Creating plots using *gnuplot*

Suparna Roychowdhury & Shibaji Banerjee

July, 2015

Introduction

Hello everyone! This is the beginning of a year long journey into different aspects and tools in Computational Physics. We would begin this journey by learning to visualize elementary functions, data and analyse them through a graphical interface called the "*gnuplot* ". This would help in developing an intuitive understanding of different aspects of elementary functions, data and the information they contain. In particular, even if rigorous model building might be one's ultimate goal, graphical methods still need to be the first step, so that a sense for the data, its behaviour, and quality is developed. As you would realize later, this would prove to an extremely important gadget which you can pick and safe-keep for your future endeavours in Physics or any form of scientific pursuit.

To get the session recorded, answer the underlined question(s) in the class itself and include the date. Include print-outs of the graph in the final submission (due next day) along with the code (do not typeset, write by hand). Fill in the index of the practical notebook with the underlined part of the title. You may leave the lab after showing the final output to one of the instructors. Use a 160 page LNB for maintaining your lab records.[1]

gnuplot is a program for exploring data graphically. Its purpose is to generate plots and graphs from data or functions. It can produce highly polished graphs, suitable for publication, and simple throw-away graphs, when were merely playing with an idea. *gnuplot* is command-linedriven: you issue commands at a prompt, and *gnuplot* will draw the current plot in response. *gnuplot* is also interactive: the output is generated and displayed immediately in an output window.

Invoking *gnuplot*

If *gnuplot* is installed on your system, it can usually be invoked by issuing the command:
`gnuplot`
at the shell prompt. Once launched, a gnuplot window opens up which displays a welcome message finally leading to a prompt showing

```
gnuplot>
```

Anything entered at this prompt will be interpreted as *gnuplot* commands until you issue an exit or quit command, or type an end-of-file (EOF) character, usually by hitting `Control-D`. Probably the simplest plotting command we can issue is

```
gnuplot> plot sin(x)
```

We begin by introducing you to a set of commands which can be issued at the `gnuplot` prompt to visualize elementary functions in different ways and to choose suitable domain and range according to user specific needs.

Learning to Plot Functions using gnuplot

FACTS

1.1 To plot a function $f(x)$

```
gnuplot> plot f(x)
```

1.2 To plot 2 functions $f(x)$ and $g(x)$

```
gnuplot> plot f(x),g(x)
```

1.3 As you might have noticed, the default x-range of *gnuplot* is (-10,10) and the y-range is selected according to the values of the function. However, these ranges can be altered and set according to your choice by issuing

```
gnuplot> plot [][y1:y2] f(x)
```

where the first pair of brackets define the x-range and the second pair gives the y-range. Here, we are perfectly happy with the x-range but want to change the y-range from default to (y1,y2).

1.4 For parametric plotting of functions, you can give two functions, one for the x and one for the y value, both depending on a common parameter, which by convention is called t. For certain kinds of curves, such a parameterization is simpler than the explicit form.

```
gnuplot>set parametric
gnuplot>plot t,t**2
gnuplot>unset parametric
```

This is same as saying `plot x**2`.

For parametric plots, there are now *three* relevant plot ranges: t-range, x-range, and y-range. The first controls the range of values assumed for the parameter (the dummy variable) t, while the other two are used (as before) to control the range of points that are visible in the plot. All three ranges can be adjusted either explicitly using `set trange [t1:t2]` or the familiar `set xrange [x1:x2]` and `set yrange [y1:y2]` commands, or inline as part of the `plot` command. If used inline, the setting for the trange precedes the other two.

1.5 To generate plots using other coordinate systems, like to enable polar mode:

```
gnuplot>set polar
gnulot>plot [-2*pi:2*pi] [-3:3] [-3:3] t*sin(t)
gnuplot>unset polar
```

Here, the independent variable is interpreted as an angle and the dependent variable as a radius. In polar mode, the independent (dummy) variable is denoted with t (not x), similar to the convention used for parametric plots. The angle may be given either in radians (multiples of π-this is the default) or in degrees. You can switch between both representations using

```
gnuplot>set angles [ degrees | radians ]
```

1.6 The `set samples` controls the number of points at which a function is evaluated to generate a plot. It defaults to 100 points, but this may not be sufficient for curves including cusps or singularities. A value of 300 - 500 works well.

```
gnuplot>show samples
gnuplot>set samples 500
```

ASSIGNMENTS

1.1 Plot the graph of $3x^2 - 8$ for x between -5 and 5.

1.2 Plot the same function, but restrict the y range between $-20 \ldots 40$.

1.3 Plot the graph of $x^3 + 1 - \exp(x)$ over $x = -8 \ldots 8$. Select suitable y range to show all the four x intercepts.

1.4 $y = \sin(x)$ over two complete periods.

1.5 Plot $3x^4 - 6x^2$ over the domain [-10,10] with automatic y scaling. After observing the graph, edit the domain and range so that you can see the x-intercepts clearly. Estimate the x-intercepts with the mouse cursor.

1.6 Define the functions $g(x) = 5\exp(-0.5x)$ and $h(x) = x + 1$ then do the following.

(a) Plot a graph that shows both functions g(x) and h(x). Experiment with different values for domain and range.

(b) Estimate the coordinates of the point of intersection of these two graphs by using your mouse. Check the answer by any calculator.

Include the code and the rough plots by hand.

1.7 Define the function $k(x) = x + 3\sin(2x)$, then do the following:

(a) Plot the graph of this function on the domain [-1,8] .

(b) Modify your plot from part (a) to include the horizontal line $y = 4$. Use this new plot to estimate the number and approximate values for x such that $k(x) = 4$.

(c) What single function could you graph that would give you the same information as in part (b)

1.8 Plot the parametric curve determined by $x = t^2 - t$ and $y = 2t - t^3$ over the t interval [-2,2].

1.9 Plot the polar equations $r = 1 + \cos\theta$ and $r = \sin 3\theta$ for $\theta = [-\pi, \pi]$

1.10 Plot $\exp(-x/100)\cos(x)$ with a suitable sampling rate.

Next, we move on to plotting data from datafiles.

Learning to Plot Data files in gnuplot

FACTS

gnuplot reads data from text files. The data is expected to be numerical and to be stored in the file in whitespace-separated columns. Lines beginning with a hashmark (#) are considered to be comment lines and are ignored. A typical datafile looks like

```
#x     y
1.2  2.3
2.4  4.5
3.2  5.6
4.5  7.6
```

The canonical way to think about this is that the x-value is in column 1 and the y-value is in column 2. If there are additional y-values corresponding to each x-value, they are listed in subsequent columns. If there in only one column, then the default x-values are taken to be integers, starting from 0 corresponding to each value of that column, which is taken to be y-values.

2.1 To plot a datafile, use its name

```
gnuplot>plot [0:4] [0:8] 'simple1.dat'
```

2.2 Since data files typically contain many different data sets, well usually want to select the columns to be used as x and y-values. This is done through the using directive to the `plot` command:

```
gnuplot>plot 'simple1.dat' using 1:2
```

2.3 If there are multiple data sets in one datafile one appended to one another, *gnuplot* can handle such data sets by plotting them as different graphs. For *gnuplot* , blank lines in a data file are significant. A double blank lines is used to distinguish data sets in a file. Each set can be addressed in the `plot` command as if it were in a separate file using the `index` directive to plot (*index* starts from 0).

```
gnuplot>plot 'simple.dat' index 0 using 1:2 w linespoints, \
>'simple.dat' index 1 using 1:2 w linespoints
```

2.3 Override default plotting style for data sets and adding title

```
gnuplot>plot [0:4] [0:8] 'simple1.dat' title 'some data' with lines
```

2.4 One way of joining the datapoints by a smooth curve is to use the smooth option. The smooth directive takes an additional parameter, which must be one of the following: `unique`, `frequency`, `bezier`, `sbezier`, `csplines`, `acsplines`, each one employing a different smoothing algorithm (you can find out more about these smoothing options from any book on numerical techniques).

```
gnuplot>plot [0:4] [0:8] 'simple1.dat' smooth csplines
```

2.5 gnuplot includes a facility to perform nonlinear least-square fits to data. The basic workflow for fitting data with a function consists of three steps: define a function, then use it as argument to fit, and finally plot the function together with the data set using plot.

```
gnuplot>f(x) = a*exp(-b*x)
gnuplot>a=200
gnuplot>b=1
gnuplot>fit f(x) 'sample.dat' u 1:2 via a,b  # 'using' provides the option
                                             # of fitting a data-set with
                                             # errorbars ie. 'u 1:2:3'
                                             # would mean that the 3rd column
                                             # contains y-errors.
gnuplot>plot 'sample.dat',f(x)
gnuplot>print a,b
```

2.6 If a data-file contains data-sets which are different sets, coming from different experiments or observations, they need to be separated by 2 blank lines sequentially. For example, take a look at the data file 'plotexp.dat'. The commands for gnuplot to read such data-files and plot the different data-sets with different markers are

```
#index 0 indicates the first data-set, index 1 indicates the 2nd data-set and index

gnuplot>plot 'plotexp.dat' index 0 u 1:2:3 ti "A. Smith (1992)" w yerr, 'plotexp.dat
```

ASSIGNMENTS

2.1 Plot a datafile containing the following data:

```
2.3
1.2
3.5
6.9
```

Note that the data are plotted against their index numbers in the file. Replot the data set so that one can see data point markers joined by segmented lines.

2.2 Replot the same dataset. Use segmented lines in one plot to join the datapoints in one trace. In another trace, join the datapoints by a smooth curve. Explore the other smoothing options (like !frequency! or !bezier!) too.

2.3 Plot the following dataset, choosing appropriate bounds.

```
1.0 200.6
2.0 100.3
3.0 43.2
4.0 25.6
5.0 8.9
```

2.4 Fit the above dataset, using a suitable function. Plot the dataset along with the function and examine visually, the nature of the fit. Note that the history of the fit process is kept in a !.log! file. Find out more about the fit process by reading the associated help and jot down the facts that you find useful.

2.5 Compare the representation of the dataset using point-markers, segmented-lines, smooth-curve and the fitted-curve all in a single plot.

2.6 You have been supplied with a data-file called *plotexp.dat*. Note that there are three data sets in the data file written sequentially. Each data block is separated by 2 blank lines. The experimental data points are stored in three columns - x, y and dy. The uncertainties are given as absolute errors in y-values. Fit the above data-set with a linear function. Plot the data-set with the fit overlayed on it.

In all the plots that youve seen so far, we plotted one variable (y) as a function of another one (x). But what if one wants to show how some quantity depends on two independent variables? In other words, how can we best visualize a single "output" variable as a function of two "input" variables? In this context, surface plots are discussed. The syntax of the `splot` (short for "surface plot") command is very similar to the syntax for the plot command.

Surface Plots using gnuplot

FACTS

3.1 To plot a surface, use

```
gnuplot>splot [x1:x2] [y1:y2] [z1:z2] f(x,y)
```

3.2 Parametric surfaces can be drawn with the following commands

```
gnuplot>set parametric
gnuplot>splot [u1:u2] [v1:v2] [x1:x2] [y1:y2] [z1:z2] f1(u,v), f2(u,v), f3(u,v)
```

3.3 3d-curves (space surves) can be plotted with

```
gnuplot>set parametric
gnuplot>splot f1(u), f2(u), f3(u)
```

3.4 To send back all options to their default values, use

```
gnuplot>reset
```

NOTES

ASSIGNMENTS

3.1 **Using the cartesian equation of a sphere, show a hemisphere of radius 2 on the x-y plane. Then, plot the whole spherical surface using piecewise forms of the z-surface. Now use any suitable parametrization of the surface to achieve the same effect. Note that the spherical surface appears squashed in the figure. What may be the cause of this distortion ? Note that you can rotate the figure using the mouse to observe it from any direction.**

3.2 **Draw any 3-D straight line using a suitable parametrization.**

1. **PLot the Paraboloid $z = x^2 + y^2$. Choose suitable ranges of x, y and z to display the following figure.**

3.3 **Plot several turns of a simple helix which advances in the z direction.**

3.4 `pm3d` **is a style for plotting surfaces in which the heights are represented as colored patches. Plot the sphere and the paraboloid using this style. Just remember that you will have to set pm3d before giving the splot command. After the plotting is over, unset the pm3d option.**

3.5 **A 'Contour diagram' represents a surface by joining together lines of equal elevation on the x-y plane. To produce a contour plot you would first need to turn the contour on (`set contour`), and then issue the usual *splot* command. After the plotting is over, unset the contour option. If the number of contour lines are too few and far between, it is possible to use a bigger number of these lines, by setting the contour level `set cntrparam levels 10` before switching on the contour option. Check that you get the correct 'isotherms' by projecting out the function $z = xy$ on the plane. Then, investigate the contours of the 2-D stationary waves by using the surface plotting option for the function $z = \cos x \cos y$**

To be able to create publication quality 2D plots

Getting Camera ready Plots using gnuplot

FACTS

What Counts:

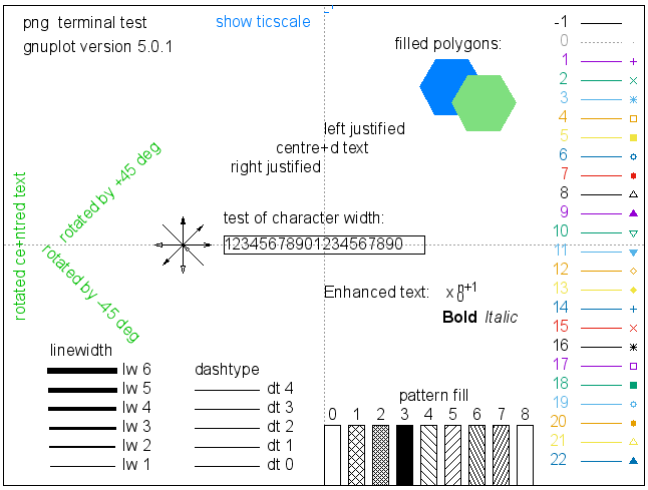
1. **Entire data-set is displayed clearly with appropriate markers, with error bars if relevant and line styles.**
2. **Appropriately titled**
3. **Proper axis labels**
4. **Appropriate legends (keys)**
5. **Camera Ready**

Styling: To plot exptl. data with error bars, one uses the command 'yerr' or 'xerr' or 'xyerr' depending on the data having only errors in y or errors in x or errors in both. The datafile has to be in the format of *column-1:* x*-data, column 2:* y*-data, column 3:* $xlow$*-data, column 4:* $xhigh$*-data, column 5:* $ylow$*-data and column-6:* $yhigh$*-data* **or *column-1:* x*-data, column 2:* y*-data, column 3:* dy*-data.*

```
gnuplot>plot 'datafile.dat' using 1:2:3 w yerrorbars        # ( x, y, dy )
gnuplot>plot 'datafile.dat' using 1:2:3:4:5:6 w xyerrorbars # ( x, y, xlow, xhigh,
                                                            #        ylow, yhigh )
```

gnuplot **has a built-in command called `test` that generates a standard 'test' image. The 'test' image shows all available line styles and fill patterns, and also attempts to demonstrate more advanced terminal capabilities, such as the ability to rotate text through an arbitrary angle. To use the 'test' command,**

```
gnuplot>test                    # shows a plot with all available
                                # line style and fill patterns
                                # and other abilities of this terminal
gnuplot>set term png
gnuplot>set output 'test.png'
gnuplot>test                    # generates a PNG file
                                # called test.png which
                                # shows all available
                                # line style and fill patterns
                                # and other abilities of this
                                # terminal.
```



Saving and Exporting: There are two ways to save your work in *gnuplot* : we can save the *gnuplot* commands used to generate a plot, so that we can regenerate the plot at a later time. Or you can export the actual graph to a file in one of a variety of supported graphics file formats, so that we can print it or include it in web pages, text documents, or presentations.

4.1 To save the commands that we used to generate a plot to file

```
gnuplot>save 'graph_plt.txt'
```

You can later load them again and in this way regenerate the plot where we left off. This will save the current values of all options, as well as the most recent plot command, to the specified file. This file can later be loaded again using the load command:
`load 'graph_plt.txt'` **The effect of loading a file is the same as issuing all the contained commands (including the actual plot command) at the gnuplot prompt.**

4.2 You can also write a set of commands in a plain text file (say for example a file named 'xyz_plt.txt') to run in *gnuplot* . Usually these files contain exactly one command per line. Several commands can be combined on a single line by separating them with a semicolon (;). The hashmark (#) is interpreted as a comment character: the rest of the line following a hashmark is ignored. The hashmark isnt interpreted as a comment character when it appears inside quoted strings. To continue a single command line sentence into more than one line, you can use the 'át the end of each line. To run this file in *gnuplot* , use the load command.

```
gnuplot>load xyz_plt.txt
```

4.3 For any graph you might want to generate and export (using gnuplot), you need to specify two things: the format of the graph (GIF, JPG, PNG, PDF and so on) and the output device (either a file or the screen). In *gnuplot* , this is done using the `set` command:

```
gnuplot>set term png
gnuplot>set output 'a.png'
```

Another terminal type `pdf` can be set by

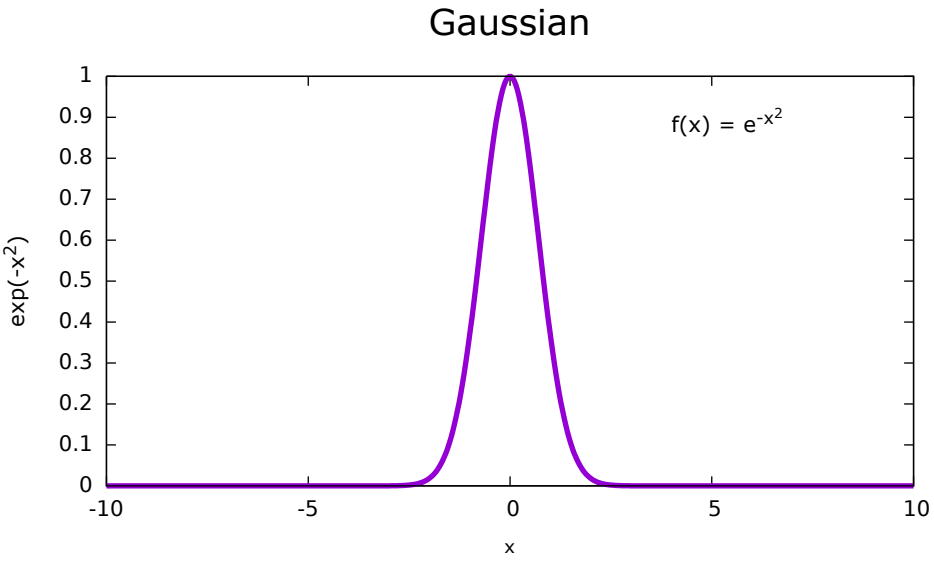
```
gnuplot>set term pdfcairo enhanced
gnuplot>set output 'a.pdf'
```

4.4 Issue any plotting commands and then, quit *gnuplot* and view the generated file. If you want to continue to use *gnuplot* interactively, reset the term.

```
gnuplot>reset
gnuplot>set term wxt enh
gnuplot>set out
```

Here is an example of a *command* file which can be loaded in *gnuplot* to create a publishable quality figure as shown. Here we have shown how to add title, x and y-axes labels, increase the font size and types of the fonts of the labels and also add a label/legend in the plot.

```
reset

#Generate pdf output

set term pdfcairo
set out 'gaussian.pdf'

#key is unset

unset key

#Generate dynamic text labels to be put up
#on a blank area of the plot.
#Note: Specification of unicode font type and size.

s=sprintf("f(x) = e^{-x^2}")
set label s at 4,0.9 font "Verdana,12"

#Set title and axis labels
#Note use of 2D text

set title 'Gaussian' font "Verdana,20" # title of the plot
set ylabel '\exp({-x^2})' font "Verdana,12"
set xlabel 'x' font "Verdana,11"
set samples 500
plot exp(-x**2) lw 4

#Get back to interactive mode

reset
set term wxt enh
set out
```



In this context, please note that in *version 4.6* we would have to do a`set term wxt enh` to see the 2D text in the labels. In version 5 (and possibly above) this is automatic. Note the syntax for adding 2D text.

Note that we form the string *s* before we put up the label using the *sprintf* function.

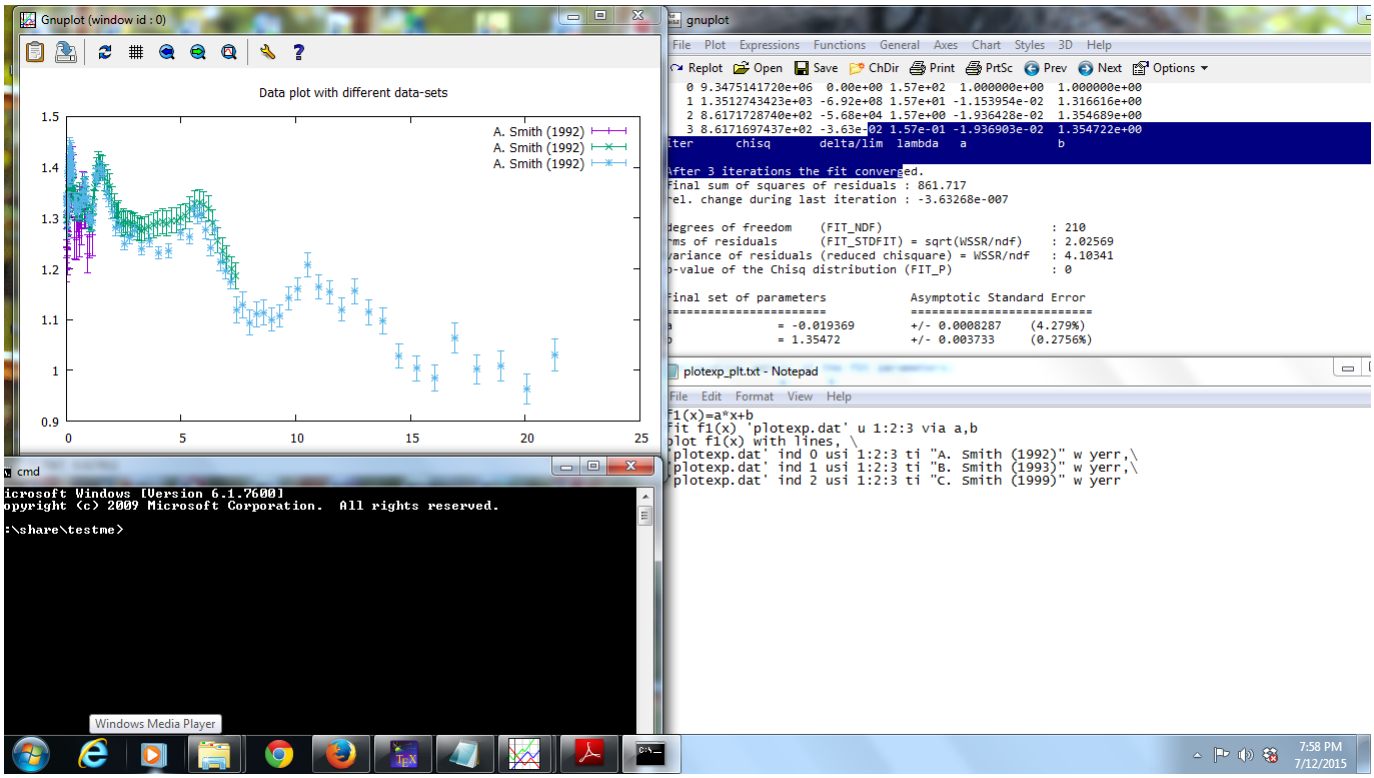
The advantage of forming a dynamic label in this way is that if any variable is being printed as a label, then its value will be updated in the label every time it is recomputed.

Note: *Greek letters* in the label can be included using the syntax `{/Symbol a}` or `{/Symbol b}`. You can find out which English character matches which Greek letter using the command `set title '{Symbol abcdefghijklmonpqrstuvwxyz}'`. In verison 5, this produces:

αβχδεφιφκλμνοπθρστυϖωξψζ

We have tried to give you an exposure to the essential aspects of this extremely user-friendly and versatile plotting software. To find out more about any aspect of this package, you could refer to the documentation or help which comes with *gnuplot* . Just type 'help any command' in *gnuplot* like `help plot` to get the documentation on that topic.

To end, here is a final look of how things might look like on your workspace once you get everything going *gnuplot*

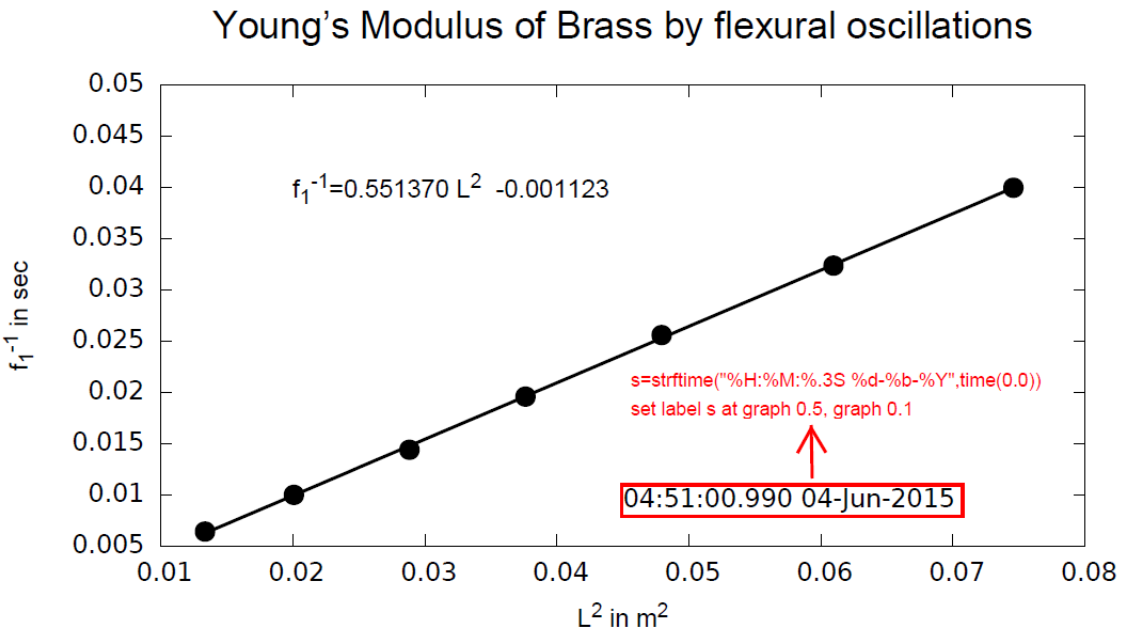

Gnuplot (window id : 0)
Data plot with different data-sets
A. Smith (1992)
A. Smith (1992)
A. Smith (1992)
gnuplot
File Plot Expressions Functions General Axes Chart Styles 3D Help
Replot Open Save ChDir Print PrtSc Prev Next Options
0 9.3475141720e+06 0.00e+00 1.57e+02 1.000000e+00 1.000000e+00
1 1.3512743423e+03 -6.92e+08 1.57e+01 -1.153954e-02 1.316616e+00
2 8.6171728740e+02 -5.68e+04 1.57e+00 -1.936428e-02 1.354689e+00
3 8.6171697437e+02 -3.63e-02 1.57e-01 -1.936903e-02 1.354722e+00
iter chisq delta/lim lambda a b
After 3 iterations the fit converged.
Final sum of squares of residuals : 861.717
rel. change during last iteration : -3.63268e-007
degrees of freedom (FIT_NDF) : 210
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 2.02569
variance of residuals (reduced chisquare) = WSSR/ndf : 4.10341
p-value of the Chisq distribution (FIT_P) : 0
Final set of parameters Asymptotic Standard Error
= -0.019369 +/- 0.0008287 (4.279%)
= 1.35472 +/- 0.003733 (0.2756%)
plotexp_plt.txt - Notepad
File Edit Format View Help
f1(x)=a*x+b
fit f1(x) 'plotexp.dat' u 1:2:3 via a,b
plot f1(x) with lines, \
'plotexp.dat' ind 0 usi 1:2:3 ti "A. Smith (1992)" w yerr,\
'plotexp.dat' ind 1 usi 1:2:3 ti "B. Smith (1993)" w yerr,\
'plotexp.dat' ind 2 usi 1:2:3 ti "C. Smith (1999)" w yerr
cmd
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\share\testme>
Windows Media Player
7:58 PM
7/12/2015

ASSIGNMENTS

In this class, we will work on a sample data-set to produce a publication ready pdf. The data shown here was taken during one of Prof. N. Shrinivasan's(IISC) refresher courses on experimental physics. The idea was to measure the Rigidity modulus of a material in the form of a wire by observing resonances in forced torsional oscillations.

1. **Create a graph from the data-set specified in the file `y_data.txt` by reading commands from a file created by you named `y_plot.txt`**
2. **Fit this to a linear equation $q(x) = m\,x + c$**
3. **Title it : "Young's Modulus of Brass by flexural oscillations"**
4. **Provide x label : L^2 in m^2**
5. **Provide y label: f_1^{-1} in sec**
6. **Choose the Arial font for the titles and labels. Choose suitable sizes for them.**
7. **Display the equation of the fitted line on top of the data. This label must be dynamically generated from the computed values of m, c.**
8. **Export the figure to the pdf file `y_fig.pdf`**
9. **Export the figure to the grey-scale pdf file `y_fig_mono.pdf`**
10. **Create a list of files (with full paths) which you have used and write a line explaining the role of each of them.**

Compare your output with the following figure: Your figure should contain the time label at a convenient point on the graph. The commands for including this label is shown in red ink on top of the figure.



References

[1] **Gnuplot in Action by Philipp K. Janert, Manning Publications Co., 2010.**

[2] **GNUPLOT, Toshihiko Kawano, 2005,** `http://t16web.lanl.gov/Kawano/gnuplot/index-e.html`