

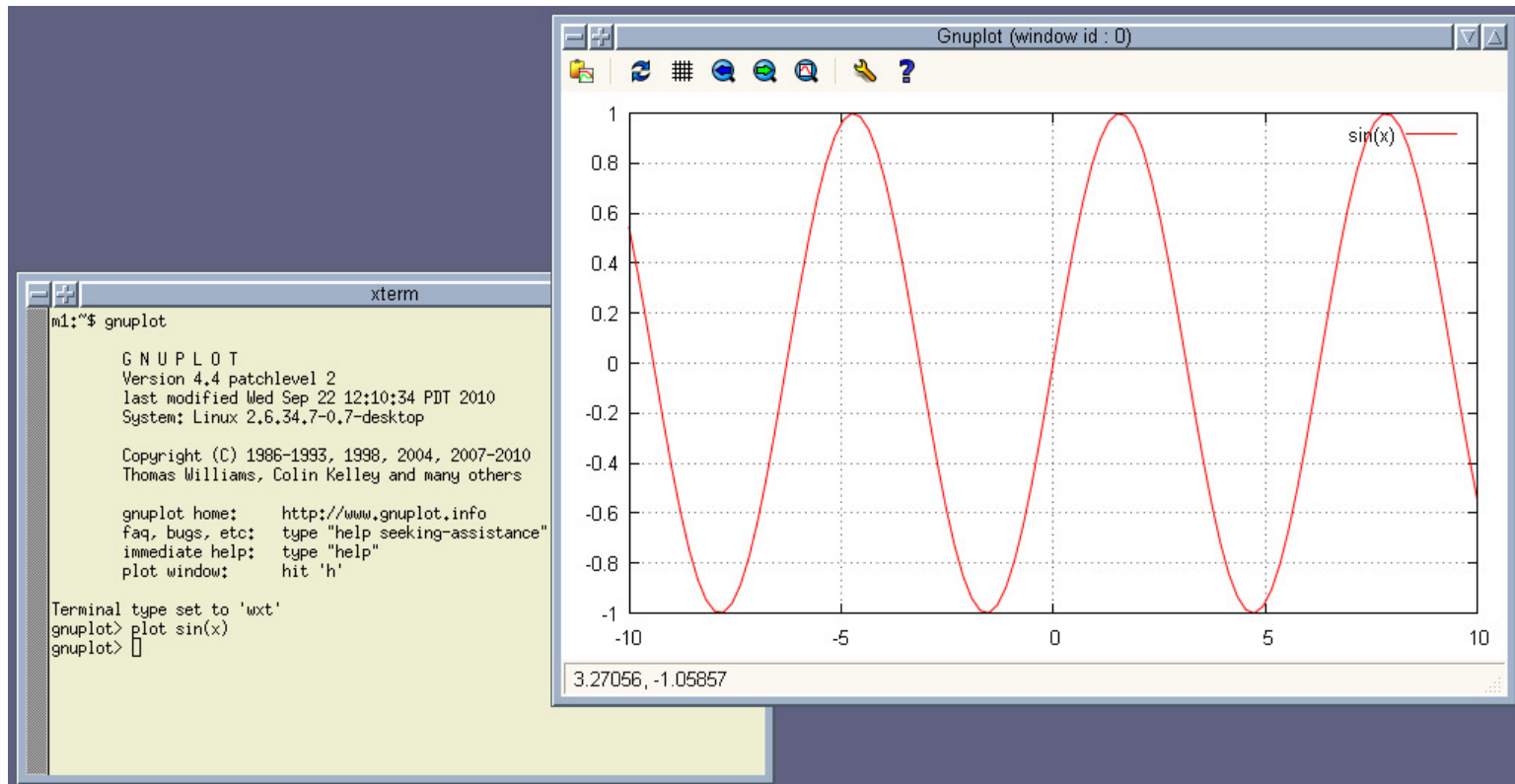
# Miscellaneous Utilities

MJ Rutter  
mjr19@cam

Easter 2013

# Gnuplot

Gnuplot is a simple graph plotting program.



Here it is shown plotting  $\sin(x)$ . It has a simple command line interface, and uses a separate window for the resulting graphics.

# The `plot` command

The `plot` command takes various options. A more complete example is:

```
gnuplot> plot [xmin:xmax][ymin:ymax] f(x),g(x)
```

where one or both of the ranges may be omitted. It can also plot data files.

```
gnuplot> plot 'results.dat' with lines
```

where `results.dat` is a simple file containing a column of `x` values and a column of `y` values.

Omit the `'with lines'` to obtain the points only, use `'with linespoints'` for both. Minimum unique abbreviations are usually accepted, e.g. `'w linesp'` for `'with linespoints'`

Gnuplot's abilities with data files include the ability to extract specific column, to plot error bars given in the data and to perform trivial calculations on the data.

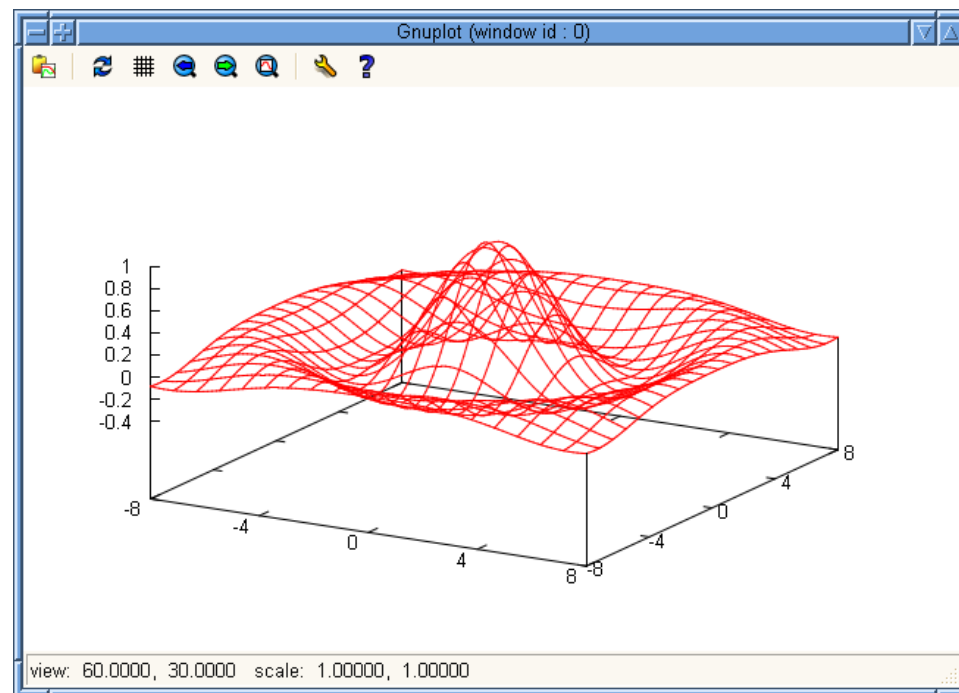
```
plot 't.dat' using ($1):($2>10 ? 10-$3 : $3)
```

(First column is x axis, y axis is third column, or ten minus third column, depending on the value in the second column.)

The online help system is quite extensive. Type `'help'` to investigate it.

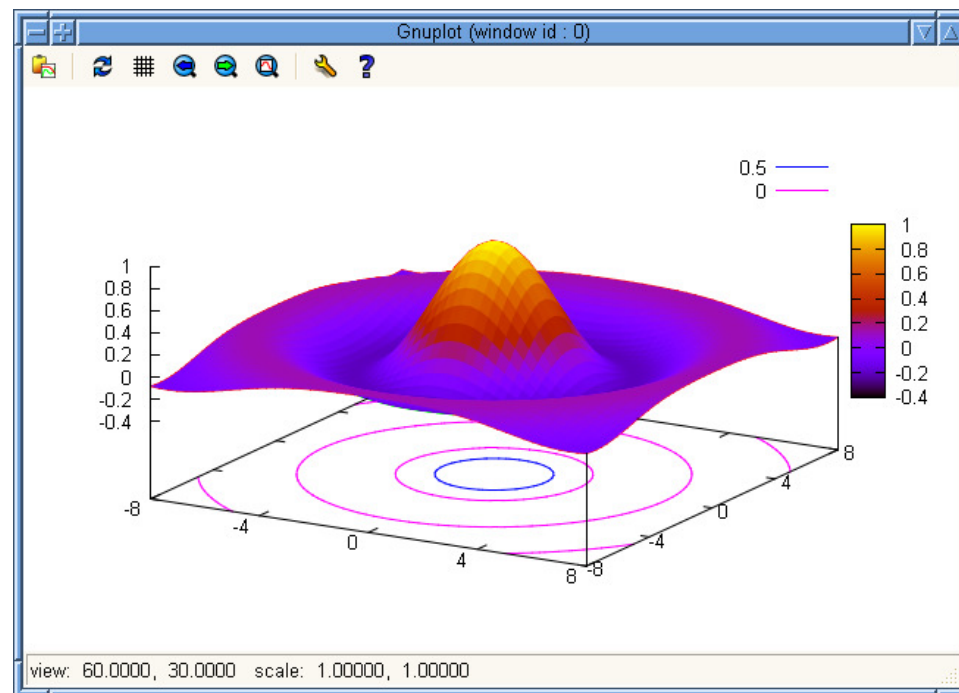
# A 3D plot

```
gnuplot> sinc(x)=sin(x)/x  
gnuplot> unset key  
gnuplot> set xtics 4  
gnuplot> set ytics 4  
gnuplot> set isosamples 20,20  
gnuplot> splot [-8:8][-8:8] sinc(sqrt(x*x+y*y))
```



# More 3D

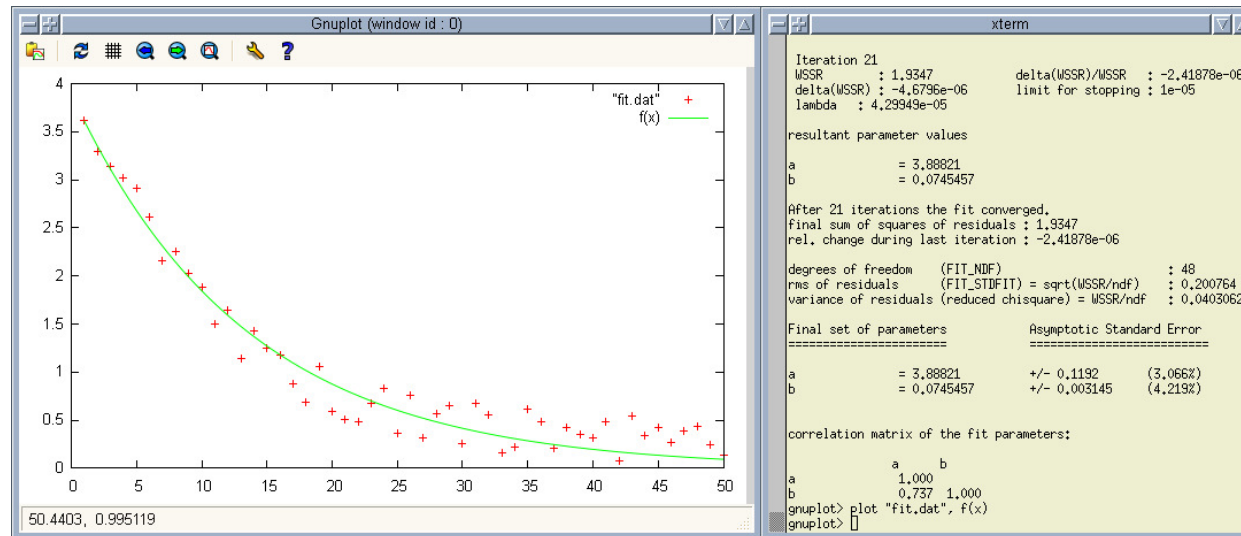
```
gnuplot> set pm3d  
gnuplot> set hidden3d  
gnuplot> set isosamples 50,50  
gnuplot> set contour  
gnuplot> set key  
gnuplot> splot [-8:8][-8:8] sinc(sqrt(x*x+y*y)) notitle
```



# Curve fitting

Gnuplot also does non-linear function fitting.

```
gnuplot> a=3
gnuplot> b=0.5
gnuplot> f(x)=a*exp(-b*x)
gnuplot> fit f(x) 'fit.dat' via a,b
```



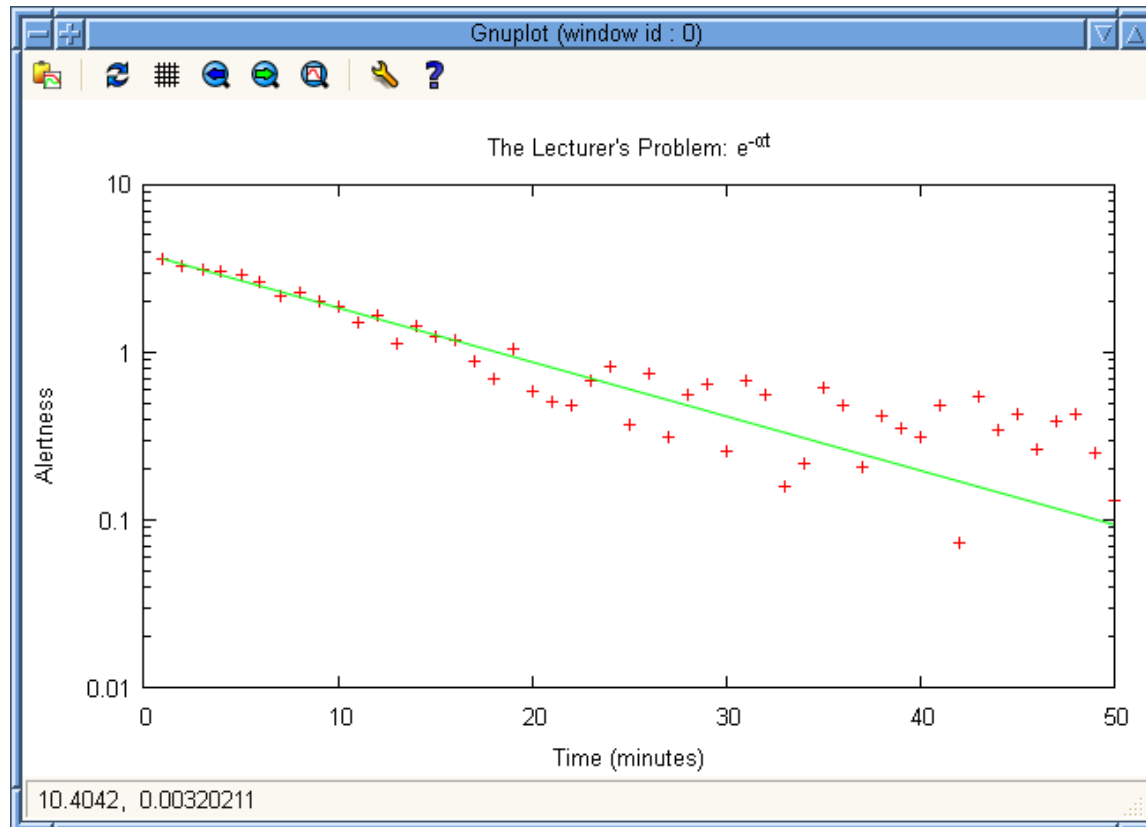
```
do i=1,50
  x=0.1*i ; call random_number(y)
  write(*,*) i,4*exp(-x)+0.5*y
end do
```

produced the data fitted. Notice the noise is sufficient to stop the fitted result ( $a = 3.89 \pm 0.1$  and  $b = 0.075 \pm 0.003$ ) being precisely the underlying function, to an alarming number of standard deviations.

# A labelled plot

```
gnuplot> set xlabel 'Time (minutes)'  
gnuplot> set ylabel 'Alertness'  
gnuplot> set title "The Lecturer's Problem: e^{-at}"  
gnuplot> set logscale y  
gnuplot> set xtics 10  
gnuplot> set nokey  
gnuplot> set term wxt enh  
gnuplot> replot
```

The "enh" terminal types parse this sort of text. The "wxt" and "post" terminals support "enh".



# Gnuplot and other Programs

By default Gnuplot's output is sent to the screen. It can be sent to a file, in which case it is necessary to set both the file name and the type of output required:

```
gnuplot> set terminal postscript
Terminal type set to 'postscript'
Options are 'landscape noenhanced defaultplex \
  leveldefault monochrome colortext \
  dashed dashlength 1.0 linewidth 1.0 butt noclip \
  palfuncparam 2000,0.003 \
  "Helvetica" 14 '
```

```
gnuplot> set term post eps enh colour
Terminal type set to 'postscript'
Options are 'eps enhanced defaultplex \
  leveldefault color colortext \
  dashed dashlength 1.0 linewidth 1.0 butt noclip \
  palfuncparam 2000,0.003 \
  "Helvetica" 14 '
```

```
gnuplot> set output 'my_plot.eps'
gnuplot> replot
gnuplot> set terminal x11
gnuplot> set output
gnuplot> replot
```



## What You Get is Better than What You See

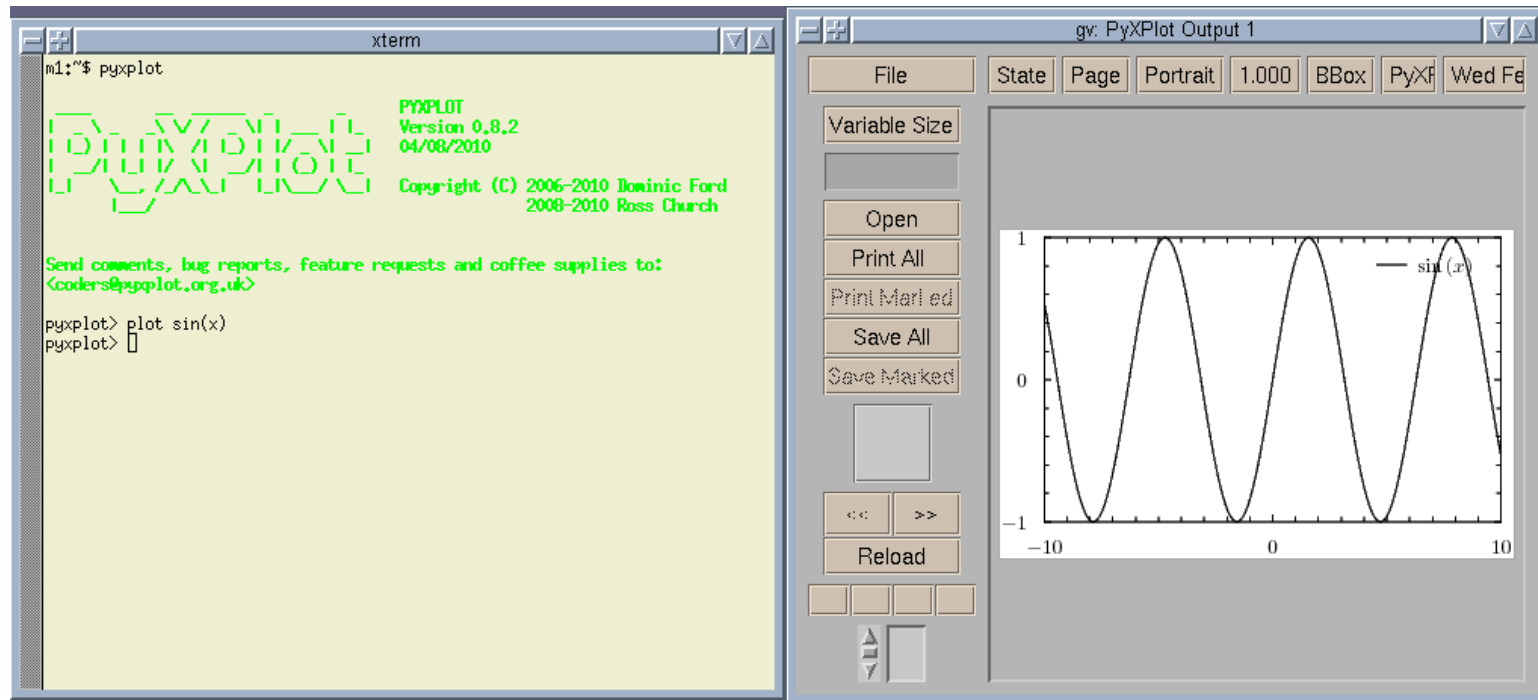
Gnuplot's various terminal types handle things, particularly text, differently. Whereas its PostScript output has always been reasonable, the on-screen output has been more limited, particularly with the older 'x11' terminal type, although it, and the newer 'wxt' terminal type are beginning to catch up.

When you have got a result you like, typing 'save 'myplot.cmd'' will save a command-file containing commands to regenerate the plot. This can be executed using the `load` command.

This suggests a different approach: write a graphing package which produces only PostScript, and rely on ghostscript to give an on-screen representation. This is precisely what PyXPlot does. So let us now repeat the Gnuplot section, but using PyXPlot.

# PyXPlot

PyXPlot was developed by two PhD students in AstroPhysics.

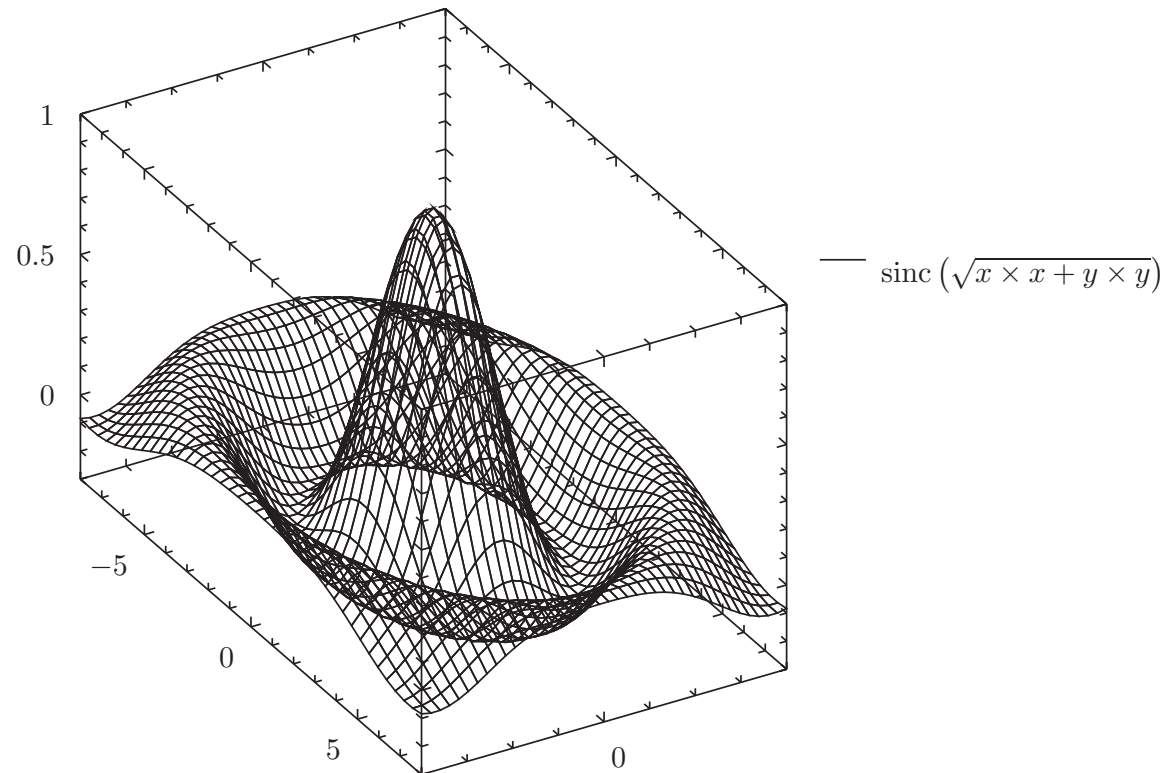


Although it has been written from scratch, its syntax is very similar to that of gnuplot.

# PyXPlot in 3D

Gnuplot's 'splot' needs replacing with 'plot 3d ... with surface'. The sinc function is already defined by pyxplot.

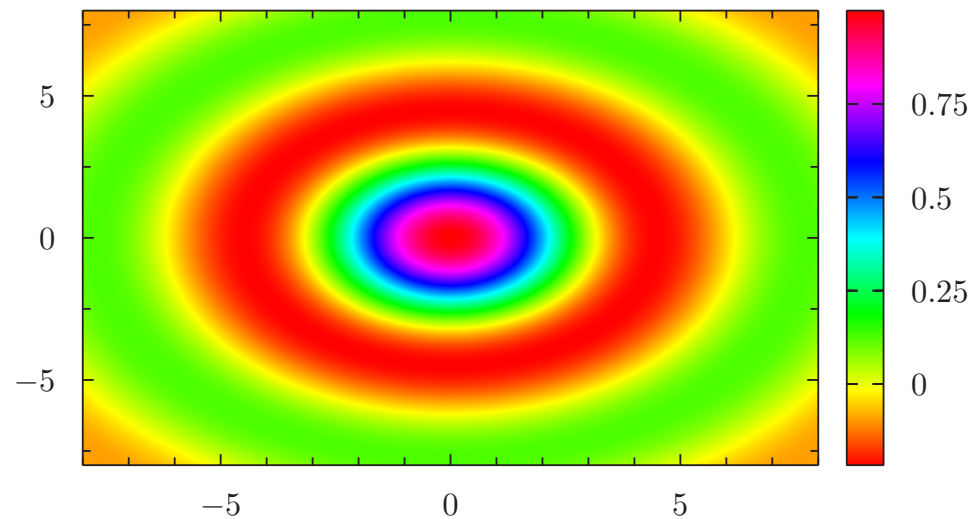
```
pyxplot> plot 3d [-8:8][-8:8] sinc(sqrt(x*x+y*y)) with surface
```



# PyXPlot in 2D

PyXPlot is slightly weak in 3D, but does 2D maps well.

```
pyxplot> set nokey  
pyxplot> set samples grid 400x400  
pyxplot> set colmap hsb(c1):1:1  
pyxplot> plot [-8:8][-8:8] sinc(sqrt(x*x+y*y)) with colourmap
```



The resulting EPS output combines vector text and axes with a bitmap graph.

# Curve fitting in PyXPlot

```
pyxplot> f(x)=a*exp(-b*x)
pyxplot> fit f(x) 'fit.dat' via a,b

# Best fit parameters were:
#
a = 3,8882103
b = 0,074545559

# Estimate of error bars on supplied data, based on their fit to model function
= 0,20070522

# Hessian matrix of log-probability distribution:
#
hessian = [ [-154,31216,4314,5863] , [4314,5863,-210705,6] ]

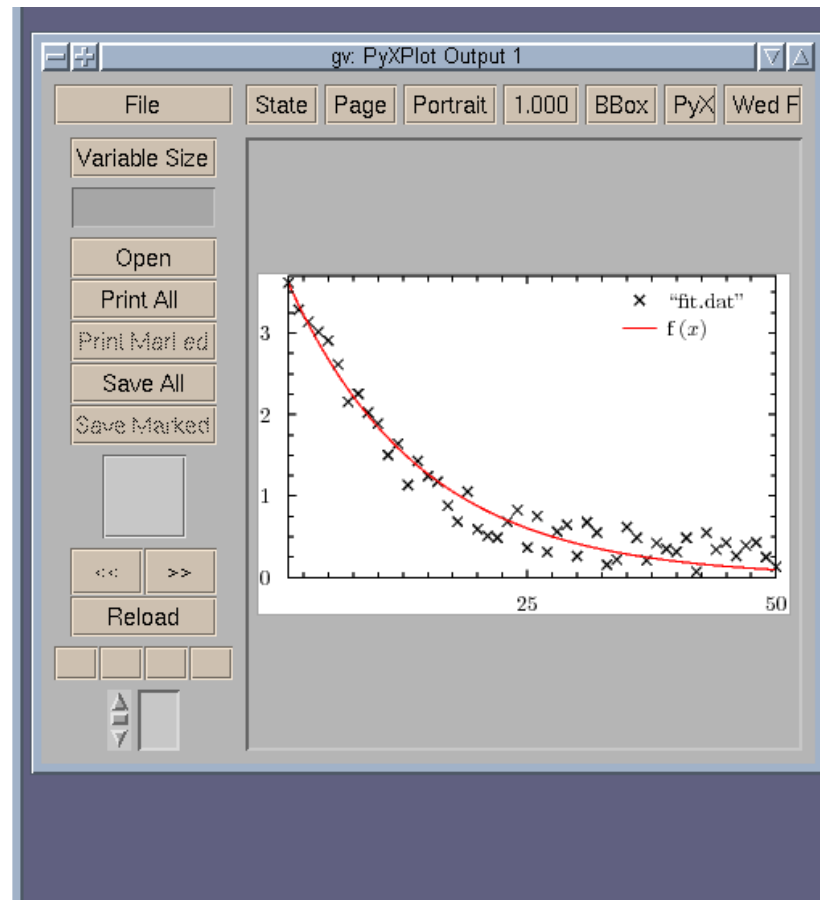
# Covariance matrix of probability distribution:
#
covariance = [ [0,015160005,3,10429105e-04] , [3,10429105e-04,1,11025677e-05] ]

# Correlation matrix of probability distribution:
#
correlation = [ [1,0,75666047] , [0,75666047,1] ]

# Uncertainties in best-fit parameters are:
#
sigma_a = 0,12312597
sigma_b = 0,0033320516

# Summary:
#
a = (3,8882103 +/- 0,12312597)
b = (0,074545559 +/- 0,0033320516)

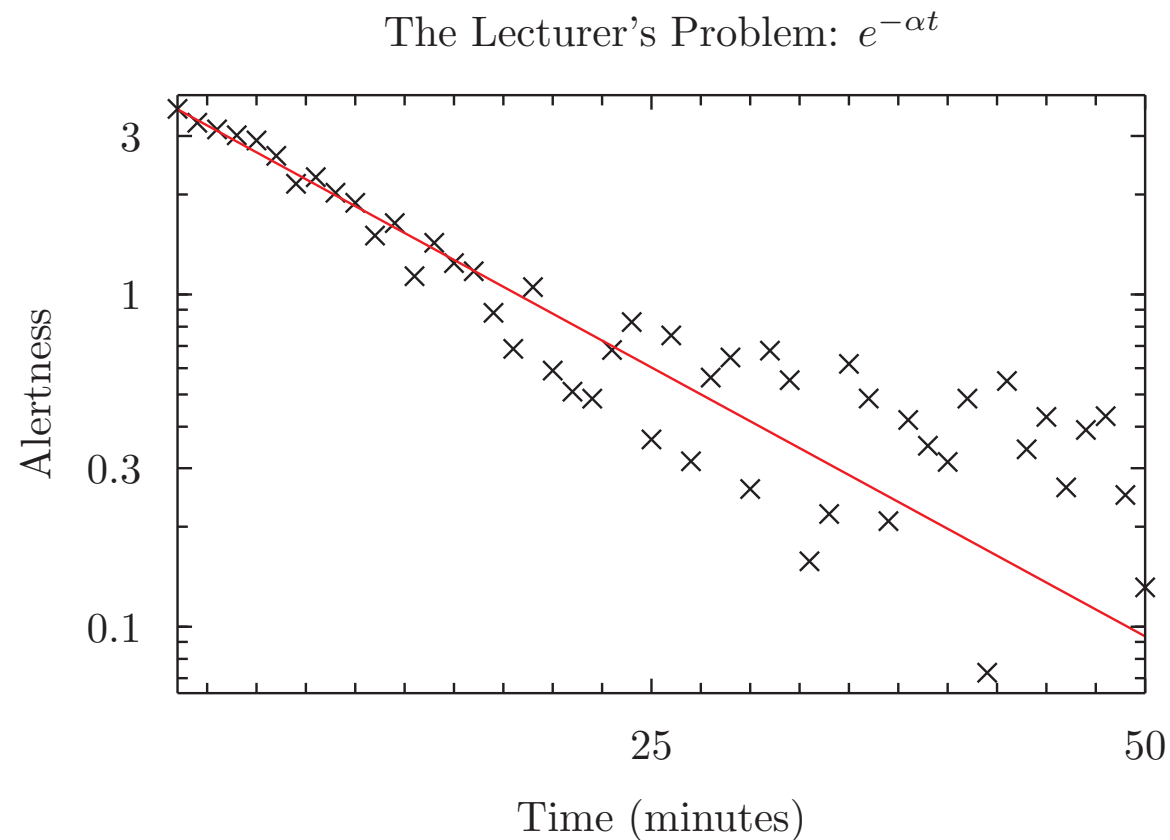
pyxplot>
pyxplot> plot 'fit.dat',f(x)
pyxplot>
```



(There can be a long pause between the parameters being reported, and the full analysis being produced.)

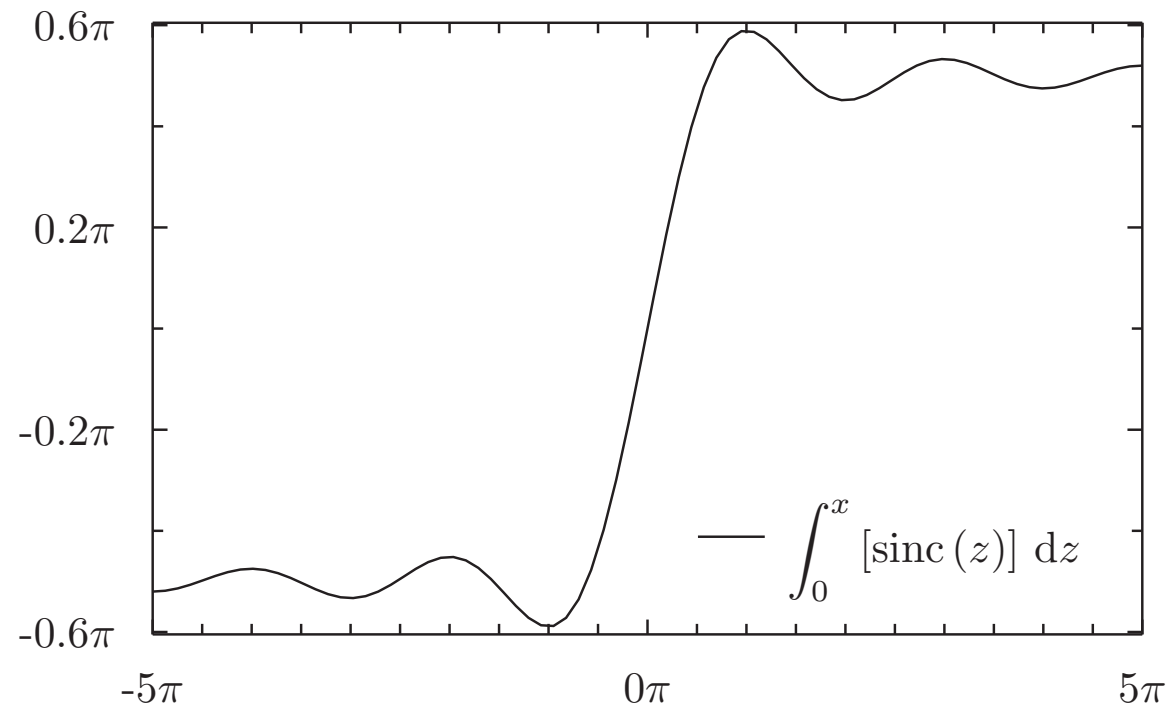
# L<sup>A</sup>T<sub>E</sub>X and PyXPlot

```
pyxplot> set title "The Lecturer's Problem:  $e^{-\alpha t}$ "  
pyxplot> set logscale y  
pyxplot> set ylabel 'Alertness'  
pyxplot> set xlabel 'Time (minutes)'  
pyxplot> set nokey ; set terminal eps ; set output 'out.eps'
```



# PyXPlot and (Numerical) Integration

```
pyxplot> set samples 80  
pyxplot> set key bottom right  
pyxplot> set xformat "%s$\pi$" (x/pi)  
pyxplot> set yformat "%s$\pi$" (y/pi)  
pyxplot> set xrange [-5*pi:5*pi]  
pyxplot> plot int_dz(sinc(z), 0, x)
```



# L<sup>A</sup>T<sub>E</sub>X and Emacs

Whereas most people write L<sup>A</sup>T<sub>E</sub>X in plain text editors, and allow their brains to parse the result and visualise the beautifully typeset result, those whose cerebral capabilities are more limited prefer a little artificial assistance.

Emacs can provide, not only by offering psychiatric help (try Help | Emacs Psychotherapist if you don't believe me), but also by integrating well with `latex` and `xdvi`, and previewing equations.

By default emacs (as installed in TCM) simply provides some syntax highlighting by colour, and uses larger fonts for the arguments of commands such as `\section{}`. This is triggered by the extension of the file being edited – `.tex` files are assumed to be L<sup>A</sup>T<sub>E</sub>X. TCM provides the AUCT<sub>E</sub>X package for emacs, which is a much more sophisticated environment for L<sup>A</sup>T<sub>E</sub>X than plain emacs provides.



# Jumping Around

If one selects Command | TeXing Options | Correlate I/O, runs `latex` via the  $\text{\TeX}$  icon, and then starts the DVI viewer using the button beside the  $\text{\TeX}$  icon (saying ‘yes’ to questions about starting servers), the DVI viewer will start showing the page corresponding to the cursor position in emacs, with the corresponding paragraph highlighted with a red box. If the cursor is moved in emacs, and the icon for the DVI view pressed again, then the existing DVI viewer simply shows the new location.

If instead one presses `{ctrl}{left mouse button}` at a point in the DVI viewer, then the mouse cursor moves back to the emacs window, and the text cursor is repositioned to the start of the corresponding paragraph.

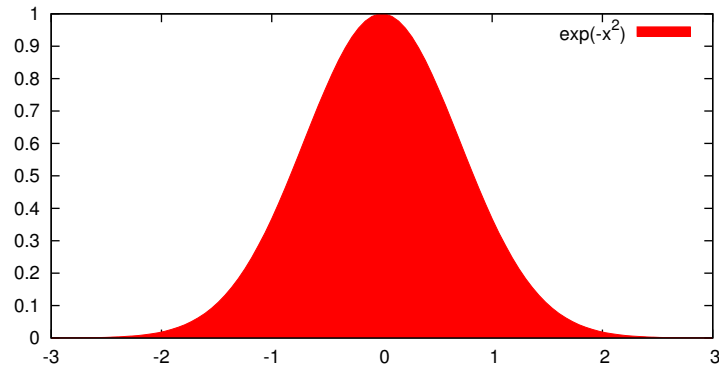
These tricks are fairly robust – they work with multiple file documents, with emacs opening files as necessary, and they work with footnotes.

# Previewing within Emacs

Emacs will replace in-line equations with previews.

$$\int_{-\infty}^{\infty} e^{-x^2} = \sqrt{\pi}$$

as well as including in-line graphics.



# Mere Highlighting

```
\foilhead{Previewing within Emacs}
```

Emacs will replace in-line equations with previews.

```
[\int_{-\infty}^{\infty} e^{-x^2} = \sqrt{\pi} \]
```

as well as including in-line graphics.

```
\begin{center}
```

```
\includegraphics[width=4in]{gauss.eps}
```

```
\end{center}
```

```
% set term post eps colour enh size 4,2
```

```
% set output "gauss.eps"
```

```
% plot [-3:3] exp(-x*x) w filledcurve title "exp(-x^2)"
```

```
\foilhead{Mere Highlighting}
```

# A Preview

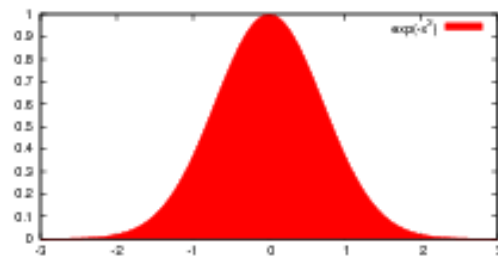
```
\foilhead{Previewing within Emacs}
```

Emacs will replace in-line equations with previews.

$$\int_{-\infty}^{\infty} e^{-x^2} = \sqrt{\pi}$$

as well as including in-line graphics.

```
\begin{center}
```



```
\end{center}
```

```
% set term post eps colour enh size 4,2  
% set output "gauss.eps"  
% plot [-3:3] exp(-x*x) w filledcurve title "exp(-x^2)"
```

```
\foilhead{Mere Highlighting}
```

# Emacs and Abbreviations

Emacs has other tricks too, such as `{ctrl}{c}{}` to insert whatever `\end{ . . . }` text corresponds to the current `\open{ . . . }`.

In `AUCTEX`, one can type `{ctrl}{c}{~}` to toggle `LATEX`'s maths mode, after which there are plenty of shortcuts for entering common Greek letters and other symbols into equations. These are prefixed with ```. So

$$\exp(i\omega t)$$

`\[ \exp(i\omega t) \]`

can be typed as

`\[ `ctrl e (i `w t) \]`

Odd mappings include `h` →  $\eta$ , `f` →  $\phi$ , `q` →  $\chi$ , `w` →  $\omega$ , `y` →  $\psi$ . Other symbols include `A` →  $\forall$ , `E` →  $\exists$ , `I` →  $\infty$ , `N` →  $\nabla$ , and several others.

# Emacs: the Last Word

I don't find previews or short-cuts very helpful, so I am really the wrong person to speak about them. (Just as I never bother with spell-checkers, even though emacs has a  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -aware one.) It can even do spelchecking as one types.

I note too that the method described here for synchronising `xdvi` with emacs using 'source specials' is slowly being replaced by a superior method based on `synctex`. As the version of  $\text{AUCT}_{\text{E}}\text{X}$  in TCM does not yet support `synctex`, I have not mentioned it.

The synchronisation method described here works with `dvi` viewers only, and hence not with those who believe in viewing `pdf` or `ps`. The newer `synctex` is supported by some `pdf` viewers. One difference is that `synctex` produces an extra file of mapping information, whereas this method places all the information directly in the `DVI` file.

For spell-checking as one types, Tools | Spell Checking | Automatic spell checking.

# Xfig

Xfig is dismissed by many people for having a user interface which is decades old. However, it is still alive and well, so must be doing something right. What it gets right is producing compact, robust, portable PostScript output, and permitting quite sophisticated manipulation of images. However, there are a couple of points which need absorbing.

The field with focus is the field the mouse cursor is in.

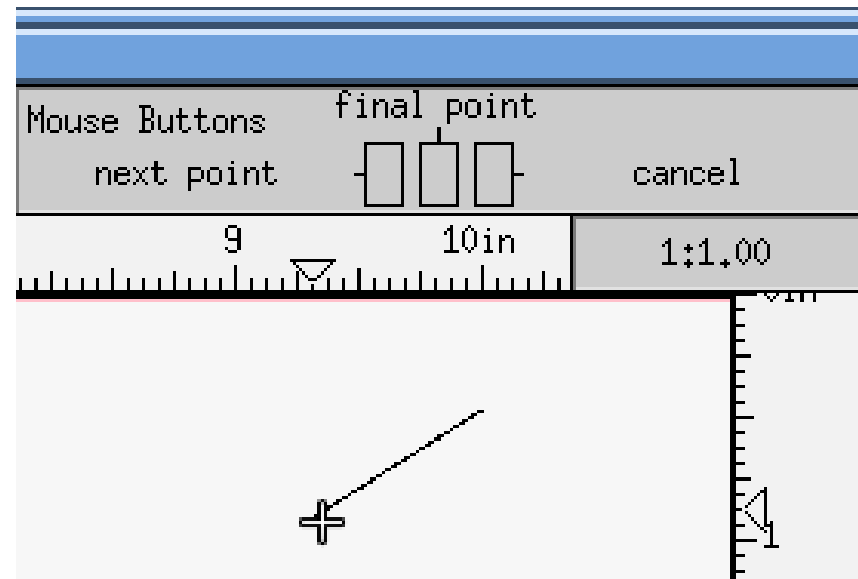
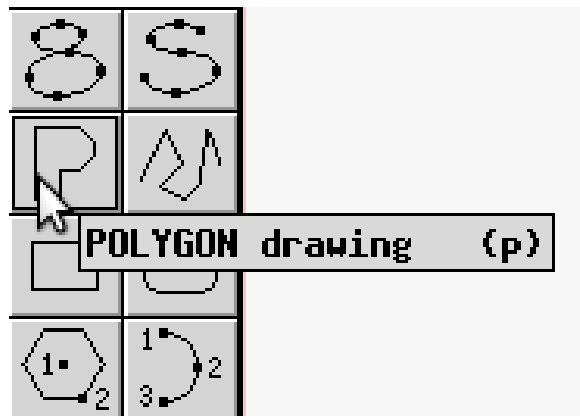
No mouse operations involve dragging.

Mice are assumed to have three buttons.

A text object is a single line of text.

# Friendly Xfig

The buttons on the left provide brief tips if the mouse is hovered over them, and the glyph at the top right describes the current functions of the three mouse buttons.




(When drawing a polyline, left button gives the next point, middle button the final point, and right button cancels the whole line.)



# Xfig for Annotating

I never trust Xfig with anything but EPS images. However, I am notoriously conservative...

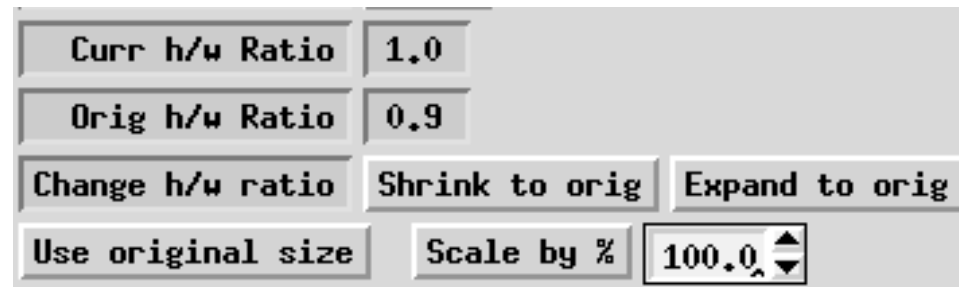
Note that Xfig's display of EPS files can be quite poor. Versions up to and including 3.2.5b dither them to 256 colours for a start (versions in TCM are patched not to do this, and the patch has now been accepted by Xfig's maintainer).




`bmp2eps PM.jpg PM.eps`, then select the 'insert picture' tool in Xfig, , and define a rectangle of approximately the desired size by clicking the left mouse button for the top left and bottom right corners. This will bring up a dialogue box for entering the file name.

Move the mouse cursor into the 'Picture filename' box (remember, *field focus follows mouse*) and type 'PM.eps' and enter. A distorted version of the image will appear.

## Getting an EPS file into Xfig

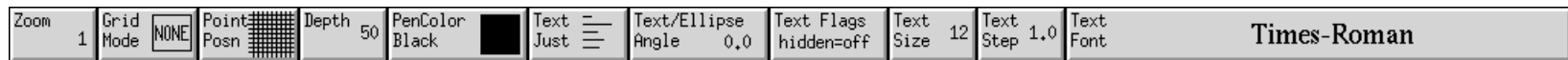
The distortion should be removed by correcting the height/width ratio by either expanding or shrinking the image to its original ratio. Then one can click 'Done' at the top left of the dialogue box to dismiss it.



Further adjustments can be made to position using the move tool, , or, to adjust size, the move control point tool, , probably followed by the edit tool, , which will allow one to correct the height / width ratio again, using the same dialogue box as was used when initially loading the image.

# Adding Text

After clicking on the text tool, **T**, the status bar at the bottom of Xfig's window becomes



As always, a left click on any item here will bring up a dialogue box allowing one to change its value. Changes effect future objects, not past *or current* objects (which can be changed via the edit tool).

The choice of fonts is limited to the standard 'PostScript 35' fonts, or the six basic L<sup>A</sup>T<sub>E</sub>X fonts.

One choses the starting point for the text with a left click. A text object is a single line, but the {Enter} key will cause a new text object to be started immediately below the current one. So we can type 'The Leader of the{Enter}Free World'.

# Ambiguity?

We now have something like




**The Leader of the  
Free World**

The naive will find this ambiguous, so we should add an arrow.

# Precedence

There is clearly an important question of precedence here. Just as that between the leader of an errant colony and Her Majesty's appointed Prime Minister is trivial to resolve, so too is that between an arrow and a picture – the arrow should unambiguously lie on top of the picture.

Like every other vector graphics package, Xfig has a concept of depth associated with every object. The default value is 50, and larger depths lie below smaller depths. So we shall make an arrow with a depth of 45.

An arrow is simply a polyline, so one first selects the polyline tool, , and then adjusts the parameters of the line to be drawn as below. (Depth, PenColor, Width, Arrow Mode, Arrow Size (width 15, length 20, see dialogue box) all changed.) Then left click for start of the line, middle click for end, and the arrow is on the end.




# Perfection



And even if the on-screen representation looks imperfect, remember that all objects are stored in vector form, and will be output to EPS as vectors.

Note too the indicators for the separate layers of depth 45 and 50 at the top right. These can be individually unchecked to hide certain layers. This can greatly assist when attempting to select overlapping objects.


# Compound Objects

Sometimes it is useful to treat a collection of objects as though it was a single object. This can be done with the Glue into Compound tool, .

Here two concentric circles have been combined with four short lines in a spare corner of the canvas. The lines are copies, or rotated copies, of each other, so are all the same length. The six objects are enclosed in a rectangle with the glue tool, defining the rectangle with the middle mouse button, and then combining the objects by pressing the right. (As usual, prompts appear in the top right.)

The resulting compound object can be copied, moved and resized as though it were a single object. (Selecting it by clicking on its marked corners tends to be more reliable than just clicking somewhere on the object.)



A compound object does not have its own depth – each constituent object retains its depth. It can be split back into its constituents using the break compound tool (next to the glue tool). There can be no use for this object in our image, so one can select the delete tool, , & destroy it.

## Other Xfig Functions

Xfig has tools for drawing circles, ellipses, regular polygons, arcs, boxes, boxes with rounded corners, polygons and multisegment lines, splines and closed splines. (The last point of a multipoint object tends to be specified with the middle mouse button.)

Objects can be copied, moved, deleted, inverted, and rotated through arbitrary angles. Text can be spell-checked.



Polylines, polygons and splines can have points added, moved or deleted using the tools shown to the left. Alternatively one can open their edit dialogue boxes.

Xfig can measure lengths, angles and areas.

Xfig has just a single level of undo. Its saved files do not include included images, merely their filenames, so one must keep all together.



## Xfig Tips

Xfig loses, or corrupts, its display remarkably frequently. Press `{ctrl}{L}`, or View | Redraw, to restore.

The scrollbars are slightly unusual. One way of operating them is to drag them, with the middle mouse button, in the direction you wish to go, *not* the opposite direction, as for a ‘conventional’ scrollbar.

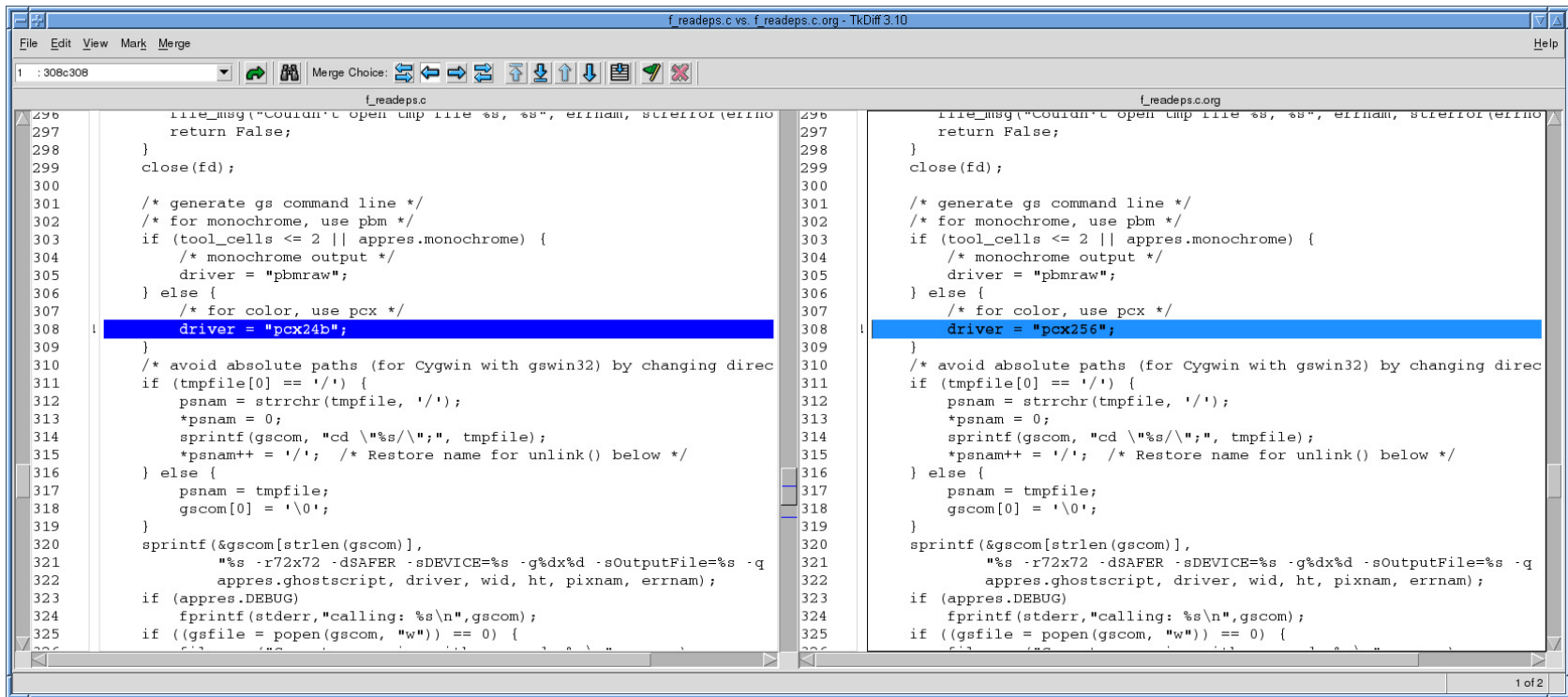
The bottom left contains the current zoom setting. If you wish to try more detailed work, it can be changed.

Xfig can display a grid for guidance. By default it displays none. It can also force points to various positions. By default it forces them to a fine grid (1/16 inches). Setting ‘Point Posn’ to none removes this.

The on-screen representation of text has each character an integer number of pixels wide. The EPS output does not. This can lead to the right-hand end of long strings moving significantly.

# tkdiff

No-one ever seems to remember the existence of this graphical utility for the side-by-side comparison of text files. Being line-based, it works best with output logs and program source files.



A similar utility for KDE is called kompare. For the files given, tkdiff needed 50MB of virtual address space, whereas kompare needed 330MB.

## More `tkdiff`

One can scroll the two windows separately or together, move forwards and backwards amongst the differences, and recompute differences if one of the source files is updated. Line numbers can be displayed or suppressed. Coloured lines in the central scroll bar mark regions with difference, different colours being used for lines which differ, lines which are present on the left and absent on the right, and vice versa.

Strangely X11 has never had a standard graphical diff utility. The pre-CDE environment on Alphas had `dxdiff`, which is very similar to `tkdiff`, but wholly commercial. People who know emacs well can make it do side-by-side diffs, but most of us can't. I know of no GUI utility which works well with reflowed text, and believes that

To those waiting with bated breath for that favourite media catchphrase, the 'U-turn', I have only one thing to say: "You turn if you want to. The lady's not for turning."

and

To those waiting with bated breath for that favourite media catchphrase, the 'U-turn', I have only one thing to say: "You turn if you want to. The lady's not for turning."

are identical. (The non-GUI `wdiff` does work for this.)