Plotting with Gnuplot

Gnuplot is a powerful tool to generate professional looking plots. It is particularly interesting for our scope since it allows to easily plot data sequences from files. Gnuplot is open source, and is available for many platforms (see http://www.gnuplot.info for more details).

In this section, you can find some quick examples that can be useful to produce plots to be included in the laboratory report. For detailed instructions, and advanced capabilities offered by gnuplot, you can refer to some tutorials that can be found on the internet, on gnuplot manual (available on the CD itself), and the inline help pages that can be invoked within gnuplot itself by the `help` command.

Gnuplot can be used both typing command in an interactive manner, or by writing the description of the plot in a file, then tell gnuplot to load it and produce the plot. Considering plotting data from files, any file in columnar format can be used to produce plot via the following command, in which points (X,Y) are plotted and connected by lines:

```
plot ”file.dat” using X:Y title ”title” with linespoint
```

`file.dat` is the name of the file to be read as input.

`X,Y` tells which column has to be used for values of the X and Y variable. For example, `using 2:3` plot the point found in the second column as X value, while values in the third column will be used as Y value.

`with linespoint`: specifies that the data have to be represented using points joined by lines. See help plot with for all possibilities.

Multiple curves can be plotted on the same plot by separating them with a comma

```
plot sin(x), ”file.dat” using X:Y title ”title” with linespoint
```

Some useful commands:

`set xlabel “text”` : set the text label on the X axis to “text”

`set ylabel “text”` : set the text label on the Y axis to “text”

To produce a png file, simply choose the png terminal, and direct the output to a file

```
set terminal png
set output “file.png”
```

Given this, it should be straight forward to understand what plot will be produced by the following script:

```
# Simple gnuplot file
# lines starting with # are ignored...

# set the label on the x and y axis
set xlabel "time [s]"
set ylabel "bytes"

# plot the data from the "data.txt" file
# using values appearing on the X column as x values
# and values appearing on the Y as column y values
# X and Y have to be choosen properly...

plot "data.txt" using X:Y title "title" with linespoint

# if you want to plot several datasets/lines, simply separate them using
a ',' (comma)

#plot "data.txt" using X:Y title "first line" with linespoint, "data2.txt"
using X2:Y2 title "second line" with linespoint

pause 10 # to see the plot on the screen for 10 seconds

# save the same on plot a png file to be imported later on

set term png
set output "plot.png"

replot # to simply reproduce the same plot as before…
```

How to plot a probability density function using gnuplot

[most of this comes from http://psy.swansea.ac.uk/staff/carter/gnuplot/gnuplot_frequency.htm]

The gnuplot commands to plot this are shown below. The principle of operation is to count the number of values that fall into specific ranges (or bins). The first thing we need to do is group the numbers into collections of similar value; this is done by the functions `rounded()` and `bin_number()`.

`rounded(x)` takes the `x` parameter and rounds it to its nearest multiple of `bin_width`. This way, the fixed values returned can be counted into bins. However, the actual value returned by `rounded()` is first shifted up or down by an amount equal to the `bin_width` multiplied by the offset. Typically, the offset will be 0.5 since this would position the bar at the centre of its range on the X axis.

The `bin_number()` function uses `bin_width` to compute how many multiples the `X` parameter is of the range (and, therefore, the particular bin into which the `X` parameter should be counted. `bin_number()` does this by converting values into grouped values (the bins). For instance, with `bin_width`=0.1, values between 0.0 and 0.09 will be rounded down to 0, 0.10 ... 0.19 will be rounded down to 0.1, 0.20 to 0.29 will be 0.20, and so on.

Once we have arranged the disparate values of the data file into a selection of predefined values, we can then count the instances of those values, which we then plot. Since we don't want to plot the actual `X` values, but the number of them in certain bins, instead of the typical using command, we use gnuplot's `smooth` command, with the `frequency` option:

Note that the plot command includes brackets around the using values. The brackets tell gnuplot to interpret these values as numerical expressions, rather than column numbers (although a dollar followed by a number explictly references a column when inside brackets). Thus, `(rounded($1))` means "pass the value in column one to the `rounded()` function, and use the value returned as the X-axis coordinate. And (1) means to increment the bin value (i.e. Y) by one for each corresponding X value.

```
# Each bar is half the (visual) width of its x-range.

bin_width = 0.1; set boxwidth bin_width/2 absolute

set style fill solid 1.0 noborder

bin_number(x) = floor(x/bin_width)

rounded(x) = bin_width * ( bin_number(x) + 0.5 )

plot "Data.dat" using (rounded($1)):(1) smooth frequency with boxes
```

The `smooth frequency` option makes the data monotonic in x; points with the same x-value are replaced by a single point having the **summed y-values**.

How to plot a Cumulative Density Function using gnuplot

If you are interested into the CDF of a dataset, then you can use the `smooth cumulative` capability of gnuplot. The `cumulative` option makes the data monotonic in x; points with the same x-value are replaced by a single point containing the **cumulative sum of y-values** of all data points with lower x-values (i.e. to the left of the current data point).

```
# Count the number of elements in the dataset.

# Use the stats helper for this

stats(Data.dat)

N= floor(STATS_records)

plot "Data.dat" using ($1):(1/N) smooth cumulative with boxes
```

For more information, you could refer to the smooth.dem tutorial available at
http://www.gnuplot.info/demo/smooth.html

How to make a heatmap with gnuplot

Assume you have a data file `data.dat` which is a matrix, in which each cell represents the value to plot, i.e., *z=f(x,y)*. The variable *z* depends on two other variables *x* and *y*. A convenient way to plot this it to use a heatmap, to show the effects rather than plotting multiple lines on a regular line-graph.

Assume the file `data.dat` has a a tuple (x,y,z) in each row. Then the `plot.gnu` file below would do the magic.

Data.dat

```
# x y z
  0 0 5
  0 1 4
  0 2 3
  0 3 1
  0 4 0

  1 0 2
  1 1 2
  1 2 0
  1 3 0
  1 4 1

  2 0 0
  2 1 0
  2 2 0
  2 3 1
  2 4 0

  3 0 0
  3 1 0
  3 2 0
  3 3 2
  3 4 3

  4 0 0
  4 1 1
  4 2 2
  4 3 4
```

Plot.gnu

```
#set the terminal as we like

set term png size 1024,768
set out "heatmap.png"

# set the viewing angle for a map
set view map

# eventually choose the interpolation
# increase the values for higher interpolation
# 1,1 for no interpolation, 0,0 for max interpolation
set pm3d interpolate 0,0

# do the heatmap
splot "data2.dat" using 1:2:3 with pm3d
```