

Using a Plot File with Gnuplot

This is a brief introduction by example to using plot files with the GNUPLOT plotting program. You'll be able to generate plots more efficiently and be able to save them for future enhancement (e.g., adding titles, more results, etc.). Here is the basic idea:

- Put all of your gnuplot commands in a file (which we'll call a "plot file");
- within gnuplot, use the "load" command (assume the plot file is called myplot.plt):

```
gnuplot> load "myplot.plt"
```

 and the commands will be executed by gnuplot;
- edit the plot file, save, and give another "load" command and the plot is updated!

You can start with just a simple plot command in the file and add bells and whistles (e.g., titles, axis labels, ranges) iteratively.

Suppose we have the data file test1.dat with this content:

```
# Coulomb potential with Ze^2=1, b=.6 (hbar=1, mass=1)
# N lowest eigenvalue
10 -0.47937
11 -0.48558
12 -0.48767
13 -0.49119
14 -0.49232
15 -0.49441
20 -0.497732
30 -0.499479
40 -0.499781
```

and we know the exact answer is $-1/2$. We want to plot the data on a log-log plot to look for power-law dependence (note: the data is *not* already in the form of logs) but we want to fit the logarithms (base 10) to a linear function. Here's how we can do all that using a plot file called test1.plt. (Remember that everything after a "#" is a comment.)

Note in the example how we can process the data in columns 1, 2, and so on by referring to them as \$1, \$2, and so on. For example, `(log10($1))` takes the logarithm of the numbers in column 1 [be sure to put ()'s around the expression].

```

# plot file test1.plt
set timestamp      # it's always a good idea to turn on the timestamp

# the title and axis labels were added after the the plot started
set title 'Coulomb potential with m=1, Ze^2=1, hbar=1'
set xlabel 'basis size D'
set ylabel 'relative error of lowest energy'
set xrange [10:40]
set xtics 10,2,40

exact = -0.5      # exact answer

f(x) = m*x+b      # for a linear fit
# $1 refers to column 1 in test1.dat, $2 to column 2, and so on
fit f(x) "test1.dat" using (log10($1)):(log10(abs(($2-exact)/exact))) via m,b
slope_title = sprintf("slope = %-.4.1f",m) # check "man sprintf" for details

set logscale
plot "test1.dat" using (($1)):(abs(($2-exact)/exact)) title 'b=0.6', \
(10**b)*x**a title slope_title

```

