PlotSaltSmoothJfree Package Documentation

May 6, 2025

Introduction

This document provides comprehensive documentation for the `PlotSaltSmoothJfree` package, which includes four Java classes: `PlotterJfree`, `PlotterJfreeMain`, `SalterJfe`, and `SmootherJfe`. The package is designed to generate and visualize polynomial graphs based on the function $y = x^3$, with options to add noise (salted) and create smoother curves using smaller step sizes. The `PlotterJfree` class handles coordinate generation and chart creation using JFreeChart, `PlotterJfreeMain` serves as the entry point, `SalterJfe` applies random noise, and `SmootherJfe` defines the step size for smoother graphs. This documentation covers the purpose, methods, fields, dependencies, and example usage of each class.

1 PlotterJfree Class

The `PlotterJfree` class is responsible for generating coordinates for polynomial graphs ($y = x^3$) and creating visualizations using JFreeChart. It supports three types of graphs: main (standard), salted (with noise), and smoothed (with smaller step sizes).

Class Overview

- **Package**: `PlotSaltSmoothJfree`
- **Purpose**: Generate coordinates and plot polynomial graphs with variants (main, salted, smoothed).
- **Dependencies**: `org.jfree.chart.*`, `javax.swing.*`, `java.util.*`
- **Constructor**: `PlotterJfree(double x, double y)`

Fields

- `private double x`: The x-coordinate of a point.
- `private double y`: The y-coordinate of a point.

Methods

- **`public PlotterJfree(double x, double y)`**
  - **Description**: Initializes a point with given x and y coordinates.
  - **Parameters**:
    * x: X-coordinate.
    * y: Y-coordinate.
- **`public double getX()`**
  - **Description**: Returns the x-coordinate.
  - **Returns**: X-coordinate as a double.

- **public double getY()**
  - **Description**: Returns the y-coordinate.
  - **Returns**: Y-coordinate as a double.
- **public static List¡PlotterJfree¿ generateMainCoordinates(double startingX, double endX, double step)**
  - **Description**: Generates coordinates for the main polynomial $y = x^3$ over the range [startingX, endX] with the specified step size. Prints coordinates to the console.
  - **Parameters**:
    - startingX: Starting x-value.
    - endX: Ending x-value.
    - step: Step size between x-values.
  - **Returns**: List of PlotterJfree objects representing points.
- **public static List¡PlotterJfree¿ generateSaltedCoordinates(double startingX, double endX, double step, SalterJfe salter)**
  - **Description**: Generates coordinates for a salted polynomial ($y = x^3 +$ noise) using the provided SalterJfe to add random noise. Prints coordinates to the console.
  - **Parameters**:
    - startingX: Starting x-value.
    - endX: Ending x-value.
    - step: Step size between x-values.
    - salter: SalterJfe object to apply noise.
  - **Returns**: List of PlotterJfree objects representing points.
- **public static List¡PlotterJfree¿ generateSmoothedCoordinates(double startingX, double endX, double step)**
  - **Description**: Generates coordinates for a smoothed polynomial ($y = x^3$) with the specified step size (typically smaller than the main graph's step). Prints coordinates to the console.
  - **Parameters**:
    - startingX: Starting x-value.
    - endX: Ending x-value.
    - step: Step size between x-values.
  - **Returns**: List of PlotterJfree objects representing points.
- **public static JFreeChart createChart(double startX, double endX, double mainStep, double smoothStep, SalterJfe salter)**
  - **Description**: Creates a line chart with three series: main polynomial, salted polynomial, and smoothed polynomial.
  - **Parameters**:
    - startX: Starting x-value.
    - endX: Ending x-value.
    - mainStep: Step size for main and salted graphs.
    - smoothStep: Step size for smoothed graph.
    - salter: SalterJfe object for adding noise.

  - **Returns**: `JFreeChart` object representing the line chart.

- **`public static void createAndShowGUI()`**
  - **Description**: Creates and displays a GUI window with a line chart of the polynomial variants.
  - **Functionality**:
    * Initializes a `JFrame` (800x600 pixels).
    * Sets default parameters: `startX = -10.0`, `endX = 10.0`, `mainStep = 1.0`.
    * Uses `SmootherJfe` with step size 0.1 and `SalterJfe` with noise amplitude 100.0.
    * Creates a chart using `createChart` and displays it in a `ChartPanel`.

Example Usage

```
PlotterJfree.createAndShowGUI();
```

This displays a JFrame window with a line chart showing three series: the main polynomial ($y = x^3$), a salted version with noise, and a smoothed version with smaller step size.

2 PlotterJfreeMain Class

The `PlotterJfreeMain` class serves as the entry point for the application, invoking the GUI creation method of `PlotterJfree`.

Class Overview

- **Package**: `PlotSaltSmoothJfree`
- **Purpose**: Run the polynomial plotting application.
- **Dependencies**: `PlotterJfree, SalterJfe, SmootherJfe`
- **Main Method**: `public static void main(String[] args)`

Methods

- **`public static void main(String[] args)`**
  - **Description**: Calls `PlotterJfree.createAndShowGUI()` to display the polynomial chart.

Example Usage

```
java PlotterJfreeMain
```

This runs the application, displaying a chart with the main, salted, and smoothed polynomial graphs.

3 SalterJfe Class

The `SalterJfe` class applies random noise to base values, used to create the salted polynomial graph.

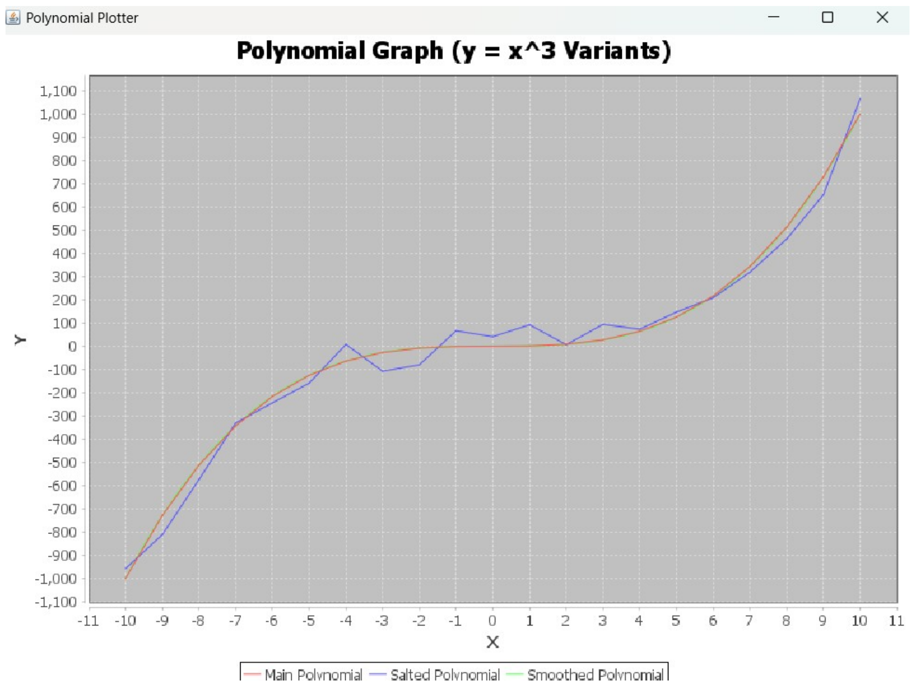

Figure 1: Overlay Graph using Jfree GUI output

Class Overview

- **Package**: `PlotSaltSmoothJfree`
- **Purpose**: Add random noise to values for the salted polynomial.
- **Dependencies**: `java.util.Random`
- **Constructor**: `SalterJfe(double noiseAmplitude)`

Fields

- `private double noiseAmplitude`: Maximum amplitude of the random noise.
- `private Random random`: Random number generator for noise.

Methods

- **`public SalterJfe(double noiseAmplitude)`**
  - **Description**: Initializes the salter with the specified noise amplitude.
  - **Parameters**:
    - `noiseAmplitude`: Maximum noise amplitude (non-negative).
  - **Throws**: `IllegalArgumentException` if `noiseAmplitude` is negative.
- **`public double applyNoise(double baseValue)`**
  - **Description**: Adds random noise in the range [-`noiseAmplitude`, +`noiseAmplitude`] to the base value.
  - **Parameters**:
    - `baseValue`: The value to which noise is added.
  - **Returns**: Base value plus random noise.
- **`public double getNoiseAmplitude()`**

  - **Description**: Returns the noise amplitude.
  - **Returns**: Noise amplitude as a double.
- **`public void setNoiseAmplitude(double noiseAmplitude)`**
  - **Description**: Sets the noise amplitude.
  - **Parameters**:
    * `noiseAmplitude`: New noise amplitude (non-negative).
  - **Throws**: `IllegalArgumentException` if `noiseAmplitude` is negative.

Example Usage

```
SalterJfe salter = new SalterJfe(100.0);
double baseValue = 1000.0;
double saltedValue = salter.applyNoise(baseValue); // Adds noise in [-100, 100]
System.out.println(”Salted value: ” + saltedValue);
```

4 SmootherJfe Class

The `SmootherJfe` class defines the step size for generating smoother polynomial graphs.

Class Overview

- **Package**: `PlotSaltSmoothJfree`
- **Purpose**: Specify the step size for smoothed polynomial graphs.
- **Constructor**: `SmootherJfe(double stepSize)`

Fields

- `private double stepSize`: Step size for generating coordinates.

Methods

- **`public SmootherJfe(double stepSize)`**
  - **Description**: Initializes the smoother with the specified step size.
  - **Parameters**:
    * `stepSize`: Step size (positive).
  - **Throws**: `IllegalArgumentException` if `stepSize` is not positive.
- **`public double getStepSize()`**
  - **Description**: Returns the step size.
  - **Returns**: Step size as a double.
- **`public void setStepSize(double stepSize)`**
  - **Description**: Sets the step size.
  - **Parameters**:
    * `stepSize`: New step size (positive).
  - **Throws**: `IllegalArgumentException` if `stepSize` is not positive.

Example Usage

```
SmootherJfe smoother = new SmootherJfe(0.1);
System.out.println("Step size: " + smoother.getStepSize()); // 0.1
```

5 Dependencies

- **JFreeChart**: For creating line charts (`org.jfree.chart.*`, `org.jfree.data.xy.*`).
- **Swing**: For displaying charts in JFrame windows (`javax.swing.*`).
- **Java Standard Library**: For data structures and random number generation (`java.util.*`).

6 Usage Notes

- Ensure a graphical environment is available to display the JFrame window.
- The `step` parameter in coordinate generation methods should be positive and appropriate for the range to avoid excessive points.
- The `SalterJfe` noise amplitude affects the visibility of the salted polynomial; adjust based on the scale of $y = x^3$.
- The `SmootherJfe` step size should be smaller than the main step size for a visibly smoother curve.
- Console output from coordinate generation methods can be removed for production use to improve performance.
- The chart range (`startX` to `endX`) should be chosen to cover the desired portion of the polynomial.

7 Example Output

Running `PlotterJfreeMain` produces console output like:

```
Main - x: -10.00, y: -1000.00
Main - x: -9.00, y: -729.00
...
Salted - x: -10.00, y: -987.45
Salted - x: -9.00, y: -740.12
...
Smoothed - x: -10.00, y: -1000.00
Smoothed - x: -9.90, y: -970.30
...
```

A JFrame window displays a line chart with three series: ”Main Polynomial,” ”Salted Polynomial,” and ”Smoothed Polynomial.”