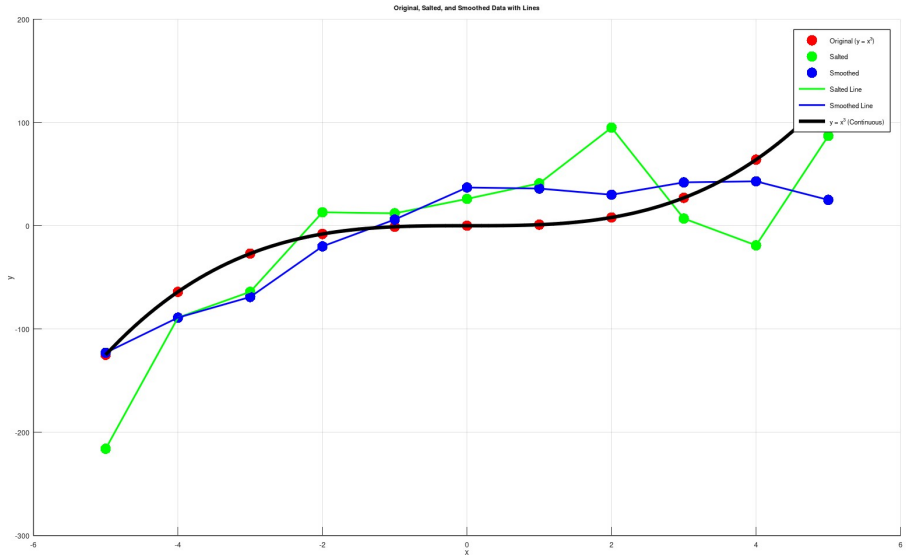

Figure 1: Ocatve Output of Plotted/Salt/Smooth Graph

  * Scatter points for original data (y, red filled circles).
  * Scatter points for salted data (`y_salted`, green filled circles).
  * Scatter points for smoothed data (`y_smoothed`, blue filled circles).
  * Lines connecting salted points (green) and smoothed points (blue).
  * Continuous $y = x^3$ curve (black, finer resolution with step 0.1).
- **Features**:
  * Title: "Original, Salted, and Smoothed Data with Lines".
  * X-label: "x", Y-label: "y".
  * Legend and grid enabled.

3 Parameters

- `startingX = -5`: Starting x-value for the data range.
- `endX = 5`: Ending x-value for the data range.
- `step = 1`: Step size for discrete x-values.
- `salt_min = -100`: Minimum random noise value.
- `salt_max = 100`: Maximum random noise value.
- `window_size = 5`: Size of the moving average window for smoothing.

4 Outputs

- **CSV Files**:
  * `PolynomialGraph.csv`: Original data (x, $y = x^3$).
  * `SaltedData.csv`: Salted data (x, $y = x^3 +$ noise).

    * `SmoothedData.csv`: Smoothed data (x, smoothed y).
  - **Figures**:
    * Scatter plot with lines comparing original, salted, and smoothed data.
    * Bar chart of the original $y = x^3$ data.
  - **Console Output**: Confirmation messages for CSV exports and data processing steps.

5 Example Usage

To run the script, save it as `polynomial_processing.m` and execute in MATLAB:

```
run polynomial_processing.m
```

Alternatively, copy the script into the MATLAB Command Window and press Enter.

Expected Console Output

```
Original data exported to PolynomialGraph.csv
Salted data exported to SaltedData.csv
Smoothed data exported to SmoothedData.csv
```

CSV File Example (`PolynomialGraph.csv`)

```
x,y
-5,-125
-4,-64
-3,-27
...
4,64
5,125
```

Visual Output

- **Figure 1**: A scatter plot with red (original), green (salted), and blue (smoothed) points, connected by green and blue lines, and a black continuous $y = x^3$ curve.
- **Figure 2**: A bar chart with blue bars representing $y = x^3$ for each x-value.

6 Usage Notes

- Ensure write permissions for the output CSV files (`PolynomialGraph.csv`, `SaltedData.csv`, `SmoothedData.csv`).
- The `step` parameter should be positive and appropriate for the range to avoid excessive or insufficient points.
- The `salt_min` and `salt_max` values control noise magnitude; adjust based on the scale of $y = x^3$ (e.g., ± 100 may be large for small x).
- The `window_size` for smoothing should be odd to ensure symmetry (e.g., 5 for 2 points on each side).
- Rounding to integers may introduce minor precision loss; remove `round` for floating-point output if needed.
- The continuous curve in the scatter plot uses a finer step (0.1); adjust `x_fine` range or step for different resolutions.

– The script clears the workspace and closes all figures at the start; save any open work before running.

7 Dependencies

* **MATLAB**: Core environment for numerical computations, file I/O, and plotting.
* No external toolboxes or libraries required.