Chapter 7: Contour Plots and Heat Maps

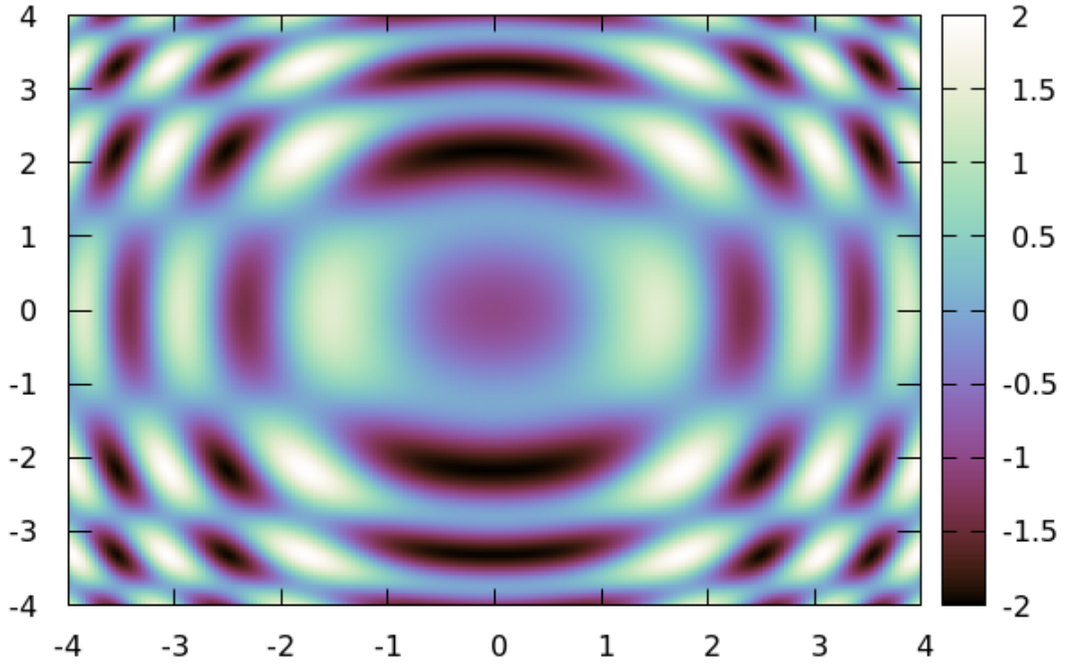
This chapter, as the previous one, deals with three- (or more) dimensional data and functions. The difference is in the way the information is visualized. Instead of a 2D rendition of a 3D surface, the plots you will learn how to make now use color, symbols, curves and labels to visualize the data or functional relationship as a plane (flat) figure. Most of these types of graphs are probably familiar to you. If you take some of the colored surface plots from the last chapter and remove the surface information, leaving the color only, and view the plots from above, you will have what is sometimes called a *heat map*, where the value is indicated simply by color. A *contour plot* traces value by drawing curves, just as in the topographical maps familiar to hikers. The two types can, of course be combined in various ways. You can also create a type of 4D plot, called a vector plot, that draws arrows or similar symbols to show the x and y components of a velocity field or something similar. Finally, any of the types of plots introduced in this chapter can be combined with the surface plots of the previous chapter — you can even embed vectors in surfaces.

Heat Maps

You make heat maps by invoking the pm3d style, just as when you make the colored surface maps of the previous chapter — and all the same options for color palettes are available. The difference is that you want to display a "bird's eye" view of the plot, by setting the view angles correctly. Gnuplot has a shortcut for this: just say `set view map`.

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
unset key
set palette cubehelix start 1.2 cycles -1. saturation 1
set view map
splot sin(y**2+x**2) - cos(x**2) with pm3d
```

Open script

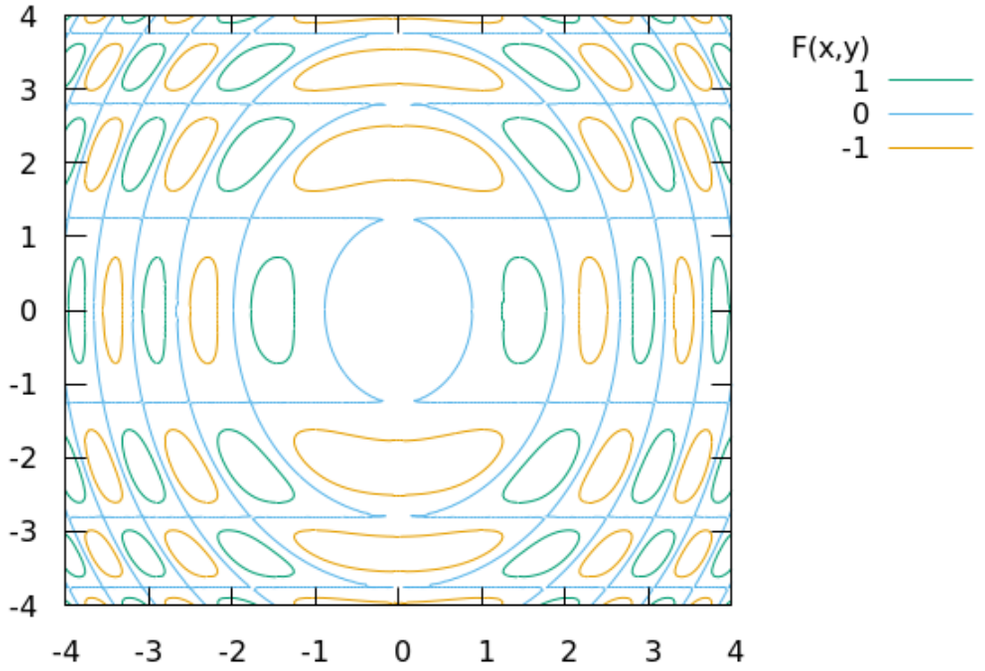


Contour Plots

This example shows how to make a basic contour plot without contour labels. We'll leave the selection of contour lines up to gnuplot, and let it draw a key, which will identify the contour values using color. In order to compare methods, we'll plot the same function in all the examples in this part of the chapter. The highlighted command tell gnuplot to draw contours on the *base*, which refers to the bottom of the plotting box. The other locations for these purposes are the *surface*, referring to the plotted surface, if there is one, and *both*. We'll see examples of all the possibilities later in the chapter. We need to turn off the surface, since we just want contours; if we don't do this, gnuplot will draw the surface isolines as well, which we don't want. The isoline and samples settings have a similar effect in contour plots as when plotting surfaces.

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
set key rmargin
unset surf
set contour base
set view map
splot sin(y**2+x**2) - cos(x**2) title "F(x,y)"
```

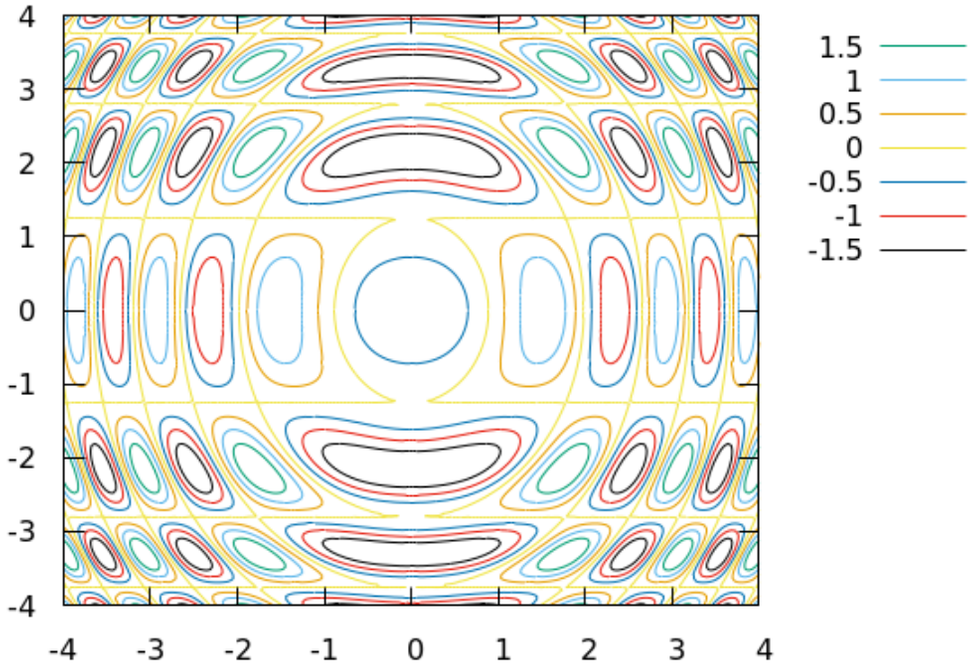
Open script



We can exert some influence over the number of contours that gnuplot draws with the highlighted command in the script below. Note that this command doesn't get you the exact number of contour lines that you ask for in most cases; gnuplot will choose a number that gives simple labels (type `help set cntrparam` for all the details). This is a good command to use if you need more or fewer contour lines than the default, but want to keep the plot neat and simple.

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
set key rmargin
unset surf
set contour base
set cntrparam levels auto 9
set view map
splot sin(y**2+x**2) - cos(x**2) title ""
```

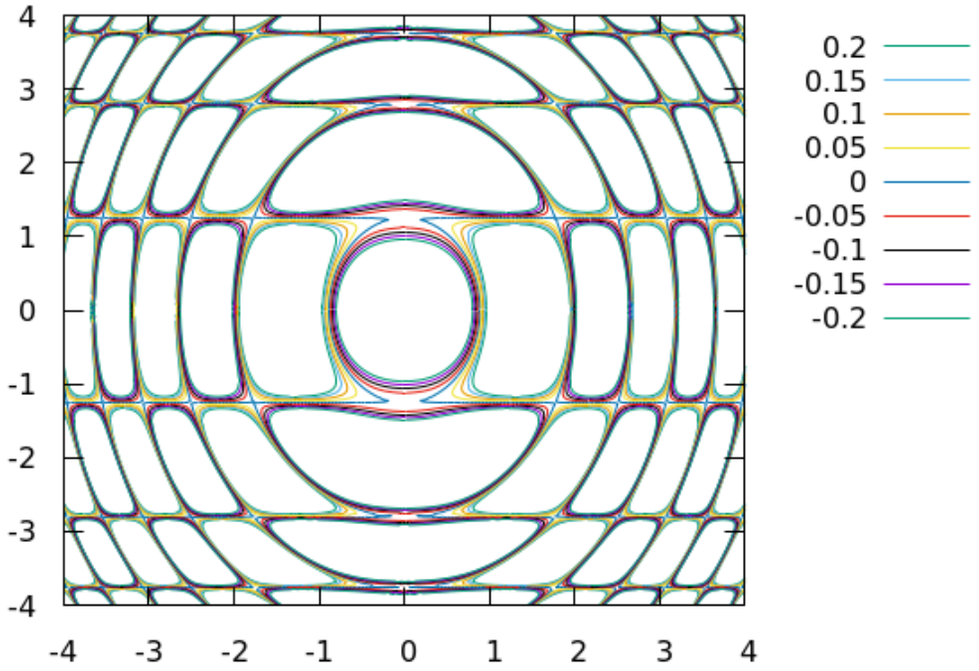
Open script



For more control over contour levels, try the command in this script. The three numbers in the command are the beginning value, the increment, and the ending value. We've chosen values to highlight the nodal areas of the function (where it is near zero).

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
set key rmargin
unset surf
set contour base
set cntrparam levels incremental -.2, .05, .2
set view map
splot sin(y**2+x**2) - cos(x**2) title ""
```

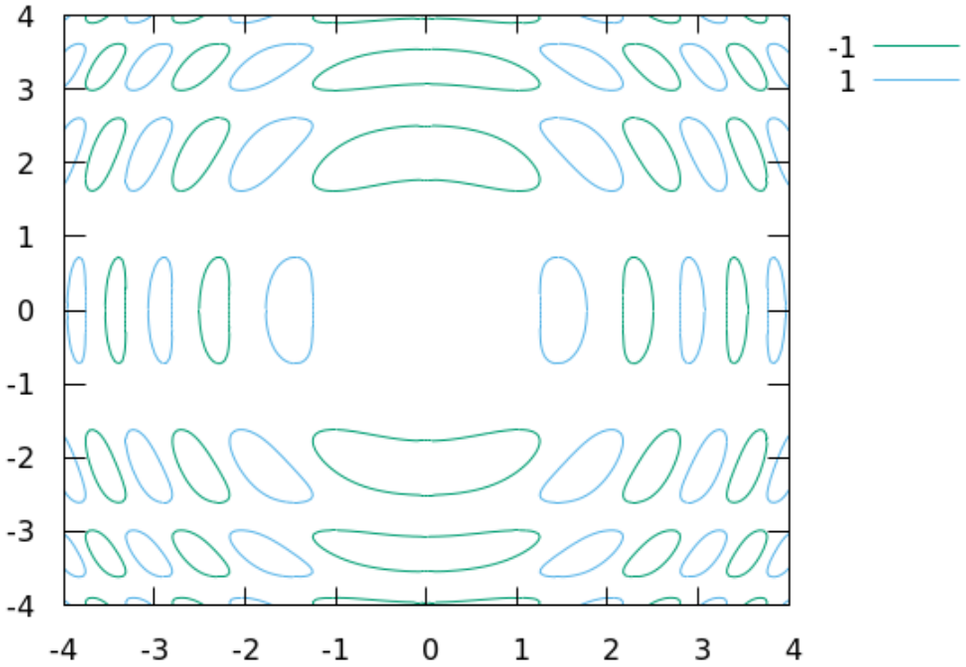
Open script



Finally, you can choose particular values to use for contour levels, as shown here:

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
set key rmargin
unset surf
set contour base
set cntrparam levels discrete 1, -1
set view map
splot sin(y**2+x**2) - cos(x**2) title ""
```

Open script

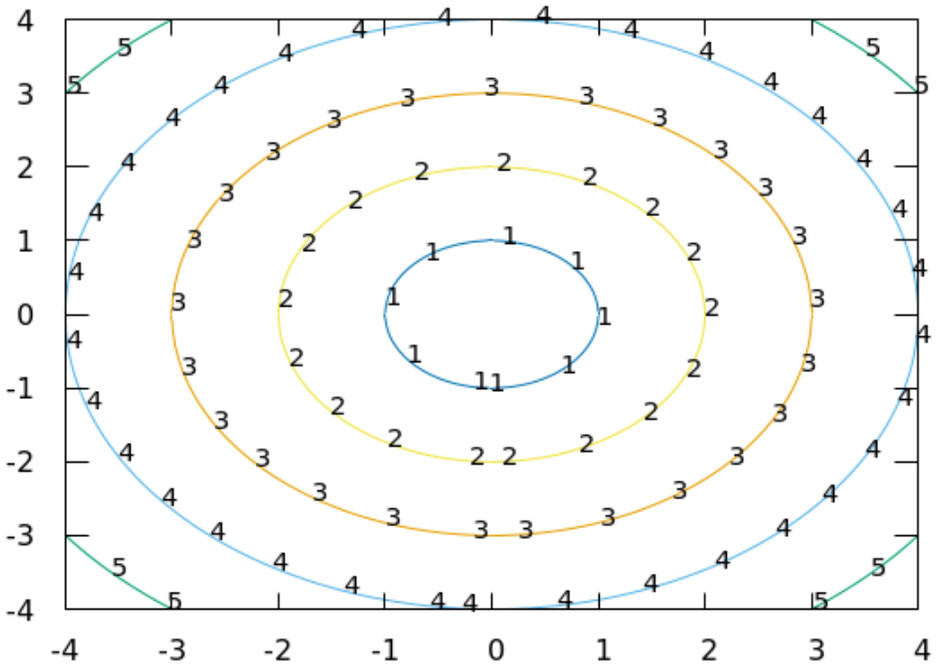


Labeled Contours

Contour plots are generally easier to read if they have numerical labels on the contour lines, rather than in a legend (but this depends on the details of your particular plot). Gnuplot's command for labeling contour lines is `set cntrlabel`; when you use it, it makes sense to turn off the legend, but you can have both if you really want to. Contours are made up of a large number of short line segments. The `cntrlabel` setting parameters control how many labels are put on each contour line, and where on the line they start: the first number is which segment of each line they start on, and the second number is the number of segments between labels. After those two parameters, you can add formatting commands, as we do below to select a font size. Getting the labels on the contours requires a double plot command: one to draw the lines and one to apply the text, using the `with labels` style command. In order to illustrate contour labeling options, we'll plot a simpler function:

```
set xrange [-4:4]
set yrange [-4:4]
set iso 200
set samp 200
unset key
unset surf
set contour base
set view map
set cntrparam levels auto 9
set cntrlabel start 1 interval 25 font ",11"
f(x,y) = sqrt(x**2 + y**2)
splot f(x,y), f(x,y) with labels
```

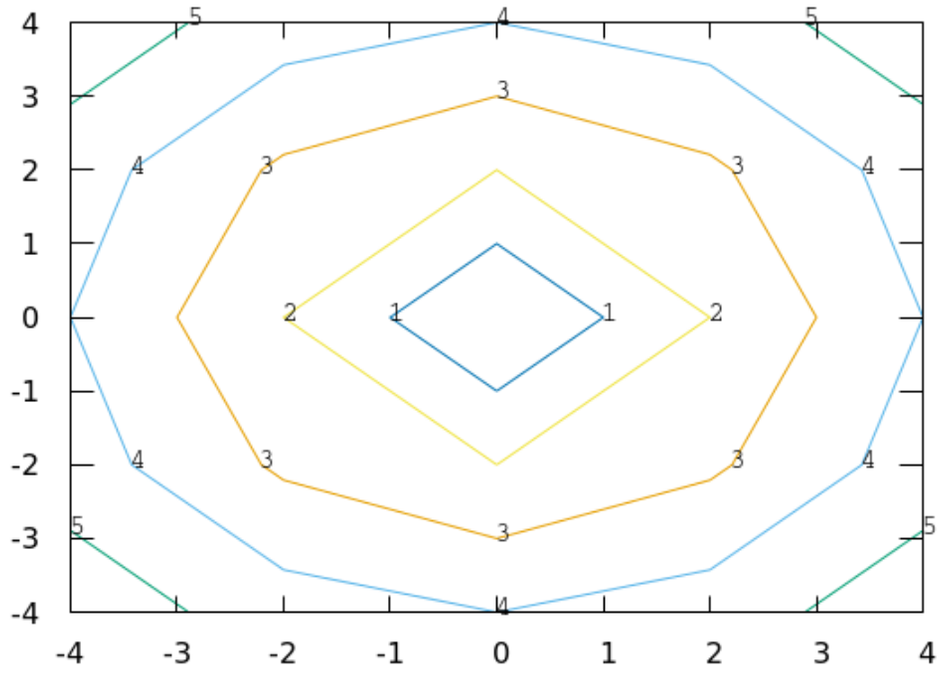
Open script



If we make the `isolines` and `samples` settings very small, so that we can easily see the individual line segments making up the contour lines, we can more clearly see the effects of the `cntrlabel` command parameters. Here is the previous plot with a label at every other line segment:

```
set xrange [-4:4]
set yrange [-4:4]
set iso 5
set samp 5
unset key
unset surf
set contour base
set view map
set cntrparam levels auto 9
set cntrlabel start 1 interval 2 font "Courier,11"
f(x,y) = sqrt(x**2 + y**2)
splot f(x,y), f(x,y) with labels
```

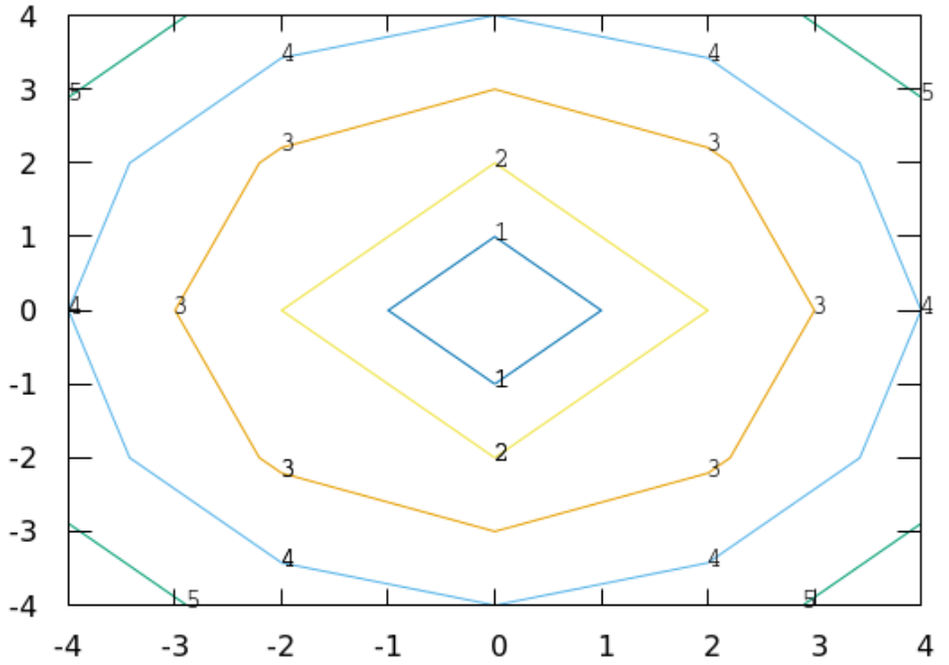
Open script



And here is the same thing, but starting at a different segment:

```
set xrange [-4:4]
set yrange [-4:4]
set iso 5
set samp 5
unset key
unset surf
set contour base
set view map
set cntrparam levels auto 9
set cntrlabel start 2 interval 2 font "Courier,11"
f(x,y) = sqrt(x**2 + y**2)
splot f(x,y), f(x,y) with labels
```

Open script



Vector Plots

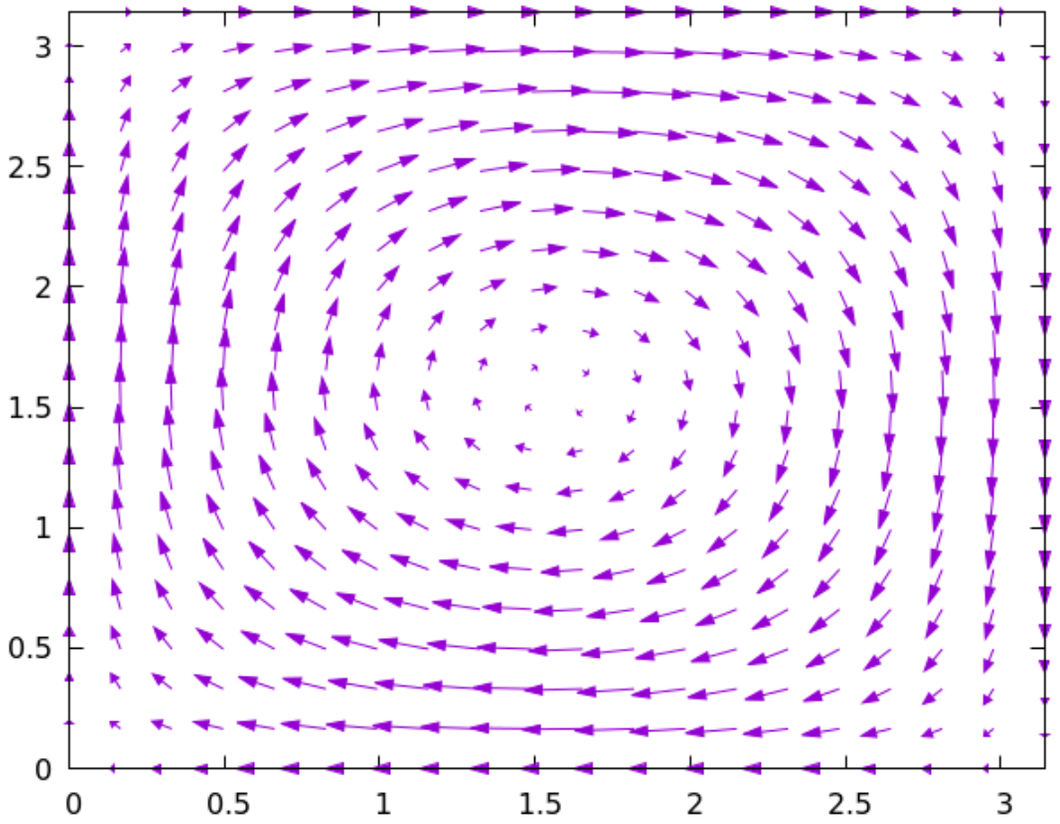
When the quantity that depends on x and y has both a magnitude and a direction it can be represented by an arrow whose length is proportional to the magnitude. If we divide the x-y plane into a grid and draw an arrow at each grid point representing the direction and magnitude at that point, we have a *vector plot*. As we are associating two quantities, the magnitude and direction (or, equivalently, Δx and Δy) for each value of x and y, we can think of this type of visualization as a 4D plot. Vector plots are used for representing fluid flows, electric fields, and many other physical systems.

Vector plots require a data file or the the equivalent pseudofile syntax. Four columns are used for x, y, Δx, and Δy, respectively. We can set the size of the arrowheads, whether they are filled or open, and the angle of the sides of the arrowheads in degrees. The example here uses filled arrowheads of a medium size with a 15° angle. For more details on how to customize the arrow style, type `help arrowstyle`.

This plot represents a rotating flow field.

```
set xrange [0:pi]
set yrange [0:pi]
set iso 20
set samp 20
unset key
a = .2
plot "++" using 1:2:(-a*sin($1)*cos($2)):(a*cos($1)*sin($2))\
          with vectors size .06, 15 filled
```

Open script

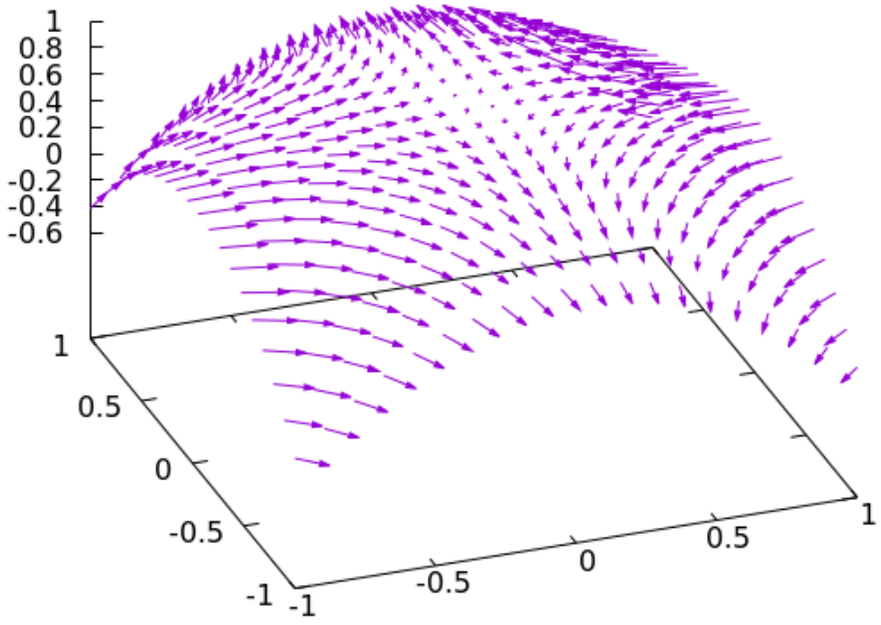


Vectors on a Surface

Gnuplot can also render a vector field *on a surface* rather than merely on a plane. You can use this to represent, for example, 3D flow fields. As in the previous case, you need to use either datafile plotting or the "++" pseudofile. The first three columns define the surface, as when creating the surface plots of the last chapter, but the surface itself is not rendered. The last three columns represent Δx, Δy, and Δz. The following example is the same rotating flow as in the previous example, with the difference that the vectors lie on a surface given by the cosine of the squared distance from the origin.

```
set xr [-1:1]; set yr[-1:1]
set view 50, 340
set iso 20; set samp 20
unset key
a = .2
splot "++" u 1:2:(cos($1**2+$2**2)):(-a*sin($1)*cos($2)):\
             (a*cos($1)*sin($2)):(0) w vec size .02, 15 fill
```

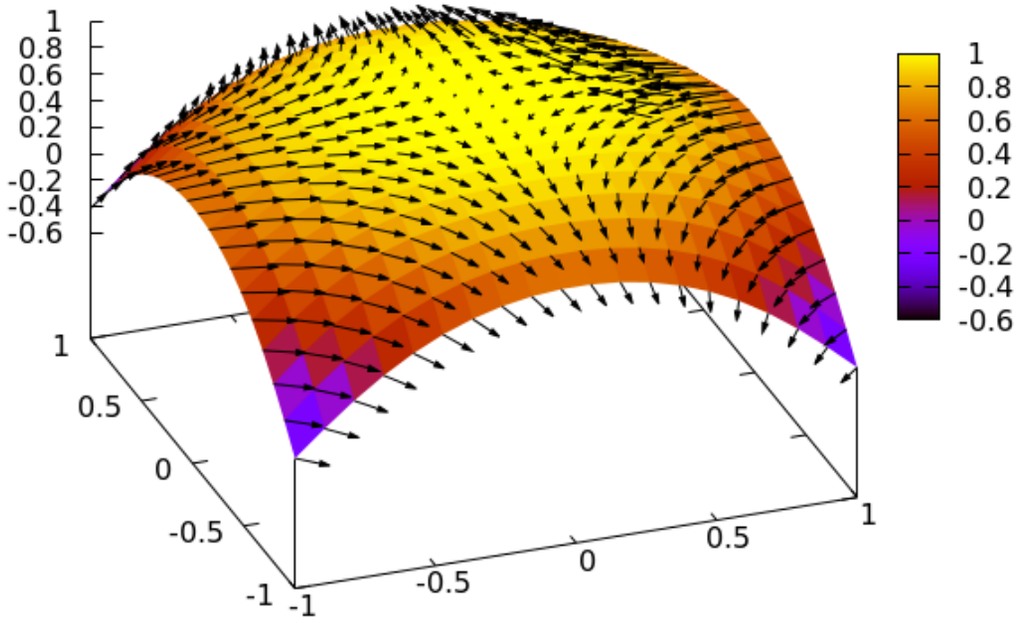
Open script



You can also render the surface in which the vectors are embedded. We do that here with a pm3d surface, which comes first in the plot command so that the arrows are plotted on top of it. In addition, we set the `linecolor` of the vectors to contrast with the coloring of the surface.

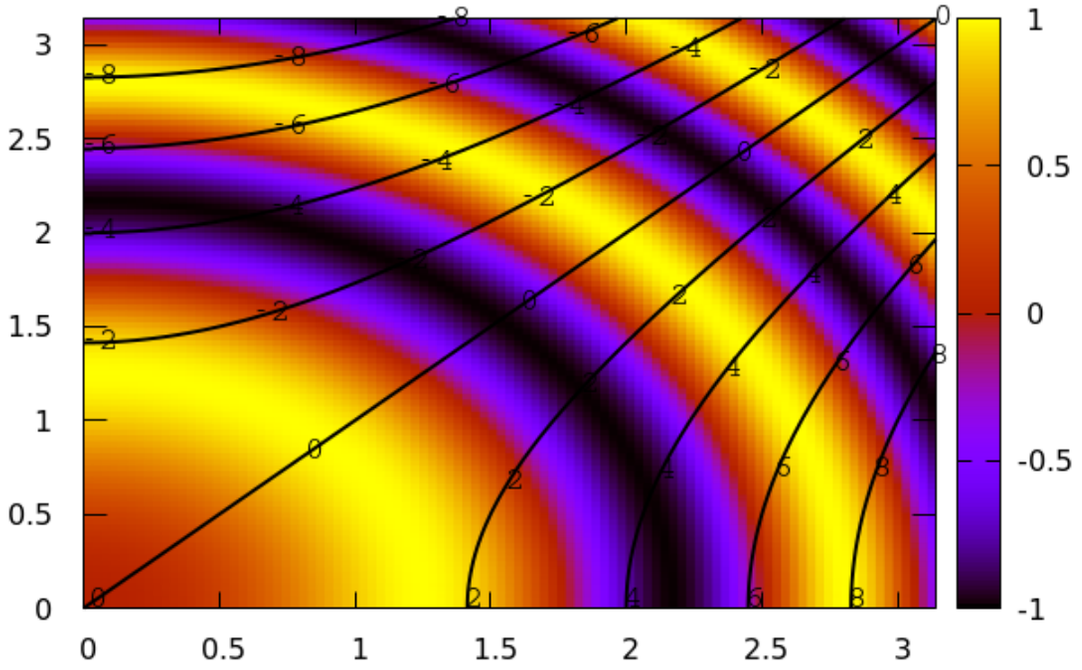
```
set xr [-1:1]; set yr[-1:1]
set view 50, 340
set iso 20; set samp 20
unset key
a = .2
splot "++" u 1:2:(cos($1**2+$2**2)) with pm3d,\
      "++" u 1:2:(cos($1**2+$2**2)):(-a*sin($1)*cos($2)):\
              (a*cos($1)*sin($2)):\
              (0) w vec size .02, 15 filled lc black
```

Open script



Combining Contour Plots and Heat Maps

You can put a contour plot and a heat map on the same graph. Why would you want to do such a thing? Sometimes people like to add contours to a heat map in order to allow the viewer to easily read off particular values; to show the data more precisely. Another reason is to visualize two related but different sets of data on the same graph. In the case of 2D plots, doing this is simple: we just plot any number of curves and identify them with labels or a legend. But with 3D plots, trying to interpret a graph containing two different sets of contours would be difficult, and plotting two heat maps simultaneously would be impossible. However, rendering one function or dataset as a heat map and the other one as a set of contours can work quite well. In this example, we exploit the optional fourth column of the `splot` command to define the colors for the heat map; the contours are taken from the third column. This requires us to use the "data" version of the `splot` command, through the "++" pseudofile. The colorbar gives the mapping from color to z-value, as usual, and we add some labels to identify the contours; this we can do using the "function" version of the `splot` command, as no columns are required. The highlighted word `onecolor`, which is an option to the `cntrlabel` command, tells gnuplot to plot all the contours the same color; the other highlighted bit, the `linewidth` and `linetype` specifiers, set the properties of the contours, allowing them to stand out against the colored background.




```
set xrange [0:pi]; set yrange [0:pi]; set iso 100; set samp
set cntrparam levels auto 10; set contour base
unset key; set view map
set cbr [-1 : 1]
set cntrlabel start 1 interval 25 font "Courier,14" onecolor
splot "++" using 1:2:($1**2-$2**2):(sin($1**2+$2**2)) with pm3d lt -1 lw 2,\
           x**2 - y**2 with labels
```

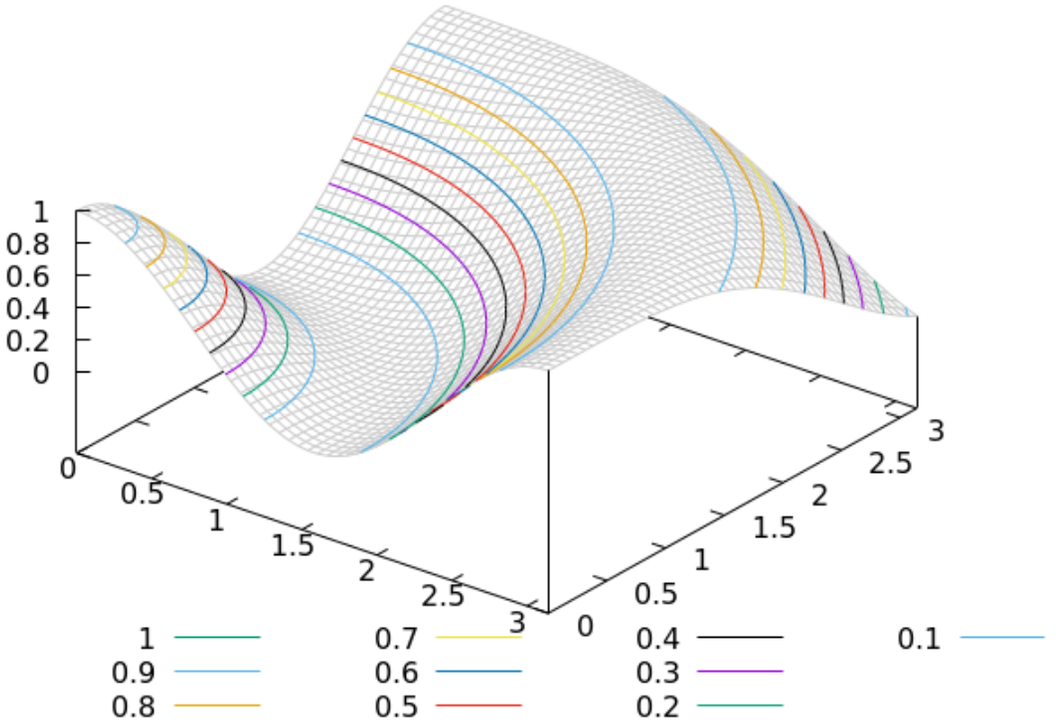
Open script

Contours with Surfaces

The command we used before to get contour plots, `set contour base`, suggests that there are other options. One of these is to put the contours on the surface. If you plot a mesh surface this way, you should draw the isolines in a light color, as we've done below, so that the contour lines are easy to see.

```
set xrange [0:pi]; set yrange [0:pi]
set iso 50; set samp 50
set ztics 0.2
set cntrparam levels auto 9
set key maxrow 3 bmargin
set contour surface
set view 43, 38
set hidden
splot cos(sqrt(x**2 + y**2))**2 lc "grey80" title ""
```

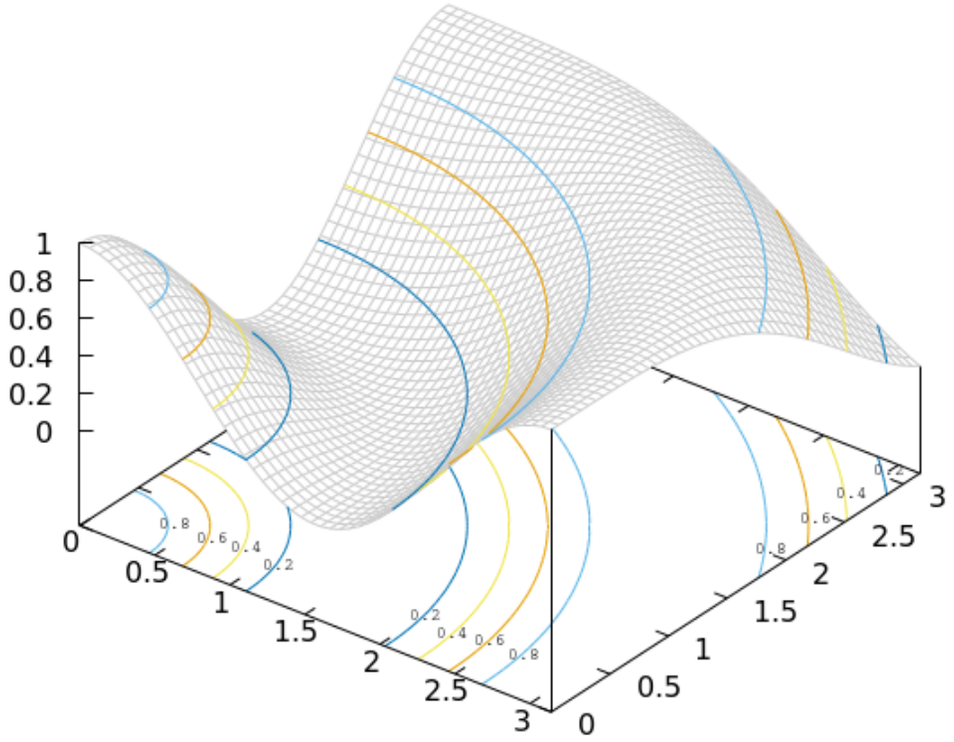
Open script



You can also display the contour lines on the base and surface at the same time (as well as only on the base when drawing a surface). Let's replot the previous example with contours on the surface and base, with labels (which are only drawn on the base).

```
set xrange [0:pi]; set yrange [0:pi]
set iso 50; set samp 50
set ztics 0.2
set cntrparam levels auto 5
set contour both
set cntrlabel start 5 interval 55 font "Courier,7"
set view 43, 38
unset key
set hidden
splot cos(sqrt(x**2 + y**2))**2 lc "grey80",\
      cos(sqrt(x**2 + y**2))**2 with labels
```

Open script

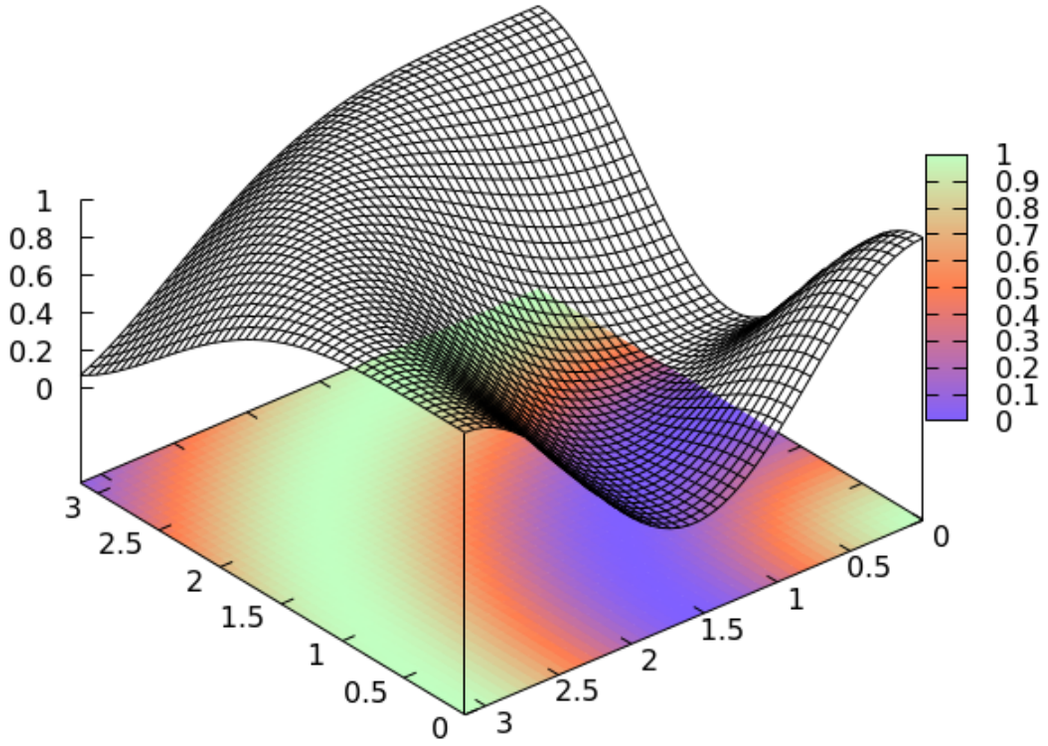


Heat Maps with Surfaces

Just as we can combine contour and surface plots, we can plot surfaces along with heat maps. We'll stick with the function we've been using in the last few examples:

```
set xrange [0:pi]; set yrange [0:pi]
set iso 50; set samp 50
set ztics 0.2
set view 43, 230
set pal def (0 "slateblue1", .5 "coral", 1 "seagreen")
unset key
set hidden front
splot cos(sqrt(x**2 + y**2))**2 lc "black",\
      cos(sqrt(x**2 + y**2))**2 with pm3d at b
```

Open script

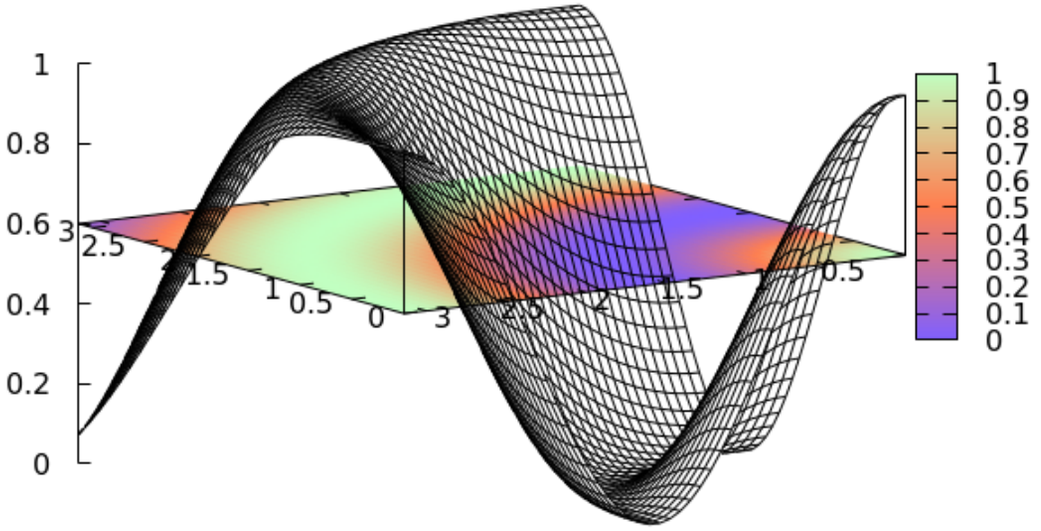


Intersecting Surfaces and Heat Maps

In all of our 3D plots so far, in this chapter and the previous one, we have allowed gnuplot to position the "base" of the plot in its default location, which is below the plotted surface. This is usually what you want. However, you can put the base, called the "xyplane" in gnuplot, anywhere you want. As before, `set hidden front` is essential to get a correct plot. Sometimes a good place for the xyplane is cutting right through the surface, as in this example:

```
set xrange [0:pi]; set yrange [0:pi]
set iso 50; set samp 50
set ztics 0.2
set view 75, 237
set pal def (0 "slateblue1", .5 "coral", 1 "seagreen")
unset key
set hidden front
set xyplane at 0.6
splot cos(sqrt(x**2 + y**2))**2 lc "black",\
      cos(sqrt(x**2 + y**2))**2 with pm3d at b
```

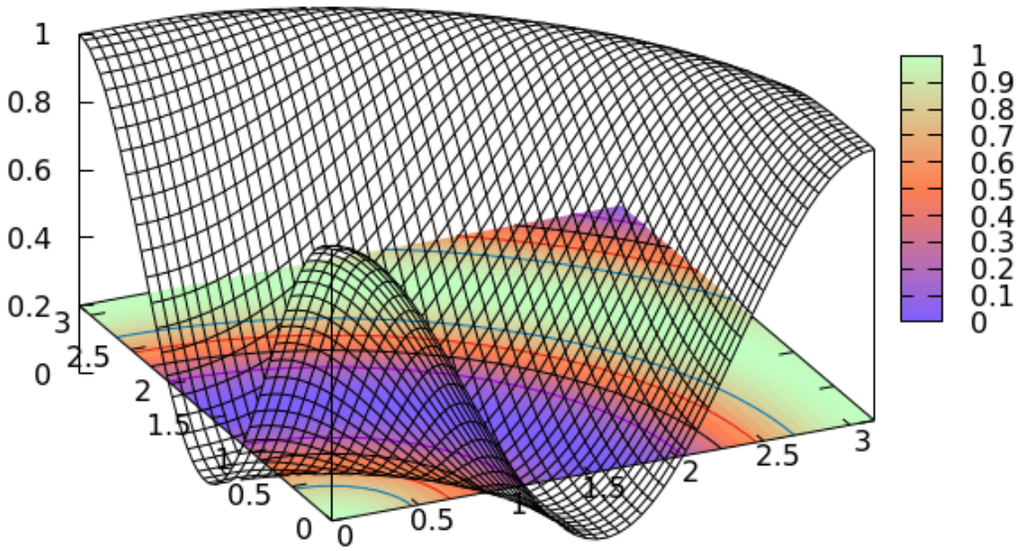
Open script



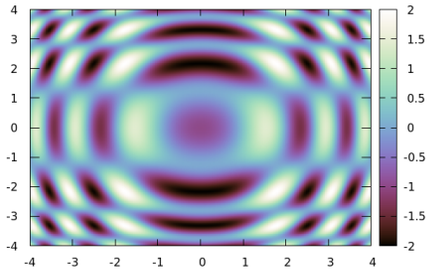
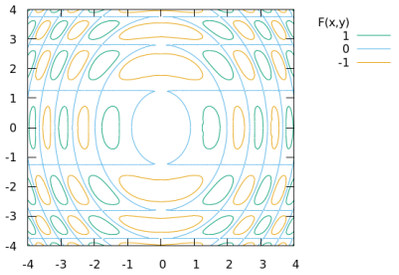
Of course, you can combine the elements in the previous example with contour lines, as here (you can also draw contours on the surface, in combination with all of this):

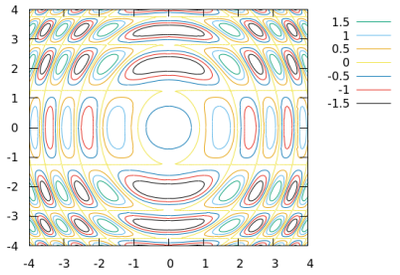
```
set xrange [0:pi]; set yrange [0:pi]
set iso 50; set samp 50
set ztics 0.2
set view 55, 335
set pal def (0 "slateblue1", .5 "coral", 1 "seagreen")
unset key
set hidden front
set contour base
set xyplane at 0.2
splot cos(sqrt(x**2 + y**2))**2 lc "black",\
      cos(sqrt(x**2 + y**2))**2 with pm3d at b
```

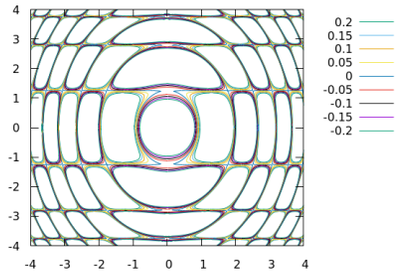
Open script

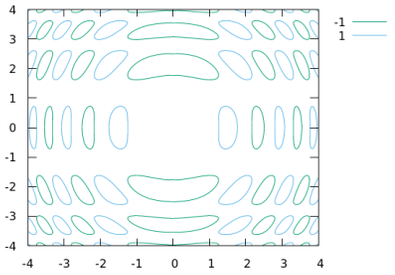


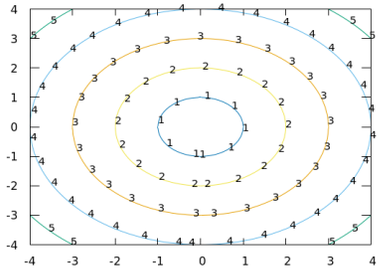
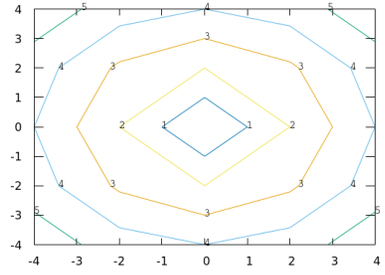
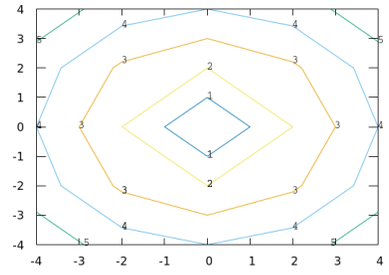
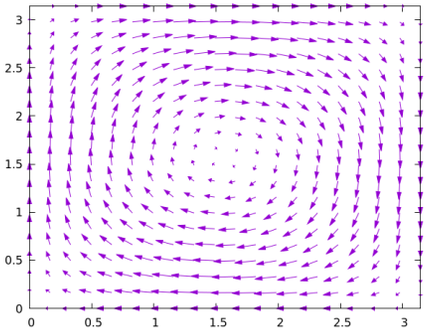
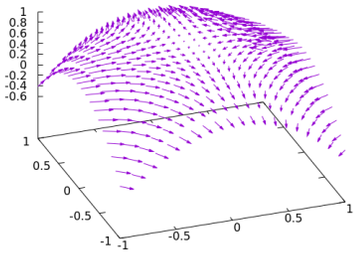
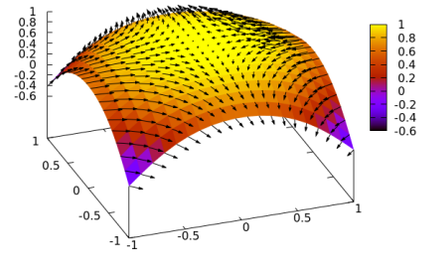
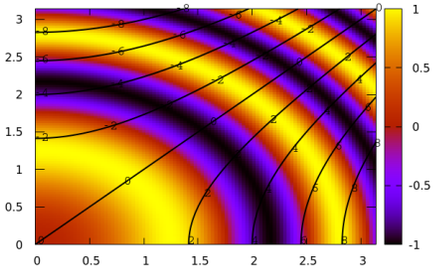
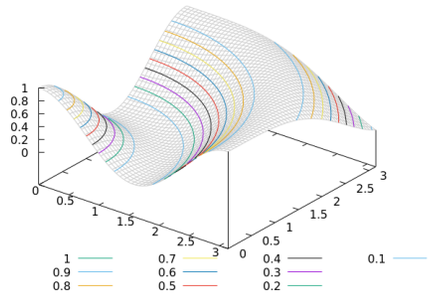
Index of Plots

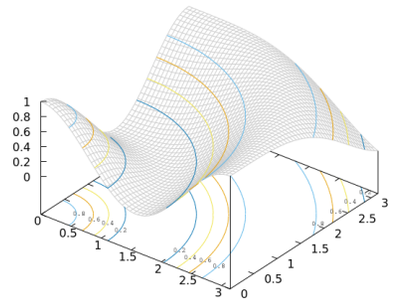
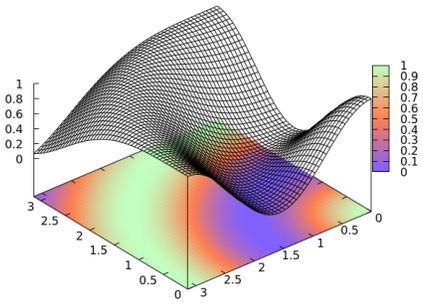



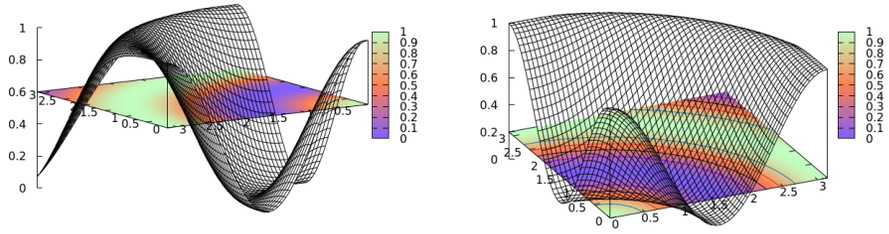







1
0.8
0.6
0.4
0.2
0
3
2.5
2
1.5
1
0.5
0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

1
0.8
0.6
0.4
0.2
0
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Index