# How to use the GNU plot

**Ajay Malkoti**

Senior Research Fellow

Academy of Scientific and Innovative Research

Shallow Seismic R & D group

CSIR-National geophysical Research Institute

AcSIR

AcSIR

Types of plotting

1. plotting a function in 1D (line plot)
2. Plotting data from file
3. plotting multiple 1D data (overlay and multiplot/subplot)
4. plotting a function in 2D, 3D as surface, contour etc plot

# The first script

- **Single line graph of asin(x)*cos(x)**

```
reset # a good practice to add it
set xlabel "X-AXIS"
set ylabel "Y-AXIS"
set xrange [0:5]
set yrange [-2:2]
a= 3.14
print a # disp value of a on screen
plot a*sin(x)*cos(x)
```

- **Overlay graph of sin(x)*cos(x) and exp(x)**

```
set xlabel "X-AXIS"
set ylabel "Y-AXIS"
set xrange [0:5]
set yrange [-2:2]
a= 20
print a
unset yrange          # exponential doent fit in earlier yrange
plot a*sin(x)*cos(x) title 'sincos',\
     exp(x) title 'exp'
```

AcSIR

# Setting up canvas

1. set title "This is my first plot" # Title
2. set xlabel "$x$ axis" # x axis label
3. set ylabel "$y$ axis" # y axis lable
4. set yrange [-50:2000]
5. set xrange [-10:100]
6. set xtics 40
7. set mxitcs 8 # minor xtics
8. set key top right # legend

AcSIR

# Some other basic commands

1. plot/splot: for simple/surface plot
2. exit or quit
3. help <topic>
4. load e.g. load 'work.gnu'
5. replot or refresh # repeats the last plot/splot command
6. unset xrange # unsetting xrange to its default
7. reset # unsetting all graph related setting

AcSIR

# Setting output

1. set terminal postscript landscape color enhanced
   set output "my-plot.ps"
2. set terminal png
   set output "my-plot.png"

AcSIR

running the command/file

1. **gnuplot> save "myplot.plt"**
   Used in GNUplot environment for *saving current setting* to file 'myplot.plt'
2. **gnuplot> load "savefile.plt"**
   Used in GNUplot environment for *loding the setting* to file 'myplot.plt'
3. **$ gnuplot savefile.plt**
   Used in shell envorenment for running the GNUplot script.

AcSIR

# GNU: 1D data I

**Type I- single line**

```
# X     Y
1.0   1.2
2.0   1.8
3.0   1.6
```

```
plot 'data.txt' using 1:2 with lines
```

**Type II- Two line(same sampling)**

```
# X    Y1    y2
1.0   1.2      1.5
2.0   1.8      2.3
3.0   1.6      2.0
```

```
plot 'data.txt' using 1:2 with lines linestyle 4,\
'data.txt' using 1:3 with lines linestyle 6
```

AcSIR

# GNU: 1D data II

**Type III- Two lines(diff sampling)**

```
# First data block (index 0)
#X      Y
1.0   1.2
2.0   1.8
3.0   1.6

# Second index block (index 1)
#X      Y
1.0   1.2
2.0   1.8
3.0   1.6
```

```
set style line 1  linecolor rgb '#0060ad' \
    linetype 1 linewidth 2  pointtype 7 pointsize 1.5
set style line 2  linecolor rgb '#dd181f' \
    linetype 1 linewidth 2  pointtype 5 pointsize 1.5
```

AcSIR

# GNU: 1D data III

```
plot 'plotting_data3.dat' index 0 with linespoints linestyle 1,\
       '' index 1 with linespoints linestyle 2

       OR

set style line 1  lc rgb '#0060ad' lt 1 lw 2  pt 7 ps 1.5
set style line 2  lc rgb '#dd181f' lt 1 lw 2  pt 5 ps 1.5
plot 'plotting_data3.dat' index 0 with linespoints ls 1,\
       '' index 1 with linespoints ls 2
```

---

**Type IV: plot with error**

```
 X     Y     Yerror\\
1.0   1.2   0.1\\
2.0   1.8   0.1\\
3.0   1.6   0.1\\
```

---

**Type V: 1 line, diff error for x,y**

AcSIR

# GNU: 1D data IV

```
 X     Y     EX    EY\\
1.0   1.2   0.8   1.5\\
2.0   1.8   0.3   2.3\\
3.0   1.6   1.0   2.1\\
```

---

**Type VI: 2 lines, diff sampling, diff error**

```
1.1   0.8   0.2\\
2.1   0.3   0.2\\
3.1   1.0   0.2\\

1.2   1.5   0.3\\
2.2   2.3   0.3\\
3.2   3.1   0.3\\
```

# My first plot

**Start GNU**

$ gnuplot

**Give command**

*gnuplot* > plot "test.dat" using 1:2 with lines,"test.dat" using 1:3 with lines,"test.dat" using 1:4 with lines

**In multiple lines**

*gnuplot* > plot "test.dat" using 1:2 with lines,\
"test.dat" using 1:3 with lines,\
"test.dat" using 1:4 with lines

# GNU plotting data from file: 1D data

`http://lowrank.net/gnuplot/datafile2-e.html`

**Data file, data.txt**

*gnuplot* >
plot "data.txt" using 1:2 with lines,
"data.txt" using 1:3 with lines,
"test.txt" using 1:4 with lines

Plotting with new function

| X | Y1 | Y2 | Y3 |
|---|---|---|---|
| -1.0000 | 0.0000 | 0.0000 | 1.0000 |
| -0.9000 | 0.5700 | 1.1769 | 0.7150 |
| -0.8000 | 1.0800 | 1.4400 | 0.4600 |
| -0.7000 | 1.5300 | 1.4997 | 0.2350 |
| -0.6000 | 1.9200 | 1.4400 | 0.0400 |
| -0.5000 | 2.2500 | 1.2990 | -0.1250 |
| -0.4000 | 2.5200 | 1.0998 | -0.2600 |
| -0.3000 | 2.7300 | 0.8585 | -0.3650 |
| -0.2000 | 2.8800 | 0.5879 | -0.4400 |
| -0.1000 | 2.9700 | 0.2985 | -0.4850 |
| 0.0000 | 3.0000 | -0.0000 | -0.5000 |
| 0.1000 | 2.9700 | -0.2985 | -0.4850 |
| 0.2000 | 2.8800 | -0.5879 | -0.4400 |
| 0.3000 | 2.7300 | -0.8585 | -0.3650 |
| 0.4000 | 2.5200 | -1.0998 | -0.2600 |
| 0.5000 | 2.2500 | -1.2990 | -0.1250 |
| 0.6000 | 1.9200 | -1.4400 | 0.0400 |
| 0.7000 | 1.5300 | -1.4997 | 0.2350 |
| 0.8000 | 1.0800 | -1.4400 | 0.4600 |
| 0.9000 | 0.5700 | -1.1769 | 0.7150 |
| 1.0000 | 0.0000 | -0.0000 | 1.0000 |

AcSIR

# Plotting 1D data with error

gnuplot> plot "test.dat" using 1:2:3 with yerrorbars
gnuplot> plot "test.dat" using 1:2:3:4 with yerrorbars

| **Data Format** | **Column** | **Using** | **With** |
|---|---|---|---|
| (X,Y) data | X Y | 1:2 | lines, points, steps, linespoints, boxes, etc. |
| Y ±dY | X Y dY | 1:2:3 | yerrorbars |
| X ± dX | X Y dX | 1:2:3 | xerrorbars |
| Y ± dY, and X ± dX | X Y dX dY | 1:2:3:4 | xyerrorbars |
| Y range [Y1,Y2] | X Y Y1 Y2 | 1:2:3:4 | yerrorbars |
| X range [X1,X2] | X Y X1 X2 | 1:2:3:4 | xerrorbars |
| Y range [Y1,Y2], and X range [X1,X2] | X Y X1 X2 Y1 Y2 | 1:2:3:4:5:6 | xyerrorbars |

AcSIR

Performing mathematical operation on a coloum and plotting

- plot 'exp.dat' using 1:exp($2) with lines or
  plot 'exp.dat' u 1:exp($2) w lines
- plot 'exp.dat' using 1:2:(sqrt($2)) with yerrorbars or
  plot 'exp.dat' u 1:2:(sqrt($2)) w yerr
- f(x) = A0*exp(-x/tau)
  A0=1000;tau=1
  plot 'exp.dat' u 1:2:(sqrt($2)) w yerr, f(x)

AcSIR

# Setting the line properties

1. **Viewing styles 4**
   gnuplot>show linetype 4
   linetype 4, linecolor rgb "#e69f00" linewidth 1.000 dashtype solid pointtype 4 pointsize default pointinterval 0
2. **Viewing all default styles(1-8)**
   gnuplot>show linetypes
3. **Defining new line style**(GNUplot ver. dependent)
   set linestyle 1 lt 1 lc 7 # black-solid
   set linestyle 2 lt 2 lc 1 # red-dashed

AcSIR

# Sample scripts: Multiplot (as subplot)

`http://gnuplot.sourceforge.net/demo/layout.html`

- set multiplot; # get into multiplot mode
  set size 1,0.5;
  set origin 0.0,0.5; plot sin(x);
  set origin 0.0,0.0; plot cos(x)
  unset multiplot

# Sample scripts: Multiplot (as subplot) I

http://gnuplot.sourceforge.net/demo/layout.html

```
set terminal postscript landscape color enhanced
set output "my-plot.ps"
# Set overall margins for the combined set of plots and size them
# to generate a requested inter-plot spacing
if (!exists("MP_LEFT"))   MP_LEFT = .1
if (!exists("MP_RIGHT"))  MP_RIGHT = .95
if (!exists("MP_BOTTOM")) MP_BOTTOM = .1
if (!exists("MP_TOP"))    MP_TOP = .9
if (!exists("MP_GAP"))    MP_GAP = 0.05

set multiplot layout 2,2 columnsfirst title "{/:Bold=15 Multiplot with exp
              margins screen MP_LEFT, MP_RIGHT, MP_BOTTOM, MP_TOP spacing

set format y "\%.1f"
set key box opaque
set ylabel 'y'
set xlabel 'x'
set xrange [-2*pi:2*pi]
```

# Sample scripts: Multiplot (as subplot) II

```
plot sin(x) lt 1
plot cos(x) lt 2

unset ylabel
unset ytics
unset xlabel
plot sin(2*x) lt 3

set xlabel 'x'
plot cos(2*x) lt 4
unset multiplot
```

# Commands I

(http://soc.if.usp.br/manual/gnuplot-doc/htmldocs/)

1. **cd** e.g. cd "c:\newdata". Note: In windows use double quots(".")
2. **pwd**
3. **save**: It saves user-defined functions, variables, the 'set term' status, all 'set' options, or all of these, plus the last 'plot' ('splot') command to the specified file.

```
save 'work.gnu'
save functions 'func.dat'
save var 'var.dat'
save set 'options.dat'
save term 'myterm.gnu'
save '-'
save '|grep title >t.gp'
```

# Commands II

4. **set** e.g. `set <option>`
   angles,arrow,autoscale,bars,bind _, bmargin,border,boxwidth,
   clabel,clip,cntrparam,color_box,colornames,contour,data_style,
   datafile,decimalsign,dgrid3d,dummy,encoding,fit _, fontpath,format _,
   function_style,functions_,grid,hidden3d,historysize,isosamples,key,
   label,linetype,lmargin,loadpath,locale,logscale,macros,mapping,margin,mouse,
   multiplot,mx2tics,mxtics,my2tics,mytics,mztics,object,offsets,origin,output,
   parametric_,plot_,pm3d,palette,pointintervalbox,pointsize,polar_,print_,psdir,
   raxis,rmargin,rrange,rtics,samples,size,style,surface,,table,terminal,
   termoption,tics,ticslevel,ticscale,timestamp,timefmt,title_,tmargin,trange,urange
   ,variables,version,view,vrange,x2data,x2dtics,x2label,x2mtics,x2range,x2tics
   ,x2zeroaxis,xdata,xdtics,xlabel,xmtics,xrange,xtics,xyplane,xzeroaxis,y2data
   ,y2dtics,y2label,y2mtics,y2range,y2tics,y2zeroaxis,ydata,ydtics,ylabel,ymtics
   ,yrange,ytics,yzeroaxis,zdata,zdtics,zzeroaxis,cbdata,cbdtics,zero,zeroaxis,
   ,zlabel,zmtics,zrange,ztics,cblabel,cbmtics,cbrange,cbtics

AcSIR

# Commands III

5. **if**

```
if (<condition>) { <command>; <command>
                   <commands>
            } else {
                   <commands>
            }
```

6. **do iterator**:

```
set multiplot layout 2,2
do for [name in "A B C D"] {
    filename = name . ".dat"
    set title sprintf("Condition \%s",name)
    plot filename title name }
 unset multiplot
```

7. **for**

# Commands IV

```
for [intvar = start:end{:increment}]
for [stringvar in "A B C D"]

plot for [filename in "A.dat B.dat C.dat"] filename using 1:2 with lines
plot for [basename in "A B C"] basename.".dat" using 1:2 with lines
set for [i = 1:10] style line i lc rgb "blue"
unset for [tag = 100:200] label tag

set for [i=1:9] for [j=1:9] label i*10+j sprintf("%d",i*10+j) at i,j
```

8. **evaluate**: This is especially useful for a repetition of similar commands.

```
set_label(x, y, text) \
= sprintf("set label '%s' at %f, %f point pt 5", text, x, y)
eval set_label(1., 1., 'one/one')
eval set_label(2., 1., 'two/one')
eval set_label(1., 2., 'one/two')
```

# Commands V

9. **fit**

```
fit {<ranges>} <expression> '<datafile>' {datafile-modifiers}
               via '<parameter file>' | <var1>{,<var2>,...}

FIT_LIMIT = 1e-6
f(x) = a*x**2 + b*x + c
fit f(x) 'measured.dat' via 'start.par'
fit f(x) 'measured.dat' using 3:($7-5) via 'start.par'
fit f(x) './data/trash.dat' using 1:2:3 via a, b, c

g(x,y) = a*x**2 + b*y**2 + c*x*y
fit g(x,y) 'surface.dat' using 1:2:3:(1) via a, b, c

fit a0 + a1*x/(1 + a2*x/(1 + a3*x)) 'measured.dat' via a0,a1,a2,a3
fit a*x + b*y 'surface.dat' using 1:2:3:(1) via a,b
fit [*:*][yaks=*:*] a*x+b*yaks 'surface.dat' u 1:2:3:(1) via a,b
fit a*x + b*y + c*t 'foo.dat' using 1:2:3:4:(1) via a,b,c

h(x,y,t,u,v) = a*x + b*y + c*t + d*u + e*v
```

AcSIR

# Commands VI

```
fit h(x,y,t,u,v) 'foo.dat' using 1:2:3:4:5:6:(1) via a,b,c,d,e
```

10. **stat (Statistical Summary)**
`stats 'filename' [using N[:M]] [name 'prefix'] [[no]output]]` It also produces three set of variables

```
first set: It reports how the data is laid out in the file:
STATS_records            # total number of in-range data records
STATS_outofrange         # number of records filtered out by range limits
STATS_invalid            # number of invalid/incomplete/missing records
STATS_blank              # number of blank lines in the file
STATS_blocks             # number of indexable data blocks in the file
```

```
The second set reports properties of the in-range data from a single colum
If the corresponding axis is autoscaled (x-axis for the 1st column, y-axis
for the optional second column) then no range limits are applied.
If two columns are being analysed in a single ‘stats‘ command, the the suf
"_x" or "_y" is appended to each variable name. I.e. STATS_min_x is the mi
value found in the first column, while STATS_min_y is the minimum value fo
```

# Commands VII

```
in the second column.
STATS_min                # minimum value of in-range data points
STATS_max                # maximum value of in-range data points
STATS_index_min          # index i for which data[i] == STATS_min
STATS_index_max          # index i for which data[i] == STATS_max
STATS_lo_quartile        # value of the lower (1st) quartile boundary
STATS_median             # median value
STATS_up_quartile        # value of the upper (3rd) quartile boundary
STATS_mean               # mean value of in-range data points
STATS_stddev             # standard deviation of the in-range data points
STATS_sum                # sum
STATS_sumsq              # sum of squares

The third set of variables is only relevant to analysis of two data column

STATS_correlation        # correlation coefficient between x and y values
STATS_slope              # A corresponding to a linear fit y = Ax + B
STATS_intercept          # B corresponding to a linear fit y = Ax + B
STATS_sumxy              # sum of x*y
```

AcSIR

# Commands VIII

```
STATS_pos_min_y          # x coordinate of a point with minimum y value
STATS_pos_max_y          # x coordinate of a point with maximum y value

It may be convenient to track the statistics from more than one file at th

stats "file1.dat" using 2 name "A"
stats "file2.dat" using 2 name "B"
if (A_mean < B_mean) {...}
```

AcSIR

# Examples I

http://gnuplot-surprising.blogspot.com/2012/05/how-to-pick-out-maximum-and-minimum.html

**Data**

```
0.1   0.28901
0.2   0.05063
0.3   0.72124
0.4   0.28427
0.5   0.50505
0.6   0.10181
0.7   0.00846
0.8   0.36249
0.9   0.48757
1.0   0.59509
1.1   0.86525
1.2   0.69662
1.3   0.50589
1.4   0.33813
1.5   0.10803
```

```
reset
set term png
set output "max_min.png"
stats "data.dat" u 1:2 nooutput
set xrange [STATS_min_x:STATS_max_x]
set label 1 "Maximun" at STATS_pos_max_y,STATS_max_y\
                                          offset 1,-0.5
set label 2 "Minimun" at STATS_pos_min_y,STATS_min_y \
                                           offset 1,0.5
plot "data.dat" w p pt 3 lc rgb"#ff0000" notitle, \
        STATS_min_y w l lc rgb"#00ffff" notitle, \
          STATS_max_y w l lc rgb"#00ffff" notitle
set output
```

# Examples II

http://www.usm.uni-muenchen.de/people/puls/lessons/intro_general/gnuplot/gnuplot_for_beginners.pdf

```
f1(x) = a1*tanh(x/b1)  # define the function to be fit
 a1 = 300; b1 = 0.005;  # initial guess for a1 and b1
 fit f1(x) 'force.dat' using 1:2 via a1, b1
Final set of parameters Asymptotic Standard Error
======================= ==========================
a1 = 308.687 +/- 10.62 (3.442%)
b1 = 0.00226668 +/- 0.0002619 (11.55%)
```

```
f2(x) = a2 * tanh(x/b2) # define the function to be fit
a2 = 300; b2 = 0.005; # initial guess for a and b
fit f2(x) 'force.dat' using 1:3 via a2, b2
Final set of parameters Asymptotic Standard Error
======================= ==========================
a2 = 259.891 +/- 12.82 (4.933%)
b2 = 0.00415497 +/- 0.0004297 (10.34%)
```

AcSIR

# Examples III

```
set key at 0.018,150 title "F(x) = A tanh (x/B)"       # title to key!
set title "Force Deflection Data \n and curve fit"     # note newline!
set pointsize 1.5 # larger point!
set xlabel 'Deflection, {/Symbol D}_x (m)'             # Greek symbols!
set ylabel 'Force, {/Times-Italic F}_A, (kN)'          # italics!
plot "force.dat" using 1:2 title 'Column data' with points pt 3, \
      "force.dat" using 1:3 title 'Beam data' with points pt 4, \
      a1 * tanh( x / b1 ) title 'Column-fit: A=309, B=0.00227', \
      a2 * tanh( x / b2 ) title 'Beam-fit: A=260, B=0.00415'
```

AcSIR

1. http://lowrank.net/gnuplot/intro/basic-e.html
2. http://www.usm.uni-muenchen.de/people/puls/lessons/intro_general/gnuplot/gnuplot_for_beginners.pdf