UCS

# Introduction to Gnuplot

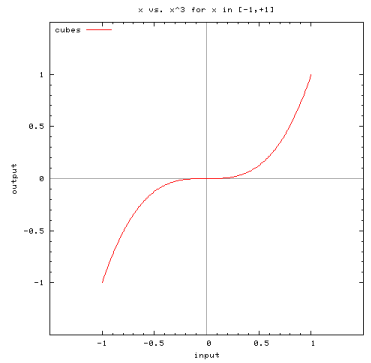
Bob Dowling
University Computing Service



UCS

# Course aims

- Simple graphs
- 2D
- Plotting data
- Scripted process
- No manual work






The aims of the course are to provide you with sufficient familiarity with Gnuplot to plot simple 2D graphs of data generated by other programs. By the end of this course you will be able to generate image files of graphs from data files in a purely scripted environment with no manual interaction required. This is the appropriate model for activity towards the end of a series of tasks in a shell script, for example.

UCS

## What the course *won't* cover

**Gnuplot can**

- 3D plots
- Plotting functions
- Polar graphs
- Histograms

**Gnuplot can't**

- Manual artistry



There are some topics that this course will not be covering. Some are because this is an *introductory* course and others because Gnuplot doesn't do them.

UCS

## What the course *will* cover

1. Introduction
2. Part one
3. Break
4. Part two
5. Questions

- Course contents
- Command line
- Viewing tool
  - eog
  - “Eye of Gnome”



We will start with this introduction. We will make sure that you can get at a Linux command line and run the tool we will be using to view the produced graphs.

UCS

## What the course *will* cover

1. Introduction
2. Part one
3. Break
4. Part two
5. Questions

- Driving Gnuplot
- Basic settings
- Size
- Ranges of values
- Tick marks



The course proper will have two halves, split by a break.
In the first half we will cover the basics of Gnuplot and get a basic graph drawn.

UCS

## What the course *will* cover

1. Introduction
2. Part one
3. Break
4. Part two
5. Questions

- Multiple graphs
- Colours
- Labels
- Frills



The second half will cover drawing multiple graphs in one drawing and setting various other attributes of the image such as colour, labels and a few extra frills.

UCS

## What the course *will* cover

1. Introduction
2. Part one
3. Break
4. Part two
5. Questions

7

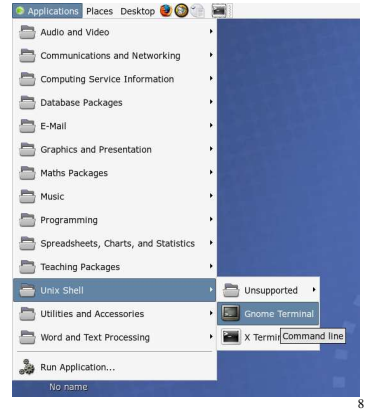
We will end with questions, obviously.

Please note that there is a set of reference material at the end of the notes that go further than the course. This is the introduction. Once you have completed it the notes should be relatively easy to follow. They may answer many of your questions.

UCS

## Terminal

- Command line
- Gnome terminal
- Applications menu




8

So let's start by launching a shell window from the **Applications** menu.

Applications → Unix shell → Gnome Terminal

UCS

## Set up some files

- “Playground”
- Input files
- Gnuplot files
- Output files

> **/ux/Lessons/Gnuplot/setup**

Directory /home/*y500*/gnuplot created.

> **cd ~/gnuplot**



We will want a few files for this course. The author has prepared a “playground” directory with the material required. Make sure you have no file or directory called “`gnuplot`” in your home directory and then run the command shown. It will unpack the directory for you.
Once it is unpacked move into the `gnuplot` directory.

```
> /ux/Lessons/Gnuplot/setup
Directory /home/y500/gnuplot created.
> cd ~/gnuplot
> ls -l
total 82
-rw-r--r--  1 … cubic.dat
-rw-r--r--  1 … cubic.gplt
-rw-r--r--  1 … example01.png
-rw-r--r--  1 … lissajou1.dat
-rw-r--r--  1 … lissajou2.dat
-rw-r--r--  1 … powers.dat
-rw-r--r--  1 … powers.gplt
```

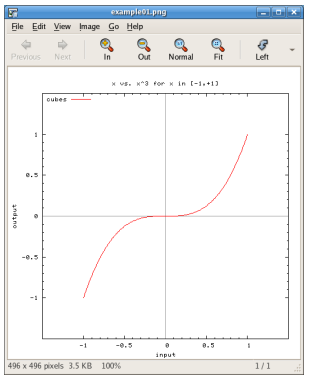
UCS

## Test you can display graphics

> **eog example01.png**

Select image window

Close or Control-Q






We want to make sure that we can see the images we will be creating. In the `~/gnuplot` directory you should find a file called `example01.png`. This is a graphics file in Portable Network Graphics format (PNG). The quick and easy tool we will be using to view the image is `eog`, which stands for “eye of gnome”. Don't worry about the in jokes there.

UCS

## Interactive use of Gnuplot

- Run interactively
- Direct commands
- Built-in help
  - Not very helpful
- X11 graphics

```
> gnuplot
G N U P L O T
Version 4.0 …
Terminal type set to 'x11'
gnuplot> plot "cubic.dat"
…
gnuplot> quit
>
```

11

1. Make sure you are in the the `~/gnuplot` directory. This example won't work otherwise.
2. Launch `gnuplot` without any arguments. It starts up interactively and presents you with a "`gnuplot>`" prompt.
3. Type the instruction: `plot "cubic.dat"` as shown on the slide. The quotes around the file name are necessary and must be double quotes.
4. Hit the Return key at the end of the command.
5. Your Gnuplot session should give you back its prompt and a graphical window should pop up. Pause to admire the graph.
6. Type `quit` at the Gnuplot prompt as shown in the slide.
7. The graph window should disappear and the shell should take control of the terminal window again.

UCS

## Batch use of Gnuplot

- File of commands
- `cubic.gplt`
- Don't need a "quit"
- Just need end of file
- "Flash" of graph

```
> more cubic.gplt
plot "cubic.dat"

> gnuplot cubic.dat
>
```

12

That's all very well but doesn't move us towards our goal of an automated system. We don't want interactivity; we want a batch system where our commands are in a file and are run from a single command line which might be part of a shell script.

In the `~/gnuplot` directory there is a file `cubic.gplt` which contains the same `plot` instruction as we typed manually. It does not have the `quit` instruction though, as reaching the end of file is equivalent.

If we run the command shown at the shell, **`gnuplot cubic.gplt`**, while looking closely at the screen you will see a quick flash of the graph appearing in response to the `plot` command and then disappearing immediately afterwards in response to the end of file.

UCS

## Change output file format

- Want PNG
  - **P**ortable **N**etwork **G**raphics
- Gnuplot “terminal”: set terminal png
- Want a file: set output "cubic.png"
- File names in quotes
- *Before* the plot

```
> more cubic.gplt
set terminal png
set output "cubic.png"
plot "cubic.dat"

> gnuplot cubic.gplt
> eog cubic.png
```



We need to fix this. We will be editing the file `cubic.dat` during this course so power up your favourite editor and start editing the file. We need to add two lines to the file before the `plot` command:

```
set terminal png
set output "cubic.png"
```

Then run the `gnuplot` command again, exactly as you did before. This time you should get no flash of graphics, but you should see a new file created, `cubic.png`. If you inspect this file with `eog` then you will see that it contains the same graph as we saw earlier, but now in a permanent file rather than in a transient window.

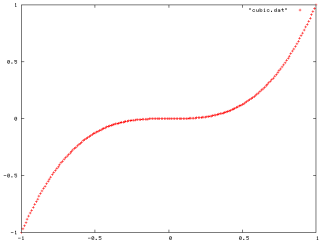
If you are unfamiliar with Unix editors, I would recommend `gedit`, a very simple text editor which will suffice for this course:

```
> gedit cubic.gplt &
```

The command is `gedit`, the file it will be editing is `cubic.gplt` and the ampersand runs the command in the background. We just need to have the editor save the file when we are done with each stage of editing it; we don't need to keep stopping and starting the editor.

UCS

## A look at the output



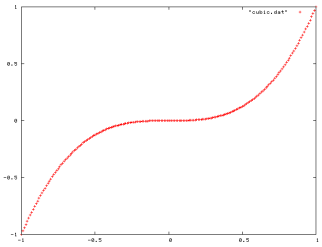

- 640×480 pixels
- Series of crosses
- Graph range = data range
- “Ticks” every 0.5
- No zero axes
- No labels
- Key uses file name
- Red



So let's look at this output. It has a number of properties and for the sake of a course that teaches Gnuplot we will pretend that we like none of them.

UCS

# Problems with the shape




Problems:

- Image file 640×480
- Graph isn't square

Want to set:

- Image dimensions
- Graph aspect ratio



We don't like this size or shape of the graph. We want the image file itself to be 512 by 512 (square) and the graph embedded within it to be the same.

UCS

# Gnuplot commands

**Image dimensions:**

set terminal png picsize *X Y* Image size in pixels



We will start with the image dimensions. Setting the dimensions of the PNG graph is done as part of the terminal definition at the same time as specifying its format. This defines the size of the image as a whole; not just the graph drawn within the image.

UCS

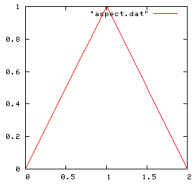
# Gnuplot commands

**Graph aspect ratio:**

set size ratio *r*

Graph's aspect ratio

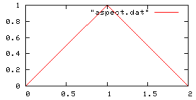
e.g. set size ratio +1:




set size ratio -*r*

Units' aspect ratio

e.g. set size ratio -1:






Now we have specified the size of the image, the canvas we will draw in, we can specify the aspect ratio of the graph we will draw.

Gnuplot has two incompatible ways to specify aspect ratios. It can either specify the aspect ratio of the plotted graph itself, or it can specify the aspect ratio of the units in the graph. To see what this means consider a graph whose *x* values vary from 0 to 2 and whose *y* values vary from 0 to 1.

If we specify the aspect ratio of the graph to be 1 then the graph appears square as on the left. If we specify the aspect ratio of the units to be -1 then the graph turns out twice as wide as it is long because *x* covers twice the range that *y* does.

To specify the aspect ratio of the graph, use a positive number of the aspect ratio. To specify the aspect ratio of the units take the ratio and specify it as a negative number.

Aspect ratios greater than 1 lead to “portrait” format graphs (taller than they are wide) and aspect ratios less than 1 lead to “landscape” format graphs (wider than they are tall).

UCS

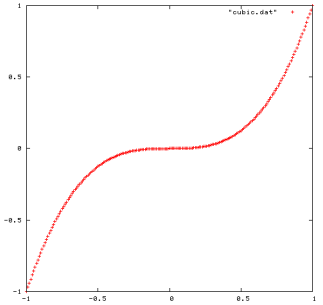
# Next version of graph

set terminal png
**picsize 512 512**

set output "cubic.png"

**set size ratio -1.0**

plot "cubic.dat"

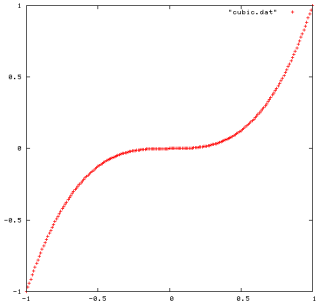





In my running example I will insist on an aspect ratio of 1 between the *units*, so I set a parameter of -1 in the Gnuplot file.

UCS

## Problems with the curve




Problem:

- Set of crosses

Want to have:

- Set of line segments



Next we decide we want to replace the set of points with a curve made of line segments.

UCS

## Gnuplot commands

**Curve made of points:**

set style data points

- "points" may be crosses or other marks

**Curve made of line segments:**

set style data lines

**Curve made of true dots:**

set style data dots



By default Gnuplot uses what it calls "points" to plot data. These are a series of marks, crosses in this case, to mark the data points. There are two alternatives to this.

To join the data points with line segments, use "`set style data lines`". This is suitable for primitive interpolation on a set of sample points from a curve. This is our case where our data points are uniformly sampled from the curve $y=x^3$.

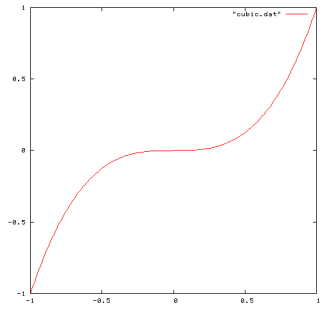
For scatter graph data you are better off using true points with the command "`set style data dots`".

UCS

## Next version of graph

```
set terminal png picsize
  512 512
set output "cubic.png"
set size ratio -1.0
set style data lines

plot "cubic.dat"
```

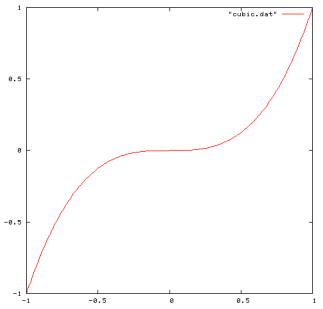





So we add the command "`set style data lines`" to our `cubic.gplt` file, again before the `plot` command, and run Gnuplot to get our new graph.

UCS

## Problems with the range




Problem:

- Graph range = Data range
- [-1.0,+1.0]×[-1.0,+1.0]

Want to have:

- Manual setting
- [-1.5,+1.5]×[-1.5,+1.5]



Now we really are moving into the realm of contrived complaints. Gnuplot's default is to set the range of the graph's axes to be the same as the range of the data. This is fairly natural default and typically what we want. Sometimes, however, we want to put some space around a graph or restrict drawing to some sub-ranges. In essence we want to be able to manually set the *x* and *y* ranges.

UCS

## Gnuplot commands

**Setting range explicitly:**

set xrange [-1.5:1.5]

set yrange [-1.5:1.5]

**Partial specification:**

| | |
|---|---|
| set xrange [*:1.5] | data minimum to 1.5 |
| set yrange [-1.5:*] | -1.5 to data maximum |



To set the *x* range manually we use the syntax "`set xrange [`$x_{min}$`:`$x_{max}$`]`". We can use an asterisk for either end of a range to mean "the corresponding data range". There is a corresponding "`set yrange`" command. Data points that lie outside either range will not be drawn.

UCS

## Next version of graph

…

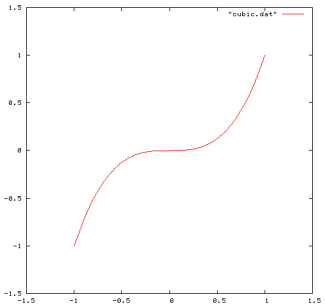
set size ratio -1.0

set style data lines

**set xrange [-1.5:1.5]**

**set yrange [-1.5:1.5]**

plot "cubic.dat"

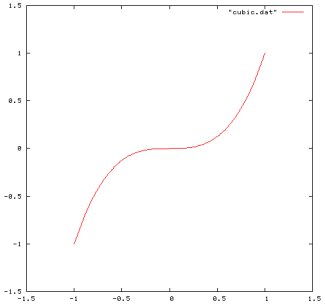





In our example we will set the ranges to be from -1·5 to +1·5 in each direction. As ever, settings must come before the `plot` instruction.

UCS

## Problems with the graph




Problem:

- Ticks every 0.5
- Ticks from -1.5 to 1.5

Want to have:

- Ticks every 0.25
- Ticks from -1.0 to 1.0

25

Now we will turn to the “ticks”. These are the little marks inside the graph's border marking intervals in the corresponding values. Each is labelled with the value it corresponds to.

Now when the author was taught graph drawing at school he was taught to keep the ticks on the outside of the border so as not to confuse the graph. In the colourful presentation we are using here that is less likely than the black and white school days with exercise book and pencil but nonetheless it ought to be an option. So far as the author can see, there is no such option. (Of course if any reader can prove me wrong, corrections will be gratefully received.)

Let us suppose we want to have ticks every 0·25 rather than every 0.5 as the default gives. Let us also suppose we want them to reflect the data ranges rather than our imposed axis ranges.

UCS

## Gnuplot commands

**Setting just the tick interval**

set xtics 0.25

set ytics 0.25

**Setting the interval and range**

set xtics -1.0,0.25,1.0

set ytics -1.0,0.25,1.0

26

To set the frequency and range of the *x* ticks we have the command “`set xtics`” (n.b. the missing “k” at the end). We cannot use the asterisk to mean “the data range” unfortunately; to use the data range in both directions just quote the frequency.

UCS

## Next version of graph

…

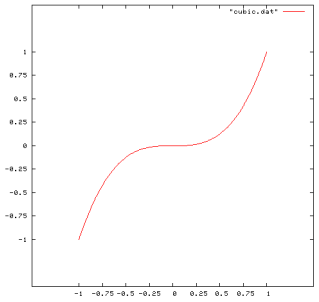
set xrange [-1.5:1.5]

set yrange [-1.5:1.5]

**set xtics -1.0,0.25,1.0**

**set ytics -1.0,0.25,1.0**

plot "cubic.dat"

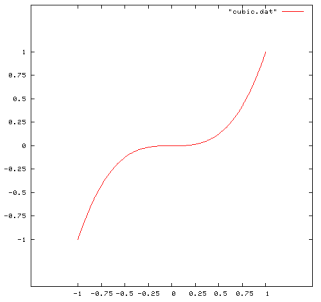





So we apply these new settings to the Gnuplot file and regenerate our graph.

UCS

## Problems with the graph




Problem:

- No sub-ticks

Want to have:

- Sub-ticks every 0.05
- 5 sub-ticks to the tick



Ticks have values printed alongside them outside the border. It is not uncommon to want smaller ticks marking intervals between the ticks, but without values alongside them. These are called “sub-ticks” or “minor ticks”. Let us suppose we want each “tick interval” of length 0·25 to be divided into five sub-intervals of 0·05.

UCS

## Gnuplot commands

**"Minor ticks"**

- Number of minor ticks for each major tick

```
set mxtics 5
set mytics 5
```



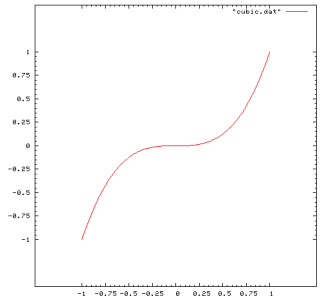
Minor ticks are controlled by "`set mxtics`". Rather than specify the length of their interval (0·05 in our example) we specify how many minor tick sub-intervals there should be per major tick interval. In this case there should be 5.

UCS

## Next version of graph

```
…
set xtics -1.0,0.25,1.0
set ytics -1.0,0.25,1.0
set mxtics 5
set mytics 5

plot "cubic.dat"
```

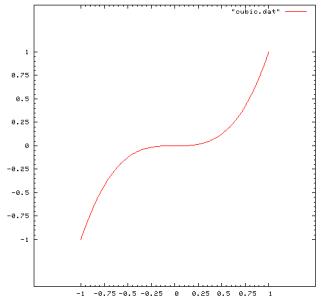





We set this in the Gnuplot file for both the *x* and *y* axes and redraw our graph.

UCS

# Problems with the graph




Problem:

- No axes

Want to have:

- Proper axes
- Running through (0,0)



Our values are being displayed on the borders so as not to clutter the graph itself. However, sometimes axes are desired to indicate the origin in a graph.

UCS

# Gnuplot commands

**Axes through the origin:**

- A “zero axis”
- Defaults to being in grey
- Will consider colours later

set zeroaxis

or

set xzeroaxis

set yzeroaxis



To turn these on Gnuplot has the commands “`set xzeroaxis`” and “`set yzeroaxis`” for each axis in turn. Typically, however, we want both or neither and the command “`set zeroaxis`” turns them both on.

By default they appear in grey. We will be covering how to control the colours used in the second half of this afternoon.

UCS

# Next version of graph

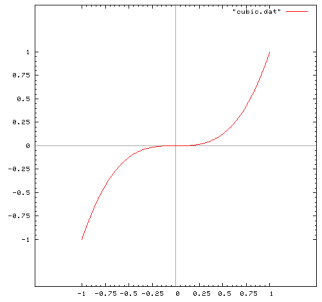
…

set mxtics 5

set mytics 5

**set zeroaxis**

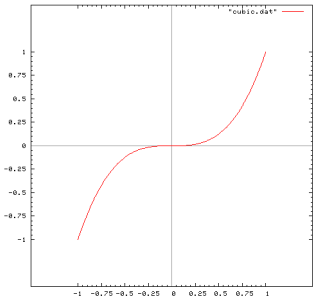
plot "cubic.dat"






UCS

# Problems with the graph




Problem:

- Key in top right
- Key uses file name

Want to have:

- Key in top left
- Manually specified key



Finally for this half of the course, we will address the key or legend in the top right corner of the graph. We may want to remove it or move it elsewhere. We also note that it uses the file name to label the curve. We want to set that manually.

UCS

# Gnuplot commands

**Location of key:**

- Only the corners are available

set key top left

set key bottom right

unset key



We will deal with the location of the key first. In Gnuplot only the four corners are available for the key; it cannot be placed half-way along a side, for example. The default is for it to appear in the top right but the "`set key`" command can take any of the four corners as an argument. If the command "`unset key`" is given then no key is drawn at all.

UCS

# Gnuplot commands

**Text for key:**

- Option on plot
- "Title" of the data
  *not* the whole graph

plot "cubic.dat" title "cubes"

plot "cubic.dat" notitle



The text that appears alongside each line (or point) in the key is set by options on the `plot` command that inputs the data from the file and actually causes the line to be drawn. The `plot` command takes a "`title`" option that allows for the annotation in the key to be set. This text must appear in double quotes even if it is a single word. Alternatively the option "`notitle`" causes it not to appear in the key at all.

UCS

## Next version of graph

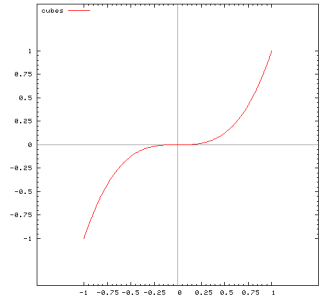
…

set xzeroaxis

set yzeroaxis

**set key top left**

plot "cubic.dat" **title "cubes"**






So we add these into our Gnuplot file and recreate the graph. Our key has moved and carries the label we specified.

UCS

## Half time exercise

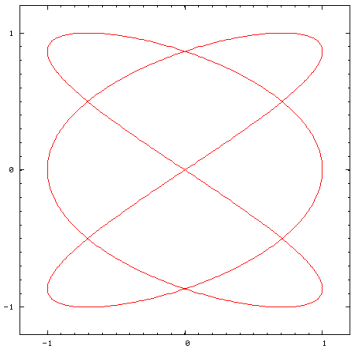
- Fifteen minutes
- Create the graph
- Then have a break

lissajou1.dat +

**lissajou1.gplt**

↓

lissajou1.png





We're going to have a break.

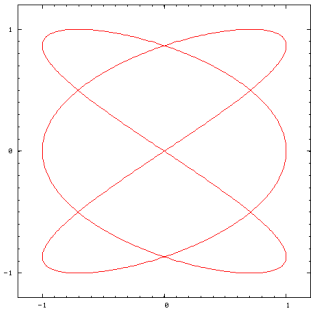
At the start of that break I would like you get try a graph of your own from scratch. In your `~/gnuplot` directory there is a file called `lissajou1.dat` which contains the data for the Lissajou plot shown in the slide. I want you to create the corresponding PNG file.

In case it isn't clear from looking at the graph: the *x* and *y* axes run from -1·2 to +1·2 and the ticks stop at ±1·0.

People who are looking to get “transferable skills” sheets signed will have to show this graph and the one from the exercise at the end of the lecture to the lecturer to get his signature.

UCS

# Welcome back



- lissajou1.gplt
- Any questions?



UCS

# Second half

- Introduction
- Part one
- Break
- Part two
- Questions
- Multiple graphs
- Colours
- Labels
- Titles

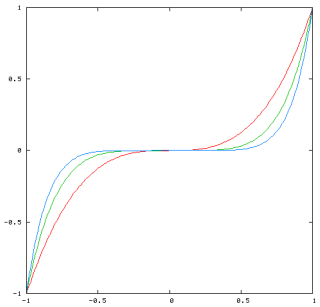


In the second half of the course we are going to cover plotting multiple curves in one graph and address how we control the colours of these curves. This will lead to a discussion of colour control in general.

Then we will talk about labelling our axes and the graph in general.

UCS

## Compound graph



- Single graph
- Several lines

- Single input file
  - Several columns
- Multiple input files
  - Two columns each



So let's suppose we want to plot multiple curves. These may come from a single input file or several input files.

UCS

## Data file

```
# x, x^3, x^5, x^7
-1.000000   -1.000000   -1.000000   -1.000000
-0.990000   -0.970299   -0.950990   -0.932065
…
0.990000    0.970299    0.950990    0.932065
1.000000    1.000000    1.000000    1.000000
```



In your `~/gnuplot` directory you will find a file `powers.dat`. This file contains four columns of numbers, with each line carrying values ($x$, $x^3$, $x^5$, $x^7$) for x running from -1·0 to +1·0. These will give us the data points we want to plot in the second half of this course.

UCS

## Properties of the data file

```
# x, x^3, x^5, x^7
-1.000000   -1.000000   -1.000000   -1.000000
-0.990000   -0.970299   -0.950990   -0.932065
…
```

- "#" introduces a comment line
- Ignored by Gnuplot
- Columns separated by whitespace



There are a couple of things to note about this data file.
The first is that it can carry comments, introduced by a hash character, "#".
The second is that the columns, as with `cubic.dat`, are separated by white space.

UCS

## Gnuplot commands

- Extend the plot command
- Specify the columns to use
- Specify the data files to use
- Comma between curve definitions
- Continue lines with a backslash

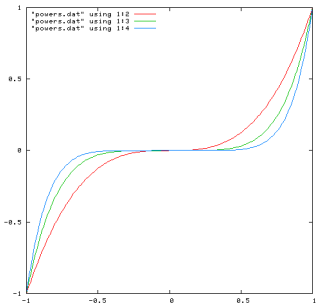
```
plot  "powers.dat" using 1:2, \
      "powers.dat" using 1:3, \
      "powers.dat" using 1:4
```



You will find a bare bones Gnuplot file called `powers.gplt` in your `~/gnuplot` directory.
The only new command within it is the extension to the `plot` command telling it to plot multiple curves. The plot command takes a series of instructions, one for each curve separated by commas. Each instruction is the name of the input file in double quotes followed by an option to identify the pair of columns to use.
This tends to lead to long lines so Gnuplot has a mechanism for breaking lines in the file without interfering with command syntax. At any point in a Gnuplot file where white space would have appeared you can type a backslash immediately followed by a line break and Gnuplot will ignore that line break, simply gluing the files together. While it is being illustrated here in the `plot` command, it can be used in any Gnuplot command.
We will generate the PNG file from this Gnuplot command file to get us going.

UCS

# Problems with the graph




Problem:

- Key is very ugly

Want to have:

- Our line names



Once again we will use the "problem/solution" model to introduce new features of Gnuplot.

The key is spectacularly ugly by default as it not only quotes the file name but follows it with the "`using`" options.

UCS

# Gnuplot commands

- Same extension of the plot command
- Once per curve in the graph

```
plot  "powers.dat" using 1:2 title "x^3", \
      "powers.dat" using 1:3 title "x^5", \
      "powers.dat" using 1:4 title "x^7"
```



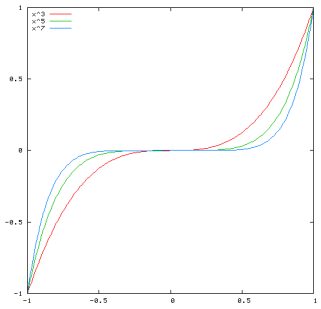
We have already met the "`title`" option on `plot`. Here is how we use it for multiple curves. Quite simply, we add it once per curve instruction.

It's worth noting that the standard builds of Gnuplot don't support "enhanced" text mode where we get real superscripts. PWF Linux's build is such a standard build.

UCS

## Next version of graph

```
plot  \
"powers.dat" using 1:2 \
title "x^3", \
"powers.dat" using 1:3 \
title "x^5", \
"powers.dat" using 1:4 \
title "x^7"
```



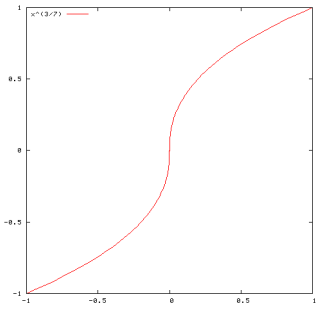


So we add those lines into our `powers.gplt` file and regenerate our graph. We now have a better key.

UCS

## Don't have to use column 1

```
plot "powers.dat" \
using 4:2 \
title "x^(3/7)"
```

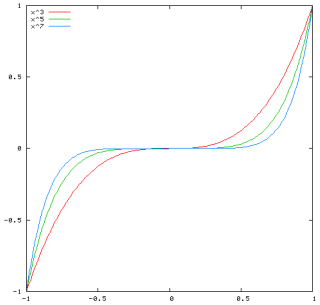




Please note that we don't have to use column one! *Any* pair of columns is valid.

UCS

## Problems with graph



Problems with graph:

- Curve colours
- Red, Green, Blue

Want to have:

- Our curve colours
- Red, Purple, Blue



As promised we now move on to the issue of colour selection. We have left it rather late in the course because it is admittedly fiddly for reasons that make no sense to me at all. I don't know why Gnuplot does it this way, but they do.

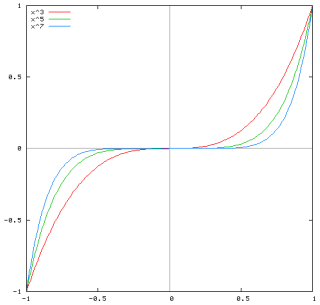
For the time being, let us suppose we just want to change the colour of the middle curve ($y=x^5$) from green to purple.

To do this we will need to examine Gnuplot's use of colour.

UCS

## Colours we have seen so far

- Background (white)
- Borders (black)
- Axes (grey)
- Curve one (red)
- Curve two (green)
- Curve three (blue)





We have seen Gnuplot use a range of colours so far, including white for the background.

UCS

## How Gnuplot uses colours

- Numbered colours for particular purposes
- Maximum of 256 colours

| Purpose | Number | Default colour |
|---|---|---|
| Background | 0 | White |
| Borders | 1 | Black |
| Axes | 2 | Grey |
| First curve | 3 | Red |
| Second curve | 4 | Green |

51

Internally, Gnuplot keeps a list of colours numbered 0 to 255. It uses a particular numbered colour for each particular purpose. So colour 0 is used for the background, colour 1 for the borders and label text, colour 2 for the axes, colour 3 for the first curve, colour 4 for the second and so on up to colour 255 for the 253rd curve.

What we get to do is to set this list of colours.

UCS

## Gnuplot commands

- Extension to set terminal
- List colours at end of command
- Hexadecimal specification: xrrggbb

```
set terminal png picsize 512 512 \
xffffff x000000 x404040 \
xff0000 x800080 x0000ff
```

52

The list of colours can be modified by listing colour values at the end of the "`set terminal`" command. The colour codes given define colours 0, 1, 2, 3, etc. There is no way to specify colour 7, say, without specifying colours 0 to 6.

Unfortunately the colours must be specified in hexadecimal colour notation which is opaque to all but the geekiest. The numbers give the "RGB" ("red green blue") make up of the colours. The leading "`x`" indicates that this number should be read in hexadecimal (base 16). The next two characters specify a number between 0 (x00) and 255 (xff) to indicate how strong the red component of the colour should be. The next two give the green component between 0 (x00) and 255 (xff). The last pair give the blue component between 0 (x00) and 255 (xff).

On most systems there is a file called `rgb.txt` (`/usr/X11R6/lib/X11/rgb.txt` on PWF Linux) which gives numerical values (albeit in decimal rather than hex) for various colours but the names given to the colours make a Dulux colour sheet seem quite tame in comparison. If anyone can guess what colour "burlywood" is without peeking then they're better at this game than the author!

UCS

# A few colours

| | | | | | |
|---|---|---|---|---|---|
| | green | x00ff00 | | black | x000000 |
| | turquoise | x00ffff | | | x404040 |
| | blue | x0000ff | | grey | x808080 |
| | magenta | xff00ff | | | xc0c0c0 |
| | red | xff0000 | | white | xffffff |



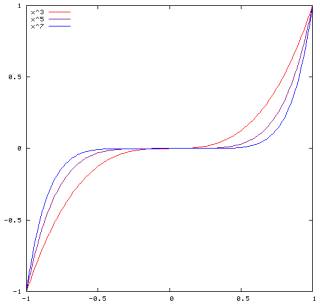
Here are a few colours you might like to consider using. Avoid the “light” colours such as yellow or orange. These do not show up either on paper or screen.

Students should note that colour balance varies wildly between machines. It is common for the colours seen on workstation screen to not exactly match those shown by the overhead data projector or those printed in the notes.

UCS

# Next version of graph

```
set terminal png \
picsize 512 512 \
xffffff x000000 x404040 \
xff0000 xff00ff x0000ff
...
```

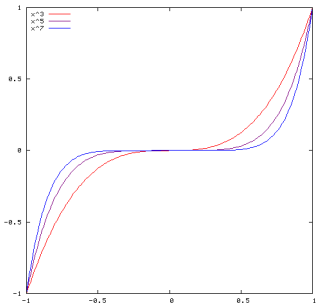




So we modify `powers.gplt` again to add a colour list to the “`set terminal`” command and then re-run Gnuplot on it.

UCS

# Problems with graph




**Problems with graph:**

- No axis labels
- No main title

**Want to have:**

- Axis labels
- Main title



Now we will discuss the axis labels and the main title of the graph. A graph ought to have them, even if it is just to record the experiment reference.

UCS

# Gnuplot commands

**Setting main title:**

```
set title "Powers"
```

- Do not confuse with
  `plot … title "x^3"`



The command to set the title on the graph is simply "`set title`". This should not be confused with the "`title`" option on plot, though. This is an entirely standalone command. We are setting the title for the whole graph rather than just the label for a single line in the graph. Note that the title must be in double quotes under all circumstances.

UCS

# Gnuplot commands

**Setting axis labels:**

set xlabel "x"

set ylabel "power of x"



The commands to set the labels on the *x* and *y* axes are similarly simple. Note that, as ever, the strings must be in double quotes even if they are single words.

UCS

# Next version of graph

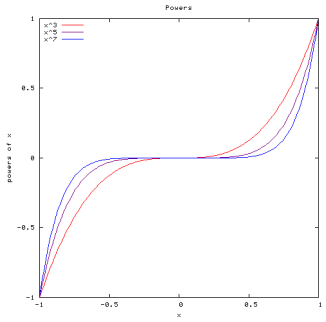
…

**set title "Powers"**

**set xlabel "x"**

**set ylabel "powers of x"**

plot "powers.dat"…

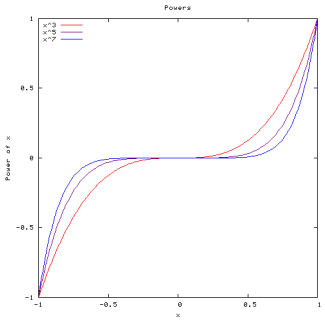





If we add these into our `powers.gplt` file and regenerate the graph we get this.

UCS

# Problems with graph




**Problems with graph:**

- Surround border

**Want to have:**

- Left border
- Bottom border



Let's make one more change. This one is really petty and arbitrary but it serves to illustrate a few of Gnuplot's shortcomings that I have skirted round in this course. I don't want you walking away thinking that everything in the garden is rosy.

What we want to do is to remove the top and right borders, or to draw only the bottom and left ones.

UCS

# Gnuplot commands

**Border edges:** N = 1+2+4+8

set border *N*

| | |
|---|---|
| 1 | bottom |
| 2 | left |
| 4 | top |
| 8 | right |



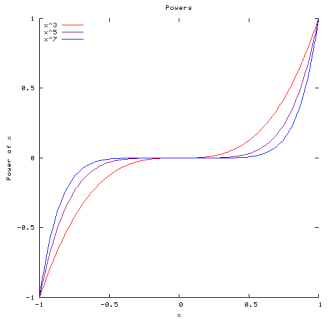
Which bits of the border are drawn or not is controlled by the "`set border`" command. This takes a number as its argument. This number is the sum of up to four numbers depending on which bits of the border you want drawn.

UCS

## Setting border not enough

set border 3

- Set border correctly
- Free-floating ticks!




61

But there's a catch. If we set border to be 3=1+2 (bottom + left) then we do only get those two edges. But the tick marks are still there! Turning off the border does not turn off the ticks.

UCS

## Gnuplot commands

**Ticks:**

Independent of borders!

set xtics nomirror

set xtics 1.0 nomirror

set xtics -1.0,0.5,1.0 nomirror

62

The ticks are independent of the borders. We need to turn them off independently. Here we discover that we can't turn off the ticks for edges independently. We can have ticks or no ticks. We can give ticks on the bottom and left edges and not the top or right edges, but we can't have ticks on the top and right but not the bottom and left.

Ticks appear by default on the left and bottom edges. You can turn them off altogether with "`unset xtics`", "`unset ytics`".

You can also turn off the ticks on the right and top edges by setting the "`nomirror`" option.

UCS

## Final version of graph

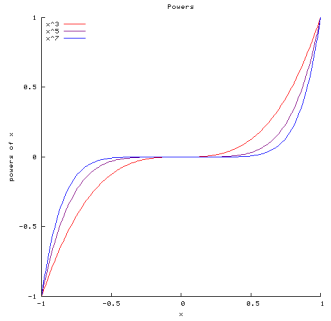
…

**set border 3**

**set xtics nomirror**

**set ytics nomirror**

plot "powers.dat"…






So we set our borders and turn off the mirrored ticks. We're done.

I remind you all that this last example is purely to show you that there are some silly limits built in to Gnuplot. I would recommend leaving the borders and ticks alone.

UCS

## Recap: How to do a graph

1. Define the terminal `set terminal png \`
   `picsize 512 512 …`

   - Output format
   - Image size
   - Colour list



Over the course of this afternoon I've taken you through the stages required to build a Gnuplot command file to create some simple graphs. If you keep your head then their is no reason why this simple step-by-step approach can't generate the Gnuplot files you need every time.

Step one: Define the terminal. The best format for a simple graphics file is PNG format. Specify the picture size in pixels. Set the colour list if you really want to.

UCS

## Recap: How to do a graph

1. Define the terminal — set output "…"
2. Output file
   - File name in quotes
   - Suffix matches format



Next specify the output file. The file name must be in double quotes.

UCS

## Recap: How to do a graph

1. Define the terminal — set size ratio …
2. Output file
3. Aspect ratio
   - +ve: Whole graph
   - -ve: Scale of units



Next set the aspect ratio of the graph in the image.
Positive aspect ratio: ratio of the graph's sides
Negative aspect ratio: ratio of the units' sizes
Ratio > 1: portrait
Ratio < 1: landcape

UCS

## Recap: How to do a graph

1. Define the terminal — set style data lines
2. Output file — set style data points
3. Aspect ratio — set style data dots
4. Points or lines

- Points are the default



Next decide if you want points, lines or dots.

UCS

## Recap: How to do a graph

1. Define the terminal — set key top left
2. Output file — unset key
3. Aspect ratio
4. Points or lines — • Default: top right
5. Place the key



Next, locate the key or legend for the graph.

UCS

## Recap: How to do a graph

1. Define the terminal
2. Output file
3. Aspect ratio
4. Points or lines
5. Place the key
6. Graph title

set title "…"

- Title in quotes



Set a title for the graph to appear at the top of the image. The title must be put in double quotes.

UCS

## Recap: How to do a graph

2. Output file
3. Aspect ratio
4. Points or lines
5. Place the key
6. Graph title
7. Axis labels

set xlabel "…"
set ylabel "…"

- Text in quotes



Then set the labels for the two axes. The strings must be in double quotes.

UCS

## Recap: How to do a graph

3. Aspect ratio — set border 3
4. Points or lines
5. Place the key — • 1+2+4+8
6. Graph title
7. Axis labels
8. Set border



If you really want to mess with the border do so. I wouldn't bother.

UCS

## Recap: How to do a graph

4. Points or lines — set xrange [-1.5:1.5]
5. Place the key — set yrange [-1.5:1.5]
6. Graph title
7. Axis labels
8. Set border
9. Set data range



Set the rage to be plotted if you aren't happy with the axes matching the data range.

UCS

## Recap: How to do a graph

5. Place the key — set xtics -1.0,0.5,1.0
6. Graph title — set ytics -1.0,0.5,1.0
7. Axis labels
8. Set border — set xtics 0.25 nomirror
9. Set data range — unset ytics
10. Set major ticks



Set the ticks. Remember to set the "`nomirror`" option if you are missing the top or right border.

UCS

## Recap: How to do a graph

6. Graph title — set mxtics 5
7. Axis labels — set mytics 5
8. Set border
9. Set data range — set mxtics 5 nomirror
10. Set major ticks — unset mytics
11. Set minor ticks

- Minor ticks per major



Then set the number of minor ticks per major tick.

UCS

## Recap: How to do a graph

7. Axis labels
8. Set border
9. Set data range
10. Set major ticks
11. Set minor ticks
12. Plot data sets

```
plot "…"
  using x:y
  title "…"
```

- File names in quotes
- Column specifiers
- Title in key
- Commas
- Backslashes



Finally, plot the data.

If your data file has more than two columns use the "`using` *m*`:`*n*" option.

Don't forget to set the caption for the line in the key with the "`title`" option or remove it from the key with the "`notitle`" option.

File names and strings must be in double quotes.

UCS

## How to do a graph

1. Define the terminal
2. Output file
3. Aspect ratio
4. Points or lines
5. Place the key
6. Graph title
7. Axis labels
8. Set border
9. Set data range
10. Set major ticks
11. Set minor ticks
12. Plot data sets



And that's it: the twelve-step scheme to defining a graph automatically.

UCS

# Final exercise

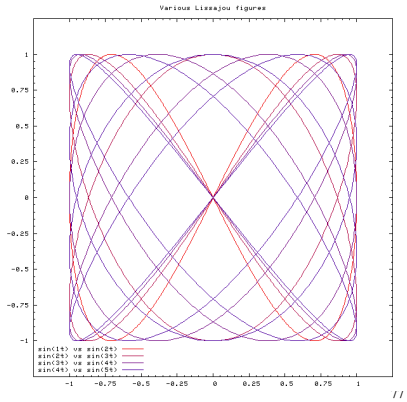
- Create graph
- Details in notes

lissajou2.dat +

**lissajou2.gplt**

↓

lissajou2.png




And a second exercise for you to do.
The data file `lissajou2.dat` has 5 columns corresponding to sin(*t*), sin(*2t*), sin(*3t*), sin(*4t*), sin(*5t*).
Plot the curves shown:

| *x* | *y* | colour |
|---|---|---|
| sin(*t*) | sin(*2t*) | xf00000 |
| sin(*2t*) | sint(*3t*) | xb00000 |
| sin(*3t*) | sin(*4t*) | x700000 |
| sin(*4t*) | sin(*5t*) | x500000 |

The x and y axes range from -1·25 to +1·25.
The image should be saved in a file called `lissajou2.png` in PNG format and be 640×640 pixels in size.
I would remind everyone seeking a “transferable skills” signature that they need to complete this exercise to qualify.