Plotting in gnuplot 4.4

A Quick Guide

Stilianos Louca

November 3, 2011

Contents

0.1 General

0.1.1 Getting help

Use

```
help <help topic>
```

for example

```
help plot
help plot style
```

0.1.2 Data format

Data is usually passed by files containing the coordinates (space-separated) and other information for the points/functions to be plotted. Each line of the file corresponds to one point to be ploted. Example formats are (x), (x y), (x y z), (x y yerror), etc.
Note:

- For alternative Formats consider the **using** directive for data plots.
- Numbers can also be separated by ", " (note the space after the comma).
- Comment lines start with "#".

For multiple graphs, a blank line in a file is interpreted as 'lift the pen'. Two blank lines represent a new 'index', where 0 is the first index.

0.1.3 Multiple data columns & titling

Each data-row may contain multiple entries, representing multiple data columns. The first file-row may contain the titles for each column (uncommented!).
Example: Consider the data-file

```
Age Height  Weight
0 0.30    3
1 0.50    6
5 0.90    20
10  1.30    35
18  1.80    76
```

To plot two captioned graphs using columns 2 & 3 respectively:

```
#display line-captions under the plot
set key under
#plot the two graphs using titles in columns 2 & 3
plot "data" using 1:2 title 2, '' using 1:3 title 3
#Alternatively: plot the two graphs using custom titles
plot "data" using 1:2 title "Children Height", '' using 1:3 title "Children Weight"
```

Note: Including a title-row in the data file, requires use of the *using* command.

0.1.4 Setting output

```
#set output terminal to X11
set terminal x11
#set output terminal to AquaTerm
set terminal aqua
set terminal aqua size 400 400
#set output terminal to LaTeX
set terminal latex
#redirect output to file
set output "outputFile.latex"
```

```
#unset redirection
unset output
```

0.1.5 Loading a command file

```
load "foo"
```

0.1.6 Setting plot resolution

Sets number of isolines in x- and y-axis.

```
set isosamples 75, 75
```

Note: Isolines are used to create surfaces of functions. This setting does not affect data file plotting.
Sets sampling rate:

```
set samples 100,100
```

Note: Affects function plots, and data plots where one of the interpolation/approximation options are used.

0.1.7 Defining variables & functions

```
a = 3
#show all defined variables
show variables

f(x) = x**2 - sin(x)
f(x,y) = 3*x - tan(y/x)
#show all user defined functions
show functions
```

0.1.8 Tics & Axis

Adjusting x,y,z axis-tics:

```
#set major x-tic-step
set xtics 100
#set minor-intervall count between major tics
set mxtics 5
#explicitly set x-tics at 0,5,10
set xtics (0,5,10)
#explicitly set tics with labels
set xtics ("pi" 3.14, "e" 2.72, "origin" 0)

#shift xtics labels two chars to the left and one char down from the default position
set xtics offset -2,-1

#disable major x-tics
unset xtics
#disable minor x-tics
unset mxtics

#set number-printing format (as in C-format specifiers for double)
set format "%3.3f"
#set number-format just for x-axis
set format x "%g"

#create a grid using major tics
set grid
unset grid

#set z-axis origin in xy-plane
set ticslevel 0
```

Setting/unsetting logarithmic scales:

```
#set log scales
set log x
set log xy
set log xyz

#unset log scale
unset log xy
```

Note: See also 0.1.9 on scaling.

0.1.9 Scaling & ranges

Setting/unsetting autoscale:

```
set autoscale
set autoscale x
unset autoscale
unset autoscale x

#set x-range of displayed values
set xrange [-10:10]

#set xrange from 0 to auto-max
set xrange [0:*]

#set completely automatic xrange
set xrange [*:*]
```

0.1.10 Title & Labels

To set an arbitrary label:

```
#add a label to position (x,y,z) = (0,1,0)
set label "Milk production" at 0,1,0
#add a label with the tag 2. Tag is used to identify the label in the future. Recommended!
set label 2 "Cows" at 2,0,0

#rotate label
set label 2 "Cows rotated" at 2,2,2 rotate right #rotate label to the right
set label 2 "Cows rotatet" at 2,2,2 rotate by 90 left
set label 2 "Cows rotatet" at 2,2,2 rotate by 45 point ps 2

#show all labels set
show label
#remove label with tag 2.
unset label 2
#change label with tag 2, sets justification to right (with respect to x,y,z coordinates)
set label 2 "Cows at farm" at 2,0,0 justification right
```

Setting titles and keys (legend):

```
#Setting title
set title "Cow production" #For multiline titles use "\n"

#setting key-labels to plot
set key
#disabling key-labels
unset key
#setting key-labels to display under the plot
set key under
#set keys on upper left corner
```

```
set key left top

#print plot time on output
set time
#disable time printing
unset time
```

Plots are keyed by default using the data-file loaded or the function used. To explicitly title plots, use **title** as

```
#Set keys for first 2 plots as "Sinus" & "Loaded data", dont key the last plot
plot sin(x) title "Sinus", 'data.csv' title "Loaded data", tan(x) title ''
```

Set axis-specific labels:

```
set xlabel "X-axis"
#move label 2 places to the right and 3 places down
set xlabel offset 2,-3
```

0.2 Greek characters

To display greek characters in labels and titles, your terminal must support them. Example terminals are `aqua enhanced`, `x11 enhanced` or `postscript eps enhanced`. Greek characters are represented by special codes of the format `{/Symbol a}`, `{ /Symbol b}` .., corresponding to the symbols α, β and so on. For more details see http://t16web.lanl.gov/Kawano/gnuplot/label-e.html.

0.3 Fitting functions

Define function template:

```
f(x) = a*tanh(x/b)
```

Initial guess for the parameters **a,b**:

```
a = 1; b=3
```

Fit to data:

```
fit f(x) 'data' using 1:2 via a,b
```

0.4 2D plotting using plot

0.4.1 Function plotting

```
plot sin(x)
plot sin(x) title "The sinus function"

#plot y=x^2 from x=0 to x=10
plot [0:10] x**2

#plot function f within [0,10]x[0,1] using dots
plot [0:10] [0:1] f(x) with dots

#plot using steps
plot sin(x) with steps
```

Note: To apply any changes to an existing plot call

```
replot
```

0.4.2 Data plotting

```
plot "data"
plot "data" with lines
plot "data" with linespoints

#connects points with smoothed out lines
plot "data" smooth unique

#connect points with smoothed out lines. very strong "smooth out"
plot "data" smooth csplines
```

0.4.3 Point style

```
set pointsize 3
plot sin(x) with points linecolor 3 linewidth 3
plot sin(x) with lines linecolor 2 linewidth 2 pointtype 7
```

Note:

- **lc** or **linecolor** defines line/point color, by -1=default, 1=red, 2=green, 3=blue, 4=purple, 5=aqua, 6=brown, 7=orange, 8=light-brown (x11 & aqua)
  See also: Section 0.5.10 on *Plot color*
- **lw** or **linewidth** defines line/point-line width.
- **pt** or **pointtype** defines point-style (i.e 6 for empty circle, 7 for full circle, 15 for x-cross).

Tip: Use **test** to see various options:

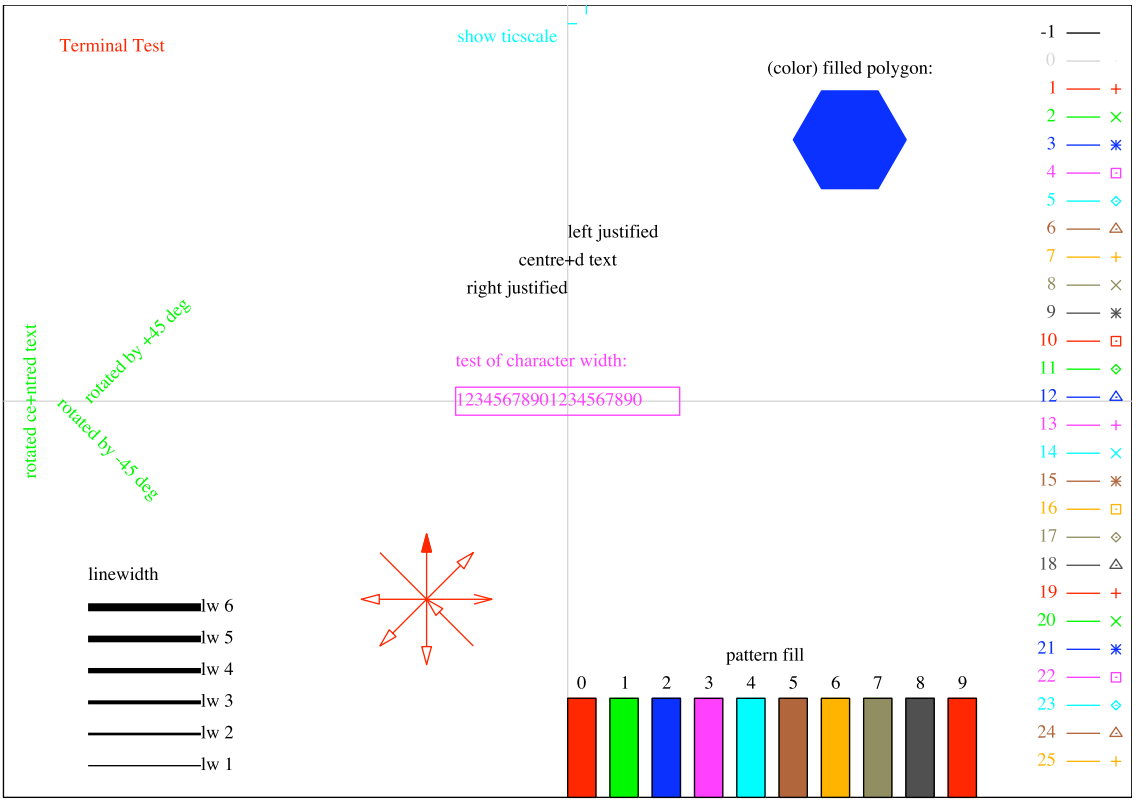


Figure 1: Pointstyle in aquaterm

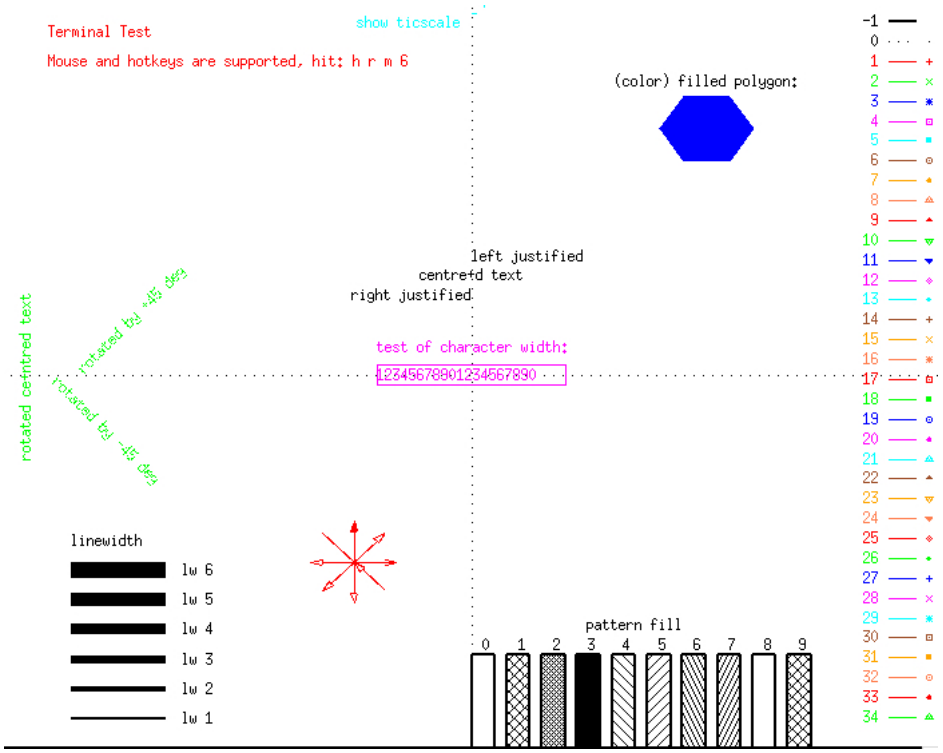


Figure 2: Pointstyle in xterm

0.4.4 Dashed lines

In many terminals it is possible to plot using different line styles (e.g. dashed, continuous etc). This behavior is enabled using the **dashed** flag when setting the output terminal, e.g.

```
set term aqua dashed
```

Plot linetype is set using the **linetype** or **lt** keyword, e.g.

```
plot sin(x) linetype 3
```

Note that merely setting the linetype may also affect the line color/width. Use **lc** and **lw** explicitly to avoid this behavior:, e.g.:

```
plot sin(x) lt 3 lc 1 lw 1
```

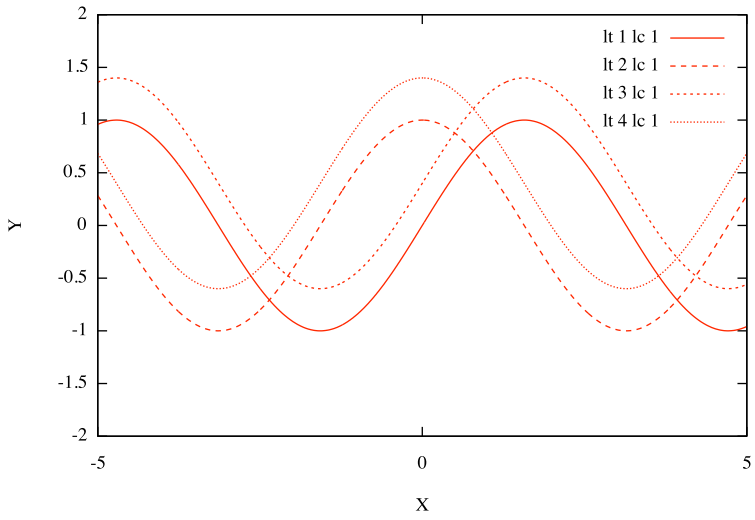


Figure 3: Example plot using various line-types in aquaterm.

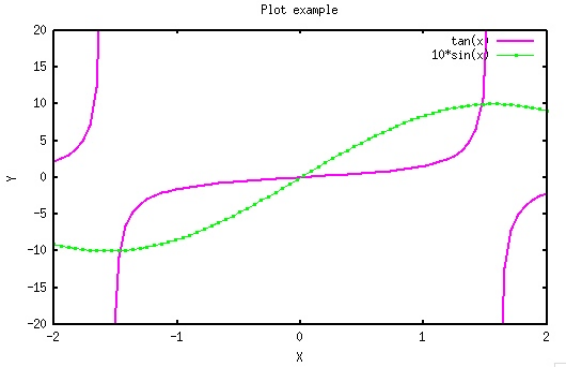
0.4.5 Plotting multiple sets

```
plot "data1" , "data2"
plot [0:2] "data1" with lines lw 2 title="Data 1", "data2" with points pt 5

#plot different data-sets from same file, as spline-interpolation and line-points respectively
plot "data1" using 1:2 smooth unique, '' using 1:3 with linepoints

#plot only indices from 4 to 7
plot "data" index 4:7
```

Example: The code

```
#axis settings
set samples 70
set xrange [-2:2]
set yrange [-20:20]

#labels
set title "Plot example"
set xlabel "X"
set ylabel "Y"

#plot
plot tan(x) with lines smooth cspline linetype 4 linewidth 2, 10*sin(x) with linespoints
    pointtype 6 pointsize 2
```

produces the plot

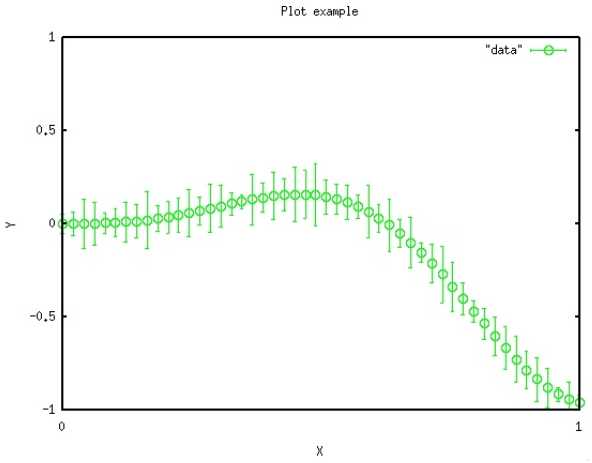


0.4.6 Error bars

Use the **with errorbars** directive to plot simple (x,y) points with an yerror at each point. The code

```
plot "data" with errorbars pointsize 2 pointtype 6 linetype 2
```

produces

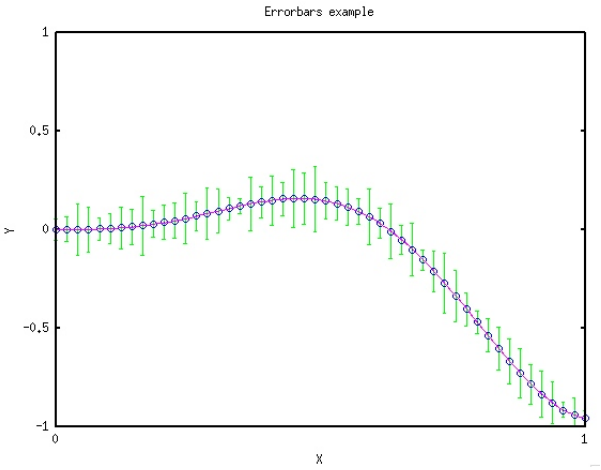


Note: Data was of format "x y yerror".

To handle/customize error-bars, points and connecting lines separately, use separate plot calls. The code

```
plot "data" with errorbars linecolor 2 pointsize 0,
  "data" with points linecolor 3 pointtype 6,
  "data" with lines linecolor 4 smooth cspline
```

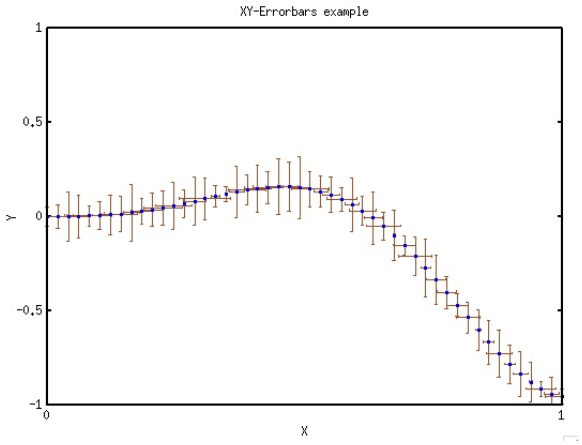
outputs



To draw x & y errorbars use the **with xyerrorbars** directive. The code

```
plot "data" with xyerrorbars linetype 6 pointsize 0,
  "data" pointtype 6 pointsize 1 linecolor 3
```

outputs



Note: Data format was "x y xerror yerror".

0.4.7 Histograms

Each column in the data file corresponds to a "category", each row corresponds to an interval. Data entries correspond to heights of histogram bars (bin sizes). The names for the intervals (bins) can be given in a separate column and used applying the **xticlabels** keyword to **using**.
Example: The code

```
set style data histogram
#leave a gap between intervals equivalent to the width of one bar
set style histogram clustered gap 1
#fill bars
set style fill solid border -1

#set axis labels
set xlabel "Year"
set ylabel "Population density (persons per square kilometer)"
#set histogram title
set title "Development of population density"
#set displayed y-range
set yrange [0:300]

#plot bars for two categories (males & females).
#Names for the intervals are taken from the first column
plot "data" using 2:xticlabels(1) title "Males", "" using 3 title "Females"
```

produces with the data

```
#Year Males Females
1991 100   102
1992 90    100
1993 80    115
1994 75    125
1995 25    20
```

the plot shown in fig. (4).

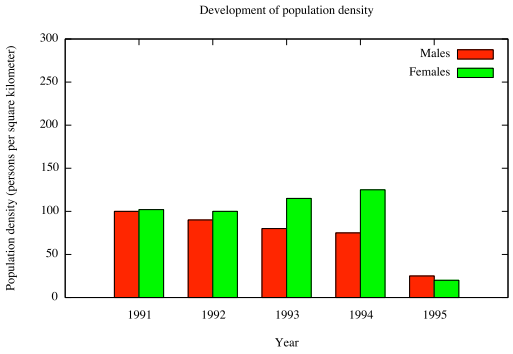


Figure 4: Example histogram using the **histogram clustered** style in gnuplot.

Using the same data file, the code

```
set style data histogram
#stack histograms of different categories
set style histogram rowstacked
#fill bars
set style fill solid border -1
#Leave a gap between histograms
set boxwidth 0.5

#set axis labels
set xlabel "Year"
set ylabel "Population density (persons per square kilometer)"
#set histogram title
set title "Development of population density"
#set displayed y-range
set yrange [0:300]

#plot stacked bars for two categories (males & females)
plot "data"  using 2:xticlabels(1) title "Males", ""  using 3 title "Females"
```

produces the plot shown in fig. (5).

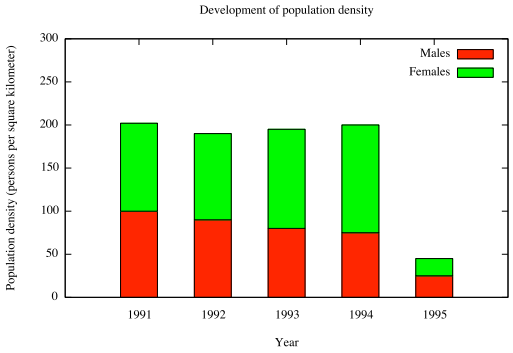


Figure 5: Example histogram using the **histogram rowstacked** style in gnuplot.

0.5 3D plotting using splot

0.5.1 Function plotting

```
#plot isolines within box [0,1]^3
splot [0:1] [0:1] [0:1] tan(x*y) title "Combined tangents"

#plot isolines with non-transparent feel (3D-hiding)
set hidden3d
splot tan(x*y)

#plot the surface defined by the function's isolines.
splot tan(x*y) with pm3d
```

Notes:

- Use 'set isosamples' and 'set samples' to increase resolution (see 0.1.6).
- Uset 'set hidden3d' to give a more 3D-hiding look to plots (applies to non-*pm3d* plots).

Example: The code

```
#set axis numbering
set ztics 20
set mztics 5
set xtics 2
set ytics 2
set grid #create grid with major x,y tics

#set resolution
set isosamples 100,100
set samples 100,100

#set axis-labels
set xlabel "X"
set ylabel "Y"
set zlabel "F"

#plot
plot [-5:5] [-5:5] [-50:50] sin(y)*x**2 with pm3d
```

produces the plot in Fig. (6).

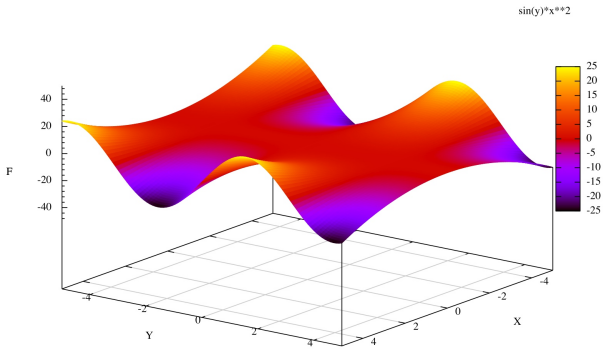


Figure 6: *pm3d*-plot of a function with **splot**.

The example codes

```
#using hidden3d
set hidden3d
splot [-3:3] [-3:3] sin(x)*cos(y)

#not using hidden3d
unset hidden3d
splot [-3:3] [-3:3] sin(x)*cos(y)
```

produce the plots seen in Fig. (7).

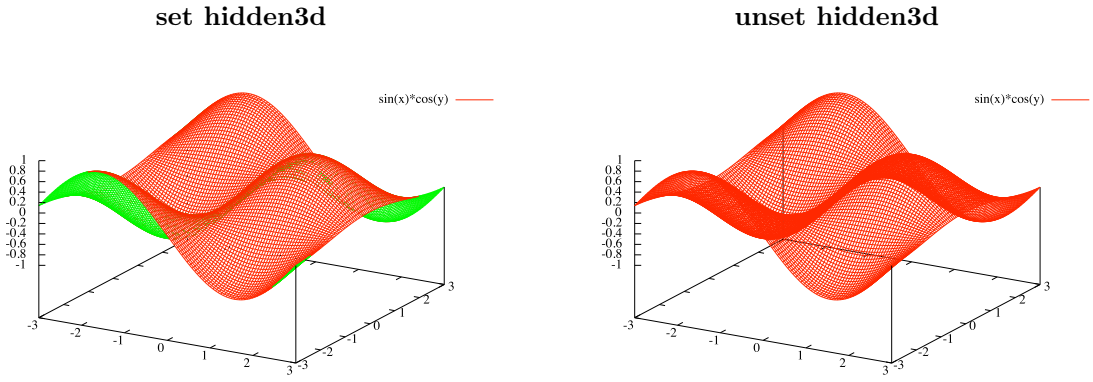


Figure 7: Non-*pm3d* function-plots with 'splot' using 'set hidden3d' (left) and without (right).

0.5.2 Simple point-plotting

Points are stored within file as x,y,z-columns, each row representing one point and plotted using the 'splot' command. Note that more columns can be used for further plotting-information (see below 0.5.9 and 0.5.10). Example code:

```
#plot columns 1,2,3 as (x,y,z) points
splot "data"

#set point-type to cross
splot "data" pointtype 1

#plot points in data.txt, by plotting 3rd column over the 1st
splot "data" using 1:3

#plot tan(x+y) over x,y (defined in columns 1 & 2)
splot [0:1] [0:1] [0:1] "data" using 1:2:tan($1+$2)

#plot points and connect with lines
splot "data" with linespoints
```

Example: The code

```
#data consists of 3 columns (x,y,z coordinates) of random points
splot "data" pointtype 2 linetype 3

#data contains subsequent points of charged particle moving in a magnetic field
splot "data" with linespoints pointtype 2
```

produces the plots seen in Fig. (8).

simple **with linespoints**

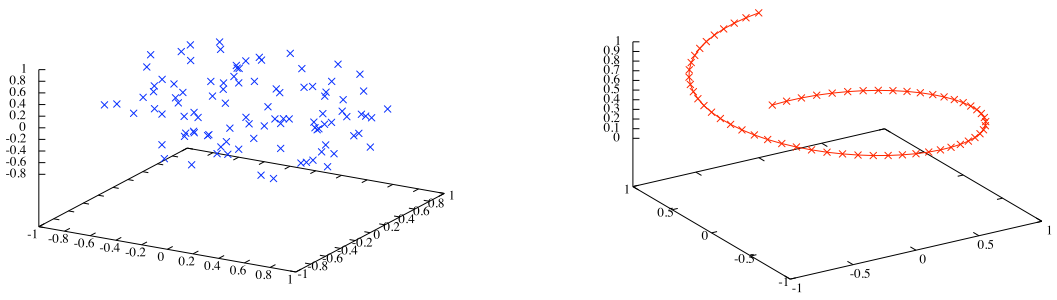


Figure 8: Simple point-plotting using 'splot'.

0.5.3 Surface plotting using sampled data

2D-Surfaces can be defined using 3-column data in the following format: Each row represents one point (x, y, z) of the surface. Data is split into equally size row-*chunks*, separated among each other by single empty lines. Each such chunk represents one *sampling line*, that is, a line crossing along the surface. Eventually, these lines are connected point by point (hence the demand they be equally sized) to built a surface in 3D-space. Use the **splot** command with the directive **with lines**, **with linespoints** or **with pm3d** to evaluate such data.
Exemple: The data file

```
#X Y Z
0 0 0
0 1 1
0 2 4
0 3 9
0 4 16

1 0 1
1 1 2
1 2 5
1 6 10
1 7 17

2 0 4
2.2 1 5
2.3 2 8
2.4 3 13
2.5 4 20

3 0 3
3 1 10
3 2 13
3 3 18
3 4 25
```

and the command

```
splot "data" with lines
```

produce the surface seen in figure (9).

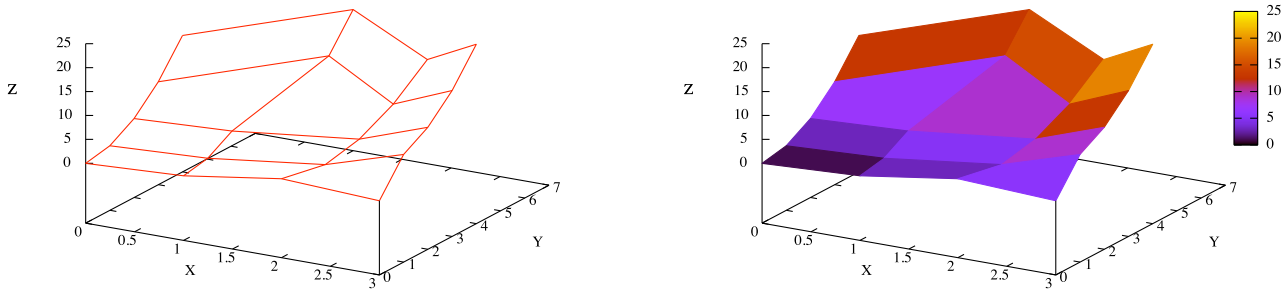


Figure 9: Surface plotted using splot, defined by gridded data. On the right, the directive **with lines** is replaced by **with pm3d**.

0.5.4 Surface plotting from scattered data

Scattered 3D-Data can be used to build a grid and define a connected 2D-surface, using the 'set dgrid3d' directive (use 'help dgrid3d' for more information).

```
#enable the dgrid3d plotting mode
set dgrid3d

#grid consists of 20 x-divisions and 50 y-divisions
set dgrid3d 50, 20 #notice the reversed order!
```

Note

- Since *dgrid3d* merely interpolates the data on a newly created grid, points need to be plotted either 'with lines' or 'with pm3d' to obtain a *surface look*.
- The closer the scatter points to a grid point, the more effect they have on that grid point. The exact norm used $(L^2, L^4, ..)$ is determined by the 3rd argument of set **dgrid3d**, e.g.

  ```
  set dgrid3d 50,20,4
  ```

  sets the norm to be L^4. In that sense, **dgrid3d** acts as a low-pass for the scattered data.
- Usage of 'set hidden3d' is also possible.

Example: The code

```
set hidden3d
set dgrid3d 50,20
splot "data" with lines #also possible: with linespoints

set dgrid3d 50,50
splot "data" with pm3d
```

produces the plots in Fig. (10).

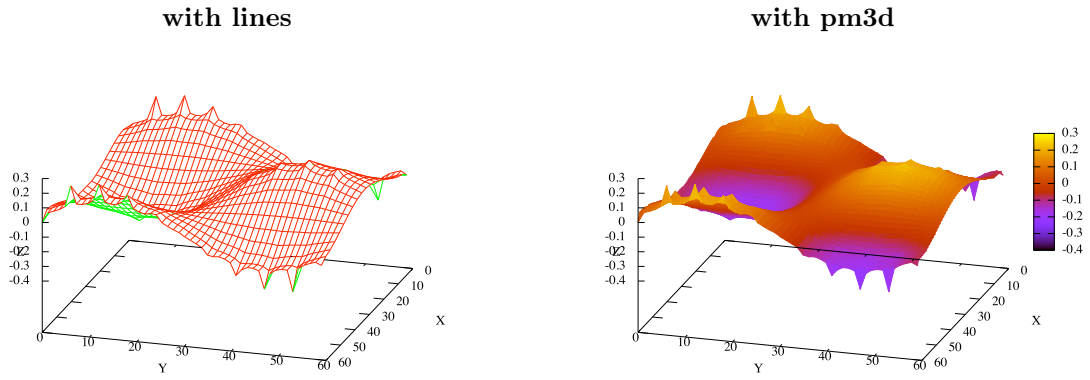


Figure 10: Surface data-plot using 'splot' in *dgrid3d* mode.

0.5.5 2D-Color maps of 3D scattered data

Using **dgrid3d** and **pm3d** one can create 2D-color maps of scattered data (x, y, z), with color corresponding to the 3rd data column. A grid is generated using **dgrid3d** and data is interpolated on the grid. Subsequently, this gridded data is used by **pm3d** to calculate a colored map of the surface.
Exemple: Using a data file containing 2 columns of randomly distributed numbers, the code

```
#enable scattered data to grid transformation
set dgrid3d 40,40,2
#switch to color map view
set pm3d map
#Algorithm for coloring grid boxes
#here, the mean value of each 4 corners of a box is taken
set corners2color mean
#set color palette
set palette gray
#plot
splot "data" using 1:2:(sin(20*sqrt($1**2+$2**2)))
```

produces the plots in Fig. (11).

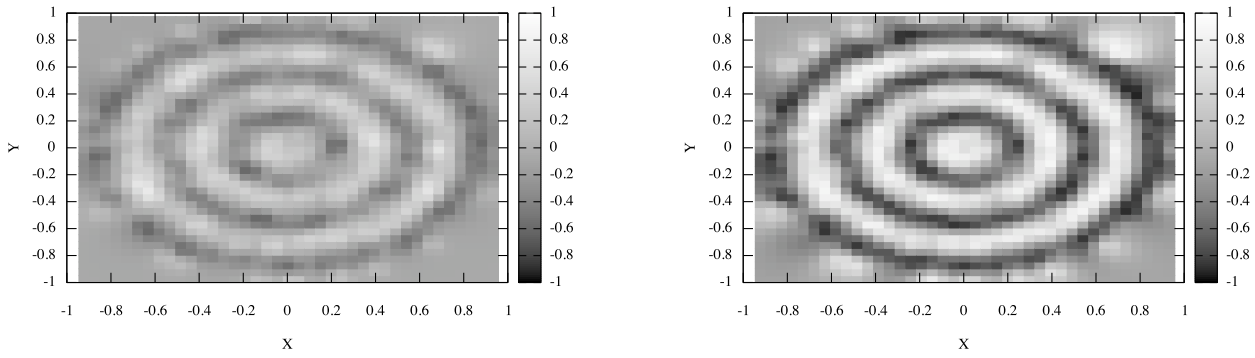


Figure 11: Color-map data pot using **dgrid3d** and **pm3d**, on the left using the L^2- and on the right using the L^4-norm..

0.5.6 2D-Color maps of data on an XY-grid

Suppose the data given as z-values on a fixed, 2-dimensional grid (matrix). One can then create a color map of that grid, using **pm3d**. If data consists of $r \times c$ matrix, a $(r-1) \times (c-1)$ grid is drawn. The color of each box is determined from the values of the 4 surrounding corners. From the gnuplot manual:

```
Syntax (the options can be given in any order):
  set pm3d {
    { at <bst combination> }
    { interpolate <steps in scan>,<steps between scans> }
    { scansautomatic | scansforward | scansbackward | depthorder }
    { flush { begin | center | end } }
    { ftriangles | noftriangles }
    { clip1in | clip4in }
    { corners2color { mean|geomean|median|min|max|c1|c2|c3|c4 } }
    { hidden3d <linestyle> | nohidden3d }
    { implicit | explicit }
    { map }
  }
```

Example: Using the data file

```
0 1 4 9
1 2 5 10
4 5 8 13
9 10 13 18
16 17 20 25
25 26 29 0
```

and the code

```
#switch to color map view
set pm3d map
#set color palette
set palette gray
#set rule for color calculating of a box
set pm3d corners2color mean
#adjust tics/range (grid is indexed by row & column number by default)
set xtics ("before" 0, "now" 1.5, "later" 3)
set ytics ("sea" 0, "land" 1.5, "sky" 3,"moon" 4.5)
#plot
splot "data" matrix with pm3d
```

produces the left plot in Fig. (12).

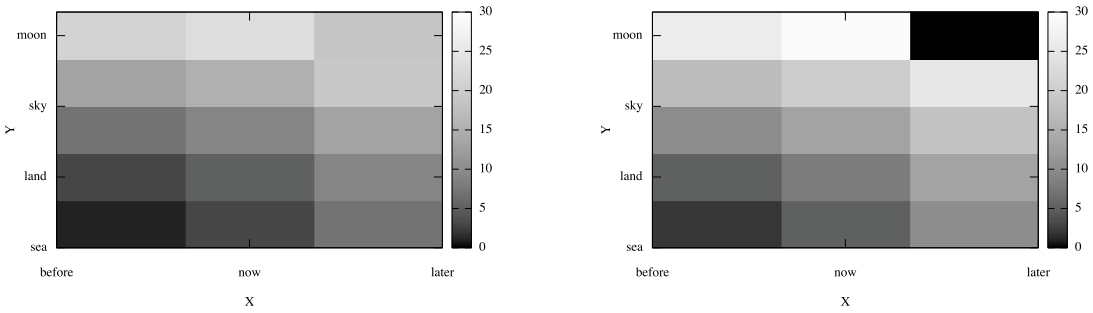


Figure 12: Color-map data plot with **pm3d** using (6×4) matrix-data. On the left using **corners2color mean**, on the right using **corners2color c4**. Here, increasing row number corresponds to increasing Y-value, with c4 (c3) representing the upper right (upper left) corner of the box.

Note how each box in the color map corresponds to a block *between* data points. In order to have a color map whose blocks really correspond to its matrix entries, one needs to add a *dummy row & column*, using a one-corner-algorithm to calculate box-color. In the above example, modify the data file to

```
0  0  0  0  0
0  0  1  4  9
0  1  2  5  10
0  4  5  8  13
0  9  10  13  18
0  16  17  20  25
0  10  26  29  0
```

and use

```
#switch to color map view
set pm3d map
#set color palette
set palette gray
#set rule for color calculating of a box
set pm3d corners2color c4
#plot matrix data & adjust axes numbering so that box centre corresponds to original row-column
splot [0.5:4.5] [0.5:6.5] "data" index 2 using ($1+0.5):($2+0.5):3 matrix with pm3d
```

to produce the map in figure (13).

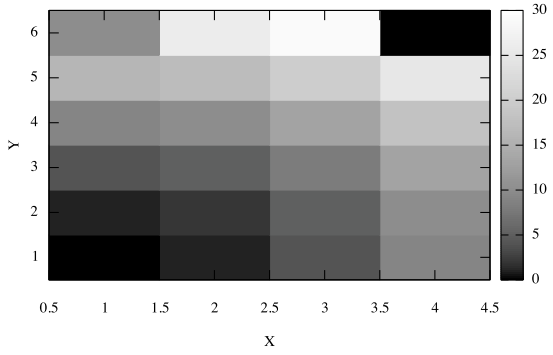


Figure 13: Color-map data plot with **pm3d** using (6×4) matrix-data, extended by one dummy-row and dummy-column. Each box corresponds to one original matrix entry. The upper-right corner of the map corresponds to lower-right entry in the data matrix.

Alternatively, one can use the directive **with image** with the **plot** command to directly plot matrix data as a bitmap, whose *pixels* correspond one-to-one to the matrix entries. The value-color mapping is still determined by **set palette** as above. As x- and y-axis, matrix-indices are used, with $(0, 0)$ corresponding to the upper-left entry in the data file and lower-left pixel-center in the plotted image. The advantage of this plot style, compared to the **splot**/**pm3d** method above, is that truly each matrix entry corresponds to one colored box without the need of dummy rows and columns.
Example: Processing the data file

```
0  1  4  9
1  2  5  10
4  5  8  13
9  10  13  18
16  17  20  25
25  26  29  0
```

with the commands

```
set title "Exemple matrix plot with image"
set palette grey
plot "data" matrix with image
```

or alternatively

```
set title "Exemple matrix plot with image"
set palette grey
plot "in" using (100*$1):(0.01*$2):($3/(1+$3))  matrix with image
```

gives the plots seen in figure (14) on the left and right respectively. Note how one has access to the matrix values via $3, to the images x- and y-coordinates (matrix columns & rows) via $1 and $2.

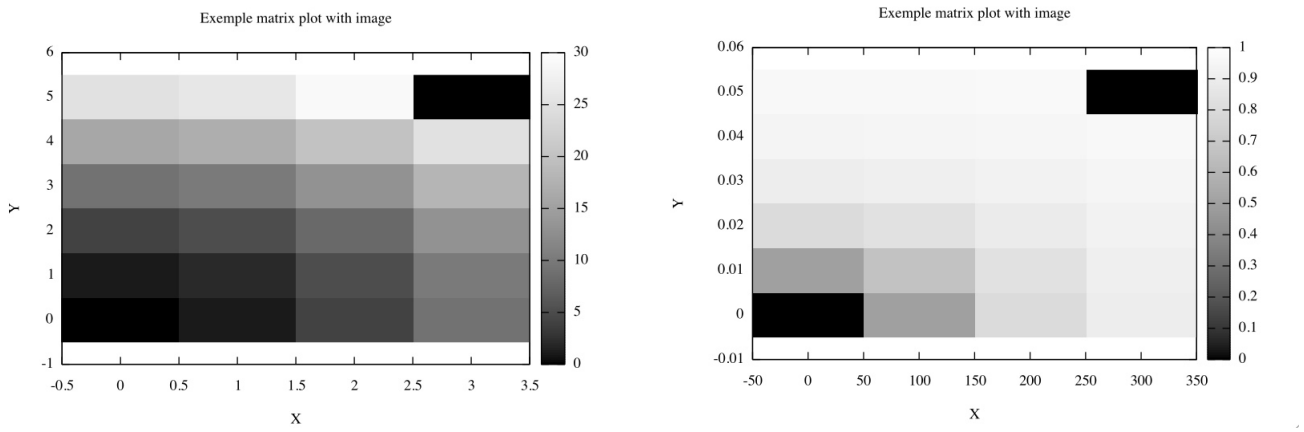


Figure 14: Color map data plots of a 6×4 data matrix using **plot matrix with image**. Each matrix entry corresponds to one image *pixel*.

Tips:

- To adjust the image frame to the actual area covered, simply explicitly set the range of the plot.
- To hide the color-mapping box on the right of the plot, precede the plot command with **unset colorbox**.

0.5.7 2D color maps using sampled data

Similarly to section 0.5.3, one can also produce color maps of sampled data, whose samples are of equal length. Using the data file from 0.5.3 and the commands

```
#go over to projected view
set view map
#enable contours
set contour
#plot color-projection of surface
splot "data" with pm3d
```

one receives the plot in figure (15).

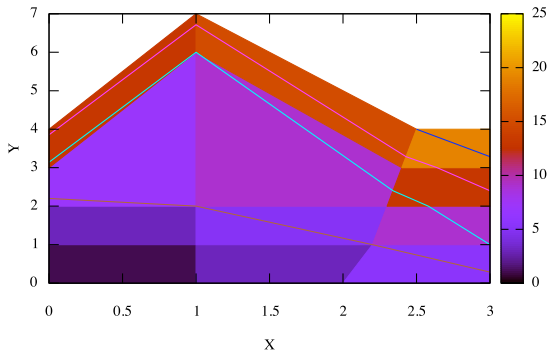


Figure 15: Color map of surface defined using sampled data.

0.5.8 Contours

Gnuplot can easily add contour-lines to 3D-Plots using the 'set contours' directive.

- 'set contours surface' places contours directly on your surface. 'set contours base' places contours at the XY-Base. 'set contours both' does both.
- Control contour settings with the 'set cntrparam (..)' directive.
- Set contour levels using 'set cntrparam levels (..)'.
- Works on *pm3d* & non-*pm3d* plots, both data (surface) and function plots.

Example code:

```
#enable contours
set contours

#set interpolation type for contours
set cntrparam bspline #other options include linear, cubicspline

#set automatic contour levels
set cntrparam levels auto

#set contour levels at specific values
set cntrparam levels discrete -1, -0.5, 0, 0.5, 1

#set contour levels starting at -2, incrementing by 0.5 up to 2
set cntrparam levels incremental -2, 0.5, 2
```

The example code

```
set hidden3d
set contours base #or surface
set cntrparam bspline
set cntrparam levels auto
splot [-3:3] [-3:3] sin(x)*cos(y)
```

produces the plots seen in Fig. (16).

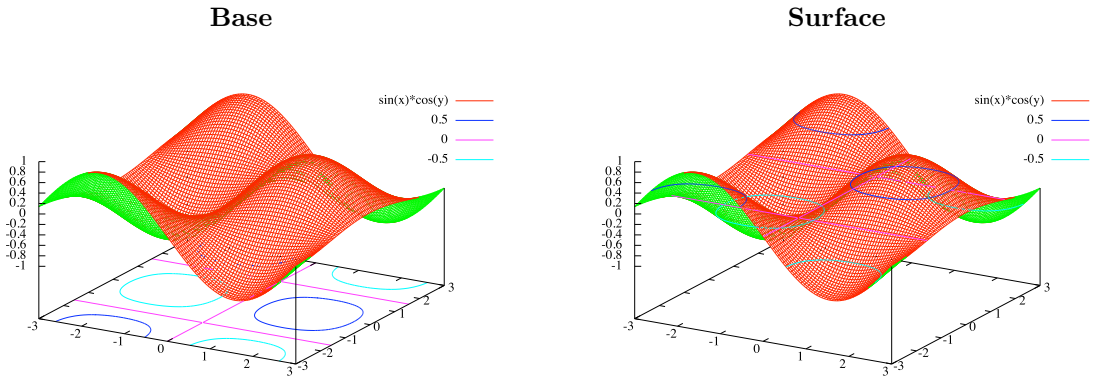


Figure 16: Contour-placement using 'set contours base' (left) and 'set contour surface' (right).

0.5.9 Point style

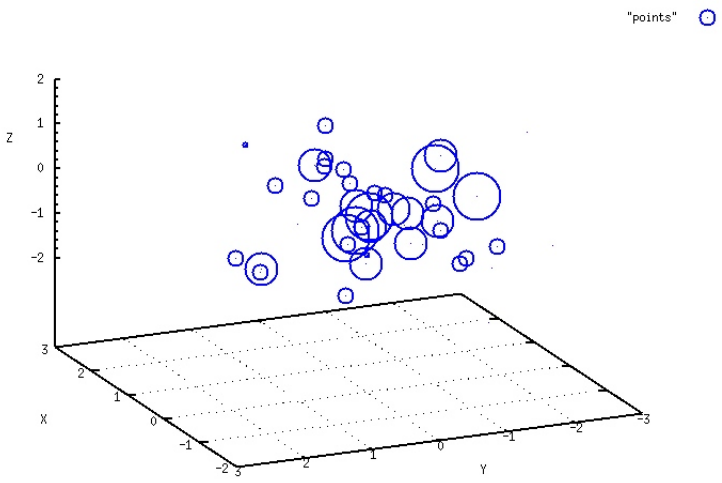
```
#plot as 1px dots
splot "data" with points pointtype 0

#plot as small empty circles
splot "data" with points pointtype 6

#plot as full circles with radius 2
splot "data" with points pointtype 7 pointsize 2

#plot as x-crosses of variable size, as defined in column 4
splot "data" with points pointtype 15 pointsize variable

#set line color to 4(purple) and width 3
splot sin(x)/y with lines lt 4 lw 3
```

See also 0.4.3.

Example: The code

```
splot [] [] [-2:2] "points" with points pointtype 6 pointsize variable linetype 3 linewidth 2
```

creates the plot



0.5.10 Plott color

Colors are defined using the **rgb** and **palette** keywords in the syntax

```
rgbcolor "colorname"
rgbcolor "#RRGGBB"
rgbcolor variable
palette frac <val>  # <val> runs from 0 to 1
palette cb <value>  # <val> lies within cbrange
palette z
```

Examples:

```
plot sin(x) linecolor rgb "green"
plot "data" linecolor palette 0.5
splot "data" lc rgb "#FF00FF" #violet
```

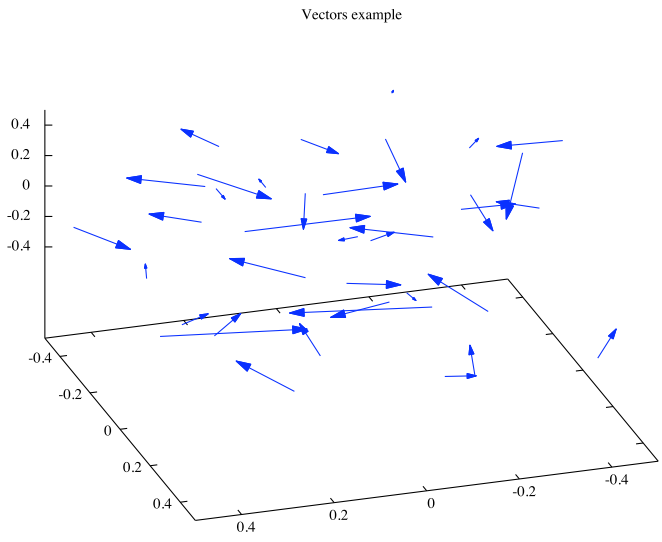
0.5.11 Plotting vectors

```
#plot x,y pairs and append vector from (x,y) to (x+dx,y+dy)
#uses 4 data-columns
plot "data" with vectors

#similar to above, uses 6 data-columns
#only works with cartesian mappings
splot "data" with vectors

#fill arrow heads
splot "data" with vectors filled
#empty arrow heads
splot "data" with vectors empty
#double-headed
splot "data" with vectors heads
#no heads
splot "data" nohead
#set vector color from palette
splot "data" with vectors linecolor palette frac 0.5
```

The code

```
splot "out" with vectors filled  lt 3
```

produces the plot



0.5.12 Variable size & color

```
#plot x,y,z points using 4th column as color
#colors should be number of format: 65536r + 256b+ g
splot "data" lc rgb variable

#plot x,y,z points using 4th column as size
splot "data" ps var

#plot x,y,z points using 4th column as size, 5th as color
splot "data" ps var lc rgb variable
```

The code

```
#define rgb function for conveniently calculating colors
rgb(r,g,b) = int(r)*65536 + int(g)*256 + int(b)

#plot data, using 4th column for size, and 5,6,7th columns for RGB color-values
splot "data" using 1:2:3:4:(rgb($5,$6,$7)) pt 7 pointsize var linecolor rgb variable
```

produces the plot seen in Fig. (17).

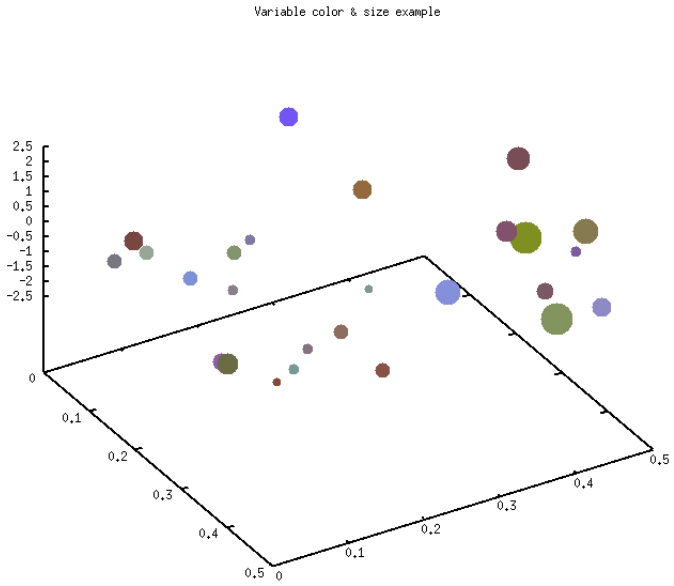


Figure 17: Points-splot with variable size and color.

0.5.13 Mappings and coordinate systems

To set the mapping used:

```
set mapping spherical #other options include cartesian, cylindrical
```

Note:

- In **spherical** 2 coordinates are used: ϑ and φ with

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\vartheta\cos\varphi \\ \sin\vartheta\cos\varphi \\ \sin\varphi \end{pmatrix}$$

- In **cylindrical** 2 coordinates are needed: ϑ and z with

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\vartheta \\ \sin\vartheta \\ z \end{pmatrix}$$