Chapter 10: Plot Positioning

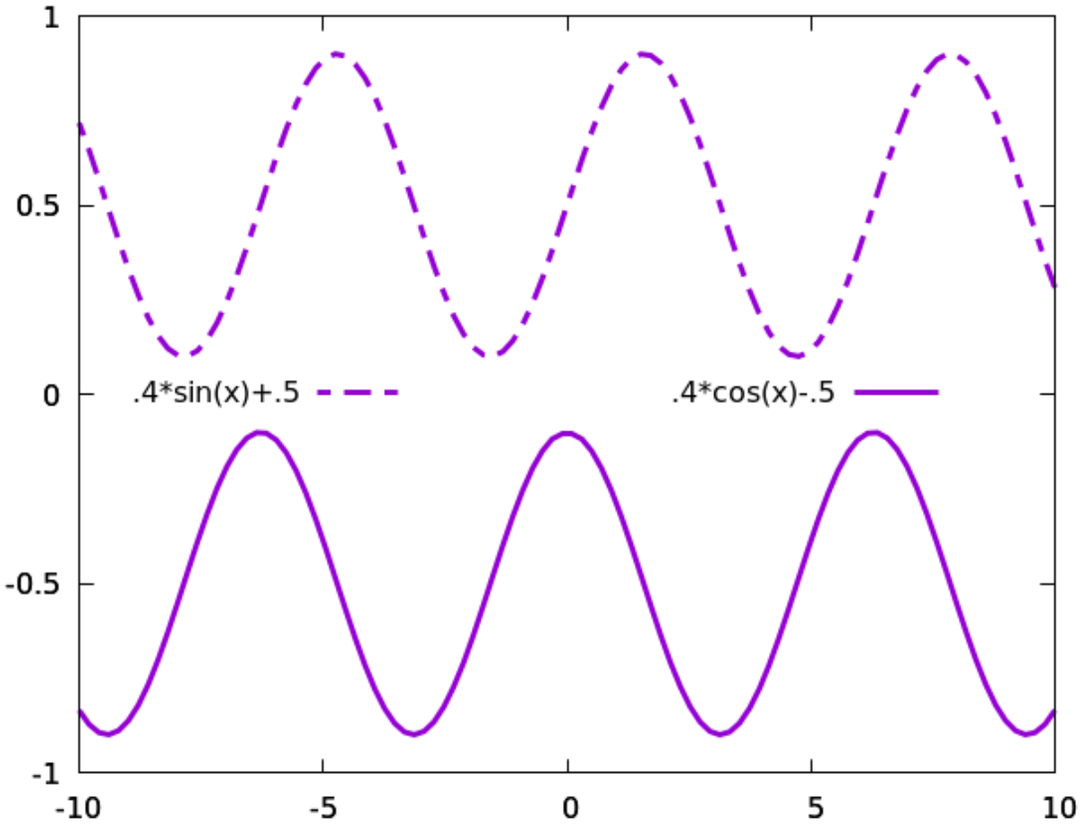
The previous chapter showed you how to embed plots within TEX documents. It's certainly possible to make use of TEX's and TikZ's various graphics commands to position multiple plots in any way you can imagine, with enough work. But gnuplot has its own way to do this: this chapter will show you how to combine several plots into a larger visualization using its *multiplot* mode. Multiplot allows you to place a set of plots anywhere on the page, in a regular array, one inside the other, etc. It is also another, and sometimes more convenient, way, besides issuing a set of `plot` commands separated by commas, to plot multiple curves, etc., within a single frame.

In the example scripts in this chapter you will see the `set multiplot` command that tells gnuplot to enter multiplot mode. If you are entering these commands at the interactive prompt, rather than running a script, you will see that the prompt, which is usually **gnuplot>** , has become **multiplot>** to remind you that you are in a special mode. To leave multiplot mode, you can enter the command `unset multiplot` (which can be abbreviated `unset multi`). In some terminals, nothing is displayed nor written to a file until the `unset multi` command is issued; in others, including most recent versions of interactive terminals, such as the Qt terminal, the plots are built up as you enter the plot commands while in multiplot mode.

In the following example, the plot of two simple functions could of course have been accomplished in the normal (non-multiplot) plotting mode. In that case, the two functions would be included in the same plot command and separated by a comma; if you entered a second, separate plot command, the existing plot would have been replaced. Multiplot mode allows you to build up the plot, adding elements without eliminating existing ones, and so can be useful for interactive experimentation. Since each plot command in multiplot mode generates its own key, this, in conjunction with key positioning, can be used, as here, for the convenient generation of plot labels. To get a similar effect without multiplot mode, you would need to turn off the key and set labels manually for the individual plots.

```
set multiplot
set yr [-1 : 1]
set key at -3,.05
plot .4*sin(x)+.5 lw 3 dt "-_"
set key at 8,.05
plot .4*cos(x)-.5 lw 3
```

Open script

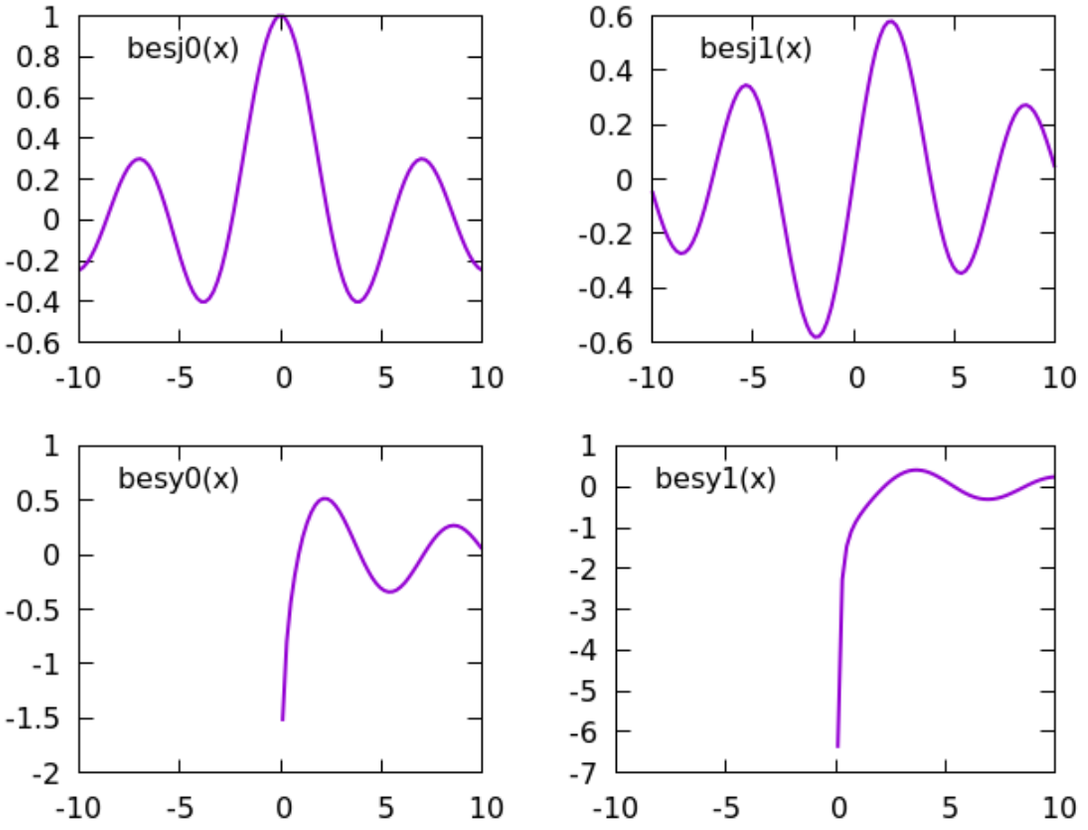


Arrays of plots

This example illustrates the use of automatic grid layouts in multiplot mode, as well as gnuplot's knowledge of Bessel functions. The command `set multiplot layout 2, 2` creates a 2 × 2 grid of plots, which are filled by subsequent plot commands from left to right and top to bottom. Of course, you can substitute any numbers in place of the "2"s to get a rectangular array of any shape you need. The key is used as a convenient titling mechanism, with the `samplen` set to avoid producing a redundant sample line (the tic length is added to the set value of `samplen`, so a negative value must be set to end up with a total length ≤ 0).

```
set multiplot layout 2, 2
set key top left samplen -1
plot besj0(x) lw 2
plot besj1(x) lw 2
plot besy0(x) lw 2
plot besy1(x) lw 2
```

Open script

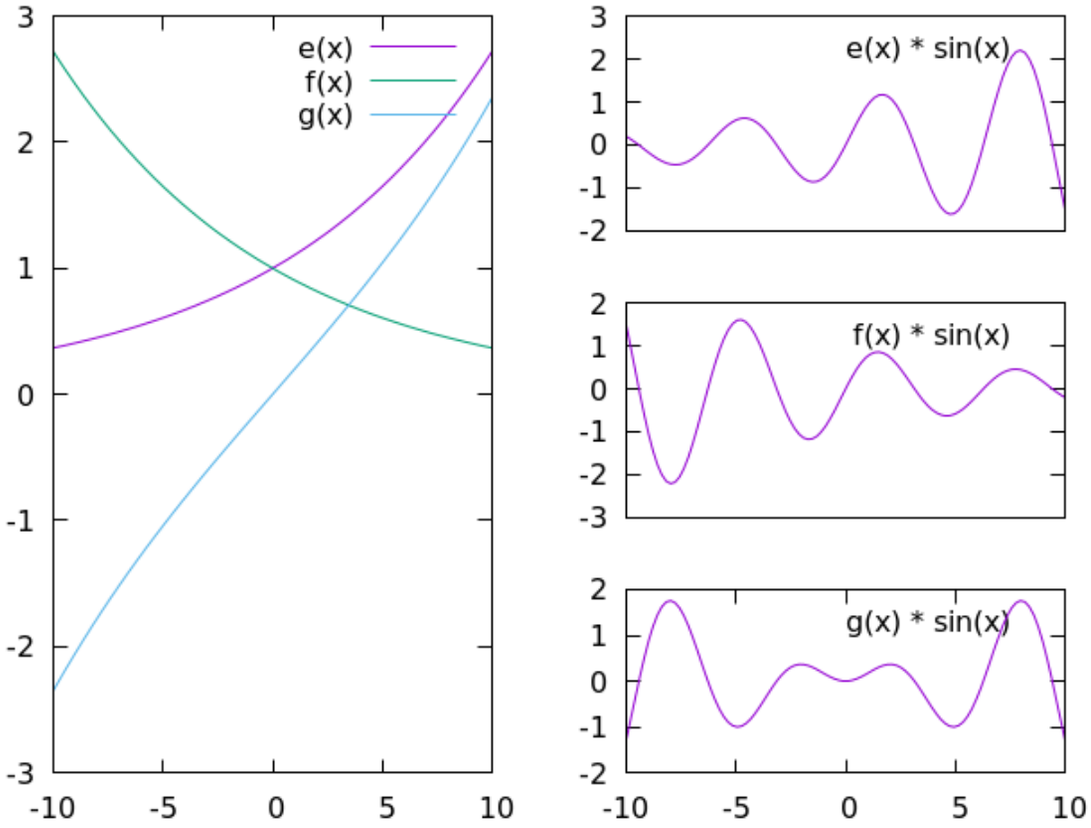


Manual plot positioning

If you need an arrangement of plots more complex than a regular rectangular grid, you must specify the origin and size for each plot manually. This is accomplished with a `set origin` and `set size` for each plot that is to make up part of the total illustration. But what coordinate system do these commands use? They can't use any of the coordinate systems that are defined relative to a plot's axes, nor the `graph` coordinate system, as these could be different for each individual plot. For these purposes, gnuplot provides `screen` coordinates, automatically used by these commands. Screen coordinates go from 0 at the bottom and left to 1 at the top and right, relative to the entire illustration. The `set origin` command sets the location of the lower-left corner of the plot, including the invisible margin.

```
set multi
e(x) = exp(x/10)
f(x) = exp(-x/10)
g(x) = e(x) - f(x)
set size .5, 1
plot e(x), f(x), g(x)
set ytics 1
set key samplen -1
set size .5, 1/3.
set origin .5, 0
plot g(x) * sin(x)
unset xtics
set origin .5, 1./3
plot f(x) * sin(x)
set origin .5, 2./3
plot e(x) * sin(x)
```

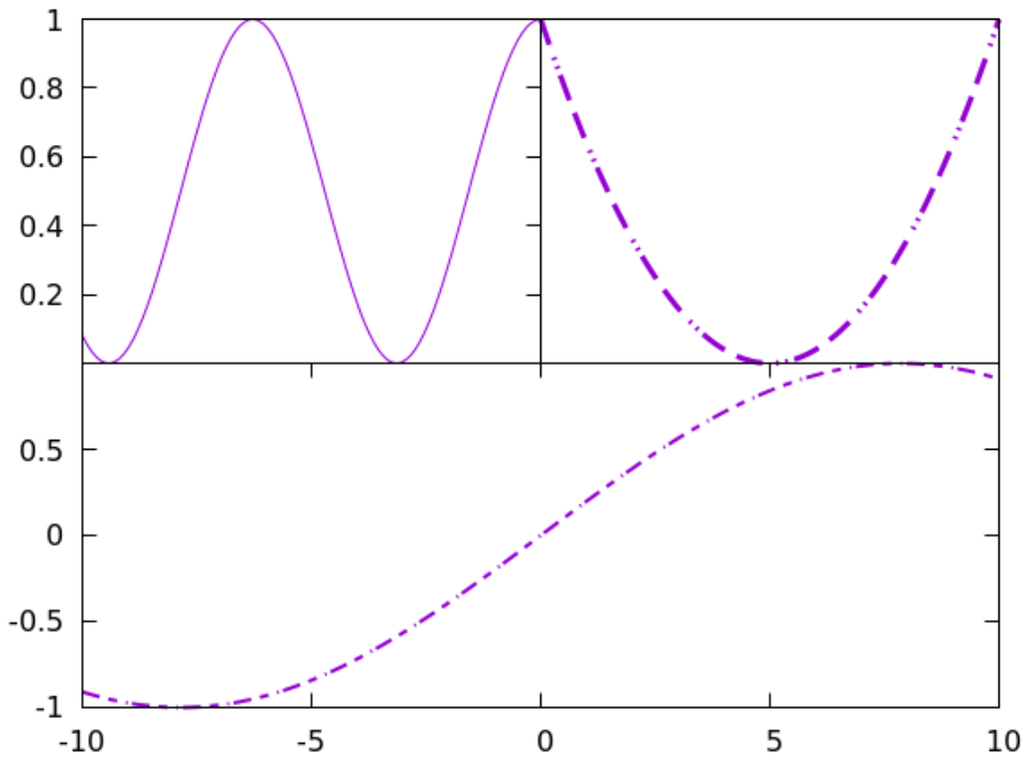
Open script



If you want to align plots precisely, using `set size` and `set origin` commands is not the best approach, because the automatic margins that gnuplot adds to make room for axis labels creates unpredictable spaces around the plots. You can set the margins manually, but that still fails to work for precise alignment. What does work is to use a special margin command that gnuplot provides just for this purpose. The `set Xmargin at screen Y` command, where `X` can be `r`, `l`, `b`, or `t`, and `Y` is the screen coordinate, causes the given edge of the plot to be placed exactly where you specify. This replaces the `size` and `origin` settings, which are ignored when these commands are issued. A simple example should make this clear:

```
set multi; unset key
set xr [-10 : 10]; set yr [-1 : 1]
set ytics -1, .5, .5
set bmargin at screen .1
set lmargin at screen .1
set rmargin at screen .9
set tmargin at screen .5
plot sin(x/5) lw 2 dt "_-."
unset xtics
set xr [-10 : 0]; set yr [0 : 1]
set ytics .2, .2, 1
set bmargin at screen .5
set tmargin at screen .9
set rmargin at screen .5
plot cos(x/2)**2
unset ytics; set xr [0 : 10]
set lmargin at screen .5
set rmargin at screen .9
plot (x-5)**2/25 lw 3 dt "_.._"
```

Open script



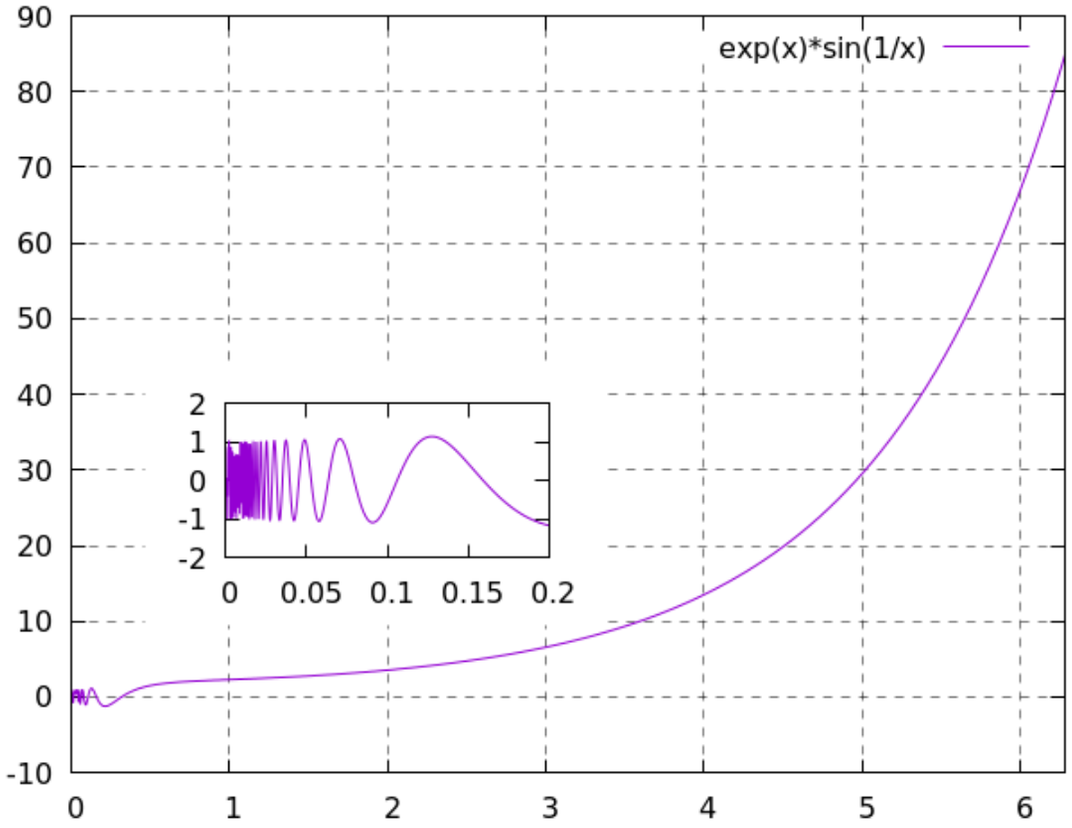
There are a few details about the previous example that it is worthwhile to dwell on. The bottom and top graphs have different y-scales, which is not a problem, since they do not share a y-axis. But we needed to set the ytics carefully to avoid a collision of axis labels. On the top-right graph, we turned off the ytics; the alignment implies that the two top graphs share the same y-range, which they do, so the labels on the top-left graph can serve for both. You must be careful, when you align plots in this way, that they do in fact have the axis ranges that their positioning implies. We turned of the xtics for the top graphs, allowing the x-axis on the bottom graph to indicate their x values. Notice how we set the x-ranges of the top graphs to agree with their alignment with the x-axis of the bottom graph. Finally, notice how the graphs are aligned precisely using the `at screen` margin commands; this would be difficult if not impossible to achieve with the `set origin` and `set size` commands.

Inset plots

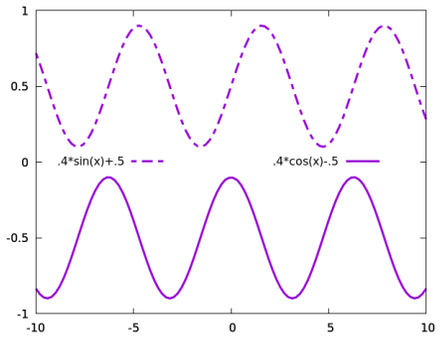
A small graph placed inside a larger one, to illustrate some detail of the latter, perhaps at a magnified scale, is called an *inset plot*. They are easy to make using gnuplot in multiplot mode, as shown in this example. The only new command in this script is the `clear` command, which erases everything in the area defined by the current `set origin` and `set size` commands. This use useful when making inset plots, to erase the grid lines in the area that will be used for the inset, as they will not align with the latter's tic marks and create confusion. The `clear` command doesn't work with areas defined by the `set Xmargin at screen` commands, only the `set origin` and `set size` commands. We also turned off the grid and key in the inset, to avoid clutter.

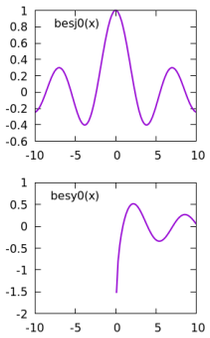
```
set multi
set samples 1000
set grid lt -1 dt "_ "
set xtics 1
set ytics 10
plot [0:2*pi] exp(x)*sin(1/x)
set origin .15, .25
set size .4, .3
clear
unset grid
unset key
set xtics .05
set ytics 1
plot [0:.2] exp(x)*sin(1/x)
```

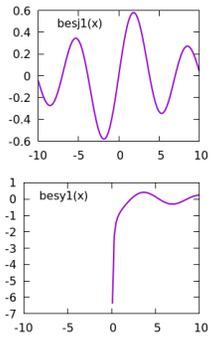
Open script

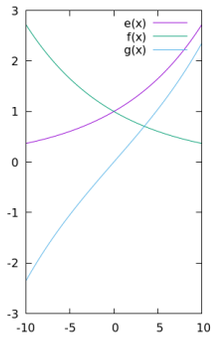


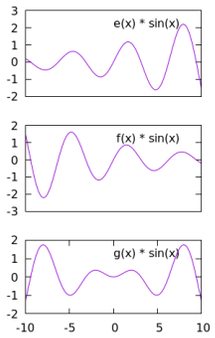
Index of Plots

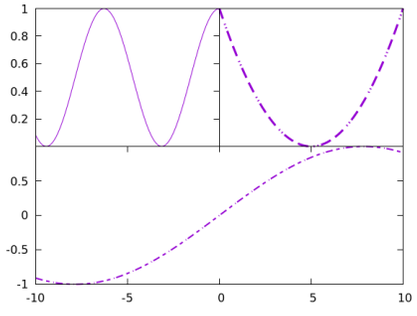


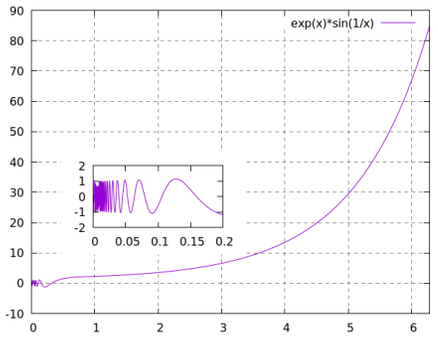












Index