JAVA SWING-BASED PLOTTING PACKAGE RESIDING WITHIN XAL*

A. Shishlo#, T. Pelaia, ORNL, Oak Ridge, TN 37831, U.S.A.
P. Chu, SLAC, Menlo Park, CA 94025, U.S.A.

Abstract

A data plotting package residing in the XAL tools set is presented. This package is based on Java SWING, and therefore it has the same portability as Java itself. The data types for charts, bar-charts, and color-surface plots are described. The algorithms, performance, interactive capabilities, limitations, and the best usage practices of this plotting package are discussed.

INTRODUCTION

Development of the plotting package started at the early stages of XAL [1] as a research project to study how fast data plotting can be updated. Later, interactive features of the package were found useful in several applications. This package is not intended to completely replace existing powerful plotting packages like JClass, but it is sufficient for instances where simple charts and color-surface plotting is required.

STRUCTURE OF THE PACKAGE

In terms of the Model-view-controller (MVC) pattern, the XAL internal plotting package is simplified, and it has only two components. The view and controller are combined together, and they are implemented in the FunctionGraphsJPanel class, which does not have any subclasses. There are four major types of data (Model components). Two are related to 2D chart plotting. The third can be used for bar-charts, and the last one for color-surface 3D plotting. The bar-charts plotting package is a separate package inside the general plotting package, and it provides necessary wrapping around base plotting classes.

DATA TYPES FOR CHARTS AND 3D

The two basic data types for 2D charts plotting are BasicGraphData (and its subclass CubicSplineGraphData) and CurveData. There are 4 subclasses for 3D color-surface data. The individual Data classes are discussed below.

BasicGraphData class

The BasicGraphData class represents data as a table for a function with possible errors for function values. It is a container for a set of triplets (x, y, error of y) ordered by x values. There should not be duplicated x points in any instance of this class. In this class linear interpolation between different x values is used, and a spline from the CubicSplineGraphData subclass is employed. The data points can be added to the container in different ways:

void addPoint(double x, double y)
void addPoint(double x, double y, double y_err)
void addPoint(double[ ] x, double[ ] y)
void addPoint(double[ ] x, double[ ] y, double[ ] y_err).

There are also a handful of methods such as "updatePoint" and "updateValues" which handle replacement of one or all data points.

The removeAllPoints() method will remove all data points, and the removePoint(int index) method removes only one data point with a particular index.

When a user adds or removes data from the container, by default it will notify all graph containers (the FunctionGraphsJPanel class instances) where this data has been added. It will also initiate a graphics repainting on all panels. This could slow down the painting if there are many data sets frequently calling for this repainting. To avoid this situation a user can switch off this trigger by calling setImmediateContainerUpdate(boolean param) with a "false" argument. In this case the repainting should be called directly for the FunctionGraphsJPanel class instance at an appropriate moment.

A user can set up different properties of the data that will affect its appearance on the graph panel. These properties are color, point size, shape, filling pattern switch, connecting line thickness, and name of the data that will appear on a legend. It is possible to switch off and on the drawings for the connecting lines or data points. The data container also has a dictionary inside, and a user can store arbitrary information related to this data set.

The example of a graphical representation of this data container is shown on Fig. 1.

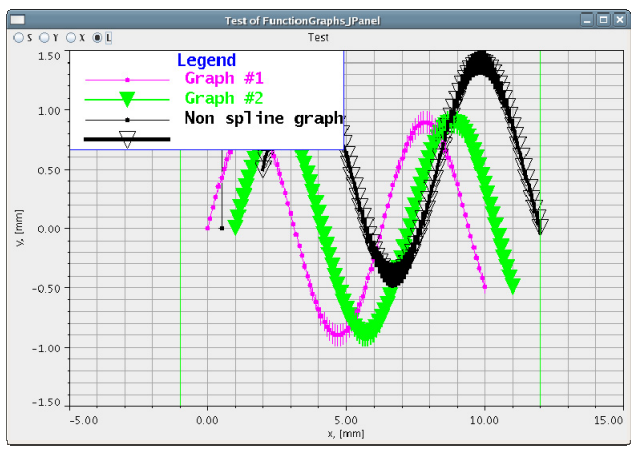

Figure 1: Example of a BasicGraphData representation on the FunctionGraphsJPanel class instance.

There is one additional subclass of the BasicGraphData named the UnwrappedGeneratorGraphData class. The addPoint method of this class is overridden to produce

* ORNL is managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DEAC05-00OR22725
#shishlo@ornl.gov

smooth data in cases of phase scans when original data can jump by ± π values.

The BasicGraphData class should be used when the amount of the data does not exceed about a thousand points. In this case the whole package demonstrates a good performance with about 20-40 Hz update frequency. If the number of points count is in the thousands, and the number of data sets is hundreds, it is better to use the lightweight CurveData class.

CurveData class

The CurveData class represents a set of (x, y) pairs. On the graphics panel the adjacent points will be connected by straight lines. Each time when the user modifies data, the findMinMax() method should be called to calculate extremes of the data. If a user wants a closed contour the last pair should be the same as the first pair. A user can specify a color, thickness, and a stroke width of the line. These data are not presented on the legend panel, and therefore they do not have names. The simplicity of this data container results in the high performance of plotting.

3D Color Surface Data

There are four subclasses of the abstract ColorSurfaceData class representing different algorithms of interpolation of the value field defined on a two dimensional rectangular grid. The ColorSurfaceData class has two abstract methods:

getValue(double x, double y)

addValue(double x, double y, double value).

The first method returns an extrapolated value at the specific position on the plane. The position is defined by x and y coordinates. The second method bins the value according the same extrapolation scheme. The implementations of this scheme are:

"smooth" uses 9-points smooth interpolation. It is used by default.

"linear" is a linear interpolation inside 4-points.

"point like" means no interpolation, it uses the nearest point in the grid.

"black&white" means no interpolation, it uses the nearest point in the grid, and the value can be 0 or 1 only.

All 3D color surface data types are produced by the Data3DFactory class by using the static method getData3D, which returns the ColorSurfaceData instance.

Only one instance of the ColorSurfaceData class can be registered inside the FunctionGraphsJPanel class instance, but it can be combined with CurveData and BasicGraphData instances (actually, combination with BasicGraphData does not make any sense).

The color scheme according to which the 3D data are presented is defined by the ColorGenerator interface method "Color getColor(double value)". This method maps the color and a value between 0 and 1. By default all 3D data classes have the RainbowColorGenerator implementation of the ColorGenerator interface. Users can define their own implementation and set it to the data class instance.

The presentation of the color surface data are defined by the size of the grid on the graphics panel. These sizes could be bigger than the sizes of the data grid. There is no allocation memory involved for this graphics grid, but the plotting time will increase proportional to the total size of this grid. The setScreenResolution(int nX, int nY) method is used to define the size of the graphics grid.

At this moment there is no legend available for the color scheme, but the screen reader can be used to see numerical values at arbitrary points on the graphics surface.

An example of the color surface data plotting with the 100x100 graphics area resolution is shown on Fig. 2.

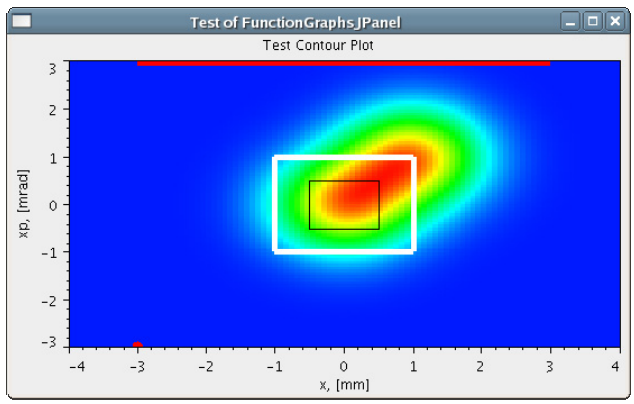

Figure 2: Example of a ColorSurfaceData representation on the FunctionGraphsJPanel class instance. The RainbowColorGenerator as a color scheme is used. The 3D color surface plot is combined with CurveData instances.

VIEW AND CONTROLLER

The FunctionGraphsJPanel class is a subclass of JPanel and implements two interfaces to provide interactive capabilities: MouseListener and MouseMotionListener.. All plotting takes place on the Graphics object of the FunctionGraphsJPanel class instance.

The three text fields clearly visible on the graphics panel (see Fig. 1, 2) are a graph title and the horizontal and vertical axes titles. A user can specify texts, fonts, and colors of the titles, and backgrounds of the graph and surrounding (border) region. The grid lines (Fig. 1) have separate colors and visibility switches (setGridLineColor, setGridLinesVisibleX, setGridLinesVisibleY).

The performance of the drawing on the graphics panel is not usually an issue by itself, but a user can switch "on" and "off" drawing on an "off-screen image" by using the setOffScreenImageDrawing method. The drawing on the off-screen image is slightly slower, but it helps to get rid of some artifacts related to the vertical layout of text regions.

Data Registration

To plot on the graphics panel of the FunctionGraphsJPanel class instance the data should be registered with this instance by using one the following methods:

- addGraphData(BasicGraphData bgd)
- addCurveData(CurveData cd)
- setColorSurfaceData(ColorSurfaceData csd).

During the registration the data will be immediately plotted. There are also methods to replace or add a collection of a data set packed in a Vector. As it was pointed out before, there can be only one ColorSurfaceData instance registered.

Zoom

The plotting region allows unlimited levels of zooming. The zoom is performed by a "press-drag-release" sequence using the left mouse button. A double click will return the state to the previous level of zooming.

Data Reader

The method getClickedPointObject() will return the ClickedPoint auxiliary class instance which has three public fields: xValueText, yValueText, and zValueText. They are JTextField instances and can be placed anywhere. After the left mouse button click, these fields will show the x, y, and z value for the mouse cursor position.

Axes Markers and Scales Control

The package includes a special class to provide interactive control of the chart axes. This control was created to provide a uniform GUI for different plotting packages that exist now and can be added in the future. To get this control the FunctionGraphsJPanel instance should be registered for this by calling,

SimpleChartPopMenu.addPopupMenuTo(chart).

This class resides in the gov.sns.tools.apputils XAL package.

Interactive Modes

There are four interactive modes for FunctionGraphsJPanel. The switchers (JRadioButton) can be placed at the left upper corner of the panel (see Fig. 1):

- setLegendButtonVisible(boolean)
- setChooseModeButtonVisible(boolean)
- setHorLinesButtonVisible(boolean)
- setVerLinesButtonVisible(boolean).

The first button makes the legend visible. The position of the legend is arbitrary, and it can be dragged to any place. The legend can be used to select the graph data. Remember that only BasicGraphData instances are shown on the legend.

The second button switches the graph panel in the graph data selection (choosing) mode. In this mode all graphs (BasicGraphData instances) have the black color except the one that has been selected by clicking on it (simultaneously the graph data point will be selected). The FunctionGraphsJPanel has a method to get the reference to this data (getGraphChosenIndex, and getPointChosenIndex).

The last two buttons make the tips of the vertical and horizontal lines visible, and a user can drag these lines with the mouse. These lines can be used to show the specific values (the limits of analysis, extremum position etc.).

GRAPH DATA OPERATIONS

To perform some operations with data, the plotting package includes a collection of static methods in the GraphDataOperations class. There are several groups of the graph operations.

The first is related to finding the intersections of the plots. It includes collections of findIntersectionX, findIntersectionY, findIntersection methods with different input parameters to find intersections for two or more plots in regions limited in both or only in one direction etc.

The getDataInsideRectangle method is used when a user limits the scope of the graphs operations by certain limits in x and y directions.

The polynomialFit methods collection is used to create polynomial fits for data.

The unwrapData method performs the same operations that UnwrappedGeneratorGraphData class does. It produces smooth data in cases of phase scans when original data can jump by $\pm\pi$ values.

BAR-CHARTS

To handle the bar-charts there is a "barchart" sub-package. It includes BarChart and BarColumnColor class, and BarColumn interface. The BarChart class is a wrapper around the FunctionGraphsJPanel class that creates set of CurveData instances to show the data defined by a Vector of BarColumn instances. The user has to implement the BarColumn interface to use this bar-chart plotting. An example of the bar-chart is shown on Fig. 3.

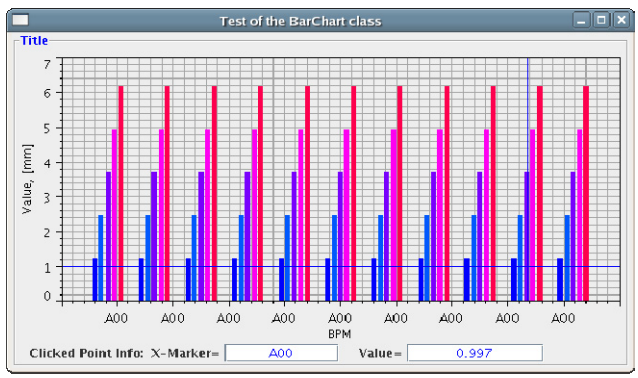

Figure 3: Example of a bar-chart plotting.

CONCLUSIONS

Despite the simplicity of the plotting package residing inside XAL, it is capable of satisfying basic demands from XAL applications and provides a satisfactory level of interactivity.

REFERENCES

[1] XAL, http://neutrons.ornl.gov/APGroup/appProg/xal/xal.htm