JEFFERSON LAB PLOTTING TOOLKIT FOR ACCELERATOR CONTROLS*

J. Chen, M. Keesee, C. Larrieu, G. Lei[#] [+]
Thomas Jefferson National Accelerator Facility, Newport News, VA

Abstract

Experimental physics generates numerous data sets that scientists analyze using plots, graphs, etc. The Jefferson Lab Plotting Toolkit, JPT, a graphical user interface toolkit, was developed at Jefferson Lab to do data plotting. JPT provides data structures for sets of data, analyzes the range of the data, calculate s the reasonabl e maximum, minimum and scale of axes, sets line styles and marker styles, plots curves and fills areas.

1 INTRODUCTION

The Continuous Electron Beam Accelerator Facility at Jefferson Lab provides high current electron beams of up to 4 GeV energy to three experimental halls. The machine consists of two superconducting linear accelerators connected together with 9 arcs. An injector system provides polarized and unpolarized electrons from two different sources. The Jefferson Lab Control System, which is built on the Experimental Physics and Industrial Control System (EPICS), is a distributed system using a client-server architecture. The system consists of two computing levels. The first level is composed of Unix workstations and X-terminals that execute a wide variety of system applications and high level applications. The second level is composed of single board computers, EPICS software and the correspondin g device control applications. [1]

One of the high level applications, called lute, analyzes and displays data from the Jefferson Lab wire scanners. The lute data analysis program relied on a commercial software package for its graphics. This package contains many more features than that are required by the few applications at Jefferson Lab that use it. So, when the controls group decided not to support the license for the commercial software for HP-UX 10.XX and higher systems, another solution was needed. The plotting toolkit, JPT, was developed to accomplish the analysis, calculation, scaling, and display of data for lute. Emphasis was placed on developing JPT to replace th e functionality of the commercial package while keeping the source codes that use it unchanged.

*Work supported by the U.S. Department of Energy, contract DE-AC05-84ER40150

[#] Email: leige@jlab.org

[+]On leave from the Institute of High Energy of Physics, Chinese Academy of Sciences

2 DESCRIPTION OF LUTE

Lute is a high level application that is used for data reduction and analysis for wire scanner data. Due to the geometry of the wires in relation to the beam, the wire scanners at Jefferson Lab provide scanned profiles representing the horizontal, x/y coupling, and vertical beam sizes. See figure 1. The wire scanner data is used for beam emittance measurement, beam matching, beam halo determination, beam energy spread measurement, and absolute beam energy measurement.

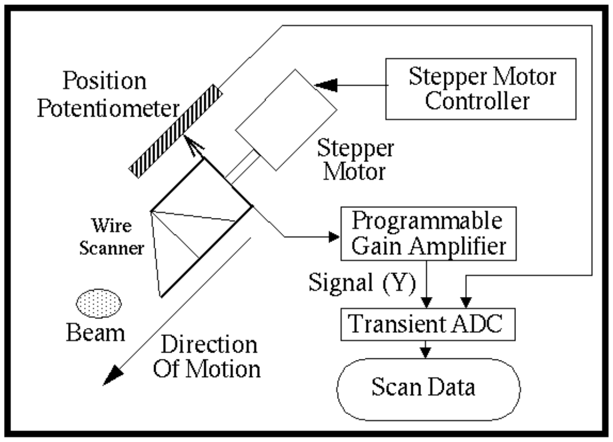
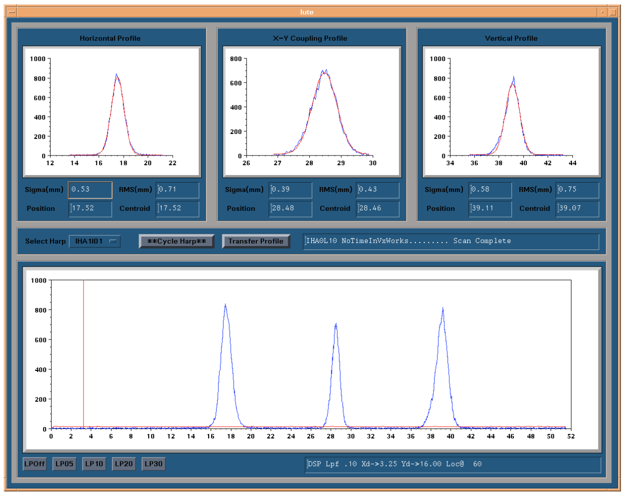


Figure 1. Jefferson Lab wire scanner

The lute application displays the wire scanner data graphically. See figure 2. The entire scan, beam signal



Sigma(mm) 0.53 RMS(mm) 0.71
Position 17.52 Centroid 17.52

Sigma(mm) 0.39 RMS(mm) 0.43
Position 28.48 Centroid 28.46

Sigma(mm) 0.58 RMS(mm) 0.75
Position 39.11 Centroid 39.07

```
IHA0L10 NoTimeInVxWorks......... Scan Complete
```

```
DSP Lpf .10 Xd->3.25 Yd->16.00 Loc@  60
```

Figure 2. Display of Lute

versus position in mm, is displayed in a large graph across the bottom of the interface, and the 3 distinct peaks (one from each wire) are displayed in 3 graphs at the top of the interface. Several calculations are also performed on the data for each peak and the results displayed on the lute interface. These include sigma, position, RMS and centroid calculations. The graphical displays and calculations assist the operators and accelerator physicists in determining beam characteristics.

3 JEFFERSON LAB PLOTTING TOOLKIT

The Jefferson Lab Plotting Toolkit (JPT) provides a graph widget that displays data graphically in a window and can interact with users. The graph widget can be used just like other X Toolkit and Motif widgets such as buttons, labels and menus. It has resources that determine how the graph will look and behave. Writing programs using JPT is similar to writing any other kind of X Toolkit and Motif program [2][3].

JPT has resources that allow control of:

- Graph data to be plotted, on one or more plots.
- Graph type (plot and area). A Plot graph draws each set as connected points of data. An Area graph draws each set as connected points of data, filled in below the points.
- Data styles: line colors and patterns; fill colors and patterns; line thickness; point style, size and color. How a data value looks when it is displayed depends on the data style that has been defined for the data values.
- Strings, position, color and font for the title.
- Strings, position, color and font for the legend.
- Graph color, font and border style.
- Axis label font; label string; color for axis and its label, tick marks and tick labels.
- Axis and data minimum and maximum, numbering and ticking increments, grid increments, origins.
- Placement of axes, annotation, origins.
- Control of user-interaction with the widget using callback resources.
- Markers and Text Areas.

JPT also provides several procedures and methods which

- Allocate and load data objects containing the numbers to be displayed;
- Attach the data to the axes;
- Create, attach, detach, destroy text object;
- Map data from pixel values to floating point values in a graph;
- Convert a floating point value to an X Toolkit parameter *ArgVal*

3.1 Implementation

JPT was developed based on the AthenaTools Plotter Widget Set (AtPlotter for short) [4]. 114 new resources were added into the original AtPlotterWidget to develop the JptPlotterWidget. Axes and plots were created inside the methods of the JptPlotterWidget and connected to the data to be displayed. Procedures were added to create data objects for plotting, to read data from files into data objects, and to attach the data object to the plotter to create plots. Data types for some of the resources were changed also. Text objects with multiple strings, foreground color, background color, font, border style etc. were created. The algorithm for calculating the tick-mark interval was improved for more accurate output of the plotting. This prevents plots that are too small, and axis tick-mark labels that are too crowded.

3.2 Make Data

JPT accepts data in two ways:

- Array data, which has successive integers as the common horizontal values (X values), and sets of floating point numbers, corresponding to each X value, for the vertical values (Y values)
- General data, each set of data has pairs of X-Y values.

When JPT begins to draw a graph, it uses the data values located in the data structure pointed to by the XtJptData resource. Data to be displayed can originate from diverse sources: Unix files, databases, real-time data feeds, or unrelated processes running on other machines.

If the data resides in a Unix file (and it does not need to be changed or updated in real time), there are two options to consider. The first option is to massage the data so that the file conforms to syntax understood by JptDataCreateFromFile(). This procedure will allocate the data object, load it with data from a named file, and return a handle to the data. Another approach is to allocate a data object using JptDataCreate(), and populate it with data by reading the data file (perhaps using fgets() or fscanf(). When the data object is loaded, the graph can be created and the XtJptData resource set to the JptDataHandle that references it.

Data that changes in real-time, and graphs that need to be updated in real-time, require more careful coding. The Xt Intrinsics provides functions that can be used to trigger application callback routines to deal with real-time events.

3.3 Axis Controls

JPT detects the minimum and maximum value of the data object and determines the extent of the axes based on the minimum and maximum data values, the origin, and the numbering increment. By default, JPT displays all data in the data object. The minimum and maximum resources can be set by applications if a particular part of the data set needs to be displayed.

The increment between tick marks and sub-tick marks on the axis is calculated by the JPT inner algorithm. The increment can also be controlled by setting the resources in the application. There are also resources to specify whether to draw grid lines, which grid line style to use, etc.

The axes are optionally labelled and the labels can be horizontal or vertical.

3.4 Fonts and Colors

JPT has resources to set colors for the widget, graph, axis, title and legend. JPT supports the specification of colors for the window background and foreground, as well as for the lines, fill patterns and points that represent data in the graph itself. Colors are specified using a string containing the color name, which should appear in the X Window Server's color database. On Unix systems, this is normally the file /usr/lib/X11/rgb.txt [5].

JPT can choose default colors for the application, so simple applications need not concern themselves with color specification. The default foreground and background color for the widget is black and white. The graph, axes, title and legend take the widget's foreground and background color as default.

A font may be specified for the title, the legend areas and for the axes annotation. JPT can use any font supported by the X Server. JPT accepts the font id to set the font for axis labels, title strings and legend strings. If the font is invalid, the "fixed" font will be used instead.

3.5 Calculate Data Scope

JPT is capable of analyzing the scope of the data to be displayed, getting a reasonable representation of the maximum, minimum, tick and sub-tick intervals on the axes. Consider an example with a real maximum value of the data for the Y-axis of 10.03, and the minimum value of 68.96, with the length of the Y-axis of about 200 pixels. It is ugly and difficult to set the ticks in the toolkit to draw the Y-axis with 10.03 as the lowest point and 68.96 as the highest point. It is better to use 10.00 as the lowest point and 70.00 as the highest point. Therefore, the algorithm that is used calculates and rounds four times to get a reasonable presentation.

3.6 Text Method

A text area is an independent rectangular region drawn over the graphed data. A data structure is provided for the programmers to define the attributes of the text area including text strings, text position, string adjustment, border style, foreground color, background color and font.

A text area can be attached to the graph in one of four ways:

- To a pixel location on the window
- To graph data X- and Y-value
- To a data point (set, point index), or
- Above or below a data point (set, point index, Y-value)

An application can use a text object to highlight special points on the graph, or as a more general-purpose label. There are methods to create, attach, detach and destroy text area. Any number of text areas can be attached to a graph, dynamically created, updated and destroyed.

3.7 Wrapper

We created a wrapper file that defines macros to map existing widget, class, resources and functions to the corresponding parts of JPT. Therefore the current applications need not change anything. They just need to re-link to the JPT library to get new executable codes that have a very similar output to what they had before.

4 CONCLUSION

JPT provides a graph widget that can be used like other widgets in the X Toolkit and OSF/Motif. It has many of the features scientific and business users need in a graph that will be embedded in another program. It is easy to use for creating X-Y plots for scientific-style graphics with an unlimited number of plots on each graph. The axes can be logarithmic or linear. It is also easy to access application data.

Figure 2 shows the display of lute using JPT.

5 ACKNOWLEDGMENT

Many thanks should be given to Valerie Bookwalter who helped us maintain the computing environment.

6 REFERENCES

[1] Karen S. White, Hari Areti, Omar Garza, "Control System Reliability at Jefferson Lab", ICALEPCS'97 Proceedings

[2] Adrian Nye, Tim O'Reilly, "X Toolkit Intrinsics Programming Manual", O'Reilly & Associates, Inc.

[3] Paul M. Ferguson & David Brennan, "Motif Reference Manual", O'Reilly & Associates, Inc.

[4] http://lune.csc.liv.ac.uk/hppd/hpux/X11/Toolkits/plotter-6.0p17

[5] Adrian Nye, "Xlib Programming Manual for Version 11", O'Reilly & Associates, Inc.