

A

PtPlot: 2D Graphs within Java

PtPlot is an excellent plotting package that lets you plot directly from your Java programs. *PtPlot* is free, written in Java (and thus runs under Unix, Linux, Mac OS, and MS Windows), is easy to use, and is actually part of *Ptolemy*, an entire computing environment that is supported by the University of California. Figure A.1 is an example of a *PtPlot* graph. Because *PtPlot* is not built into Java, your Java program needs to import the *PtPlot* package and work with its classes. We suggest that you download the most recent version over the Web.

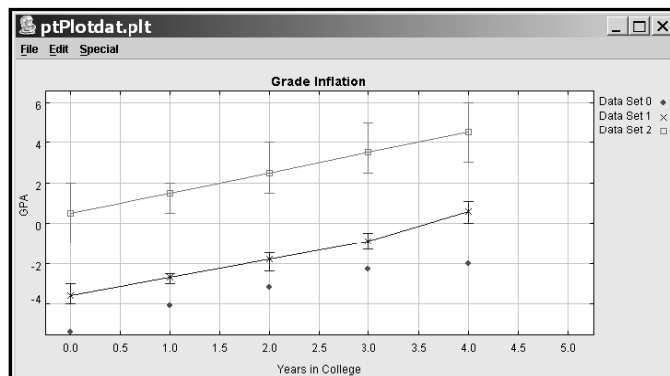


Fig. A.1 Sample output from *PtPlot* in which three data sets are placed on one plot. Observe the error bars on two of the sets.

The program `EasyPtPlot.java` in Listing A.1 is an example of a how to construct a simple graph of $\cos(x)$ versus x with *PtPlot*. On line 2 we see the statement `import ptolemy.plot.*;` that imports the *PtPlot* classes. (In order for this to work for you, you may have to modify your `CLASSPATH` environmental variable.) *PtPlot* represents your plot as a *Plot object*, which we name `myPlot` and create on line 7. We then add various features, step by step, to `myPlot` to make it just the plot we want. As is standard with objects in Java, we first give the name of the object and then modify it with “dot modifiers.” Rather than tell *PtPlot* what ranges for x and y to plot, we let *PtPlot* set the x and y ranges based on the data it is given. By having `true` as the fourth ar-

Computational Physics. Problem Solving with Computers (2nd edn).

Rubin H. Landau, Manuel José Páez, Cristian C. Bordeianu

Copyright © 2007 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim

ISBN: 978-3-527-40626-5

gument in `myPlot.addPoint(0, x, y, true)`, we are telling PtPlot to connect the plotted points. For the plot to appear on the screen, line 16 creates a `PlotApplication` with the plot as input.

Listing A.1: `EasyPtPlot.java` plots a function using the package PtPlot. Note that the `Application` object must be created to see the plot on your screen.

```
// EasyPtPlot.java: Simple PtPlot application

import ptolemy.plot.*;

public class EasyPtPlot {
    public static final double Xmin = -5., Xmax = 5.;    // Graph domain
    public static final int Npoint = 500;

    public static void main(String[] args) {

        Plot plotObj = new Plot();                        // Create Plot object
        plotObj.setTitle("f(x) vs x");
        plotObj.setXLabel("x");
        plotObj.setYLabel("f(x)");
        // plotObj.setSize(400, 300);
        // plotObj.setXRange(Xmin, Xmax);
        // plotObj.addPoint(int Set, double x, double y, boolean connect)
        double xStep = (Xmax - Xmin) / Npoint;

        // Plotting loop
        for ( double x = Xmin; x <= Xmax; x += xStep) {
            double y = Math.sin(x)*Math.sin(x);
            plotObj.addPoint(0, x, y, true);
        }
        PlotApplication app = new PlotApplication(plotObj);    // Display
    }
}
```

We encourage you to make your plot more informative by including further options in the commands, or by using the pull-down menus in the PtPlot window displaying your plot. The options are found in the description of the methods on the PtPlot Website [90], and include:

Calling PtPlot from Your Program

Plot myPlot = new Plot();	Name and create plot object <code>myPlot</code>
PlotApplication app = new PlotApplication(myPlot);	Display
myPlot.setTitle("f(x) vs x");	Add title to plot
myPlot.setXLabel("x");	Label x axis
myPlot.setYLabel("f(x)");	Label y axis
myPlot.addPoint(0, x, y, true);	Add (x, y) to set 0, connect points
myPlot.addPoint(1, x, y, false);	Add (x, y) to set 1, no connect points
myPlot.addLegend(0, "Set 0");	Label data set 0 in legend
myPlot.addPointWithErrorBars(0, x, y, yLo, yHi, true);	Plot, $(x, y - YLo), (x, y + yHi)$ + error bars
myPlot.clear(0);	Remove all points from data set 0
myPlot.clear(false);	Remove data from all sets
myPlot.clear(true);	Remove all points, default options
myPlot.setSize(500, 400);	Set plot size in pixels (optional)
myPlot.setXRange(-10., 10.);	Set an x range (default fit to data)
myPlot.setYRange(-8., 8.);	Set a y range (default fit to data)
myPlot.setXLog(true);	Use log scale for x axis
myPlot.setYLog(true);	Use log scale for y axis
myPlot.setGrid(false);	Turn off the grid
myPlot.setColor(false);	Color in black and white
myPlot.setButtons(true);	Display zoom-to-fit button on plot
myPlot.fillPlot();	Adjust x, y ranges to fit data
myPlot.setImpulses(true, 0);	Lines from points to x axis, set 0
myPlot.setMarksStyle("none," 0);	none, points, <i>etc.</i>
myPlot.setBars(true);	Display data as bar charts
String s = myPlot.getTitle();	Extract title (or other properties)

Once you have a PtPlot application on your screen, explore some of the ways to modify your plot from the application window:

1. Examine the Edit pull-down menu (underlined letters are shortcuts). Select Edit and pull it down.
2. From the Edit pull-down menu select Format. You should get a window (Fig. A.2) that lets you control various options in your graph.
3. Experiment with the Format menu; change the graph so only points are plotted and so that your name is in the title.
4. Select a central portion of your plot and **zoom in** on it by drawing a box (with mouse button depressed) starting from the upper left corner and then moving down before you release the mouse button. You **zoom out**

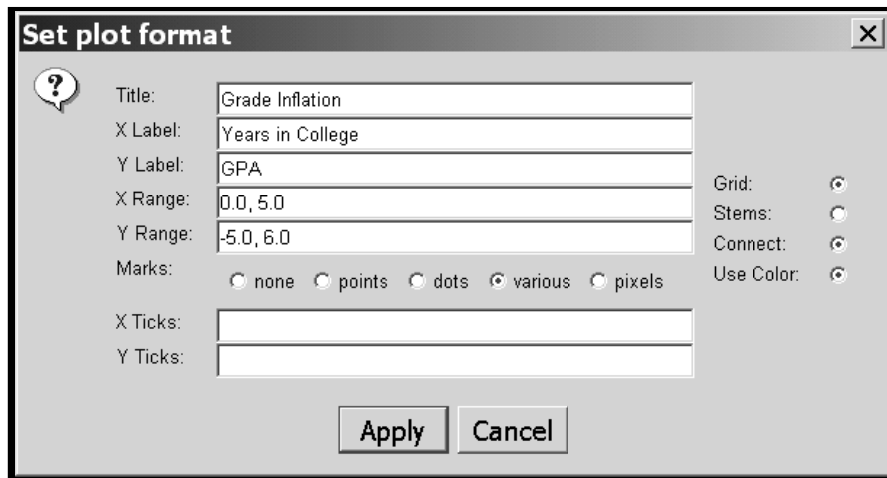


Fig. A.2 The Format submenu located under the Edit menu in a PtPlot application. This submenu controls the plot's basic features.

by drawing a box from the lower right corner and moving up. You may also resize your graph by selecting Special/Reset Axes or by resetting the x and y ranges. And of course, you always have the option of starting over by closing the Java window and running the `java` command again.

5. Scrutinize the File menu and its options for printing your graphs, as well as exporting them to files in postscript (.ps) and other formats.

It is also possible to have PtPlot make graphs by reading data from a file in which the x and y values are separated by spaces, tabs, or commas. There is even the option of including PtPlot formatting commands in the file with data. The program `TwoPlotExample.java` on the CD and its data file `data.plt` show how to place two plots side by side, and how to read in a data file containing error bars and various symbols for the points. In the simplest form, a *PtPlot Data Format* is just a text file with a single x, y point per line. To illustrate, Fig. A.1 was produced from the data file `PtPlotdat.plt`:

Sample PtPlot Data file `PtPlotdat.plt`

```
# This is a comment: Sample data for PtPlot TitleText: Grade
Inflation XRange: 0,5 YRange: -5, 6 Grid: on XLabel: Years in
College YLabel: GPA Marks: various NumSets:3 Color: on DataSet: Data
Set 0 Lines:off 0,-5.4 1,-4.1 2,-3.2 3,-2.3 4,-2 DataSet: Data Set
1 Lines:on 0,-3.6, -4,-3 1,-2.7, -3, -2.5 2,-1.8, -2.4,-1.5 3,-0.9,
-1.3, -0.5 4, 0.6, 0,1.1 DataSet: Data Set 2 0,0.5, -1,2 1, 1.5,
0.5, 2 2, 2.5, 1.5, 4 3, 3.5, 2.5, 5 4, 4.5, 3, 6
```

To plot up your data files directly from the command line, enter

```
> java ptolemy.plot.PlotApplication dataFile // Plot data in dataFile
```

This causes the standard PtPlot window to open and display your data. If this does not work, then your `CLASSPATH` variable may not be defined properly, or PtPlot may not be installed.

Reading in your data from the PtPlot window itself is an alternative. Either use an already open window, or issue Java's run command:

```
> java ptolemy.plot.PlotApplication // Open PtPlot window
```

To look at your data from the PtPlot window, choose File → Open → `FileName`. By default, PtPlot will look for files with suffixes `.plt` or `.xml`. However, you may enter any name you want, or pull down the Filter menu and select `*` to see all of your files. The same holds for the File → SaveAs option. In addition, you may Export your plot as an Encapsulated PostScript (`.eps`) file, a format useful for inserting in printed documents. (You may also use drawing programs to edit the output from PtPlot or to convert it into other formats.)

As with any good plot, you should label your axes, add a title, and add what is needed to be informative and clear. To do this, incorporate PtPlot commands with your data, or work in the PtPlot window with the pull-down menus under Edit and Special. The options are essentially the same as the ones you would call from your program:

TitleText: <code>f(x) vs. x</code>	Add title to plot
XLabel: <code>x</code>	Label x axis
YLabel: <code>y</code>	Label y axis
XRange: <code>0, 12</code>	Set x range (default: fit to data)
YRange: <code>-3, 62</code>	Set y range (default: fit to data)
Marks: <code>none</code>	(Default) No marks at points, lines connects points
Marks: <code>points</code>	or: dots, various, pixels
Lines: <code>on/off</code>	Do not connect points with lines; default: <code>on</code>
Impulses: <code>on/off</code>	Lines down from points to x axis; default: <code>off</code>
Bars: <code>on/off</code>	Bar graph (turn off lines) default: <code>off</code>
Bars: <code>width (, offset)</code>	Bar graph; bars of <code>width</code> and (optional) <code>offset</code>
DataSet: <code>string</code>	Specify data set to plot; <code>string</code> appears in legend
<code>x, y</code>	Specify a data point; comma, space, tab separators
move: <code>x, y</code>	Do not connect this point to previous
<code>x, y, yLo, yHi</code>	Plot $(x, y - yLo), (x, y + yHi)$ with error bars

If commands appear before `DataSet` directives, then the command will apply to all data sets. If commands appear after `DataSet` directives, then it will apply to that data set only.