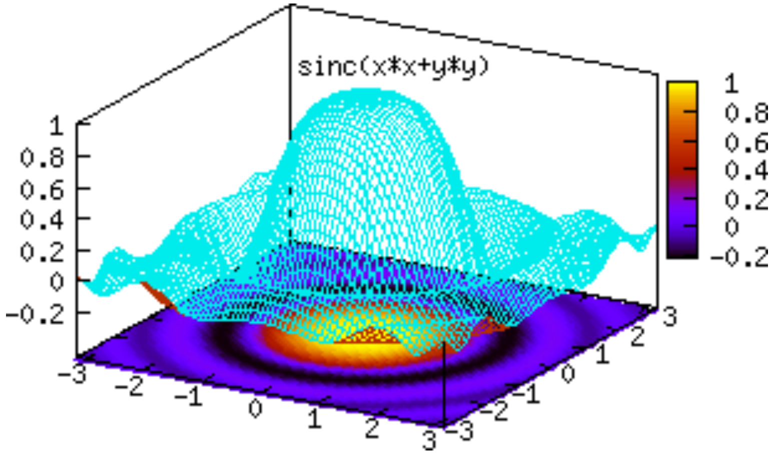
gnuplot

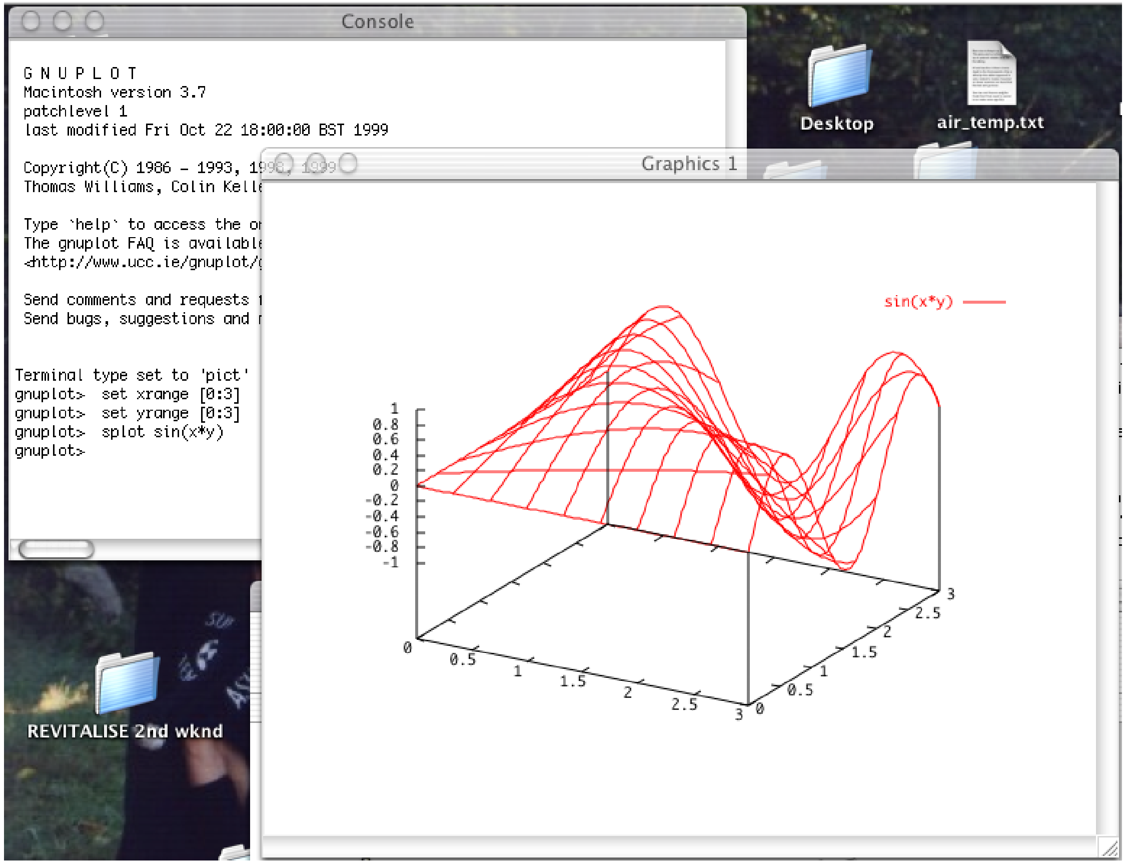
GnuPlot

- www.gnuplot.info
- A free GNU tool.
- Supports 2D & 3D Plotting.
- Very feature rich.
- Command line usage.
- Great for quick and dirty plots and text.



Basic Concepts

- Console Window
  - Command line prompt is *gnuplot>*
  - Menu selections for Windows version
- Plotting Window
- Help
  - ?
  - Help <topic>



```
G N U P L O T
Macintosh version 3.7
patchlevel 1
last modified Fri Oct 22 18:00:00 BST 1999

Copyright(C) 1986 - 1993, 1998, 1999
Thomas Williams, Colin Kell

Type `help` to access the o
The gnuplot FAQ is availabl
<http://www.ucc.ie/gnuplot/

Send comments and requests
Send bugs, suggestions and

Terminal type set to 'pict'
gnuplot>  set xrange [0:3]
gnuplot>  set yrange [0:3]
gnuplot>  splot sin(x*y)
gnuplot>
```

Data file output.dat arranged in columns…

Example file: “output.dat”

```
0.001000   72.565480   0.000435   0.015116   0.018278   1.209130
0.002000   72.520960   0.000870   0.015110   0.021045   1.392828
0.003000   72.476440   0.001305   0.015103   0.023514   1.556945
0.004000   72.431920   0.001741   0.015096   0.025747   1.705559
0.005000   72.387400   0.002176   0.015090   0.027789   1.841619
0.006000   72.342880   0.002612   0.015083   0.029673   1.967315
0.007000   72.298360   0.003047   0.015076   0.031423   2.084319
0.008000   72.253840   0.003482   0.015069   0.033061   2.193931
0.009000   72.209320   0.003918   0.015063   0.034602   2.297184
0.010000   72.164800   0.004354   0.015056   0.036058   2.394906
0.011000   72.120280   0.004789   0.015049   0.037439   2.487776
```

Doesn’t matter if it neatly lines up, gnuplot just looks for blank spaces….

The default is gnuplot assumes column 1 is x data and column 2 is y data…

Set terminal…. Where the output goes…

To plot to screen with just text characters to preview…

```
gnuplot> set terminal dumb
```

To make a postscript file

```
gnuplot> set terminal postscript
```

To make a jpeg file

```
gnuplot> set terminal jpeg
```

To output plot to a file…

```
gnuplot> set output ‘picture.ps’

gnuplot> set output ‘picture.jpeg’
```

To plot data in a file…

```
gnuplot> plot ‘output.dat’
```

To plot file in along a certain x,y range…

```
gnuplot> plot [0:2][-2:2] ‘output.dat’
```

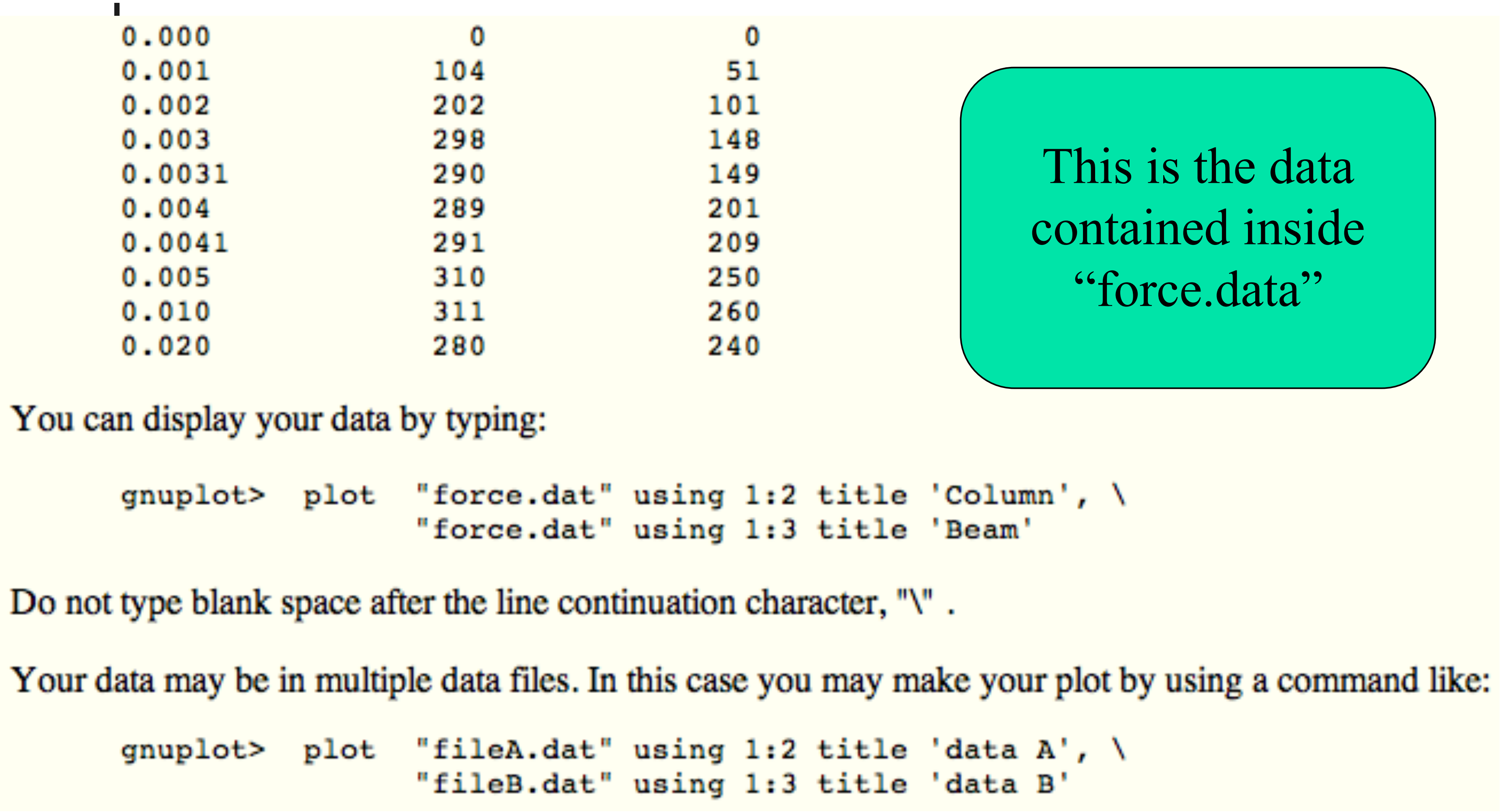
To plot different columns, for example col. 2 for x and col. 3 for y…

```
gnuplot> plot ‘output.dat’ using 2:3
```

Default is points… to plot with lines…

```
gnuplot> plot ‘output.dat’ using lines
```

```
    1                2                3
```



```
0.000          0        0
0.001        104       51
0.002        202      101
0.003        298      148
0.0031       290      149
0.004        289      201
0.0041       291      209
0.005        310      250
0.010        311      260
0.020        280      240
```

This is the data contained inside "force.data"

You can display your data by typing:

```
gnuplot>  plot  "force.dat" using 1:2 title 'Column', \
                "force.dat" using 1:3 title 'Beam'
```

Do not type blank space after the line continuation character, "\" .

Your data may be in multiple data files. In this case you may make your plot by using a command like:

```
gnuplot>  plot  "fileA.dat" using 1:2 title 'data A', \
                "fileB.dat" using 1:3 title 'data B'
```

Function Plots

- 2-D variable is x

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

- Use replot to display a second function

  ```
  gnuplot> replot cos(x)
  ```

- Math functions in fortran style

  ```
  gnuplot> plot (1.2e-5)*(x**1.4)/exp(x)
  ```

- Many functions (use ‘help functions’ to list)
  - Eg. abs() sinh() log() rand() acos() sqrt()

Operators and their usages:

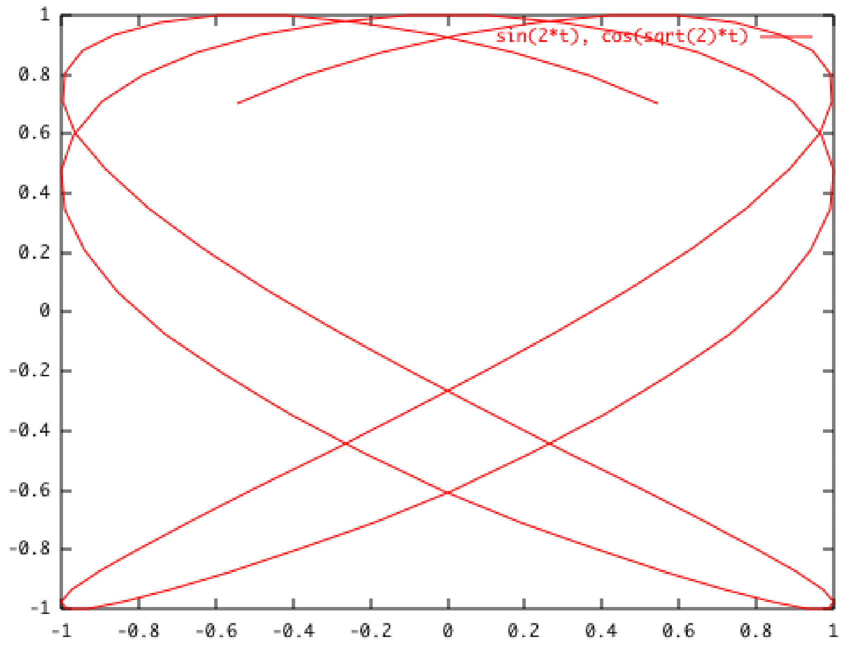
Symbol	Example	Explanation
**	a**b	exponentiation
*	a*b	multiplication
/	a/b	division
+	a+b	addition
-	a-b	subtraction
==	a==b	equality
!=	a!=b	inequality
&&	a&&b	logical AND
ll	allb	logical OR

Parametric Plots

```
gnuplot> set parametric
```

- 2-D "dummy" variable is t

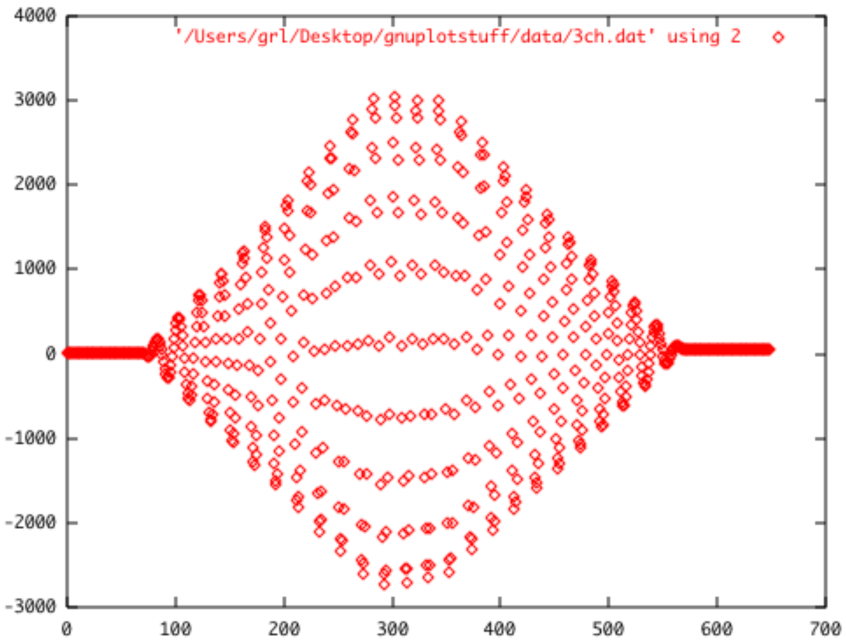
```
gnuplot> plot sin(2*t),
  cos(sqrt(2)*t)
```



Data Plots

- Single channel

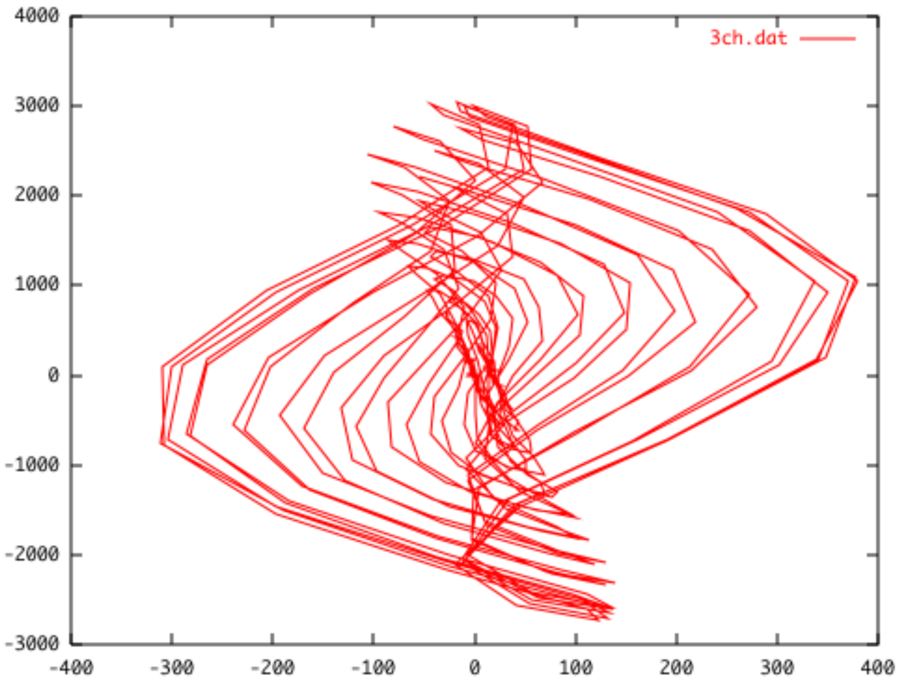
```
gnuplot> plot '/phys381/data/381.dat' using 2
```



Data Plots

- Cross plots

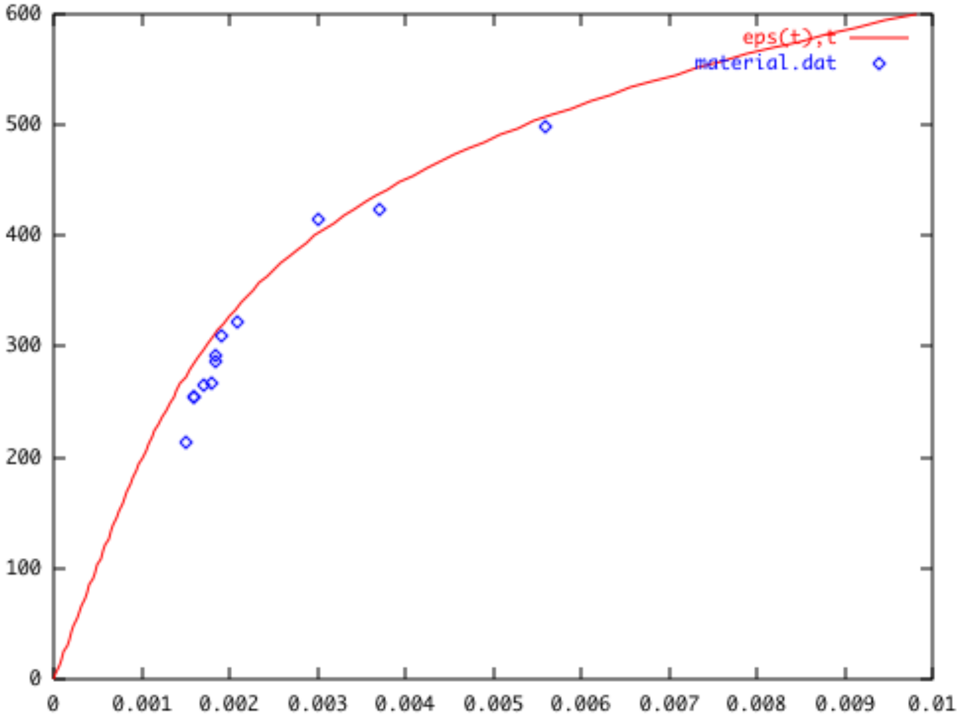
  gnuplot> plot ’381.dat' using 3:2 with lines



Combined Plots

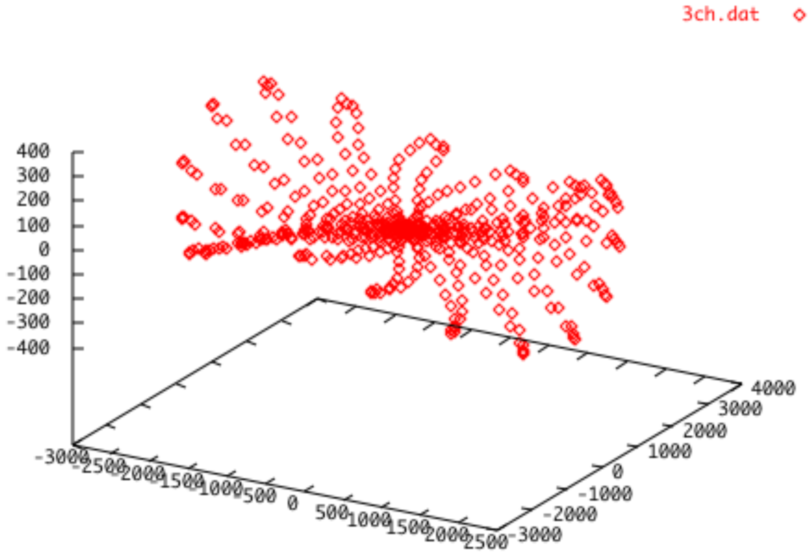
gnuplot> set parametric

gnuplot> plot eps(t),t, ’material.dat'



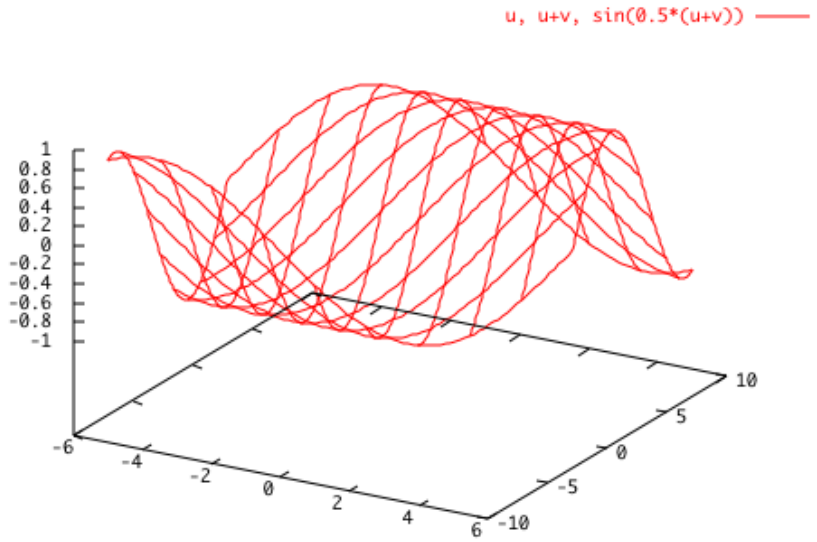
3-D Scatter Plots

```
gnuplot> splot ’381.dat'
```



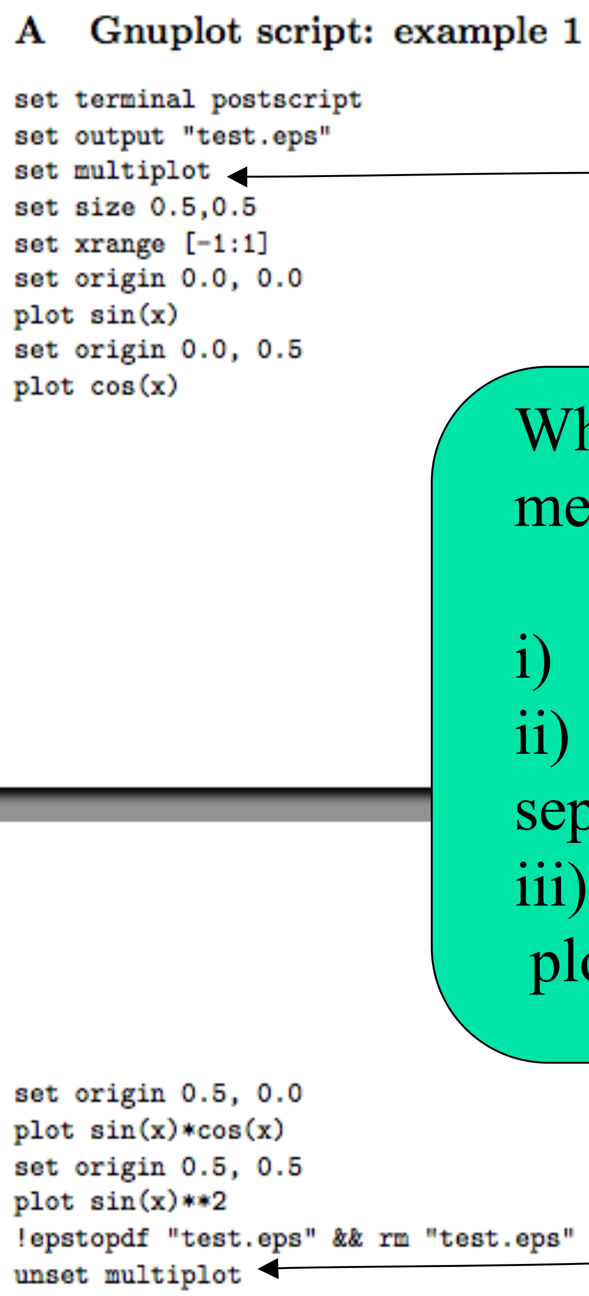
3-D Function Plots

```
gnuplot> set parametric
gnuplot> splot u,u+v,sin(0.5*(u+v))
```



Multiplot

You MUST read Appendix C in Ouyed&Butler textbook!



A Gnuplot script: example 1

```
set terminal postscript
set output "test.eps"
set multiplot
set size 0.5,0.5
set xrange [-1:1]
set origin 0.0, 0.0
plot sin(x)
set origin 0.0, 0.5
plot cos(x)
```

```
set origin 0.5, 0.0
plot sin(x)*cos(x)
set origin 0.5, 0.5
plot sin(x)**2
!epstopdf "test.eps" && rm "test.eps"
unset multiplot
```

When you set MULTIPLOT it could mean:

i) Plot multiple figures in one window

ii) plot multiple figures EACH in a separate panel

iii) Set multiple panels with multiple plots in each one of them

Multiplot example

- stack several `plot` commands

  `set multiplot`

- scale the plot

  `set size`

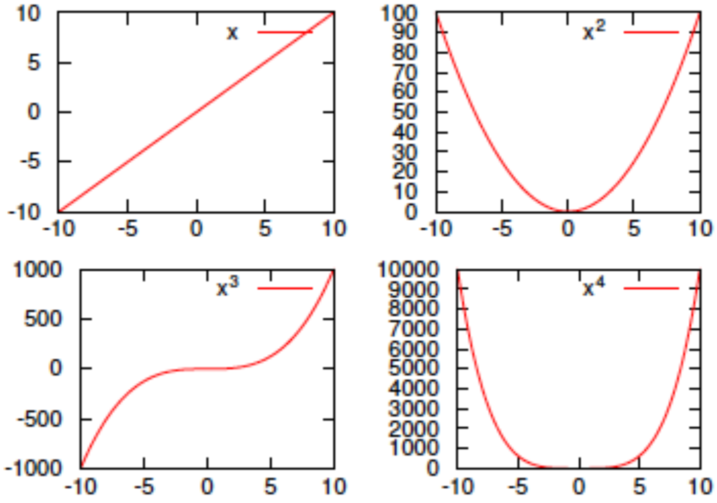
- place the plot

  `set origin`

- leave multiplot mode

  `unset multiplot`

```
set multiplot
set origin 0,0
set size 0.5,0.5
plot x*x*x t "x^3"
set origin 0,0.5
plot x t "x"
set origin 0.5,0.5
plot x*x t "x^2"
set origin 0.5,0
plot x*x*x*x t "x^4"
unset multiplot
```



Gnuplot codes also referred to as …

``Scripts”

For your scientific work you will prefer scripts.
Store the commands in a text file.

myscript.gnuplot

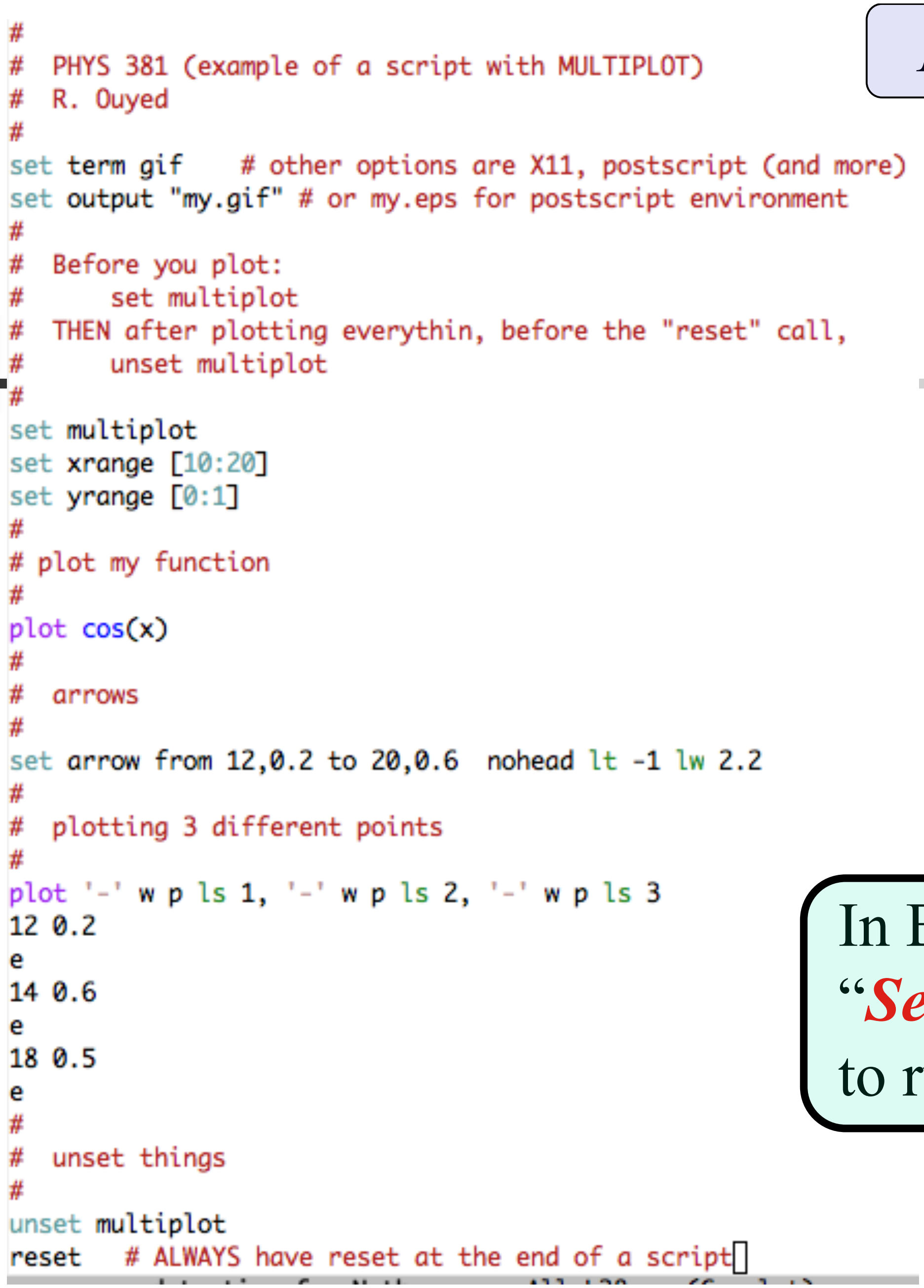
```
set term wxt                                              # select display type
set ytics 0.5                                             # define distance of labeled
                                                          #  tick marks on the y-axis
set mytics 5                                              # number of small
                                                          #  tick marks on the y-axis
set xrange [-pi:pi]                                       # set x range of the plot
set xtics ("-pi" -pi,"-pi/2" -pi/2,0,"pi/2" pi/2,"pi" pi) # custom labels
set samples 200                                           # increase sampling rate
set key at -pi/8,0.8 invert samplen 2                     # place and format
                                                          #  key of symbols
f(x) = sin(x)                                             # define a function
plot f(x) t "sin(x)", f(2*x) t "sin(2x)"                  # plot two functions
pause -1  "hit␣ENTER␣to␣exit␣script"                      # don't close gnuplot
```

load this script in GNUPLOT by typing either

- `gnuplot myscript.gnuplot` on the command-line
- `load 'myscript.gnuplot'` in GNUPLOT

File: ouyed-example.gp

Extension “gp”



```
#
#  PHYS 381 (example of a script with MULTIPLOT)
#  R. Ouyed
#
set term gif     # other options are X11, postscript (and more)
set output "my.gif" # or my.eps for postscript environment
#
#  Before you plot:
#      set multiplot
#  THEN after plotting everythin, before the "reset" call,
#      unset multiplot
#
set multiplot
set xrange [10:20]
set yrange [0:1]
#
# plot my function
#
plot cos(x)
#
#  arrows
#
set arrow from 12,0.2 to 20,0.6  nohead lt -1 lw 2.2
#
#  plotting 3 different points
#
plot '-' w p ls 1, '-' w p ls 2, '-' w p ls 3
12 0.2
e
14 0.6
e
18 0.5
e
#
#  unset things
#
unset multiplot
reset   # ALWAYS have reset at the end of a script
```

In Emacs select:
“***Send Buffer to Gnuplot***”
to run the script.

Gnuplot and PDF figures

The Debian version of gnuplot ships without pdf support due to legal restraints.

This is what you need to add to your gnuplot script (file) to save your plot in “pdf” format:

```
set terminal postscript eps enhanced color solid rounded

set output "plotname.eps”

... then your commands here ….

!epstopdf plotname.eps && rm plotname.eps
```

What’s the “!” mark For ?

The last line invokes a linux converter that turns your .eps plot into an .pdf plot.

If you want to keep the eps file remove the “&& rm plotname.eps” part.

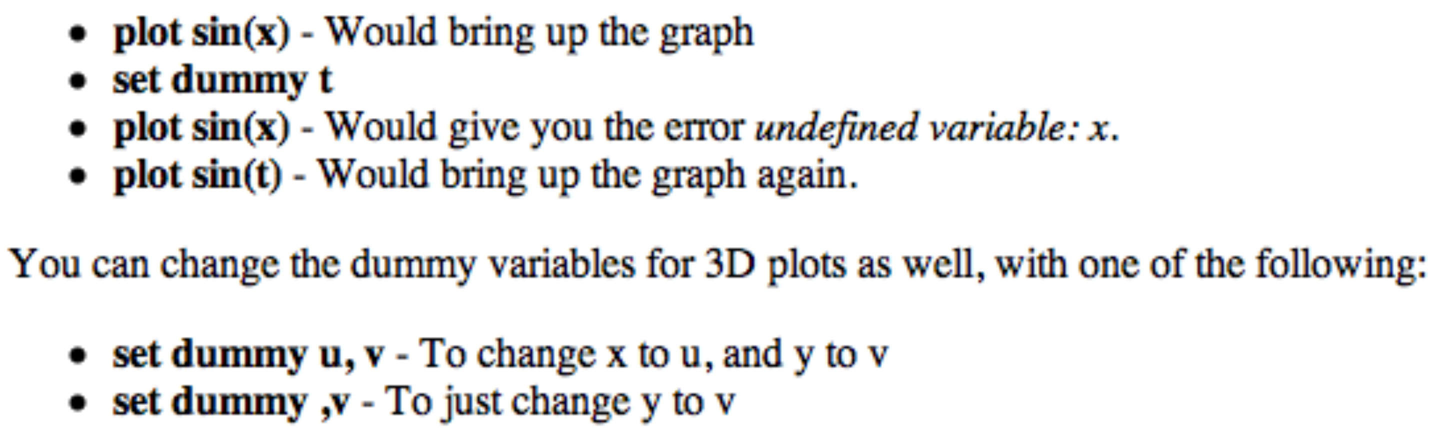
Dummy Variable

Changing Variables

Sometime, you would rather use other variables names than *x and y* for your plots. GnuPlot allows you to change the name of the independent variable.

You do this using the ***set dummy*** command.

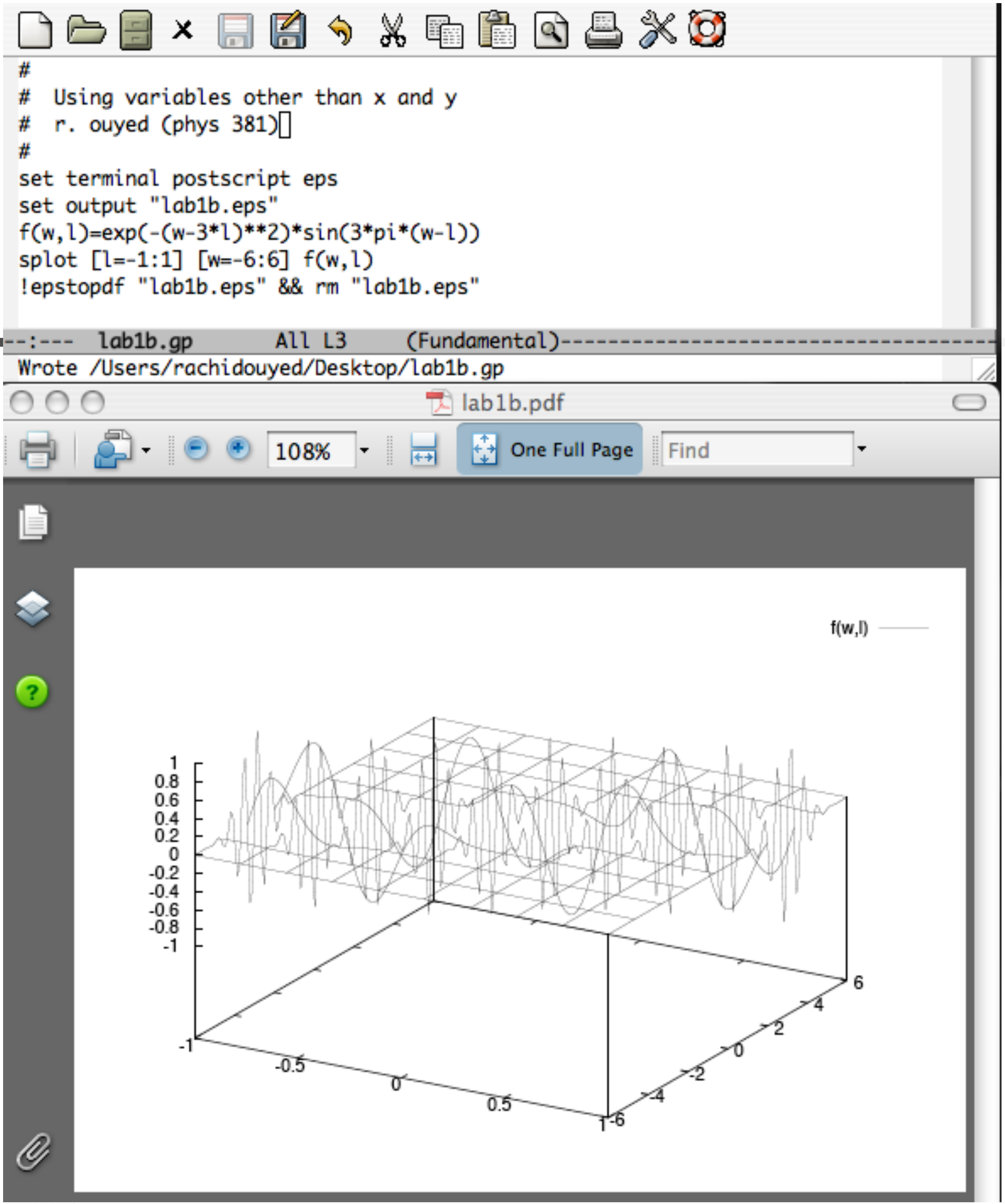
For instance, if you were working with 2D plots with *t* as the independent variable,



- **plot sin(x)** - Would bring up the graph
- **set dummy t**
- **plot sin(x)** - Would give you the error *undefined variable: x*.
- **plot sin(t)** - Would bring up the graph again.

You can change the dummy variables for 3D plots as well, with one of the following:

- **set dummy u, v** - To change x to u, and y to v
- **set dummy ,v** - To just change y to v



```
#
#  Using variables other than x and y
#  r. ouyed (phys 381)
#
set terminal postscript eps
set output "lab1b.eps"
f(w,l)=exp(-(w-3*l)**2)*sin(3*pi*(w-l))
splot [l=-1:1] [w=-6:6] f(w,l)
!epstopdf "lab1b.eps" && rm "lab1b.eps"
```

```
--:---  lab1b.gp      All L3     (Fundamental)------------------------------------
Wrote /Users/rachidouyed/Desktop/lab1b.gp
```

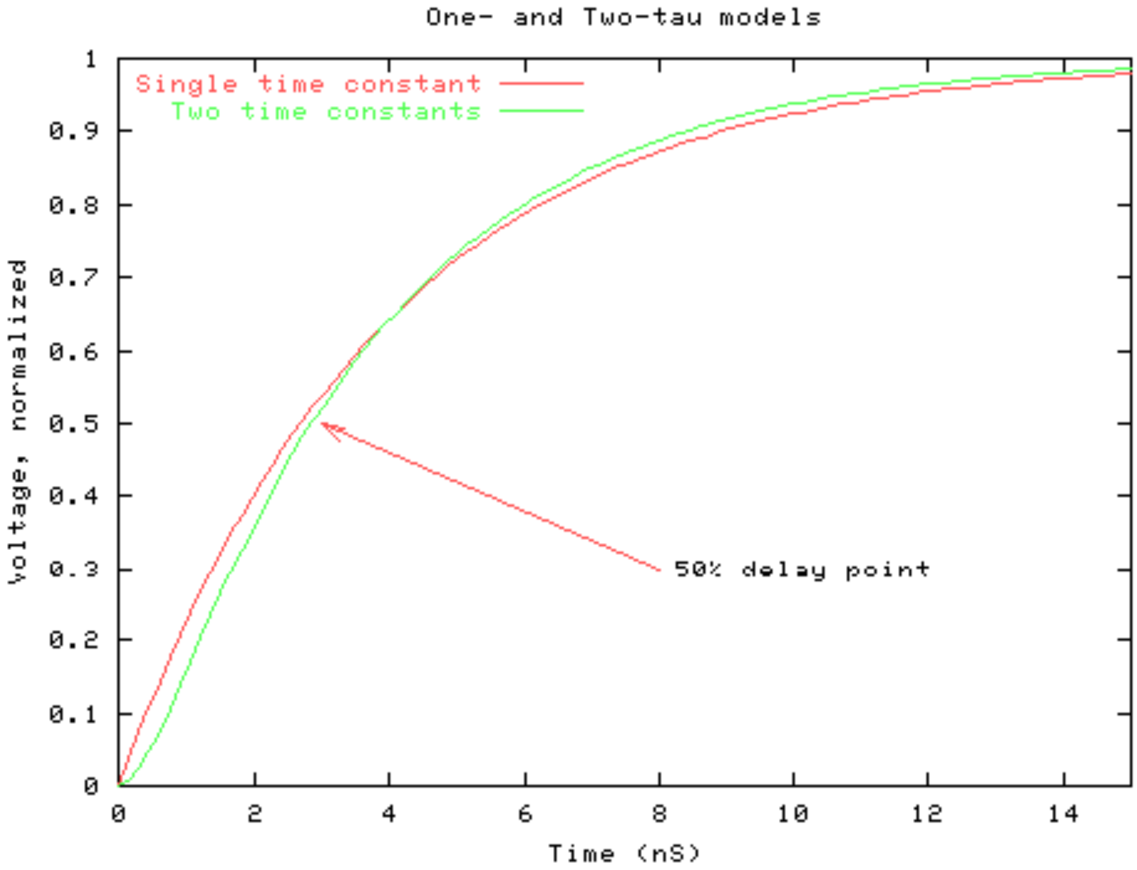
Study The Following Examples

1-My First Graph: basics

Script:

Each step is followed by a "replot"

```
➢plot 1-exp(-x/3.8825)
➢pause -1
➢set xrange [0:15]; replot
➢plot 1-exp(-x/3.8825) title "Single
time constant"
➢set xlabel "Time (nS)"; replot
➢set ylabel "Voltage, normalized"
➢set key top left
➢replot 1-(3.44*exp(-x/3.44)-0.44*exp(-
x/0.44))/3.0 title "Two time constants"
➢set title "One- and Two-tau models"
➢set arrow 1 from 8,0.3 to 3.0,0.5 head
➢set label 1 "50% delay point" at
8.2,0.3 left
```



Related commands:

```
➢set key x,y
➢set [no]log (x|y)
➢set autoscale (x|y)
```

➢*Note: Screen shots are low-quality to keep the file size down. High-quality .eps plots discussed later.*

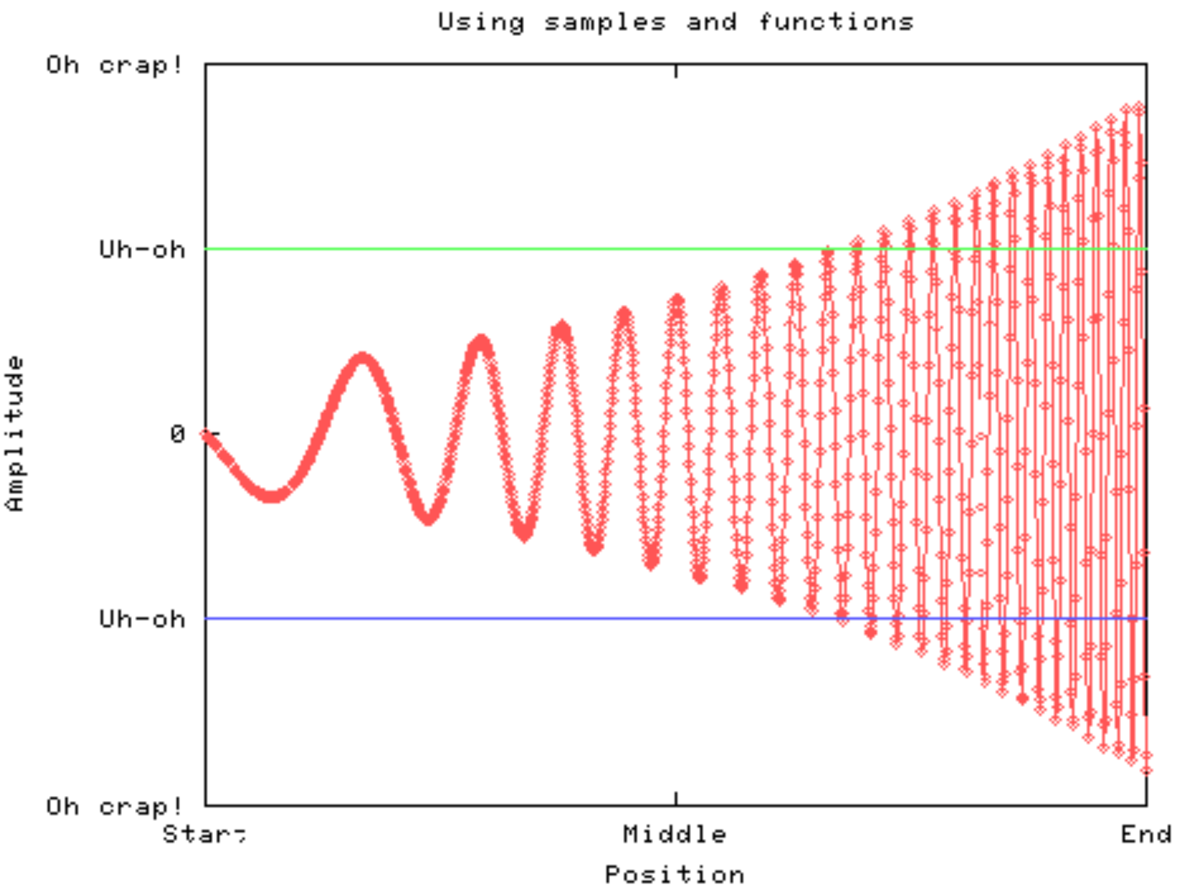
2-Plotting functions and sampling

Script:

```
➢clear; reset
➢set xrange [x1:x2]; set yrange [y1:y2]
➢set xlabel "…"; set ylabel "…"
➢set title "Using samples and functions"
➢f(x) = x**5
➢pi = 3.14159; sf = 4.5
➢plot (sf**x)*sin(f(x)*pi) notitle with linespoints
➢set samples 1000     ← normally, get 100 points
➢set xtics ("Start" 1, "Middle" 1.6, "End" 2.2)
➢set ytics ("Oh crap!" -30, "Uh-oh" –15, "0" 0, "Uh-oh" 15, "Oh crap!" 30)
➢replot 15 notitle; replot -15 notitle
```

Related commands:

```
➢show variables
➢show functions
➢high=110; f2c(t)=(x-32)*5.0/9.0
➢set yrange [f2c(20):f2c(high)]
```



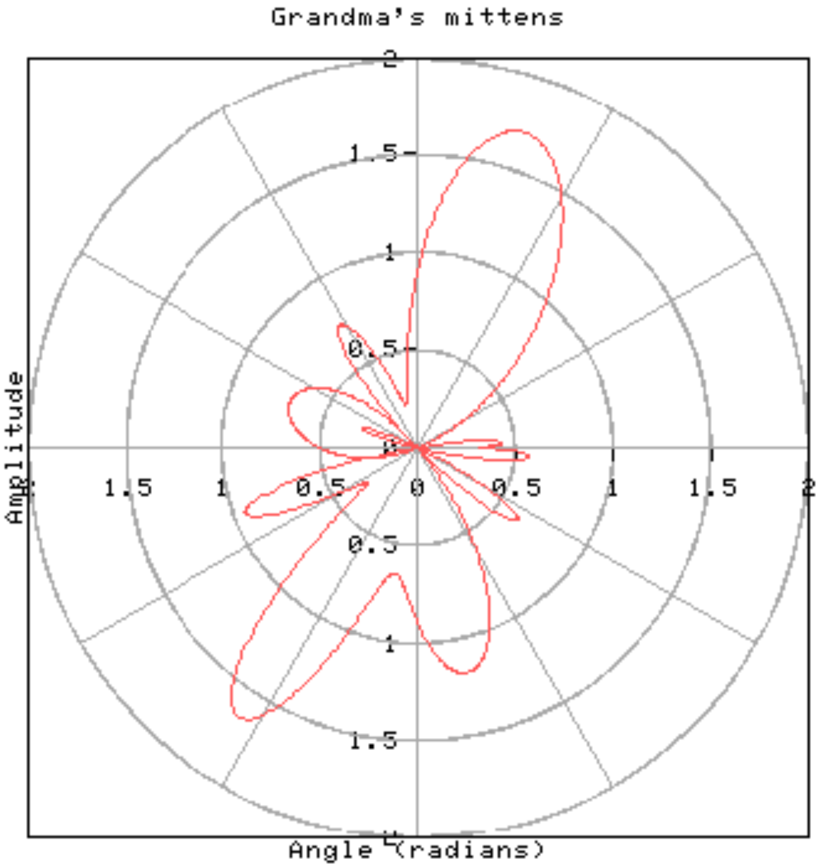
3-More 2D plots

Script:

- set samples 1000
- set xlabel; set ylabel; set title
- set xrange [-pi:pi] *← pi predefined!*
- plot sin(x)*cos(x) + sin(x)*sin(x) - 0.5*cos(2*x*x) notitle
- set grid
- set polar
- set trange[-pi:pi]
- plot sin(t)*cos(t) + sin(t)*sin(t) - 0.5*cos(2*t*t) notitle
- set grid polar
- set xtics axis; set ytics axis
- set xrange [-2:2]; set yrange [-2:2]
- set size square
- set title "Grandma's mittens"

Related commands:

- set size ratio *aspectratio*
- set size *xscale,yscale*
- set parametric *<- polar is a special case*



4-Basic 3D plots

Script:

Just to illustrate "set parametric"; could also use `splot sin(x)*cos(y)` *w/o parametric*

➢set xlabel; set ylabel

➢set zlabel "Amplitude"

➢set parametric

➢splot u,v,sin(u)*cos(v) title "Standing Waves"

➢set isosamples 75,75 ← *10 is normal*

➢set contour base

➢set cntrparam level incremental -1, 0.2, 10 ← *start,incr,num*

➢set clabel '%4.2f' ← *C's scanf*

➢set contour surface

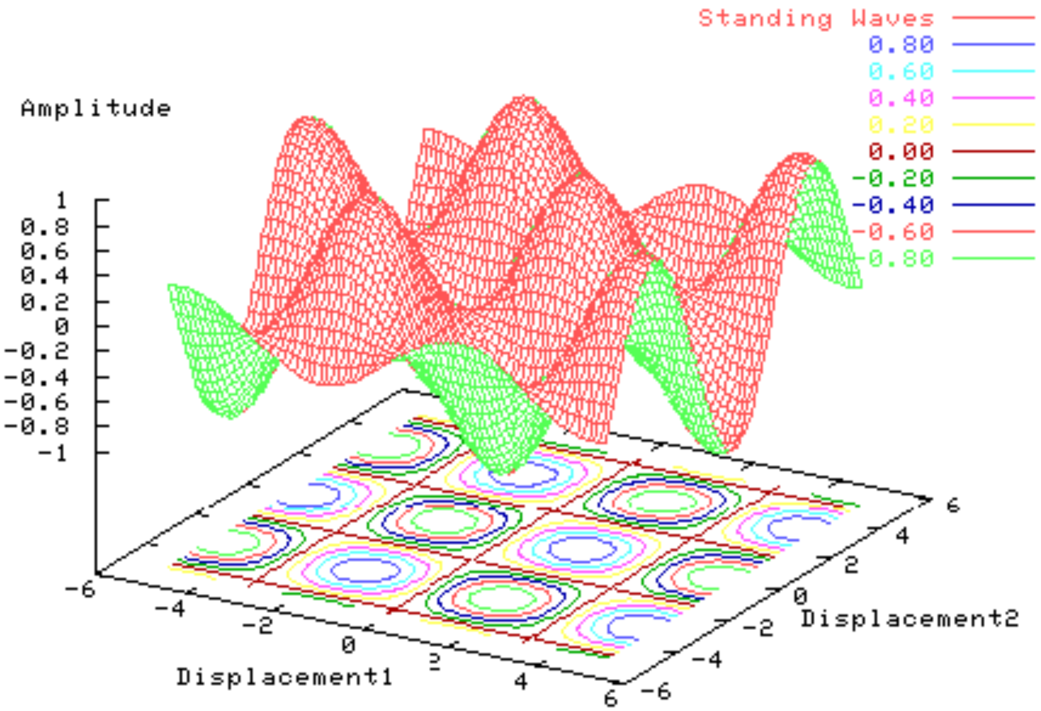
➢set contour base; set nosurface

➢set surface;

➢set view 20,60

➢set view 60,30 ← *xrot, zrot*

➢set hidden3d



Related commands:

➢set contour [base|surface|both]

➢set [no]surface

5-Plotting from data files

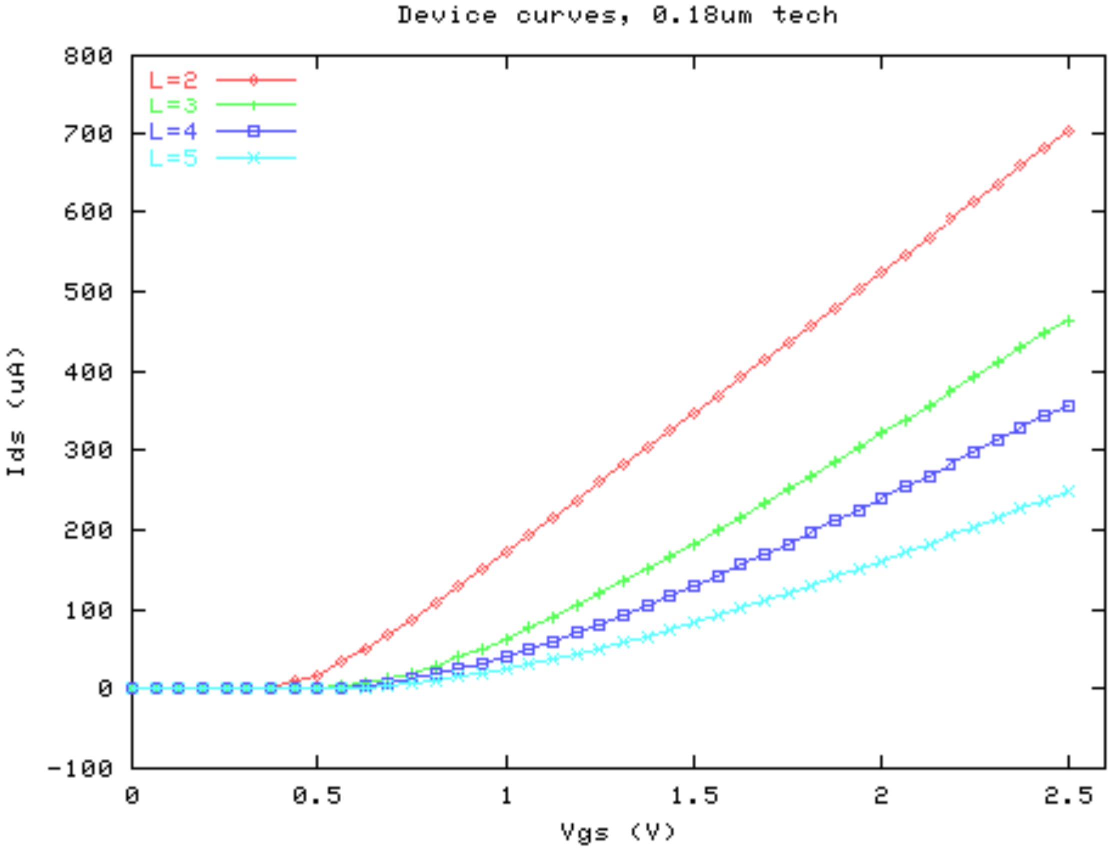
Script:

- set xlabel; set ylabel; set title
- set key top left
- plot "plot5.dat" title "IV curves"
- plot "plot5.dat" using ($1*2.5/2e-9):($2*-1e6) title "IV curves"
- set xlabel "Vgs (V)"; set ylabel "…"
- set xrange [0:2.6]
- plot "plot5.dat" index 2 using ($1*2.5/2e-9):($2*-1e6) title "L=4"
- replot "plot5.dat" index 3 using ($1*2.5/2e-9):($2*-1e6) title "L=5" with lines
- set data style linespoints
- plot "plot5.dat2" u ($1*2.5/2e9):($3*-1e6) title "L=3"
- plot "plot5.dat3" u ($1*2.5/2e9):($2*-1e6) '%lf,%lf,%lf,%lf,%lf' title "L=2"

Notes:

- plot <FILE> index n …

  requires \n\n between datasets



6-Plotting from data files

Script:

- `set xlabel; set ylabel; set title`
- `set xrange [0:110]`
- `plot "plot6.dat" u 1:3 t "Leakage" w p`

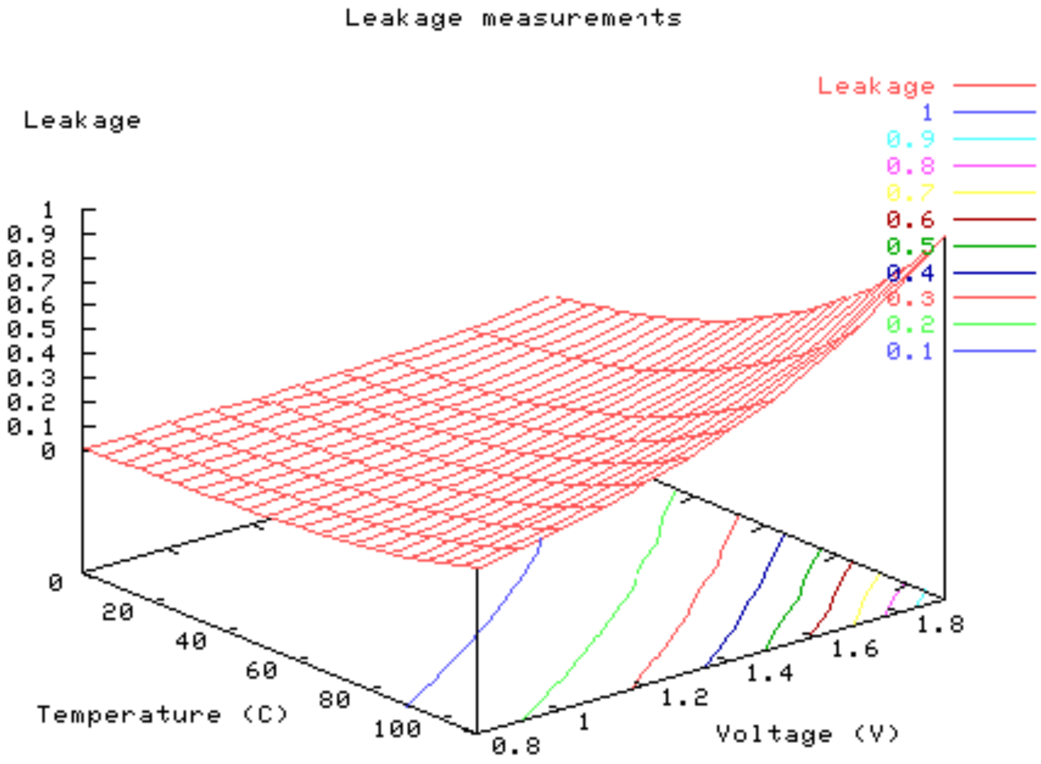
  only need "$" for expressions
- `plot "plot6.dat" u 1:3 t "Leakage" w l`
- `plot "plot6.dat2" u 1:3 t "Leakage" w l`

  \n in data prevents line-connecting
- `set xrange [0.8:1.9]`
- `set xlabel "Voltage (V)"`
- `plot "plot6.dat2" u 2:3 t "Leakage" w l`
- `set xrange [0:110]; set yrange [0.8:1.9`
- `set xlabel "Temperature (C)" ,-1`
- `set ylabel "Voltage (V)" ,-1`

  xoff=0,yoff=-1 in x's
- `set zlabel "Leakage"`
- `splot "plot6.dat2" u 1:2:3 t "Leak" w l`

  splot using x:y:z
- `set view ,50`
- `set contour base`
- `set hidden3d`

  only works for lines or linespoints



7-Axes. Ternary operations.

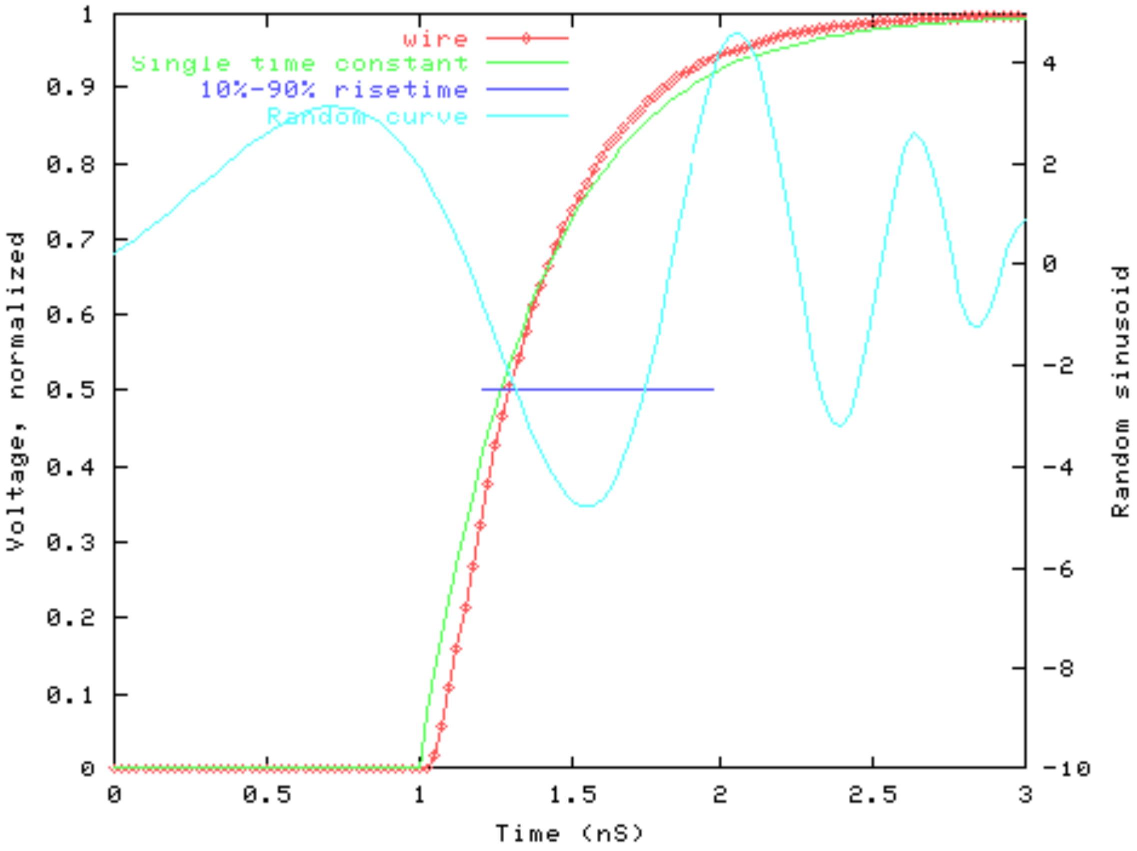
Script:

```
➢set xrange; set xlabel; set ylabel
➢set key top left
➢plot "plot7.dat" u ($1*1e9):($2/1.8) t "wire" w lp
➢replot 1-exp(-(x-1)/.38825) t "Single time constant"
➢plot "plot7.dat" u ($1*1e9):($2/1.8) t "wire" w lp
➢replot (x<1) ? 0 : 1-exp(-(x-1)/.38825) t "Single time constant"
➢replot x>1.2 && x<2 ? 0.5:1/0 t "10%-90% risetime"
➢replot 5*sin(exp(x))*sin(x)+0.2 axes x1y2 t "Random curve"
➢set y2tics
➢set ytics nomirror
➢set y2label "Random sinusoid"
➢set y2range [-10:5]
```

Related commands:

```
➢plot 'file' u 1:($4<0?1/0:($2+$3)/2)
```

plots average of $2,$3 only if $4>=0



8-Nifty side-note

Ternary operator surprisingly powerful!

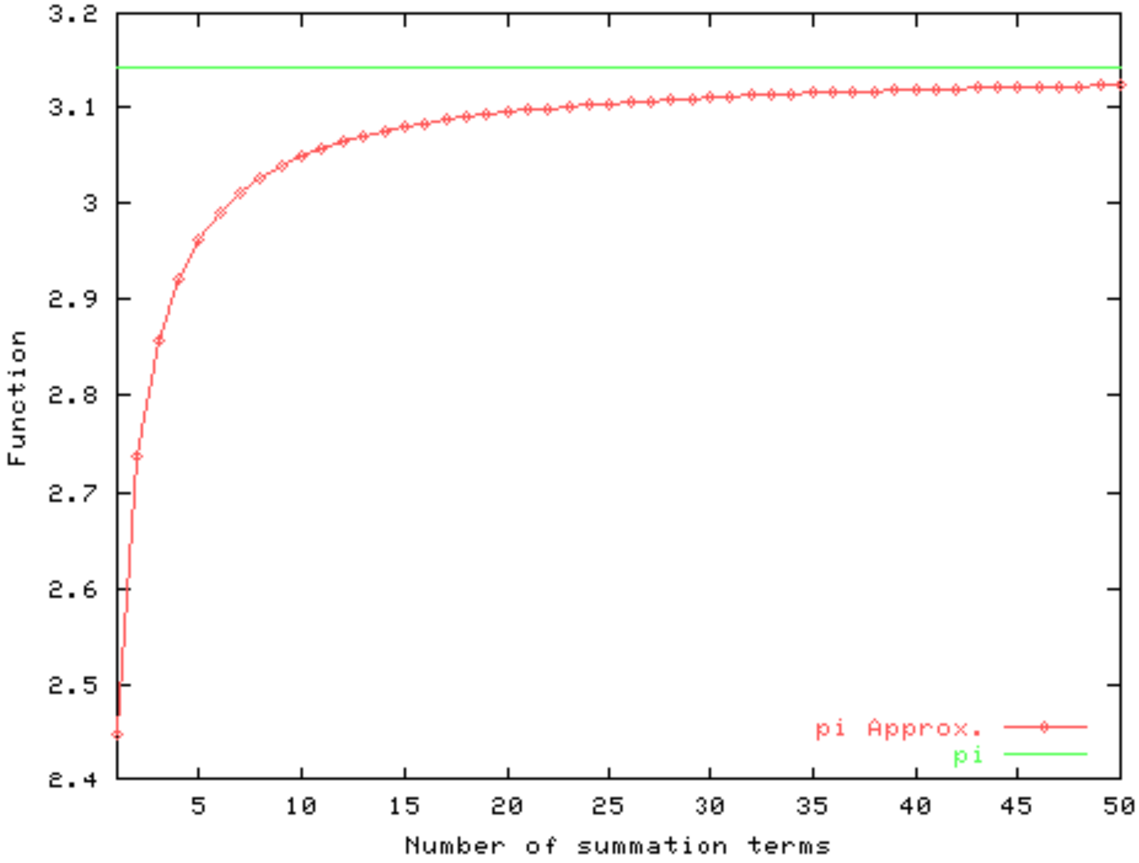
How quickly does $\sqrt{6\sum_{i=1}^{\infty}\frac{1}{i^2}}$ converge to pi?

Script:

```
➢set xlabel "Number of summation terms"
➢set ylabel "Function"
➢set xrange [1:50]
➢set samples 50        ← critical: integers only!
➢set key bottom right
➢f_part(x) = 1/(x*x)
➢f_sum(x) = f_part(x) + ((x>1) ?
f_sum(x-1) : 0)
➢f(x) = sqrt(6*f_sum(x))
➢plot f(x) title "pi Approx." w lp
➢replot pi
```

Answer: Not very quickly!

Note: *Stack space is limited; plotting from [0:100] runs out of stack space ☹ (do it using two functions)*



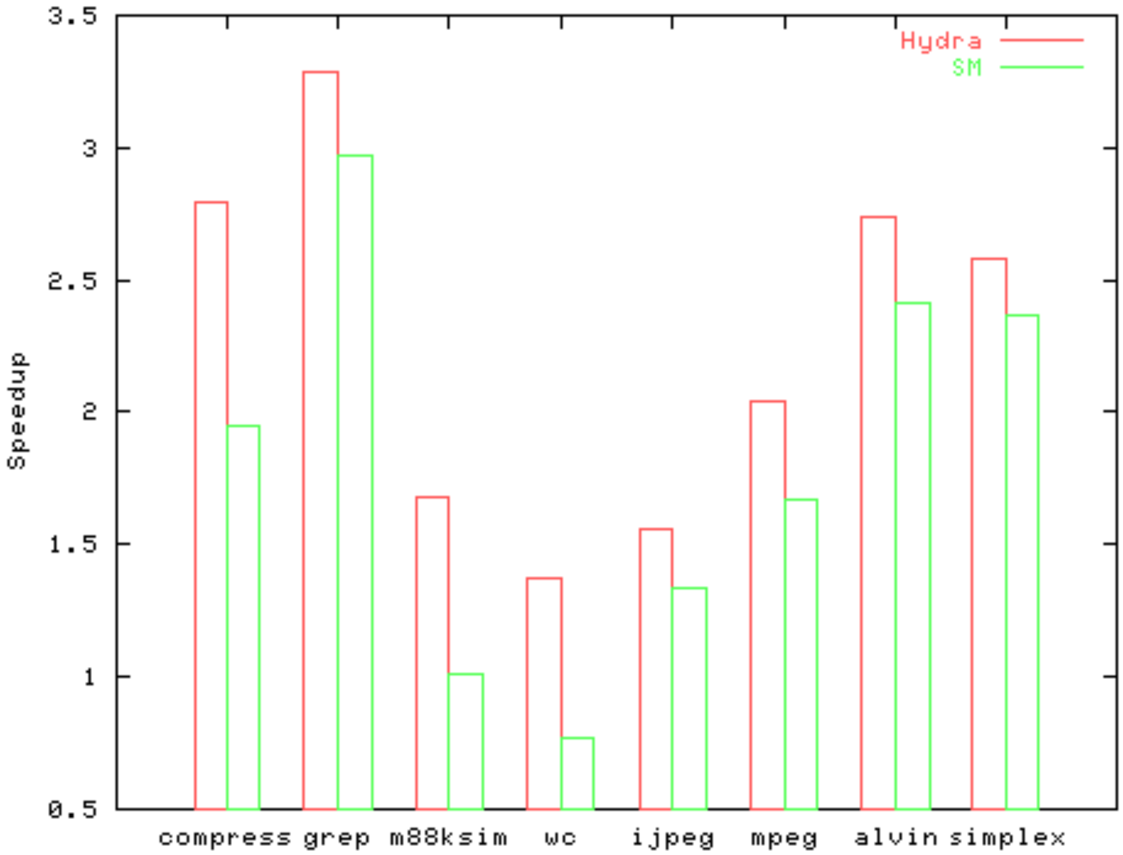
9-Bar graphs

Script:

```
➢set xtics ("compress" 1, "grep" 2,
"m88ksim" 3, "wc" 4, "ijpeg" 5, "mpeg"
6, "alvin" 7, "simplex" 8)

➢set ylabel "Speedup"

➢set xrange [0:9]

➢plot "plot8.dat" u 1:2 t "Hydra" w lp

➢replot "plot8.dat" u 1:3 t "SM" w lp

➢plot "plot8.dat" u 1:2 t "Hydra" w
boxes

➢replot "plot8.dat" u 1:3 t "SM" w
boxes

➢set boxwidth 0.3

➢plot "plot8.dat" u ($1-0.15):2 t
"Hydra" w boxes

➢replot "plot8.dat" u ($1+0.15):3 t
"SM" w boxes
```

Notes:

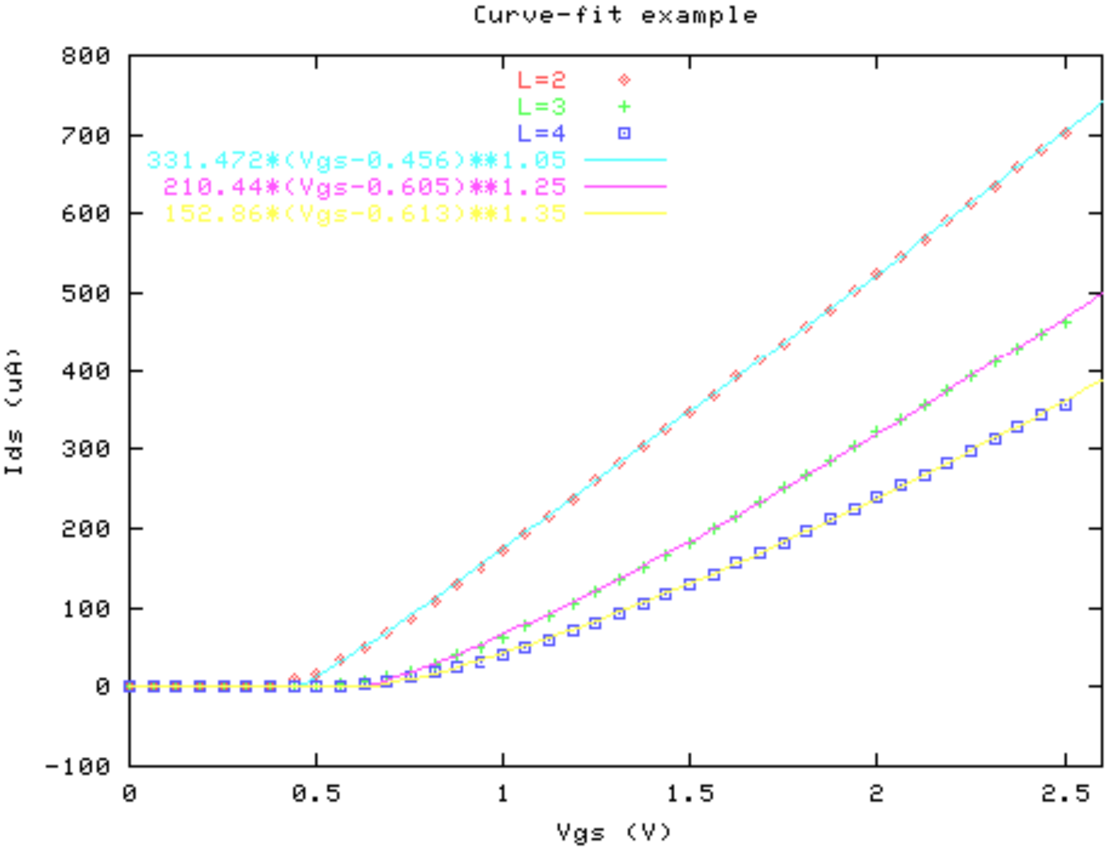
No way to fill in the boxes using stock gnuplot (although some post-processing hacks exist, including simply using Frame)



10-Curve-fitting

Script:

```
➢set xlabel; set ylabel; set title
➢set xrange [0:2.6]; set key
➢plot "plot5.dat2" u ($1*2.5/2e-9):($2*-1e6) t "L=2" w p
➢replot "plot5.dat2" u ($1*2.5/2e-9):($3*-1e6) t "L=3" w p
➢f1(x) = x>b1 ? a1*((x-b1)**c1) : 0
➢fit f1(x) "plot5.dat2" u ($1*2.5/2e-9):($2*-1e6) via a1,b1,c1
➢replot f1(x) title "331.472*(Vgs-0.456)**1.05" w l
➢f2(x) = x>b2 ? a2*((x-b2)**c2) : 0
➢fit f2(x) "plot5.dat2" u ($1*2.5/2e-9):($3*-1e6) via a2,b2,c2
➢replot f2(x) title "210.44*(Vgs-0.605)**1.25" w l
```



Notes:

Max 3000 data points for curvefitting

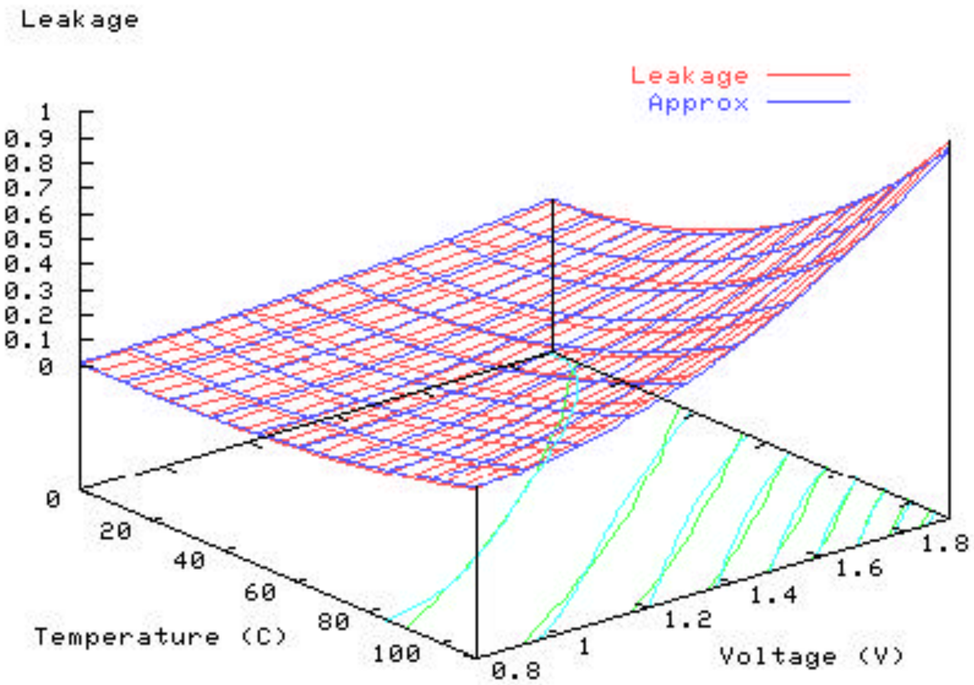
fit.log holds the iterative information

Must manually type in the fitted values for titles/labels. Most often requested feature for v3.8!

11-Curve-fitting, another example

```
Script:
➢(x|y}range; (x|y|z)label
➢set data style lines
➢set view ,50; set key 60,1.9,1
➢splot "plot6.dat2" u 1:2:3 t "Leakage"
➢f(x,y) = a+b*x+c*y
➢fit f(x,y) "plot6.dat2" u 1:2:3:(1) via
a,b,c
➢replot f(x,y)
➢splot "plot6.dat2" u 1:2:($3-f($1,$2)) not
➢f(x,y) = a+b*x+c*y*y+d*y+e*x*y*y+f*x*y
➢fit f(x,y) "plot6.dat2" u 1:2:3:(1) via
a,b,c,d,e,f; replot f(x,y)
➢splot "plot6.dat2" u 1:2:($3-f($1,$2)) not
➢f(x,y) = a + b*x*x + c*x + d*y*y + e*y +
f*x*x*y*y + g*x*x*y + h*x*y*y + i*x*y
➢fit f(x,y) "plot6.dat2" u 1:2:3:(1) via
a,b,c,d,e,f,g,h,i; replot f(x,y)
➢splot "plot6.dat2" u 1:2:($3-f($1,$2)) not
➢set contour base; set noclabel
➢splot "plot6.dat2" u 1:2:3 t "Leakage"
➢replot f(x,y) t "Approx"
```

x:y:z:(1) indicates evenly-weighted data. "help fit" for more details



More on Gnuplot …

A Data file containing indexed BLOCKS

SINGLE BLANK LINE:

A “datablock” in gnuplot is a set of data points separated by a single blank line.

For multiple graphs, a blank line in a file is interpreted as **'lift the pen'.**

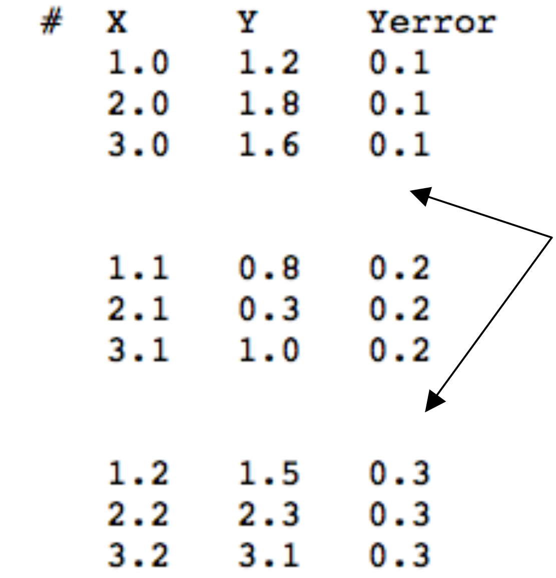
DOUBLE BLANK LINES:

Index

Two blank lines indicate a new 'index'.

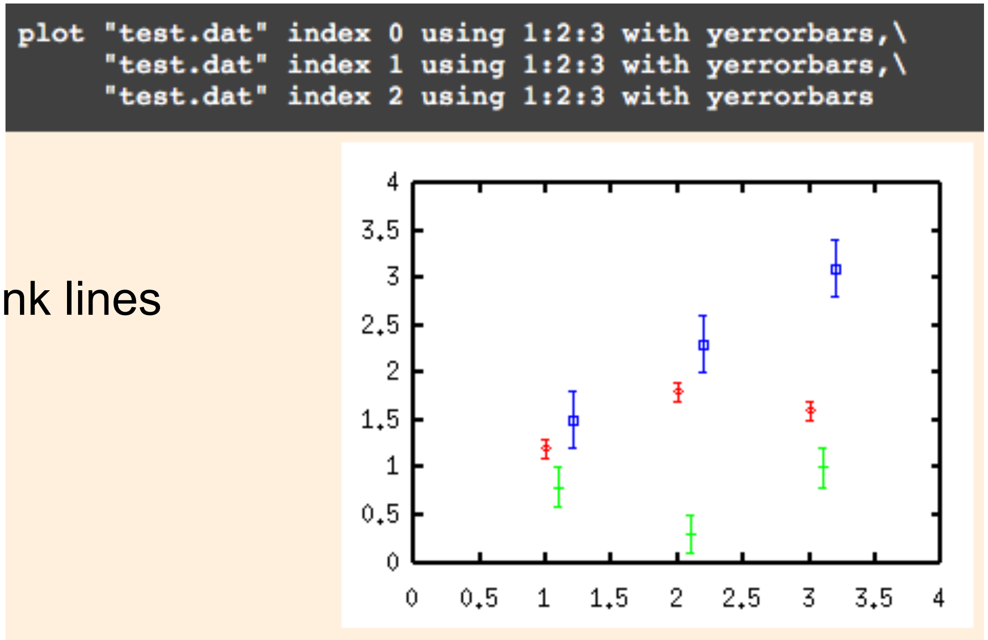
To plot only the data from indices 4 through 6 in a file, where index 0 is the first index: **plot 'file' index 4:6**

test.dat



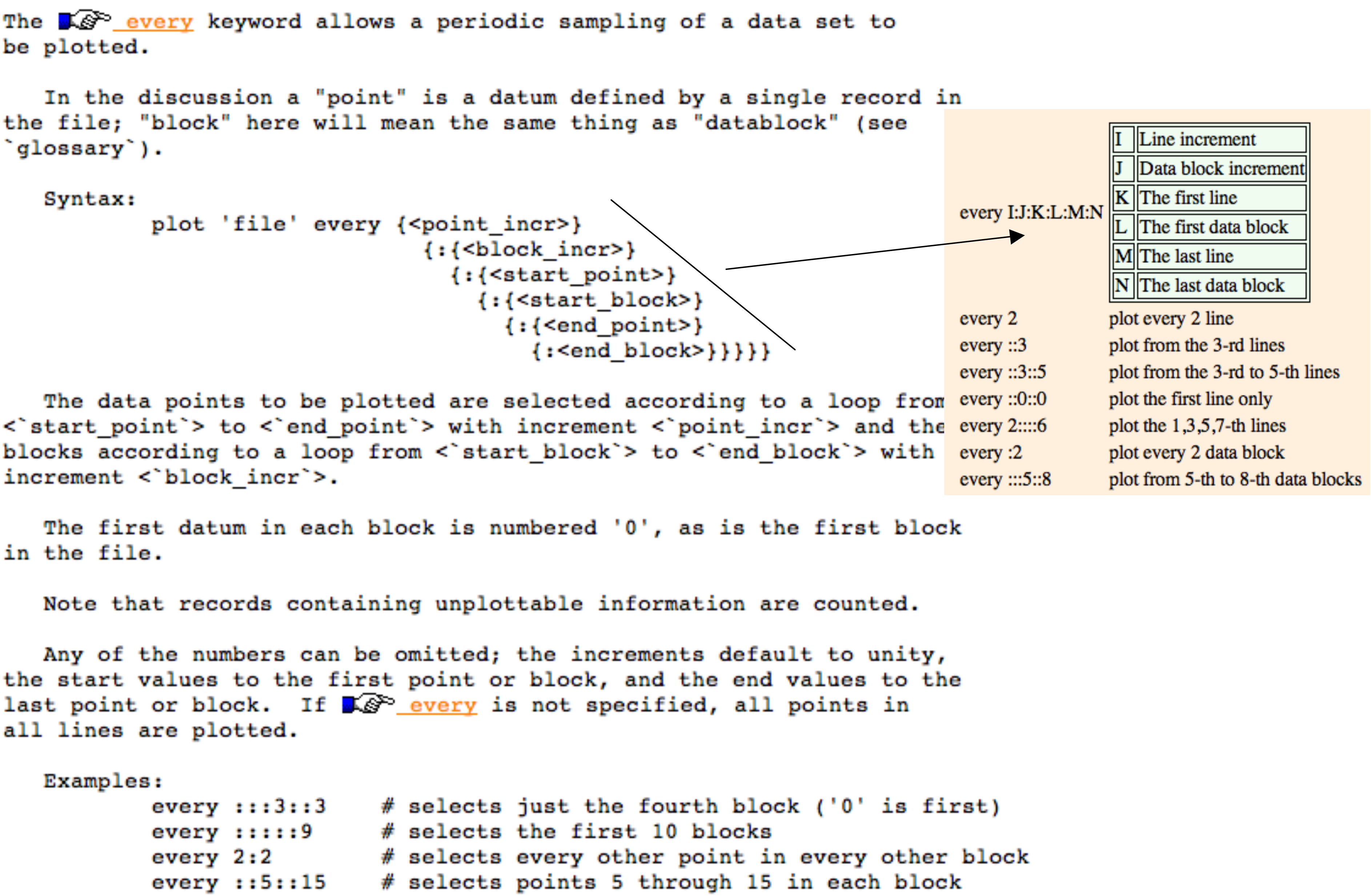
```
# X       Y       Yerror
  1.0     1.2     0.1
  2.0     1.8     0.1
  3.0     1.6     0.1


  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
```

2 blank lines



```
plot "test.dat" index 0 using 1:2:3 with yerrorbars,\
     "test.dat" index 1 using 1:2:3 with yerrorbars,\
     "test.dat" index 2 using 1:2:3 with yerrorbars
```

Reading parts of a DATA file



The every keyword allows a periodic sampling of a data set to be plotted.

In the discussion a "point" is a datum defined by a single record in the file; "block" here will mean the same thing as "datablock" (see `glossary`).

Syntax:

```
      plot 'file' every {<point_incr>}
                          {:{<block_incr>}
                            {:{<start_point>}
                              {:{<start_block>}
                                {:{<end_point>}
                                  {:<end_block>}}}}}
```

The data points to be plotted are selected according to a loop from <`start_point`> to <`end_point`> with increment <`point_incr`> and the blocks according to a loop from <`start_block`> to <`end_block`> with increment <`block_incr`>.

The first datum in each block is numbered '0', as is the first block in the file.

Note that records containing unplottable information are counted.

Any of the numbers can be omitted; the increments default to unity, the start values to the first point or block, and the end values to the last point or block. If every is not specified, all points in all lines are plotted.

Examples:

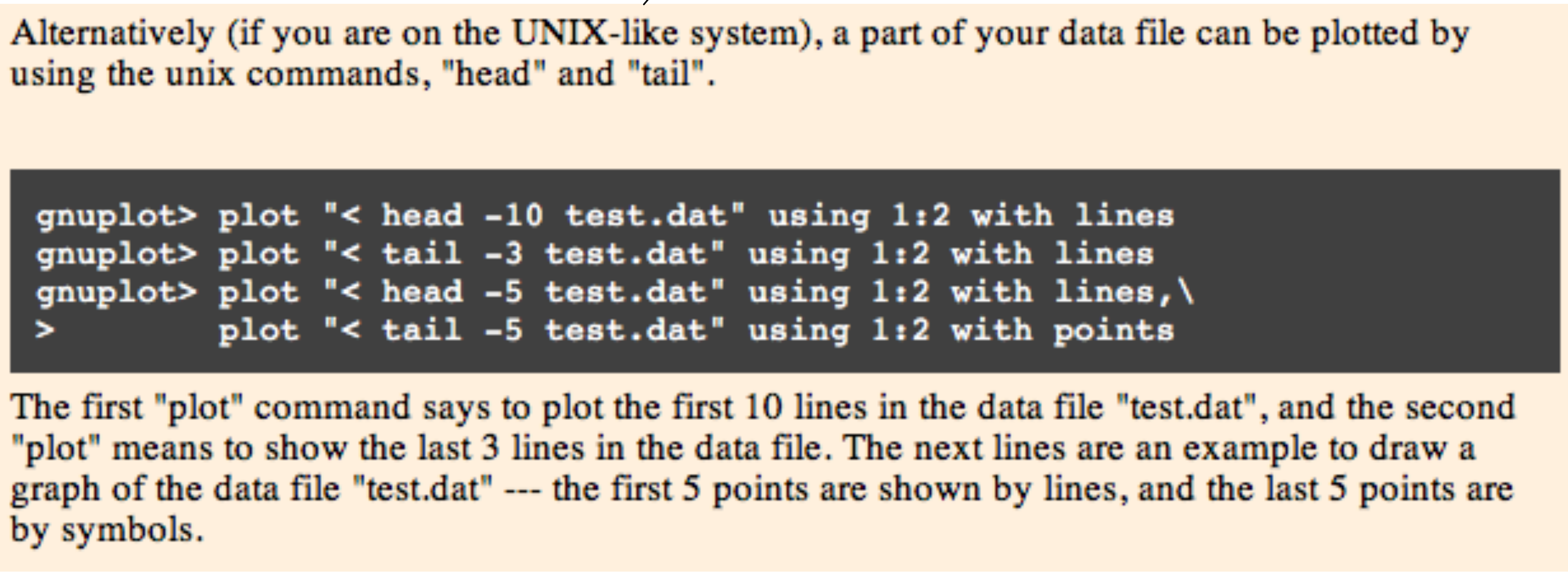
```
      every :::3::3    # selects just the fourth block ('0' is first)
      every :::::9     # selects the first 10 blocks
      every 2:2        # selects every other point in every other block
      every ::5::15    # selects points 5 through 15 in each block
```

every I:J:K:L:M:N

I	Line increment
J	Data block increment
K	The first line
L	The first data block
M	The last line
N	The last data block

every 2	plot every 2 line
every ::3	plot from the 3-rd lines
every ::3::5	plot from the 3-rd to 5-th lines
every ::0::0	plot the first line only
every 2::::6	plot the 1,3,5,7-th lines
every :2	plot every 2 data block
every :::5::8	plot from 5-th to 8-th data blocks

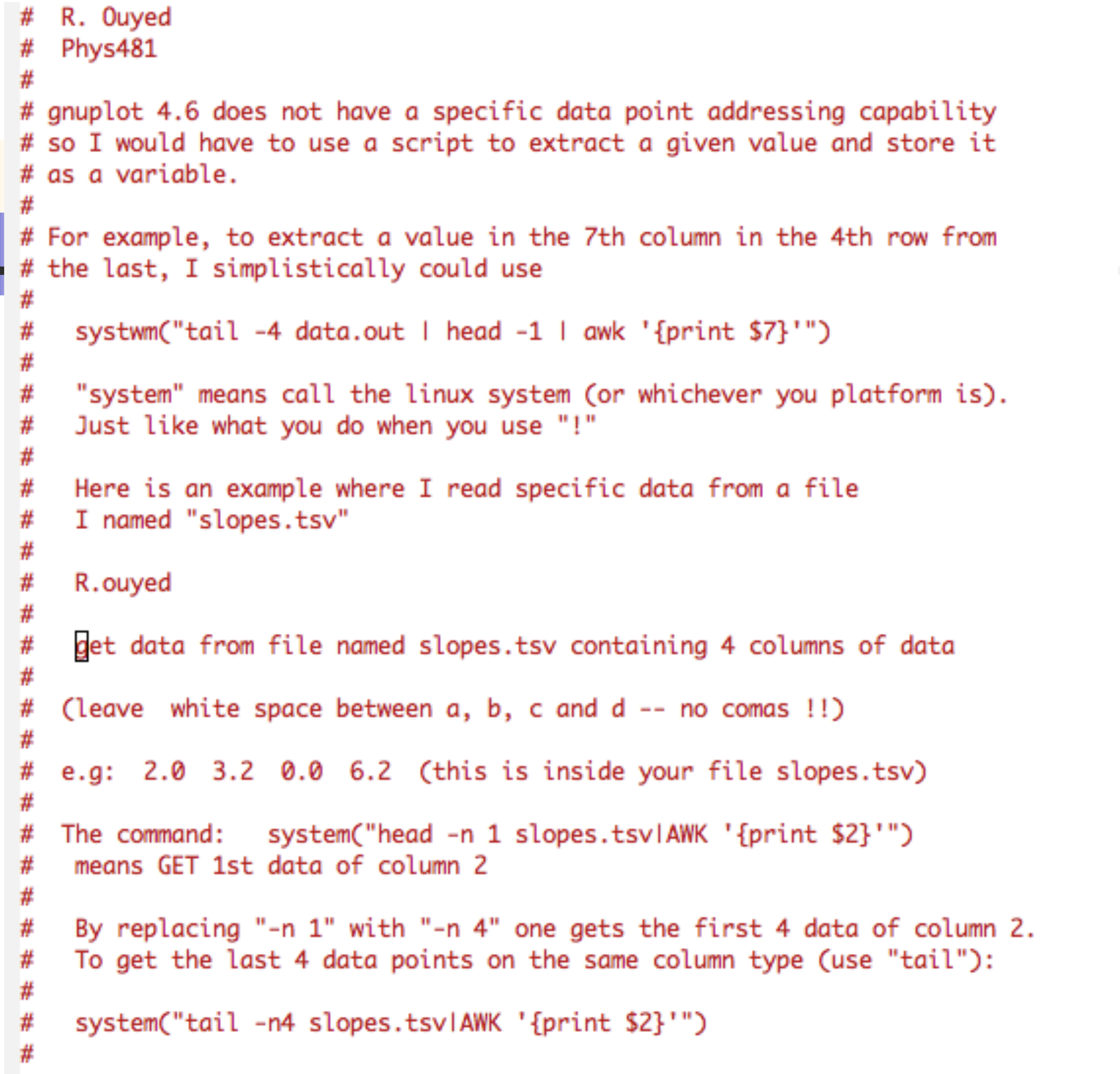
e.g. Linux



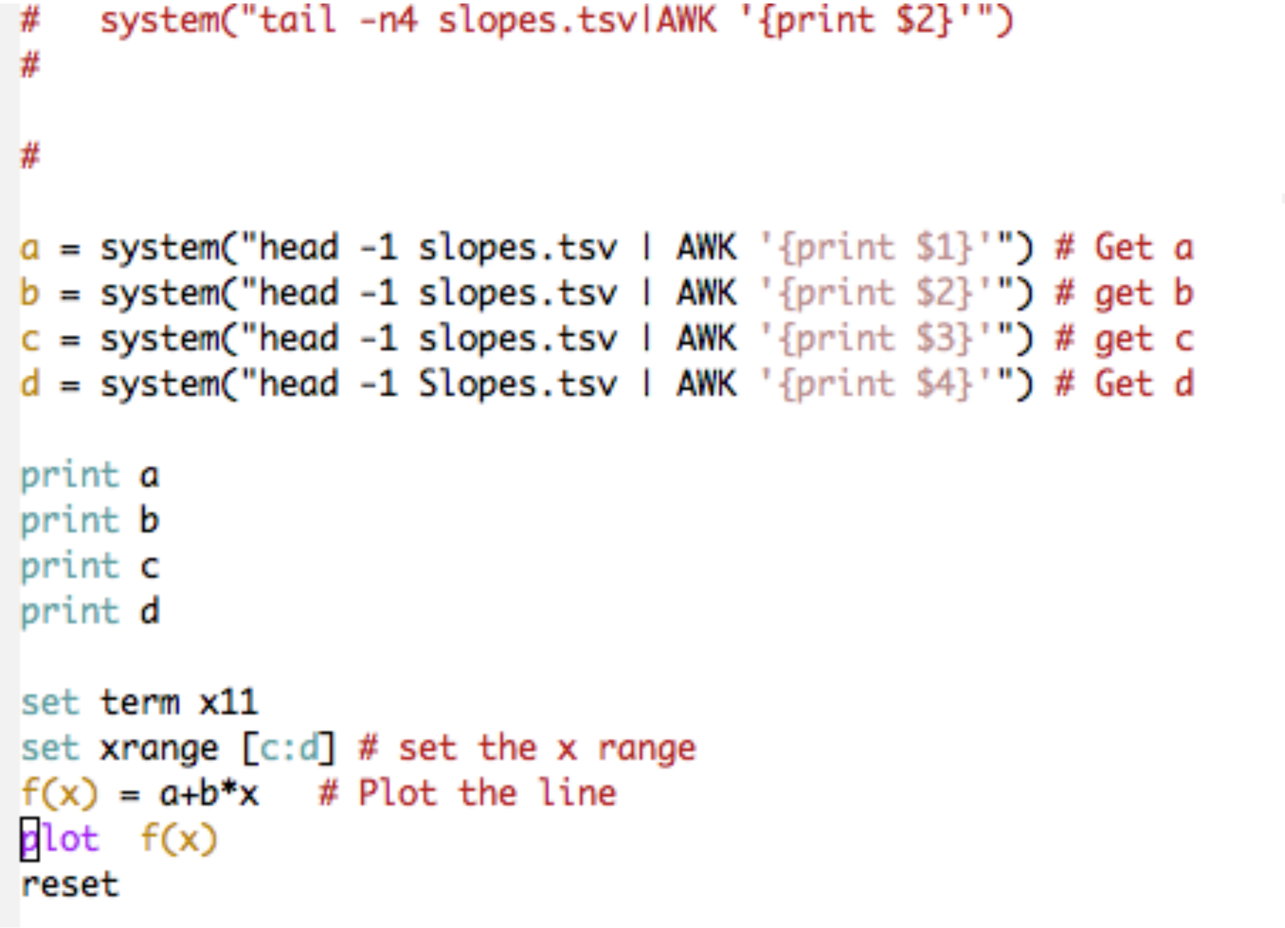
Alternatively (if you are on the UNIX-like system), a part of your data file can be plotted by using the unix commands, "head" and "tail".

```
gnuplot> plot "< head -10 test.dat" using 1:2 with lines
gnuplot> plot "< tail -3 test.dat" using 1:2 with lines
gnuplot> plot "< head -5 test.dat" using 1:2 with lines,\
>        plot "< tail -5 test.dat" using 1:2 with points
```

The first "plot" command says to plot the first 10 lines in the data file "test.dat", and the second "plot" means to show the last 3 lines in the data file. The next lines are an example to draw a graph of the data file "test.dat" --- the first 5 points are shown by lines, and the last 5 points are by symbols.



```
#  R. Ouyed
#  Phys481
#
# gnuplot 4.6 does not have a specific data point addressing capability
# so I would have to use a script to extract a given value and store it
# as a variable.
#
# For example, to extract a value in the 7th column in the 4th row from
# the last, I simplistically could use
#
#   systwm("tail -4 data.out | head -1 | awk '{print $7}'")
#
#   "system" means call the linux system (or whichever you platform is).
#   Just like what you do when you use "!"
#
#   Here is an example where I read specific data from a file
#   I named "slopes.tsv"
#
#   R.ouyed
#
#   get data from file named slopes.tsv containing 4 columns of data
#
#  (leave  white space between a, b, c and d -- no comas !!)
#
#  e.g:  2.0  3.2  0.0  6.2  (this is inside your file slopes.tsv)
#
#  The command:   system("head -n 1 slopes.tsv|AWK '{print $2}'")
#   means GET 1st data of column 2
#
#   By replacing "-n 1" with "-n 4" one gets the first 4 data of column 2.
#   To get the last 4 data points on the same column type (use "tail"):
#
#   system("tail -n4 slopes.tsv|AWK '{print $2}'")
#
```



```
#   system("tail -n4 slopes.tsv|AWK '{print $2}'")
#


#


a = system("head -1 slopes.tsv | AWK '{print $1}'") # Get a
b = system("head -1 slopes.tsv | AWK '{print $2}'") # get b
c = system("head -1 slopes.tsv | AWK '{print $3}'") # get c
d = system("head -1 Slopes.tsv | AWK '{print $4}'") # Get d

print a
print b
print c
print d

set term x11
set xrange [c:d] # set the x range
f(x) = a+b*x    # Plot the line
plot  f(x)
reset
```

IFs and … LOOPS

Do loops and animation in Gnuplot:

The easy way !

```
 Reset
set term gif animate
set output "animate.gif”
n=24 #n frames
dt=2*pi/n
set xrange [0:4*pi]
set yrange [-1:1]
```

```
do for [i=0:n] {
plot sin(x+i*dt)/(1. + i/12.) w l lw 1.5 title sprintf("t=%i",i)
}
```

Multiple commands separated by semi-column (;)
Inside the curly brackets !!

Do loops and animation in Gnuplot:

The hard way !

Gnuplot loops:

If, Reread & Every

The reread command causes the current `gnuplot` command file, as specified by a `load` command or on the command line, to be reset to its starting point before further commands are read from it.

This essentially implements an endless loop of the commands from the beginning of the command file to the reread command. (But this is not necessarily a disaster-- **reread** can be very useful when used in conjunction with **if** AND **every** for).

The **reread** command has no effect if input from standard input.

Suppose the file "looper.gp" contains the commands

```
a=a+1
plot sin(x*a)
pause -1
if(a<5) reread
```

and from within `gnuplot` you submit the commands

```
a=0
load 'looper.gp'
```

The result will be four plots (separated by the pause message).

Looping over name of output file (Text Concatenation)

(the looping variable I=1,N)

```
text1= “pmotion”
text2= I
text3 = “.gif”
set output text1.I.text3
```

As you loop over I this generate files named:

pmotion1.gif, pmotion2.gif, …, pmotionN.gif

Looping over name of output file (Text Concatenation)

What happens if instead you use:
Text2 = “I” ?

Example:

```
I=I+1
text1  = “phys481”
text2  = I
text3 = “.gif”
set output text1.I.text3
plot sin(x*I)
pause -1
if(I<5) reread
```

Looping over name of output file (Text Concatenation)

Anoter Example:

```
# Gnuplot can be used to produce animations in gif format.
# You only need to   choose the appropriate terminal
#
#   set terminal gif animate delay 4
#
#  Here delay 4 means that there will be 0.01x4 seconds
# between frames in your
#  animation (1 / 0.04 = 25 frames per second).
#
#  set output "TrigAnimation.gif"
#
#  Gnuplot is not very good at optimization,
# the resulting gifs prove to be quite large.
#
#  Instead you should do something like this:
```

```
set terminal gif
print "theta is = ", theta
myfile = "output".theta.".gif”
set output myfile
set view 60, theta
splot exp(-x*x)*erf(y)
theta = theta + 100
if(theta<360) reread
reset
```

Suppose the file ”mydata" contains six columns of numbers with a total yrange from 0 to 10; the first is x and the next are five different functions of x. Suppose also that the file "plotter" contains the commands

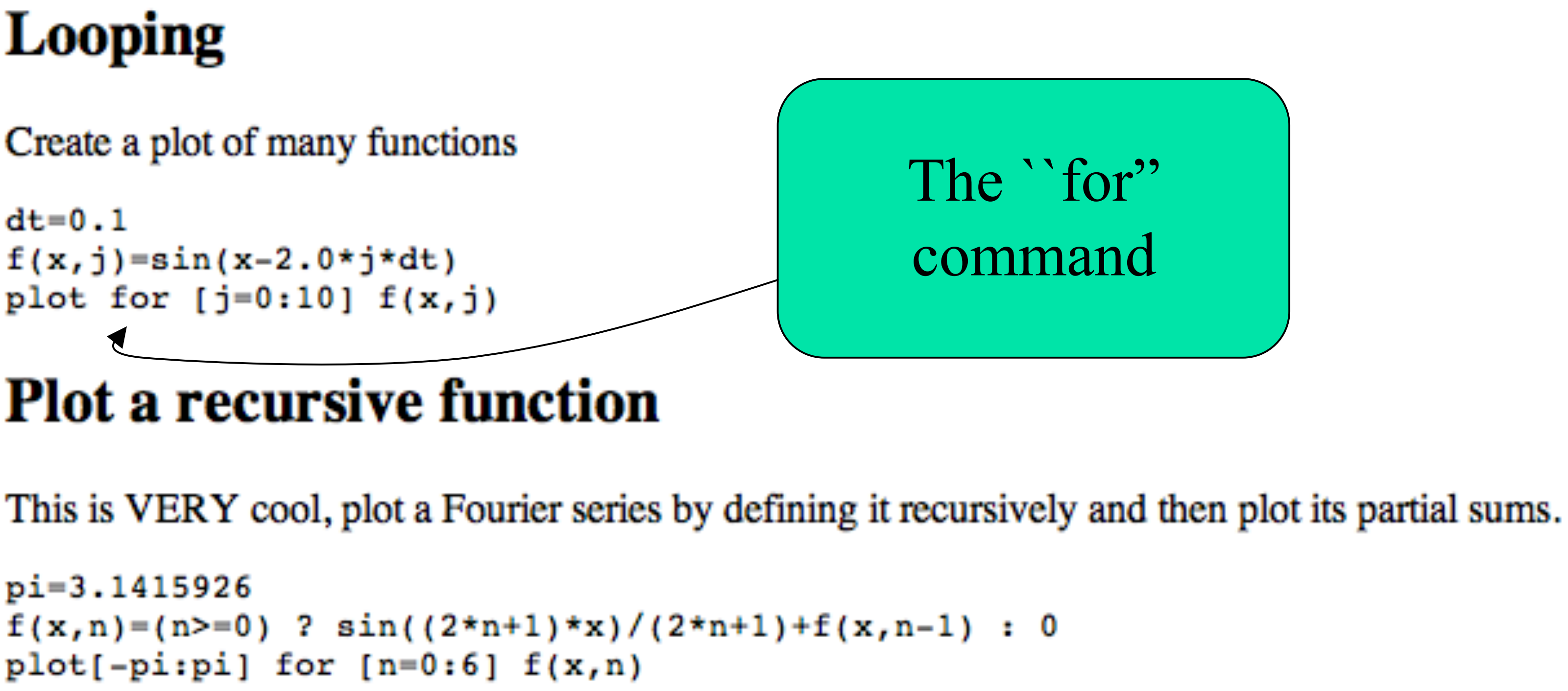
```
c_p = c_p+1
plot "$0" using 1:c_p with lines linetype c_p
if(c_p <  n_p) reread
```

and from within `gnuplot` you submit the commands

```
n_p=6
c_p=1
unset key
set yrange [0:10]
set multiplot
call 'plotter' ’mydata’
unset multiplot
```

The result is a single graph consisting of five plots. The yrange must be set explicitly to guarantee that the five separate graphs (drawn on top of each other in multiplot mode) will have exactly the same axes. The linetype must be specified; otherwise all the plots would be drawn with the same type.

More ``looping tricks”



Looping

Create a plot of many functions

```
dt=0.1
f(x,j)=sin(x-2.0*j*dt)
plot for [j=0:10] f(x,j)
```

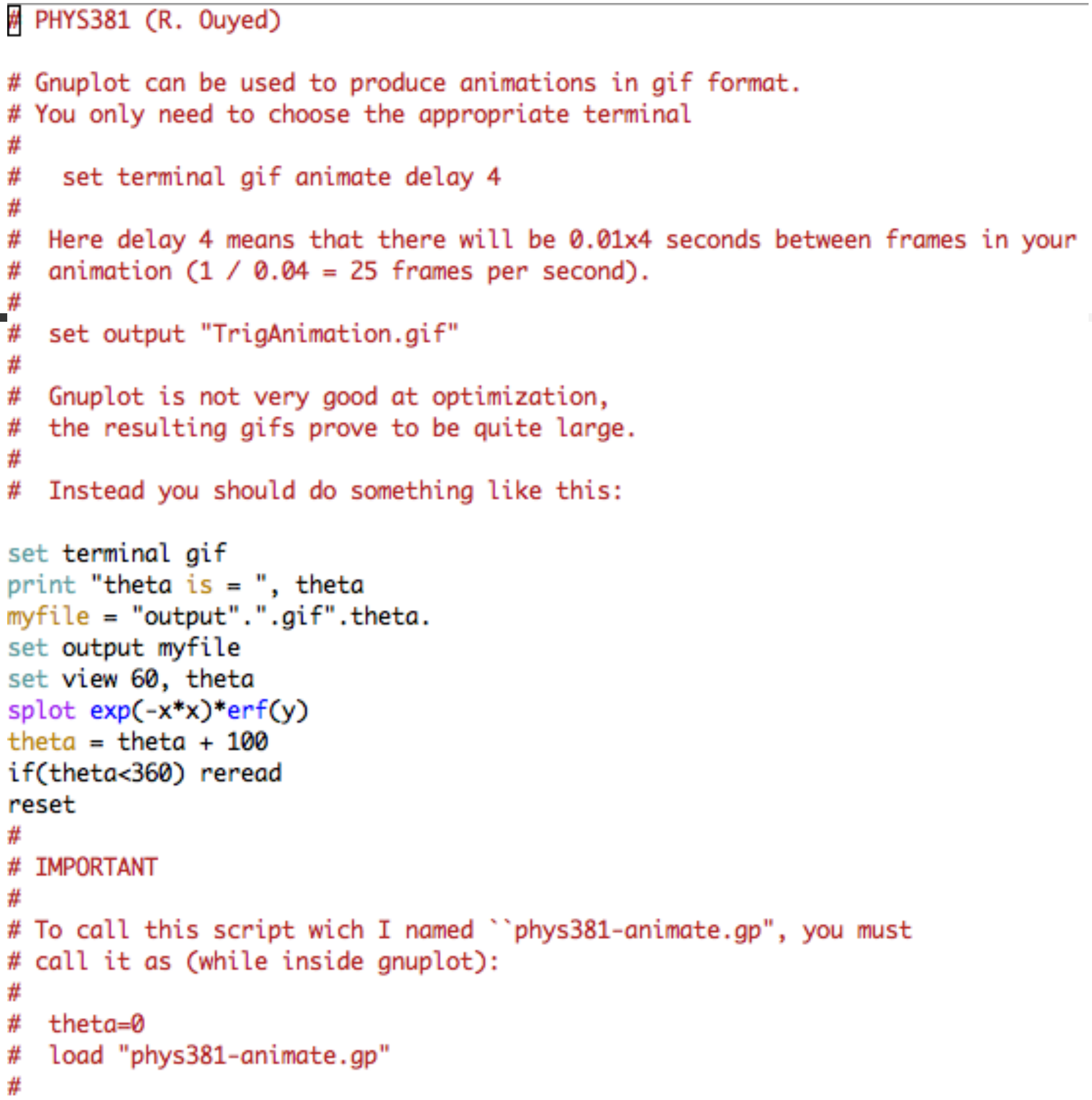
The ``for” command

Plot a recursive function

This is VERY cool, plot a Fourier series by defining it recursively and then plot its partial sums.

```
pi=3.1415926
f(x,n)=(n>=0) ? sin((2*n+1)*x)/(2*n+1)+f(x,n-1) : 0
plot[-pi:pi] for [n=0:6] f(x,n)
```

ANIMATIONS
with
Gnuplot



```
# PHYS381 (R. Ouyed)

# Gnuplot can be used to produce animations in gif format.
# You only need to choose the appropriate terminal
#
#   set terminal gif animate delay 4
#
#  Here delay 4 means that there will be 0.01x4 seconds between frames in your
#  animation (1 / 0.04 = 25 frames per second).
#
#  set output "TrigAnimation.gif"
#
#  Gnuplot is not very good at optimization,
#  the resulting gifs prove to be quite large.
#
#  Instead you should do something like this:

set terminal gif
print "theta is = ", theta
myfile = "output"."gif".theta.
set output myfile
set view 60, theta
splot exp(-x*x)*erf(y)
theta = theta + 100
if(theta<360) reread
reset
#
# IMPORTANT
#
# To call this script wich I named ``phys381-animate.gp", you must
# call it as (while inside gnuplot):
#
#  theta=0
#  load "phys381-animate.gp"
#
```

Using ``convert” to make animations

The following command may be used to produce an animated gif from a series of images. The user specifies a wild card e.g. * to specify a series of input images. The *delay option* specifies the delay between each frame in hundredths of a second and the loop option with loop 0 makes the animated gif *loop around* continuously

```
convert -delay 20 -loop 0 infile*.gif animateinfile.gif
```

A useful tip to remember when converting a series of images is
 to write the index of on image using 2 or 3 digits, i.e.
 XXX or XX 01 02.... 09, 10 or 001, 002,.... 009,..... 010, 099, 100 etc...

This approach ensures that the operating system lists the image files in the correct numerical order.

Other VERY useful tricks/commands ….

To do arithmetic on columns ($n applies to column n):

```
plot 'table.dat' using ($3/$1):($2*134.44)
```

To plot lines 4-20 from a file with filled square points and double-weight solid lines:

```
plot "< awk 'NR>3 && NR<20 {print $1,$2}' axis.dat" w lp
lt 1 lw 2 pt 5
```

'w lp lt 1 lw 2 pt 5' means 'with linespoints linetype 1 lineweight 2 pointtype 5'

To place a single 'X' point at x=4, y=5 (useful for keys):

```
plot "< echo 4 5" w p pt 2
```

Thank you!