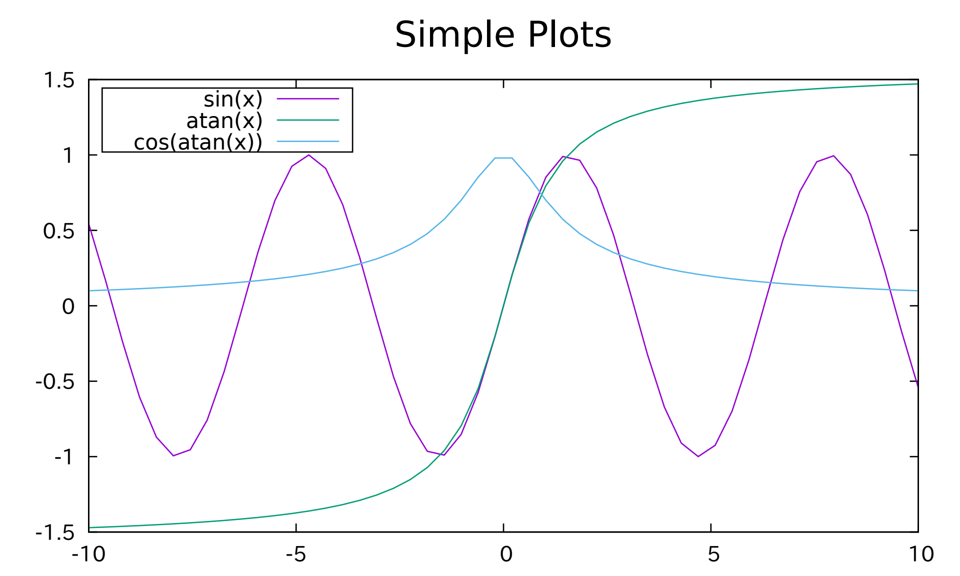
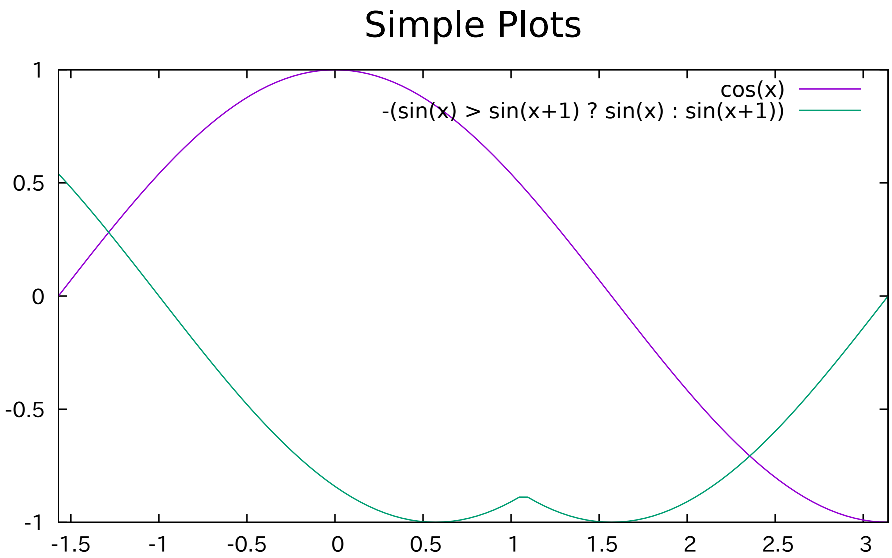
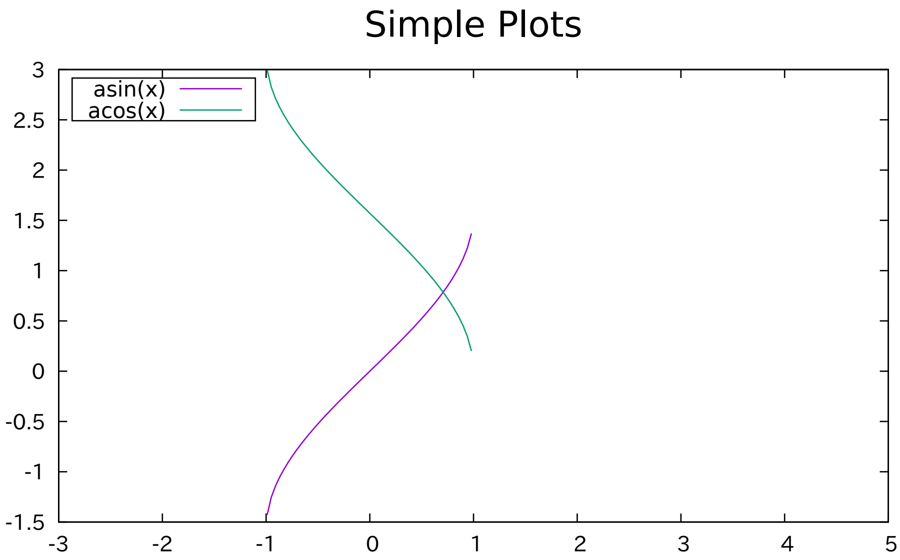
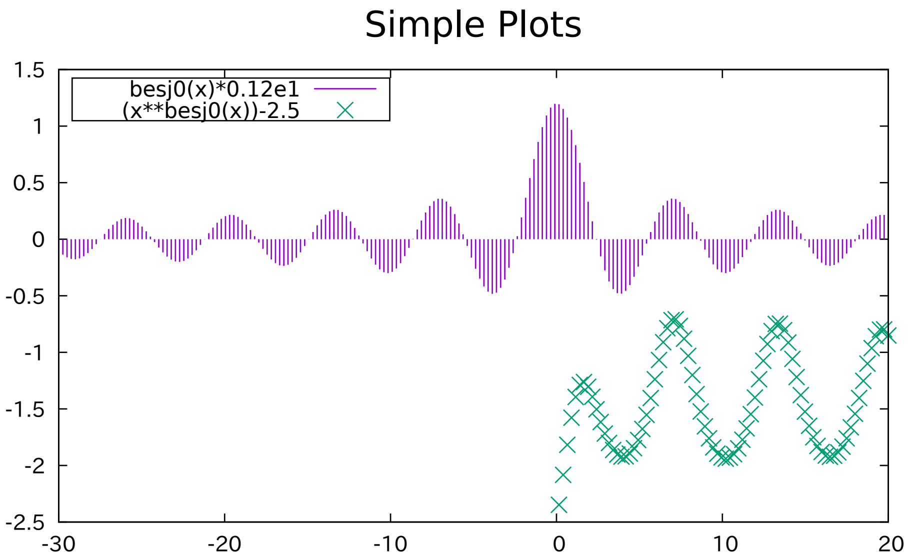
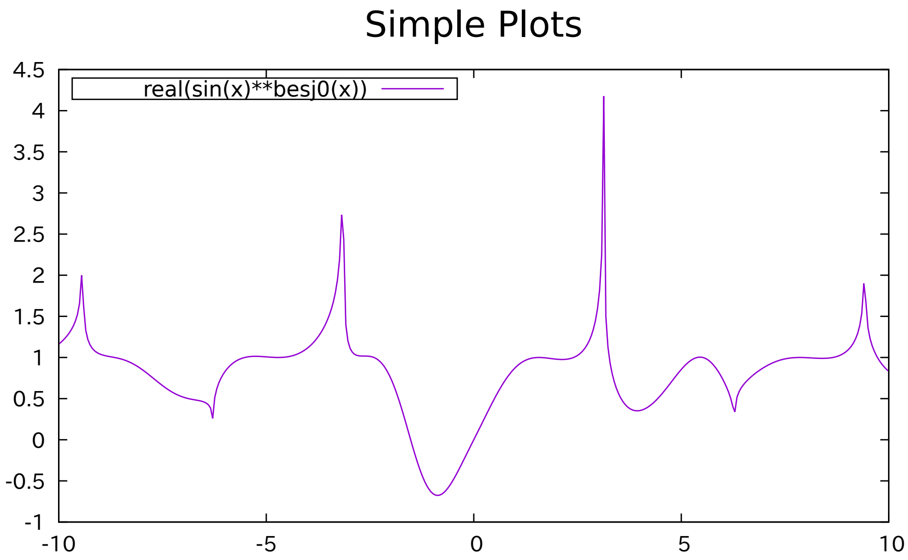

Simple Plots
sin(x)
atan(x)
cos(atan(x))
1.5
1
0.5
0
-0.5
-1
-1.5
-10
-5
0
5
10

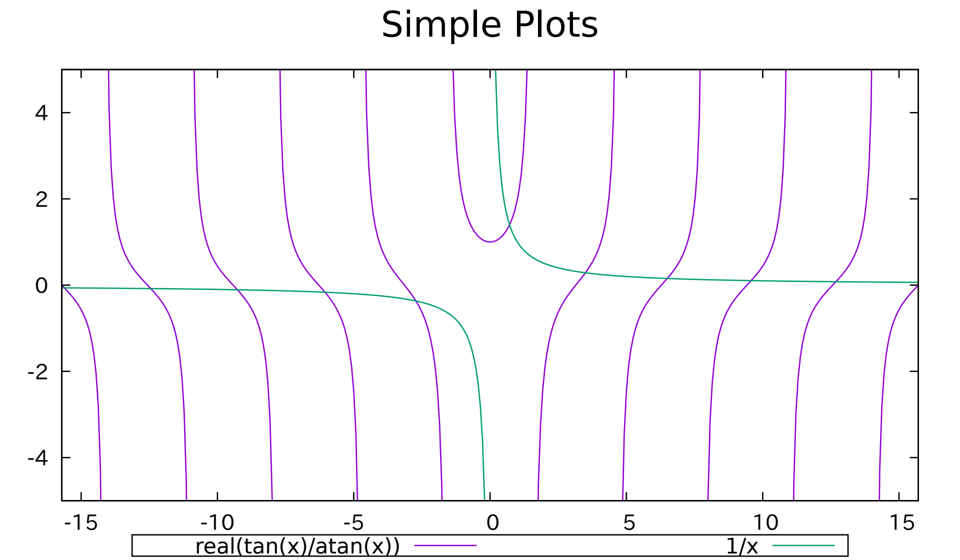

Simple Plots
cos(x)
-(sin(x) > sin(x+1) ? sin(x) : sin(x+1))
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

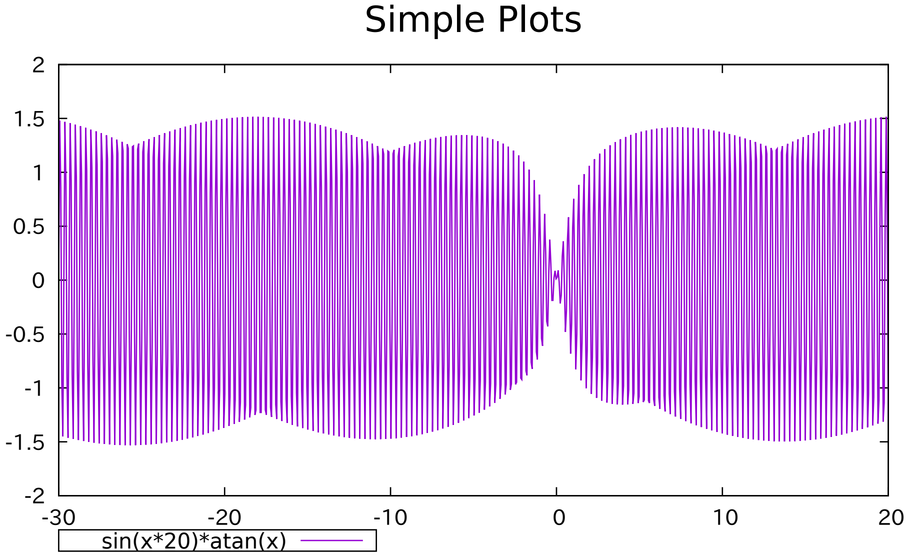

Simple Plots
asin(x)
acos(x)
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-3
-2
-1
0
1
2
3
4
5

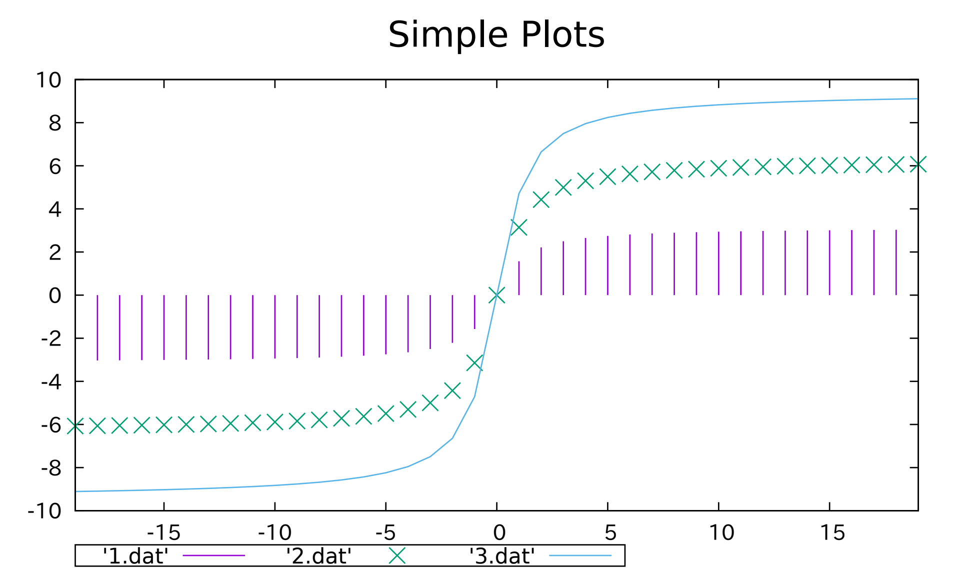

Simple Plots
besj0(x)*0.12e1
(x**besj0(x))-2.5
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-30
-20
-10
0
10
20

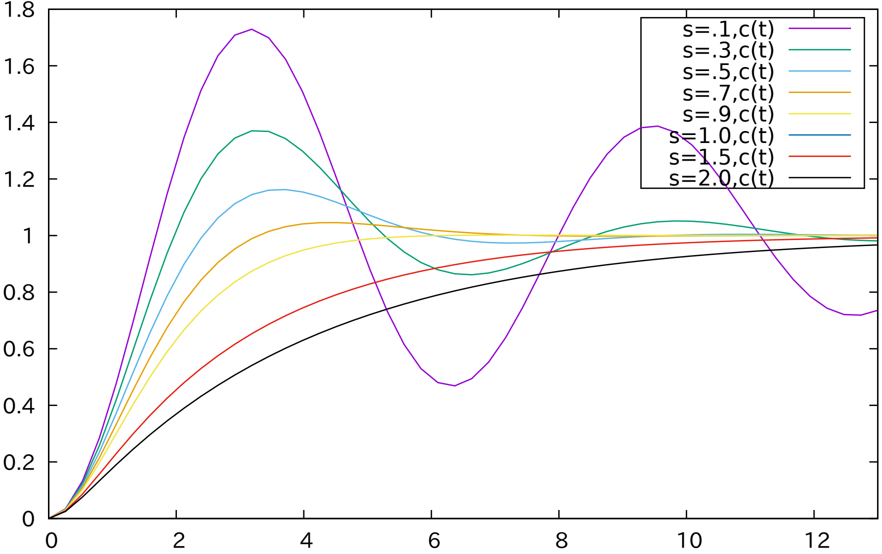


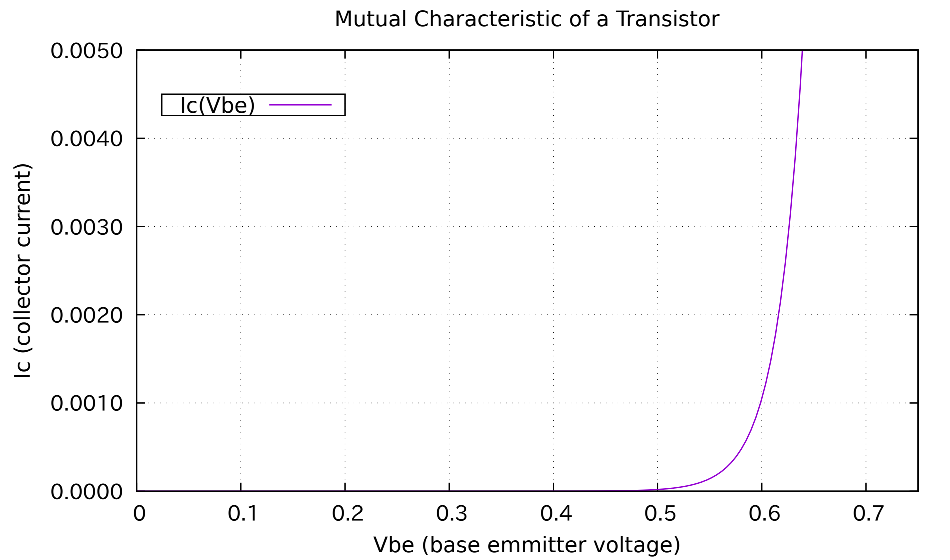
Simple Plots
real(sin(x)**besj0(x))
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-10
-5
0
5
10


Simple Plots
4
2
0
-2
-4
-15
-10
-5
0
5
10
15
real(tan(x)/atan(x))
1/x

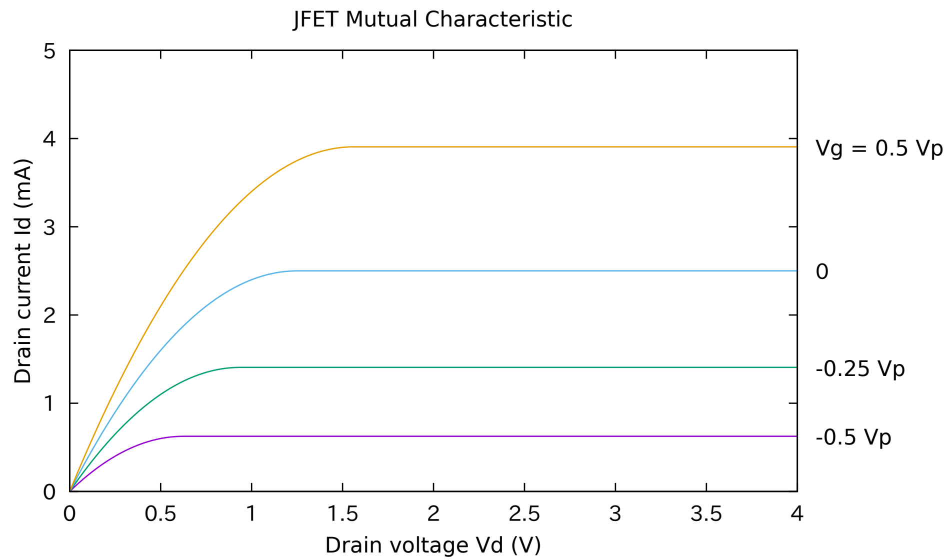

Simple Plots
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-30
-20
-10
0
10
20
sin(x*20)*atan(x)

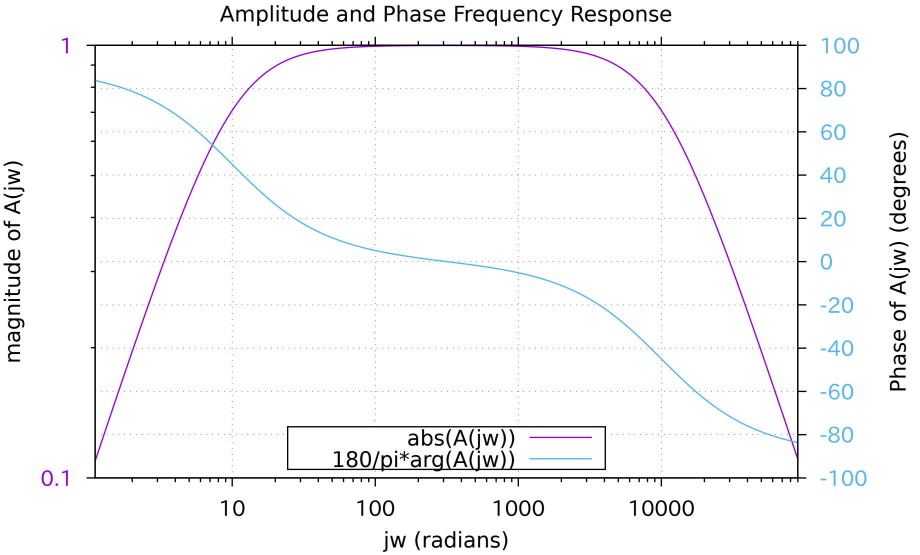

Simple Plots
10
8
6
4
2
0
-2
-4
-6
-8
-10
-15
-10
-5
0
5
10
15
'1.dat'
'2.dat'
'3.dat'

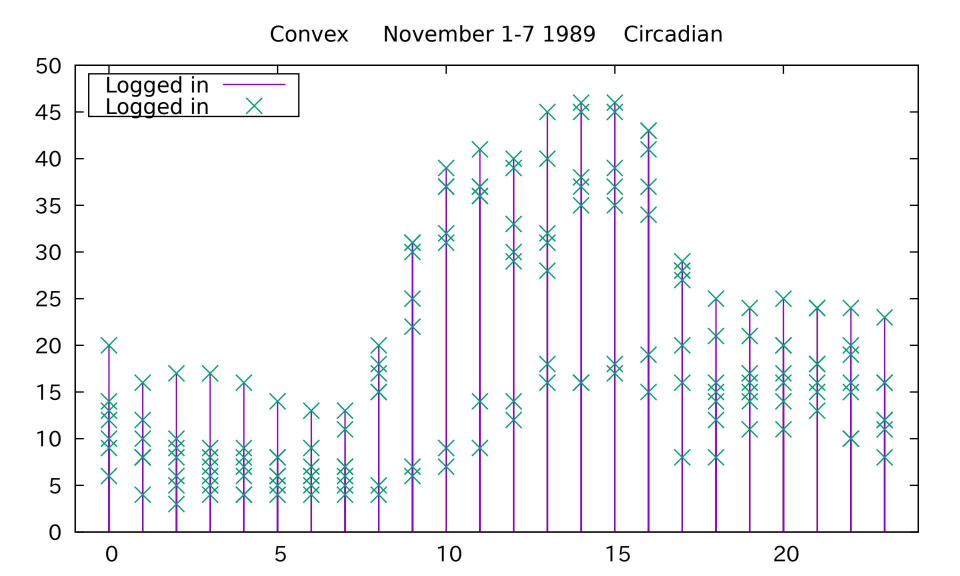

s=.1,c(t)
s=.3,c(t)
s=.5,c(t)
s=.7,c(t)
s=.9,c(t)
s=1.0,c(t)
s=1.5,c(t)
s=2.0,c(t)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
2
4
6
8
10
12

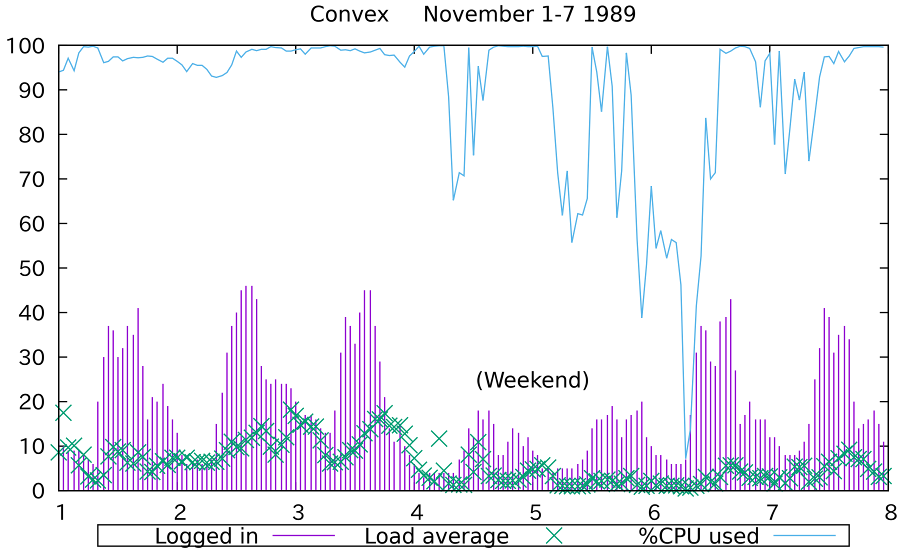


Mutual Characteristic of a Transistor
0.0050
0.0040
0.0030
0.0020
0.0010
0.0000
Ic (collector current)
Ic(Vbe)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Vbe (base emmitter voltage)

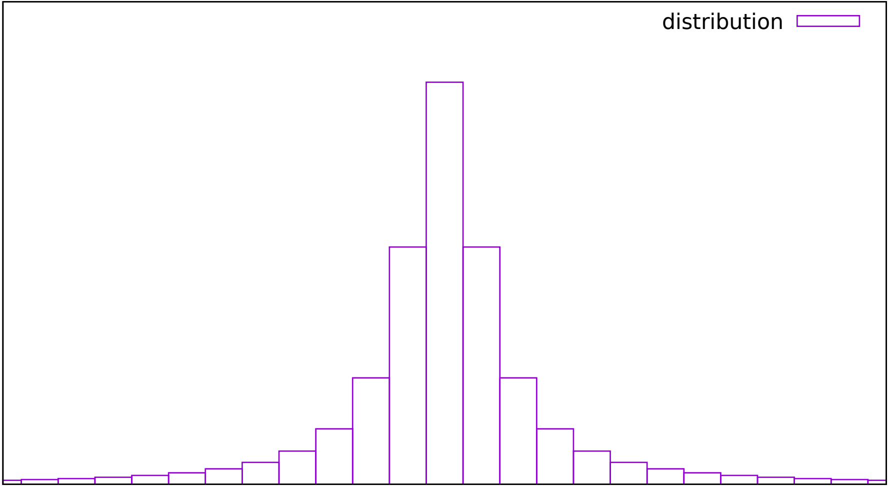

JFET Mutual Characteristic
Drain current Id (mA)
Drain voltage Vd (V)
0
1
2
3
4
5
0
0.5
1
1.5
2
2.5
3
3.5
4
Vg = 0.5 Vp
0
-0.25 Vp
-0.5 Vp

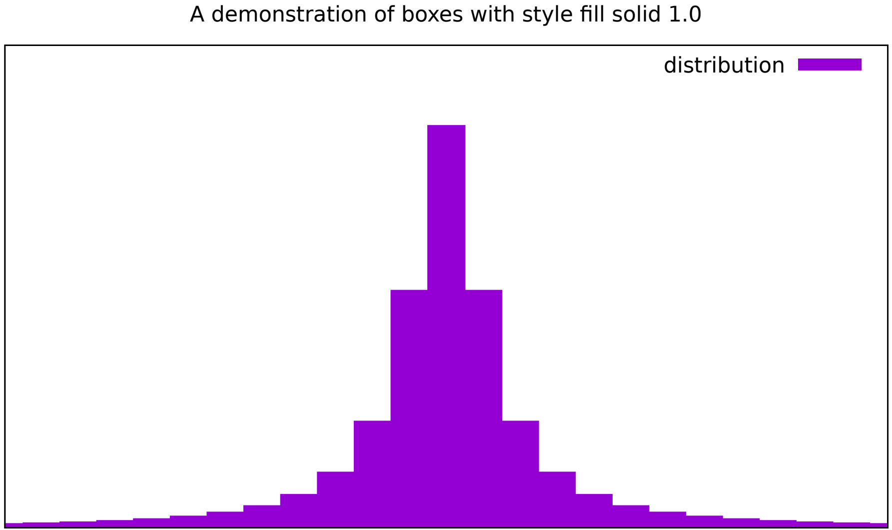

Amplitude and Phase Frequency Response
1
0.1
100
80
60
40
20
0
-20
-40
-60
-80
-100
10
100
1000
10000
magnitude of A(jw)
Phase of A(jw) (degrees)
jw (radians)
abs(A(jw))
180/pi*arg(A(jw))


Convex November 1-7 1989 Circadian
Logged in
Logged in
0
5
10
15
20
25
30
35
40
45
50
0
5
10
15
20

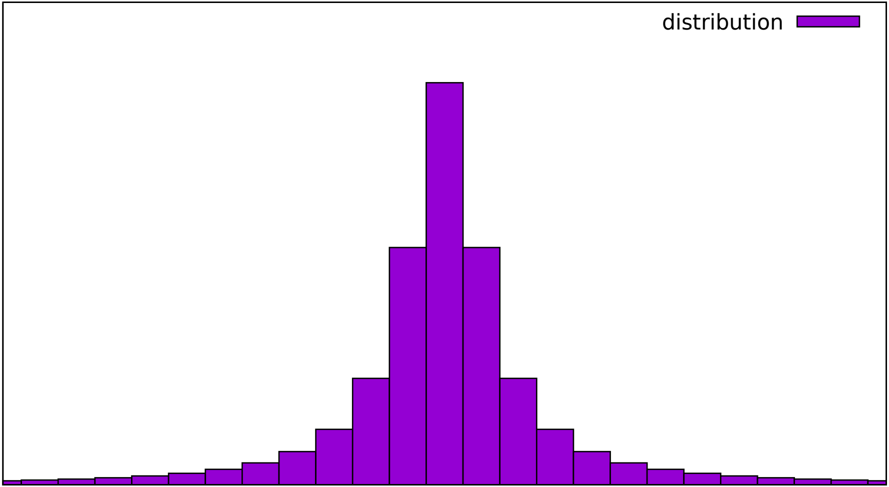

Convex November 1-7 1989
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
(Weekend)
Logged in
Load average
%CPU used

A demonstration of boxes with default properties

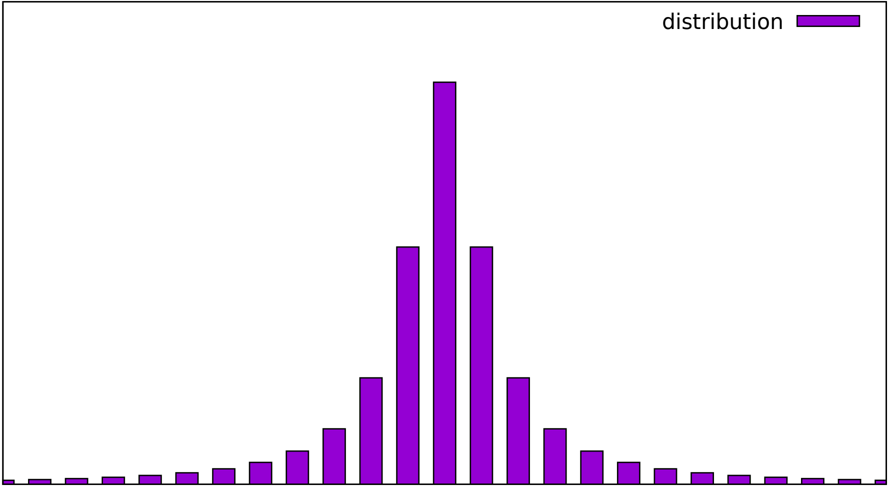


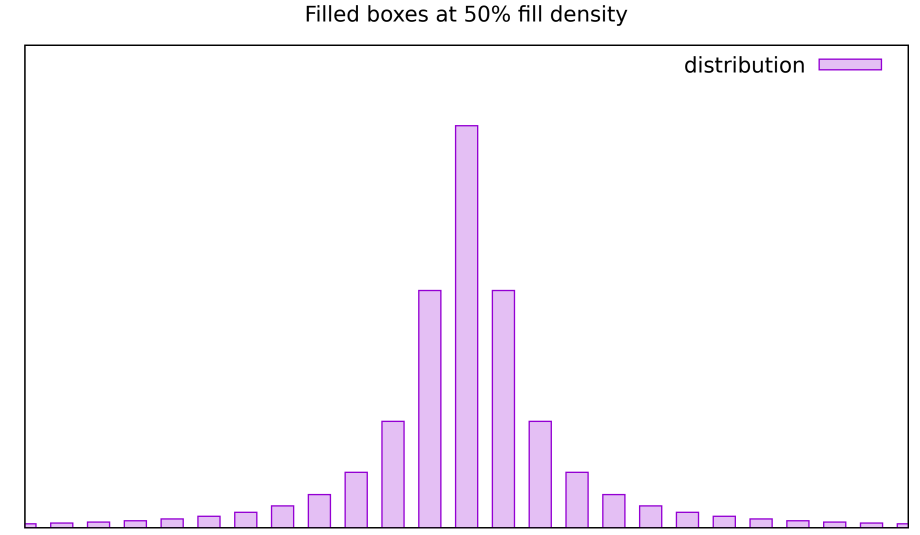

A demonstration of boxes with style fill solid 1.0
distribution

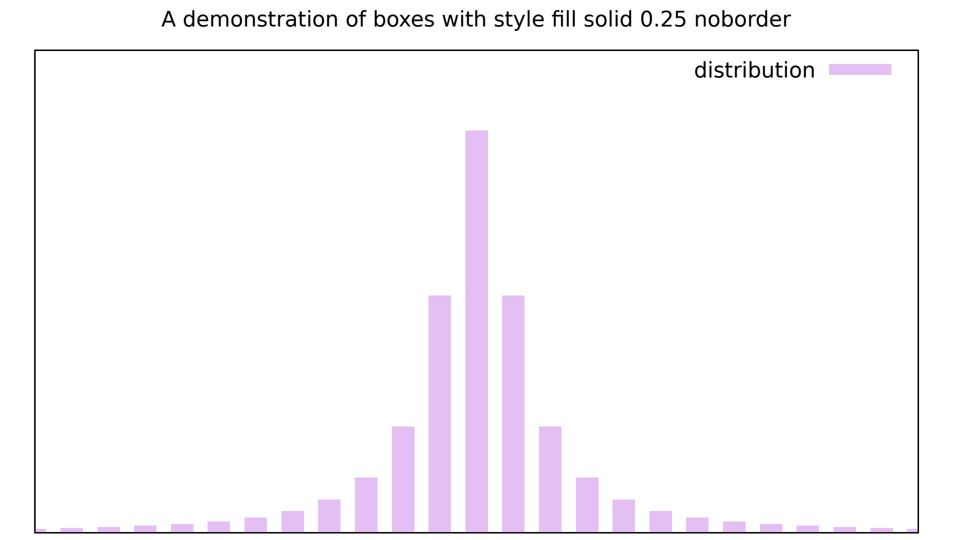
A demonstration of boxes with style fill solid border -1

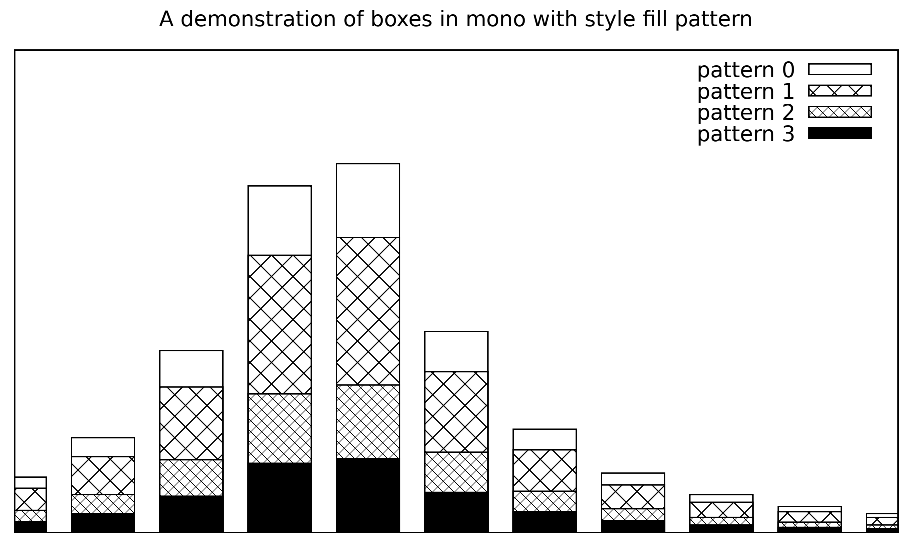


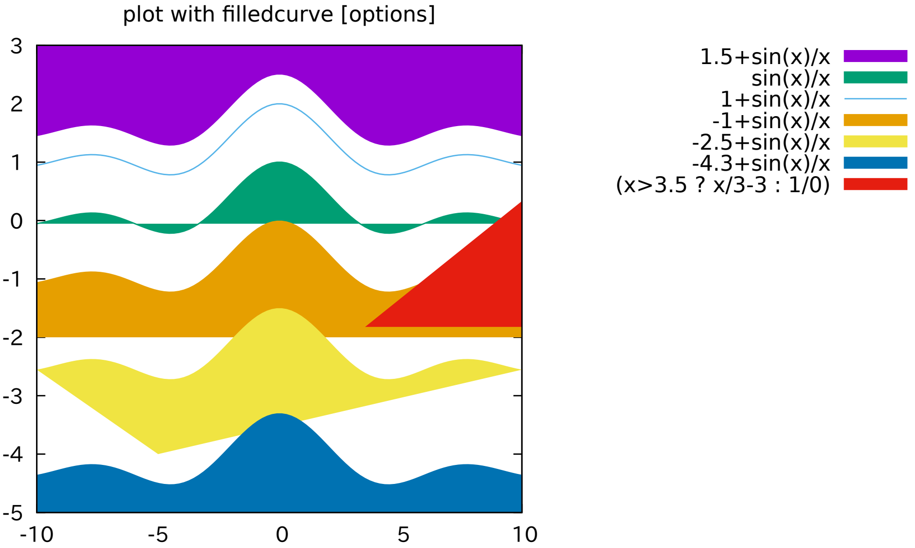
Filled boxes of reduced width

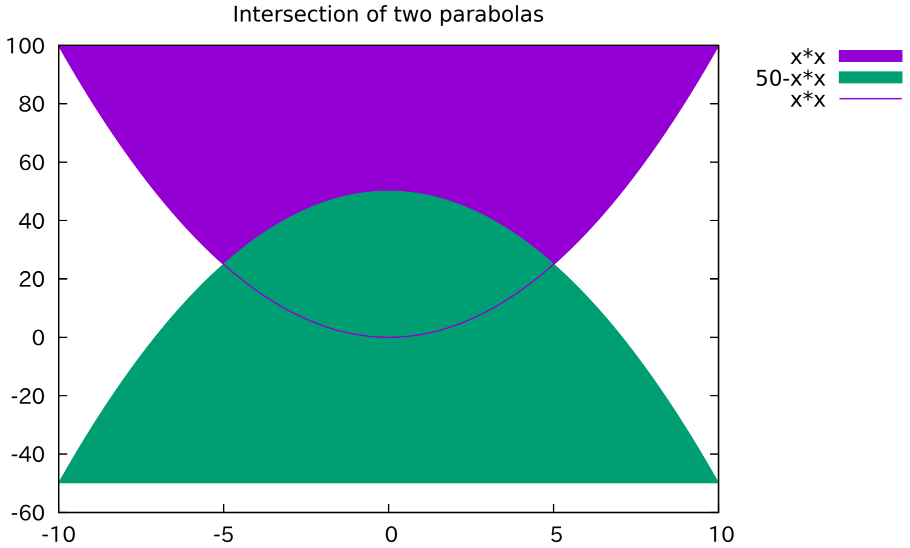



Filled boxes at 50% fill density
distribution

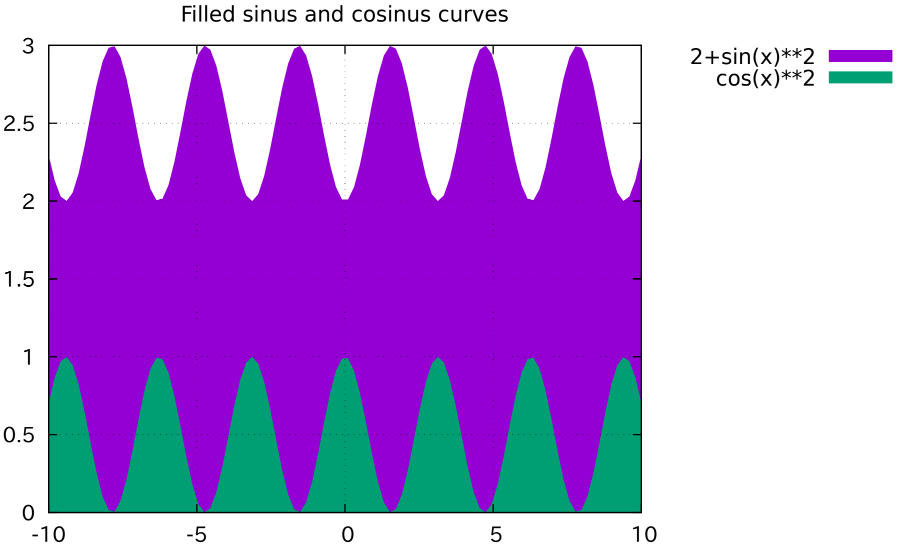

A demonstration of boxes with style fill solid 0.25 noborder
distribution

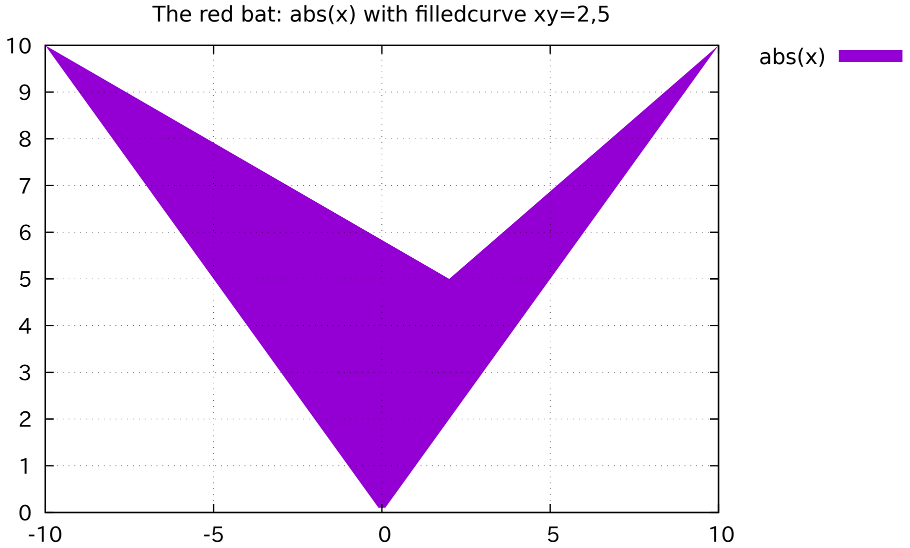

A demonstration of boxes in mono with style fill pattern
pattern 0
pattern 1
pattern 2
pattern 3

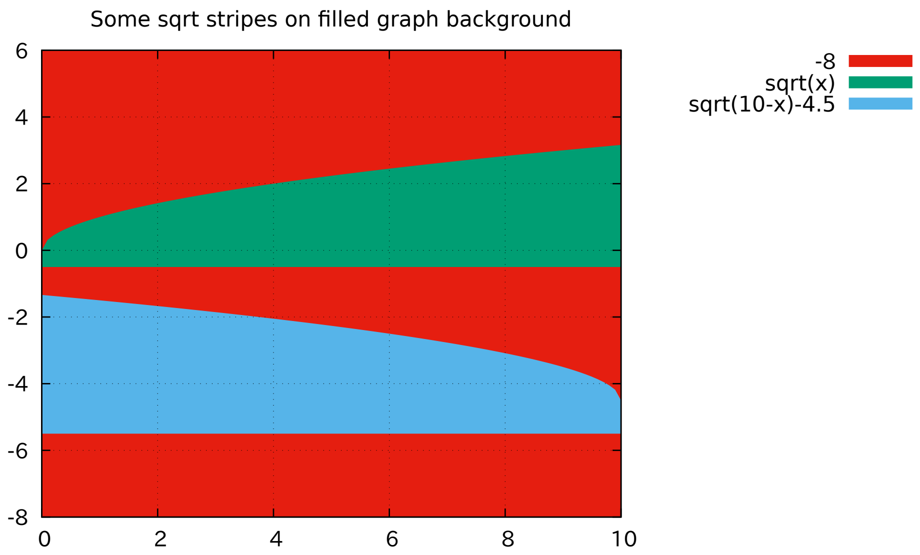

plot with filledcurve [options]
3
2
1
0
-1
-2
-3
-4
-5
-10
-5
0
5
10
1.5+sin(x)/x
sin(x)/x
1+sin(x)/x
-1+sin(x)/x
-2.5+sin(x)/x
-4.3+sin(x)/x
(x>3.5 ? x/3-3 : 1/0)



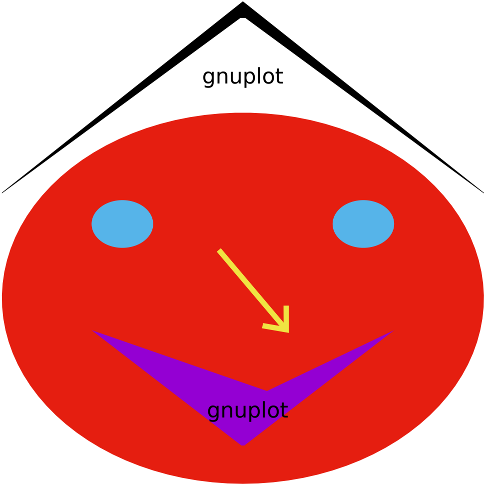
Intersection of two parabolas
100
80
60
40
20
0
-20
-40
-60
-10
-5
0
5
10
x*x
50-x*x
x*x


Filled sinus and cosinus curves
2+sin(x)**2
cos(x)**2
3
2.5
2
1.5
1
0.5
0
-10
-5
0
5
10

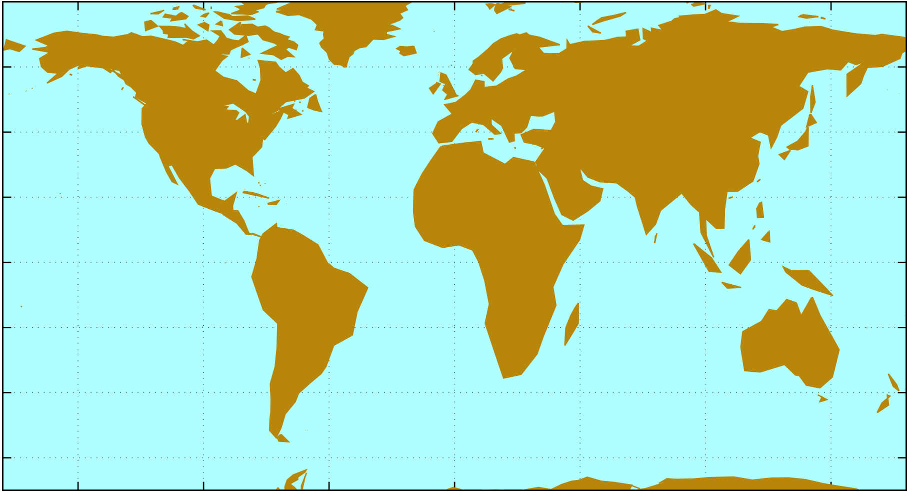

The red bat: abs(x) with filledcurve xy=2,5
abs(x)
10
9
8
7
6
5
4
3
2
1
0
-10
-5
0
5
10

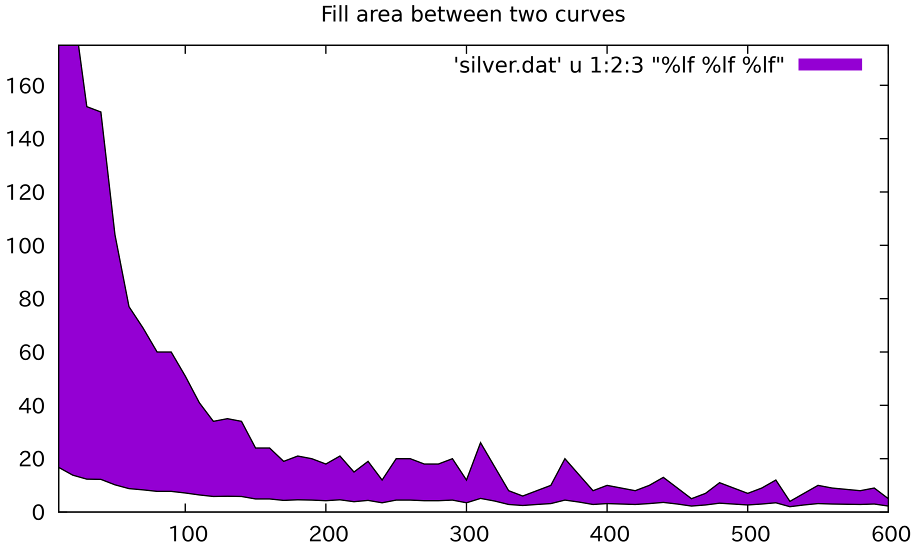

Some sqrt stripes on filled graph background
6
4
2
0
-2
-4
-6
-8
0
2
4
6
8
10
-8
sqrt(x)
sqrt(10-x)-4.5

Let's smile with parametric filled curves

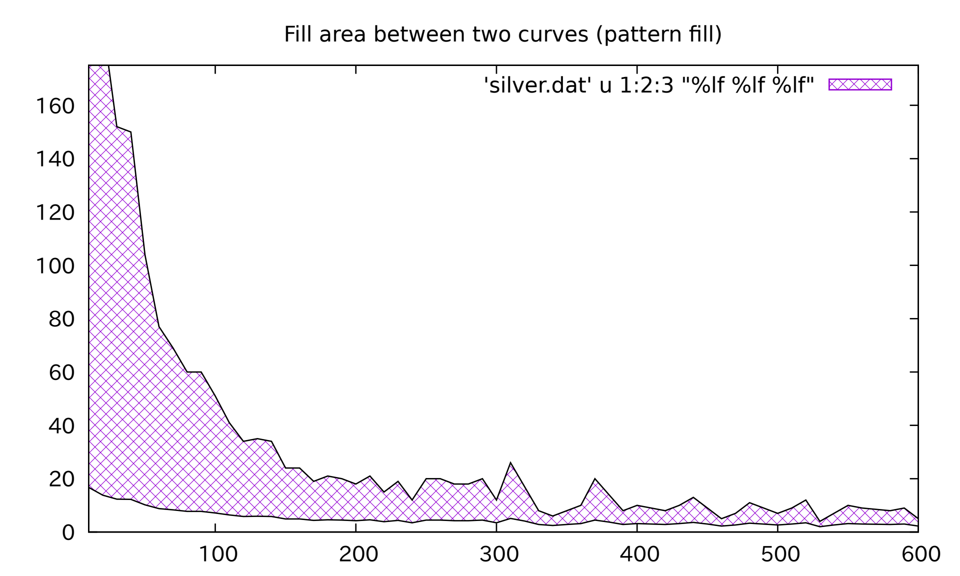


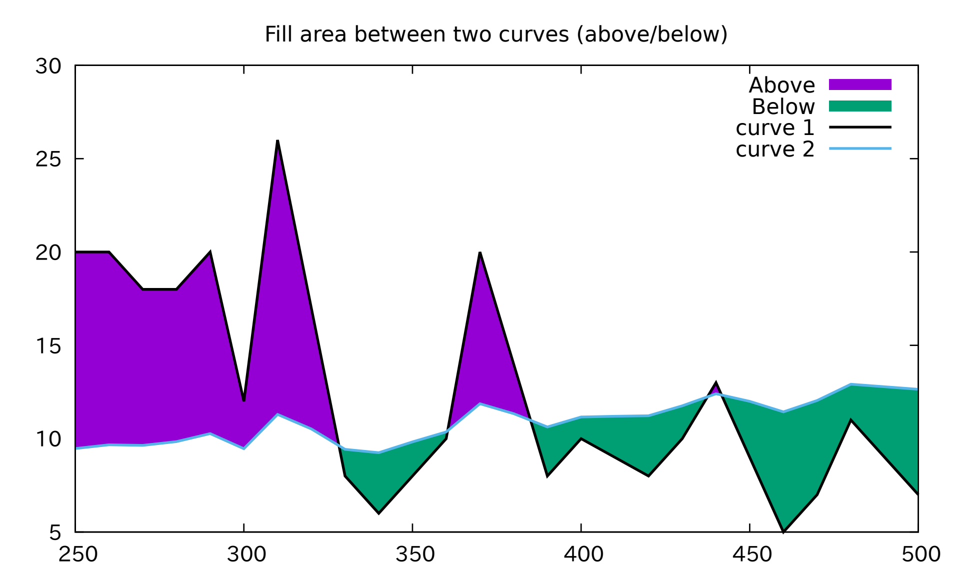
world.dat plotted with filledcurves

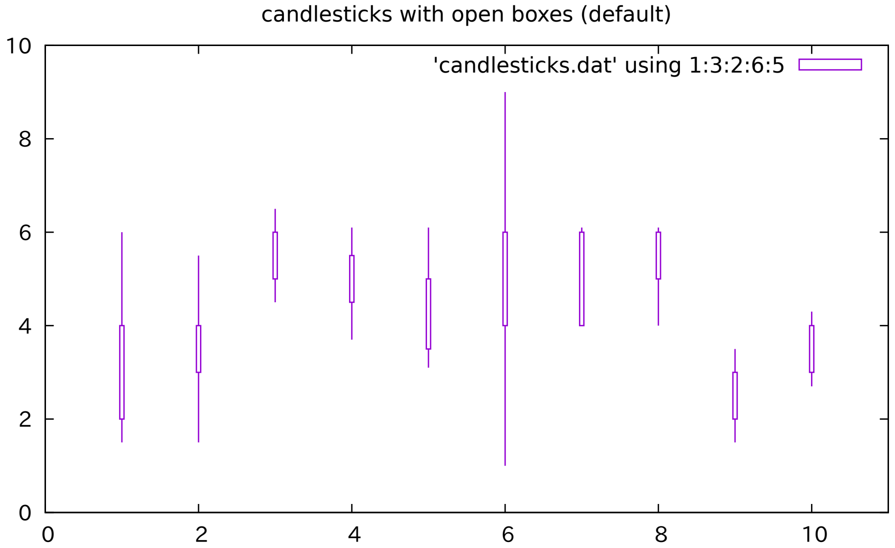


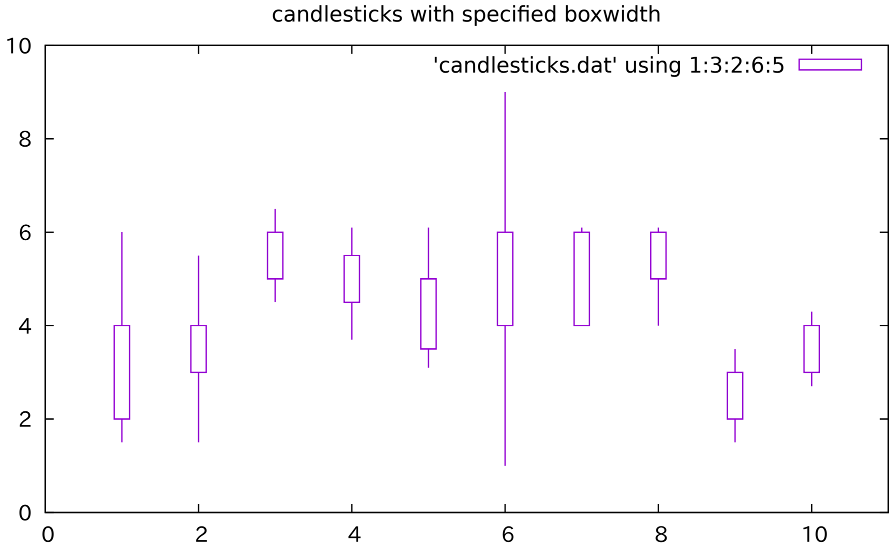


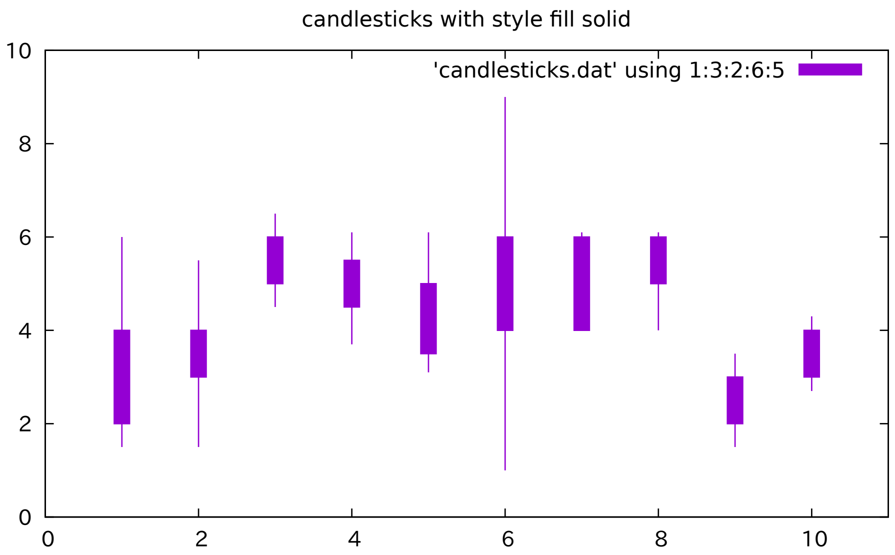
Fill area between two curves
'silver.dat' u 1:2:3 "%lf %lf %lf"
0
20
40
60
80
100
120
140
160
100
200
300
400
500
600

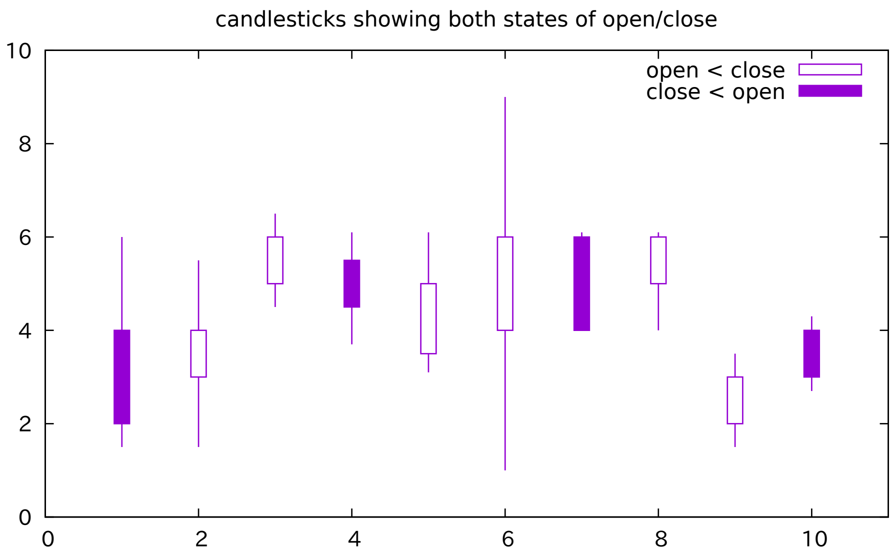

Fill area between two curves (pattern fill)
'silver.dat' u 1:2:3 "%lf %lf %lf"
0
20
40
60
80
100
120
140
160
100
200
300
400
500
600

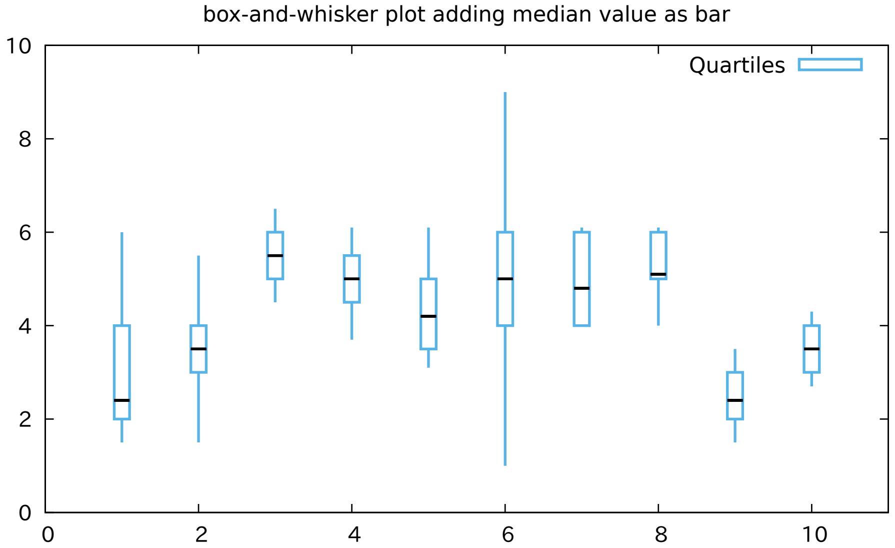

Fill area between two curves (above/below)
Above
Below
curve 1
curve 2
30
25
20
15
10
5
250
300
350
400
450
500

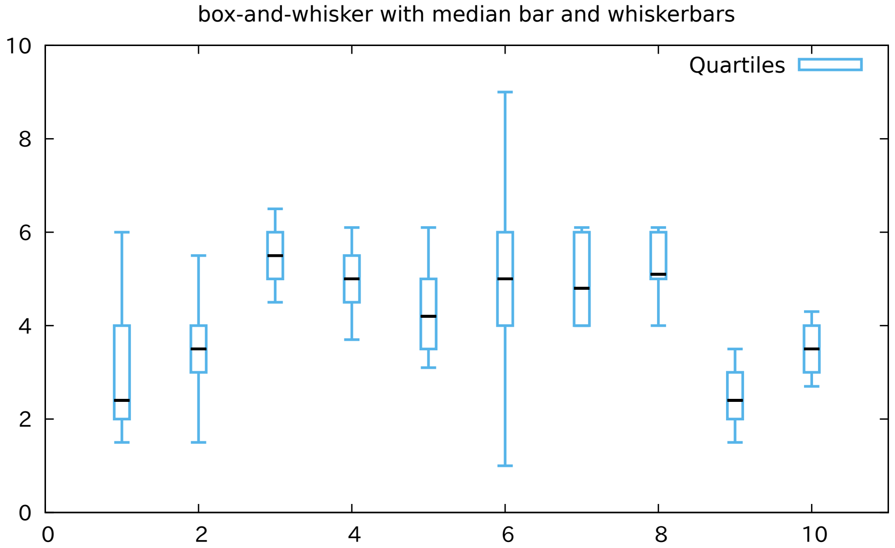

candlesticks with open boxes (default)
'candlesticks.dat' using 1:3:2:6:5
0
2
4
6
8
10
0
2
4
6
8
10

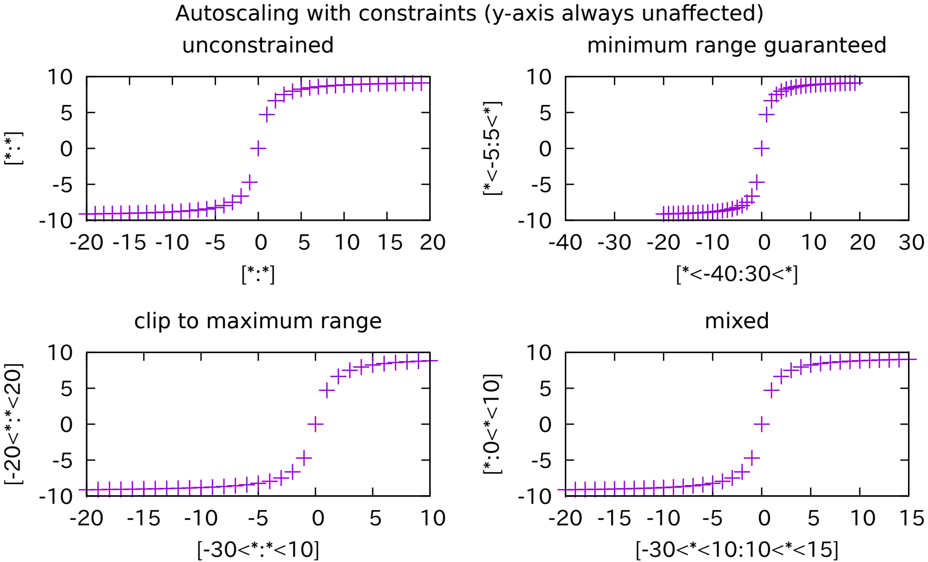

candlesticks with specified boxwidth
'candlesticks.dat' using 1:3:2:6:5
0
2
4
6
8
10
0
2
4
6
8
10

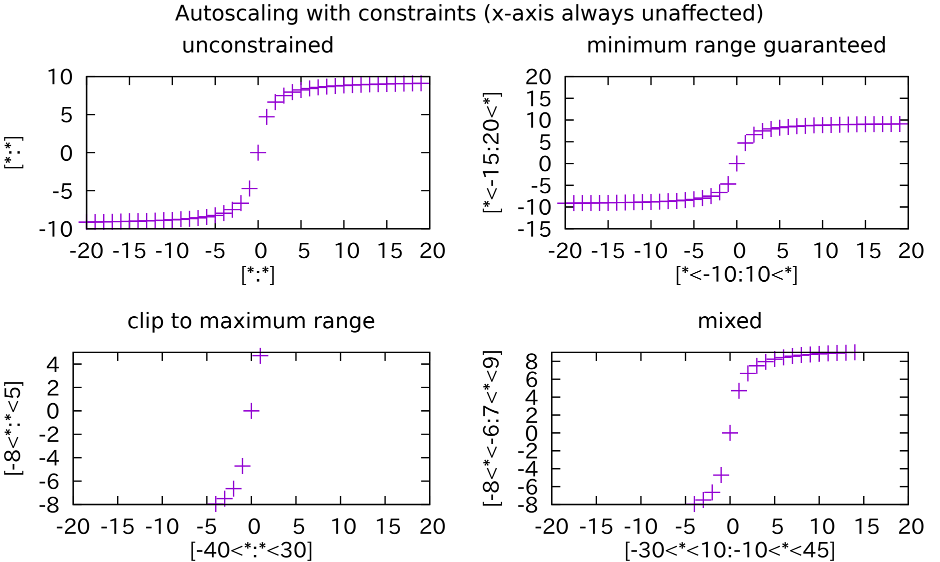

candlesticks with style fill solid
'candlesticks.dat' using 1:3:2:6:5
0
2
4
6
8
10
0
2
4
6
8
10


candlesticks showing both states of open/close
open < close
close < open
0
2
4
6
8
10
0
2
4
6
8
10

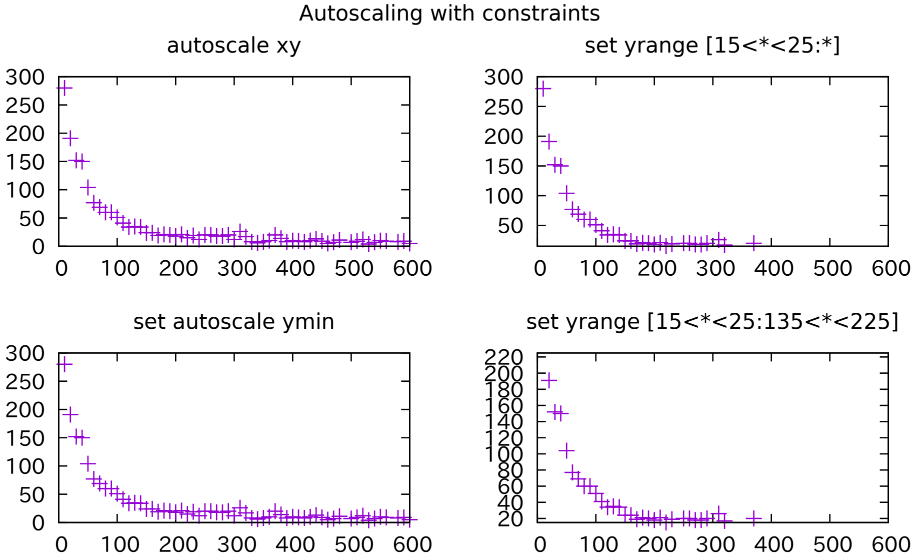

box-and-whisker plot adding median value as bar
Quartiles
0
2
4
6
8
10
0
2
4
6
8
10


box-and-whisker with median bar and whiskerbars
Quartiles
10
8
6
4
2
0
0
2
4
6
8
10

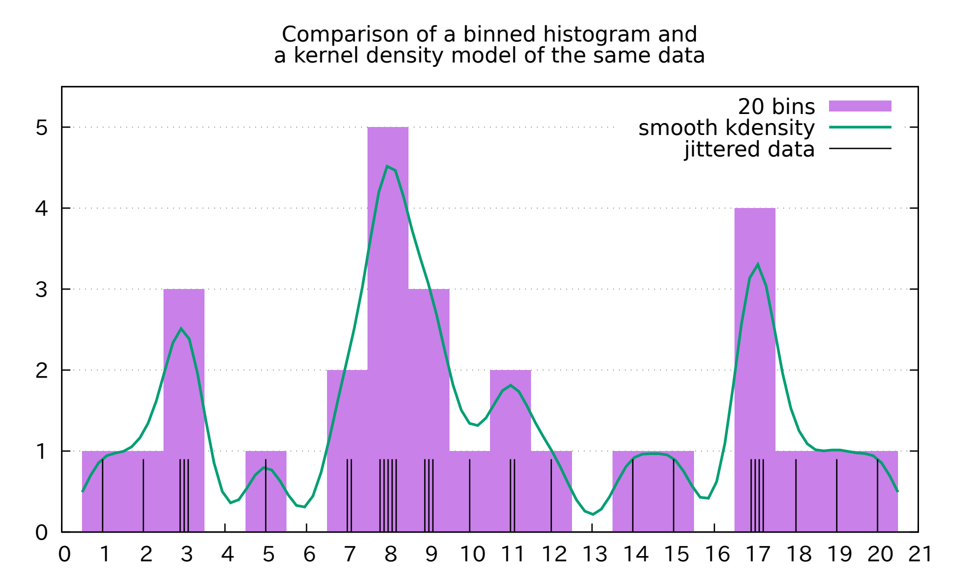

Autoscaling with constraints (y-axis always unaffected)
unconstrained
[*:*]
[*:*]
minimum range guaranteed
[*<-5:5<*]
[*<-40:30<*]
clip to maximum range
[-20<*:*<20]
[-30<*:*<10]
mixed
[*:0<*<10]
[-30<*<10:10<*<15]

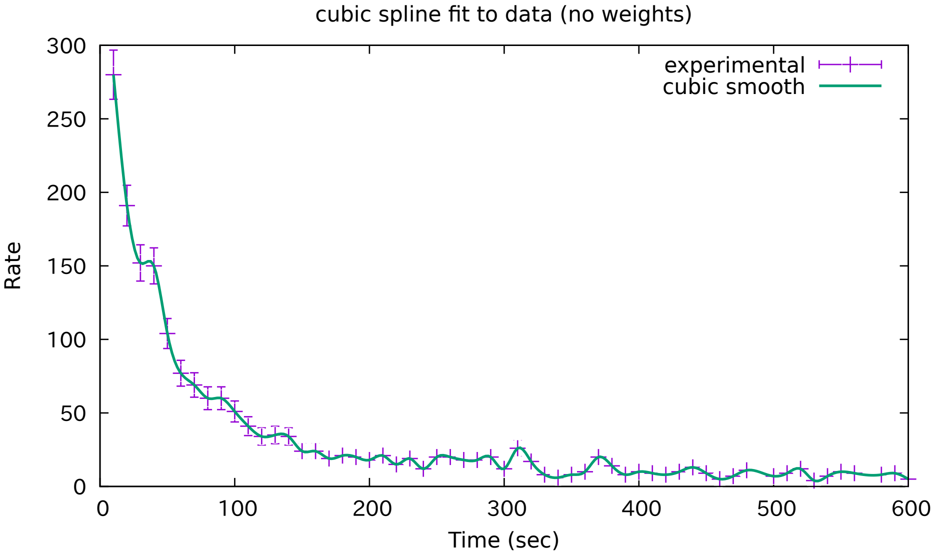


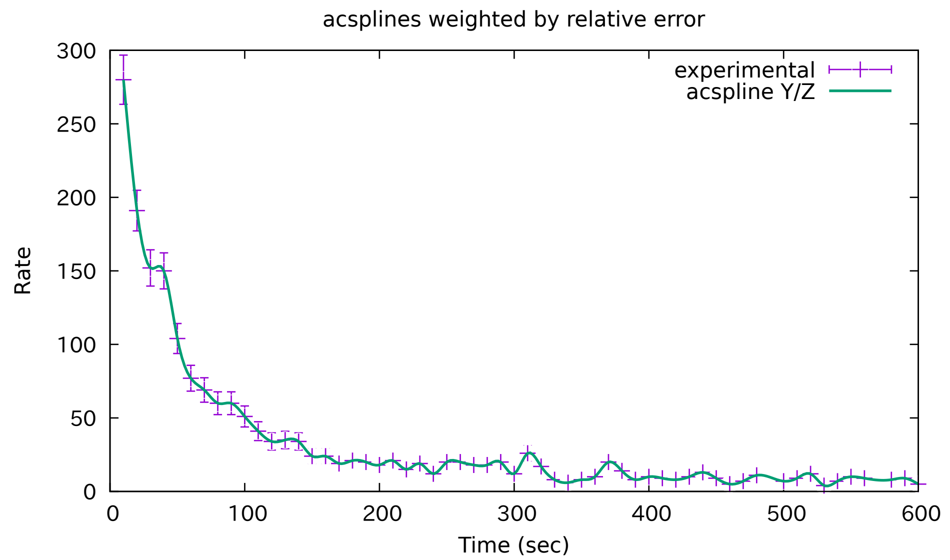
Autoscaling with constraints (x-axis always unaffected)
unconstrained
[*:*]
[*:*]
minimum range guaranteed
[*<-15:20<*]
[*<-10:10<*]
clip to maximum range
[-8<*:*<5]
[-40<*:*<30]
mixed
[-8<*<-6:7<*<9]
[-30<*<10:-10<*<45]

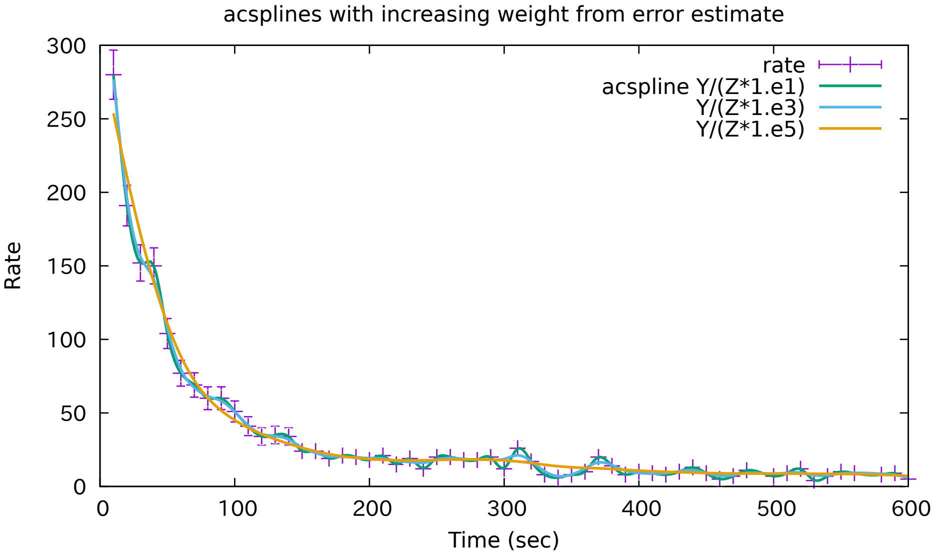


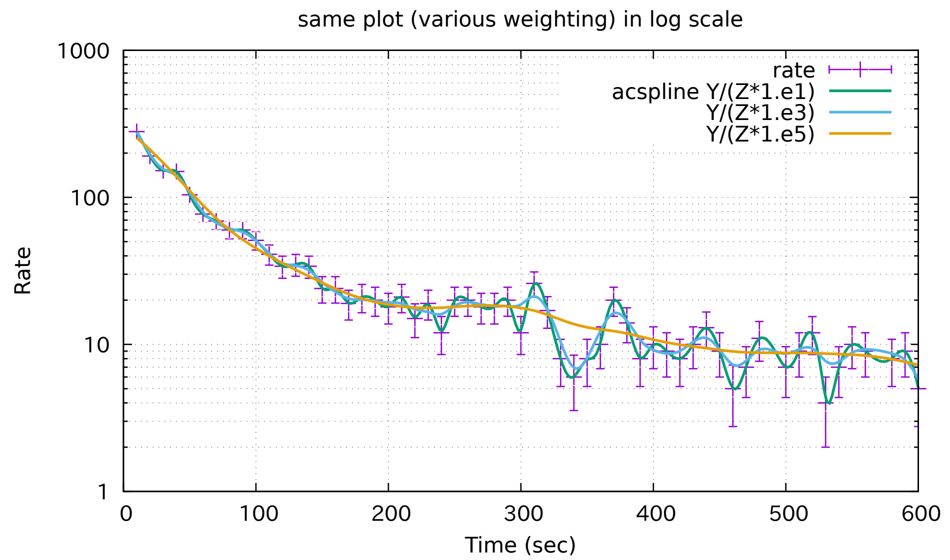
Autoscaling with constraints
autoscale xy
set yrange [15<*<25:*]
set autoscale ymin
set yrange [15<*<25:135<*<225]
0
100
200
300
400
500
600
0
50
100
150
200
250
300
20
40
60
80
100
120
140
160
180
200
220

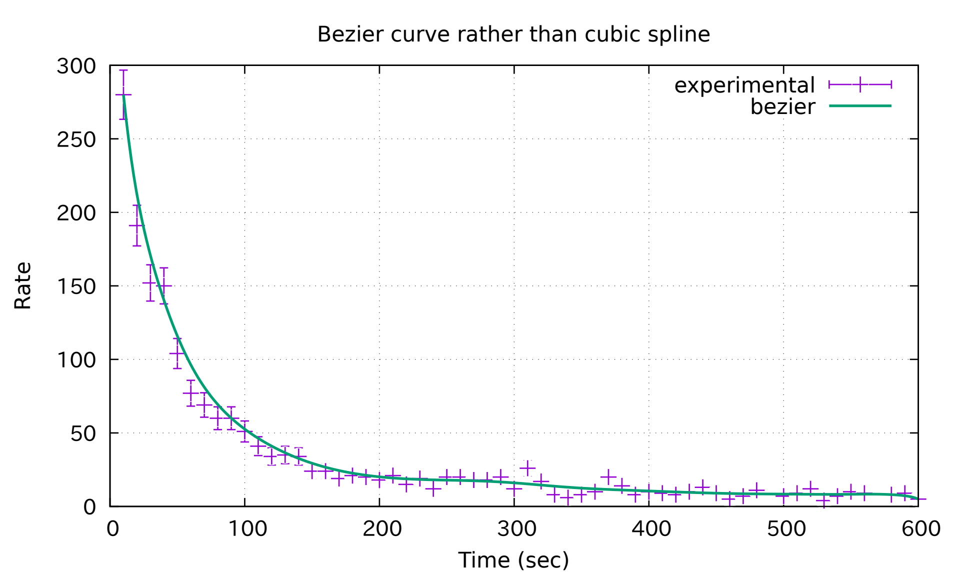

Comparison of a binned histogram and
a kernel density model of the same data
20 bins
smooth kdensity
jittered data
0
1
2
3
4
5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

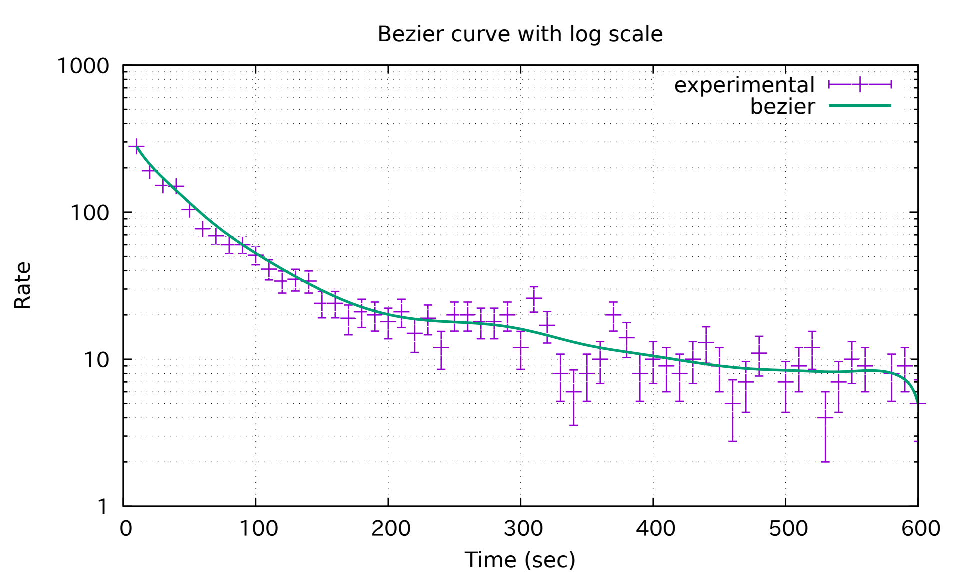

cubic spline fit to data (no weights)
experimental
cubic smooth
Rate
Time (sec)
0
50
100
150
200
250
300
0
100
200
300
400
500
600

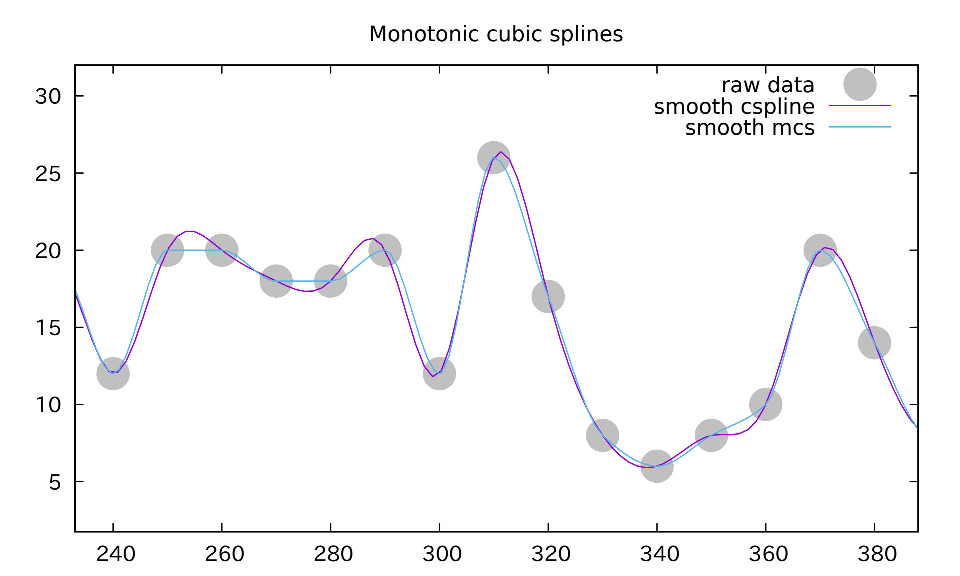

acsplines weighted by relative error
experimental
acspline Y/Z
Rate
Time (sec)
0
50
100
150
200
250
300
0
100
200
300
400
500
600


acsplines with increasing weight from error estimate
rate
acspline Y/(Z*1.e1)
Y/(Z*1.e3)
Y/(Z*1.e5)
Rate
Time (sec)
0
50
100
150
200
250
300
0
100
200
300
400
500
600

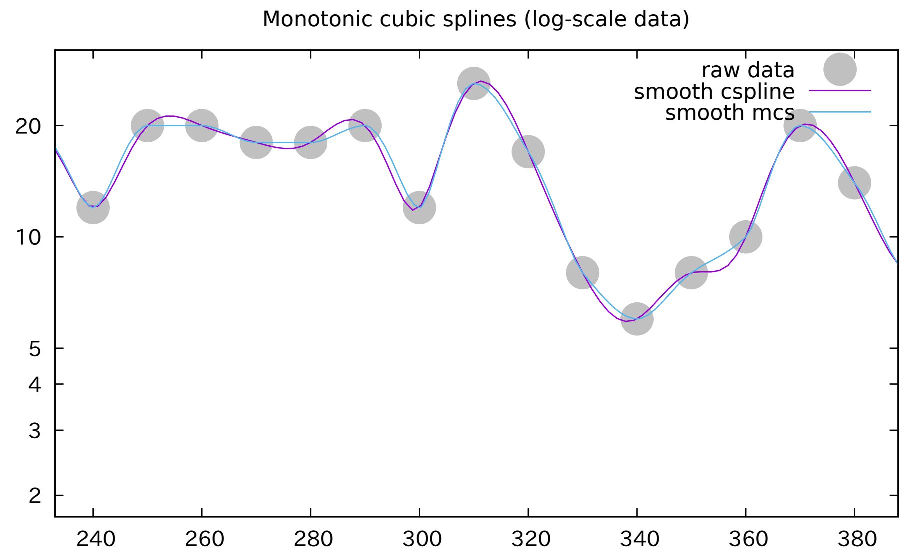

same plot (various weighting) in log scale
rate
acspline Y/(Z*1.e1)
Y/(Z*1.e3)
Y/(Z*1.e5)
Rate
Time (sec)
1000
100
10
1
0
100
200
300
400
500
600

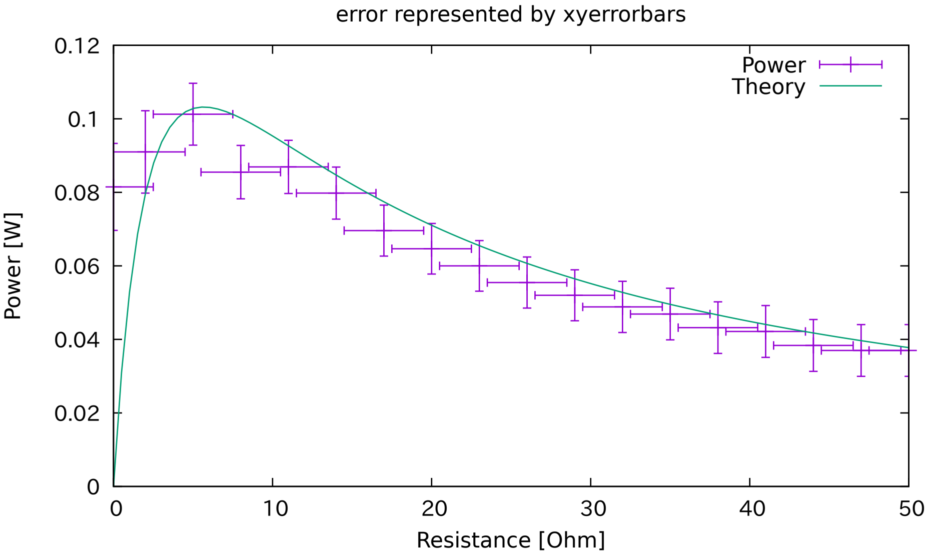

Bezier curve rather than cubic spline
experimental
bezier
Rate
Time (sec)
0
50
100
150
200
250
300
0
100
200
300
400
500
600

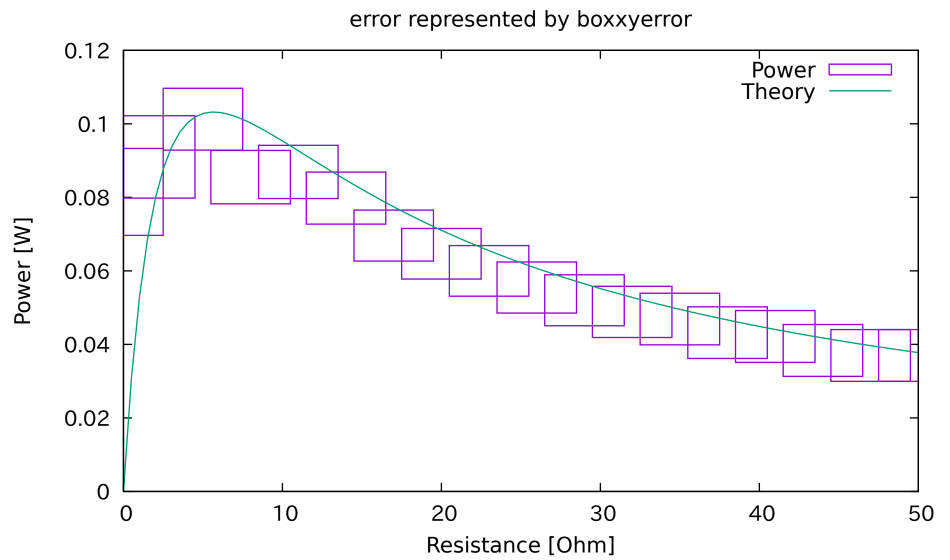

Bezier curve with log scale
1000
100
10
1
Rate
0
100
200
300
400
500
600
Time (sec)
experimental
bezier

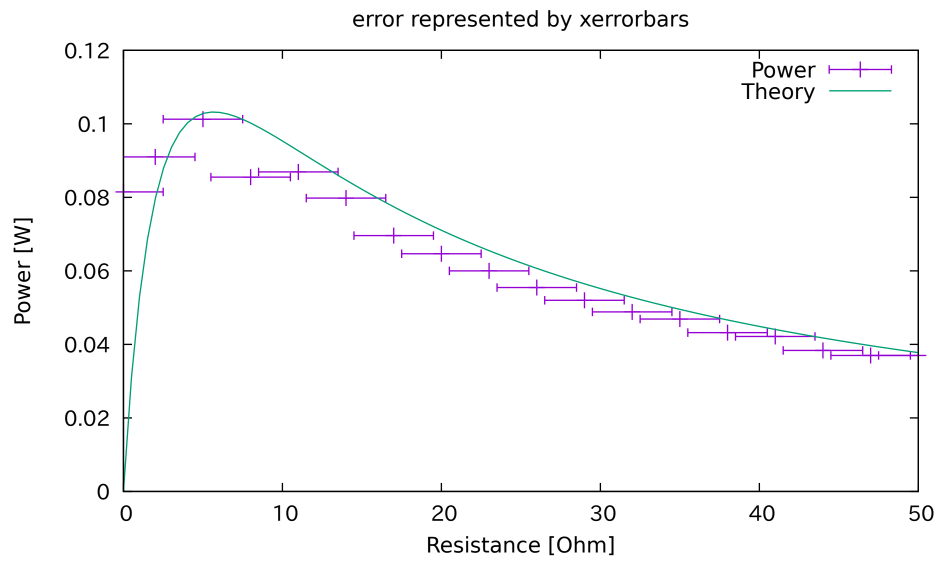


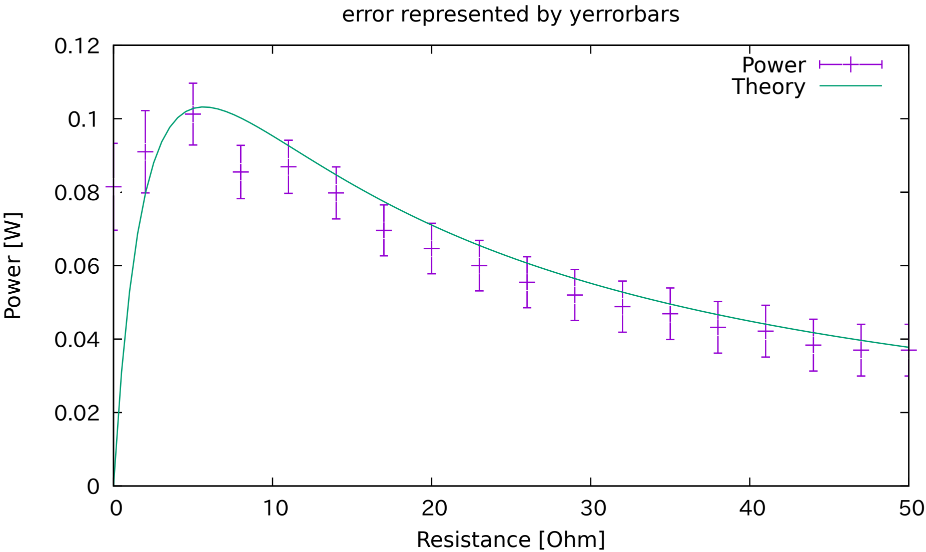
Monotonic cubic splines
raw data
smooth cspline
smooth mcs
30
25
20
15
10
5
240
260
280
300
320
340
360
380


Monotonic cubic splines (log-scale data)
raw data
smooth cspline
smooth mcs
20
10
5
4
3
2
240
260
280
300
320
340
360
380

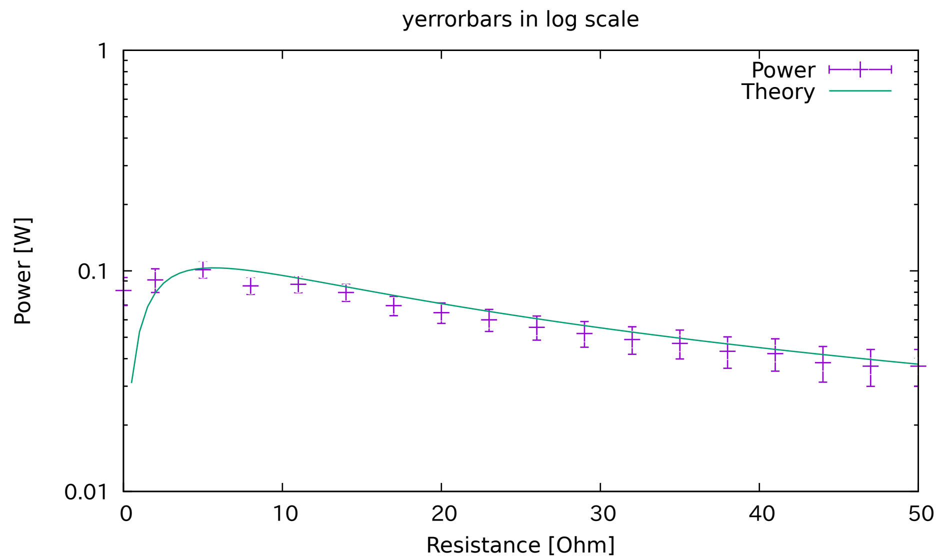

error represented by xyerrorbars
Power
Theory
Power [W]
Resistance [Ohm]
0
0.02
0.04
0.06
0.08
0.1
0.12
0
10
20
30
40
50

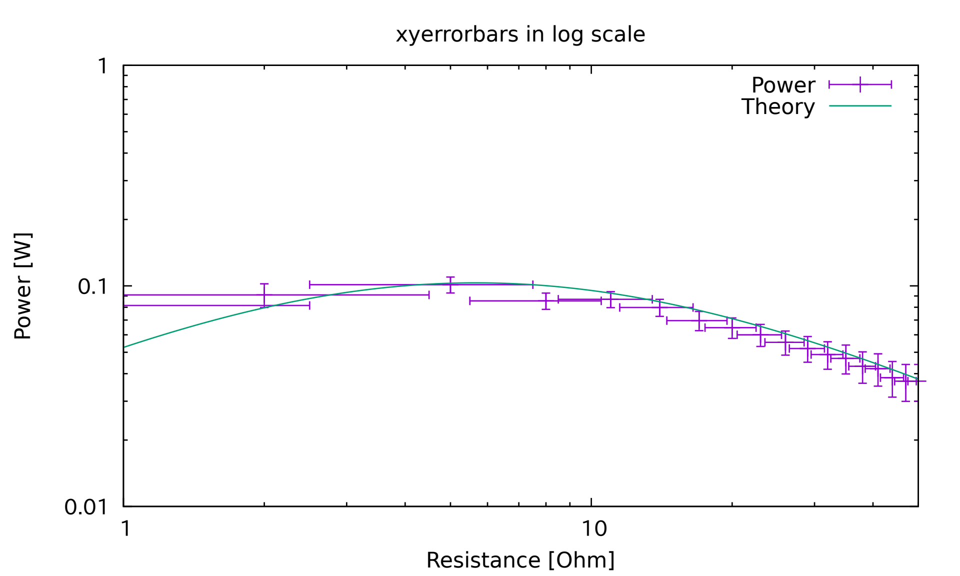

error represented by boxxyerror
Power
Theory
Power [W]
Resistance [Ohm]
0
0.02
0.04
0.06
0.08
0.1
0.12
0
10
20
30
40
50


error represented by xerrorbars
Power
Theory
Power [W]
Resistance [Ohm]
0
0.02
0.04
0.06
0.08
0.1
0.12
0
10
20
30
40
50

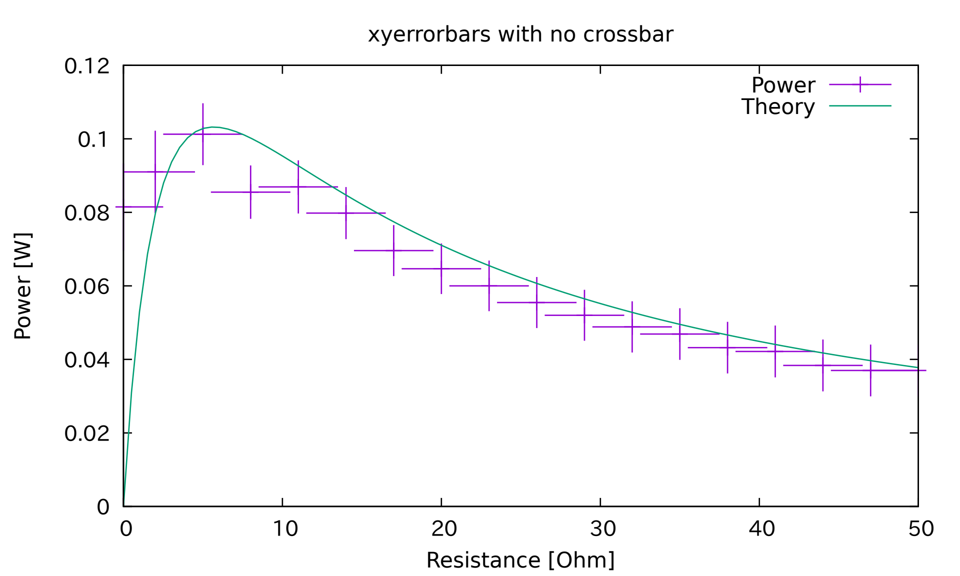


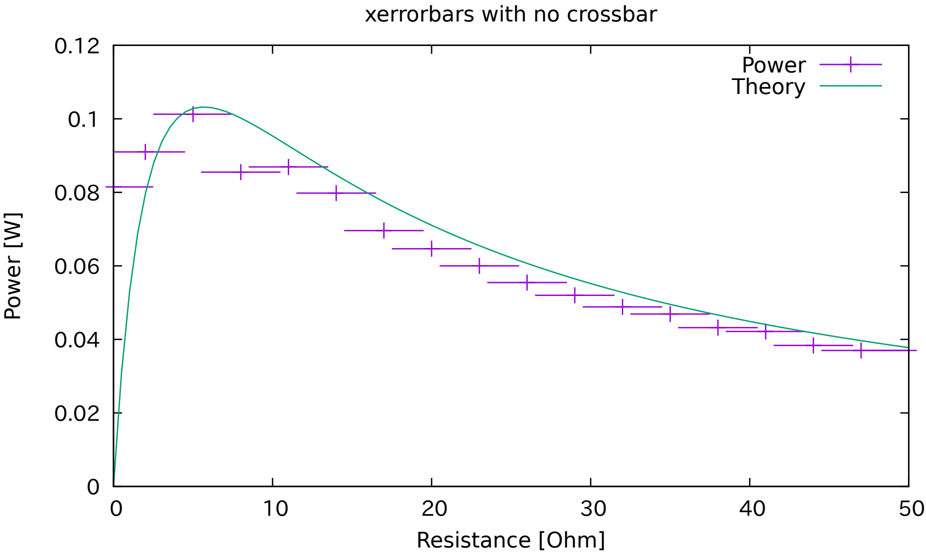
error represented by yerrorbars
Power
Theory
Power [W]
Resistance [Ohm]
0.12
0.1
0.08
0.06
0.04
0.02
0
10
20
30
40
50


yerrorbars in log scale
Power [W]
Resistance [Ohm]
Power
Theory
1
0.1
0.01
0
10
20
30
40
50

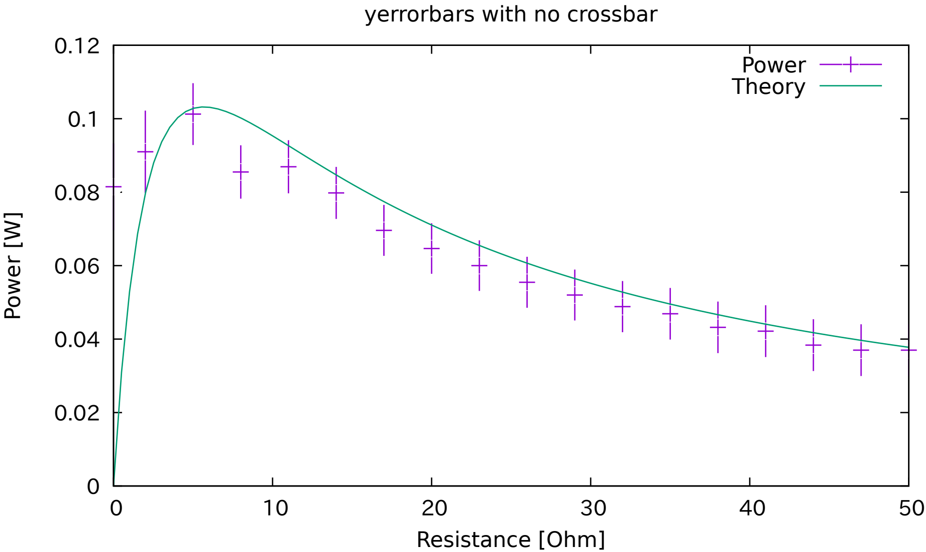


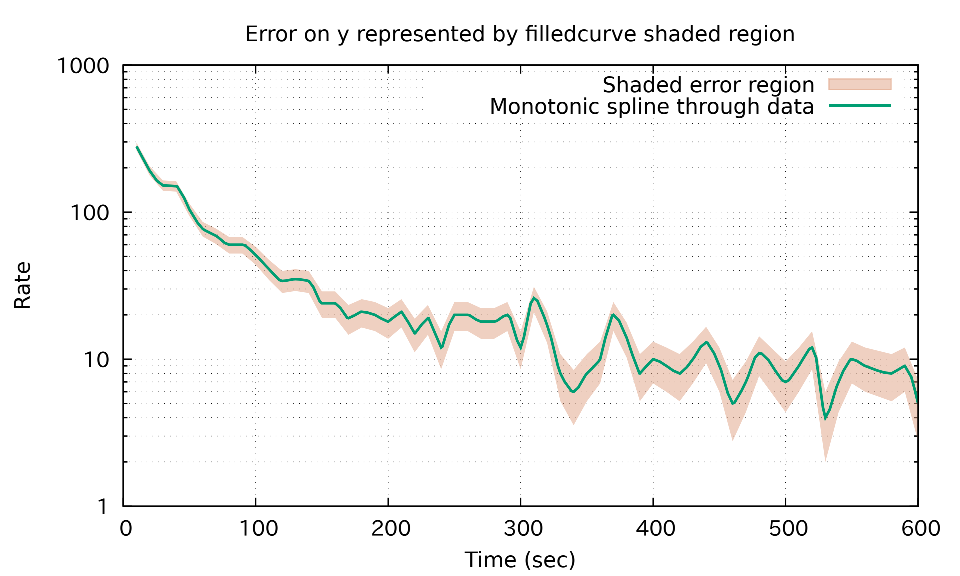
xyerrorbars in log scale
Power
Theory
Power [W]
1
0.1
0.01
1
10
Resistance [Ohm]

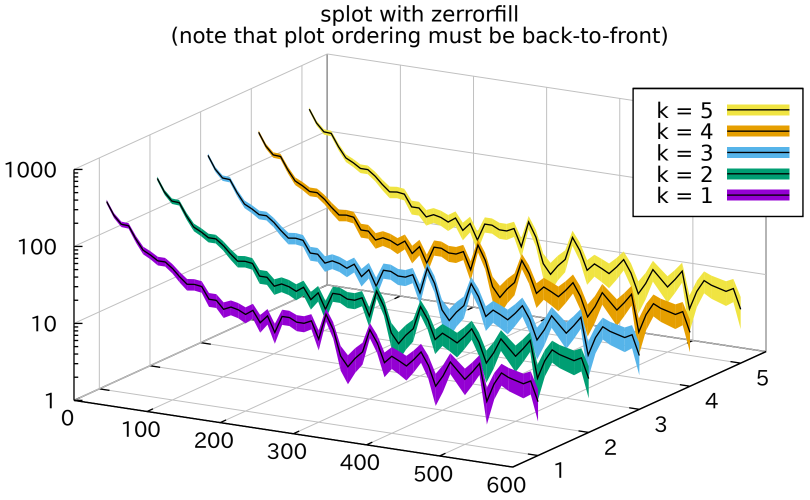

xyerrorbars with no crossbar
Power
Theory
Power [W]
Resistance [Ohm]
0.12
0.1
0.08
0.06
0.04
0.02
0
0
10
20
30
40
50

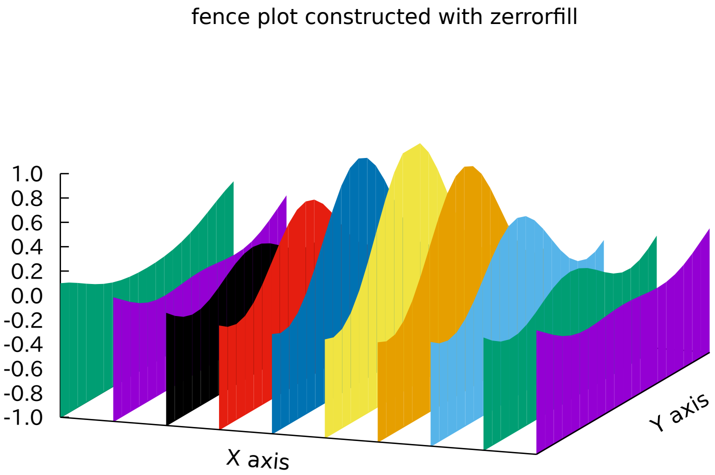


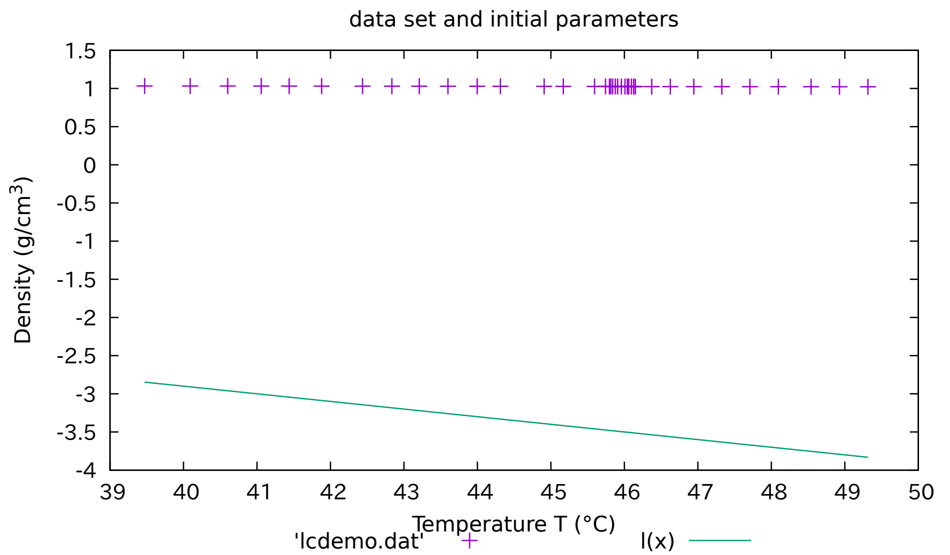
xerrorbars with no crossbar
Power
Theory
Power [W]
Resistance [Ohm]
0.12
0.1
0.08
0.06
0.04
0.02
0
0
10
20
30
40
50


yerrorbars with no crossbar
Power
Theory
Power [W]
Resistance [Ohm]
0
0.02
0.04
0.06
0.08
0.1
0.12
0
10
20
30
40
50

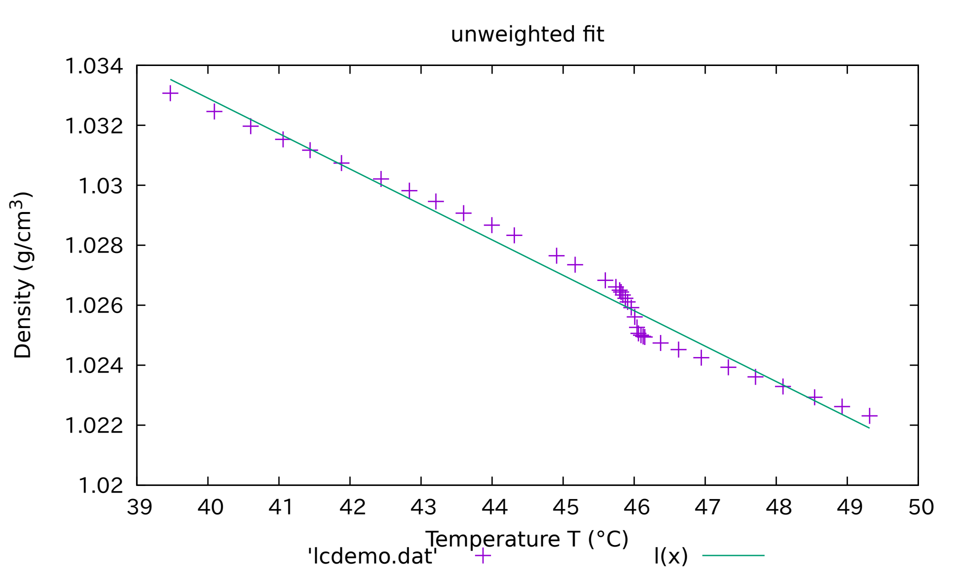

Error on y represented by filledcurve shaded region
Shaded error region
Monotonic spline through data
Rate
Time (sec)
1000
100
10
1
0
100
200
300
400
500
600

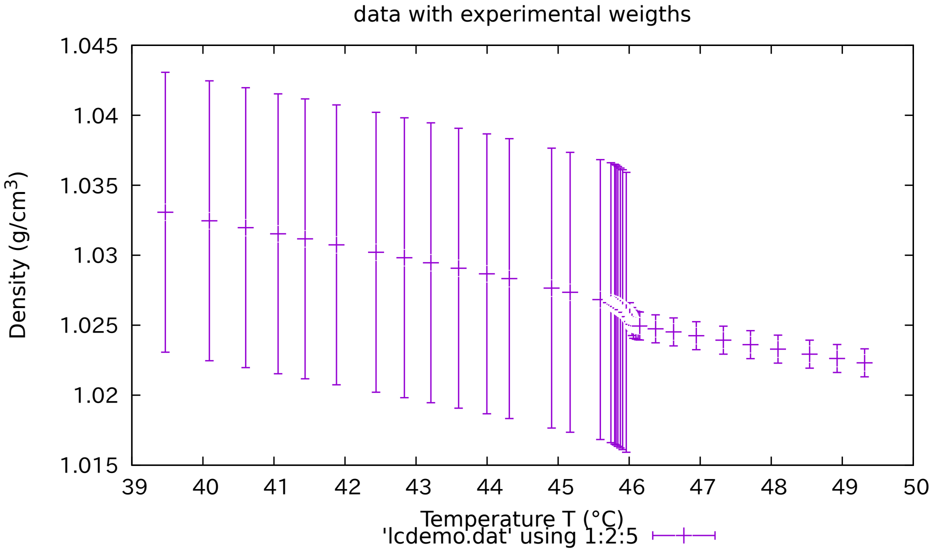

splot with zerrorfill
(note that plot ordering must be back-to-front)
k = 5
k = 4
k = 3
k = 2
k = 1
1000
100
10
1
0
100
200
300
400
500
600
1
2
3
4
5


fence plot constructed with zerrorfill
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
X axis
Y axis

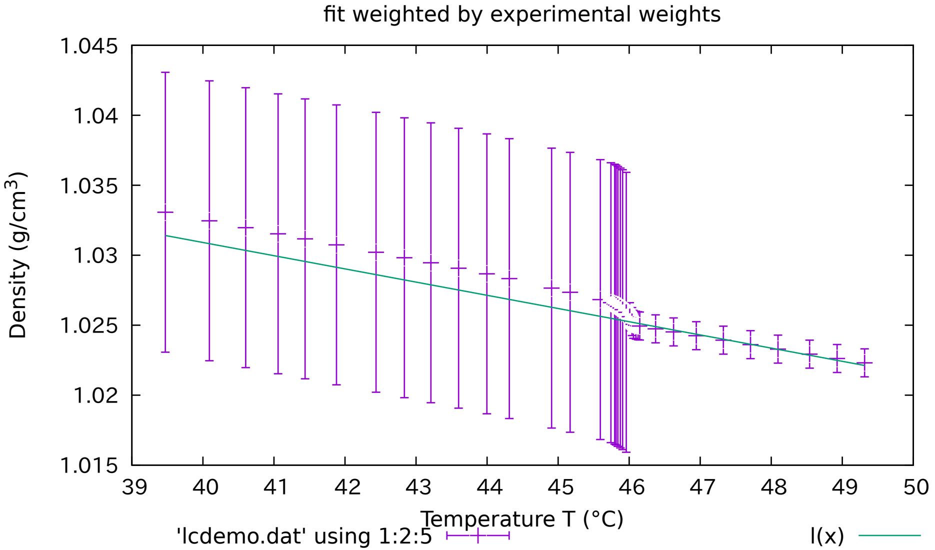


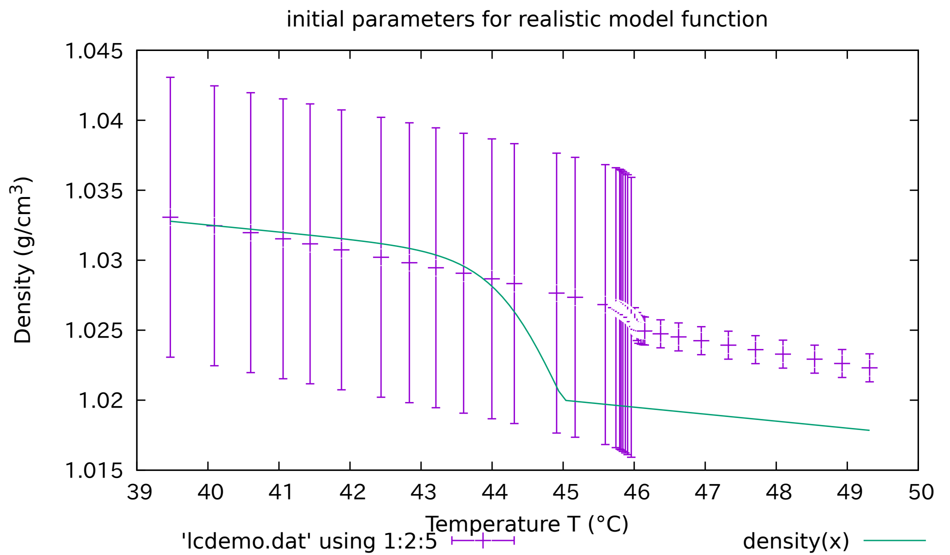
data set and initial parameters
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
Density (g/cm^3)
39
40
41
42
43
44
45
46
47
48
49
50
Temperature T (°C)
'lcdemo.dat'
l(x)


unweighted fit
1.034
1.032
1.03
1.028
1.026
1.024
1.022
1.02
39
40
41
42
43
44
45
46
47
48
49
50
Density (g/cm^3)
Temperature T (°C)
'lcdemo.dat'
l(x)

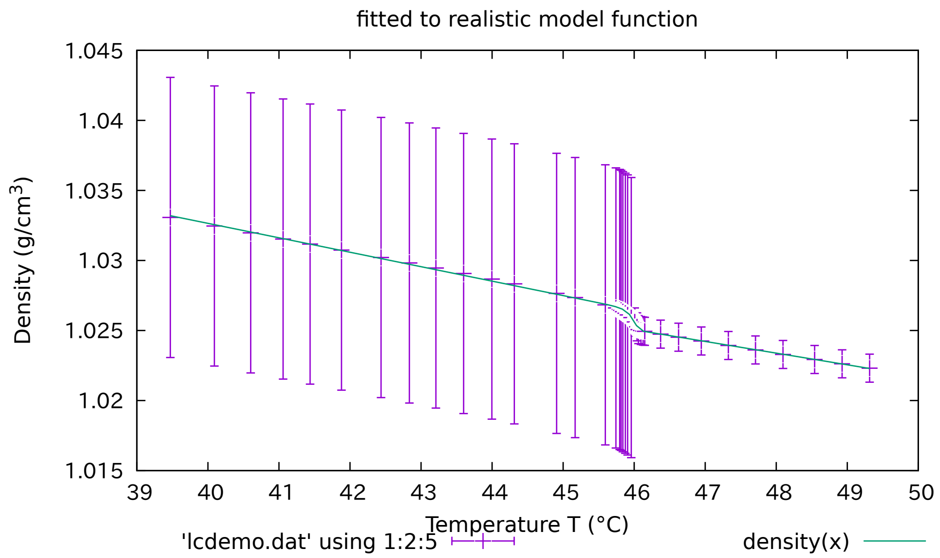


data with experimental weigths
Density (g/cm^3)
Temperature T (°C)
'lcdemo.dat' using 1:2:5
1.045
1.04
1.035
1.03
1.025
1.02
1.015
39
40
41
42
43
44
45
46
47
48
49
50

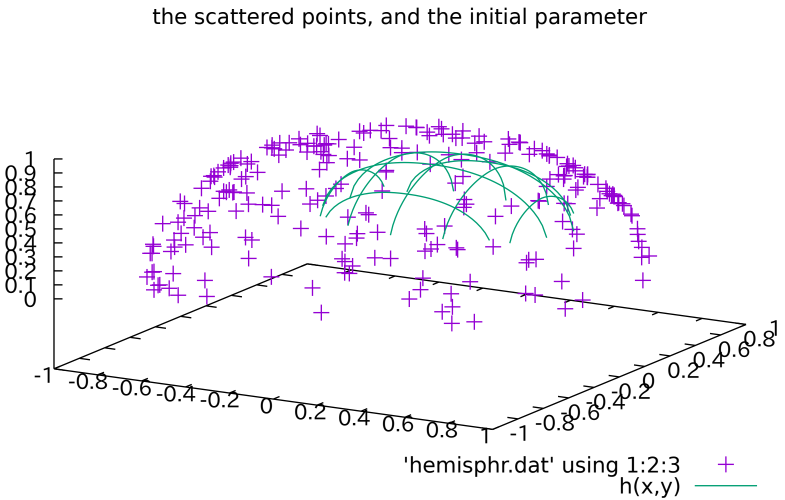

fit weighted by experimental weights
Density (g/cm3)
1.045
1.04
1.035
1.03
1.025
1.02
1.015
39
40
41
42
43
44
45
46
47
48
49
50
Temperature T (°C)
'lcdemo.dat' using 1:2:5
l(x)

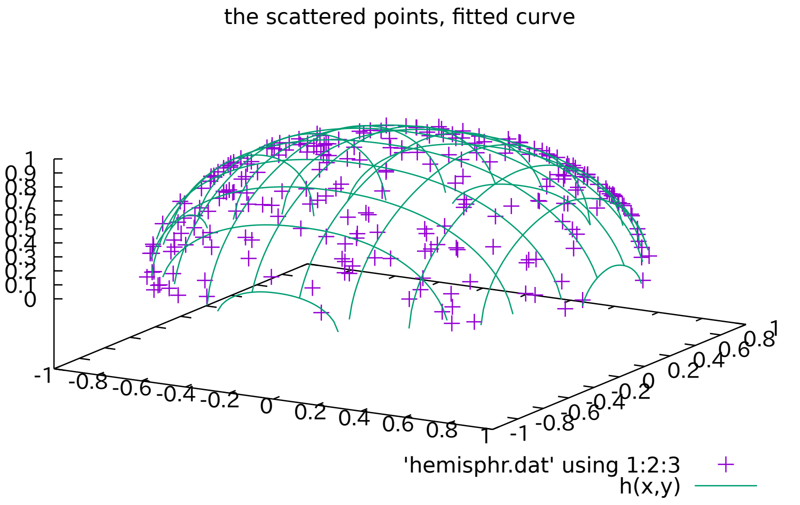


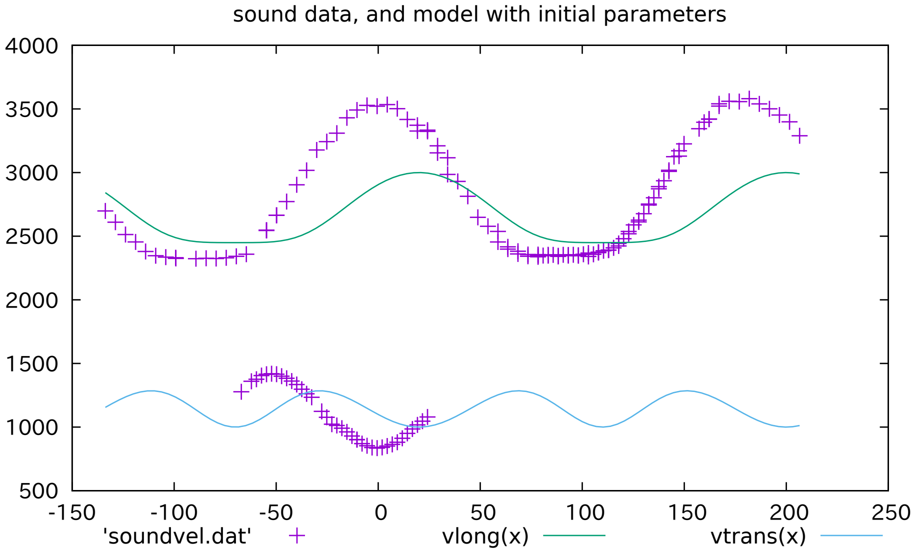
initial parameters for realistic model function
1.045
1.04
1.035
1.03
1.025
1.02
1.015
Density (g/cm^3)
39
40
41
42
43
44
45
46
47
48
49
50
Temperature T (°C)
'lcdemo.dat' using 1:2:5
density(x)

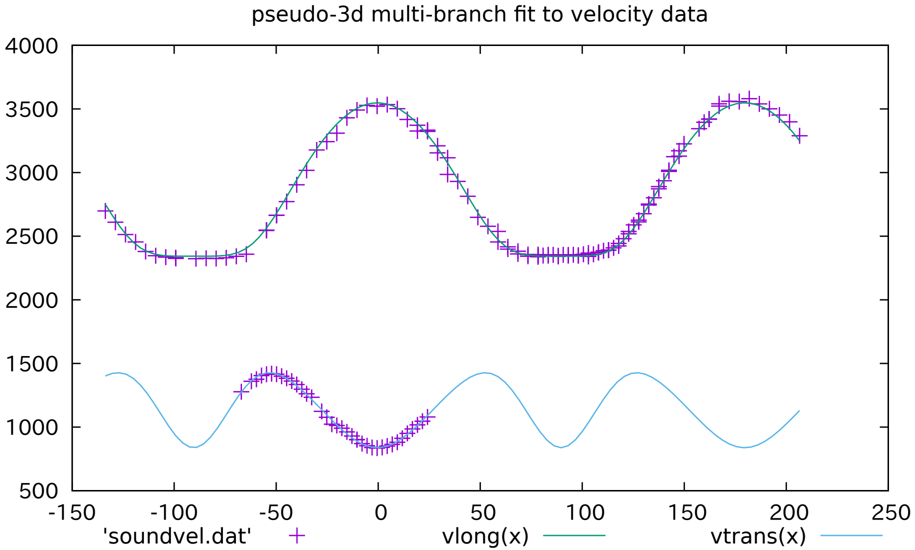


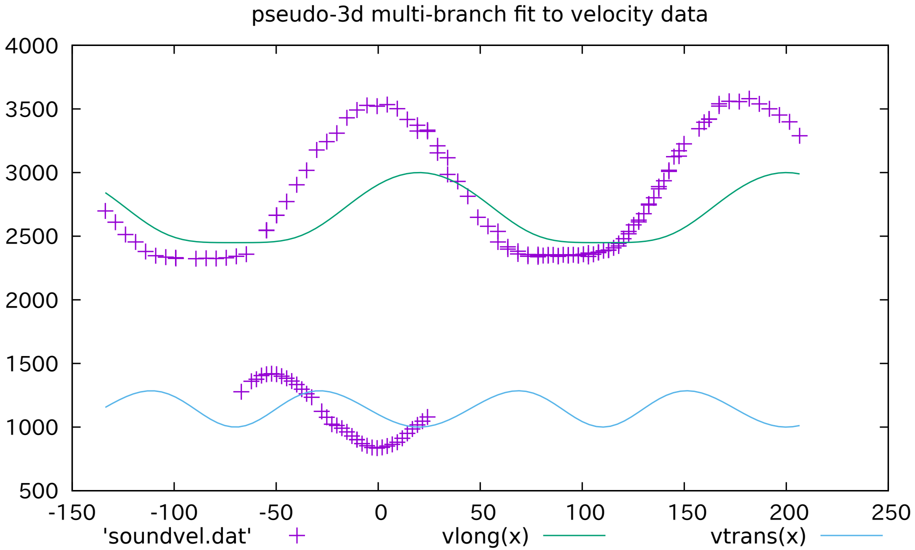
fitted to realistic model function
1.045
1.04
1.035
1.03
1.025
1.02
1.015
Density (g/cm3)
39
40
41
42
43
44
45
46
47
48
49
50
Temperature T (°C)
'lcdemo.dat' using 1:2:5
density(x)


the scattered points, and the initial parameter
'hemisphr.dat' using 1:2:3
h(x,y)

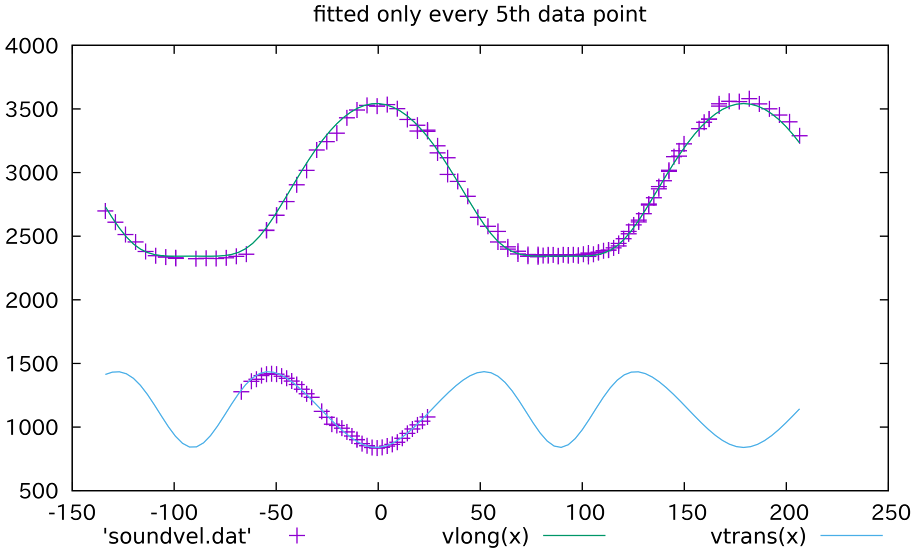

the scattered points, fitted curve
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
'hemisphr.dat' using 1:2:3
h(x,y)


sound data, and model with initial parameters
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)

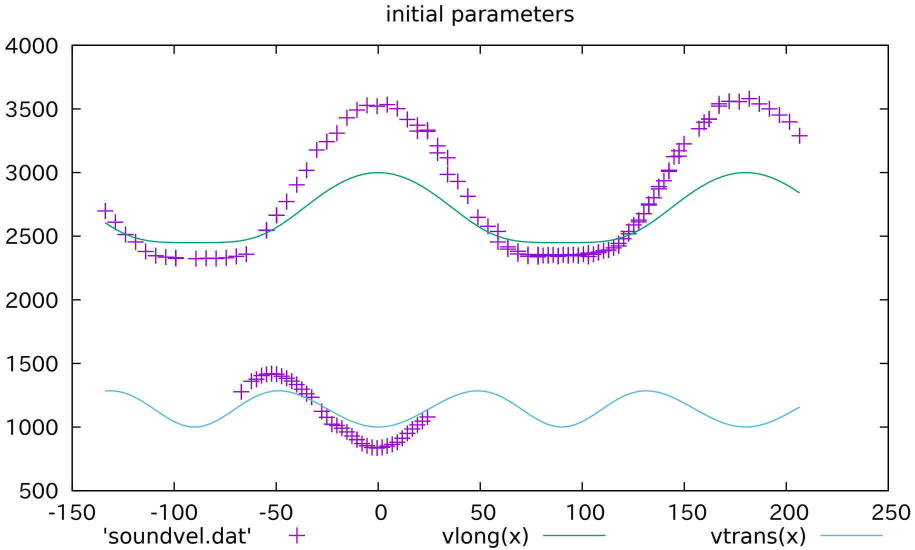

pseudo-3d multi-branch fit to velocity data
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)



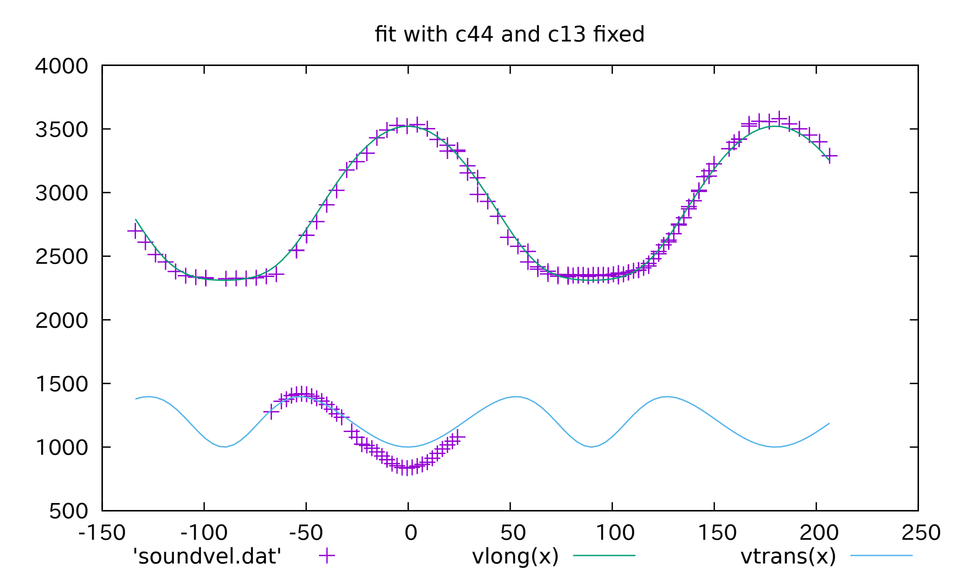
pseudo-3d multi-branch fit to velocity data
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)



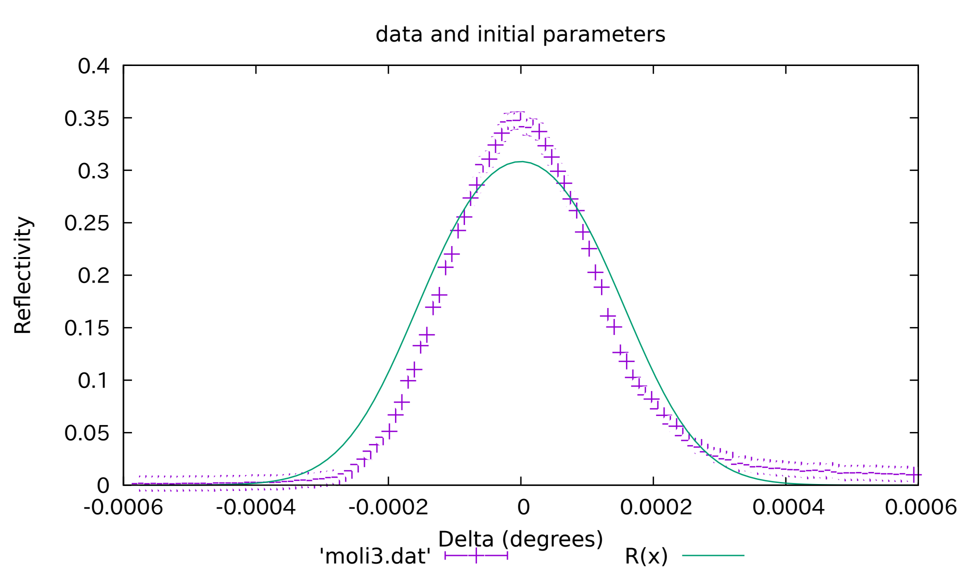
fitted only every 5th data point
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)

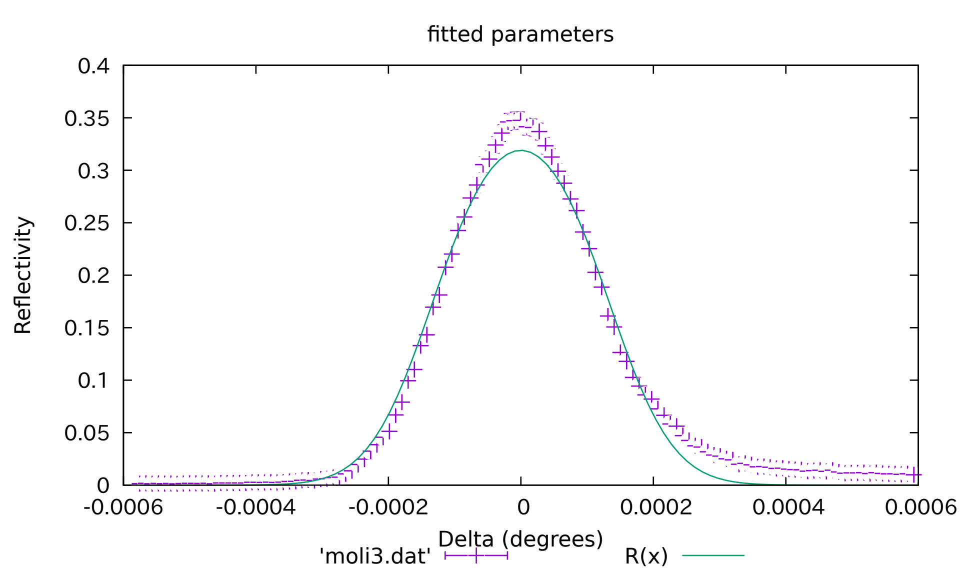


initial parameters
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)

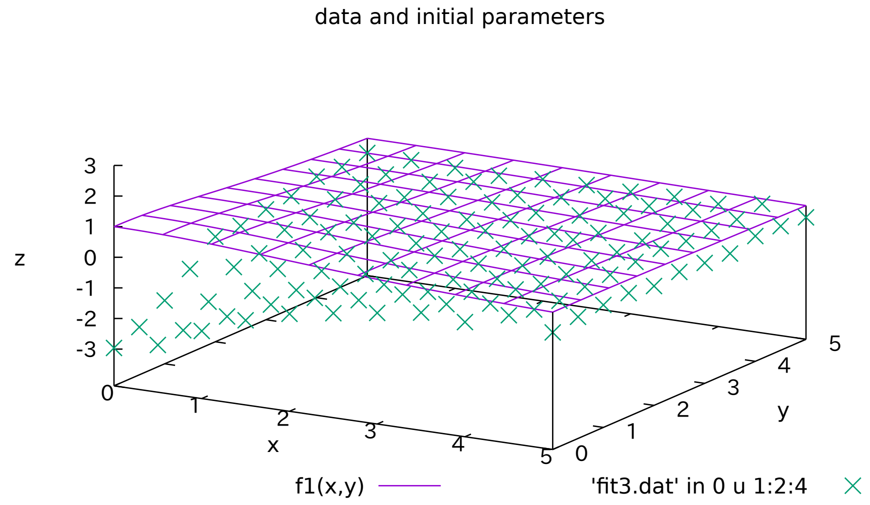


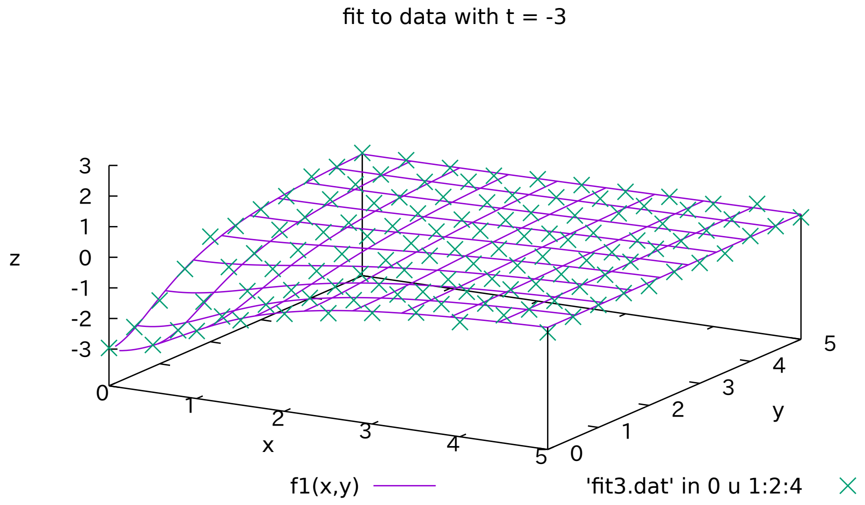
fit with c44 and c13 fixed
4000
3500
3000
2500
2000
1500
1000
500
-150
-100
-50
0
50
100
150
200
250
'soundvel.dat'
vlong(x)
vtrans(x)


data and initial parameters
Reflectivity
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.0006
-0.0004
-0.0002
0
0.0002
0.0004
0.0006
Delta (degrees)
'moli3.dat'
R(x)

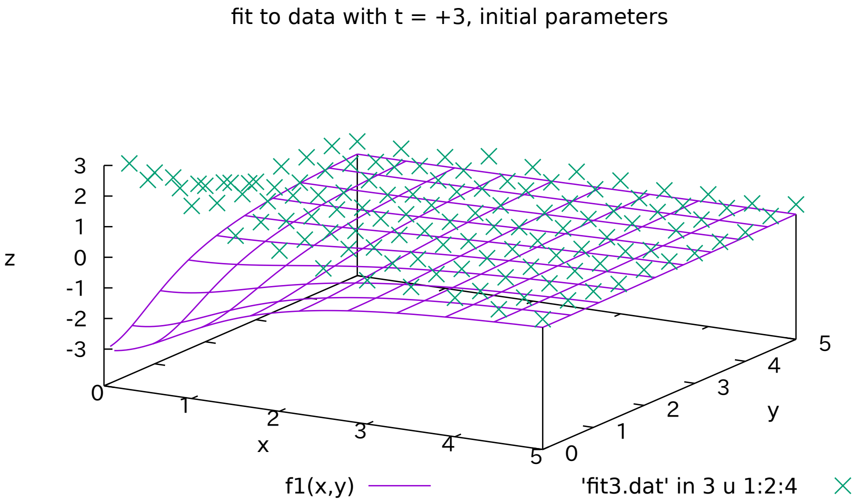


fitted parameters
Reflectivity
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.0006
-0.0004
-0.0002
0
0.0002
0.0004
0.0006
Delta (degrees)
'moli3.dat'
R(x)

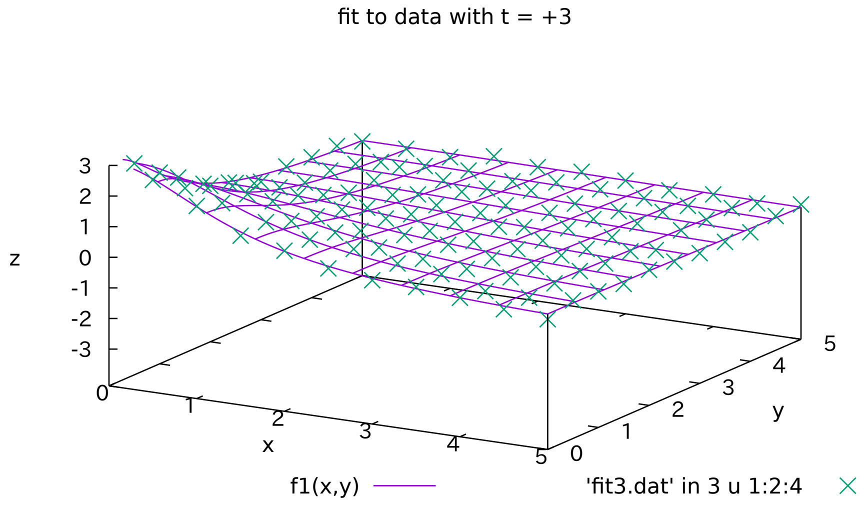

data and initial parameters
z
x
y
f1(x,y)
'fit3.dat' in 0 u 1:2:4

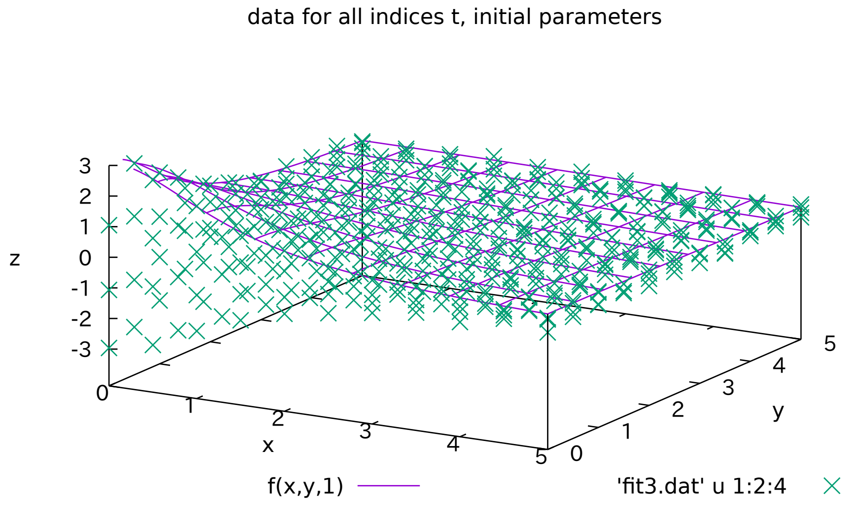


fit to data with t = -3
z
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
x
0
1
2
3
4
5
y
f1(x,y)
'fit3.dat' in 0 u 1:2:4

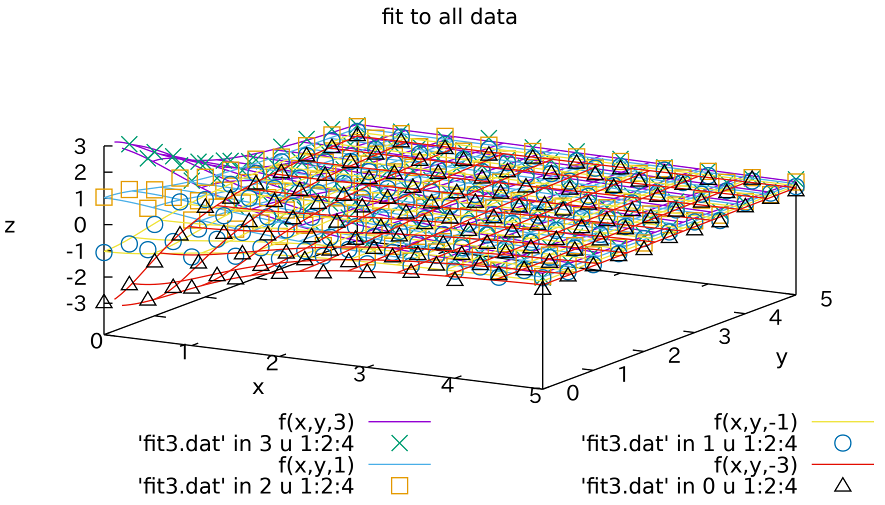


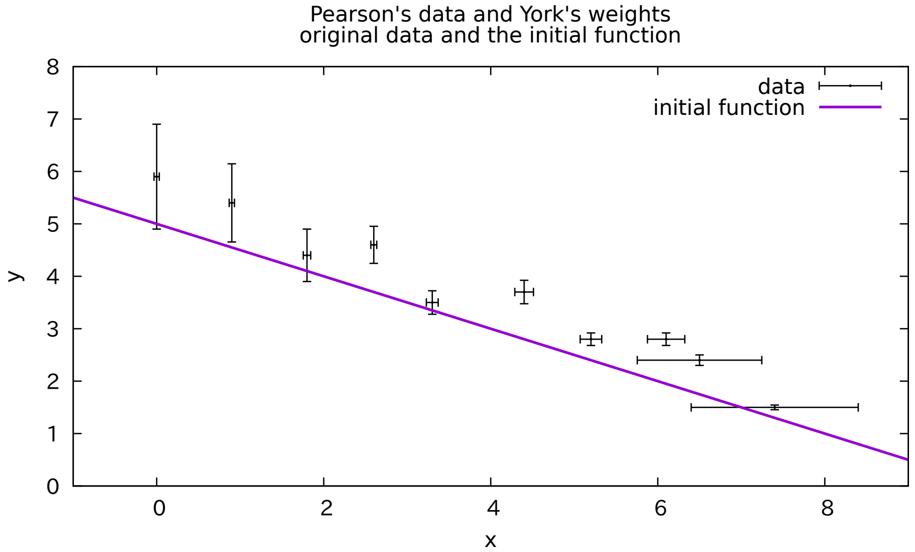
fit to data with t = +3, initial parameters
z
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
x
0
1
2
3
4
5
y
f1(x,y)
'fit3.dat' in 3 u 1:2:4

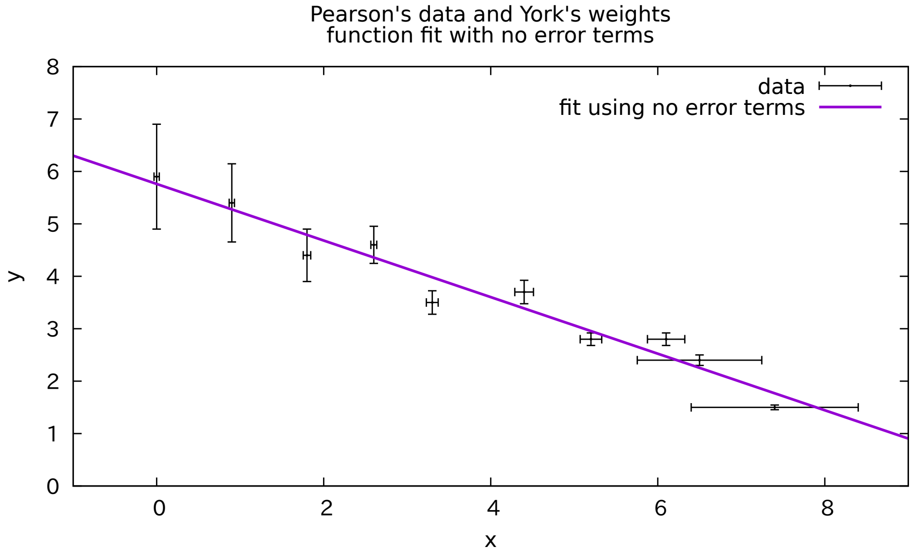

fit to data with t = +3
z
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
x
0
1
2
3
4
5
y
f1(x,y)
'fit3.dat' in 3 u 1:2:4

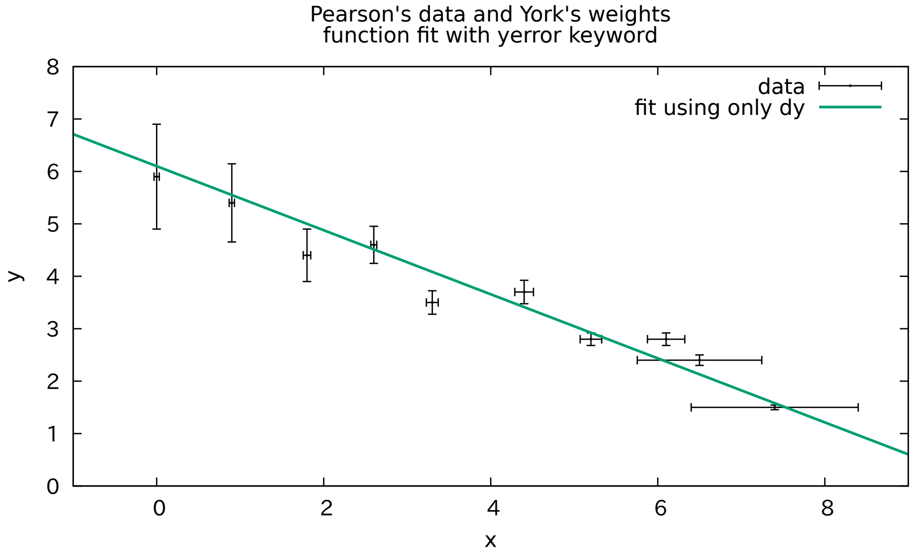


data for all indices t, initial parameters
z
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
x
0
1
2
3
4
5
y
f(x,y,1)
'fit3.dat' u 1:2:4

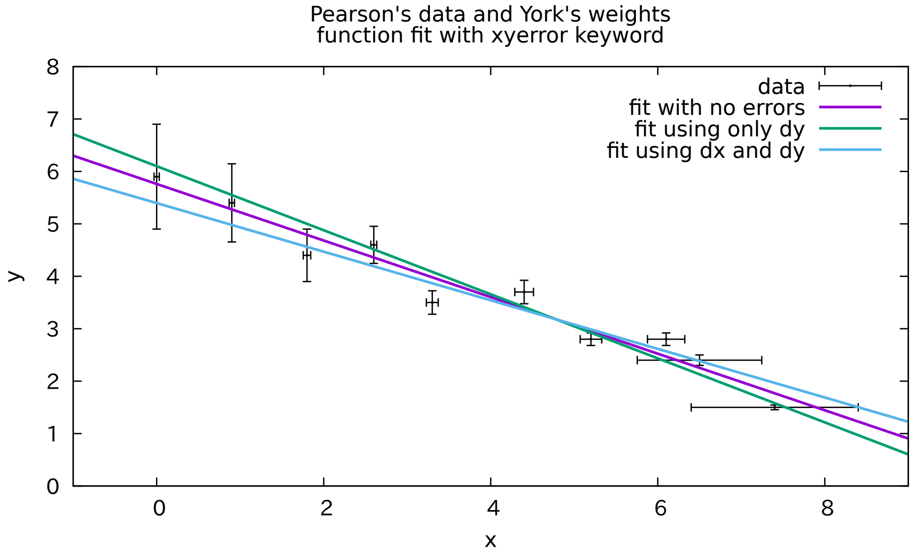

fit to all data
z
x
y
3
2
1
0
-1
-2
-3
0
1
2
3
4
5
0
1
2
3
4
5
f(x,y,3)
'fit3.dat' in 3 u 1:2:4
f(x,y,1)
'fit3.dat' in 2 u 1:2:4
f(x,y,-1)
'fit3.dat' in 1 u 1:2:4
f(x,y,-3)
'fit3.dat' in 0 u 1:2:4

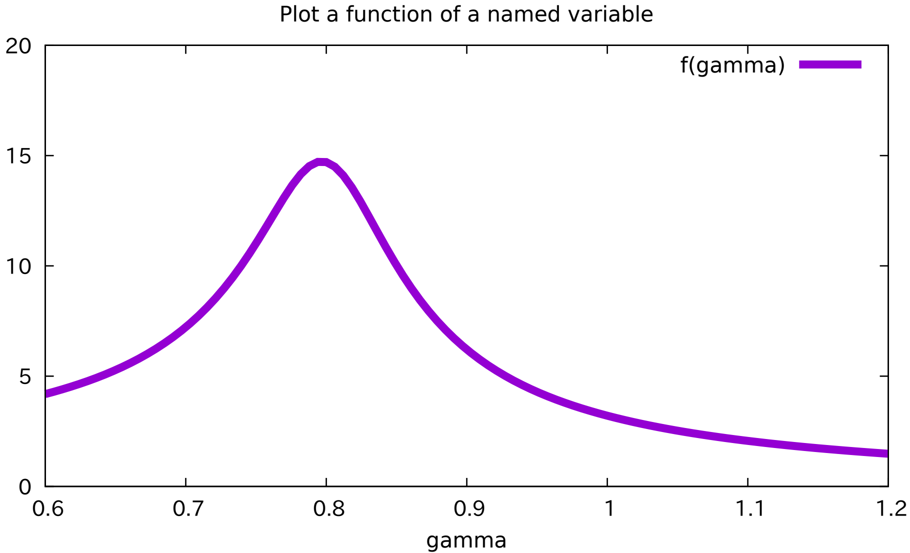

Pearson's data and York's weights
original data and the initial function
data
initial function
y
x
0
1
2
3
4
5
6
7
8
0
2
4
6
8

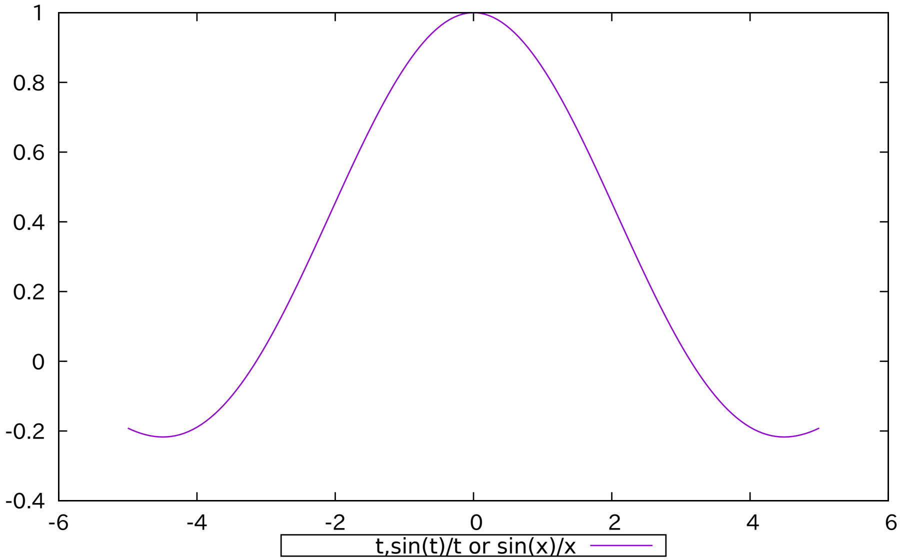

Pearson's data and York's weights
function fit with no error terms
data
fit using no error terms
y
x
0
1
2
3
4
5
6
7
8
0
2
4
6
8

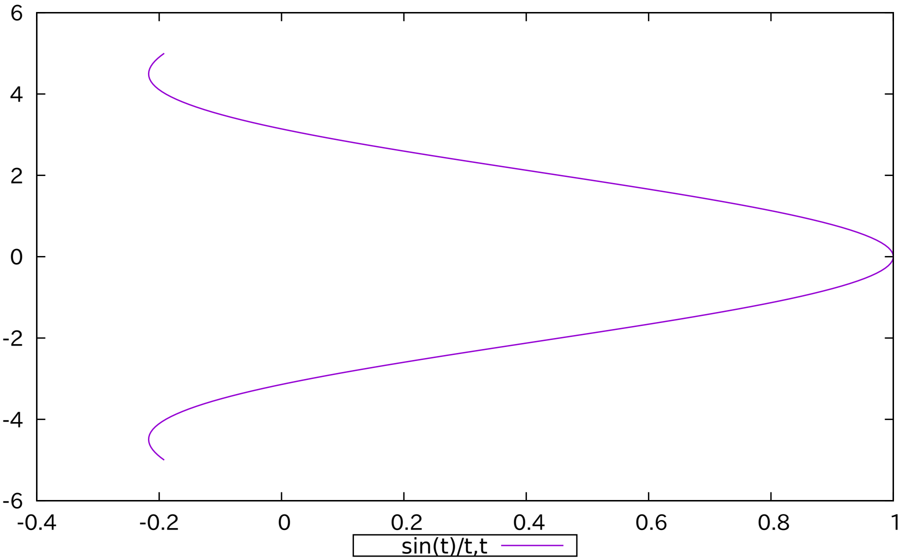


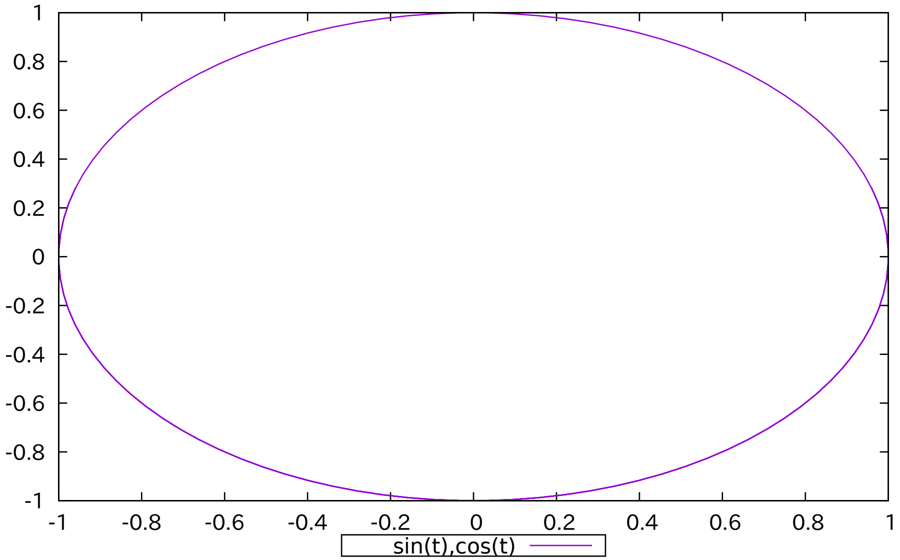
Pearson's data and York's weights
function fit with yerror keyword
data
fit using only dy
y
x
0
1
2
3
4
5
6
7
8
0
2
4
6
8

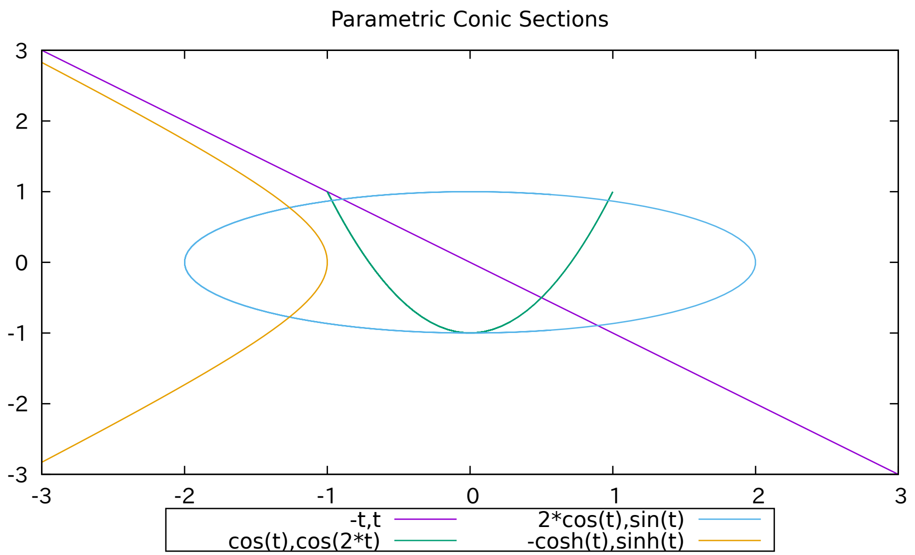

Pearson's data and York's weights
function fit with xyerror keyword
data
fit with no errors
fit using only dy
fit using dx and dy
y
x
0
1
2
3
4
5
6
7
8
0
2
4
6
8

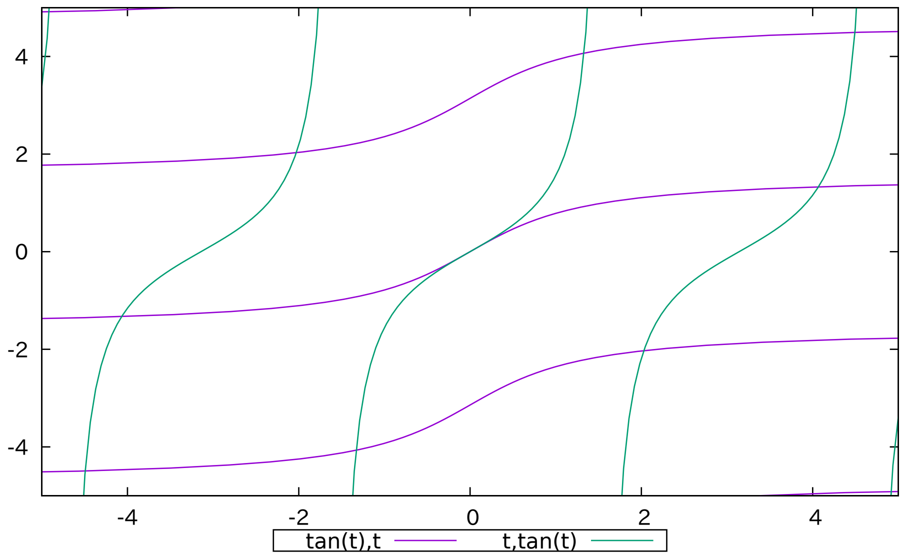

Plot a function of a named variable
f(gamma)
20
15
10
5
0
0.6
0.7
0.8
0.9
1
1.1
1.2
gamma

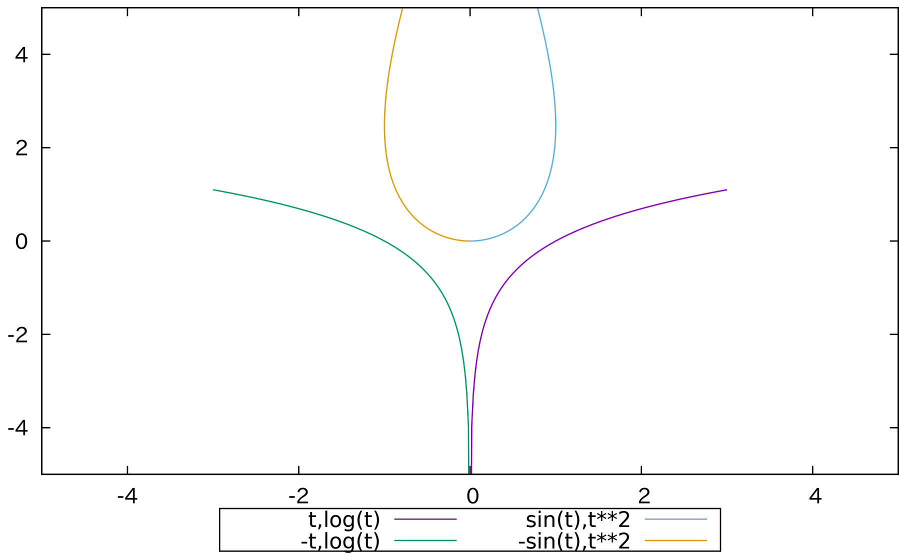

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-6
-4
-2
0
2
4
6
t,sin(t)/t or sin(x)/x


6
4
2
0
-2
-4
-6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
sin(t)/t,t

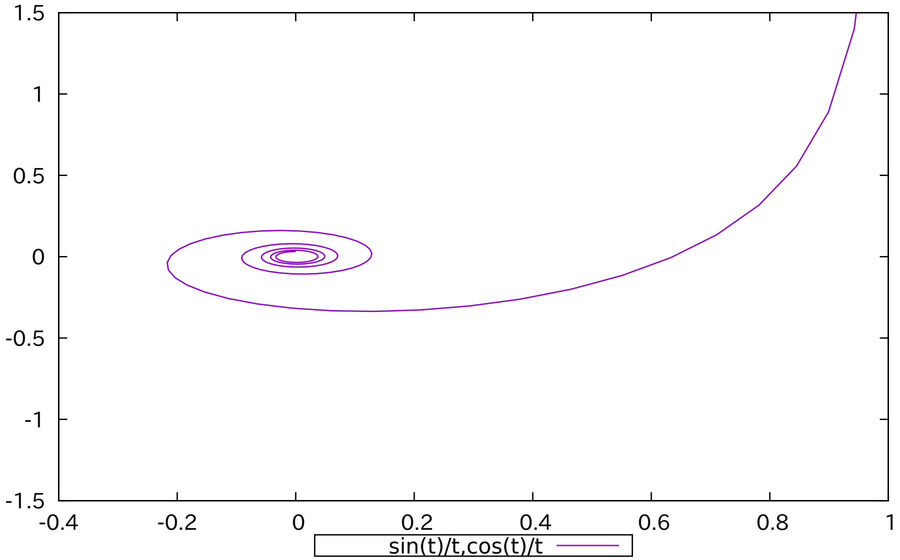

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
sin(t),cos(t)


Parametric Conic Sections
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
-t,t
2*cos(t),sin(t)
cos(t),cos(2*t)
-cosh(t),sinh(t)


4
2
0
-2
-4
-4
-2
0
2
4
tan(t),t
t,tan(t)

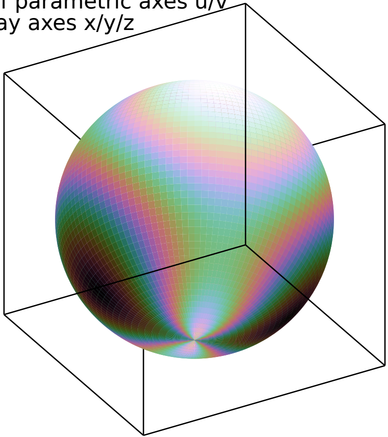


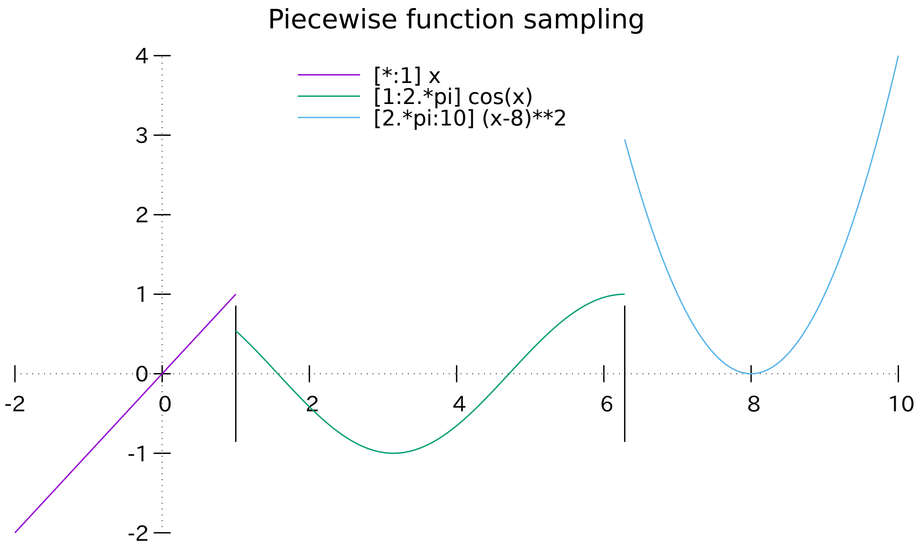
4
2
0
-2
-4
-4
-2
0
2
4
t,log(t)
-t,log(t)
sin(t),t**2
-sin(t),t**2

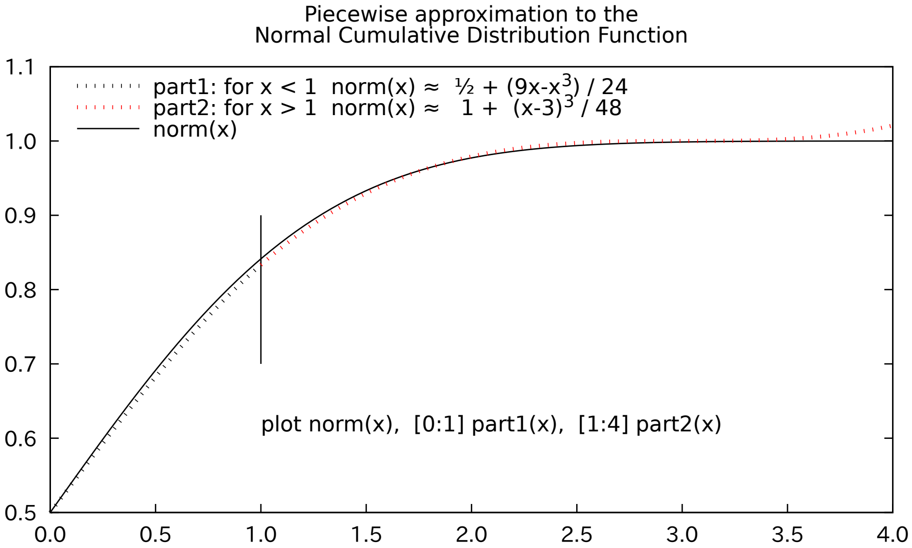


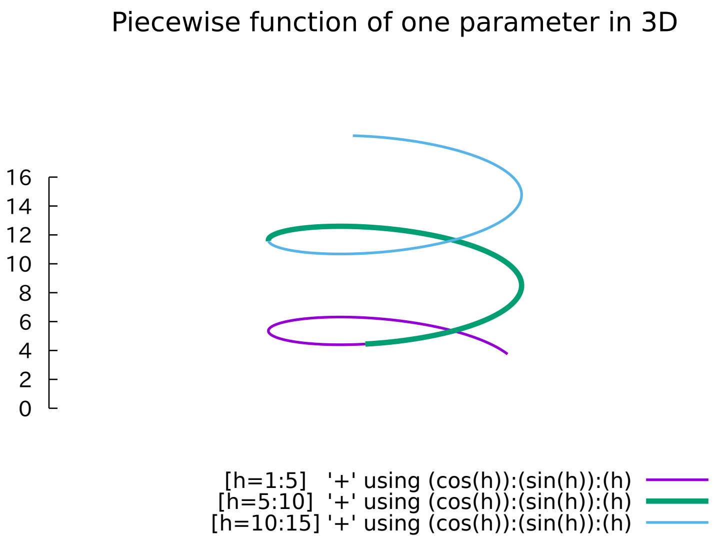
1.5
1
0.5
0
-0.5
-1
-1.5
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
sin(t)/t,cos(t)/t

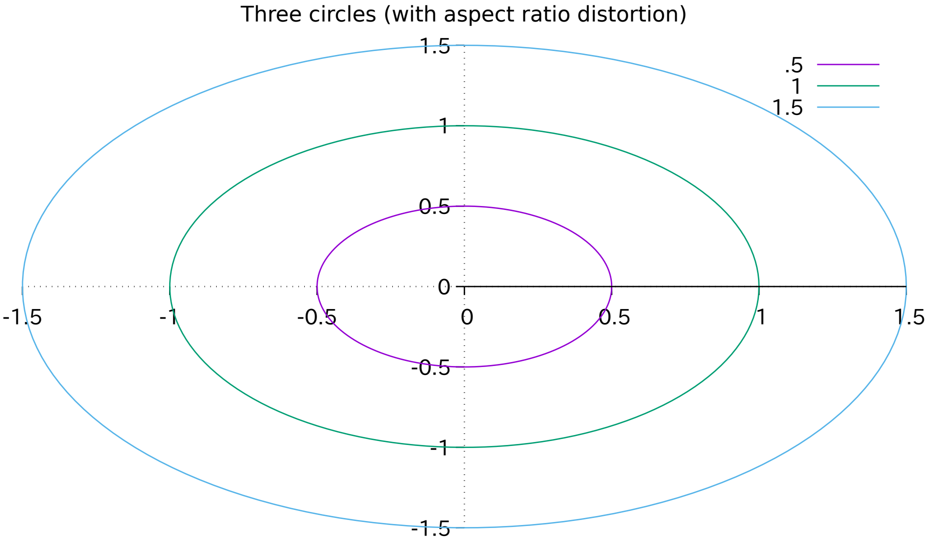
Decouple range of parametric axes u/v
from that of display axes x/y/z

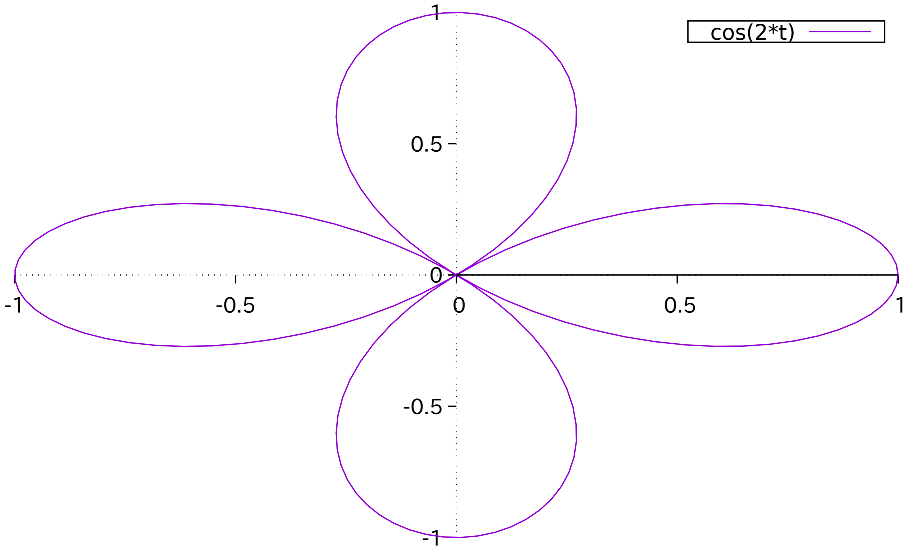



Piecewise function sampling
[*:1] x
[1:2.*pi] cos(x)
[2.*pi:10] (x-8)**2
4
3
2
1
0
-1
-2
-2
0
2
4
6
8
10

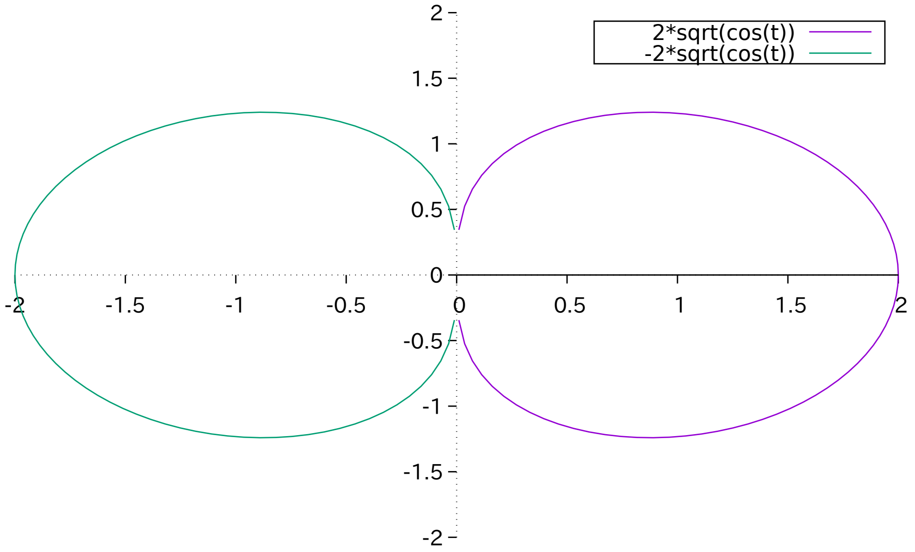

Piecewise approximation to the
Normal Cumulative Distribution Function
part1: for x < 1 norm(x) ≈ ½ + (9x-x³) / 24
part2: for x > 1 norm(x) ≈ 1 + (x-3)³ / 48
norm(x)
plot norm(x), [0:1] part1(x), [1:4] part2(x)
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

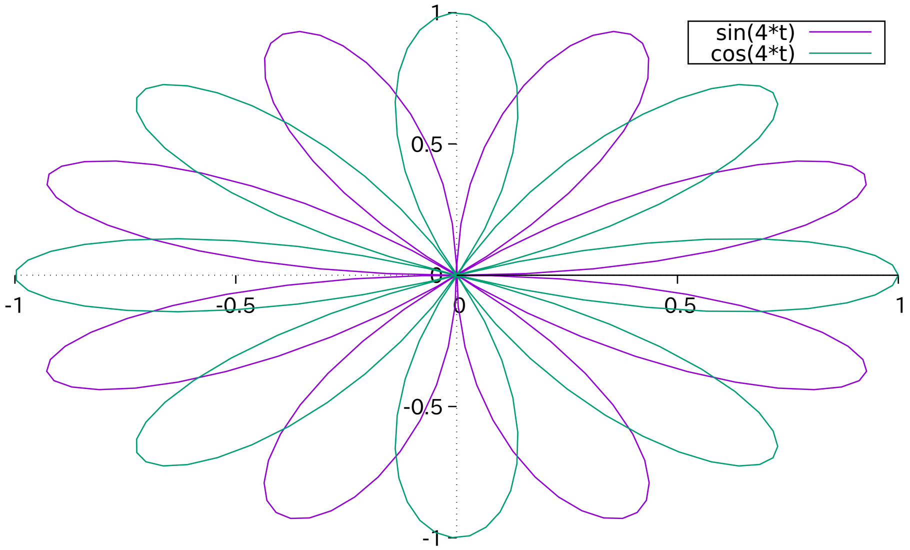

Piecewise function of one parameter in 3D
16
14
12
10
8
6
4
2
0
[h=1:5] '+' using (cos(h)):(sin(h)):(h)
[h=5:10] '+' using (cos(h)):(sin(h)):(h)
[h=10:15] '+' using (cos(h)):(sin(h)):(h)


Three circles (with aspect ratio distortion)
.5
1
1.5
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

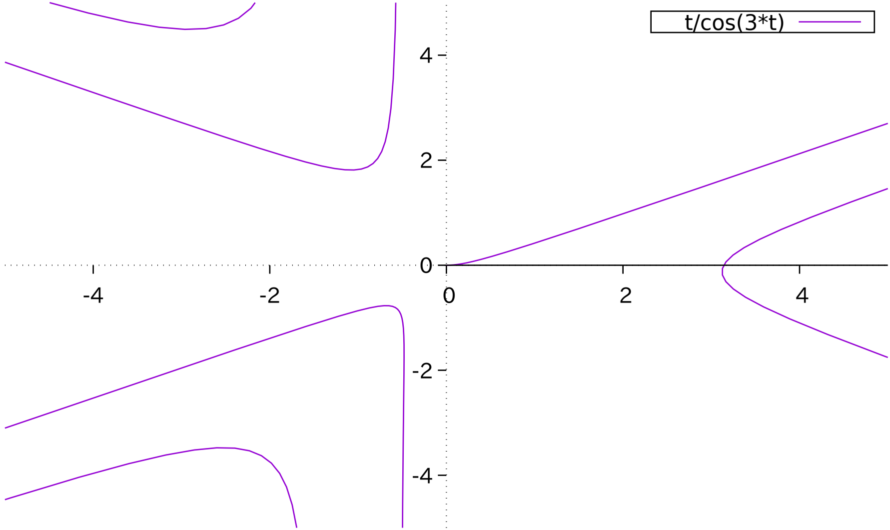


cos(2*t)
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

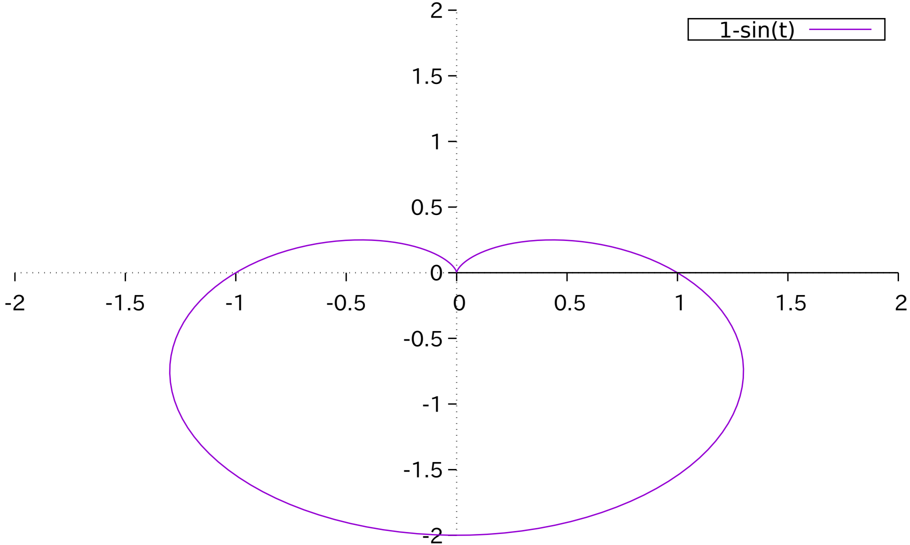

2*sqrt(cos(t))
-2*sqrt(cos(t))
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2



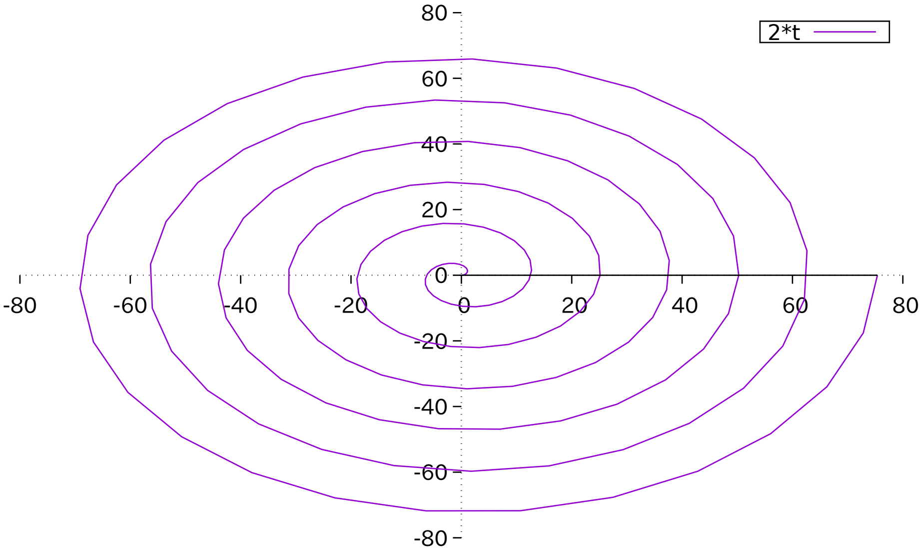
sin(4*t)
cos(4*t)
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1



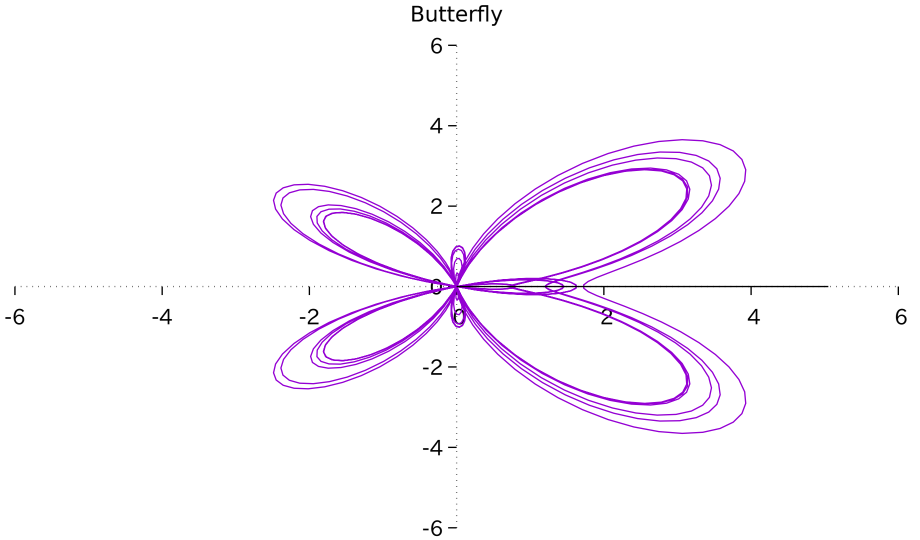
t/cos(3*t)
4
2
0
-2
-4
-4
-2
0
2
4

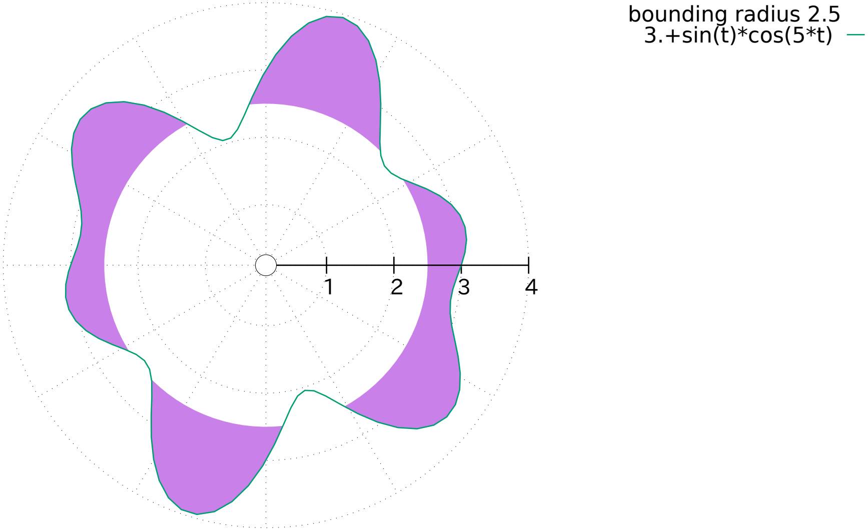


1-sin(t)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

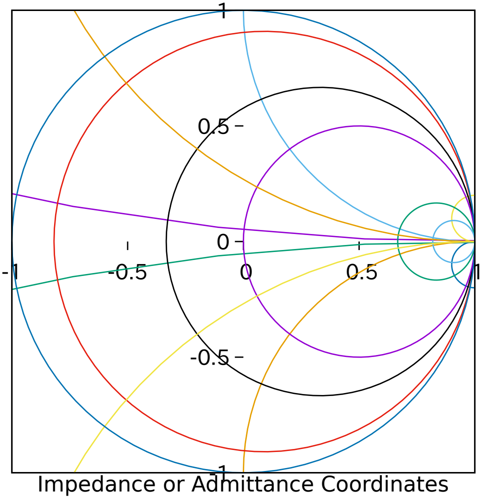


2*t
80
60
40
20
0
-20
-40
-60
-80
-80
-60
-40
-20
0
20
40
60
80

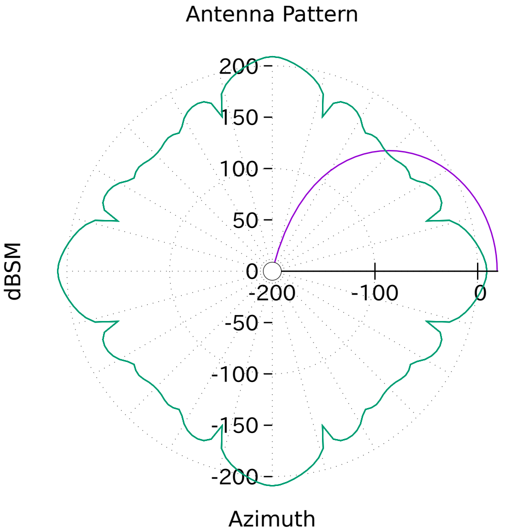

Butterfly
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6


bounding radius 2.5
3.+sin(t)*cos(5*t)
1
2
3
4

Primitive Smith Chart

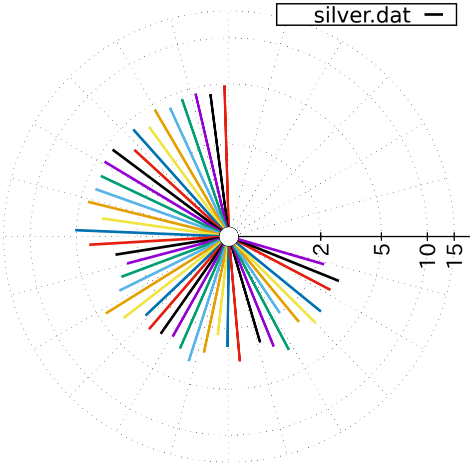


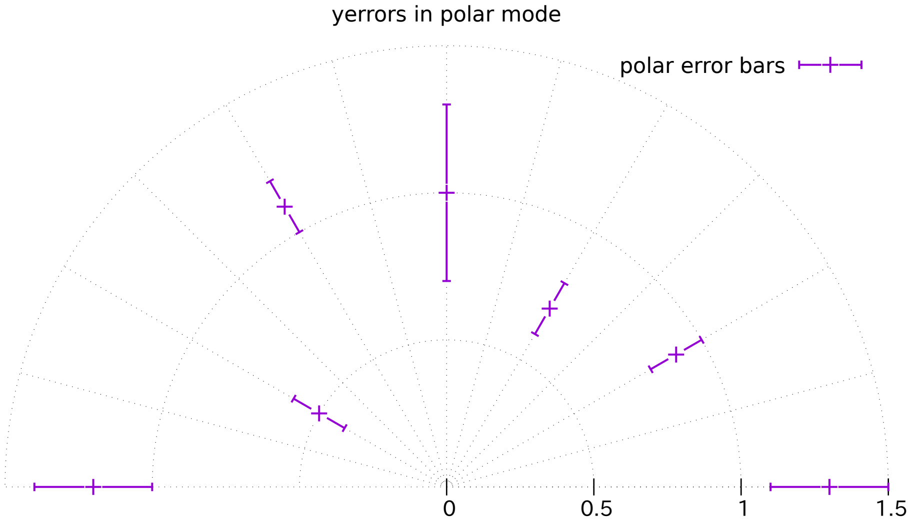
水 24 5月 2023


Antenna Pattern
200
150
100
50
0
-50
-100
-150
-200
-200
-100
0
dBSM
Azimuth

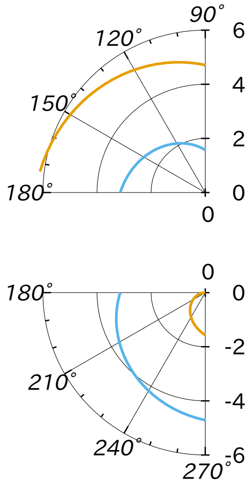
水 24 5月 2023

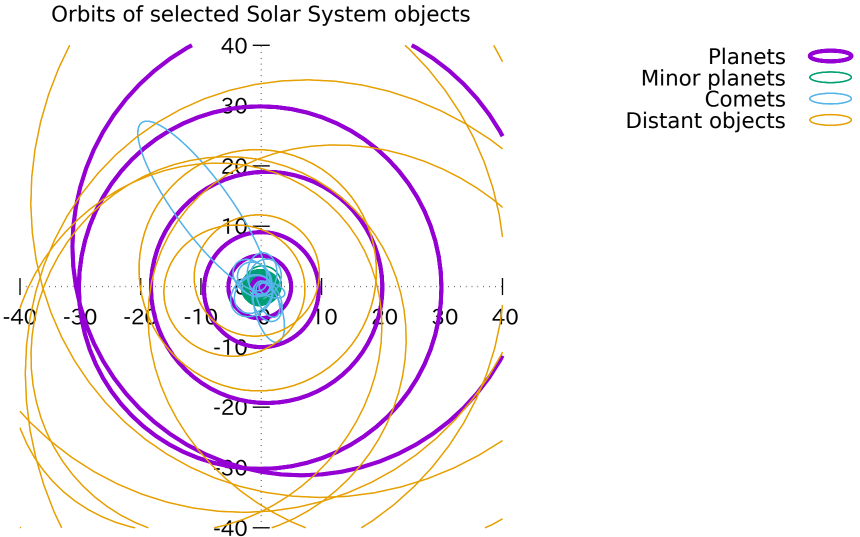
log scale polar axis, trange in degrees



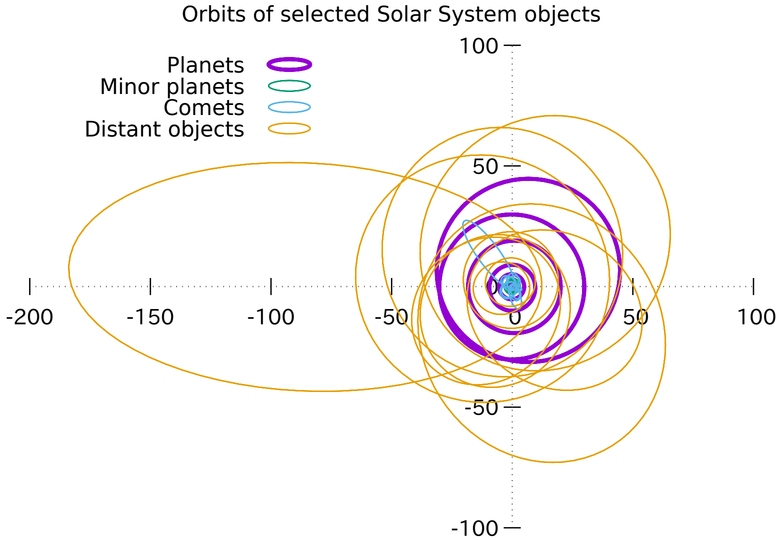

yerrors in polar mode
polar error bars
0
0.5
1
1.5

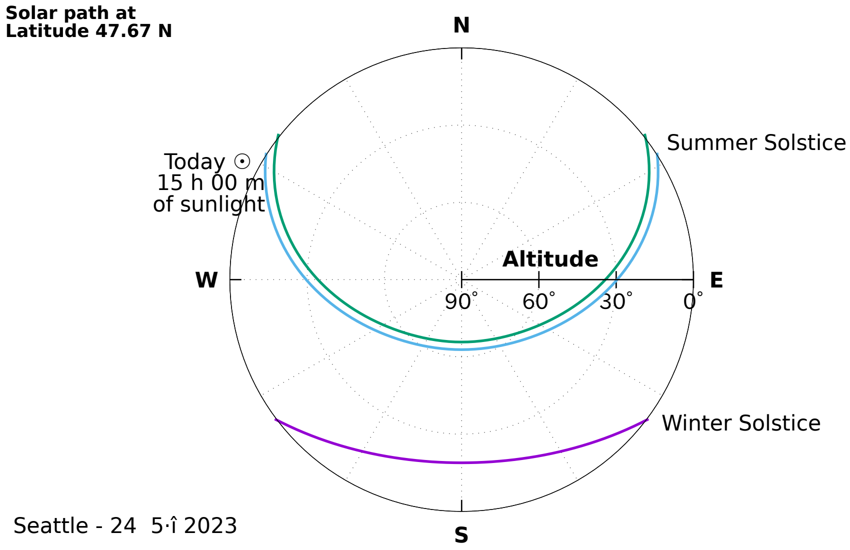
Polar Quadrants



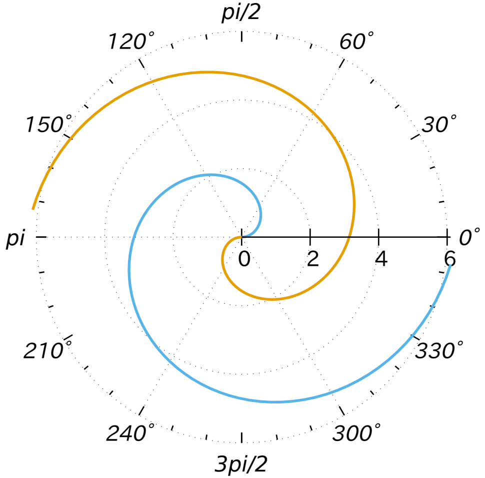


Orbits of selected Solar System objects
Planets
Minor planets
Comets
Distant objects
40
30
20
10
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20
30
40

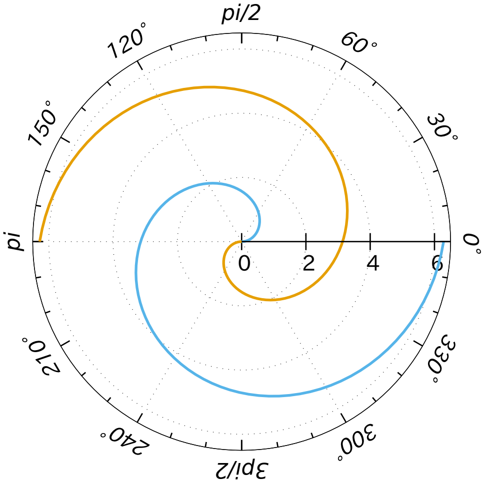

Orbits of selected Solar System objects
Planets
Minor planets
Comets
Distant objects
100
50
-50
-100
-200
-150
-100
-50
0
50
100

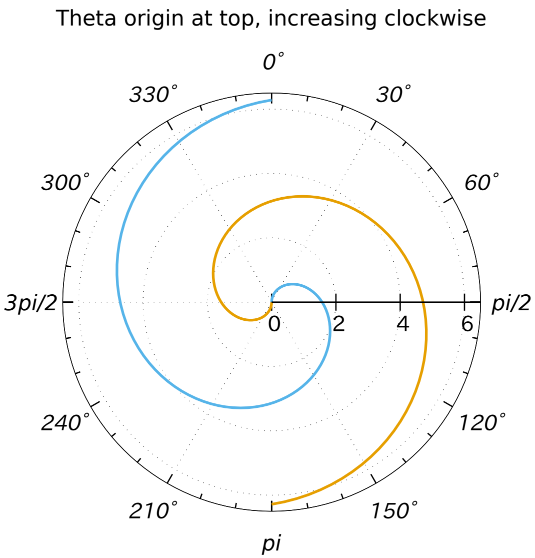

Solar path at
Latitude 47.67 N
N
Summer Solstice
Today ☉
15 h 00 m
of sunlight
W
Altitude
E
90°
60°
30°
0°
Winter Solstice
S
Seattle - 24 5·î 2023

Angle labels (ttics) for polar plots

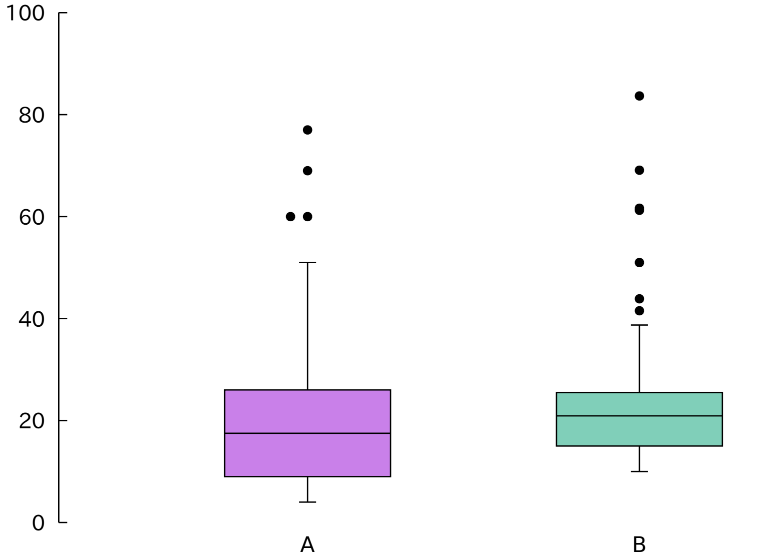


Polar plot with border and rotated labels for ttics

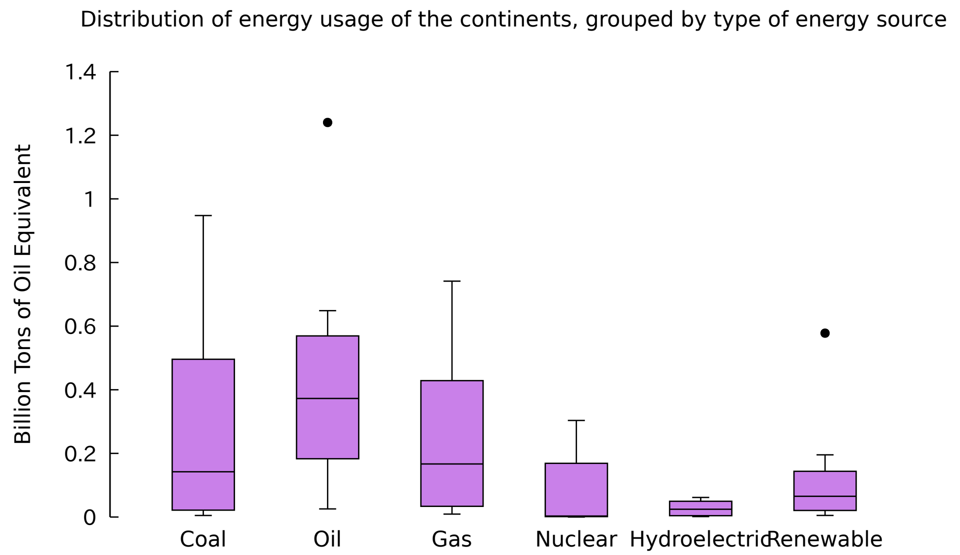




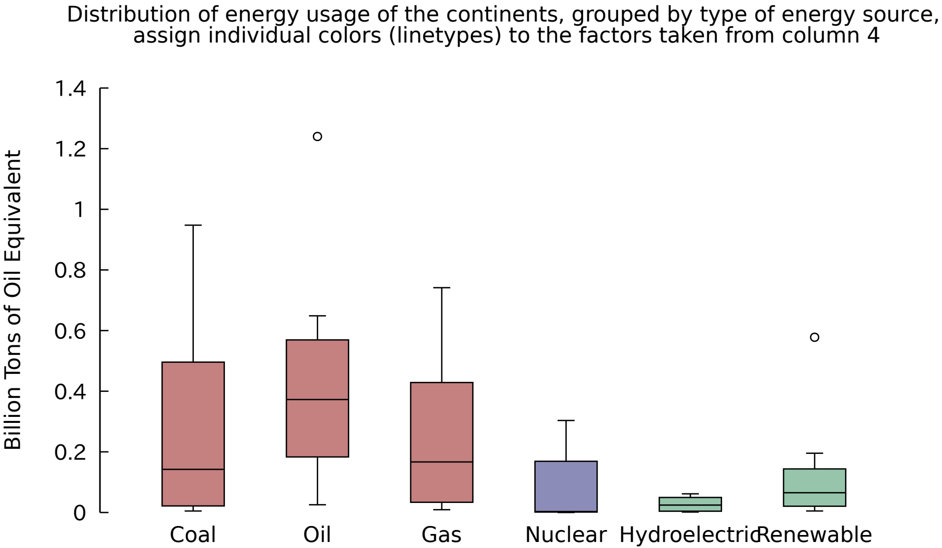
Theta origin at top, increasing clockwise
0°
30°
60°
pi/2
120°
150°
pi
210°
240°
3pi/2
300°
330°
0
2
4
6

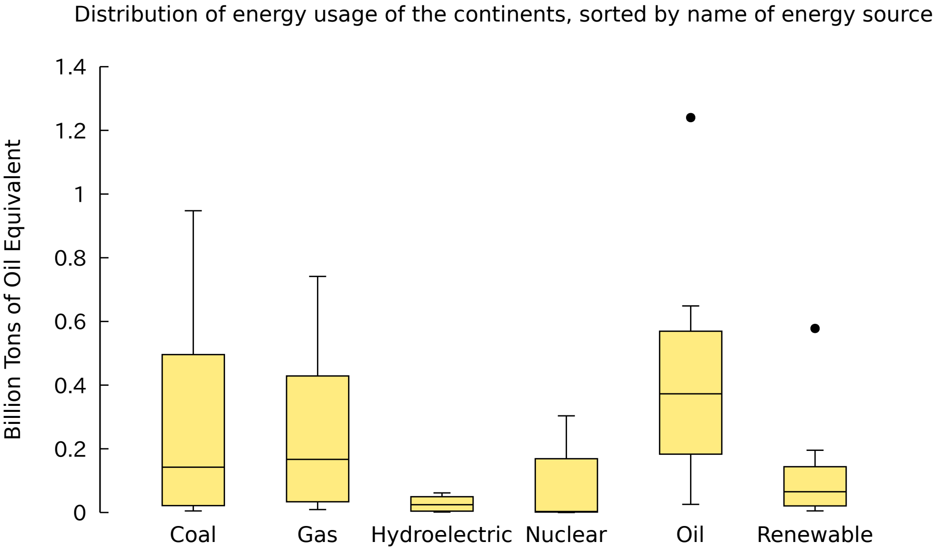


100
80
60
40
20
0
A
B

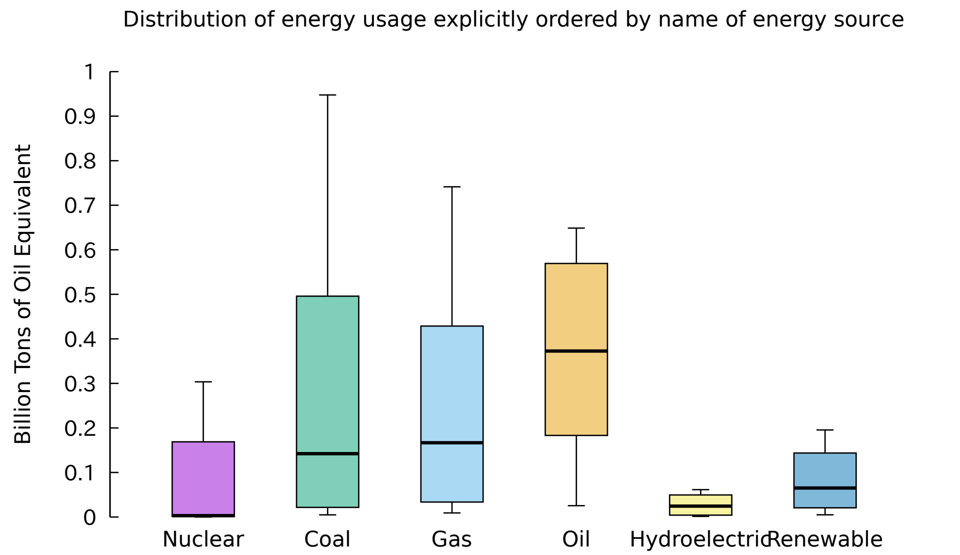


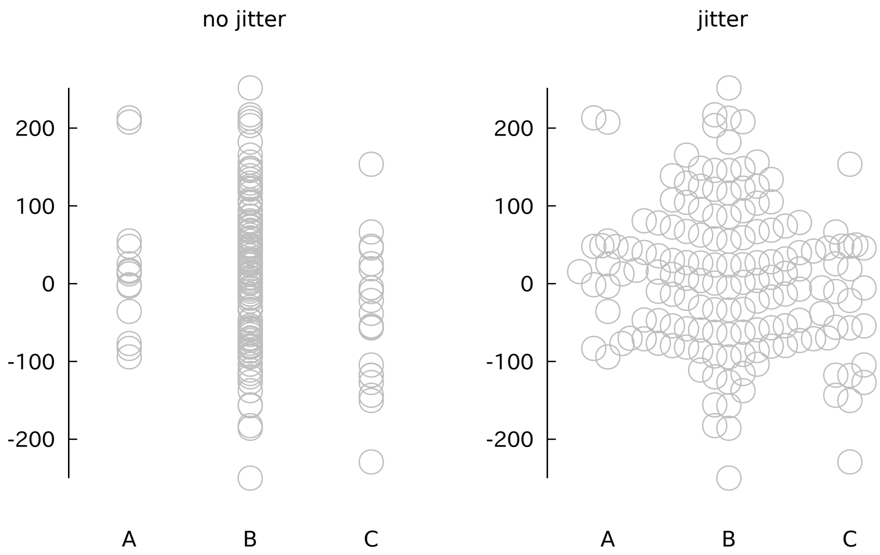
Distribution of energy usage of the continents, grouped by type of energy source
Billion Tons of Oil Equivalent
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Coal
Oil
Gas
Nuclear
Hydroelectric
Renewable

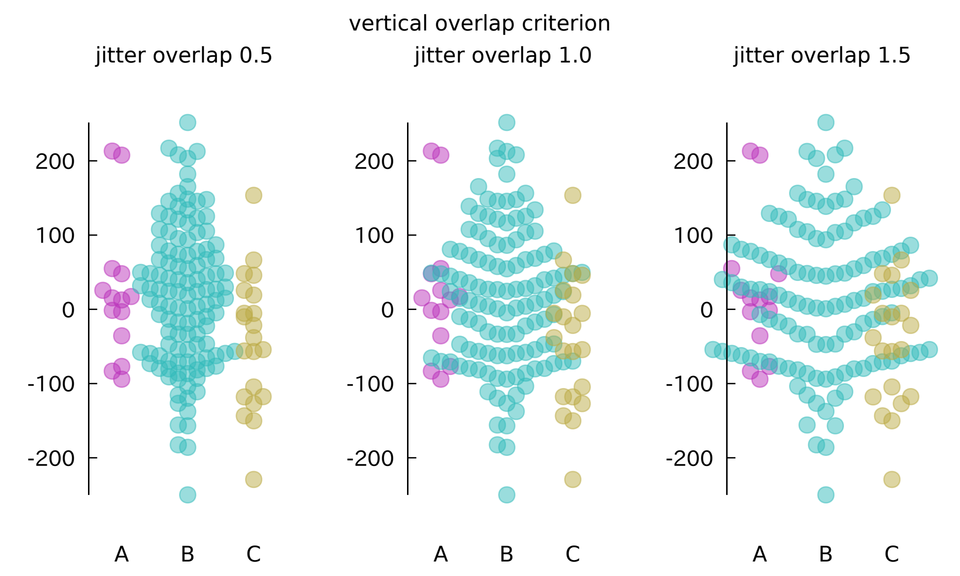

Distribution of energy usage of the continents, grouped by type of energy source,
assign individual colors (linetypes) to the factors taken from column 4
Billion Tons of Oil Equivalent
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Coal
Oil
Gas
Nuclear
Hydroelectric
Renewable

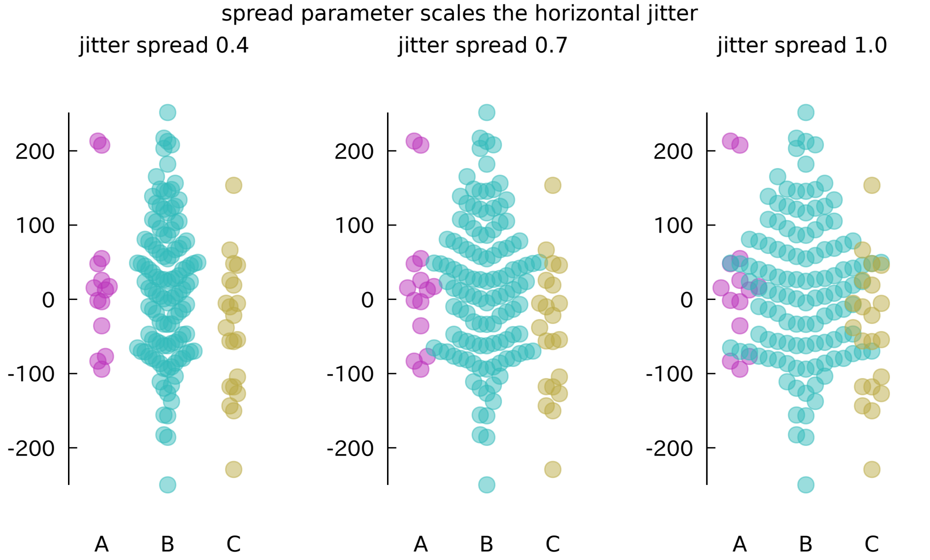


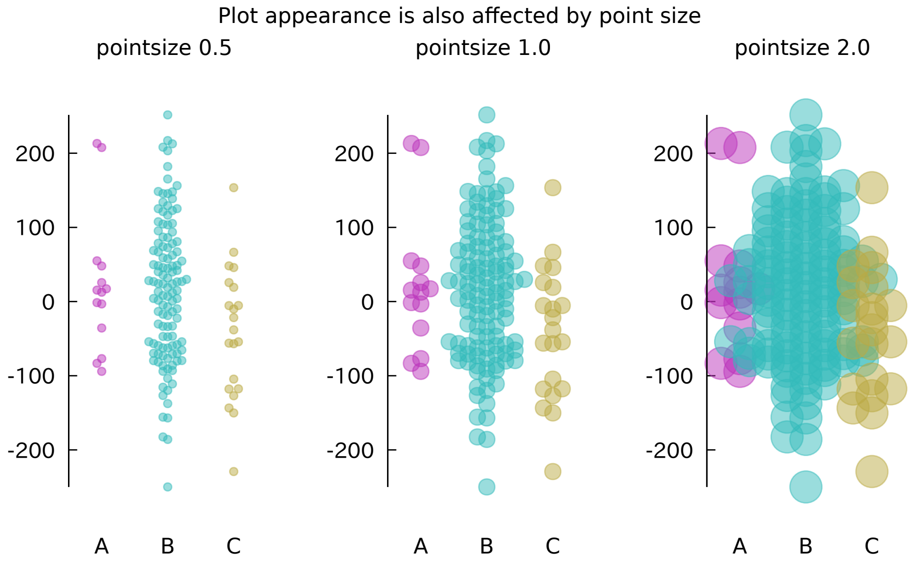
Distribution of energy usage of the continents, sorted by name of energy source
Billion Tons of Oil Equivalent
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Coal
Gas
Hydroelectric
Nuclear
Oil
Renewable


Distribution of energy usage explicitly ordered by name of energy source
Billion Tons of Oil Equivalent
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Nuclear
Coal
Gas
Oil
Hydroelectric
Renewable

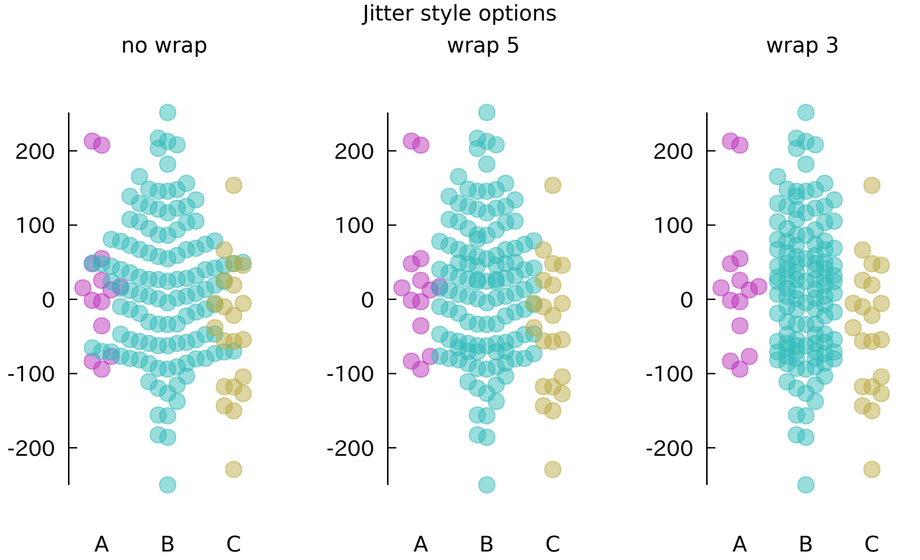

no jitter
jitter
200
100
0
-100
-200
A
B
C


vertical overlap criterion
jitter overlap 0.5
jitter overlap 1.0
jitter overlap 1.5
200
100
0
-100
-200
A
B
C

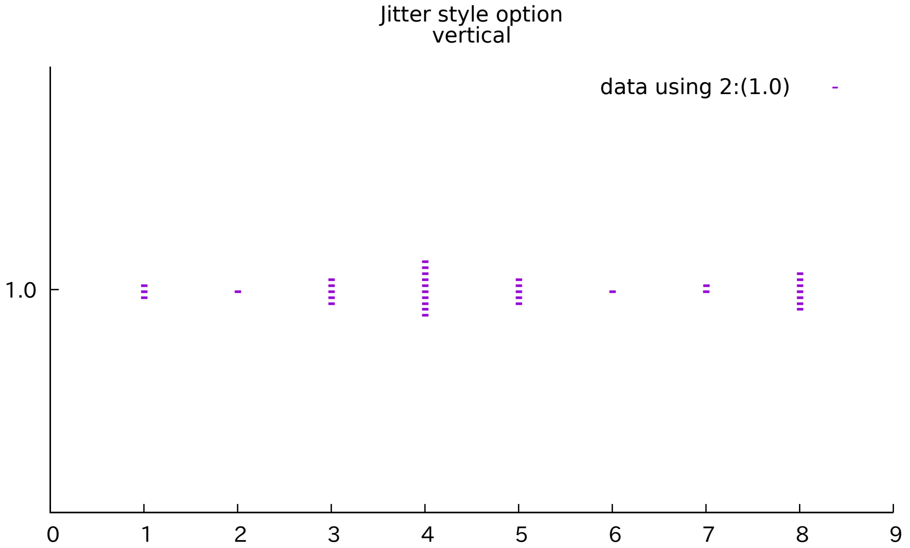

spread parameter scales the horizontal jitter
jitter spread 0.4
jitter spread 0.7
jitter spread 1.0
200
100
0
-100
-200
A
B
C

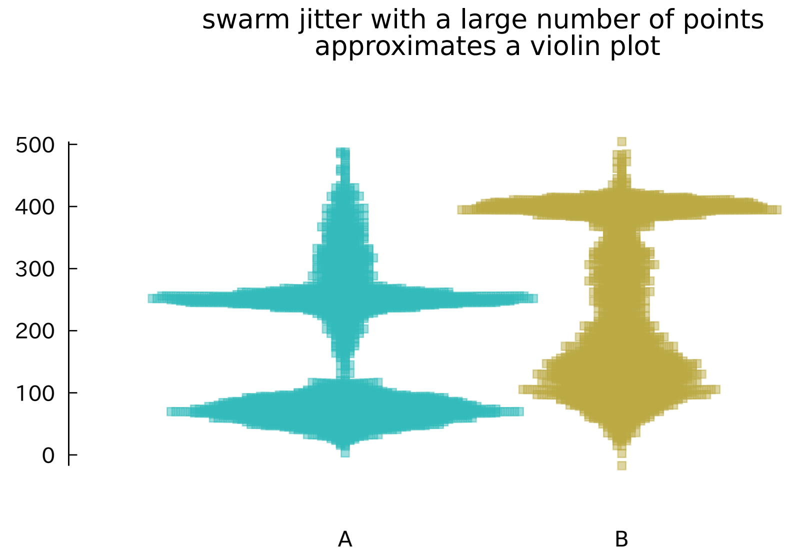

Plot appearance is also affected by point size
pointsize 0.5
pointsize 1.0
pointsize 2.0
200
100
0
-100
-200
A
B
C

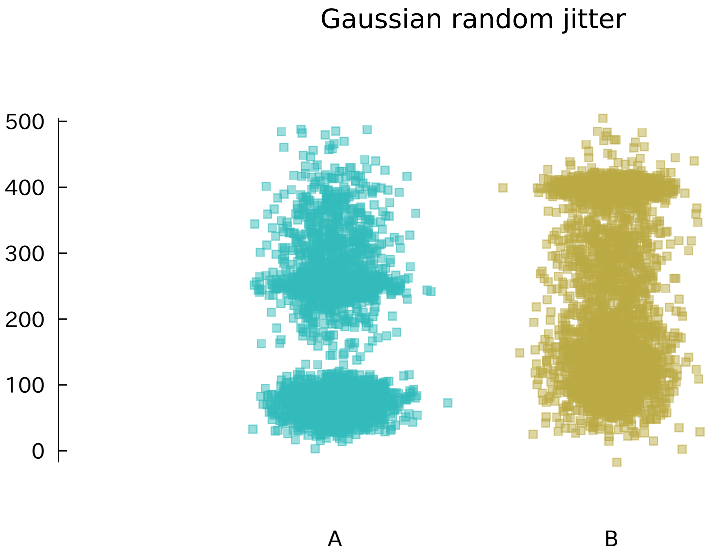

Jitter style options
swarm (default)
square
200
100
0
-100
-200
A
B
C

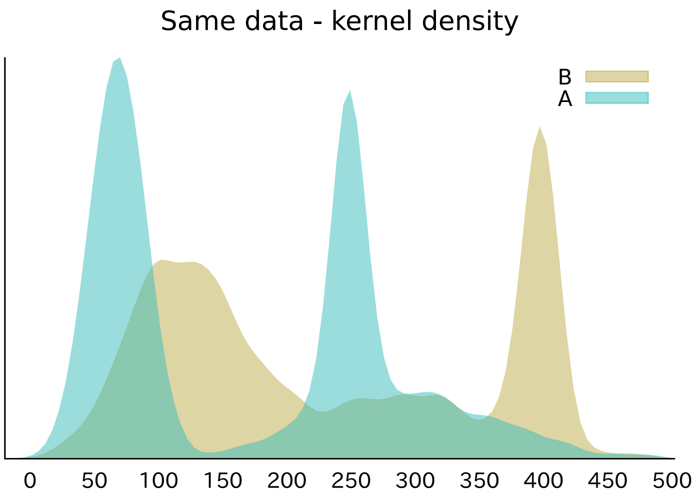


Jitter style options
no wrap
wrap 5
wrap 3
200
100
0
-100
-200
A
B
C


Jitter style option
vertical
data using 2:(1.0)
1.0
0
1
2
3
4
5
6
7
8
9

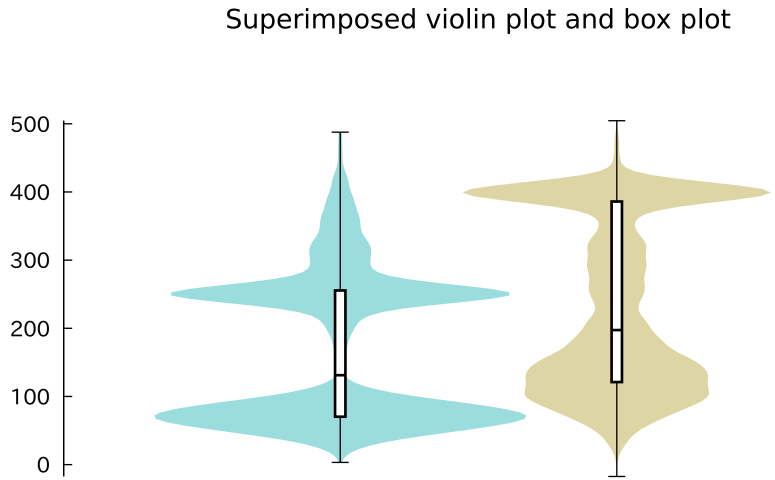

swarm jitter with a large number of points
approximates a violin plot
500
400
300
200
100
0
A
B


Gaussian random jitter
500
400
300
200
100
0
A
B

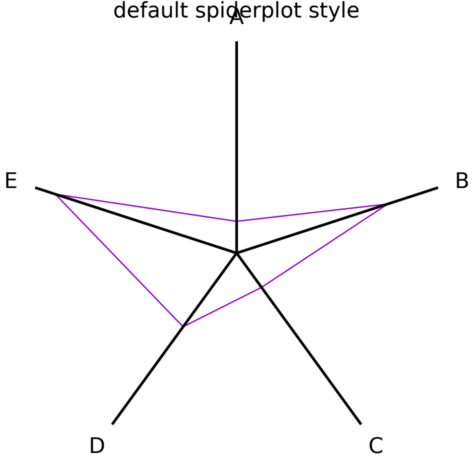


Same data - kernel density
B
A
0
50
100
150
200
250
300
350
400
450
500

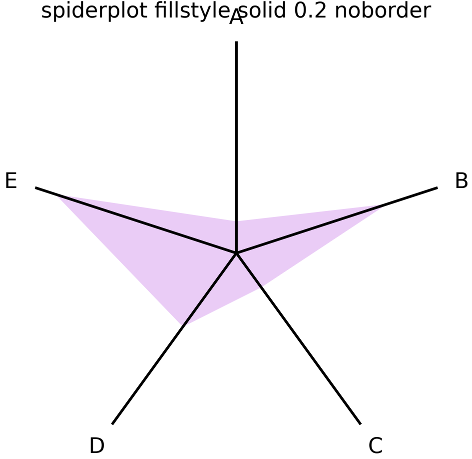
kdensity mirrored sideways to give a violin plot
500
400
300
200
100
0


Superimposed violin plot and box plot
500
400
300
200
100
0

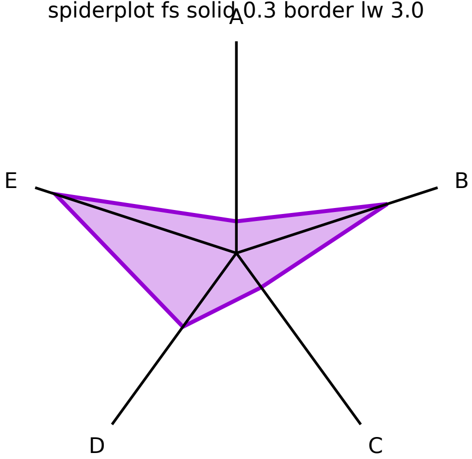
default spiderplot style

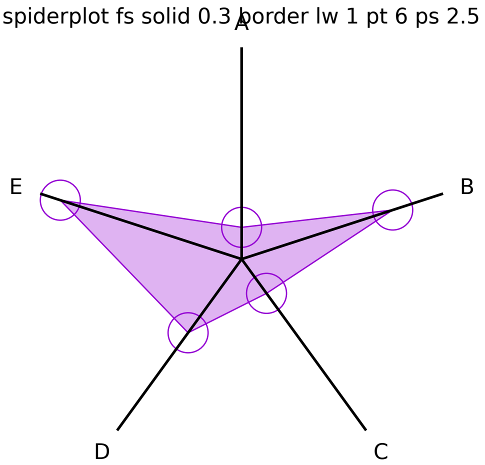


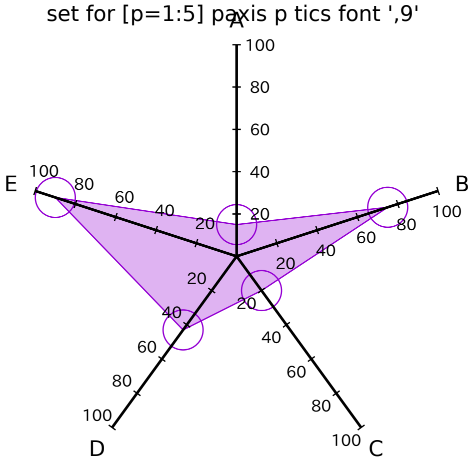
spiderplot fillstyle solid 0.2 noborder

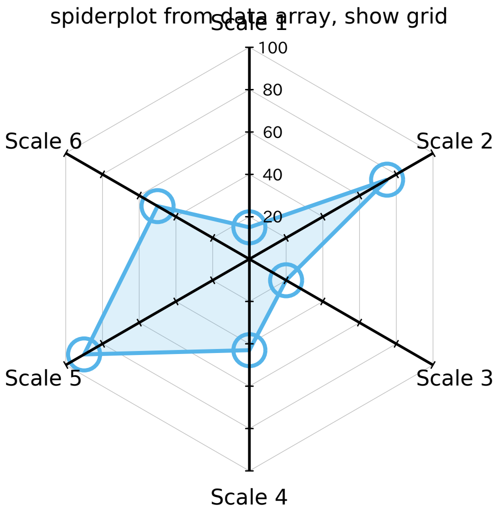


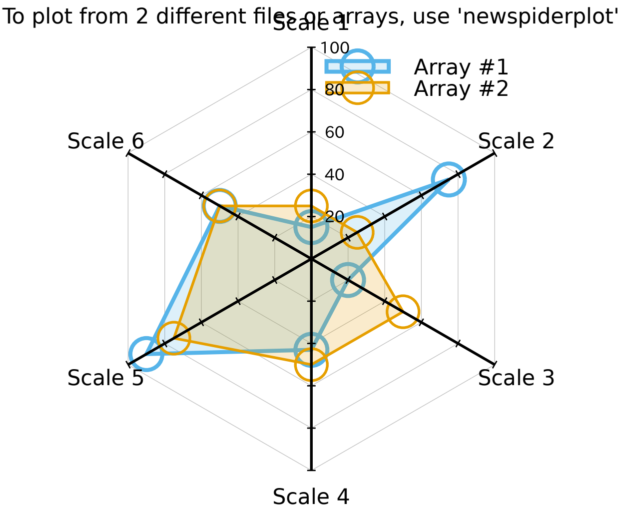
spiderplot fs solid 0.3 border lw 3.0



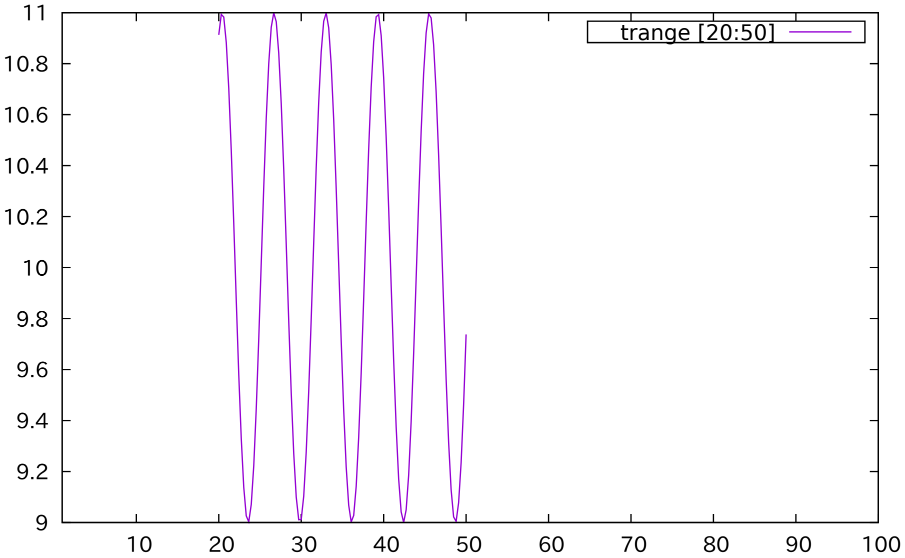

spiderplot fs solid 0.3 border lw 1 pt 6 ps 2.5
A
B
C
D
E

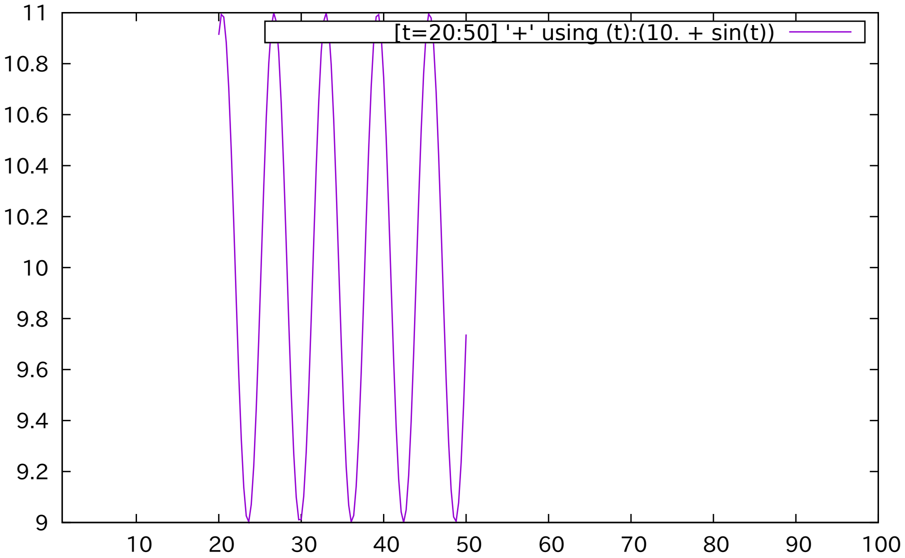

set for [p=1:5] paxis p tics font ',9'
A
B
C
D
E
20
40
60
80
100


spiderplot from data array, show grid
Scale 1
Scale 2
Scale 3
Scale 4
Scale 5
Scale 6
100
80
60
40
20

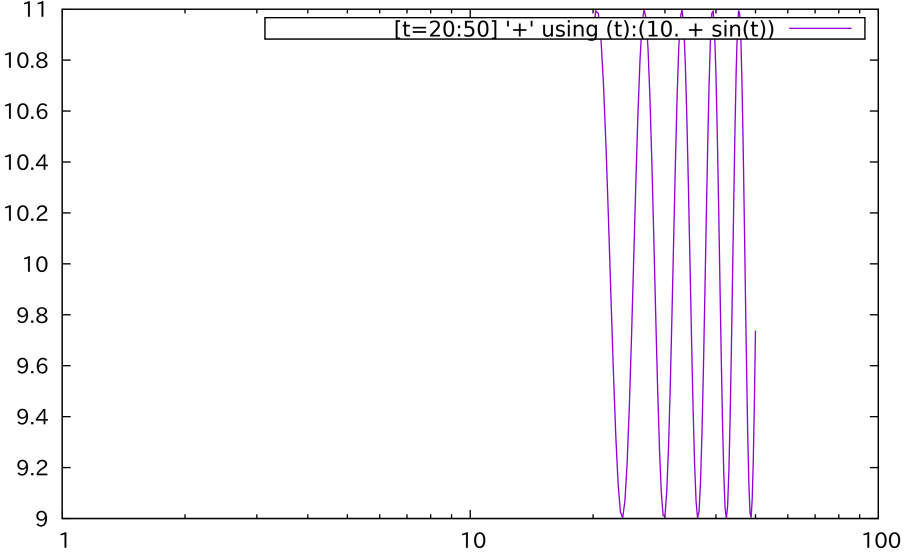


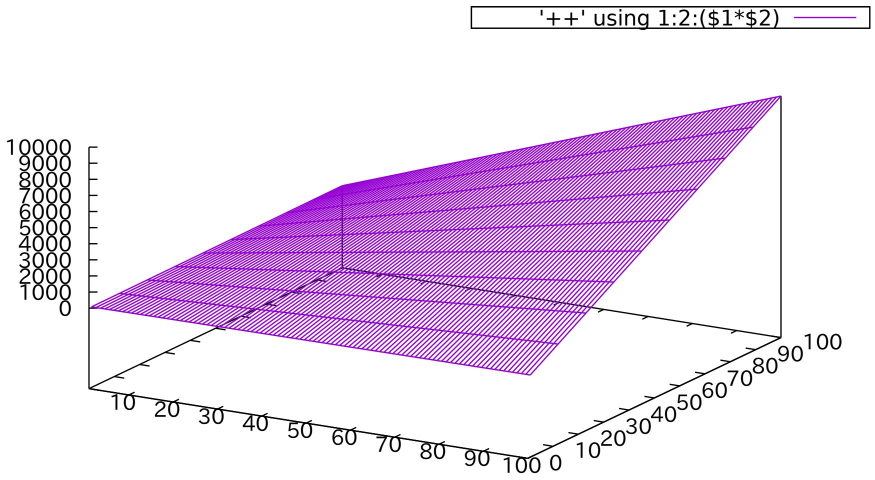
To plot from 2 different files or arrays, use 'newspiderplot'
Scale 1
Scale 2
Scale 3
Scale 4
Scale 5
Scale 6
100
80
60
40
20
Array #1
Array #2

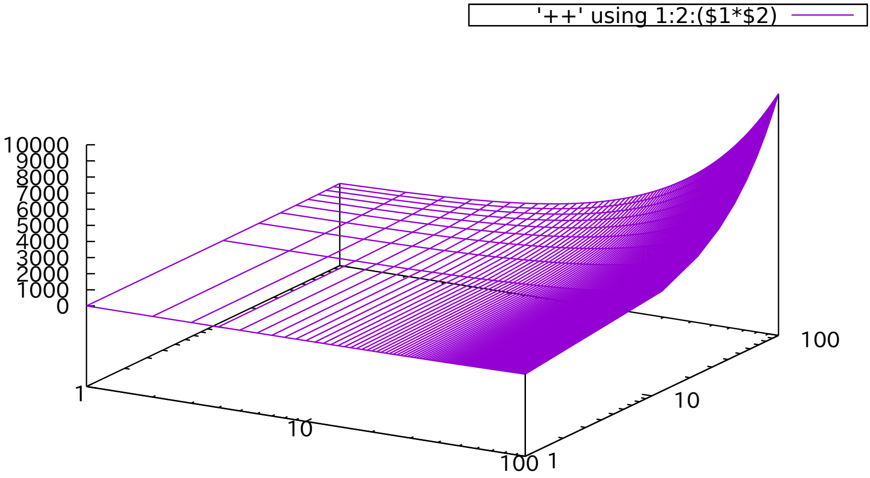

trange [20:50]
11
10.8
10.6
10.4
10.2
10
9.8
9.6
9.4
9.2
9
10
20
30
40
50
60
70
80
90
100



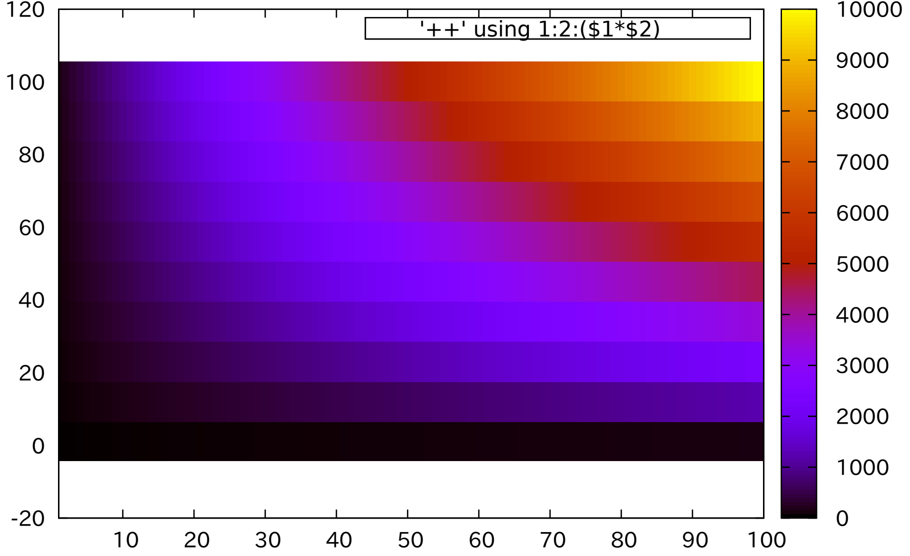
[t=20:50] '+' using (t):(10. + sin(t))
11
10.8
10.6
10.4
10.2
10
9.8
9.6
9.4
9.2
9
10
20
30
40
50
60
70
80
90
100

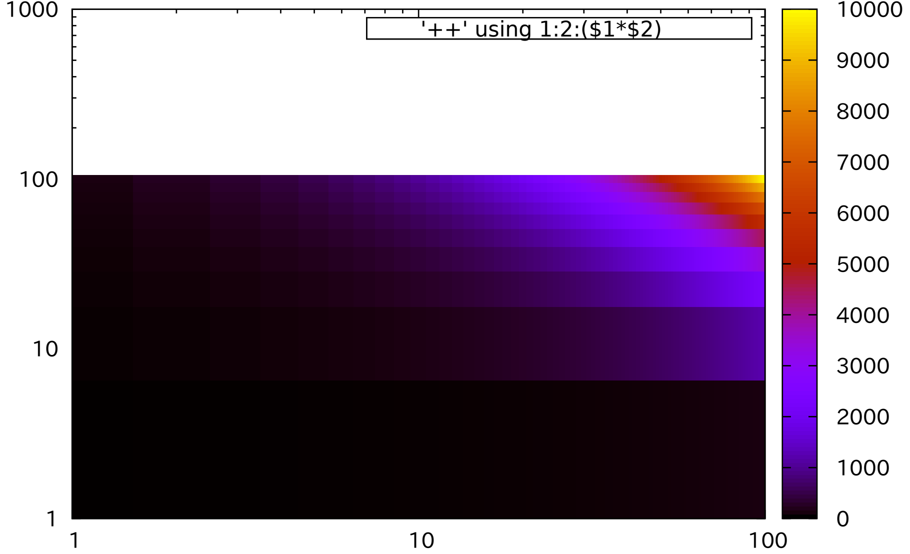

[t=20:50] '+' using (t):(10. + sin(t))
11
10.8
10.6
10.4
10.2
10
9.8
9.6
9.4
9.2
9
1
10
100


'++' using 1:2:($1*$2)
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
10
20
30
40
50
60
70
80
90
100
0
10
20
30
40
50
60
70
80
90
100

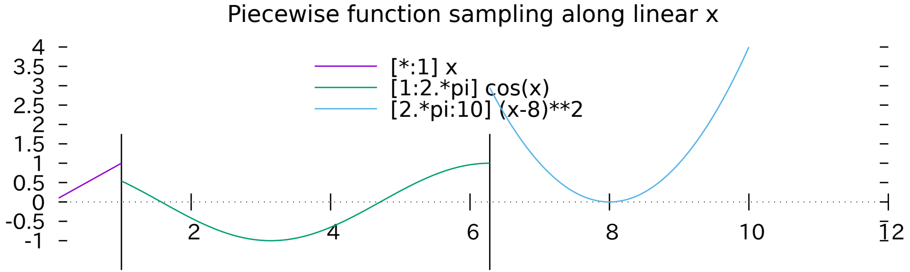


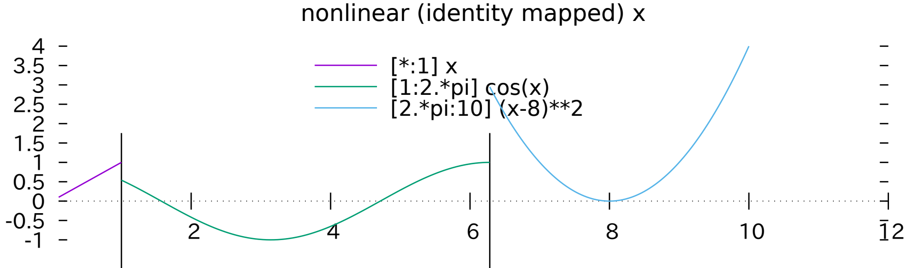
'++' using 1:2:($1*$2)
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
10
100
1
10
100

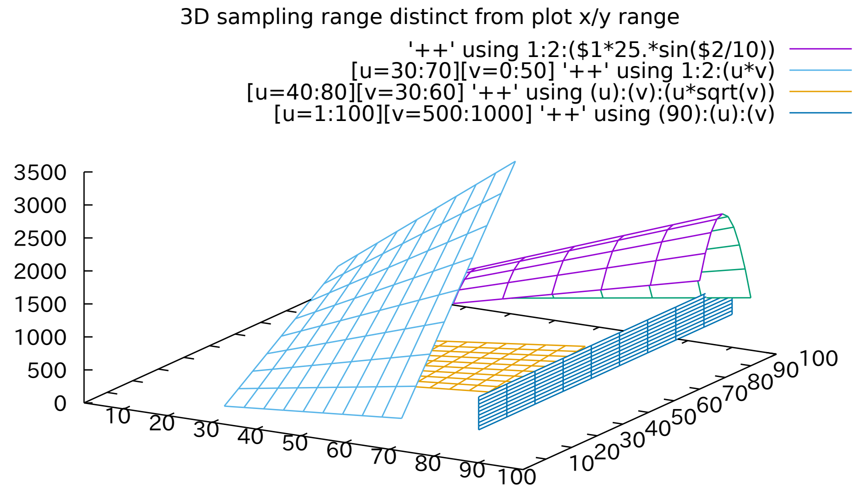

'++' using 1:2:($1*$2)
120
100
80
60
40
20
0
-20
10
20
30
40
50
60
70
80
90
100
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

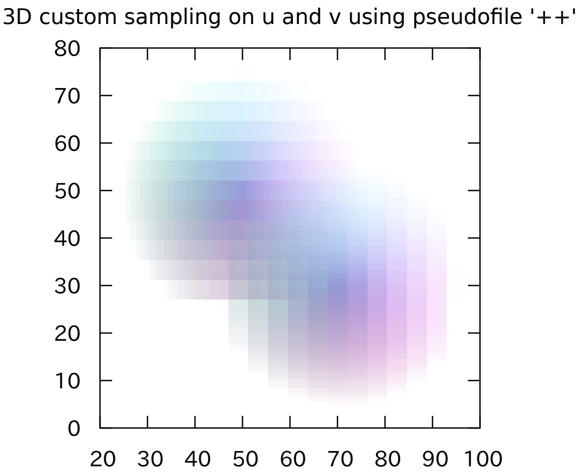


'++' using 1:2:($1*$2)
1000
100
10
1
1
10
100
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

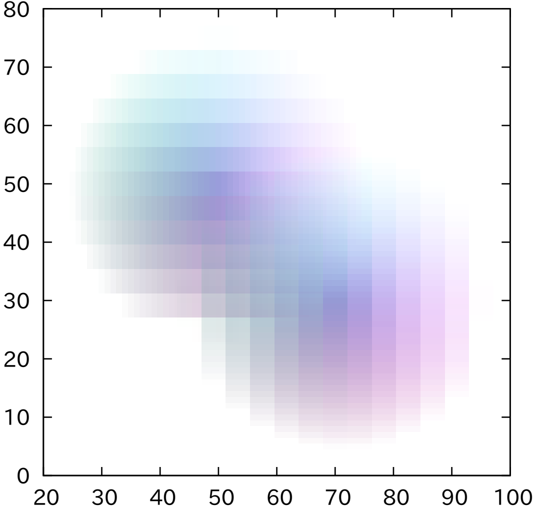

Piecewise function sampling along linear x
[*:1] x
[1:2.*pi] cos(x)
[2.*pi:10] (x-8)**2
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
2
4
6
8
10
12

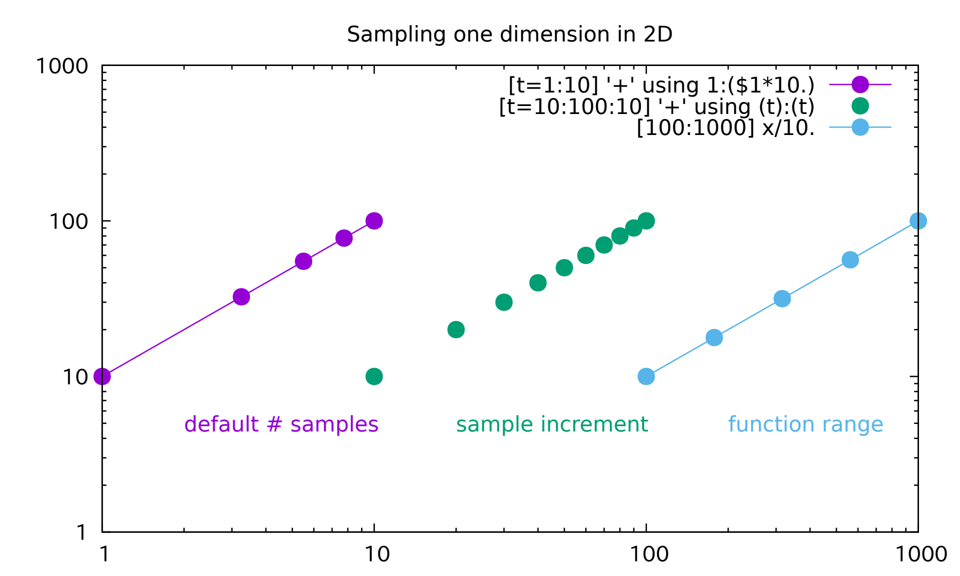

nonlinear (identity mapped) x
[*:1] x
[1:2.*pi] cos(x)
[2.*pi:10] (x-8)**2
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
2
4
6
8
10
12

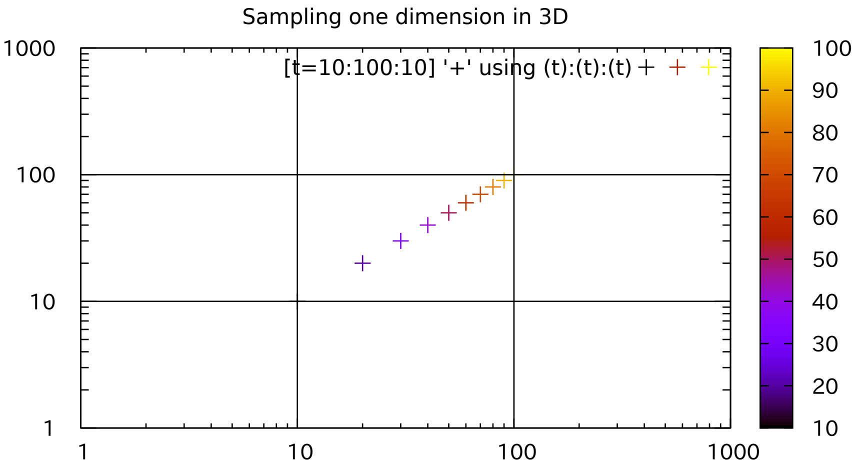

3D sampling range distinct from plot x/y range
'++' using 1:2:($1*25.*sin($2/10))
[u=30:70][v=0:50] '++' using 1:2:(u*v)
[u=40:80][v=30:60] '++' using (u):(v):(u*sqrt(v))
[u=1:100][v=500:1000] '++' using (90):(u):(v)
3500
3000
2500
2000
1500
1000
500
0
10
20
30
40
50
60
70
80
90
100
10
20
30
40
50
60
70
80
90
100

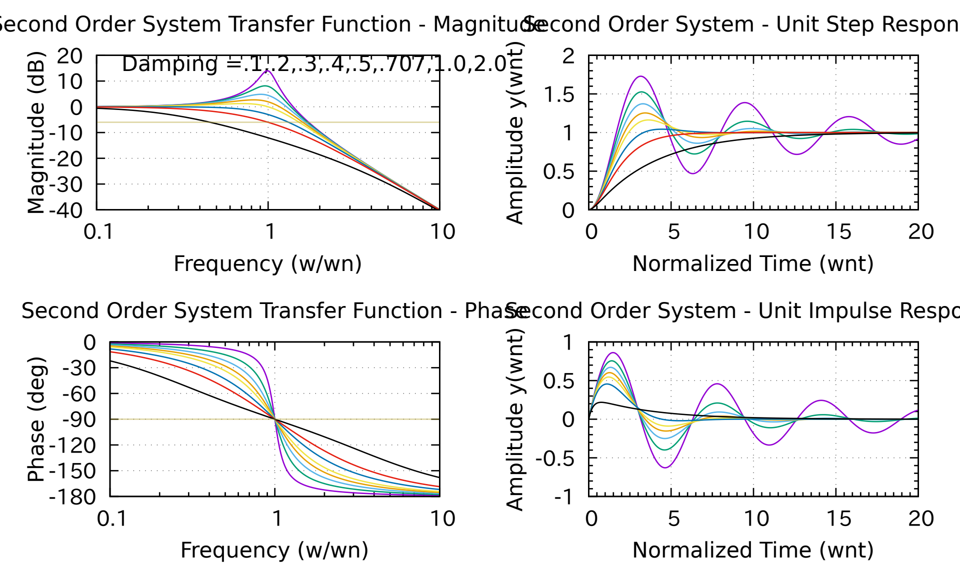

3D custom sampling on u and v using pseudofile '++'
80
70
60
50
40
30
20
10
0
20
30
40
50
60
70
80
90
100

2D custom sampling on u and v using pseudofile '++'

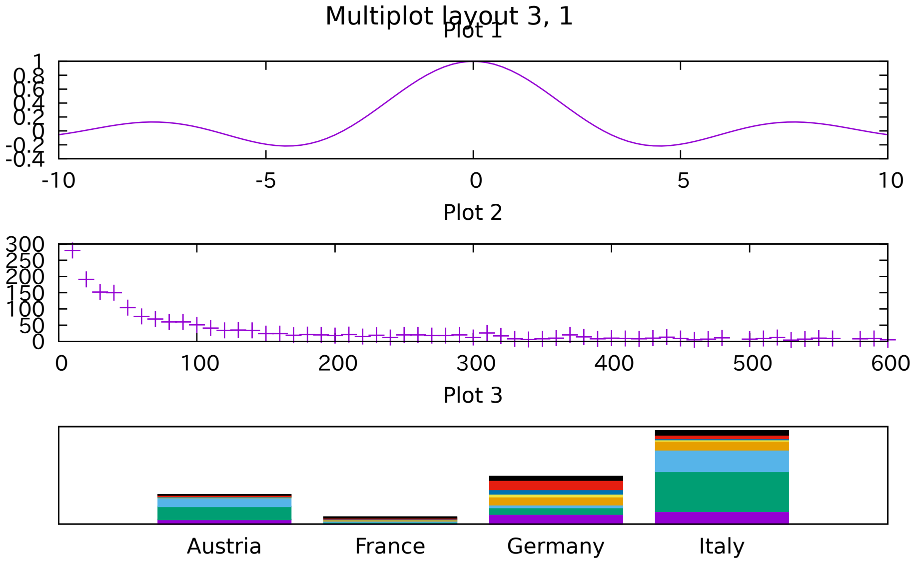


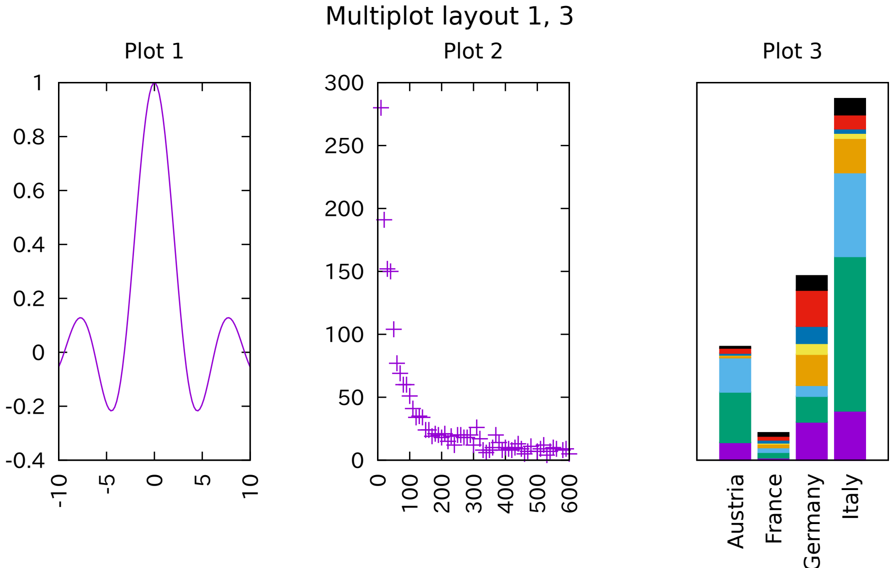

Sampling one dimension in 2D
1000
100
10
1
1
10
100
1000
[t=1:10] '+' using 1:($1*10.)
[t=10:100:10] '+' using (t):(t)
[100:1000] x/10.
default # samples
sample increment
function range

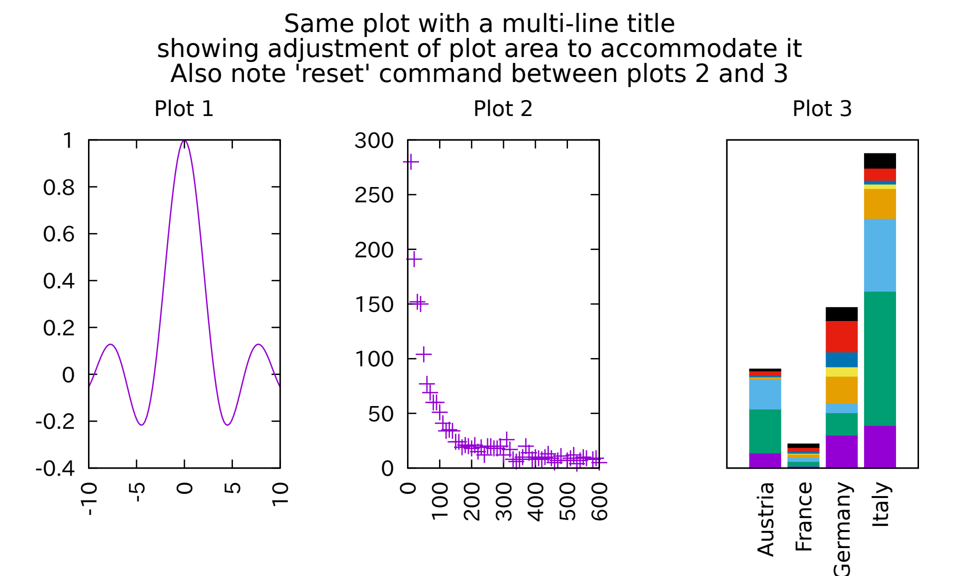

Sampling one dimension in 3D
[t=10:100:10] '+' using (t):(t):(t)
1000
100
10
1
1
10
100
1000
100
90
80
70
60
50
40
30
20
10

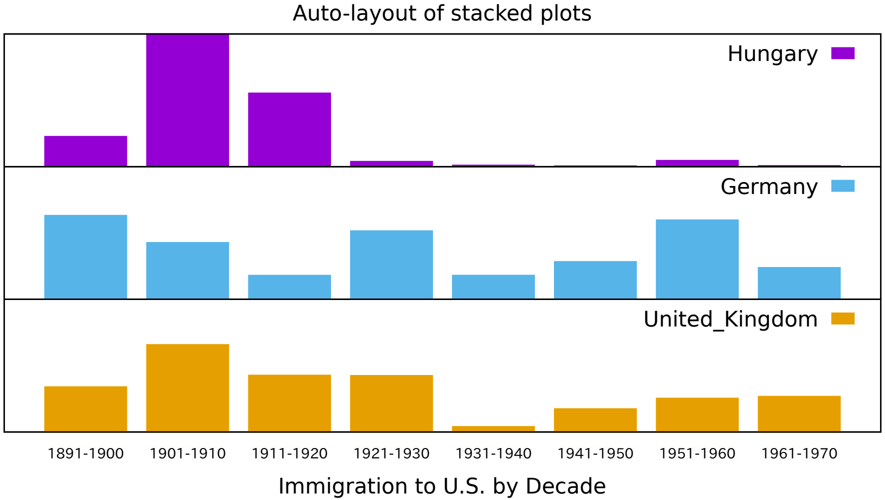


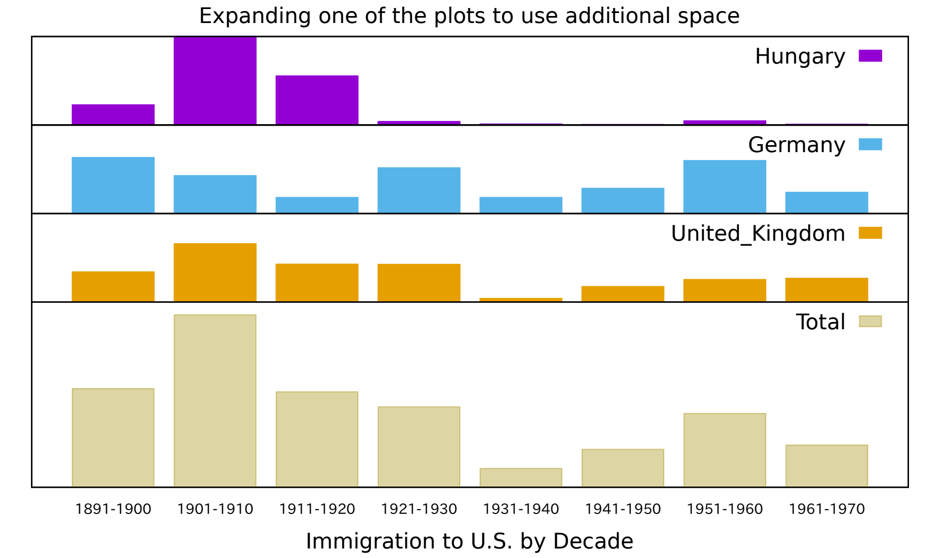
Second Order System Transfer Function - Magnitude
Damping =.1,.2,.3,.4,.5,.707,1.0,2.0
Magnitude (dB)
20
10
0
-10
-20
-30
-40
0.1
1
10
Frequency (w/wn)
Second Order System - Unit Step Response
Amplitude y(wnt)
2
1.5
1
0.5
0
5
10
15
20
Normalized Time (wnt)
Second Order System Transfer Function - Phase
Phase (deg)
0
-30
-60
-90
-120
-150
-180
0.1
1
10
Frequency (w/wn)
Second Order System - Unit Impulse Response
Amplitude y(wnt)
1
0.5
0
-0.5
-1
0
5
10
15
20
Normalized Time (wnt)

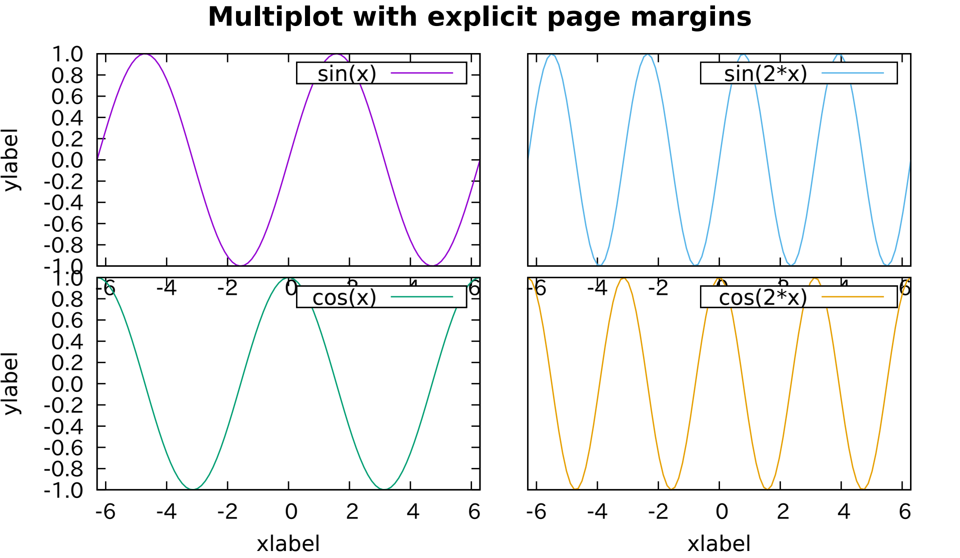

Multiplot layout 3, 1
Plot 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
Plot 2
300
250
200
150
100
50
0
0
100
200
300
400
500
600
Plot 3
Austria
France
Germany
Italy


Multiplot layout 1, 3
Plot 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
Plot 2
300
250
200
150
100
50
0
0
100
200
300
400
500
600
Plot 3
Austria
France
Germany
Italy


Same plot with a multi-line title
showing adjustment of plot area to accommodate it
Also note 'reset' command between plots 2 and 3
Plot 1
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
Plot 2
300
250
200
150
100
50
0
0
100
200
300
400
500
600
Plot 3
Austria
France
Germany
Italy

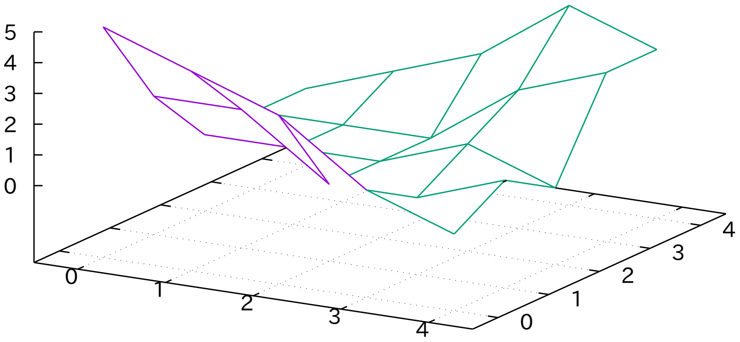

Auto-layout of stacked plots
Hungary
Germany
United_Kingdom
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration to U.S. by Decade

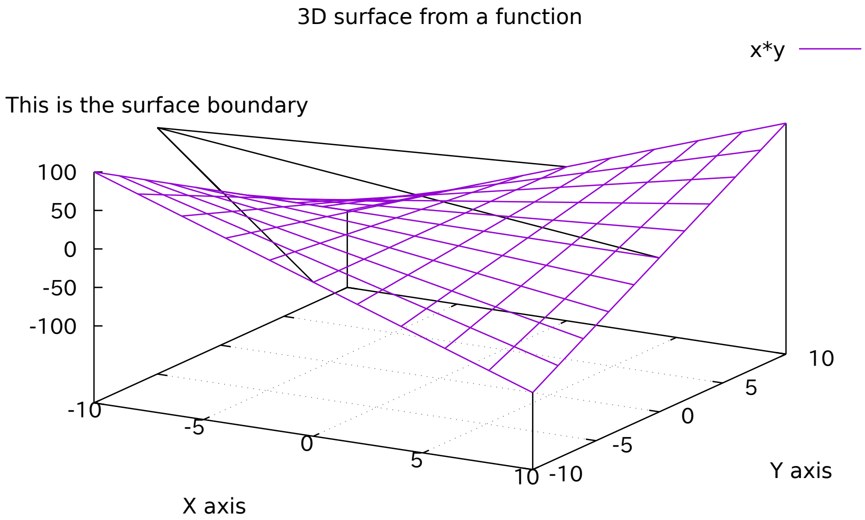

Expanding one of the plots to use additional space
Hungary
Germany
United_Kingdom
Total
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration to U.S. by Decade



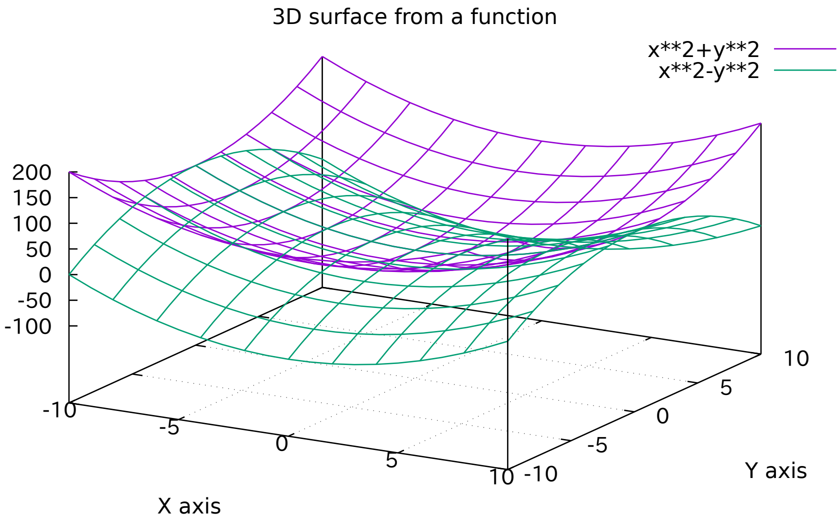
Multiplot with explicit page margins
sin(x)
sin(2*x)
cos(x)
cos(2*x)
ylabel
ylabel
xlabel
xlabel
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-6
-4
-2
0
2
4
6

3D surface from a grid (matrix) of Z values

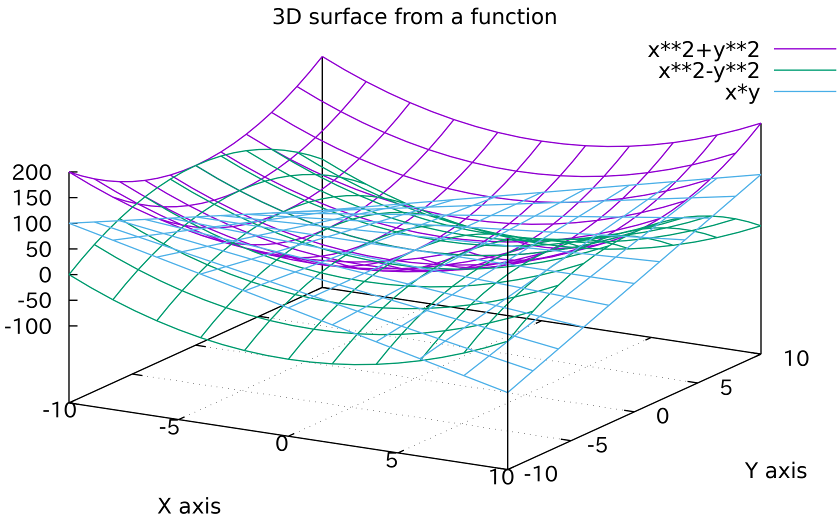


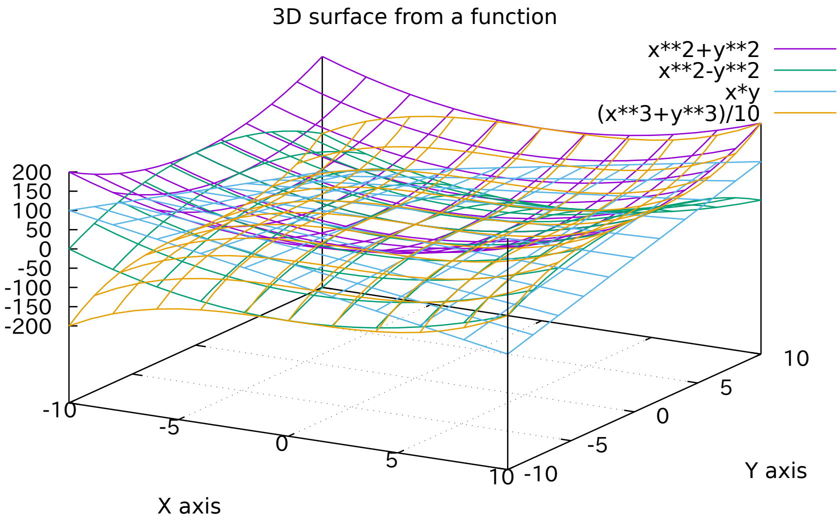


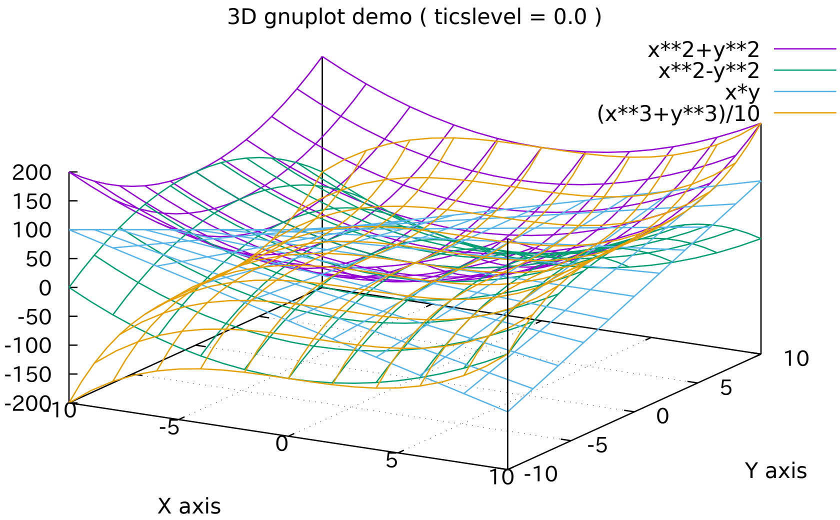
3D surface from a function
x*y
This is the surface boundary
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10
X axis
Y axis

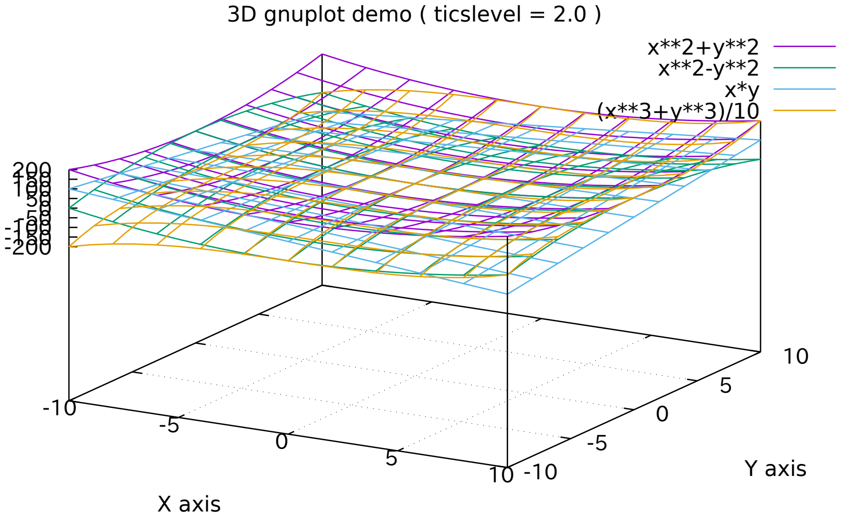


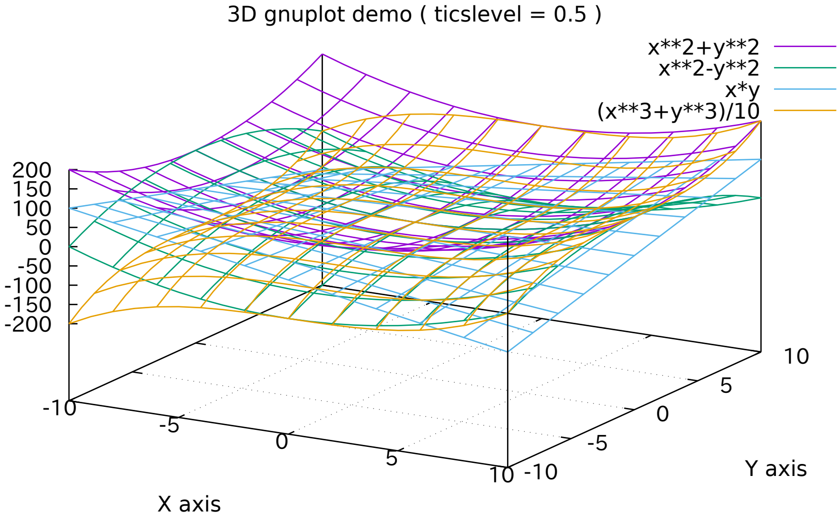
3D surface from a function
x**2+y**2
x**2-y**2
200
150
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10
X axis
Y axis

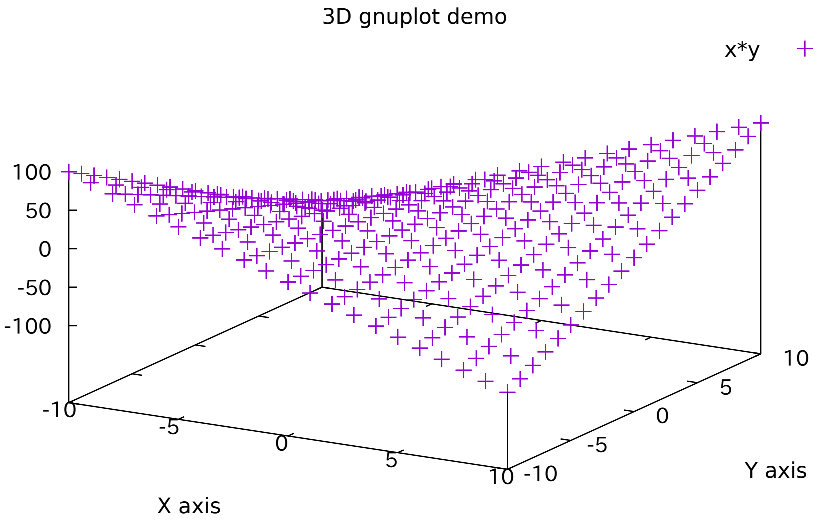

3D surface from a function
x**2+y**2
x**2-y**2
x*y
200
150
100
50
0
-50
-100
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis

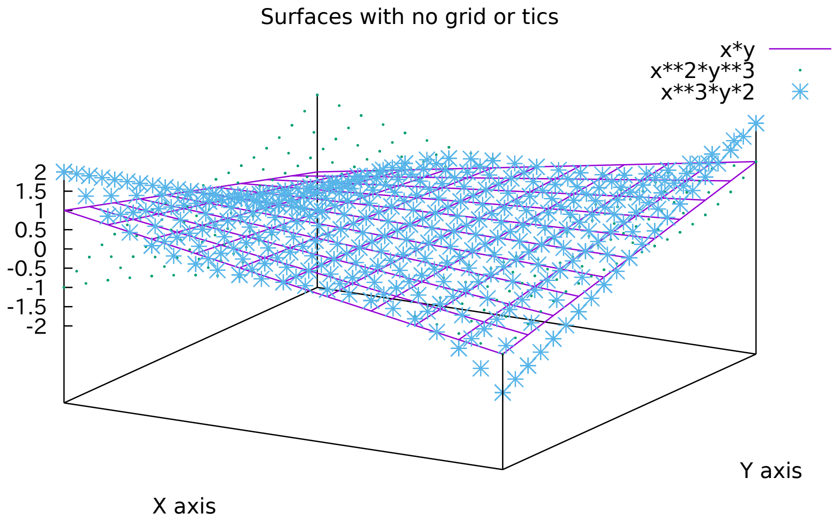

3D surface from a function
x**2+y**2
x**2-y**2
x*y
(x**3+y**3)/10
200
150
100
50
0
-50
-100
-150
-200
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis


3D gnuplot demo (ticslevel = 0.0)
x**2+y**2
x**2-y**2
x*y
(x**3+y**3)/10
200
150
100
50
0
-50
-100
-150
-200
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis

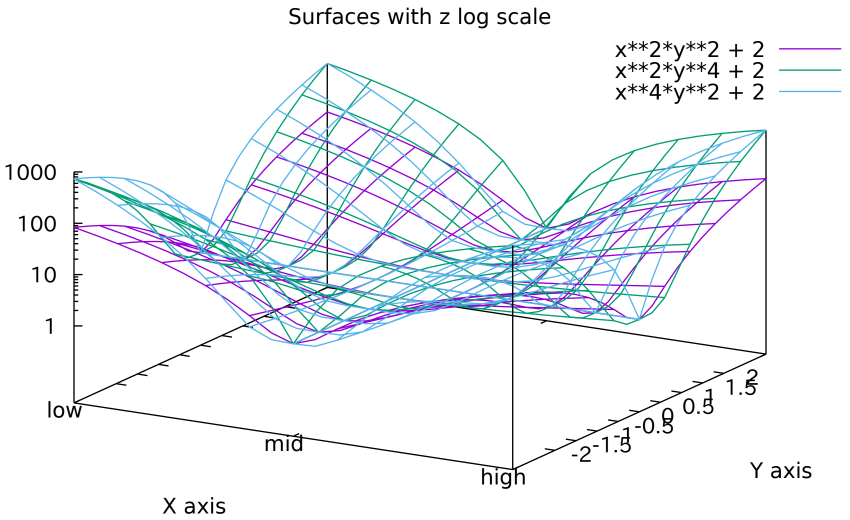

3D gnuplot demo (ticslevel = 2.0)
x**2+y**2
x**2-y**2
x*y
(x**3+y**3)/10
200
-200
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis


3D gnuplot demo (ticslevel = 0.5)
x**2+y**2
x**2-y**2
x*y
(x**3+y**3)/10
200
150
100
50
0
-50
-100
-150
-200
-10
-5
0
5
10
-10
-5
0
5
10
X axis
Y axis

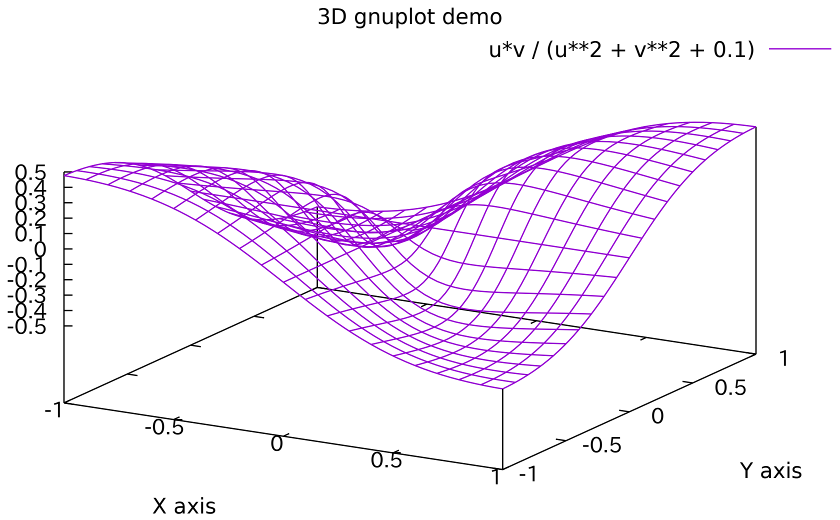

3D gnuplot demo
x*y
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10
X axis
Y axis

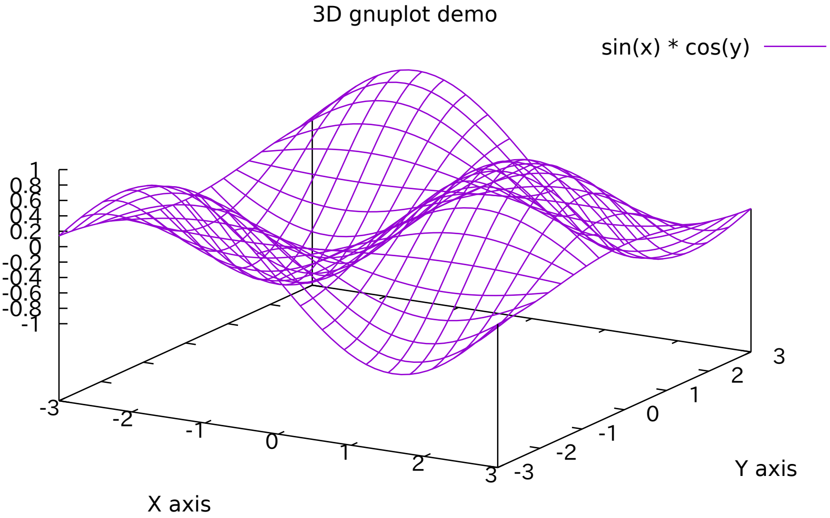


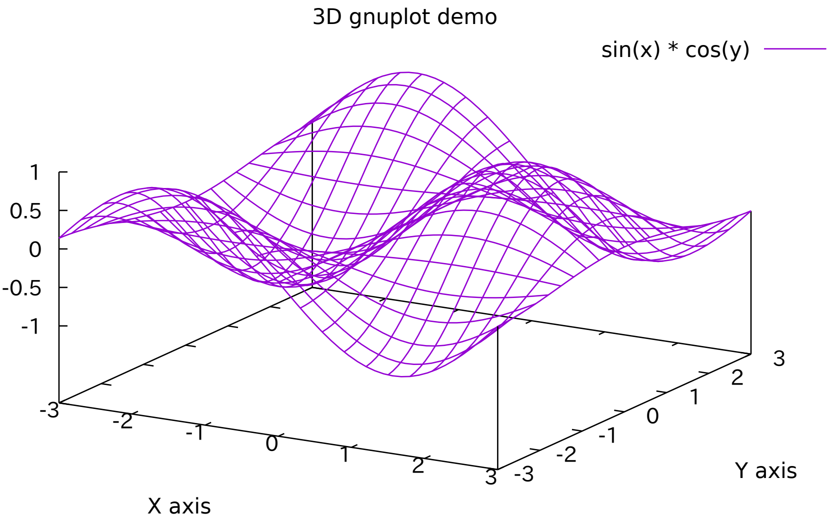
Surfaces with no grid or tics
x*y
x**2*y**3
x**3*y*2
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
X axis
Y axis

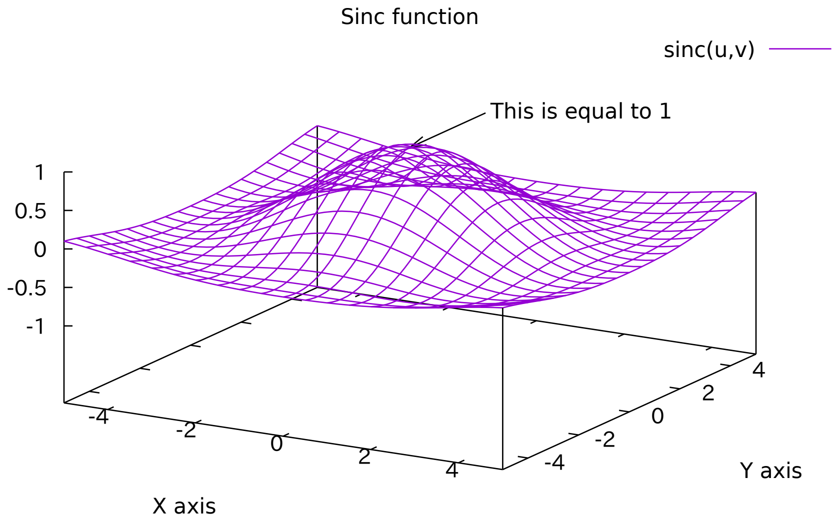


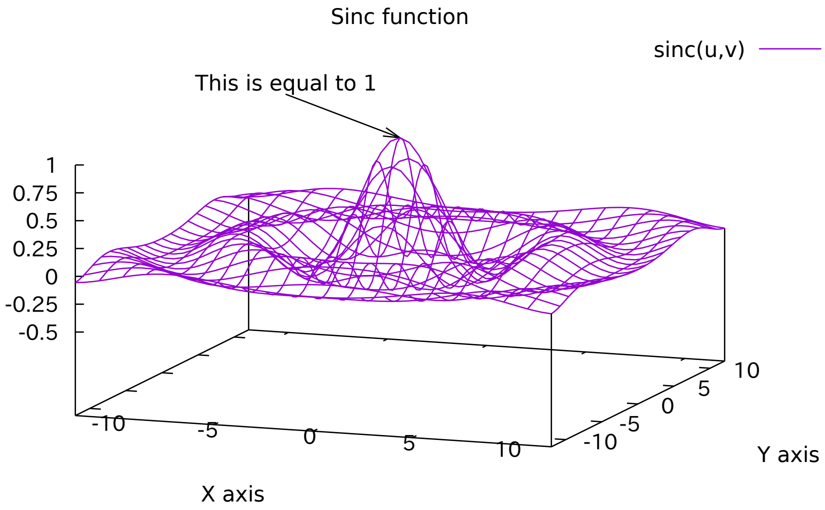
Surfaces with z log scale
x**2*y**2 + 2
x**2*y**4 + 2
x**4*y**2 + 2
1000
100
10
1
low
mid
high
X axis
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Y axis

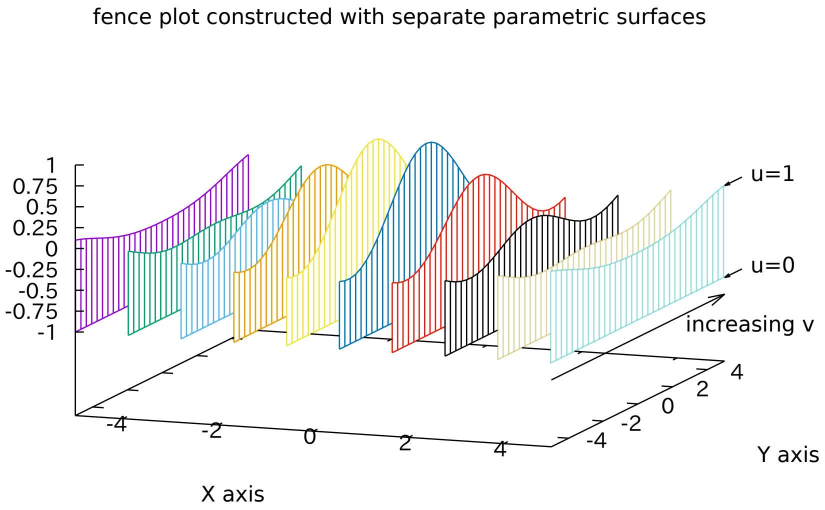

3D gnuplot demo
u*v / (u**2 + v**2 + 0.1)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1
X axis
Y axis

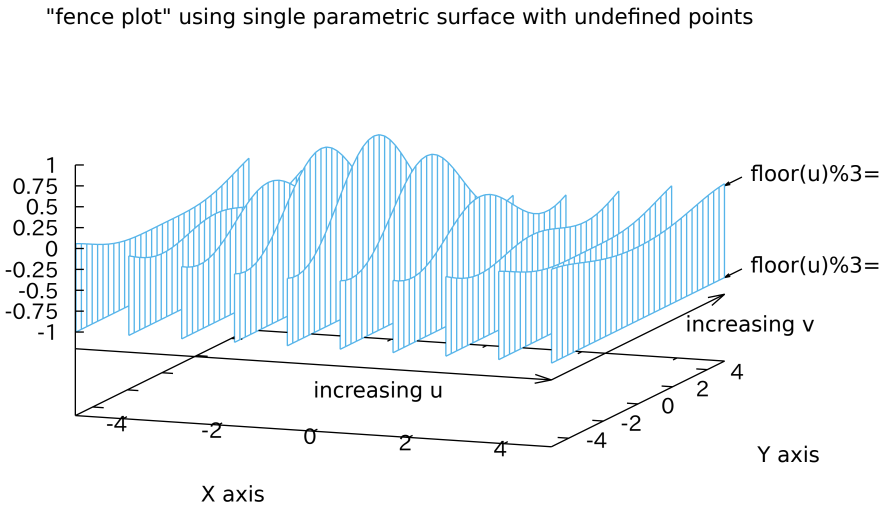

3D gnuplot demo
sin(x) * cos(y)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-3
-2
-1
0
1
2
3
X axis
-3
-2
-1
0
1
2
3
Y axis


3D gnuplot demo
sin(x) * cos(y)
1
0.5
0
-0.5
-1
-3
-2
-1
0
1
2
3
X axis
-3
-2
-1
0
1
2
3
Y axis

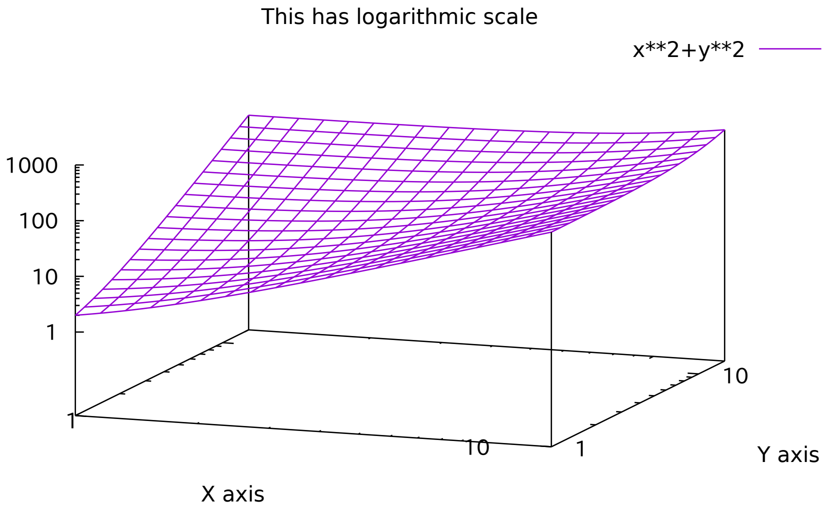

Sinc function
sinc(u,v)
This is equal to 1
1
0.5
0
-0.5
-1
-4
-2
0
2
4
X axis
-4
-2
0
2
4
Y axis


Sinc function
sinc(u,v)
This is equal to 1
1
0.75
0.5
0.25
0
-0.25
-0.5
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis

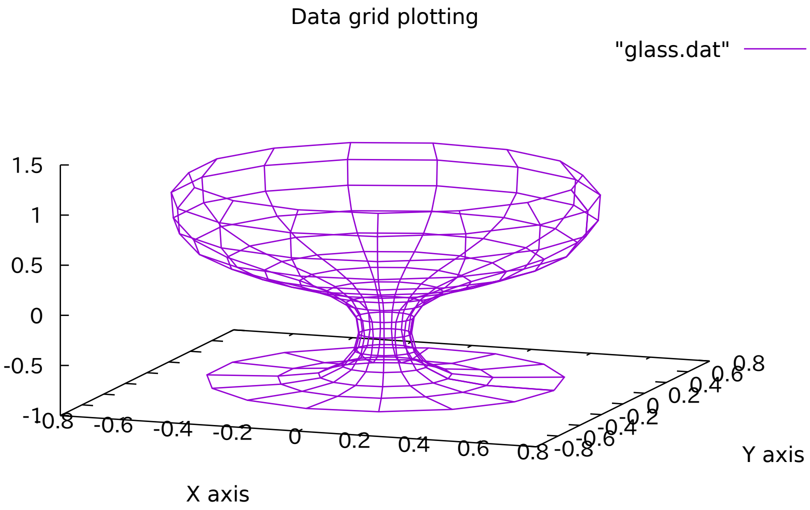

fence plot constructed with separate parametric surfaces
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
u=1
u=0
increasing v
-4
-2
0
2
4
X axis
-4
-2
0
2
4
Y axis

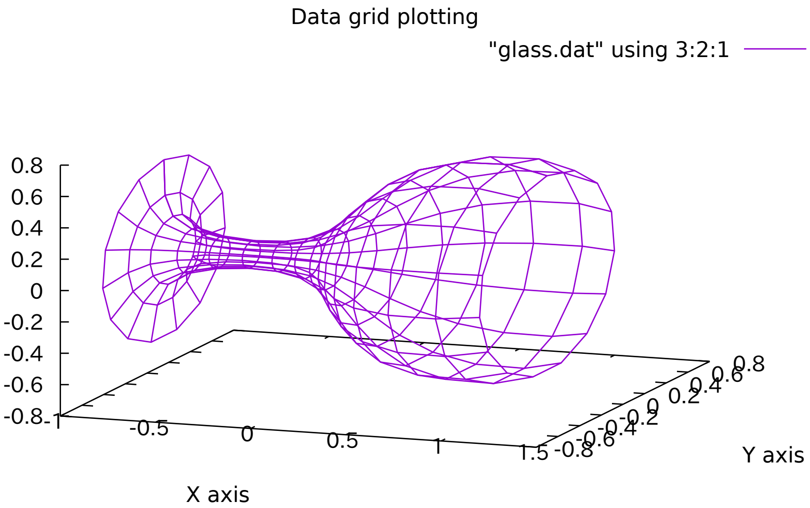


"fence plot" using single parametric surface with undefined points
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
floor(u)%3=
floor(u)%3=
increasing v
increasing u
-4
-2
0
2
4
4
2
0
-2
-4
Y axis
X axis

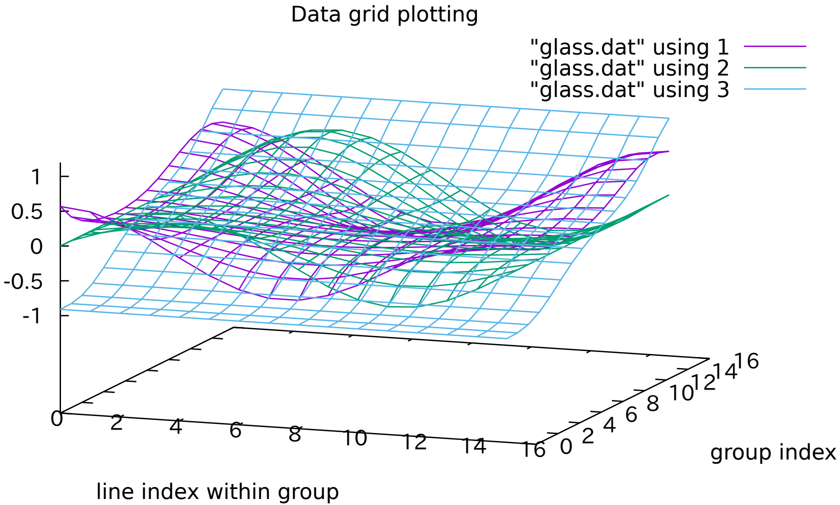


This has logarithmic scale
x**2+y**2
1000
100
10
1
1
10
1
10
X axis
Y axis

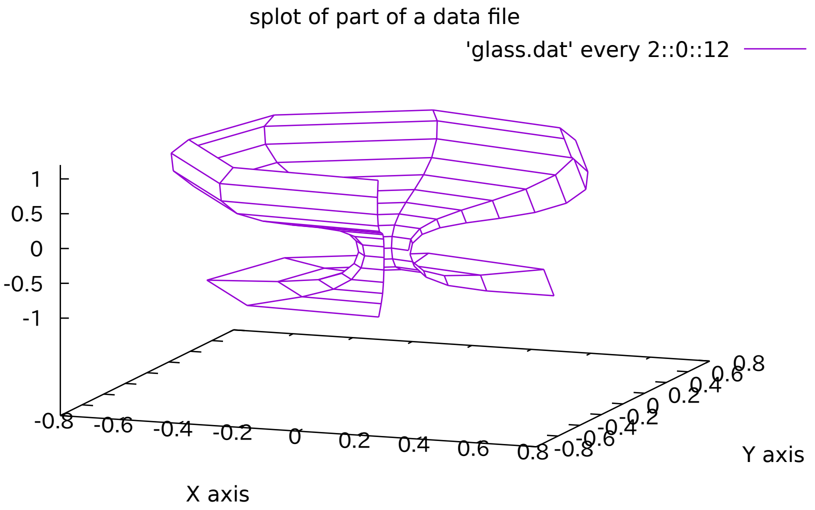

Data grid plotting
"glass.dat"
1.5
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
X axis
Y axis

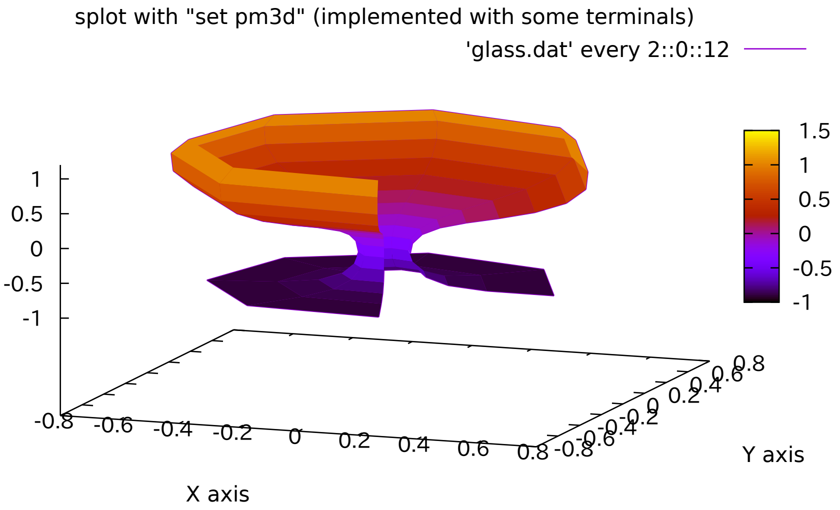


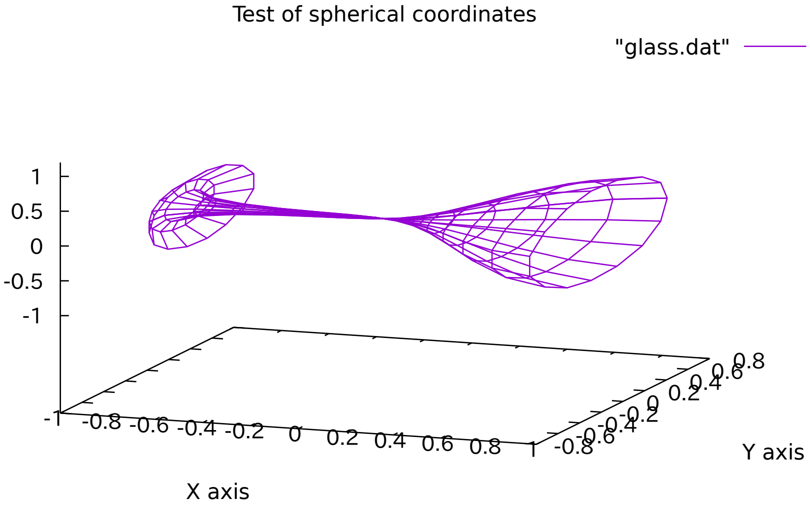
Data grid plotting
"glass.dat" using 3:2:1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
0
0.5
1
1.5
X axis
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
Y axis



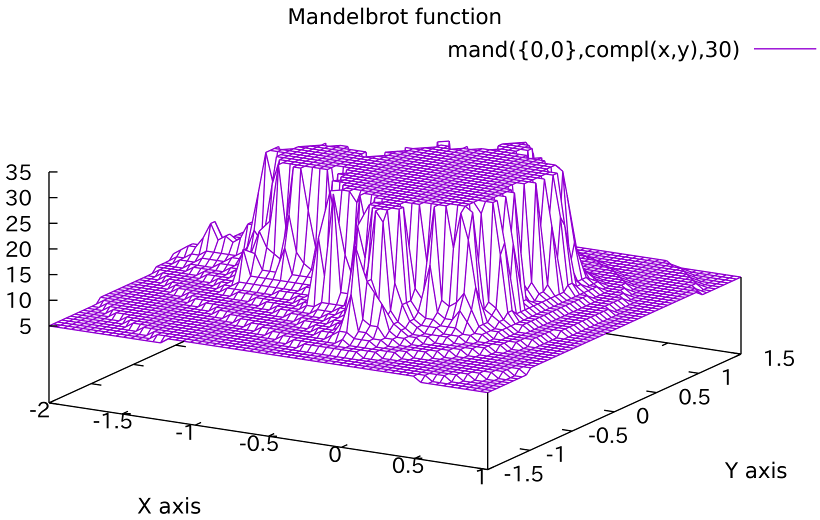
Data grid plotting
"glass.dat" using 1
"glass.dat" using 2
"glass.dat" using 3
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14
16
0
2
4
6
8
10
12
14
16
line index within group
group index


splot of part of a data file
'glass.dat' every 2::0::12
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
X axis
Y axis

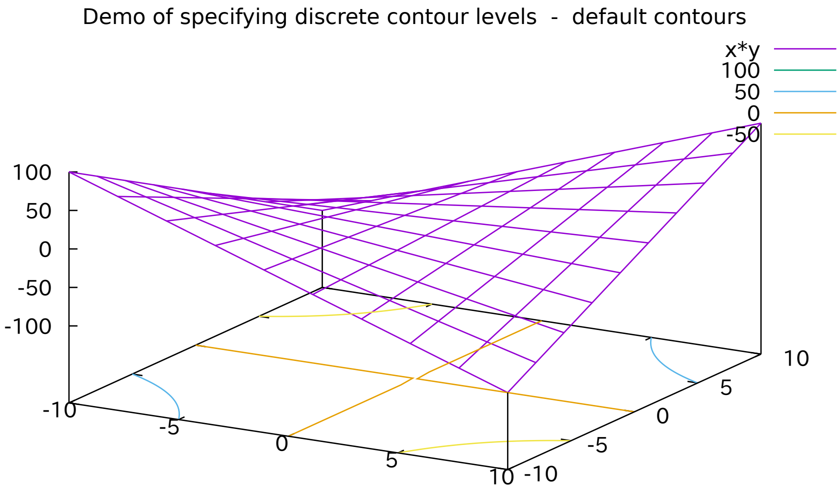

splot with "set pm3d" (implemented with some terminals)
'glass.dat' every 2::0::12
1
0.5
0
-0.5
-1
1.5
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
X axis
Y axis

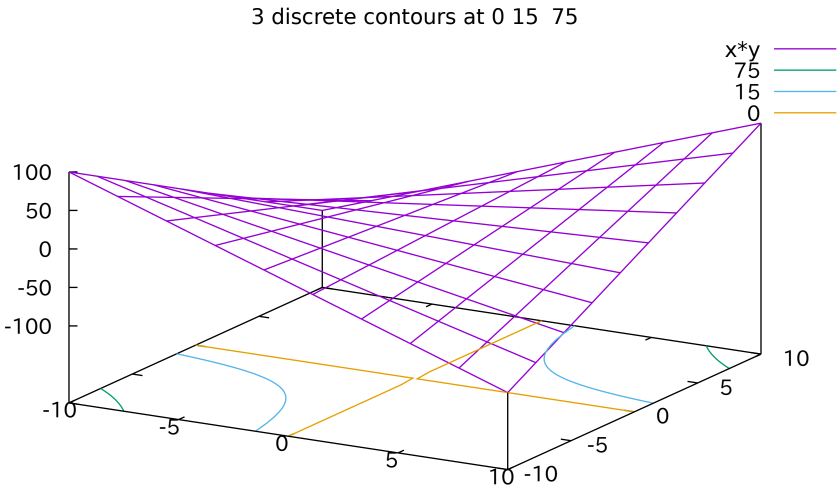


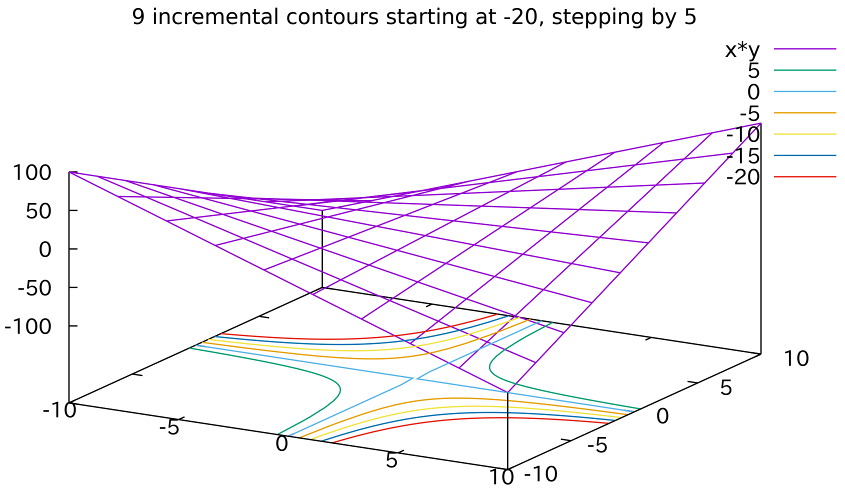
Test of spherical coordinates
"glass.dat"
1
0.5
0
-0.5
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
X axis
Y axis



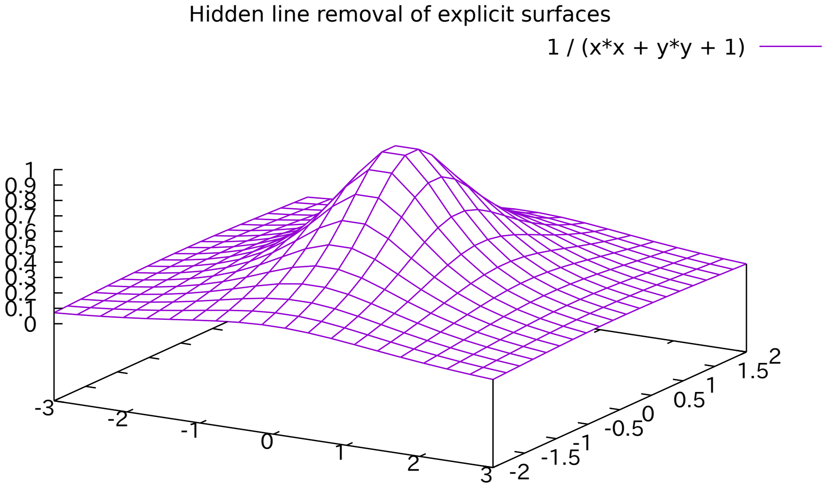
Mandelbrot function
mand({0,0},compl(x,y),30)
35
30
25
20
15
10
5
-2
-1.5
-1
-0.5
0
0.5
1
X axis
-1.5
-1
-0.5
0
0.5
1
1.5
Y axis

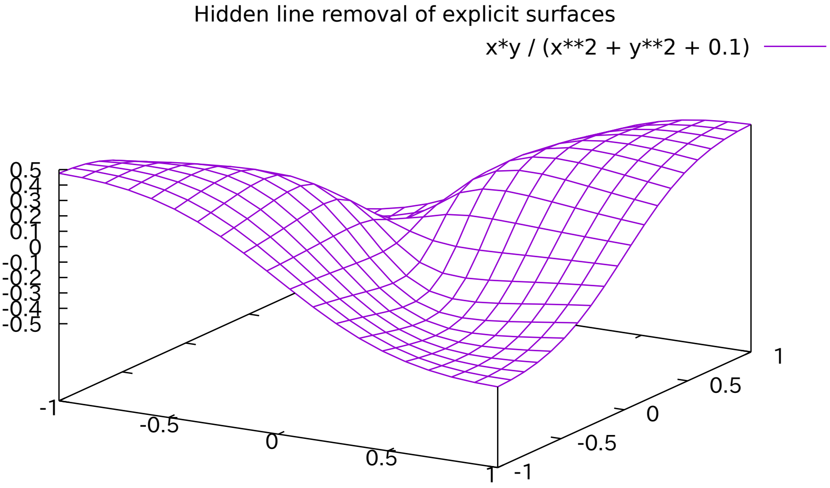

Demo of specifying discrete contour levels - default contours
x*y
100
50
0
-50
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10

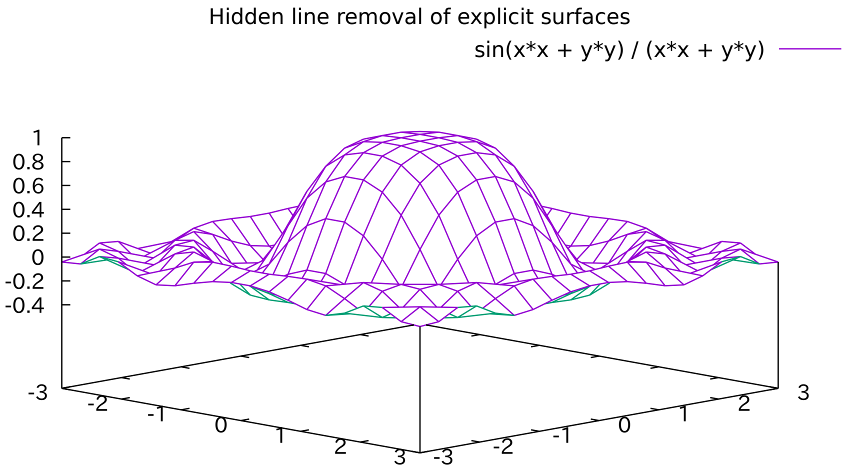

3 discrete contours at 0 15 75
x*y
75
15
0
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10



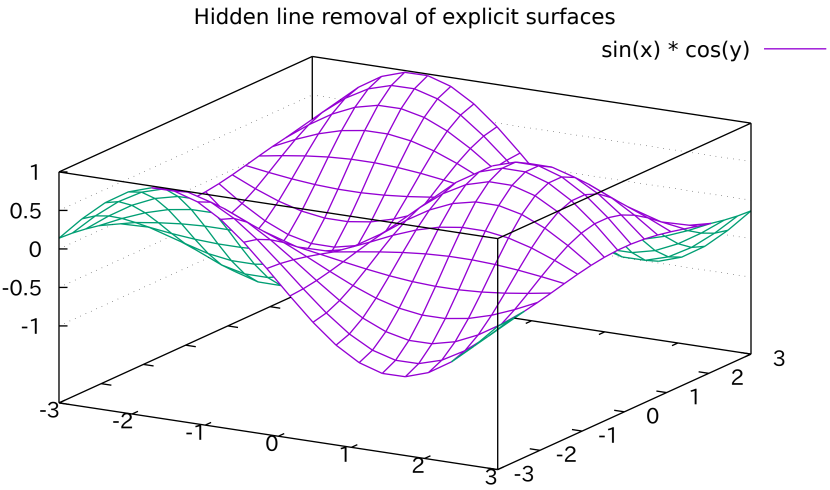
9 incremental contours starting at -20, stepping by 5
x*y
5
0
-5
-10
-15
-20
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
10


Hidden line removal of explicit surfaces
1 / (x*x + y*y + 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-3
-2
-1
0
1
2
3
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Hidden line removal of explicit surfaces
x*y / (x**2 + y**2 + 0.1)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1

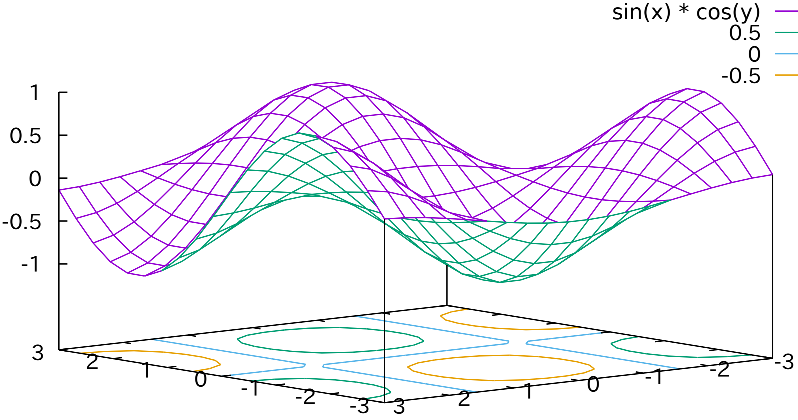


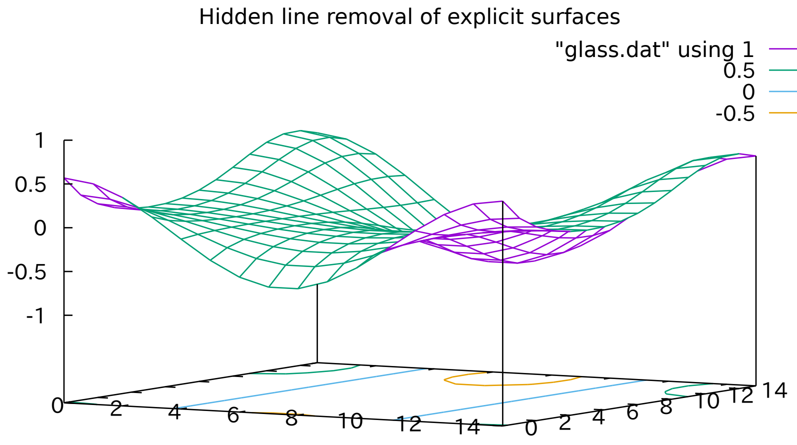
Hidden line removal of explicit surfaces
sin(x*x + y*y) / (x*x + y*y)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-3
-2
-1
0
1
2
3
-3
-2
-1
0
1
2
3



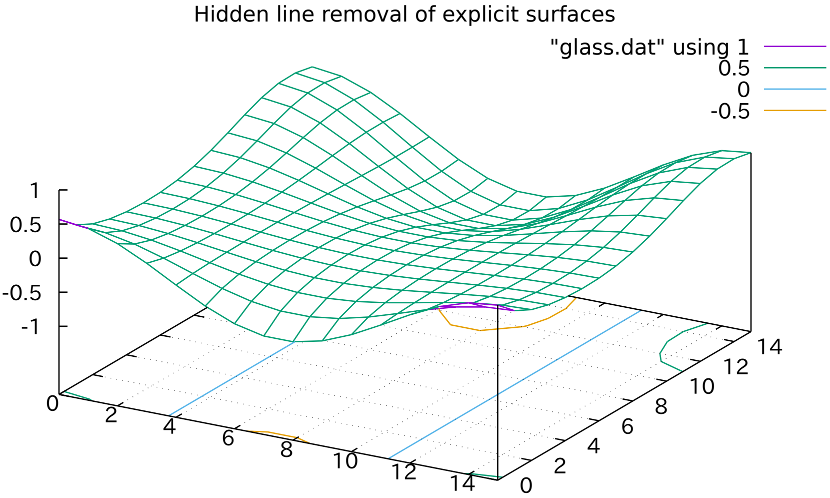
Hidden line removal of explicit surfaces
sin(x) * cos(y)
1
0.5
0
-0.5
-1
-3
-2
-1
0
1
2
3
-3
-2
-1
0
1
2
3

Hidden line removal of explicit surfaces

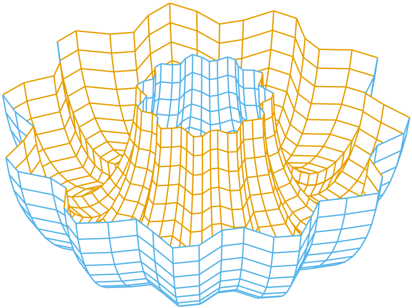




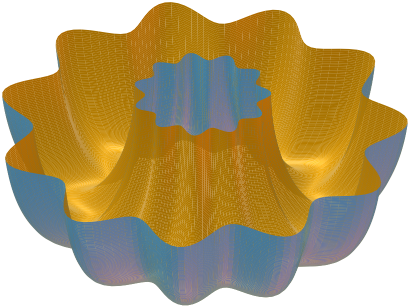
Hidden line removal of explicit surfaces
"glass.dat" using 1
0.5
0
-0.5
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

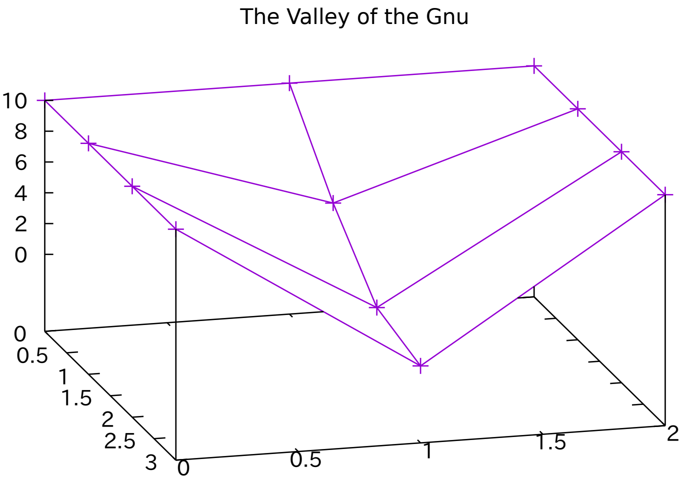

Hidden line removal of explicit surfaces
"glass.dat" using 1
0.5
0
-0.5
1
0.5
0
-0.5
-1
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

hidden3d 2-color surface

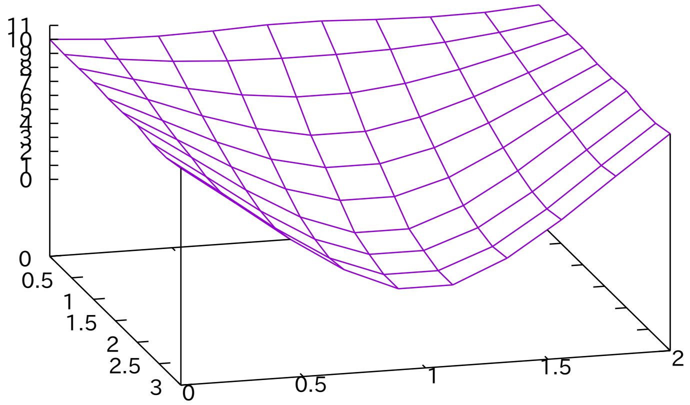


pm3d 2-color surface

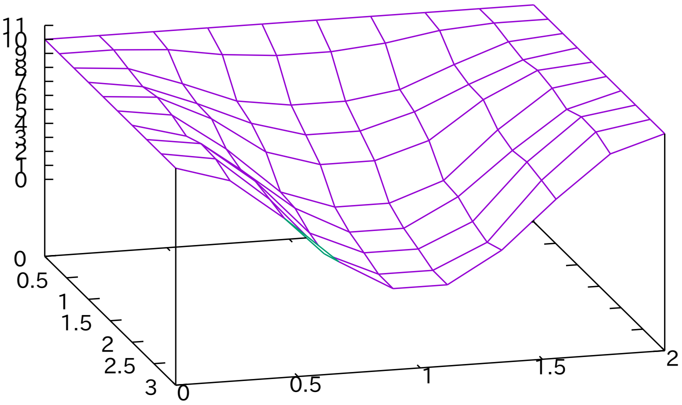


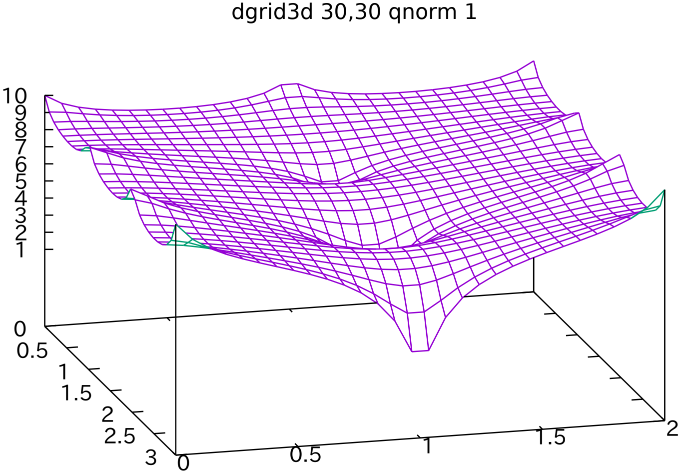

The Valley of the Gnu
10
8
6
4
2
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

dgrid3d splines

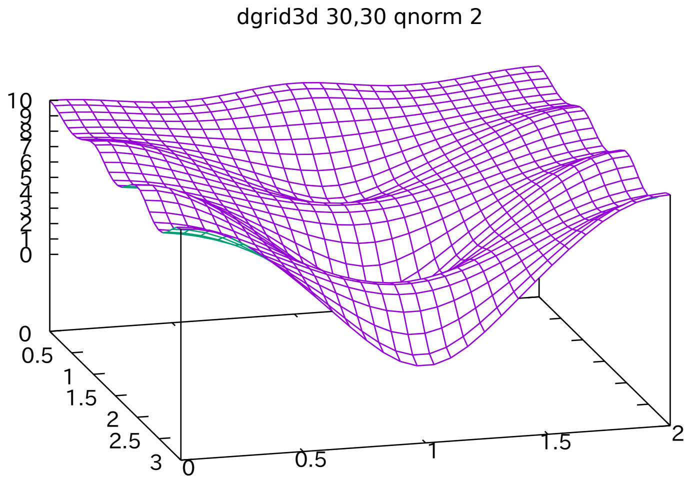


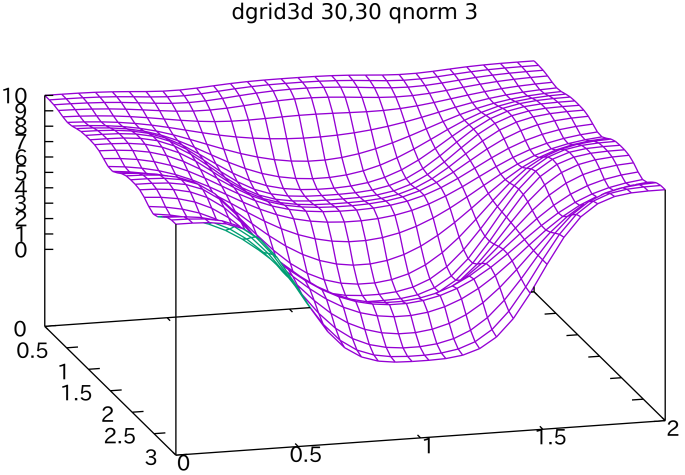
dgrid3d Hann function

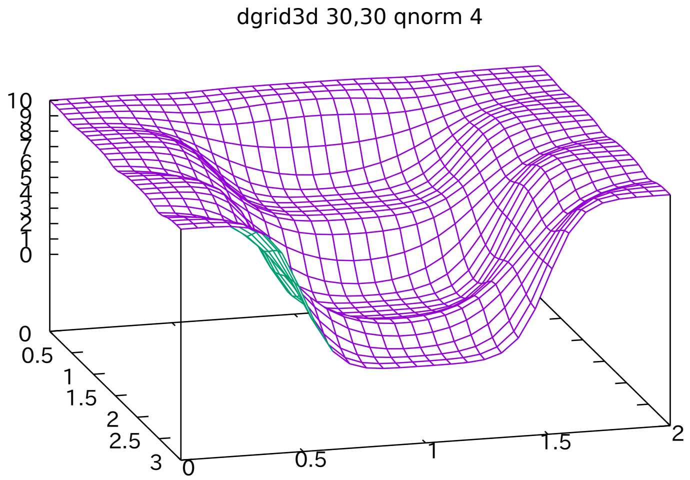




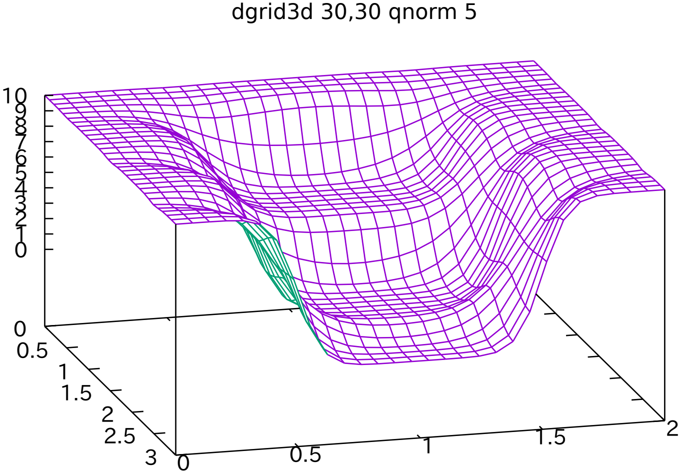
dgrid3d 30,30 qnorm 1
10
9
8
7
6
5
4
3
2
1
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

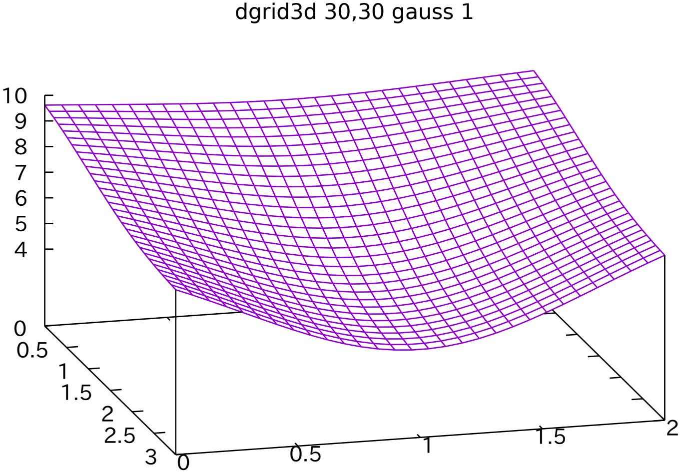

dgrid3d 30,30 qnorm 2
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

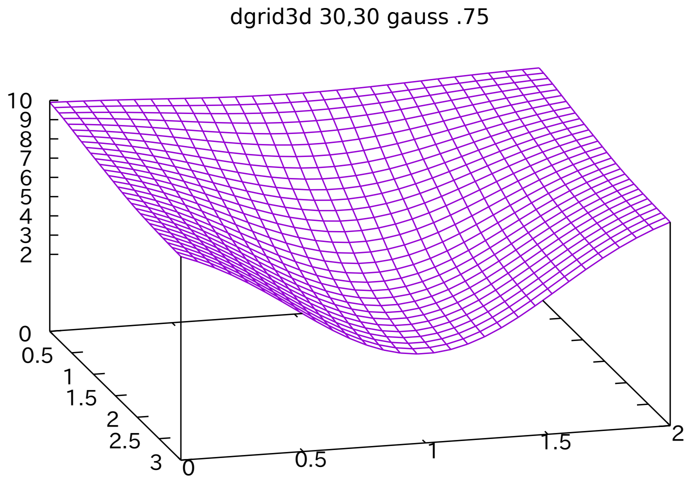

dgrid3d 30,30 qnorm 3
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

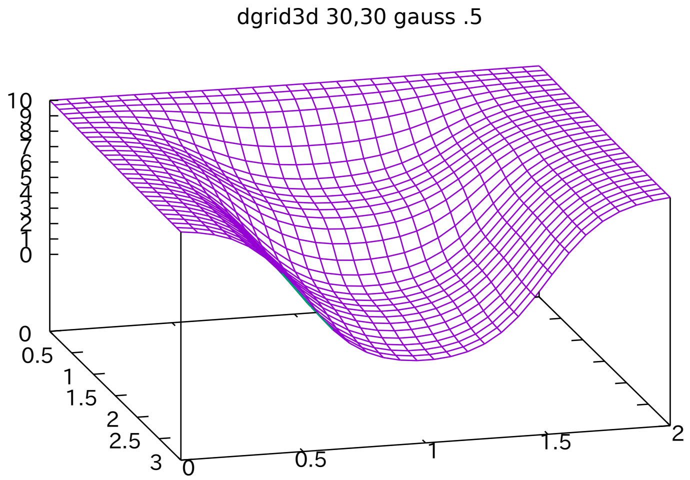


dgrid3d 30,30 qnorm 4
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2


dgrid3d 30,30 qnorm 5
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

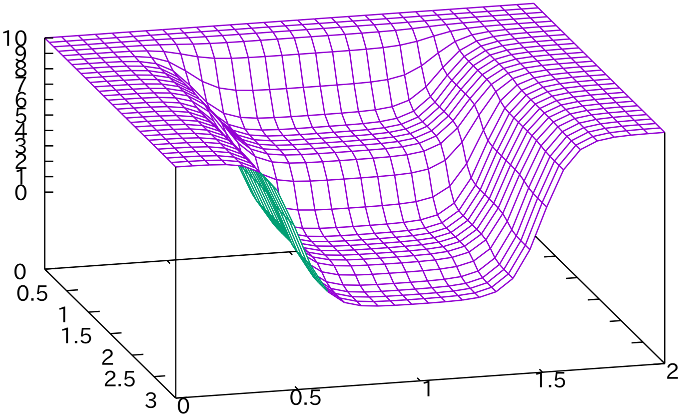

dgrid3d 30,30 gauss 1
10
9
8
7
6
5
4
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

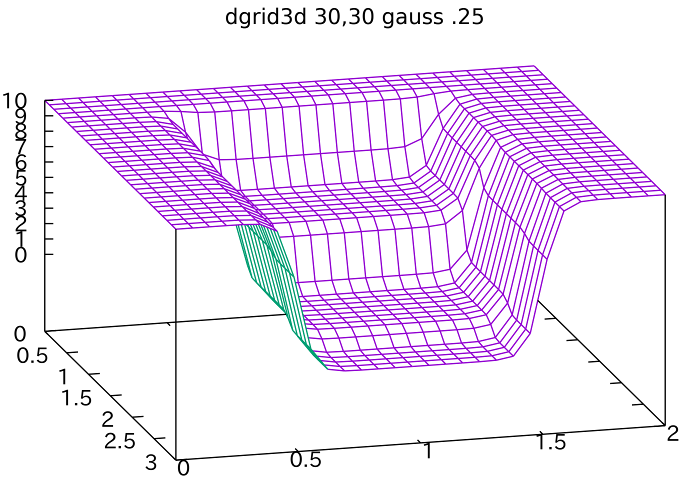

dgrid3d 30,30 gauss .75
10
9
8
7
6
5
4
3
2
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

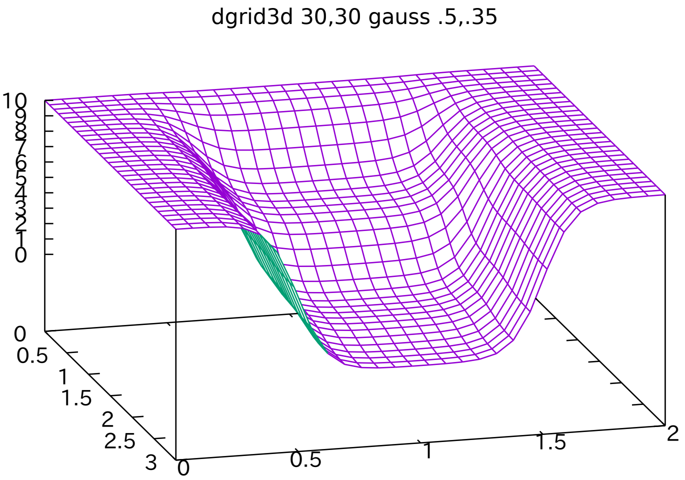


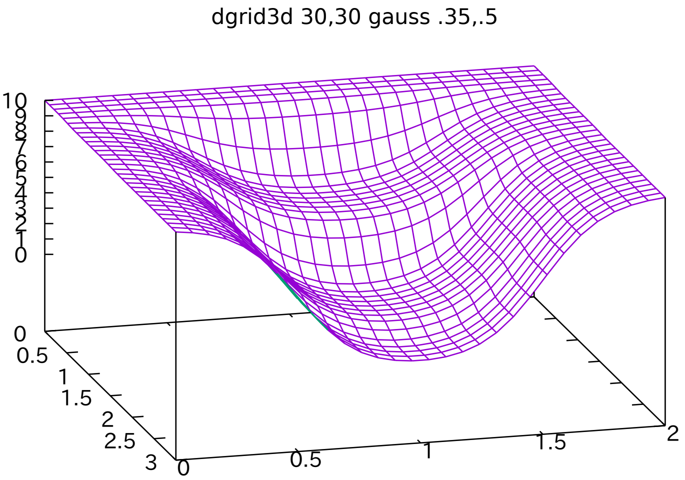
dgrid3d 30,30 gauss .5
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

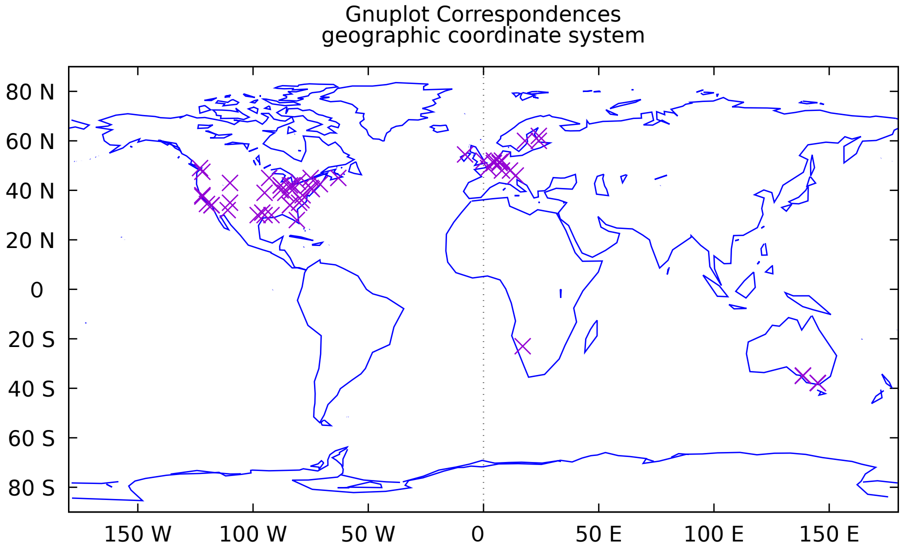
dgrid3d 30,30 gauss .35



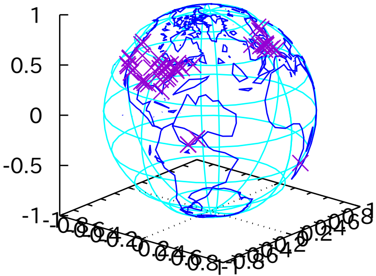

dgrid3d 30,30 gauss .25
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

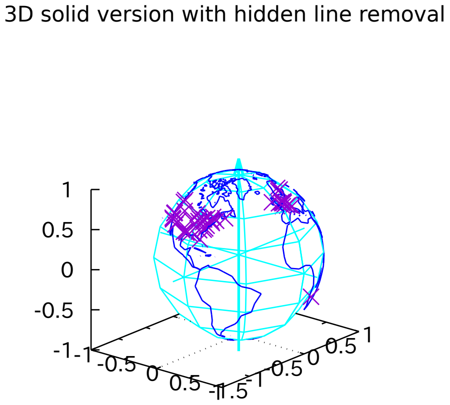

dgrid3d 30,30 gauss .5,.35
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2


dgrid3d 30,30 gauss .35,.5
10
9
8
7
6
5
4
3
2
1
0
0
0.5
1
1.5
2
2.5
3
0
0.5
1
1.5
2

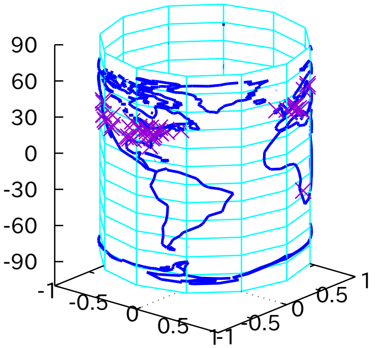

Gnuplot Correspondences
geographic coordinate system
80 N
60 N
40 N
20 N
0
20 S
40 S
60 S
80 S
150 W
100 W
50 W
0
50 E
100 E
150 E

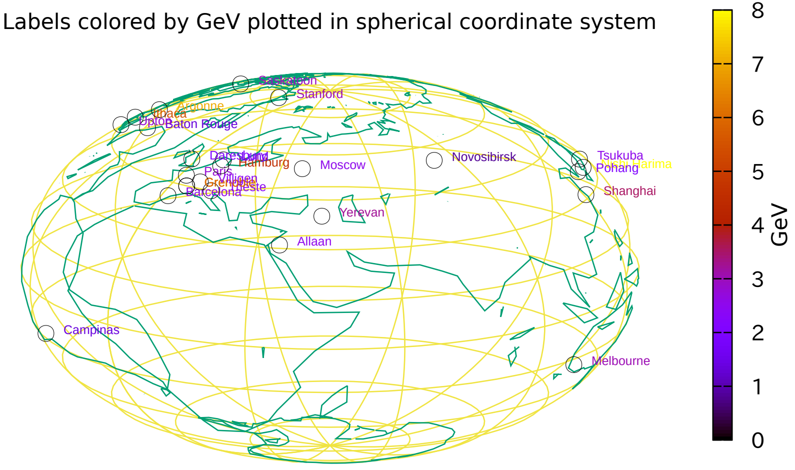
3D version using spherical coordinate system




3D solid version with hidden line removal
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
-1.5
-1
-0.5
0
0.5
1

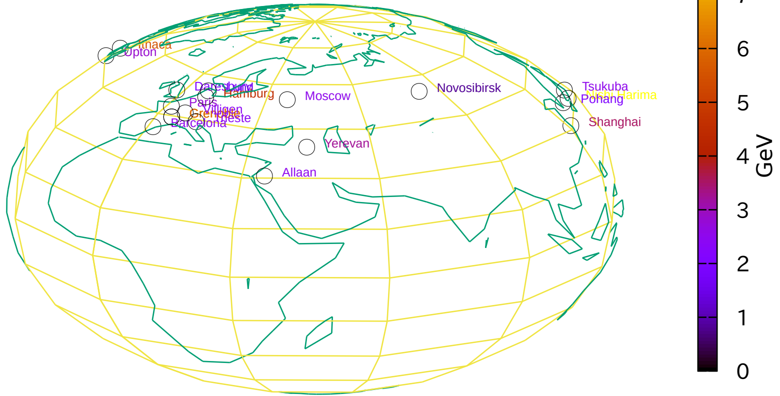
3D version using cylindrical coordinate system

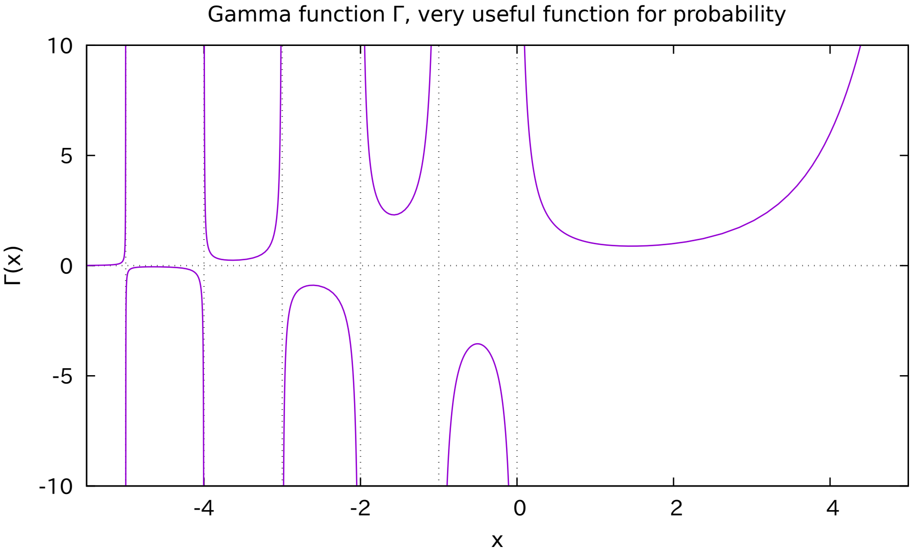




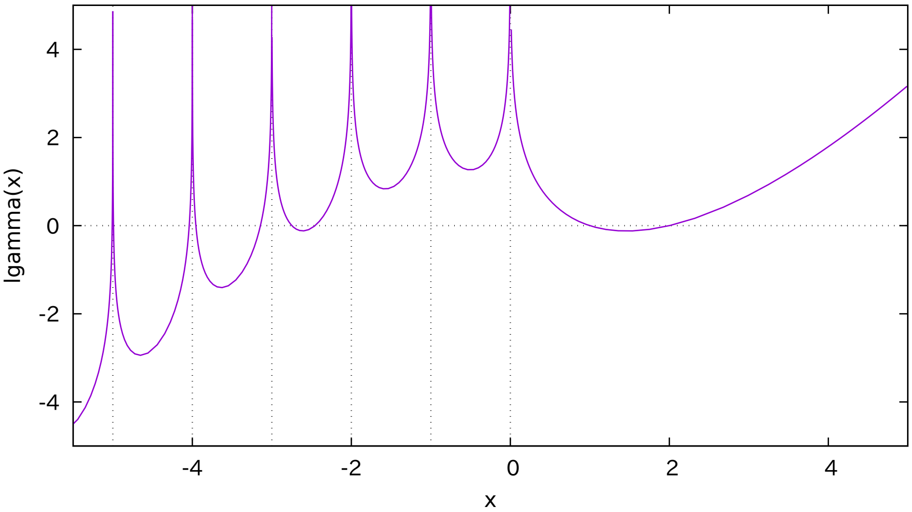
Labels colored by GeV plotted in spherical coordinate system
Saskatoon
Stanford
Argonne
Ithaca
Upton
Baton Rouge
Daresbury
Lund
Hamburg
Paris
Villigen
Grenoble
Trieste
Barcelona
Moscow
Novosibirsk
Tsukuba
Pohang
Nishi Harima
Shanghai
Yerevan
Allaan
Campinas
Melbourne
GeV
0
1
2
3
4
5
6
7
8

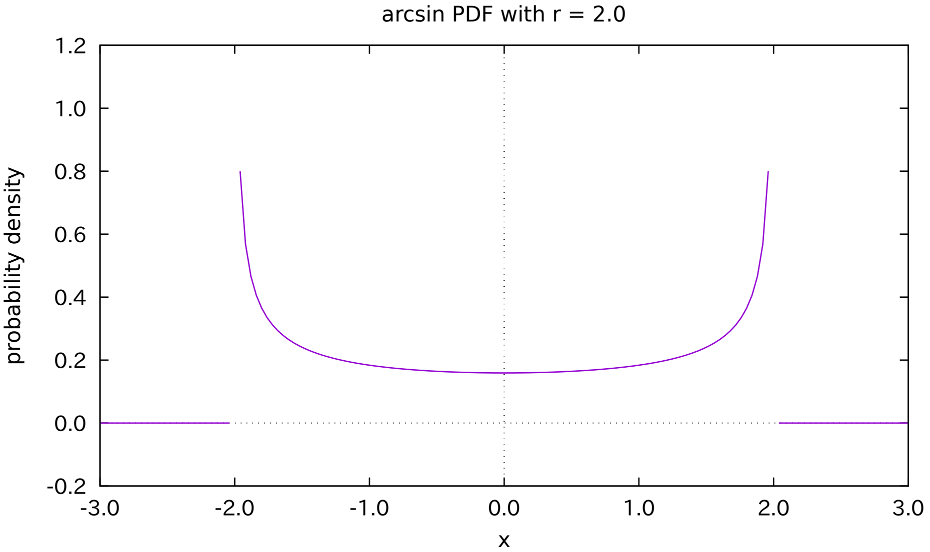
Labels with hidden line removal

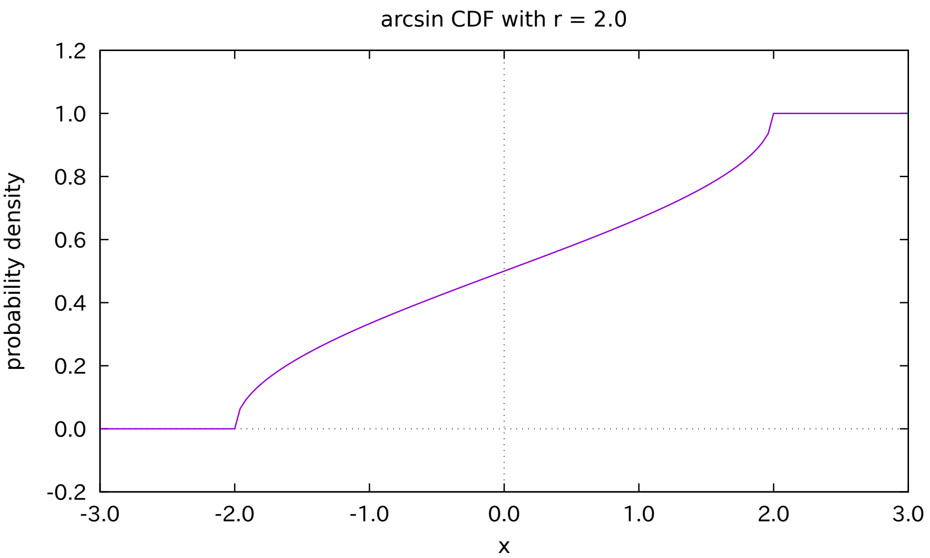


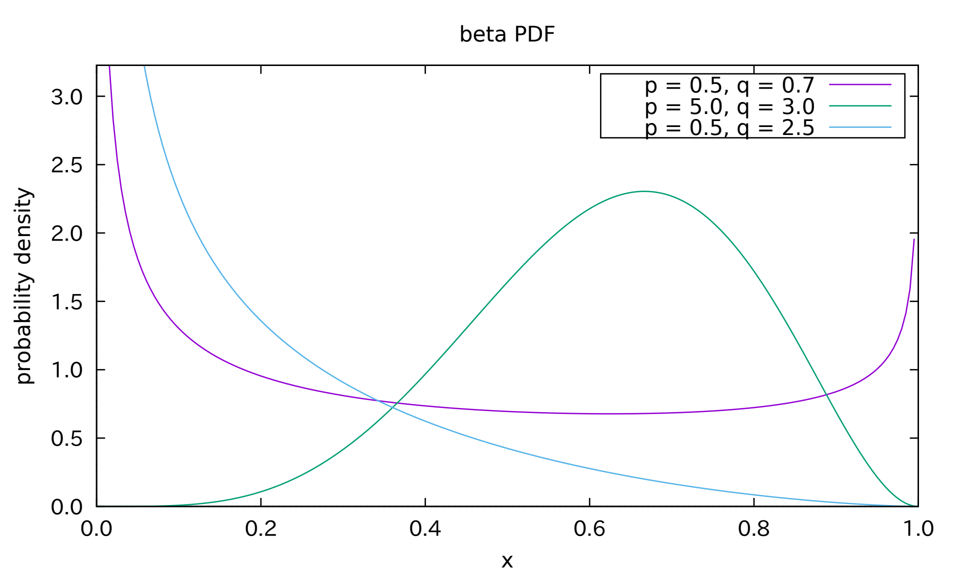

Gamma function Γ, very useful function for probability
10
5
0
-5
-10
Γ(x)
-4
-2
0
2
4
x

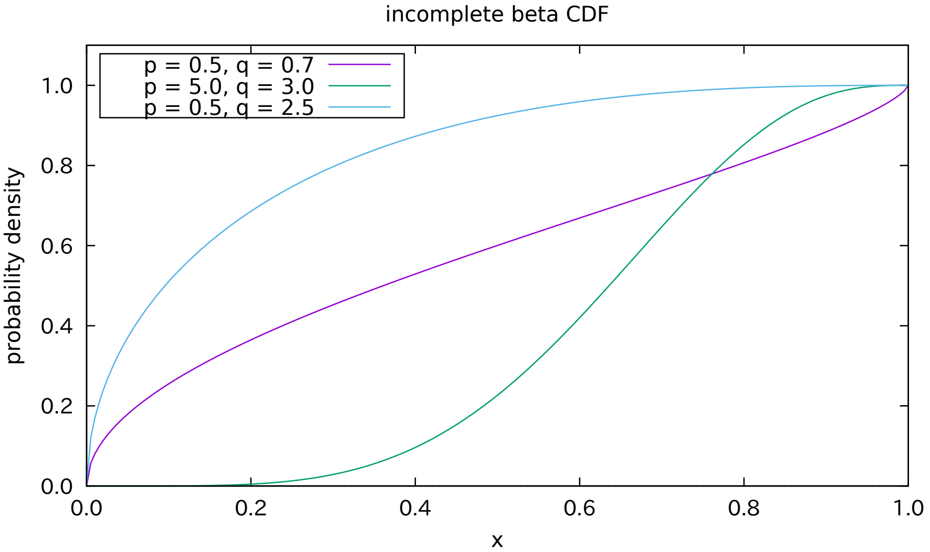
log gamma function, similarly very useful function

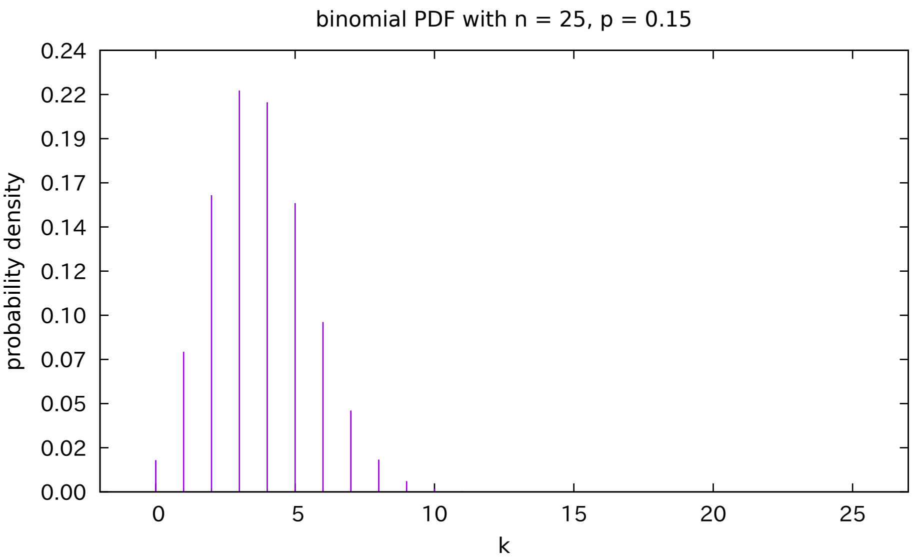


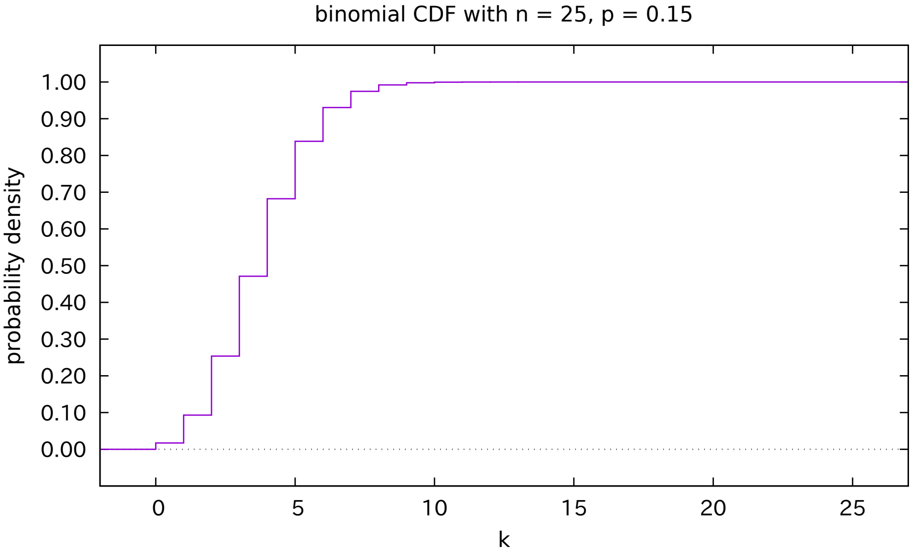

arcsin PDF with r = 2.0
probability density
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0


arcsin CDF with r = 2.0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
probability density
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
x

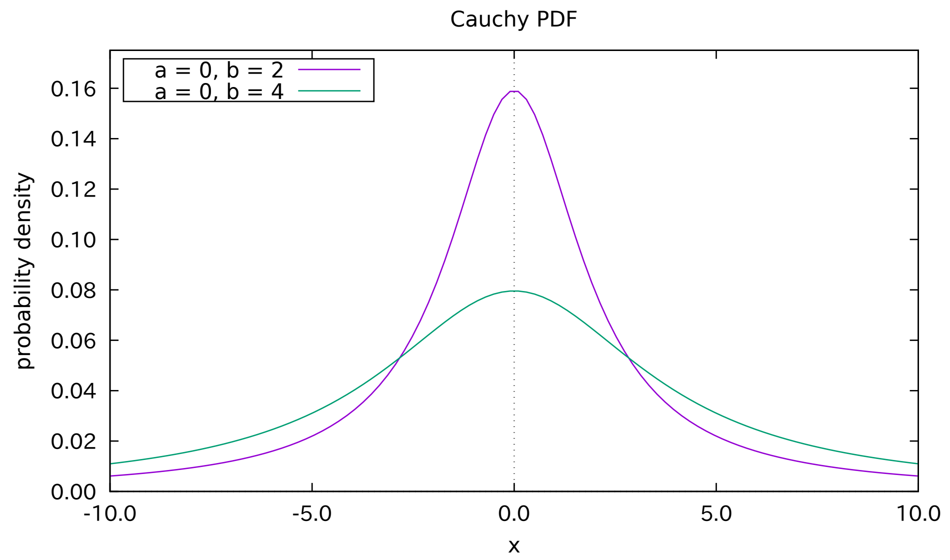

beta PDF
p = 0.5, q = 0.7
p = 5.0, q = 3.0
p = 0.5, q = 2.5
probability density
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.2
0.4
0.6
0.8
1.0

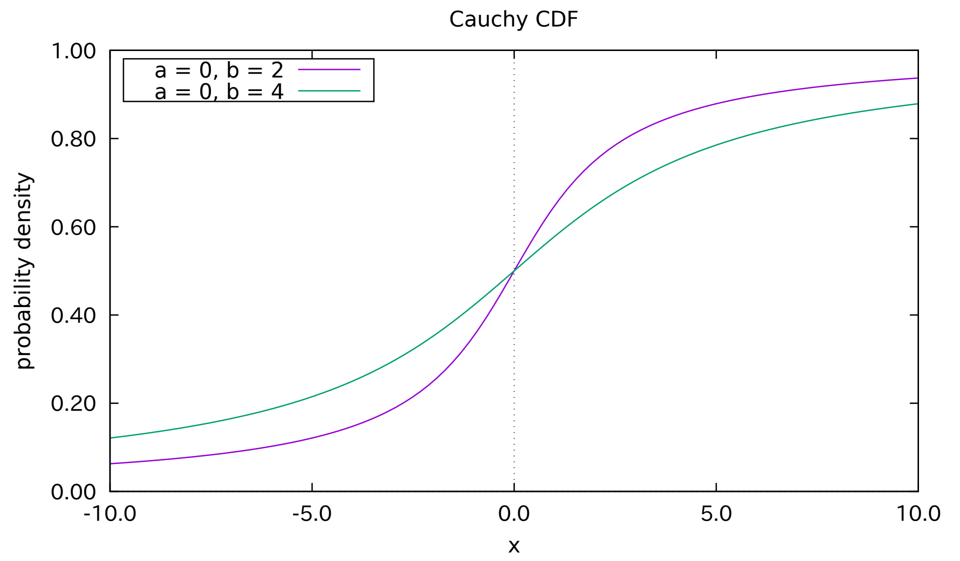

incomplete beta CDF
p = 0.5, q = 0.7
p = 5.0, q = 3.0
p = 0.5, q = 2.5
probability density
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x


binomial PDF with n = 25, p = 0.15
probability density
0.24
0.22
0.19
0.17
0.14
0.12
0.10
0.07
0.05
0.02
0.00
0
5
10
15
20
25
k

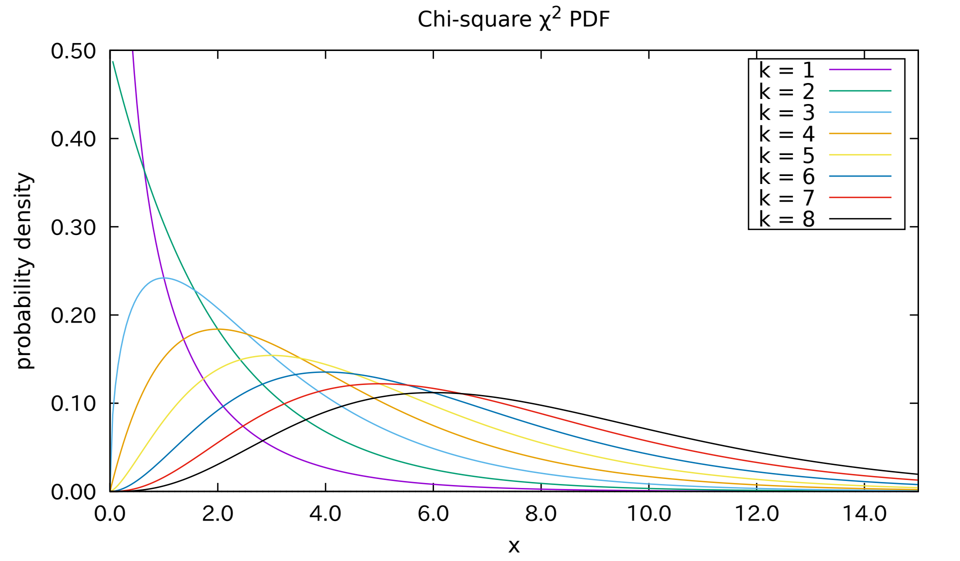


binomial CDF with n = 25, p = 0.15
probability density
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0
5
10
15
20
25
k

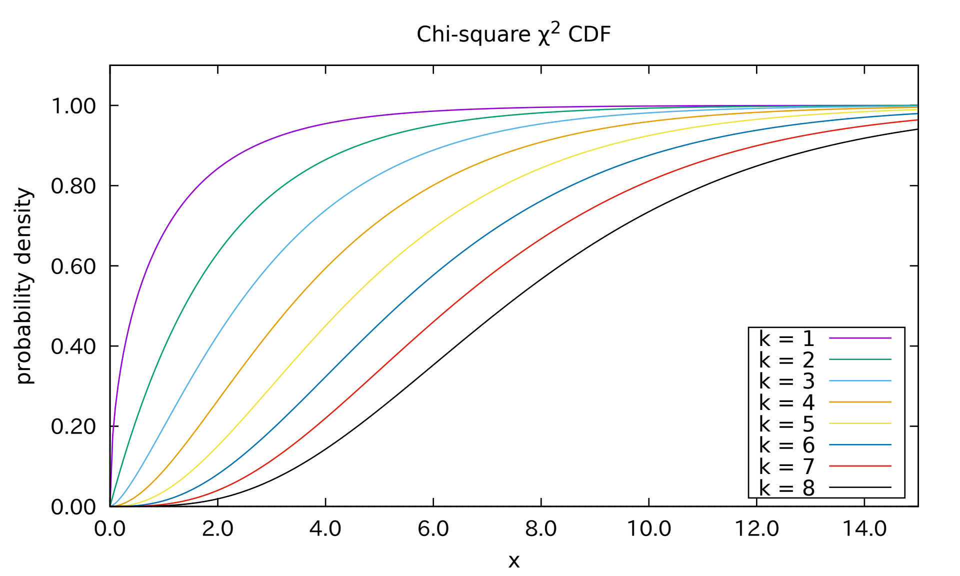

Cauchy PDF
a = 0, b = 2
a = 0, b = 4
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
probability density
-10.0
-5.0
0.0
5.0
10.0
x



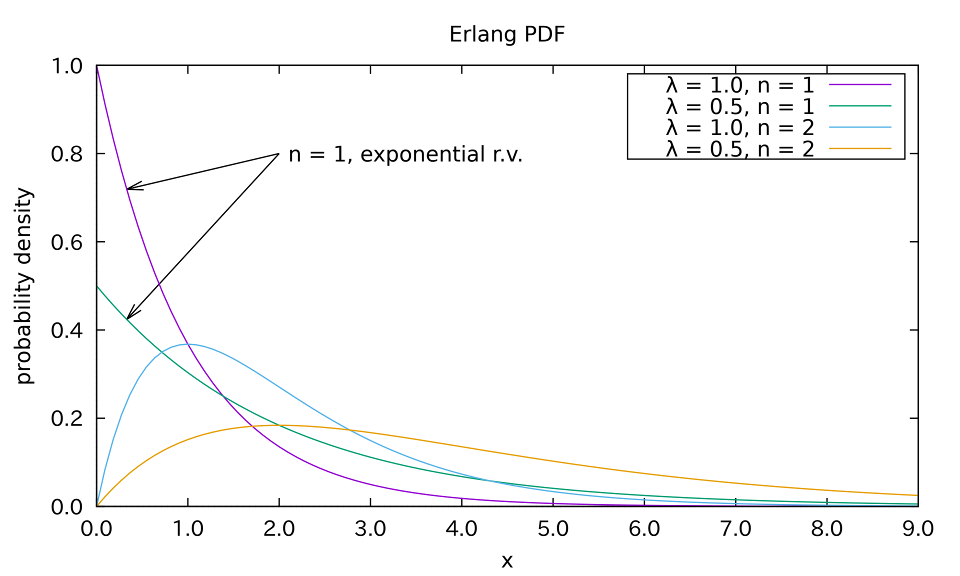
Cauchy CDF
a = 0, b = 2
a = 0, b = 4
probability density
1.00
0.80
0.60
0.40
0.20
0.00
-10.0
-5.0
0.0
5.0
10.0
x



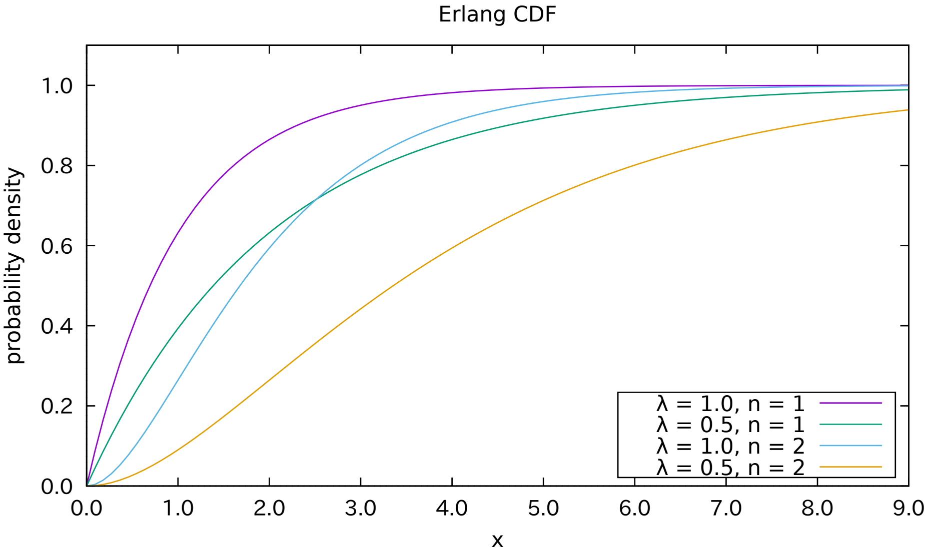
Chi-square χ2 PDF
0.50
0.40
0.30
0.20
0.10
0.00
probability density
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
x
k = 1
k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8

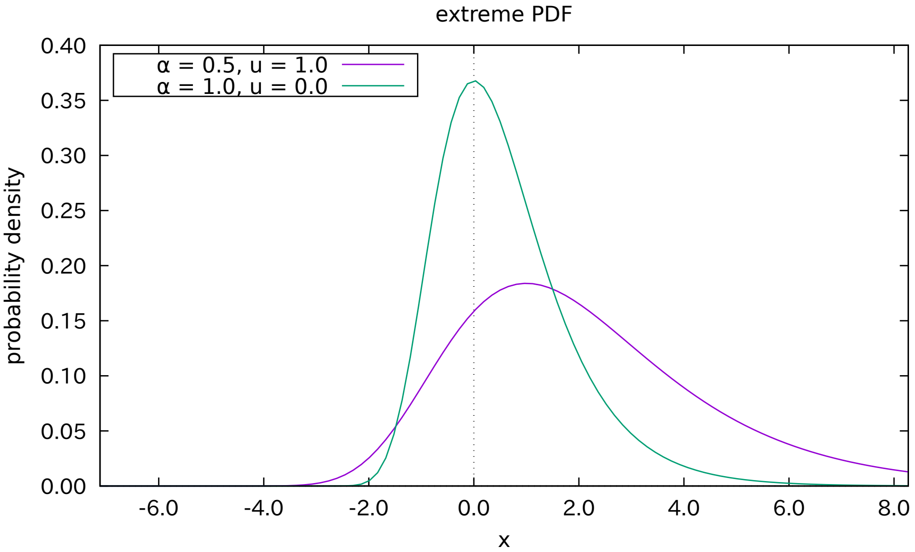


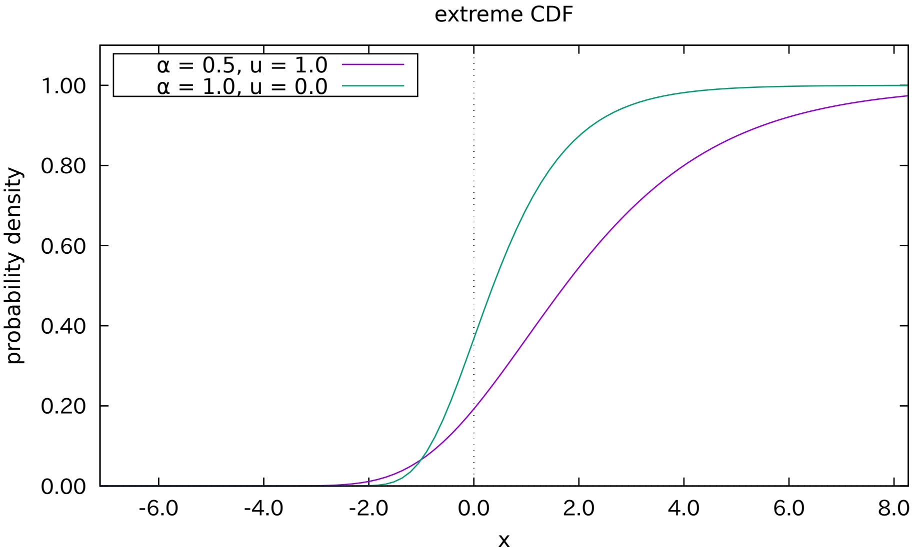
Chi-square χ2 CDF
1.00
0.80
0.60
0.40
0.20
0.00
probability density
0.0
2.0
4.0
6.0
8.0
10.0
12.0
14.0
x
k = 1
k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8



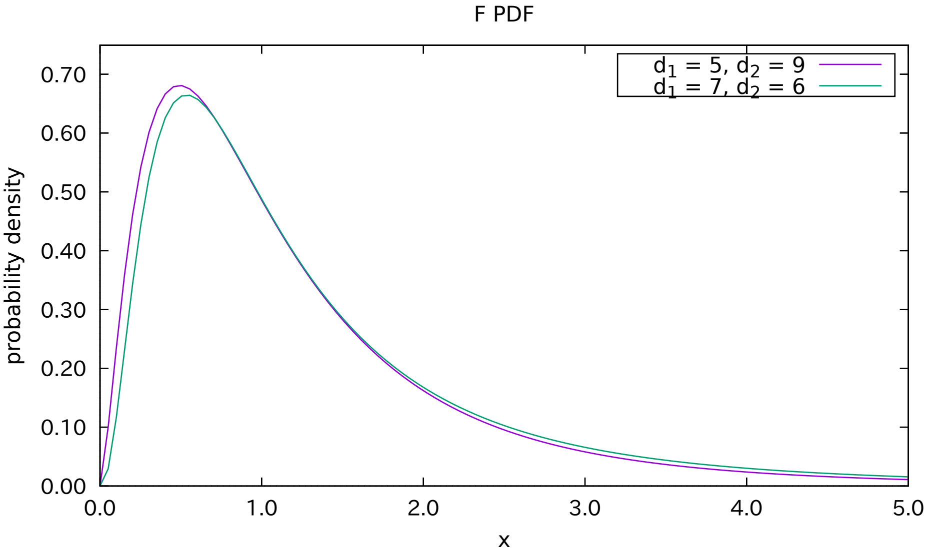
Erlang PDF
1.0
0.8
0.6
0.4
0.2
0.0
probability density
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
x
λ = 1.0, n = 1
λ = 0.5, n = 1
λ = 1.0, n = 2
λ = 0.5, n = 2
n = 1, exponential r.v.


Erlang CDF
probability density
x
λ = 1.0, n = 1
λ = 0.5, n = 1
λ = 1.0, n = 2
λ = 0.5, n = 2
0.0
0.2
0.4
0.6
0.8
1.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0

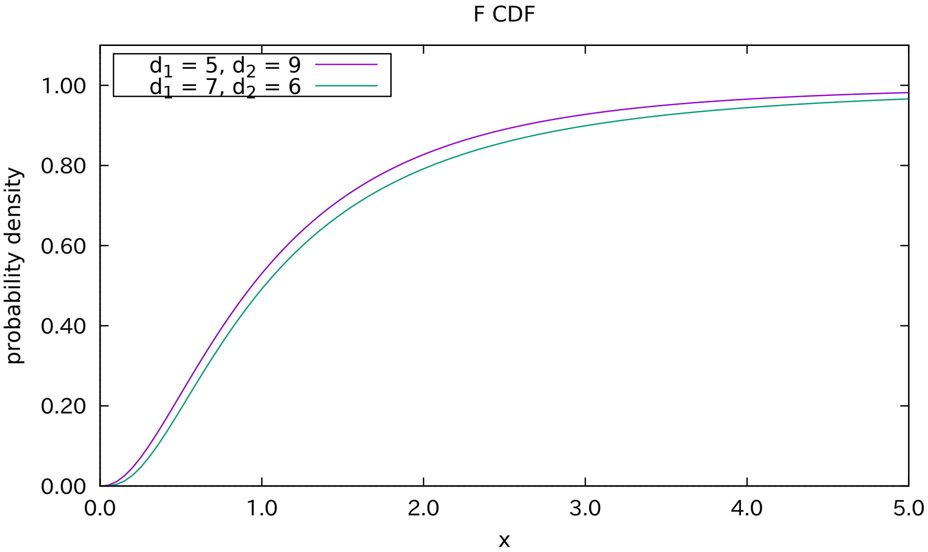

extreme PDF
α = 0.5, u = 1.0
α = 1.0, u = 0.0
probability density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
-6.0
-4.0
-2.0
0.0
2.0
4.0
6.0
8.0
x

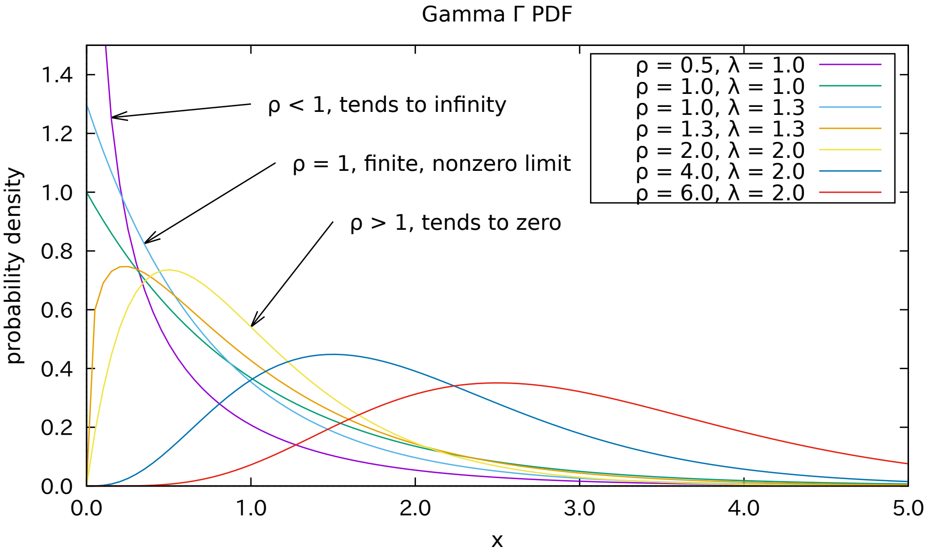


extreme CDF
α = 0.5, u = 1.0
α = 1.0, u = 0.0
1.00
0.80
0.60
0.40
0.20
0.00
probability density
-6.0
-4.0
-2.0
0.0
2.0
4.0
6.0
8.0
x

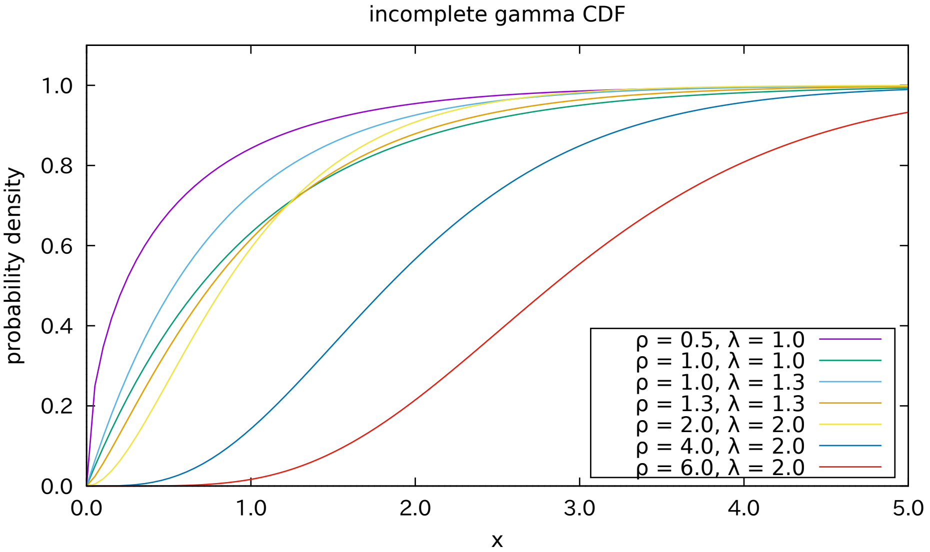


F PDF
probability density
x
$d_1 = 5, d_2 = 9$
$d_1 = 7, d_2 = 6$
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.0
1.0
2.0
3.0
4.0
5.0

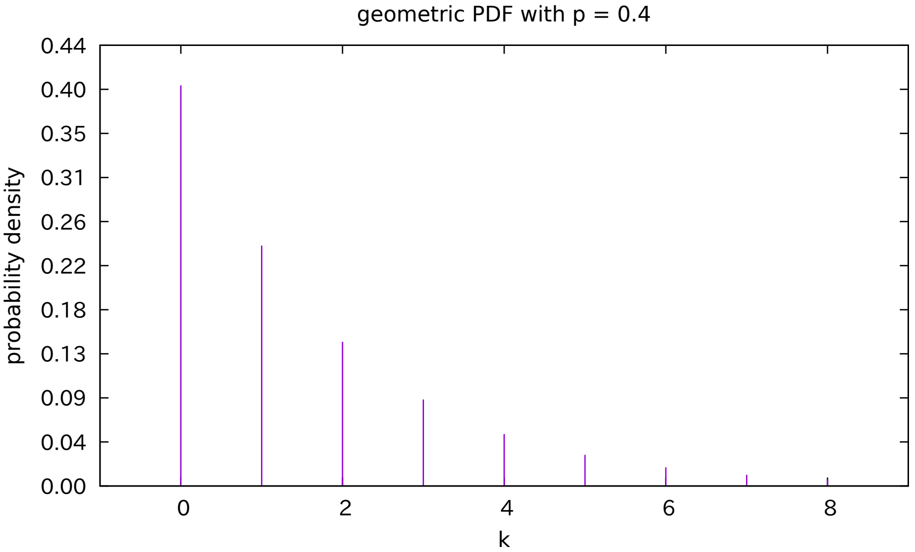

F CDF
d_1 = 5, d_2 = 9
d_1 = 7, d_2 = 6
probability density
1.00
0.80
0.60
0.40
0.20
0.00
0.0
1.0
2.0
3.0
4.0
5.0
x



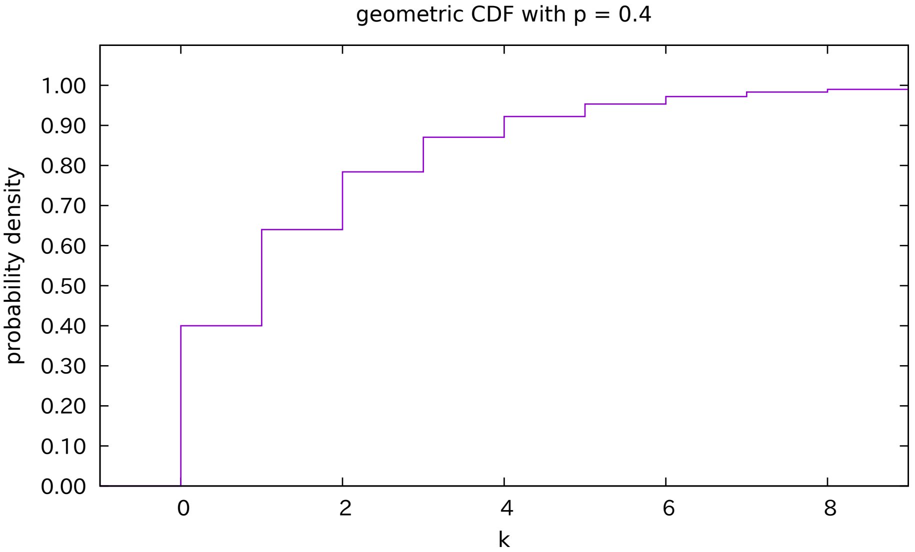
Gamma Γ PDF
ρ = 0.5, λ = 1.0
ρ = 1.0, λ = 1.0
ρ = 1.0, λ = 1.3
ρ = 1.3, λ = 1.3
ρ = 2.0, λ = 2.0
ρ = 4.0, λ = 2.0
ρ = 6.0, λ = 2.0
ρ < 1, tends to infinity
ρ = 1, finite, nonzero limit
ρ > 1, tends to zero
probability density
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
x

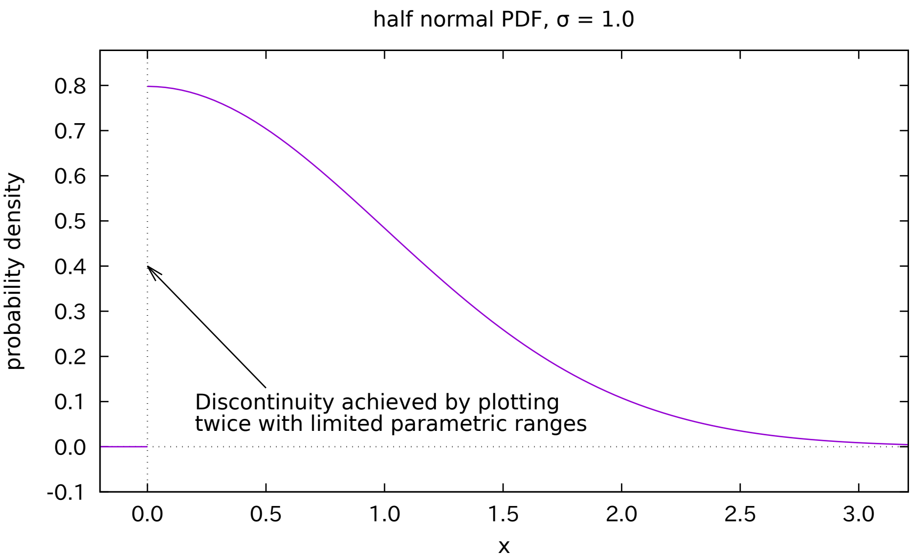


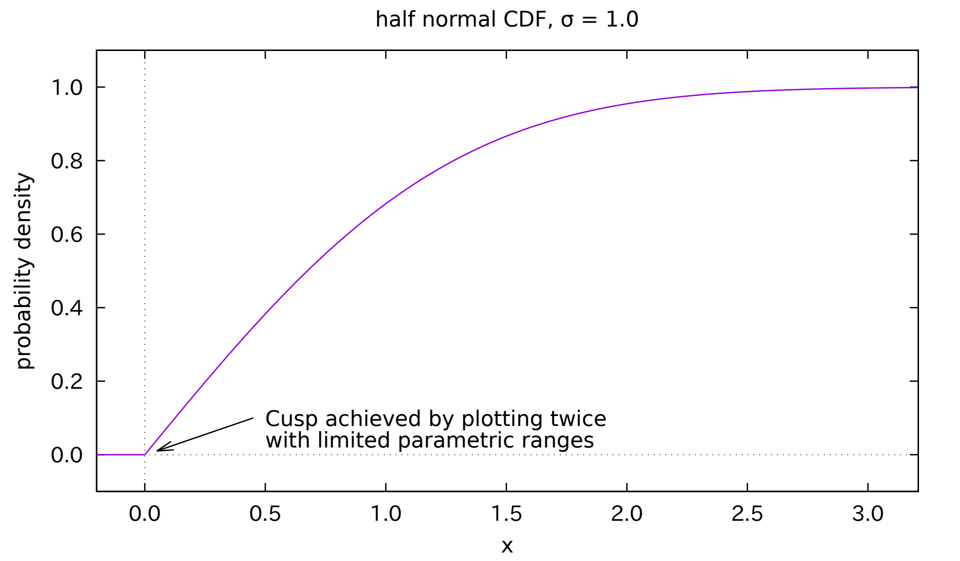
incomplete gamma CDF
probability density
x
0.0
0.2
0.4
0.6
0.8
1.0
0.0
1.0
2.0
3.0
4.0
5.0
ρ = 0.5, λ = 1.0
ρ = 1.0, λ = 1.0
ρ = 1.0, λ = 1.3
ρ = 1.3, λ = 1.3
ρ = 2.0, λ = 2.0
ρ = 4.0, λ = 2.0
ρ = 6.0, λ = 2.0



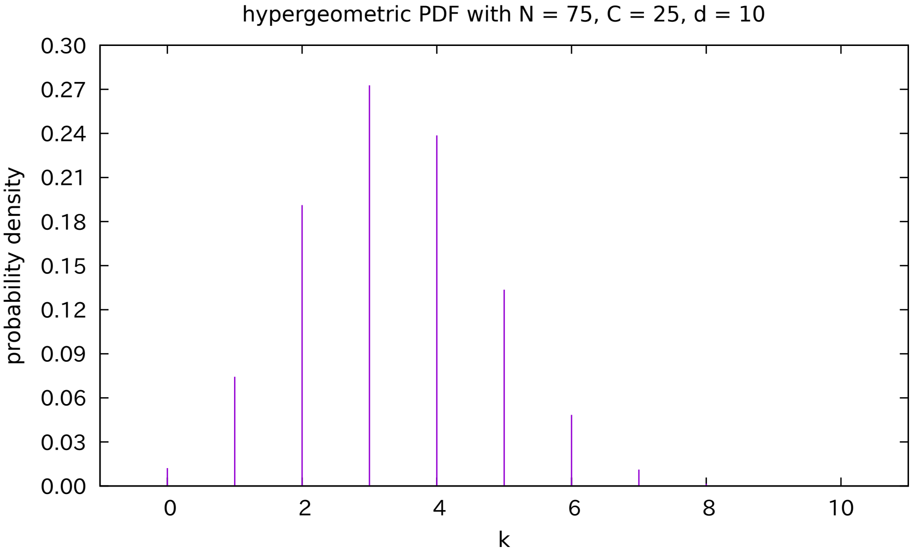
geometric PDF with p = 0.4
probability density
0.44
0.40
0.35
0.31
0.26
0.22
0.18
0.13
0.09
0.04
0.00
0
2
4
6
8
k


geometric CDF with p = 0.4
probability density
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0
2
4
6
8
k

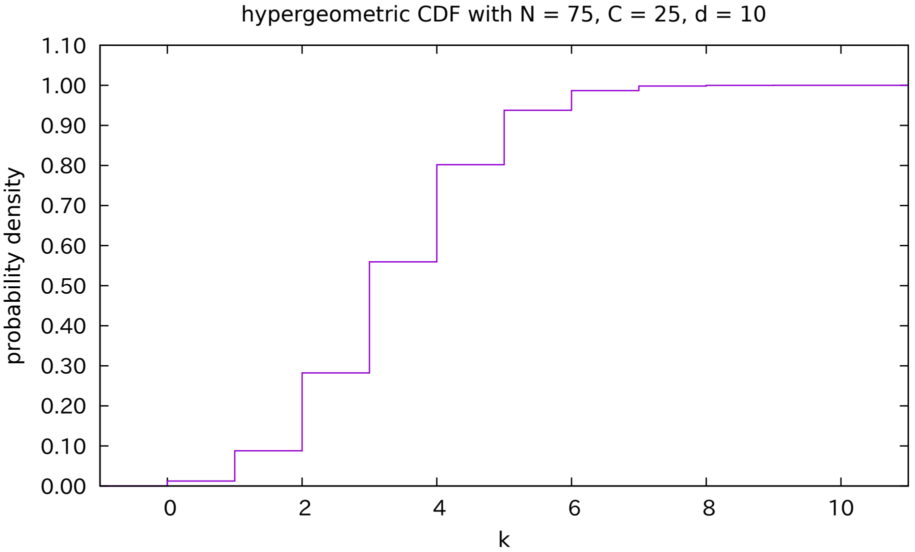

half normal PDF, σ = 1.0
probability density
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x
Discontinuity achieved by plotting
twice with limited parametric ranges

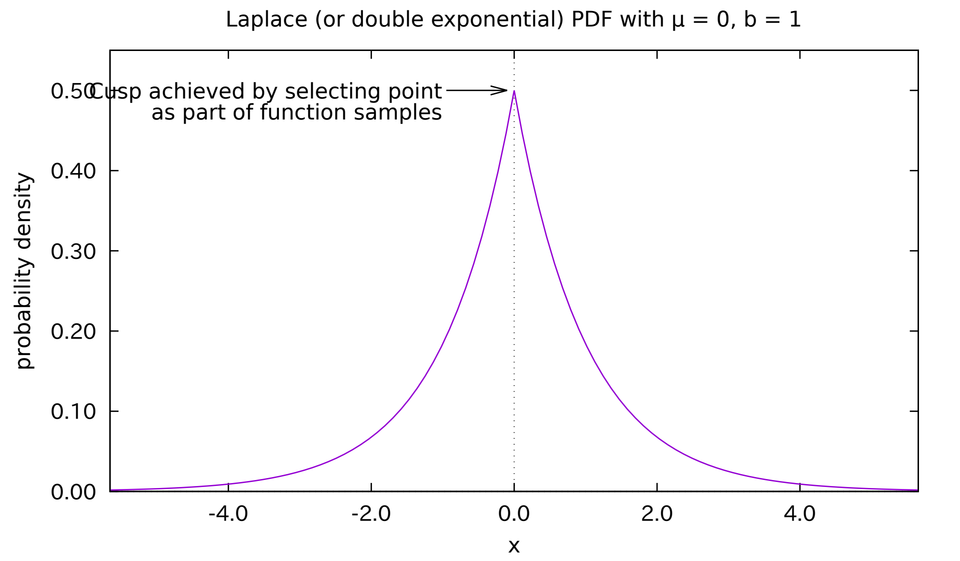


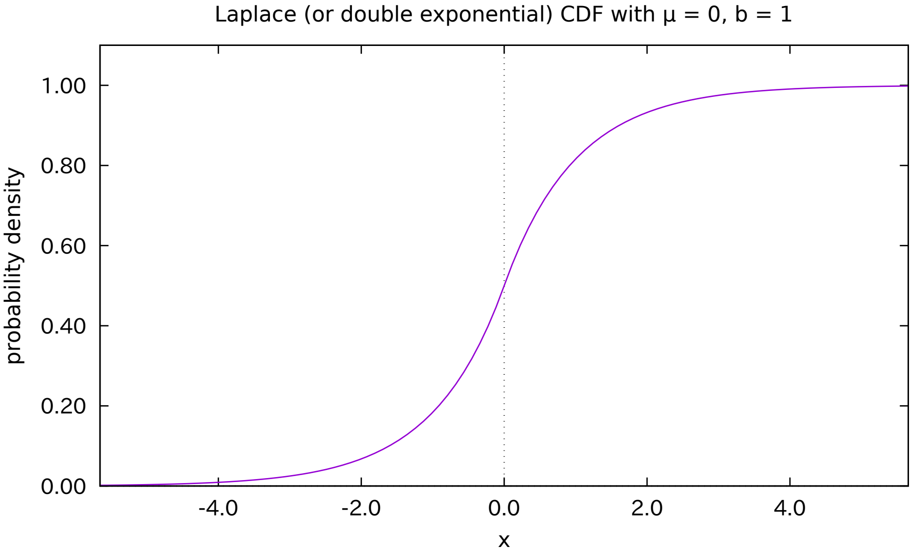
half normal CDF, σ = 1.0
1.0
0.8
0.6
0.4
0.2
0.0
probability density
Cusp achieved by plotting twice
with limited parametric ranges
0.0
0.5
1.0
1.5
2.0
2.5
3.0
x

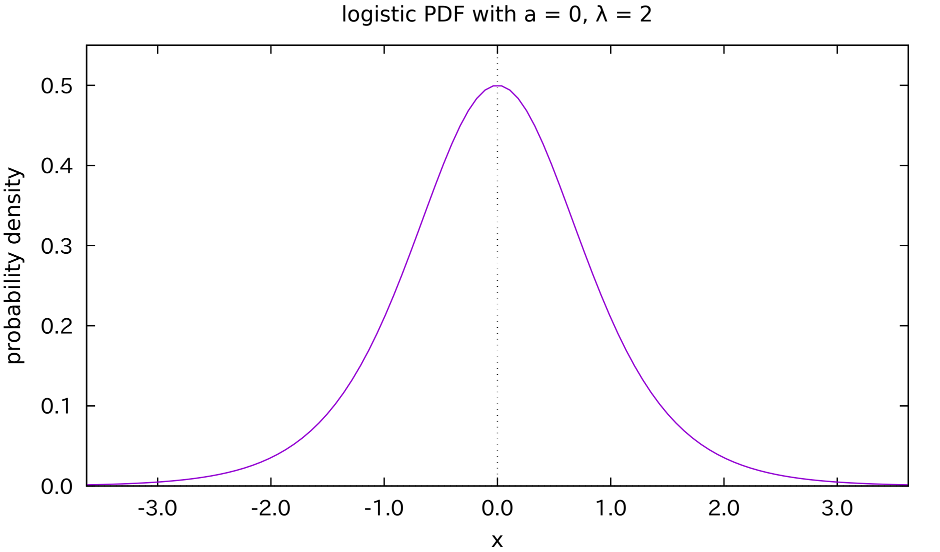


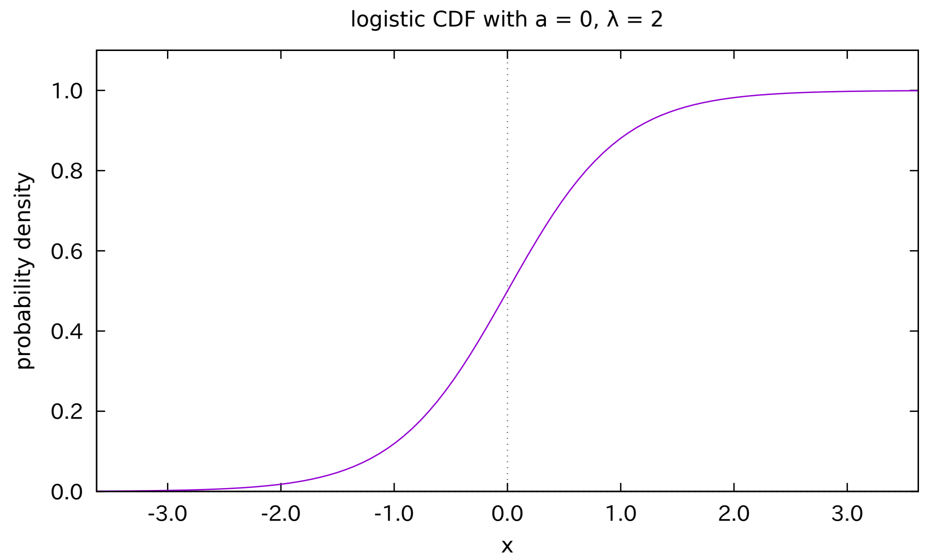
hypergeometric PDF with N = 75, C = 25, d = 10
probability density
k
0.30
0.27
0.24
0.21
0.18
0.15
0.12
0.09
0.06
0.03
0.00
0
2
4
6
8
10

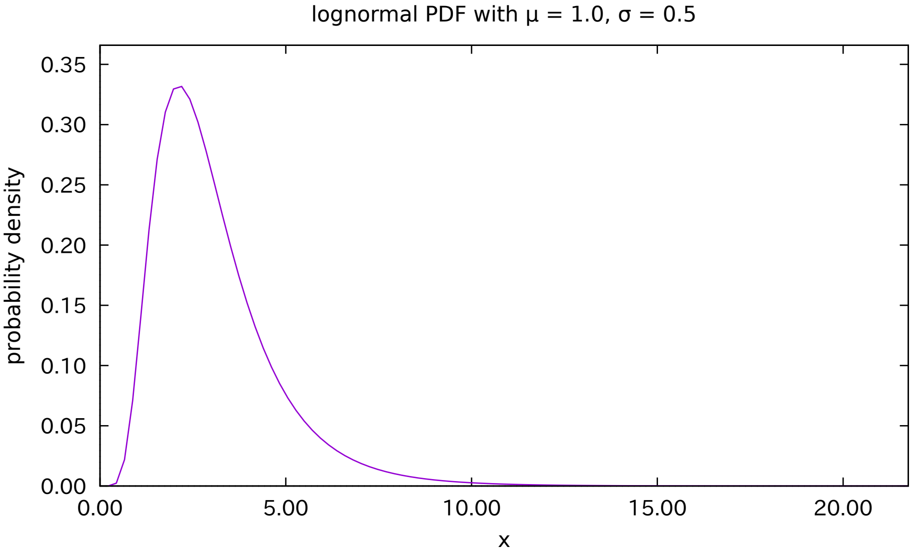

hypergeometric CDF with N = 75, C = 25, d = 10
probability density
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0
2
4
6
8
10
k

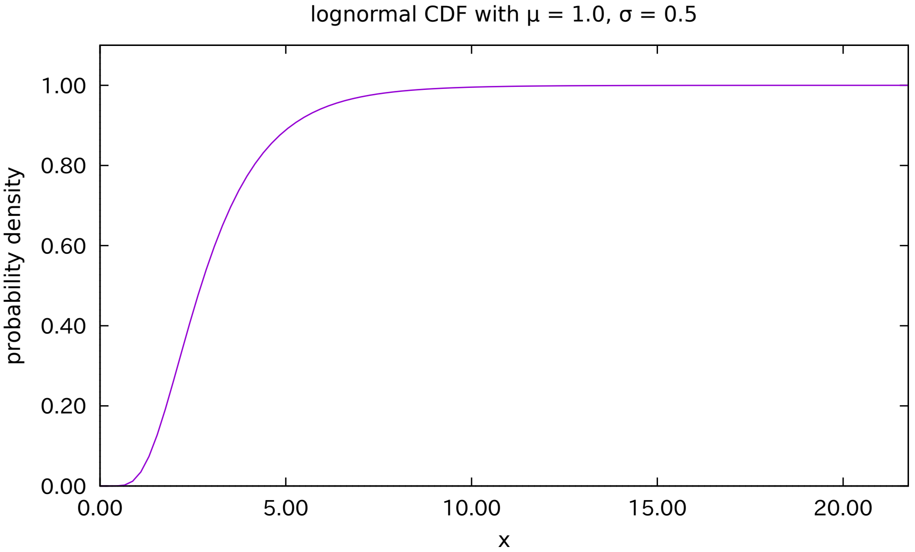

Laplace (or double exponential) PDF with μ = 0, b = 1
Cusp achieved by selecting point
as part of function samples
probability density
0.00
0.10
0.20
0.30
0.40
0.50
-4.0
-2.0
0.0
2.0
4.0
x


Laplace (or double exponential) CDF with μ = 0, b = 1
1.00
0.80
0.60
0.40
0.20
0.00
probability density
-4.0
-2.0
0.0
2.0
4.0
x

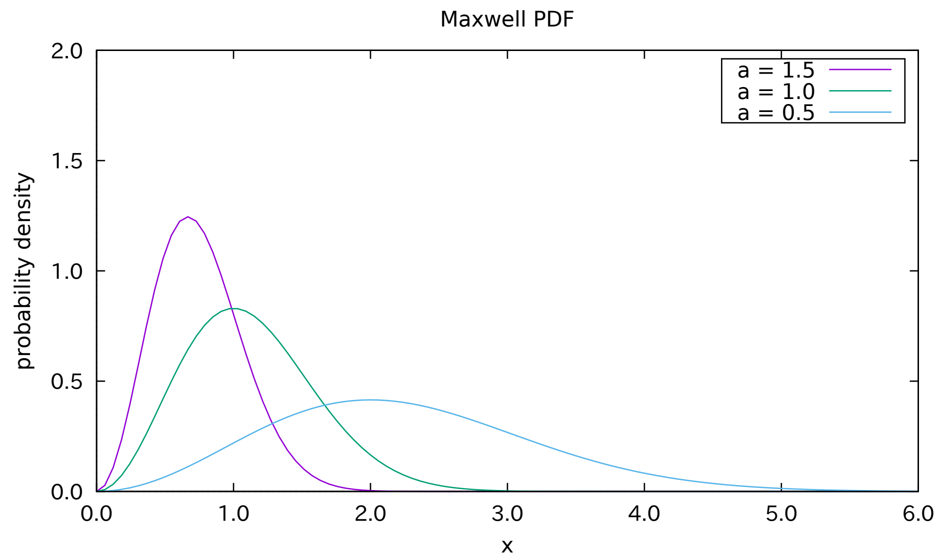

logistic PDF with a = 0, λ = 2
probability density
0.0
0.1
0.2
0.3
0.4
0.5
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
x


logistic CDF with a = 0, λ = 2
probability density
1.0
0.8
0.6
0.4
0.2
0.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
x

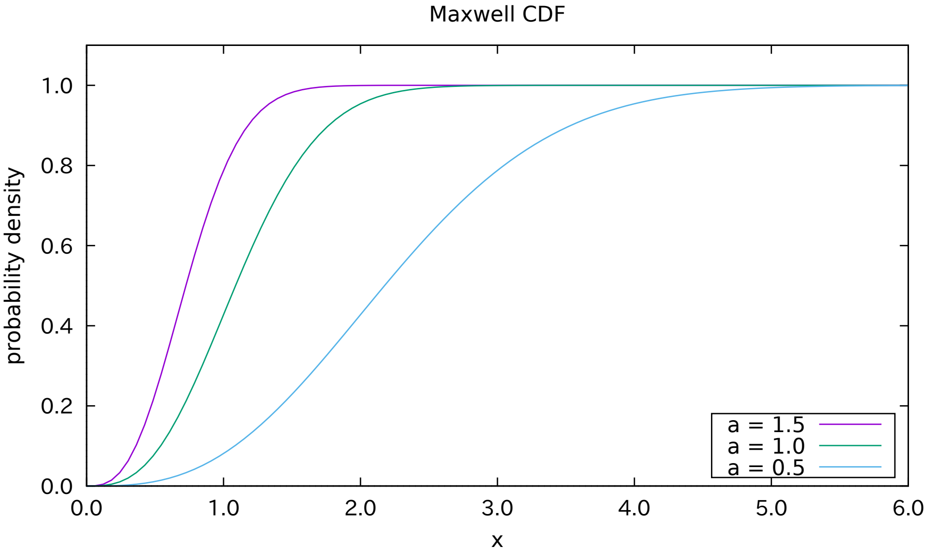

lognormal PDF with μ = 1.0, σ = 0.5
probability density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.00
5.00
10.00
15.00
20.00
x

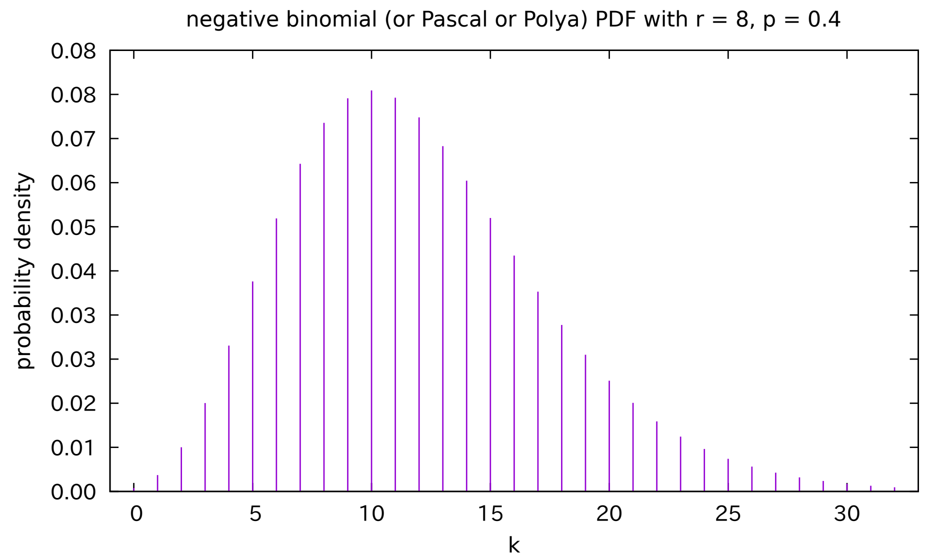


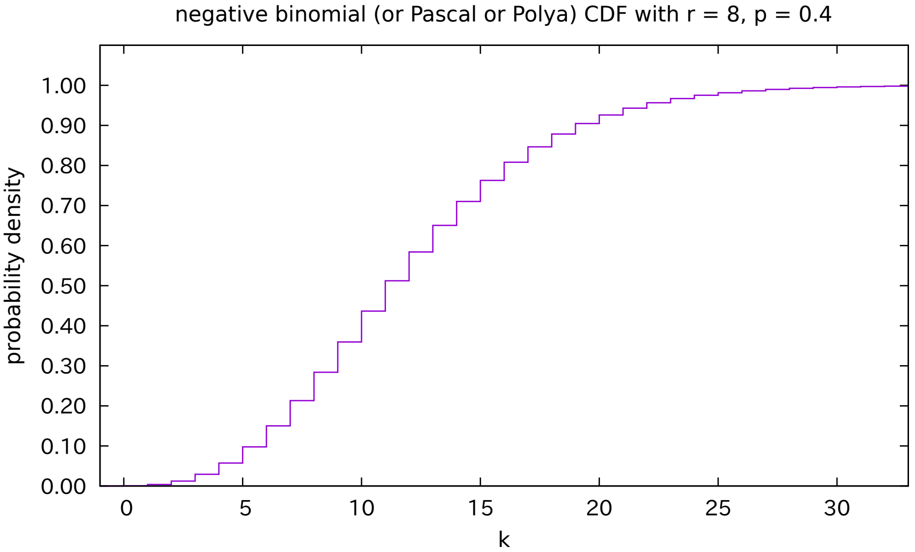
lognormal CDF with μ = 1.0, σ = 0.5
probability density
1.00
0.80
0.60
0.40
0.20
0.00
0.00
5.00
10.00
15.00
20.00
x



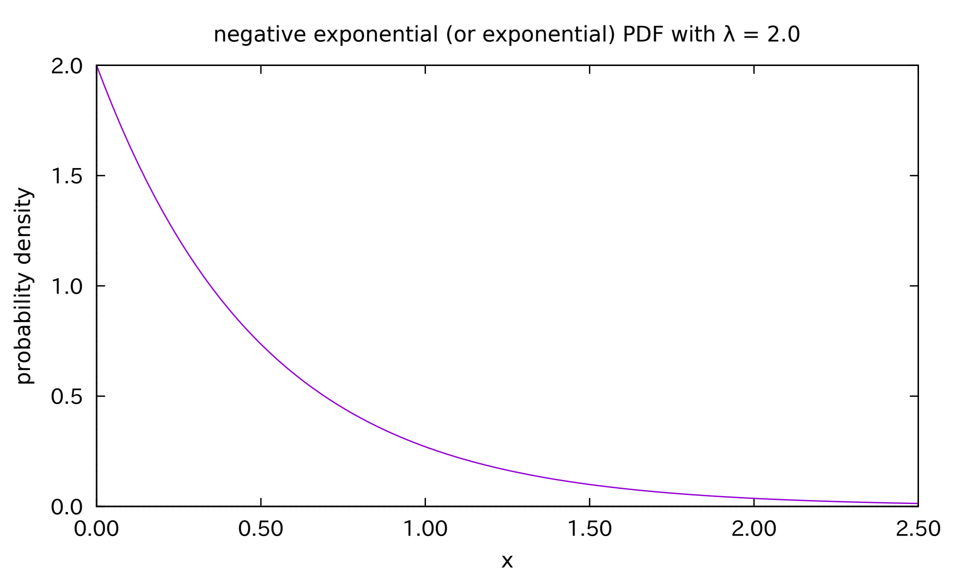
Maxwell PDF
a = 1.5
a = 1.0
a = 0.5
probability density
x
0.0
0.5
1.0
1.5
2.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0

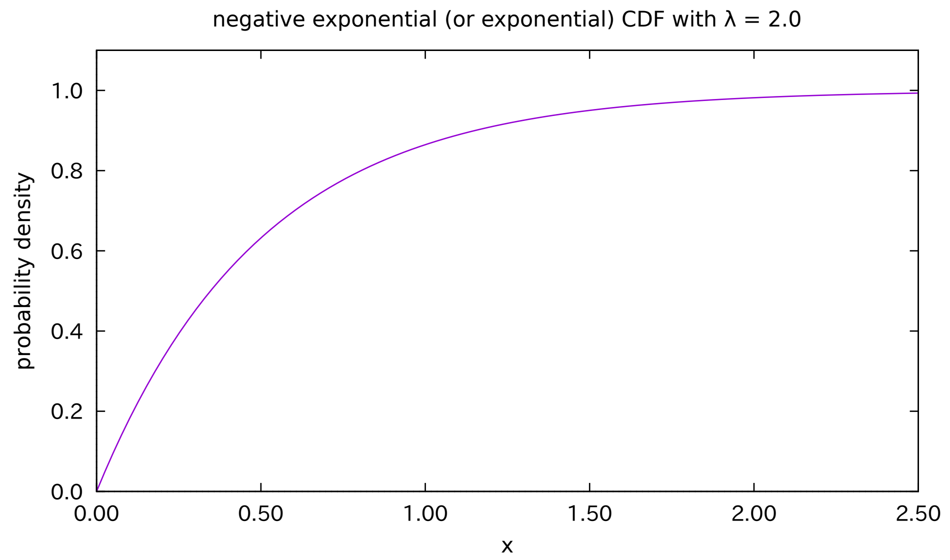

Maxwell CDF
probability density
x
1.0
0.8
0.6
0.4
0.2
0.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
a = 1.5
a = 1.0
a = 0.5

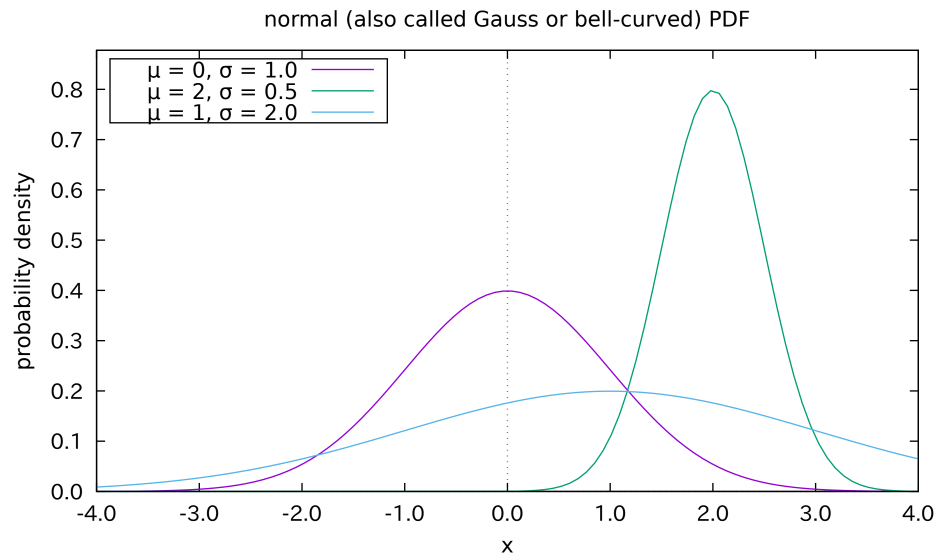

negative binomial (or Pascal or Polya) PDF with r = 8, p = 0.4
probability density
0.00
0.01
0.02
0.03
0.03
0.04
0.05
0.06
0.07
0.08
0.08
0
5
10
15
20
25
30
k



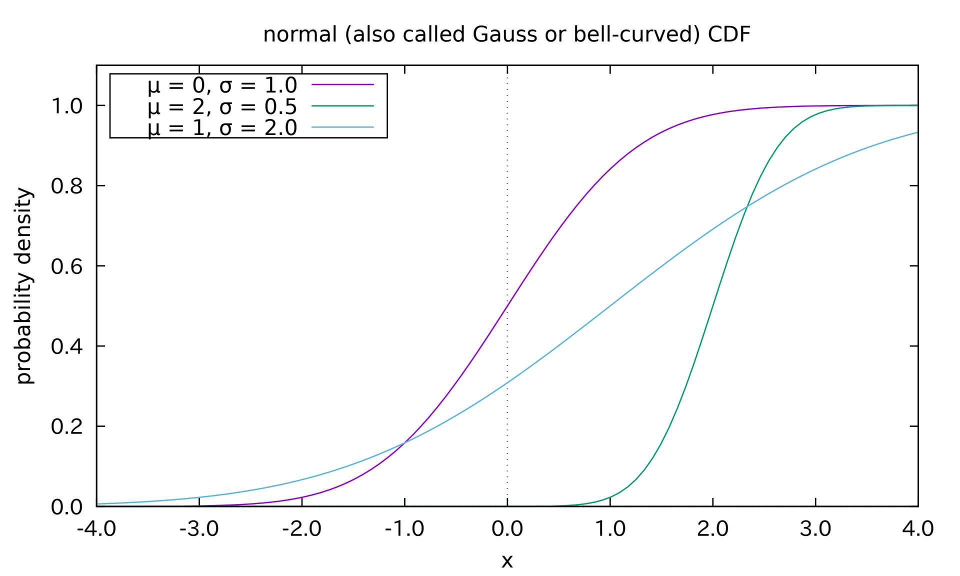
negative binomial (or Pascal or Polya) CDF with r = 8, p = 0.4
probability density
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0
5
10
15
20
25
30
k


negative exponential (or exponential) PDF with λ = 2.0
probability density
2.0
1.5
1.0
0.5
0.0
0.00
0.50
1.00
1.50
2.00
2.50
x

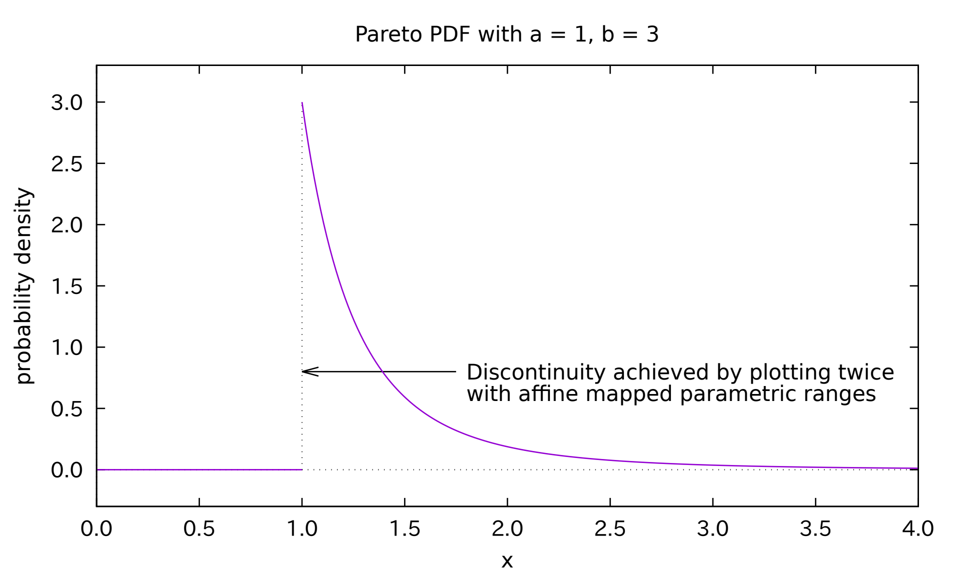

negative exponential (or exponential) CDF with λ = 2.0
probability density
x
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.50
1.00
1.50
2.00
2.50



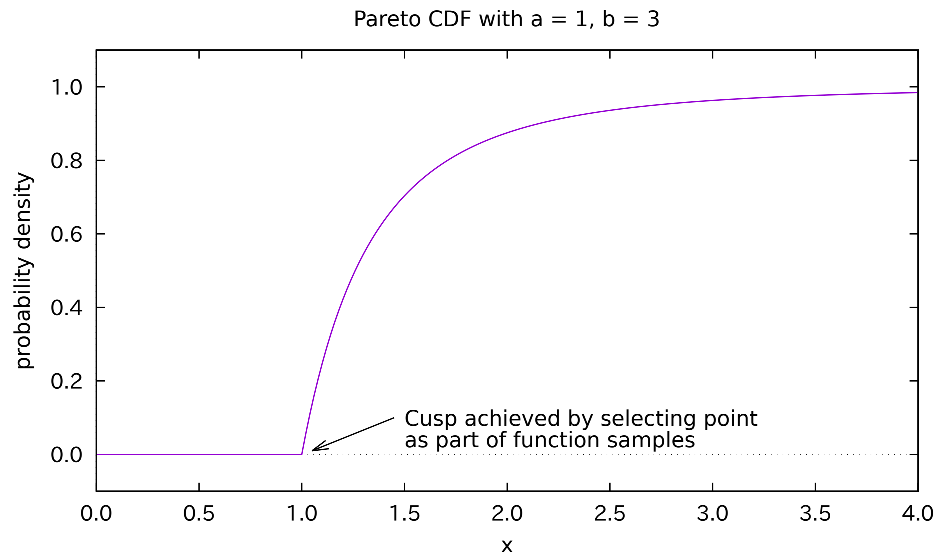
normal (also called Gauss or bell-curved) PDF
μ = 0, σ = 1.0
μ = 2, σ = 0.5
μ = 1, σ = 2.0
probability density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
-4.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0
x



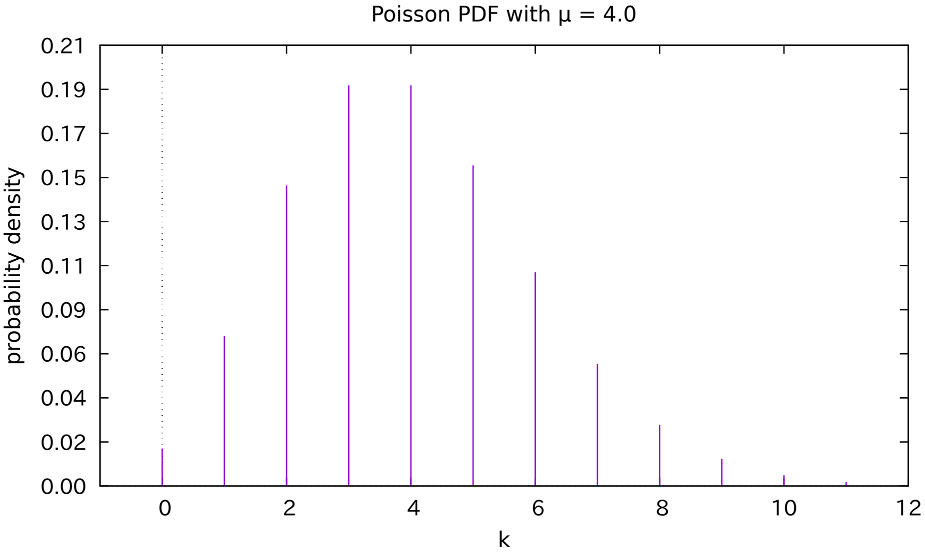
normal (also called Gauss or bell-curved) CDF
μ = 0, σ = 1.0
μ = 2, σ = 0.5
μ = 1, σ = 2.0
probability density
1.0
0.8
0.6
0.4
0.2
0.0
-4.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0
x



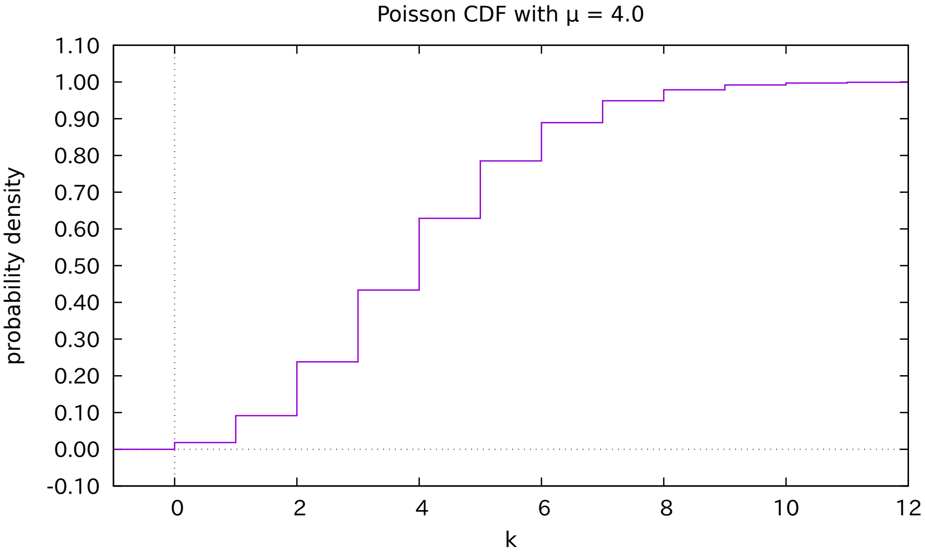
Pareto PDF with a = 1, b = 3
probability density
x
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Discontinuity achieved by plotting twice
with affine mapped parametric ranges



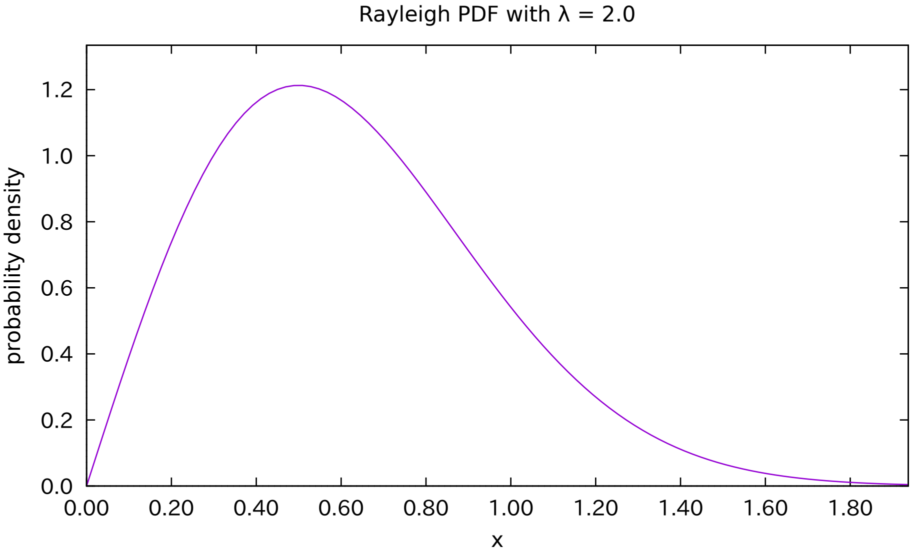
Pareto CDF with a = 1, b = 3
probability density
x
Cusp achieved by selecting point
as part of function samples
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

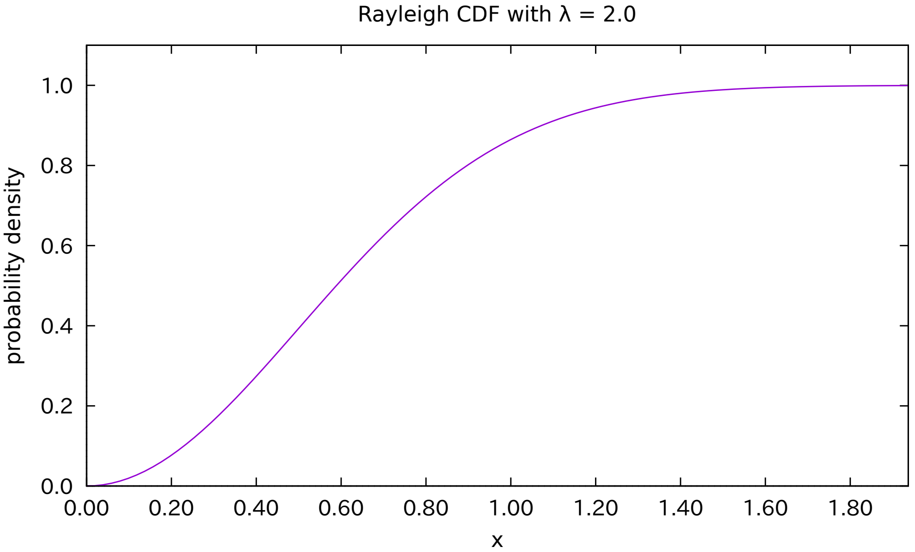


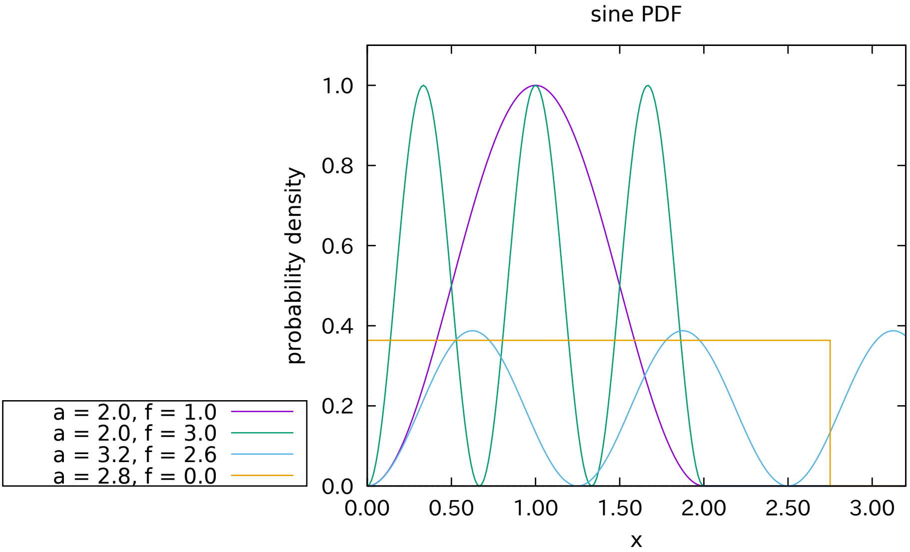
Poisson PDF with μ = 4.0
probability density
0.21
0.19
0.17
0.15
0.13
0.11
0.09
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
k

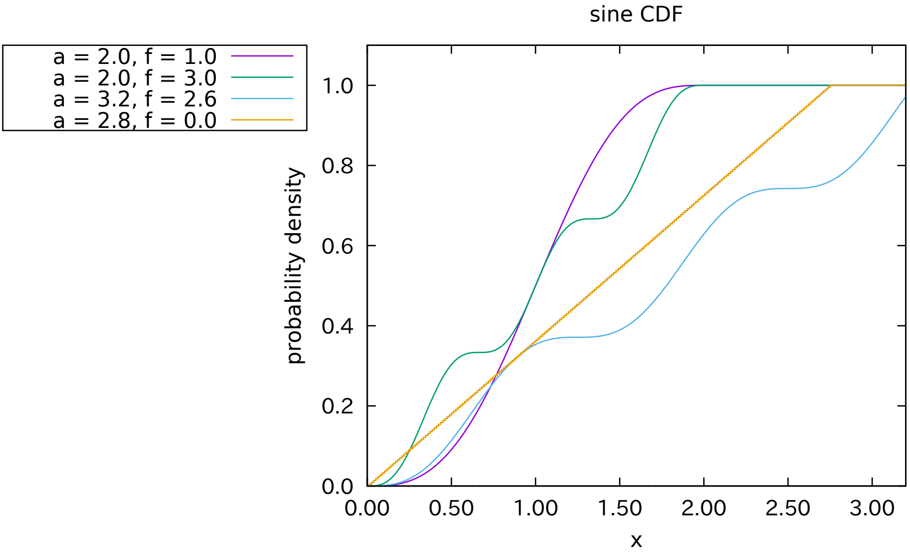


Poisson CDF with μ = 4.0
probability density
k
1.10
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
-0.10
0
2
4
6
8
10
12

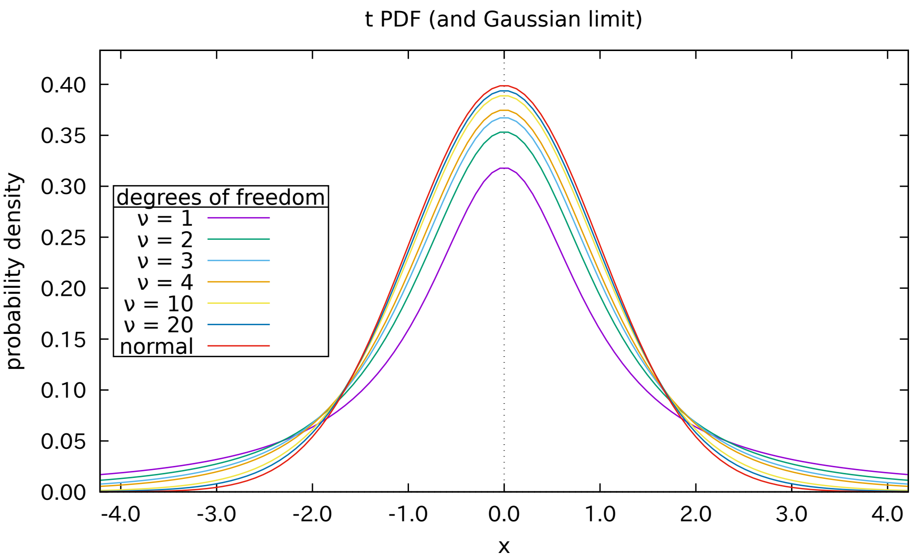

Rayleigh PDF with λ = 2.0
probability density
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80
x


Rayleigh CDF with λ = 2.0
probability density
x
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.20
0.40
0.60
0.80
1.00
1.20
1.40
1.60
1.80

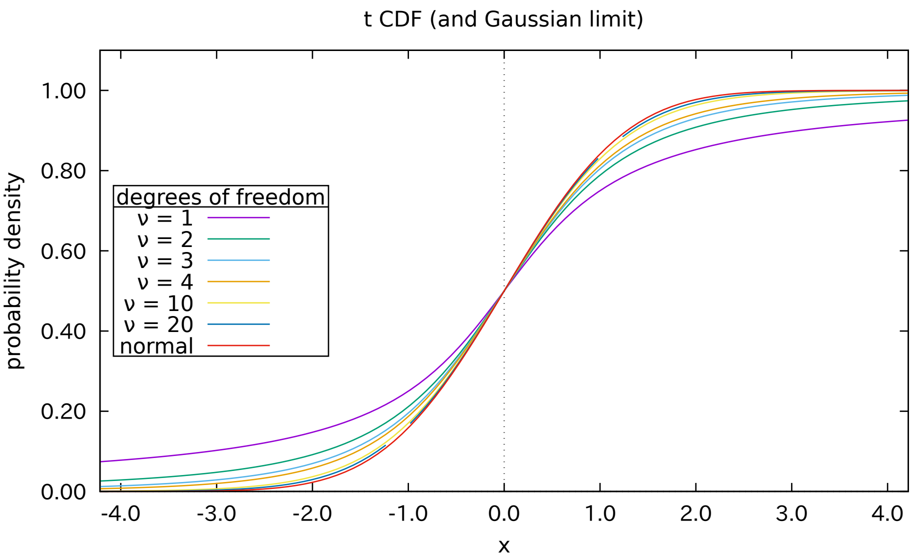

sine PDF
probability density
x
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.50
1.00
1.50
2.00
2.50
3.00
a = 2.0, f = 1.0
a = 2.0, f = 3.0
a = 3.2, f = 2.6
a = 2.8, f = 0.0

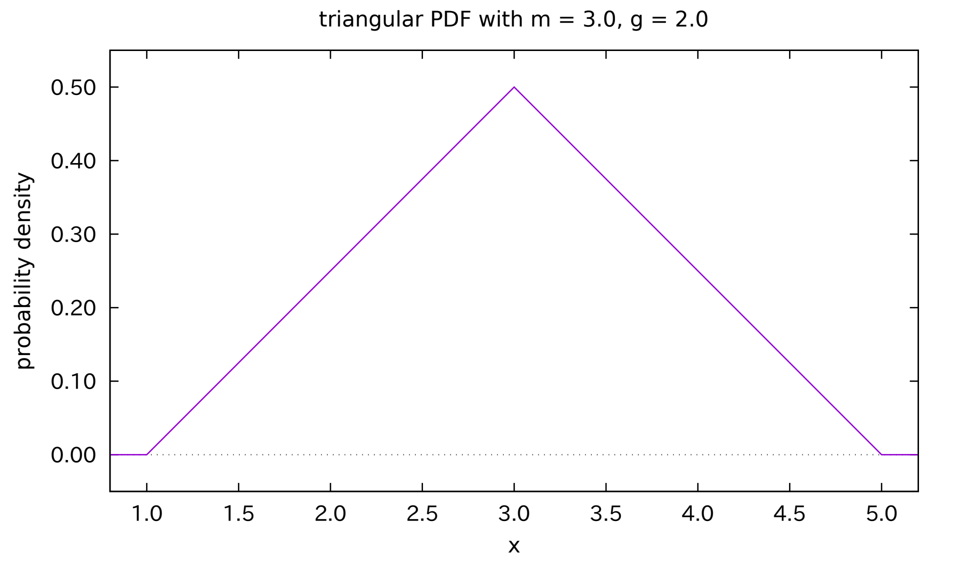


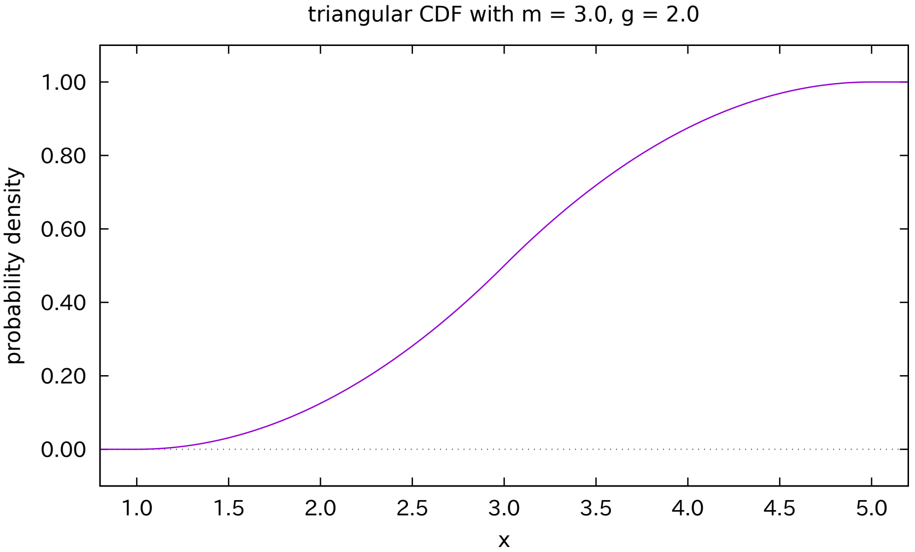
sine CDF
a = 2.0, f = 1.0
a = 2.0, f = 3.0
a = 3.2, f = 2.6
a = 2.8, f = 0.0
probability density
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.50
1.00
1.50
2.00
2.50
3.00
x

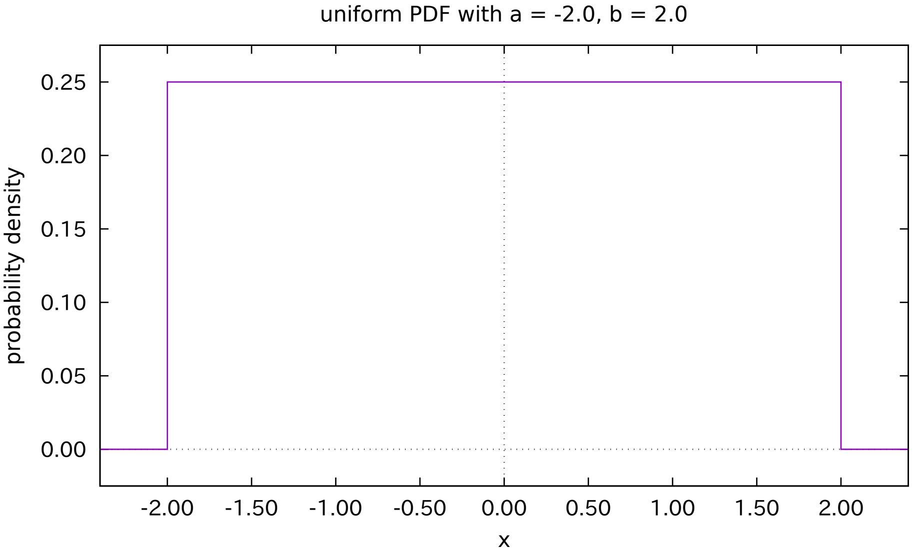


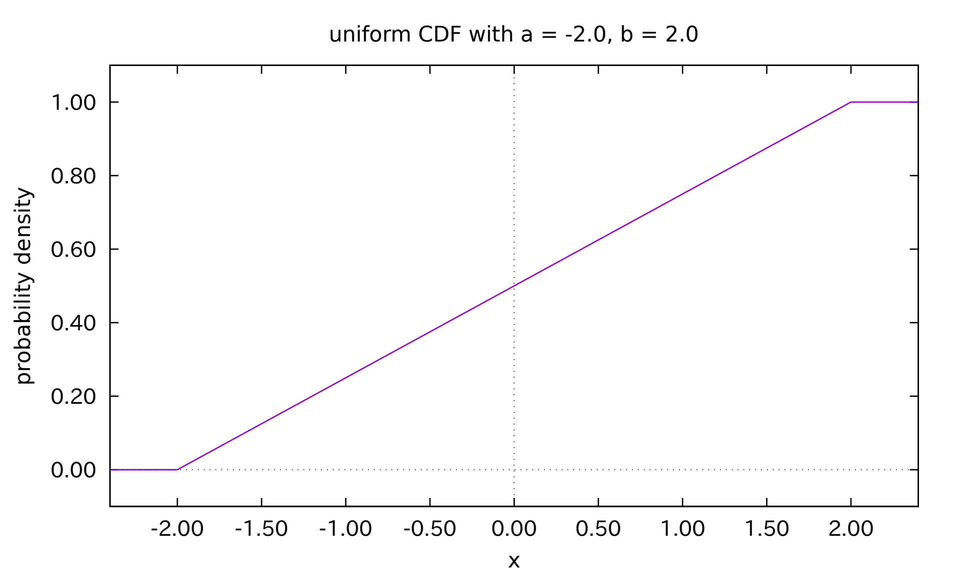
t PDF (and Gaussian limit)
probability density
x
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
-4.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0
degrees of freedom
ν = 1
ν = 2
ν = 3
ν = 4
ν = 10
ν = 20
normal

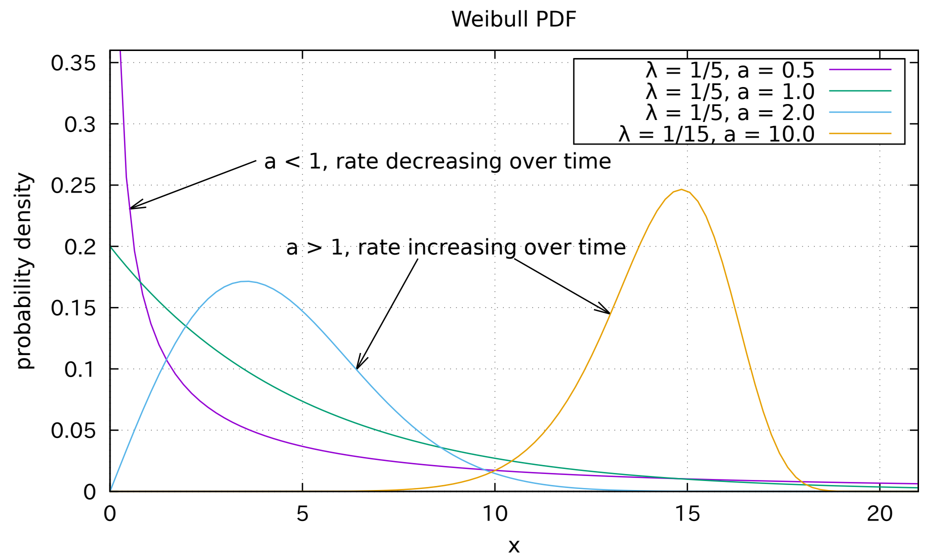

t CDF (and Gaussian limit)
probability density
x
degrees of freedom
ν = 1
ν = 2
ν = 3
ν = 4
ν = 10
ν = 20
normal
1.00
0.80
0.60
0.40
0.20
0.00
-4.0
-3.0
-2.0
-1.0
0.0
1.0
2.0
3.0
4.0

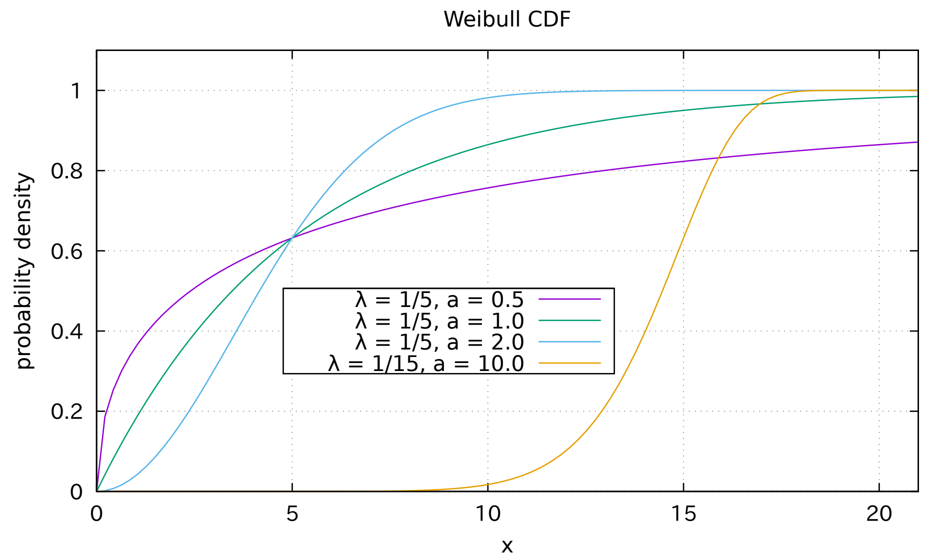

triangular PDF with m = 3.0, g = 2.0
probability density
0.50
0.40
0.30
0.20
0.10
0.00
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
x

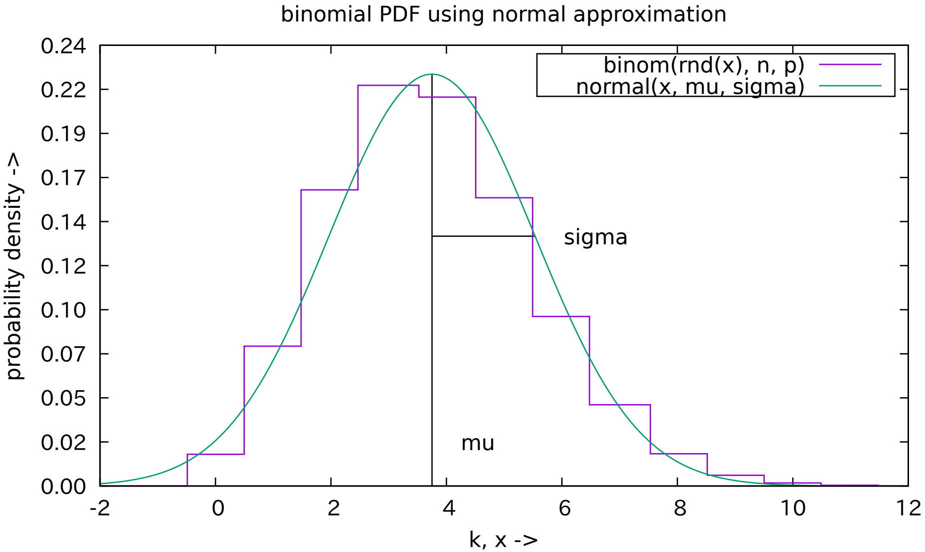

triangular CDF with m = 3.0, g = 2.0
probability density
1.00
0.80
0.60
0.40
0.20
0.00
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
x

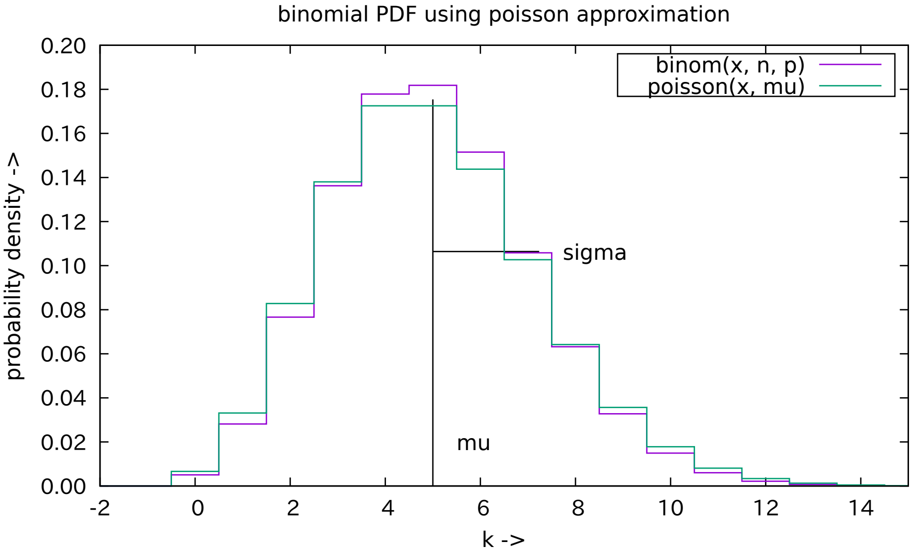

uniform PDF with a = -2.0, b = 2.0
probability density
0.25
0.20
0.15
0.10
0.05
0.00
-2.00
-1.50
-1.00
-0.50
0.00
0.50
1.00
1.50
2.00
x

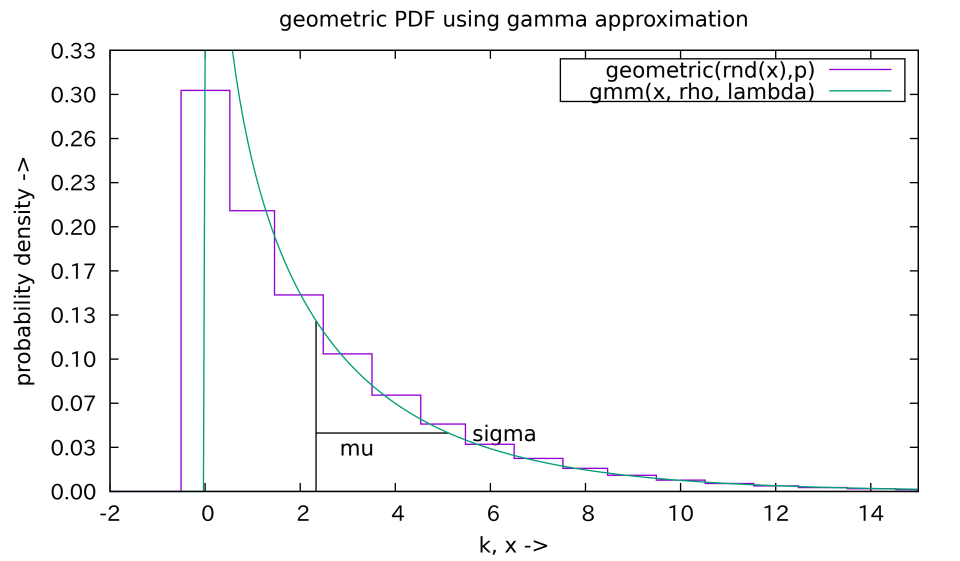

uniform CDF with a = -2.0, b = 2.0
probability density
1.00
0.80
0.60
0.40
0.20
0.00
-2.00
-1.50
-1.00
-0.50
0.00
0.50
1.00
1.50
2.00
x

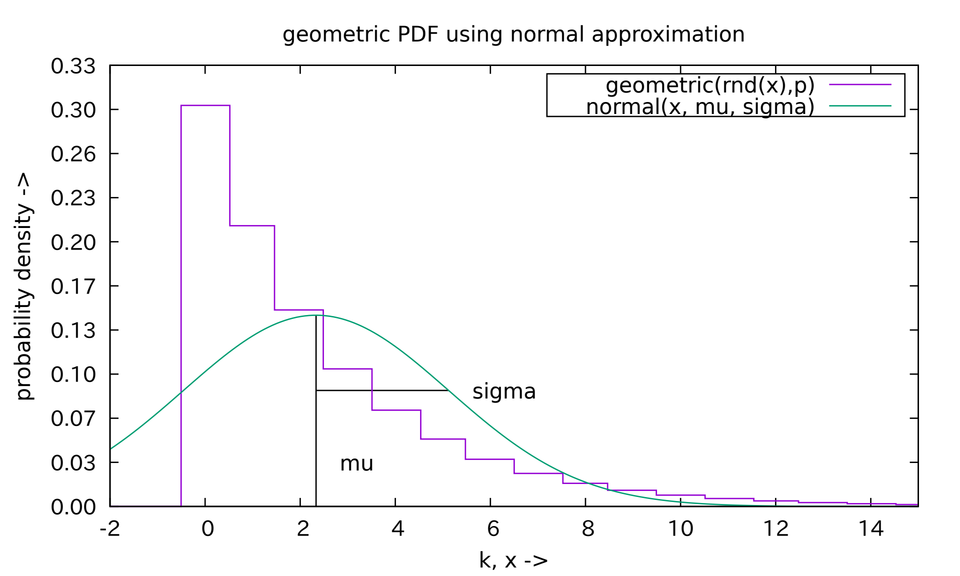

Weibull PDF
λ = 1/5, a = 0.5
λ = 1/5, a = 1.0
λ = 1/5, a = 2.0
λ = 1/15, a = 10.0
a < 1, rate decreasing over time
a > 1, rate increasing over time
probability density
x
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0
5
10
15
20

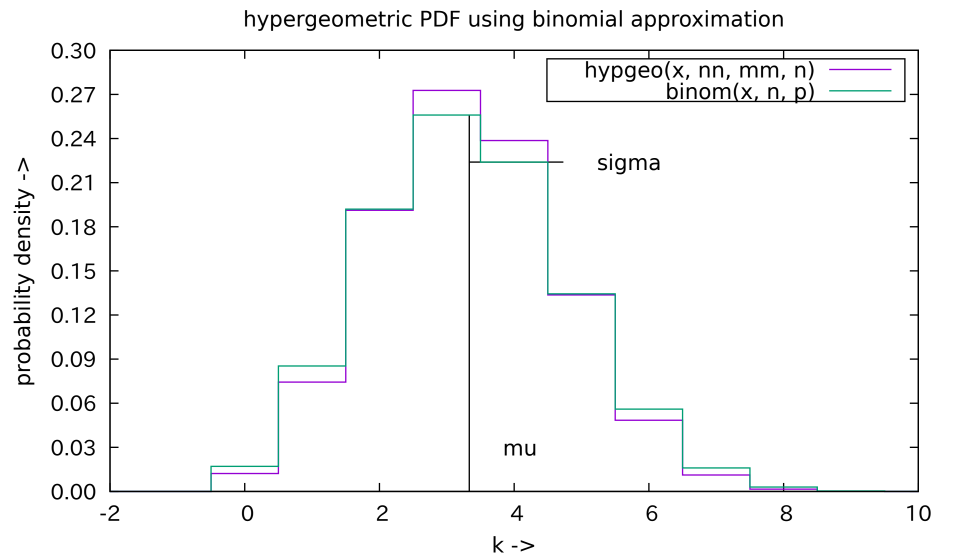

Weibull CDF
probability density
x
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20
λ = 1/5, a = 0.5
λ = 1/5, a = 1.0
λ = 1/5, a = 2.0
λ = 1/15, a = 10.0


binomial PDF using normal approximation
binom(rnd(x), n, p)
normal(x, mu, sigma)
probability density ->
k, x ->
sigma
mu
0.24
0.22
0.19
0.17
0.14
0.12
0.10
0.07
0.05
0.02
0.00
-2
0
2
4
6
8
10
12

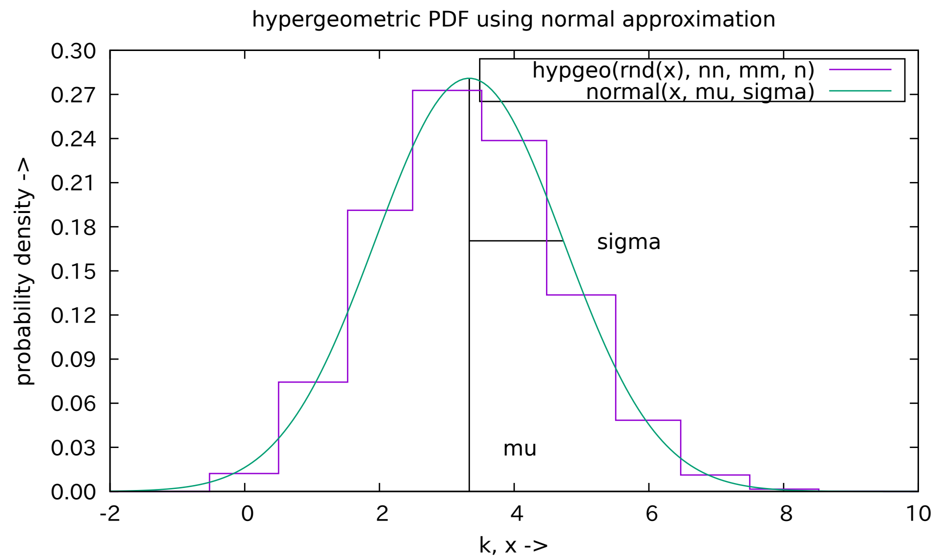

binomial PDF using poisson approximation
binom(x, n, p)
poisson(x, mu)
sigma
mu
probability density ->
k ->
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-2
0
2
4
6
8
10
12
14

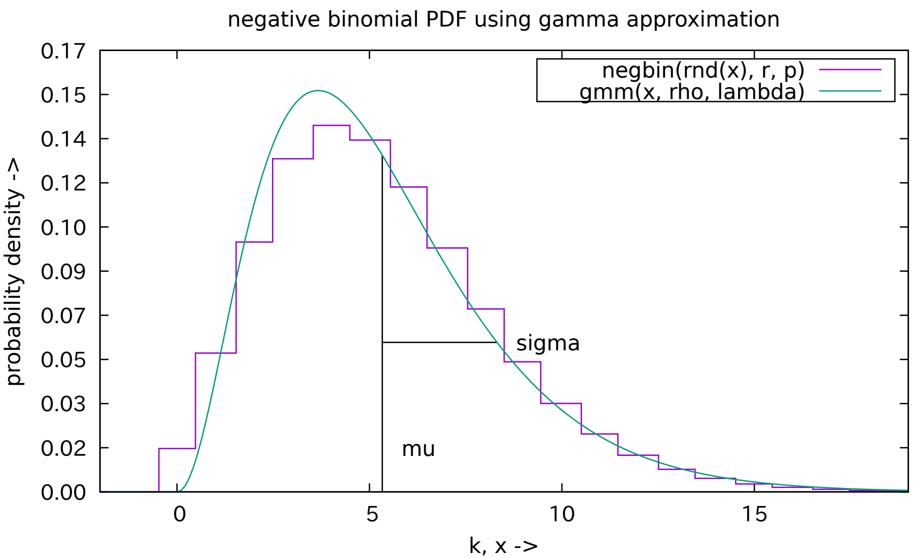

geometric PDF using gamma approximation
geometric(rnd(x),p)
gmm(x, rho, lambda)
probability density ->
k, x ->
sigma
mu
0.33
0.30
0.26
0.23
0.20
0.17
0.13
0.10
0.07
0.03
0.00
-2
0
2
4
6
8
10
12
14

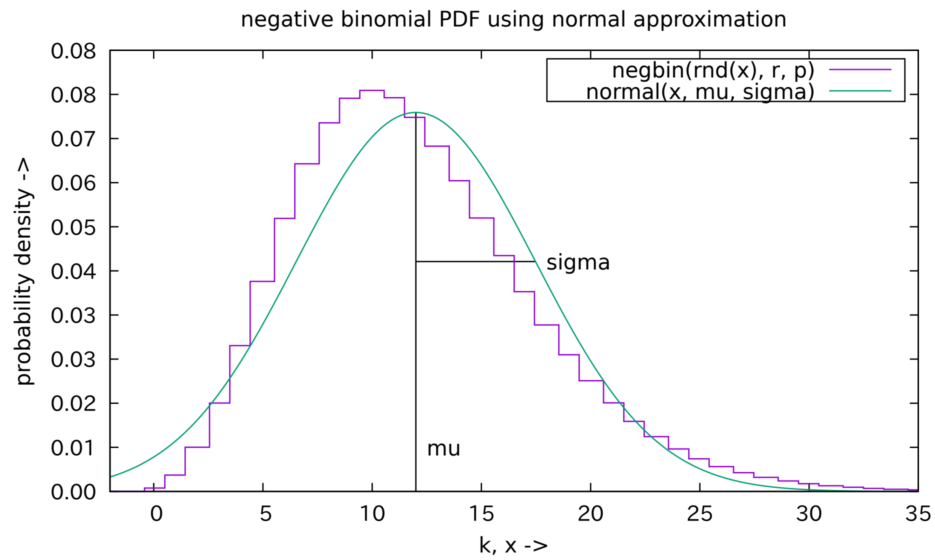

geometric PDF using normal approximation
geometric(rnd(x),p)
normal(x, mu, sigma)
sigma
mu
probability density ->
k, x ->
0.33
0.30
0.26
0.23
0.20
0.17
0.13
0.10
0.07
0.03
0.00
-2
0
2
4
6
8
10
12
14



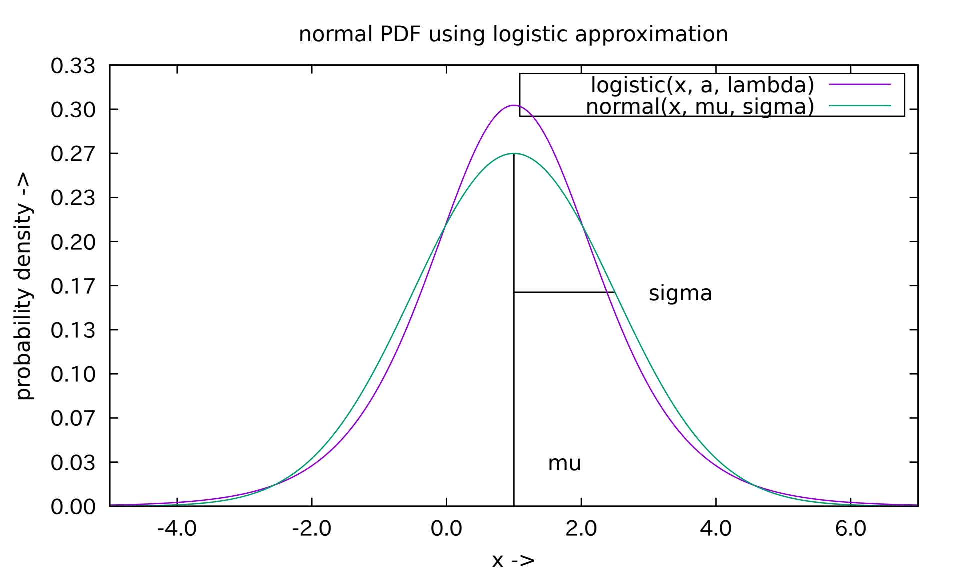
hypergeometric PDF using binomial approximation
hypgeo(x, nn, mm, n)
binom(x, n, p)
sigma
mu
probability density ->
k ->
0.30
0.27
0.24
0.21
0.18
0.15
0.12
0.09
0.06
0.03
0.00
-2
0
2
4
6
8
10

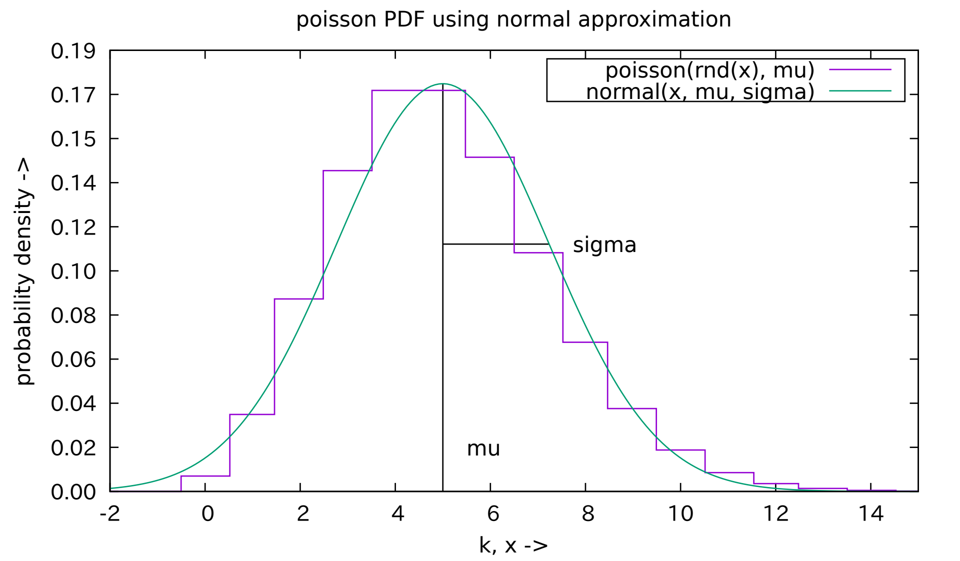

hypergeometric PDF using normal approximation
hypgeo(rnd(x), nn, mm, n)
normal(x, mu, sigma)
sigma
mu
probability density ->
k, x ->
0.30
0.27
0.24
0.21
0.18
0.15
0.12
0.09
0.06
0.03
0.00
-2
0
2
4
6
8
10

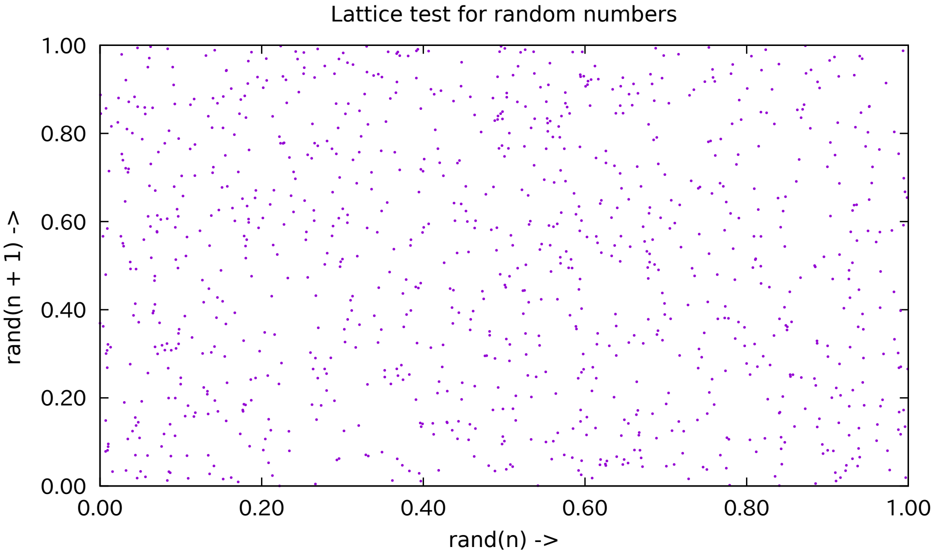

negative binomial PDF using gamma approximation
negbin(rnd(x), r, p)
gmm(x, rho, lambda)
sigma
mu
probability density ->
k, x ->
0.17
0.15
0.14
0.12
0.10
0.09
0.07
0.05
0.03
0.02
0.00
0
5
10
15

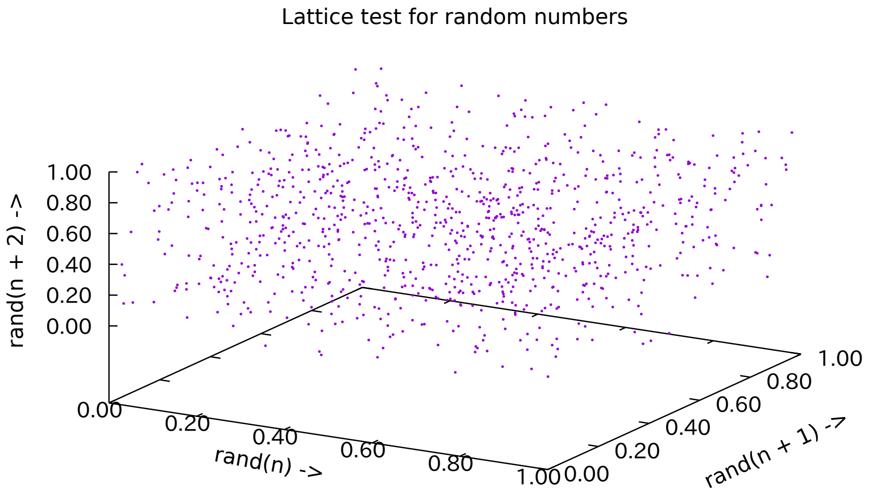


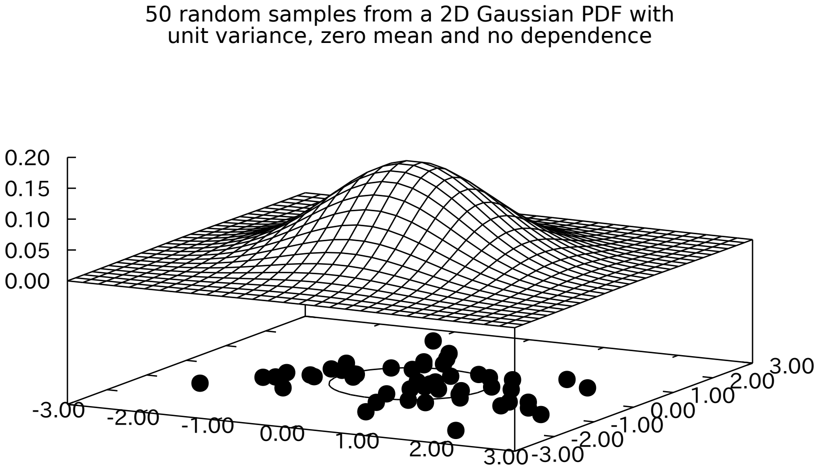
negative binomial PDF using normal approximation
negbin(rnd(x), r, p)
normal(x, mu, sigma)
sigma
mu
probability density ->
k, x ->
0.00
0.01
0.02
0.03
0.03
0.04
0.05
0.06
0.07
0.08
0.08
0
5
10
15
20
25
30
35

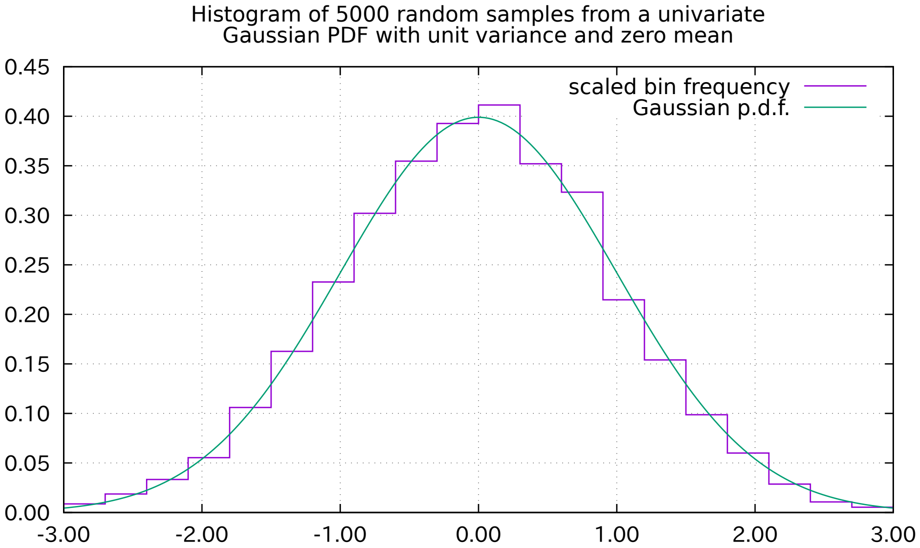

normal PDF using logistic approximation
logistic(x, a, lambda)
normal(x, mu, sigma)
probability density ->
x ->
sigma
mu
0.33
0.30
0.27
0.23
0.20
0.17
0.13
0.10
0.07
0.03
0.00
-4.0
-2.0
0.0
2.0
4.0
6.0

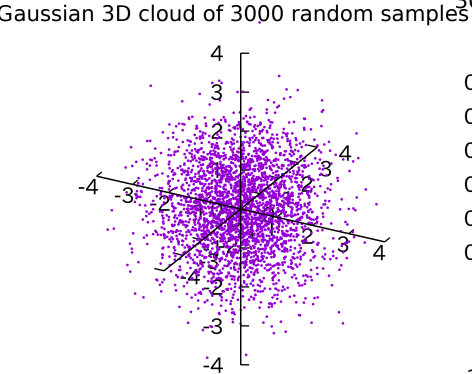

poisson PDF using normal approximation
poisson(rnd(x), mu)
normal(x, mu, sigma)
sigma
mu
probability density ->
k, x ->
0.19
0.17
0.15
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-2
0
2
4
6
8
10
12
14

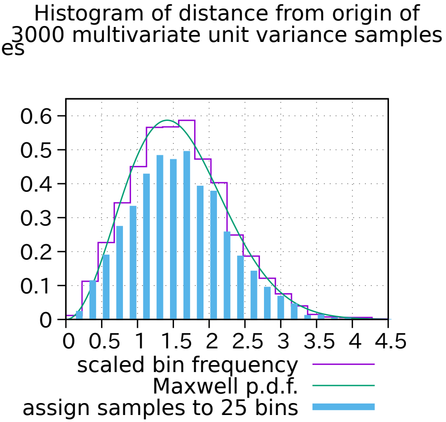

Lattice test for random numbers
1.00
0.80
0.60
0.40
0.20
0.00
rand(n + 1) ->
0.00
0.20
0.40
0.60
0.80
1.00
rand(n) ->


Lattice test for random numbers
rand(n + 2) ->
1.00
0.80
0.60
0.40
0.20
0.00
0.00
0.20
0.40
0.60
0.80
1.00
rand(n) ->
0.00
0.20
0.40
0.60
0.80
1.00
rand(n + 1) ->

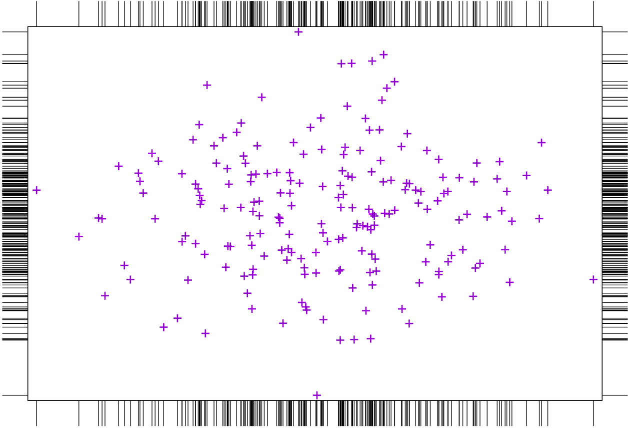

50 random samples from a 2D Gaussian PDF with
unit variance, zero mean and no dependence
0.20
0.15
0.10
0.05
0.00
-3.00
-2.00
-1.00
0.00
1.00
2.00
3.00
-3.00
-2.00
-1.00
0.00
1.00
2.00
3.00

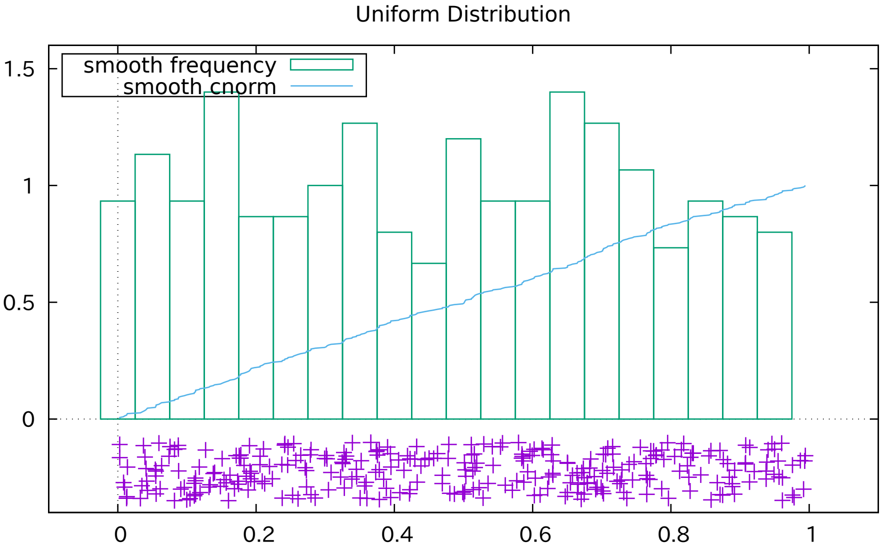

Histogram of 5000 random samples from a univariate
Gaussian PDF with unit variance and zero mean
scaled bin frequency
Gaussian p.d.f.
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
-3.00
-2.00
-1.00
0.00
1.00
2.00
3.00


Gaussian 3D cloud of 3000 random samples
4
3
2
-2
-3
-4

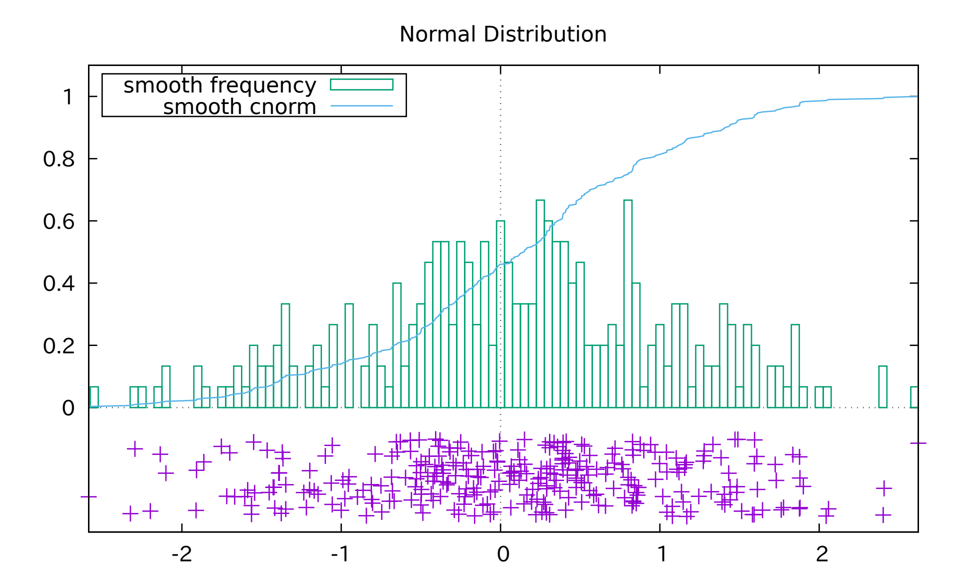

Histogram of distance from origin of
3000 multivariate unit variance samples
0.6
0.5
0.4
0.3
0.2
0.1
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5
scaled bin frequency
Maxwell p.d.f.
assign samples to 25 bins

Rug Plot

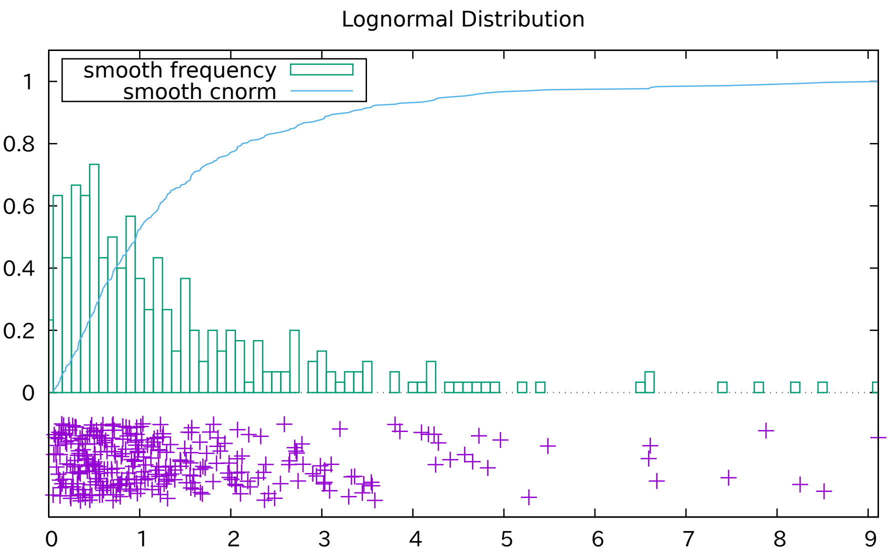


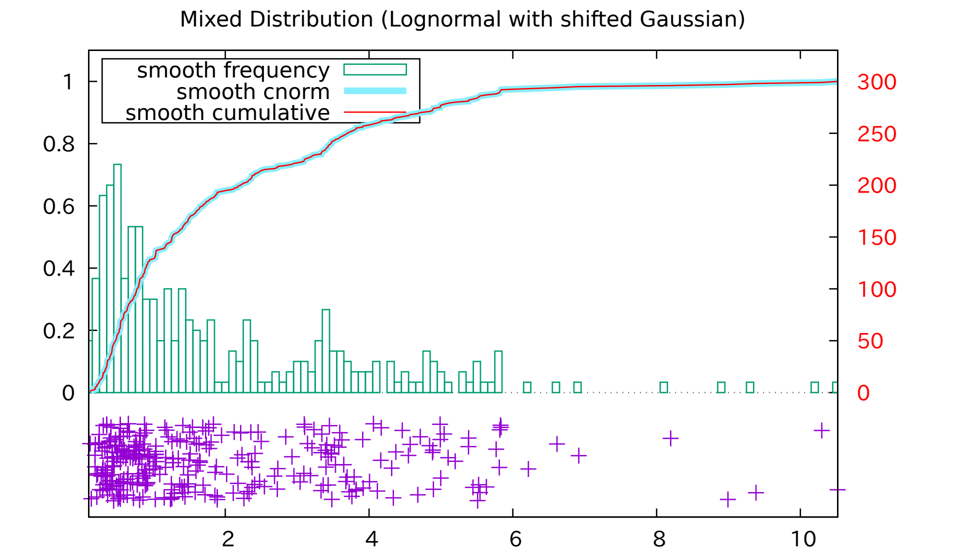


Uniform Distribution
smooth frequency
smooth cnorm
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

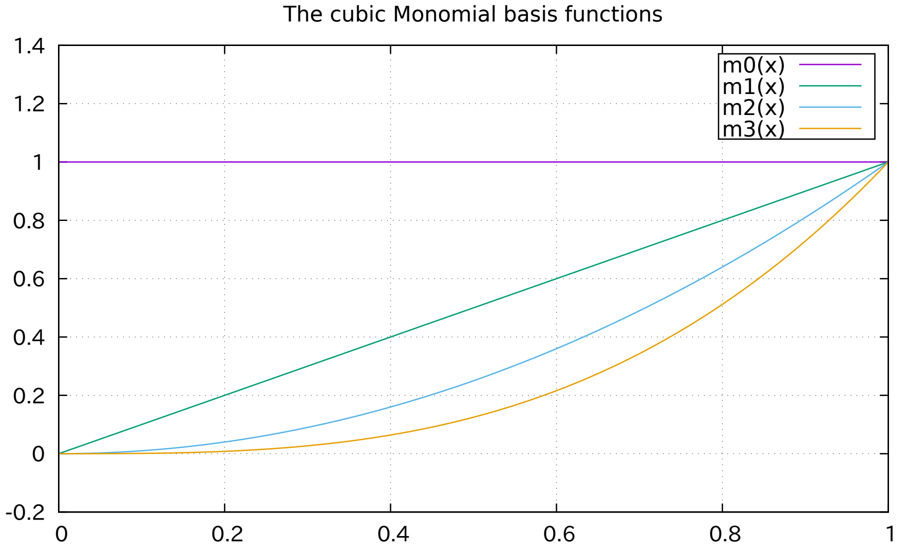


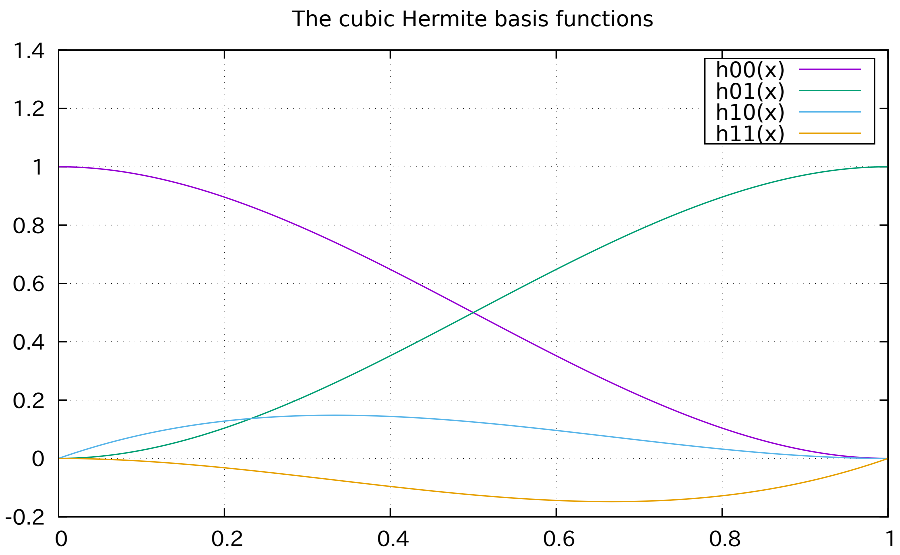
Normal Distribution
smooth frequency
smooth cnorm
1
0.8
0.6
0.4
0.2
0
-2
-1
0
1
2

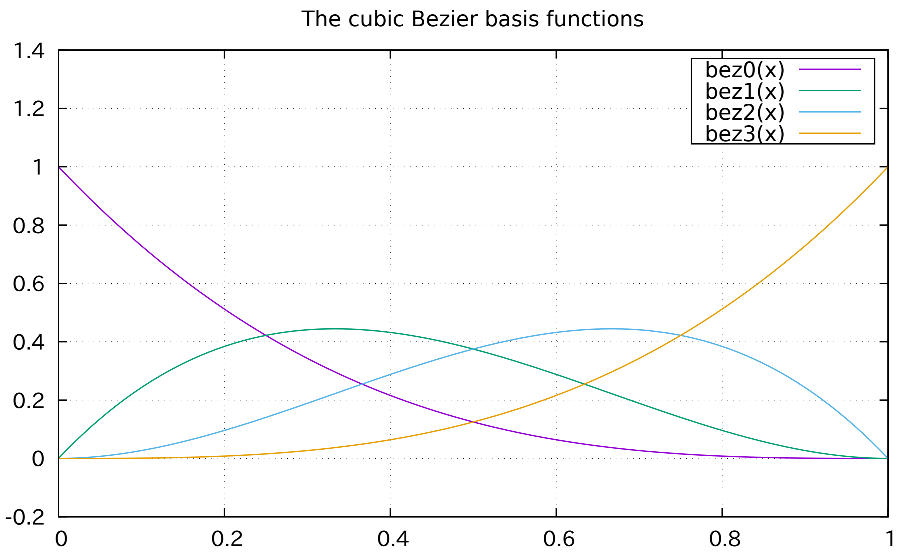

Lognormal Distribution
smooth frequency
smooth cnorm
0
0.2
0.4
0.6
0.8
1
0
1
2
3
4
5
6
7
8
9

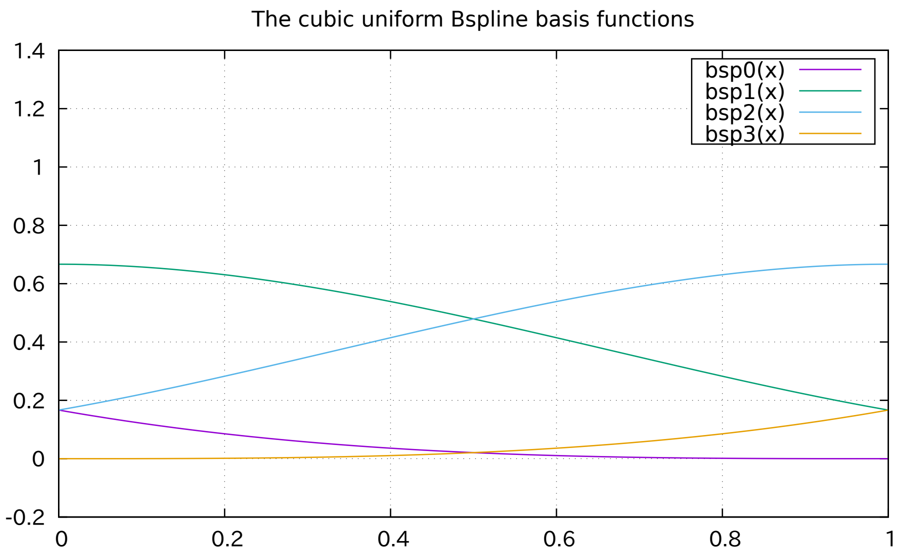


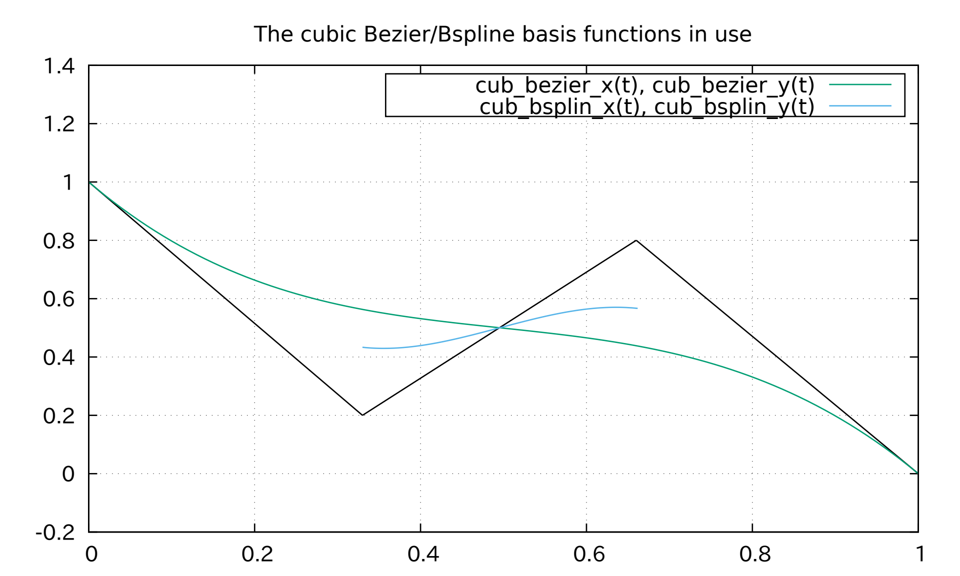
Mixed Distribution (Lognormal with shifted Gaussian)
smooth frequency
smooth cnorm
smooth cumulative
1
0.8
0.6
0.4
0.2
0
300
250
200
150
100
50
0
2
4
6
8
10

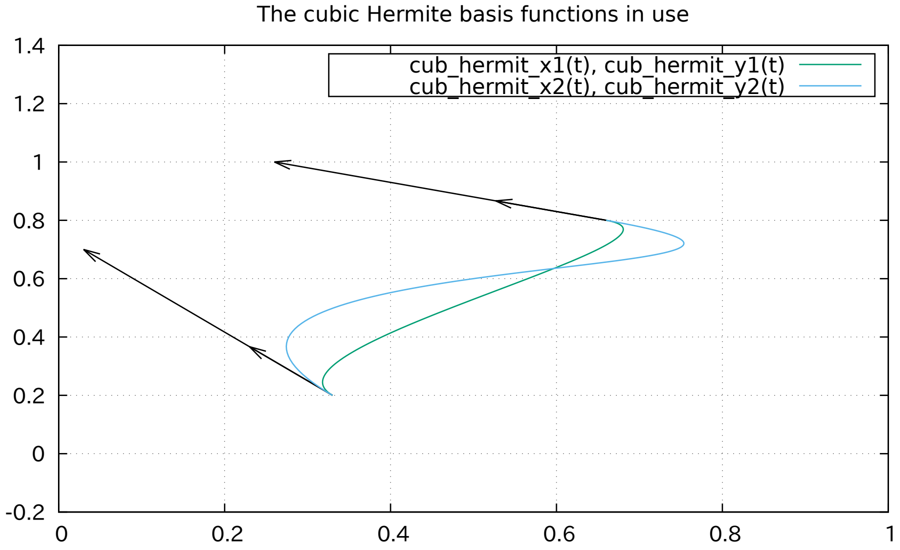

The cubic Monomial basis functions
m0(x)
m1(x)
m2(x)
m3(x)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1


The cubic Hermite basis functions
h00(x)
h01(x)
h10(x)
h11(x)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1

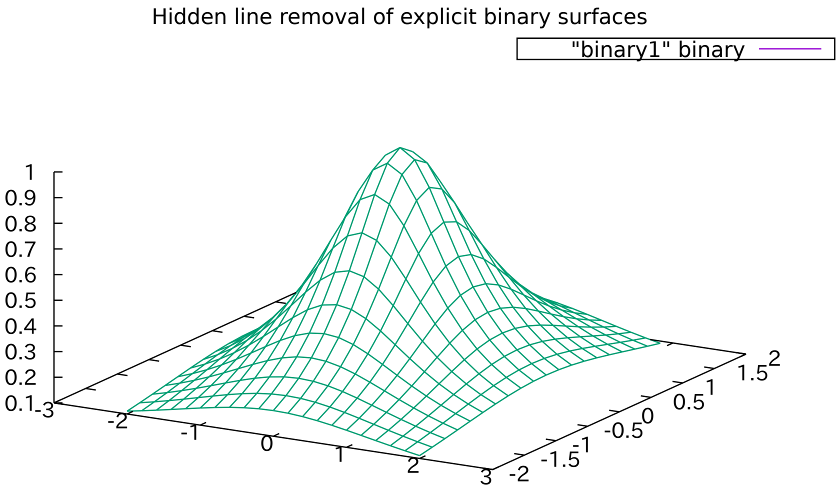

The cubic Bezier basis functions
bez0(x)
bez1(x)
bez2(x)
bez3(x)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1


The cubic uniform Bspline basis functions
bsp0(x)
bsp1(x)
bsp2(x)
bsp3(x)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1


The cubic Bezier/Bspline basis functions in use
cub_bezier_x(t), cub_bezier_y(t)
cub_bsplin_x(t), cub_bsplin_y(t)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1

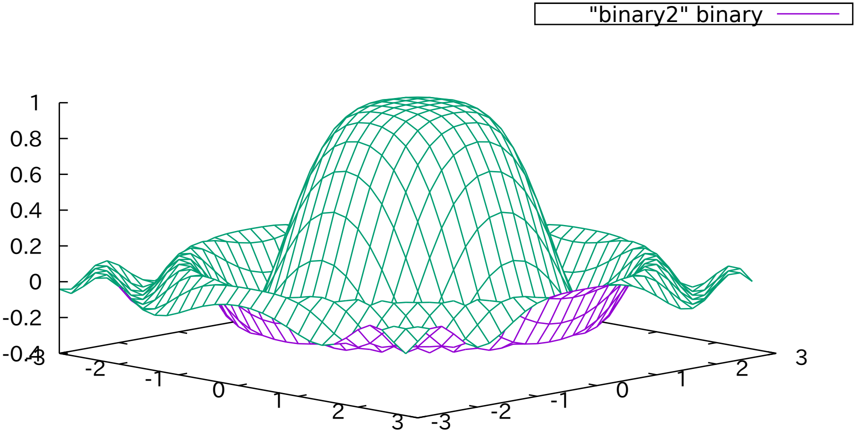


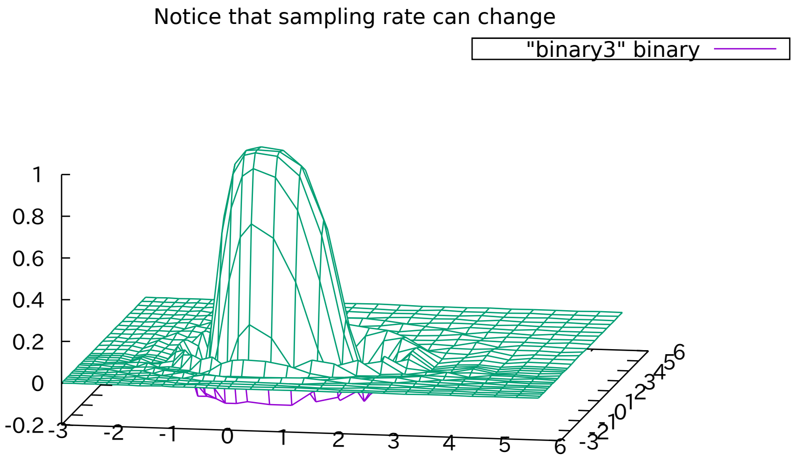
The cubic Hermite basis functions in use
cub_hermit_x1(t), cub_hermit_y1(t)
cub_hermit_x2(t), cub_hermit_y2(t)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1



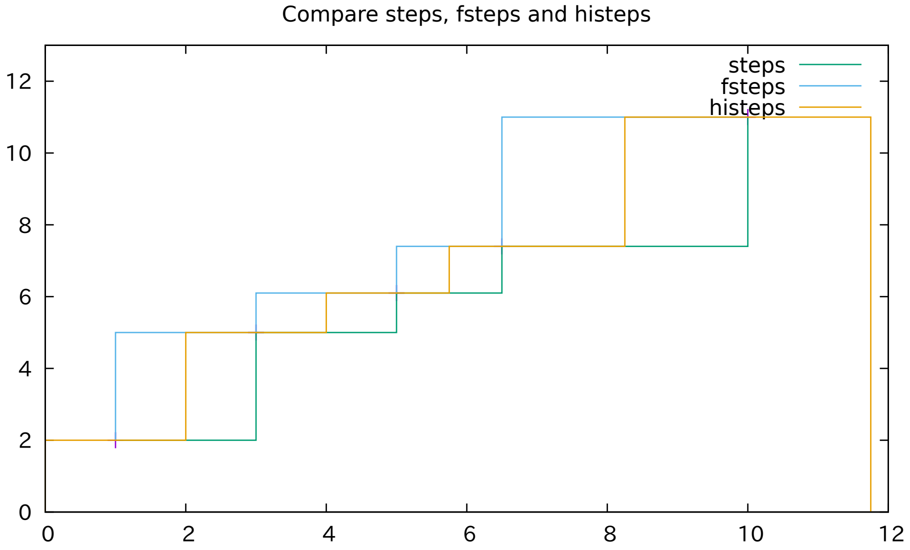
Hidden line removal of explicit binary surfaces
"binary1" binary
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
-3
-2
-1
0
1
2
3
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

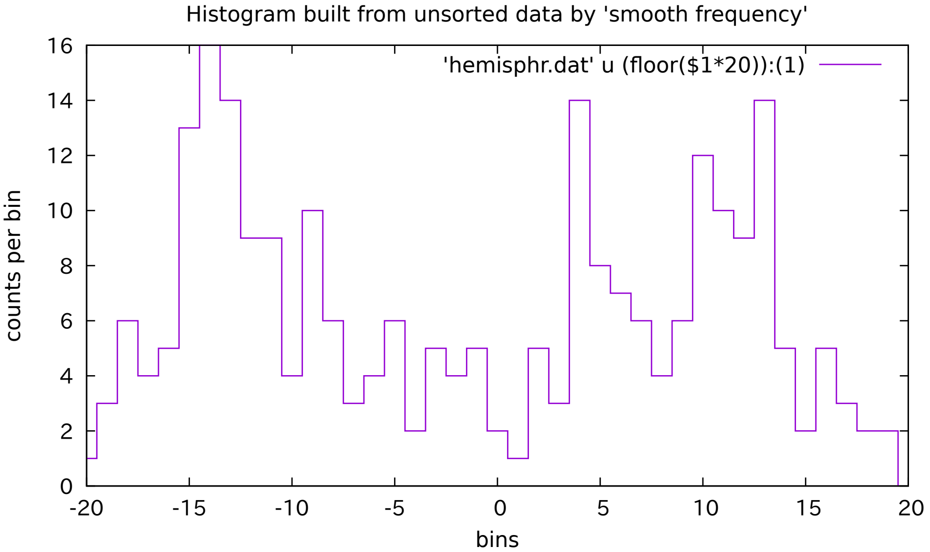
Hidden line removal of explicit binary surfaces

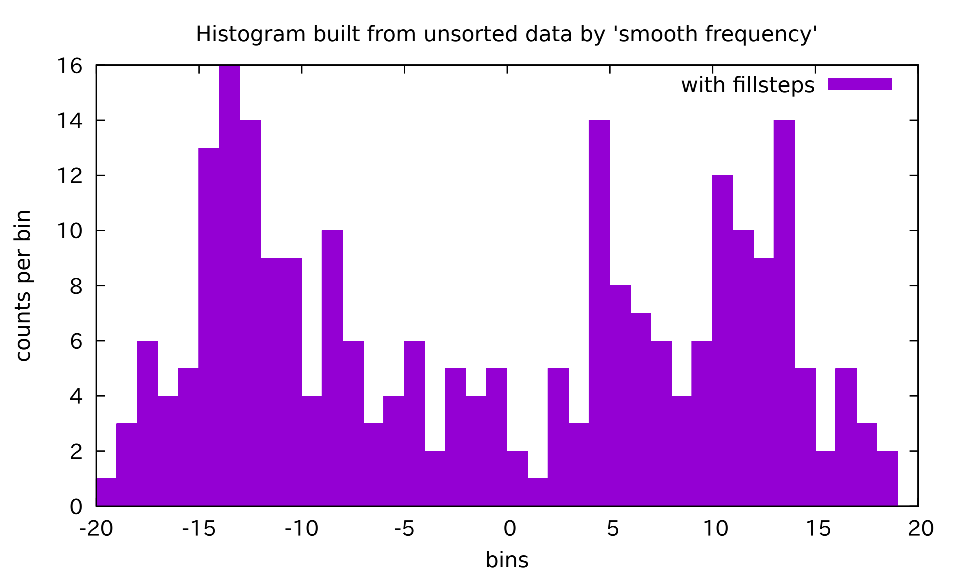


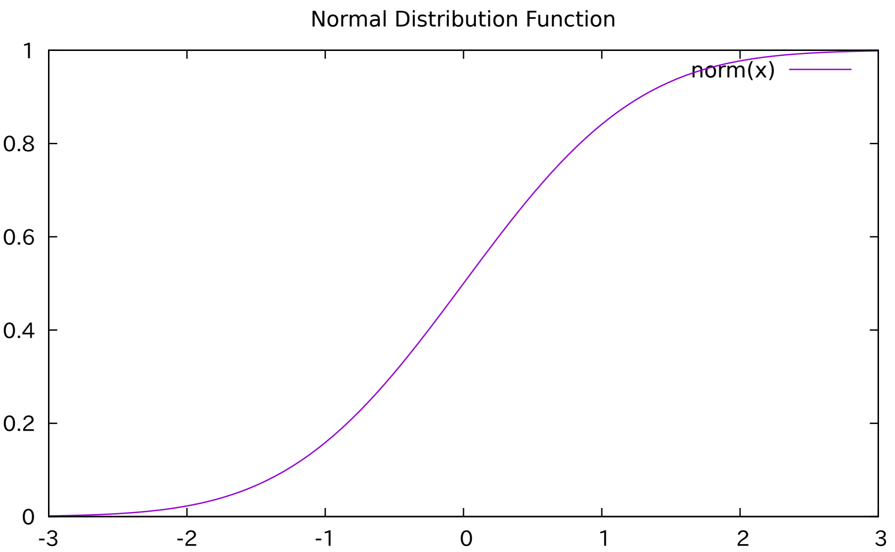


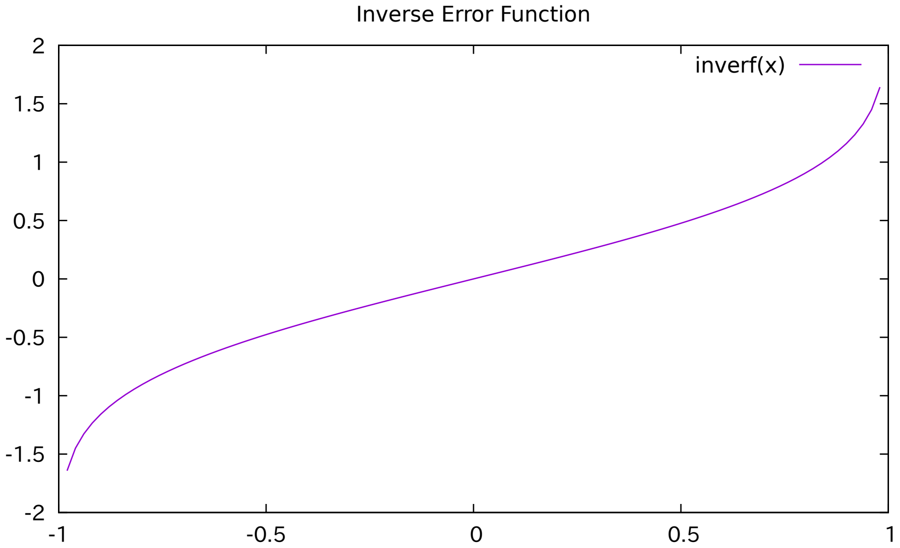
Notice that sampling rate can change
"binary3" binary
1
0.8
0.6
0.4
0.2
0
-0.2
-3
-2
-1
0
1
2
3
4
5
6
-3
-2
-1
0
1
2
3
4
5
6

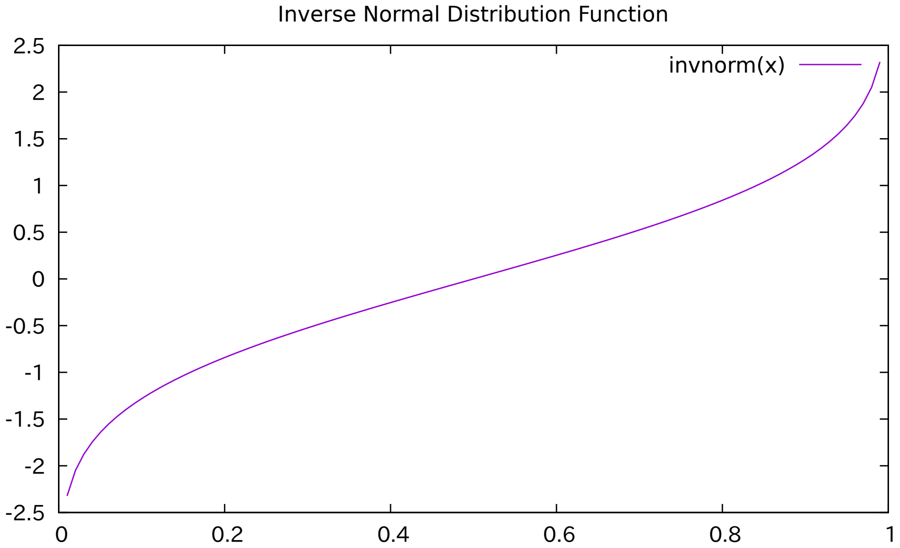

Compare steps, fsteps and histeps
steps
fsteps
histeps
0
2
4
6
8
10
12
0
2
4
6
8
10
12

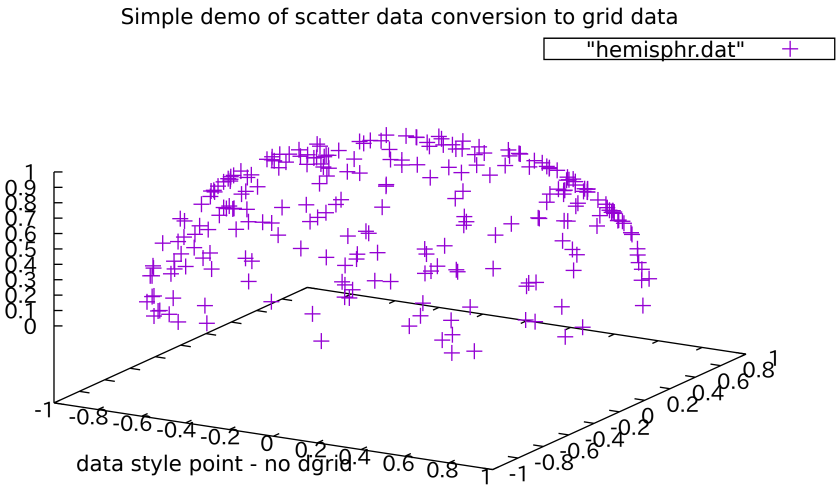

Histogram built from unsorted data by 'smooth frequency'
'hemisphr.dat' u (floor($1*20)):(1)
counts per bin
bins
0
2
4
6
8
10
12
14
16
-20
-15
-10
-5
0
5
10
15
20

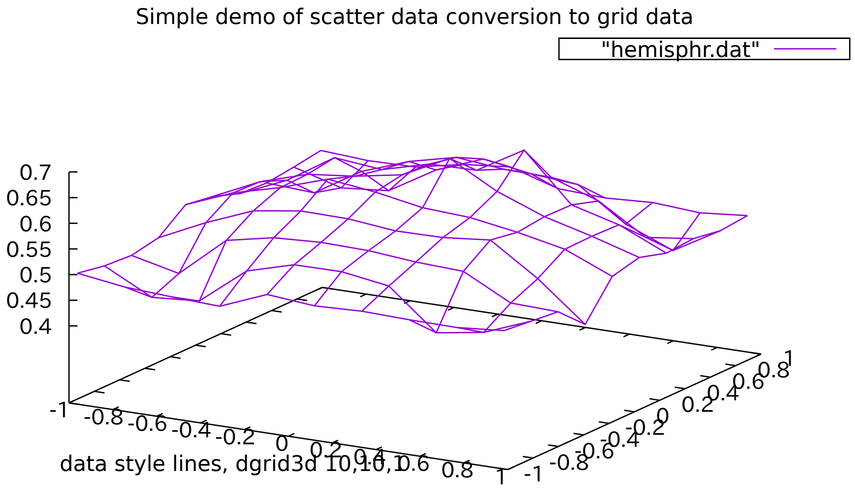

Histogram built from unsorted data by 'smooth frequency'
with fillsteps
counts per bin
bins
0
2
4
6
8
10
12
14
16
-20
-15
-10
-5
0
5
10
15
20


Normal Distribution Function
norm(x)
1
0.8
0.6
0.4
0.2
0
-3
-2
-1
0
1
2
3

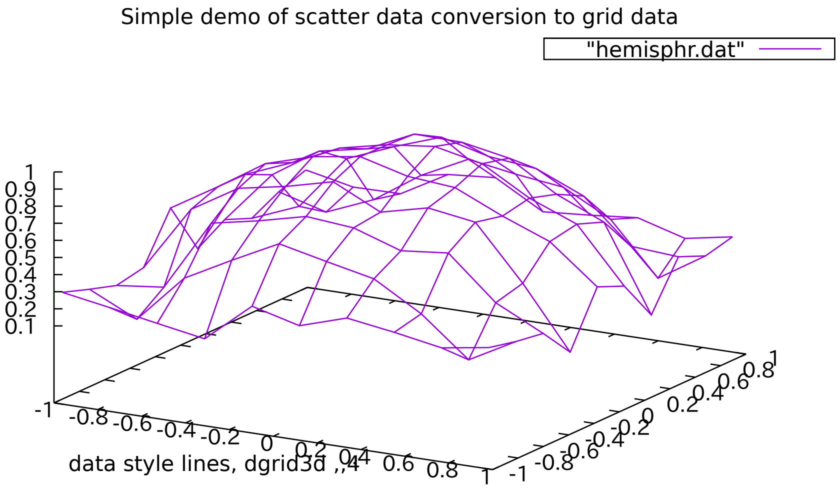

Inverse Error Function
inverf(x)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
-0.5
0
0.5
1

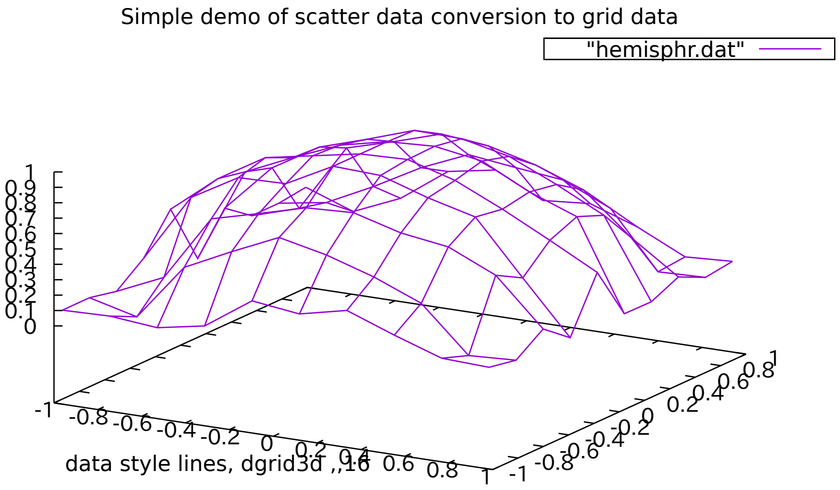

Inverse Normal Distribution Function
invnorm(x)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.2
0.4
0.6
0.8
1


Simple demo of scatter data conversion to grid data
"hemisphr.dat"
data style point - no dgrid
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
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

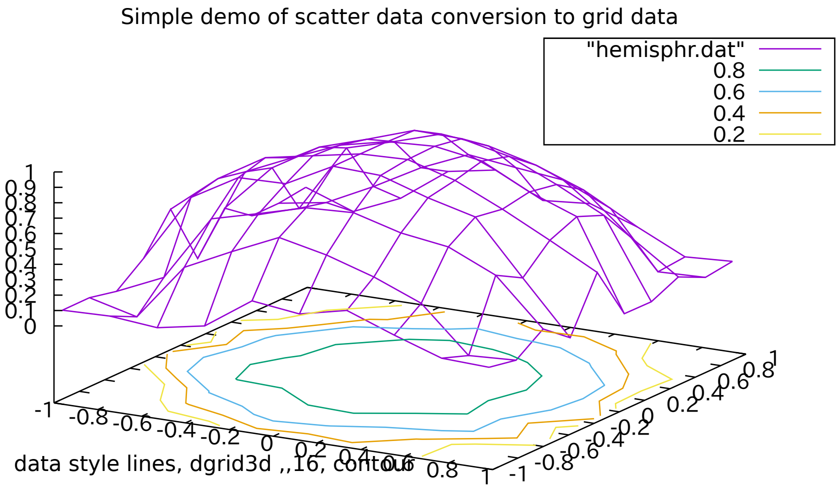


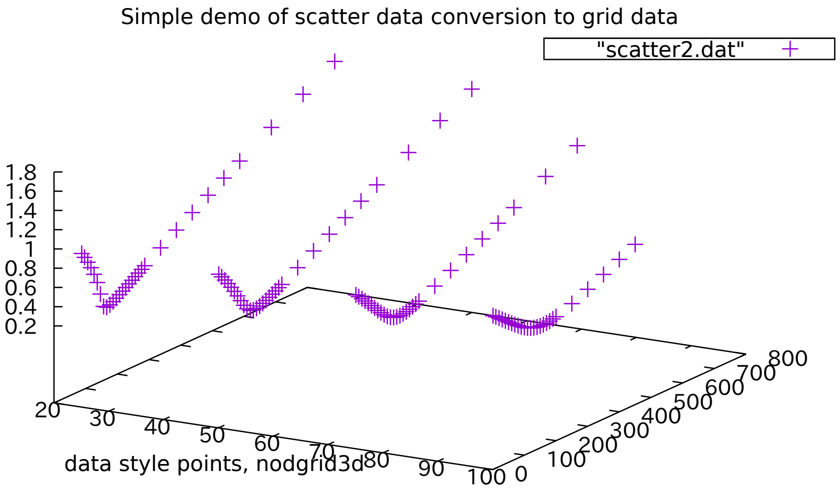
Simple demo of scatter data conversion to grid data
"hemisphr.dat"
0.7
0.65
0.6
0.55
0.5
0.45
0.4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
data style lines, dgrid3d 10,10,1


Simple demo of scatter data conversion to grid data
"hemisphr.dat"
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
data style lines, dgrid3d ,,4

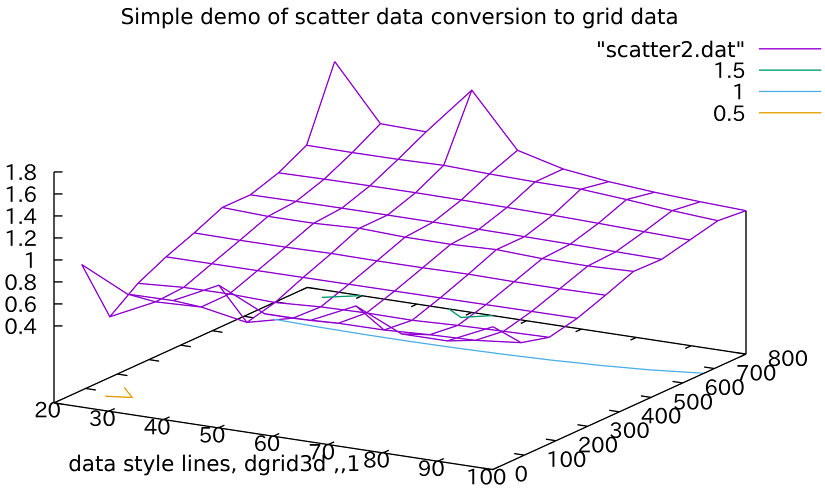


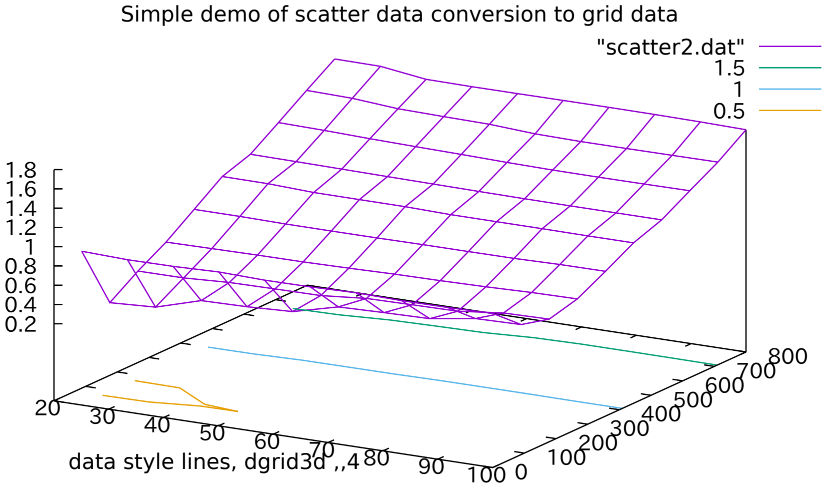
Simple demo of scatter data conversion to grid data
"hemisphr.dat"
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
data style lines, dgrid3d ,,16


Simple demo of scatter data conversion to grid data
"hemisphr.dat"
0.8
0.6
0.4
0.2
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
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
data style lines, dgrid3d ,,16, contour



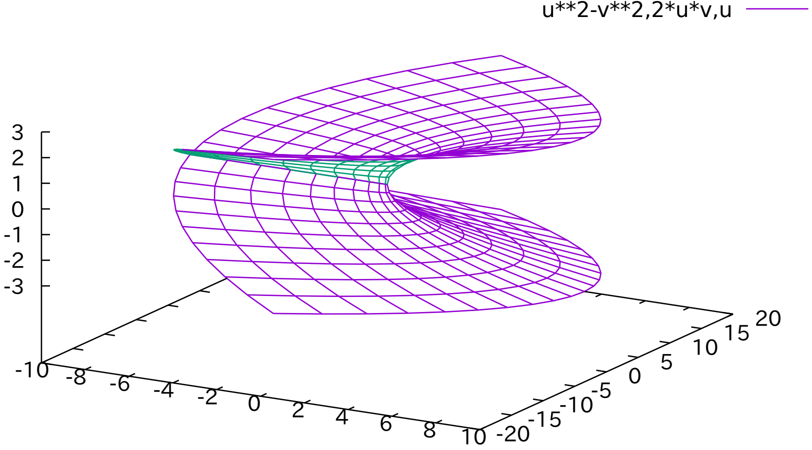
Simple demo of scatter data conversion to grid data
"scatter2.dat"
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
data style points, nodgrid3d

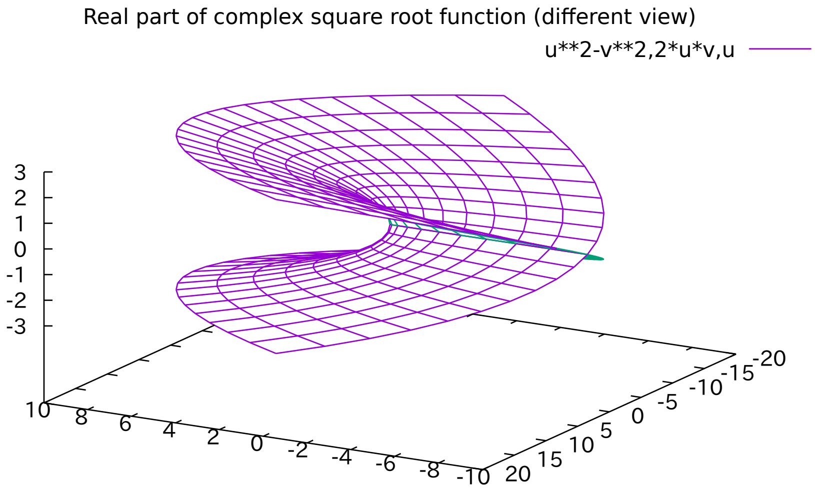

Simple demo of scatter data conversion to grid data
"scatter2.dat"
1.5
1
0.5
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
data style lines, dgrid3d ,,1



Simple demo of scatter data conversion to grid data
"scatter2.dat"
1.5
1
0.5
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800
data style lines, dgrid3d ,,4

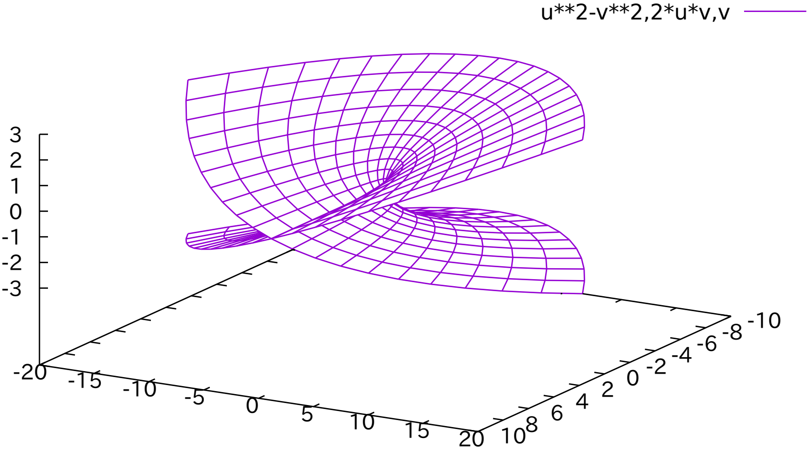
Real part of complex square root function



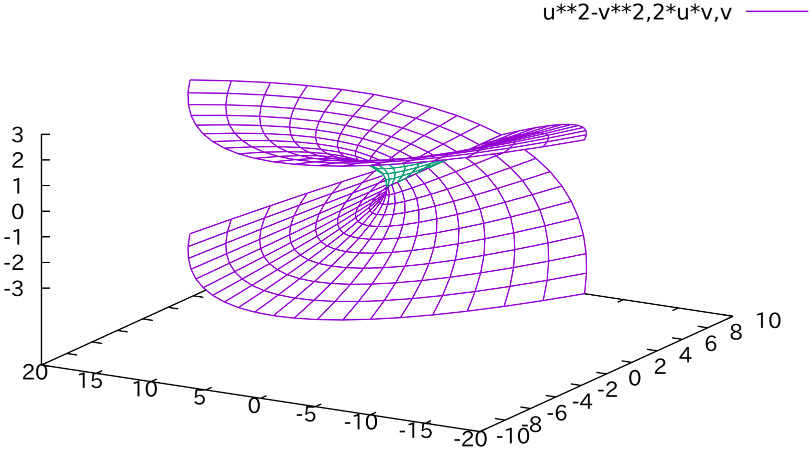


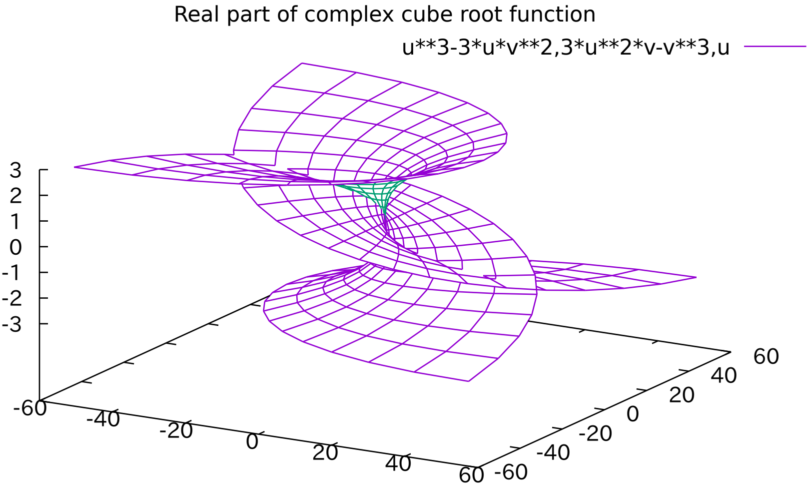
Real part of complex square root function (different view)
u**2-v**2,2*u*v,u
3
2
1
0
-1
-2
-3
10
8
6
4
2
0
-2
-4
-6
-8
-10
20
15
10
5
0
-5
-10
-15
-20

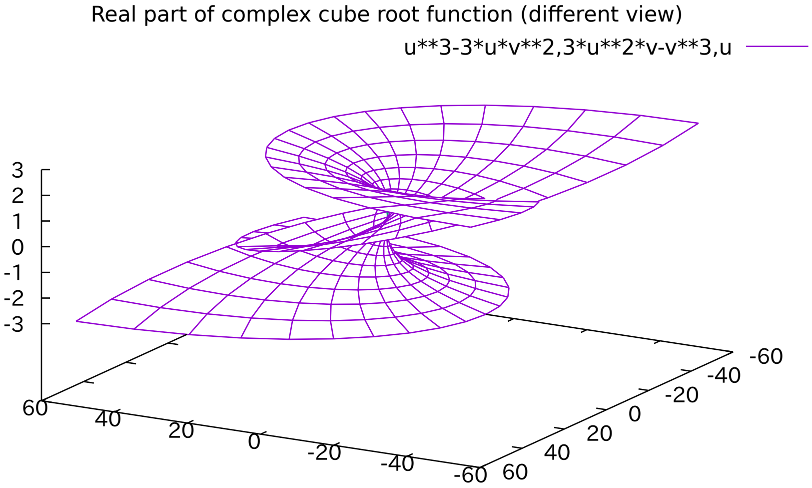
Imaginary part of complex square root function

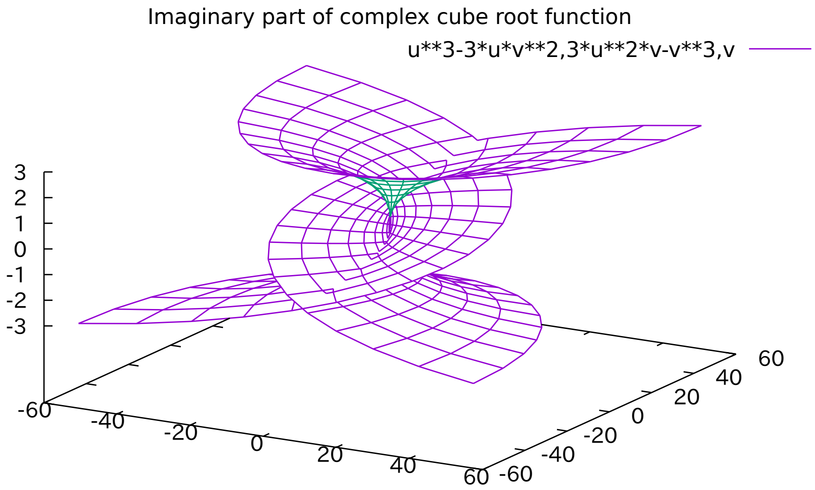


Imaginary part of complex square root function (different view)

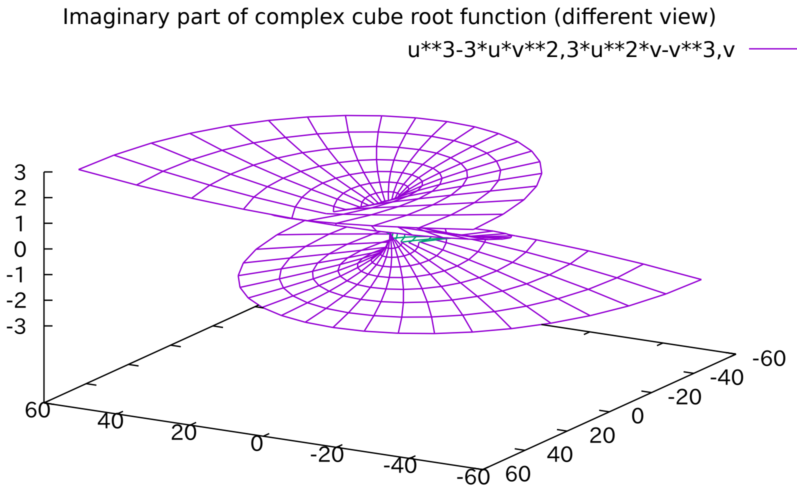


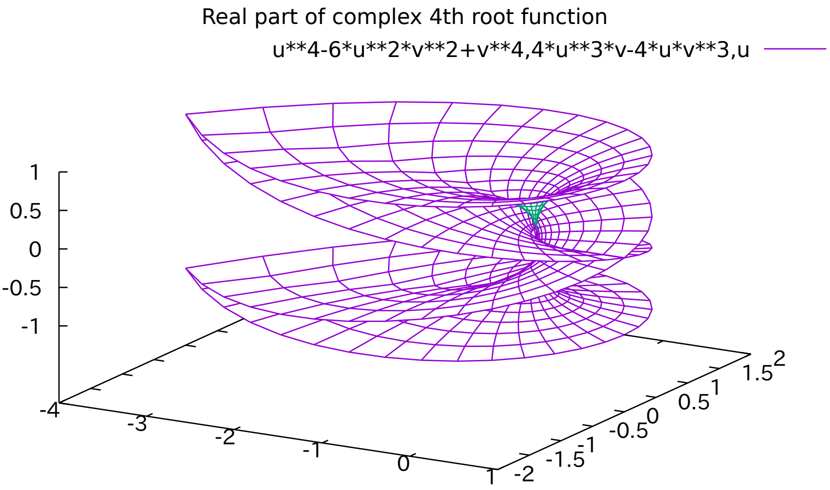

Real part of complex cube root function
u**3-3*u*v**2,3*u**2*v-v**3,u
3
2
1
0
-1
-2
-3
-60
-40
-20
0
20
40
60
-60
-40
-20
0
20
40
60

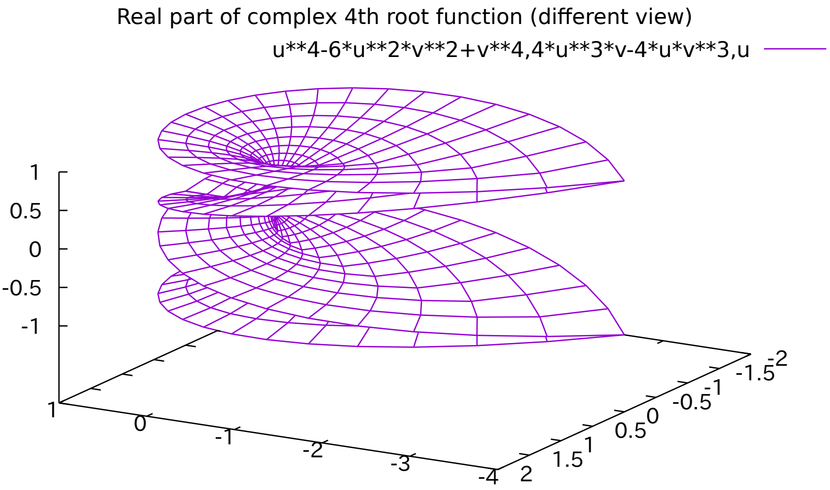

Real part of complex cube root function (different view)
u**3-3*u*v**2,3*u**2*v-v**3,u
3
2
1
0
-1
-2
-3
60
40
20
0
-20
-40
-60
-60
-40
-20
0
20
40
60

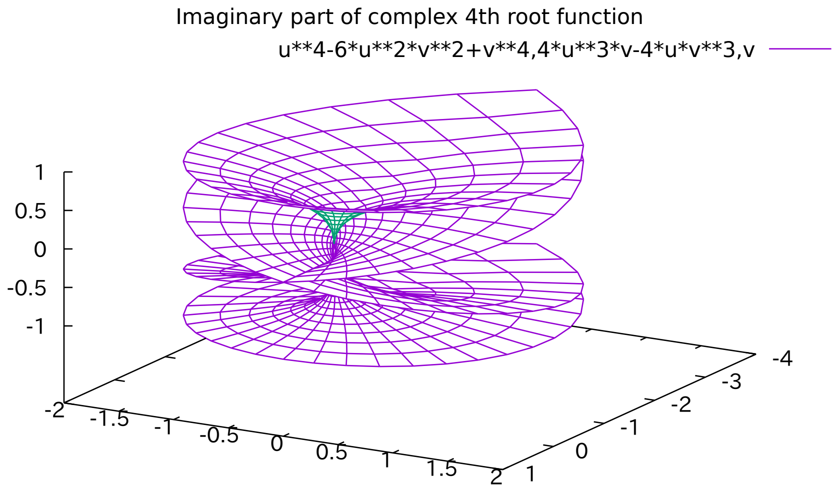


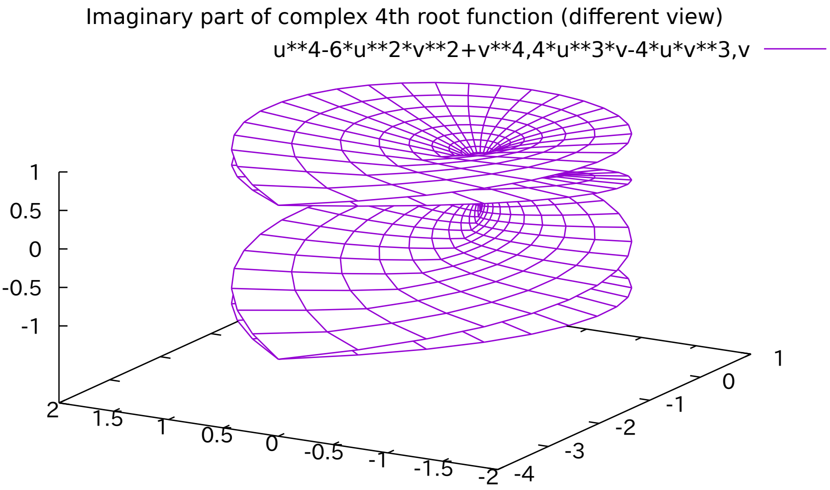
Imaginary part of complex cube root function
u**3-3*u*v**2,3*u**2*v-v**3,v
3
2
1
0
-1
-2
-3
-60
-40
-20
0
20
40
60
-60
-40
-20
0
20
40
60


Imaginary part of complex cube root function (different view)
u**3-3*u*v**2,3*u**2*v-v**3,v
3
2
1
0
-1
-2
-3
60
40
20
0
-20
-40
-60
-60
-40
-20
0
20
40
60

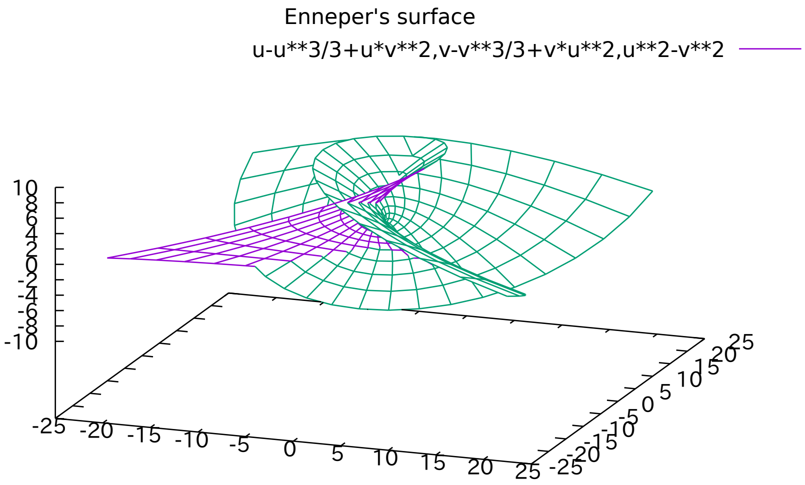

Real part of complex 4th root function
u**4-6*u**2*v**2+v**4,4*u**3*v-4*u*v**3,u
1
0.5
0
-0.5
-1
-4
-3
-2
-1
0
1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

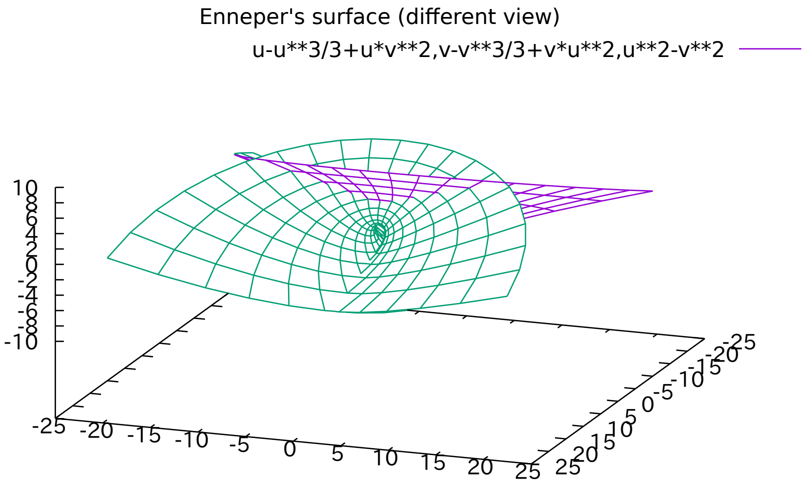

Real part of complex 4th root function (different view)
u**4-6*u**2*v**2+v**4,4*u**3*v-4*u*v**3,u
1
0.5
0
-0.5
-1
1
0
-1
-2
-3
-4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2


Imaginary part of complex 4th root function
u**4-6*u**2*v**2+v**4,4*u**3*v-4*u*v**3,v
1
0.5
0
-0.5
-1
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
1
0
-1
-2
-3
-4

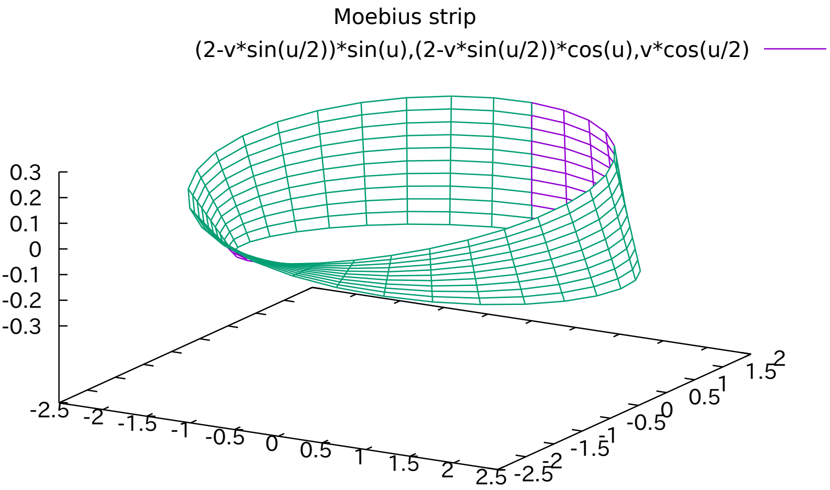


Imaginary part of complex 4th root function (different view)
u**4-6*u**2*v**2+v**4,4*u**3*v-4*u*v**3,v
1
0.5
0
-0.5
-1
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-4
-3
-2
-1
0
1

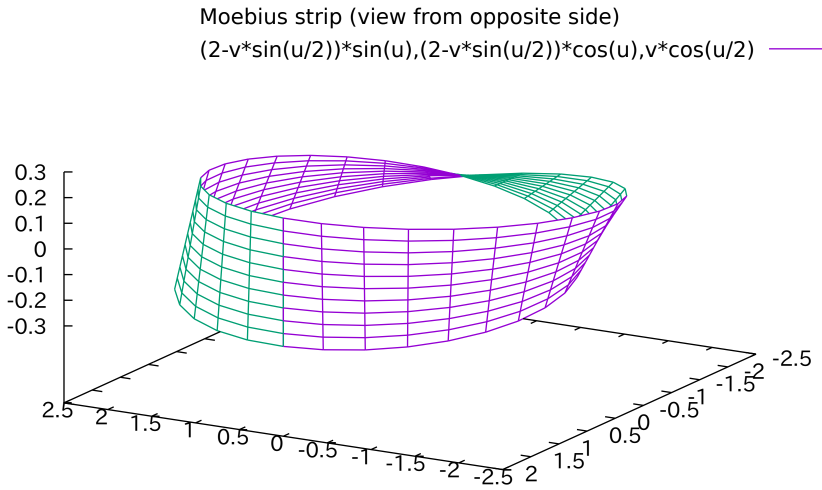

Enneper's surface
u-u**3/3+u*v**2,v-v**3/3+v*u**2,u**2-v**2
10
8
6
4
2
0
-2
-4
-6
-8
-10
-25
-20
-15
-10
-5
0
5
10
15
20
25



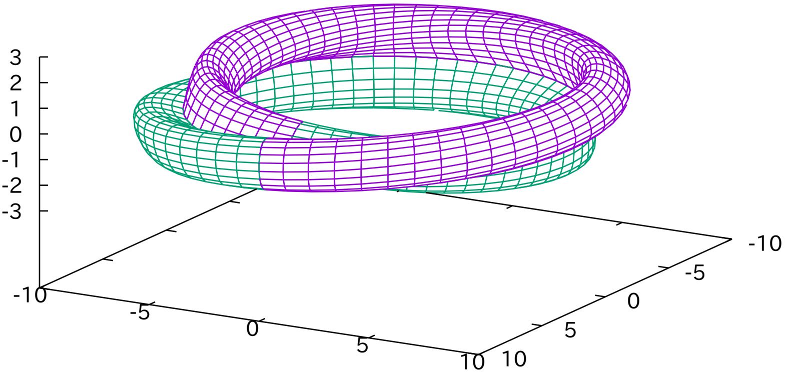
Enneper's surface (different view)
u-u**3/3+u*v**2,v-v**3/3+v*u**2,u**2-v**2
10
8
6
4
2
0
-2
-4
-6
-8
-10
-25 -20 -15 -10 -5 0 5 10 15 20 25
-25 -20 -15 -10 -5 0 5 10 15 20 25

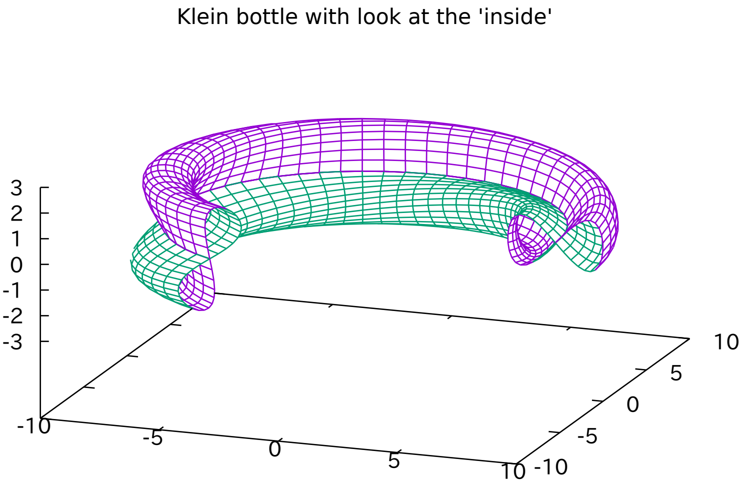


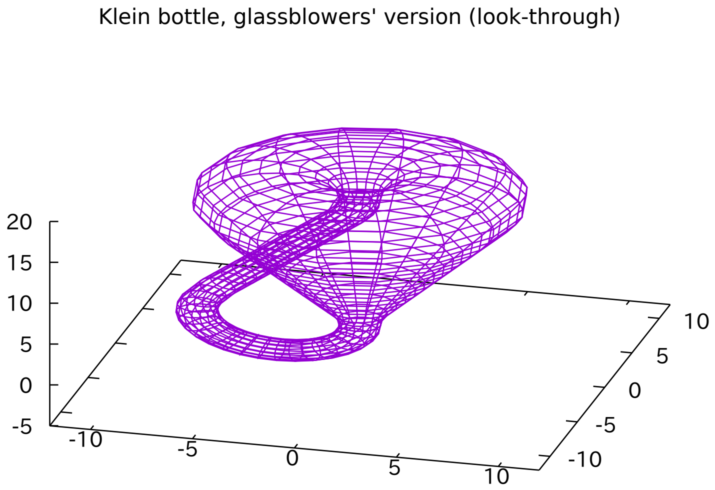
Moebius strip
(2-v*sin(u/2))*sin(u),(2-v*sin(u/2))*cos(u),v*cos(u/2)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Moebius strip (view from opposite side)
(2-v*sin(u/2))*sin(u),(2-v*sin(u/2))*cos(u),v*cos(u/2)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5

Klein bottle

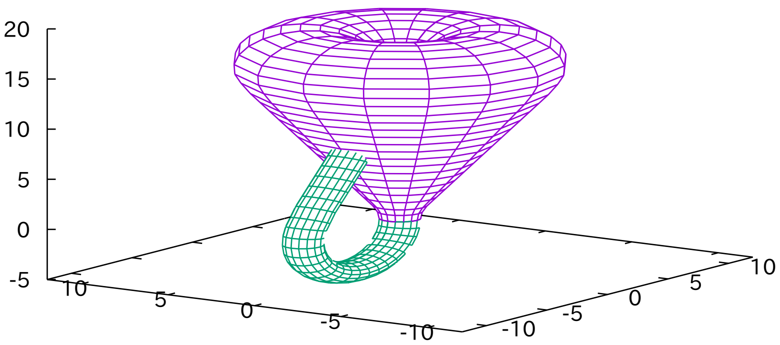


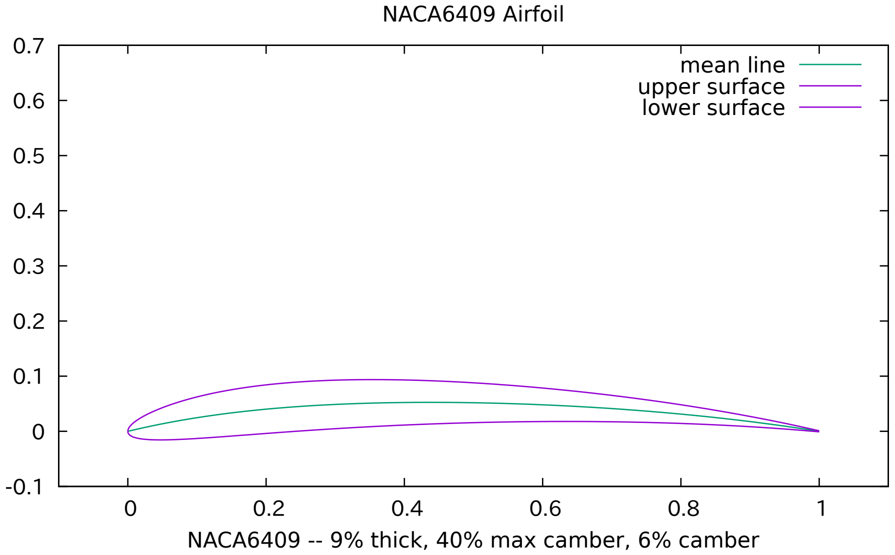

Klein bottle with look at the 'inside'
3
2
1
0
-1
-2
-3
-10
-5
0
5
10
-10
-5
0
5
10

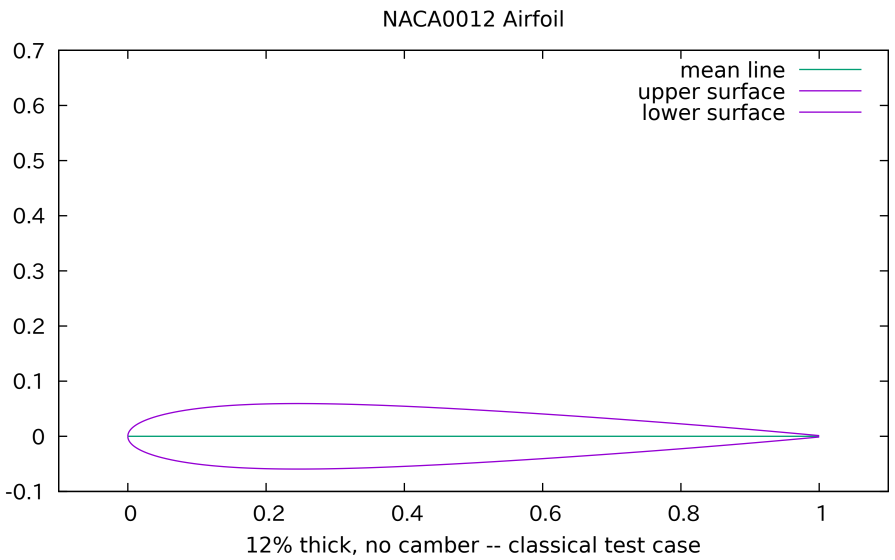


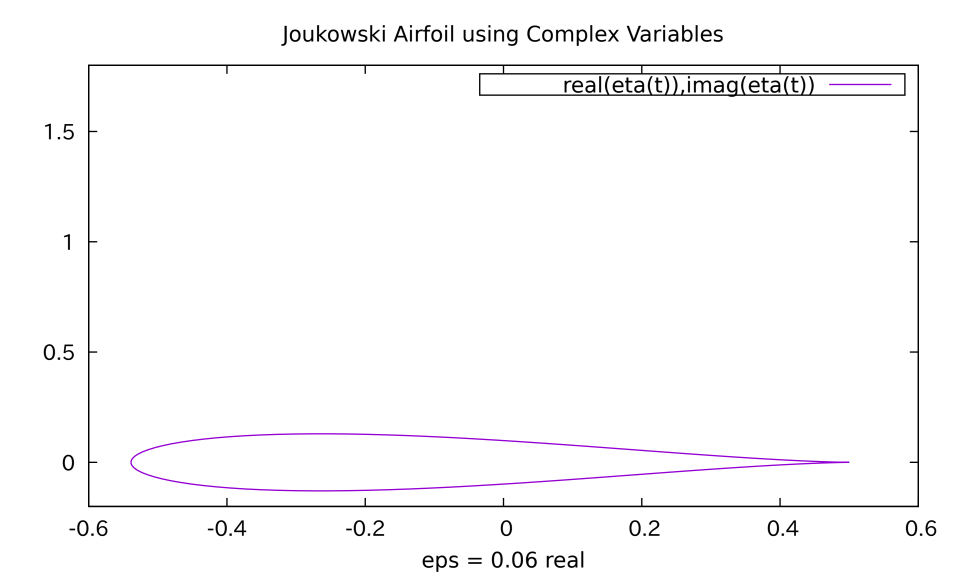
Klein bottle, glassblowers' version (look-through)
20
15
10
5
0
-5
-10
-5
0
5
10
10
5
0
-5
-10

Klein bottle, glassblowers' version (solid)

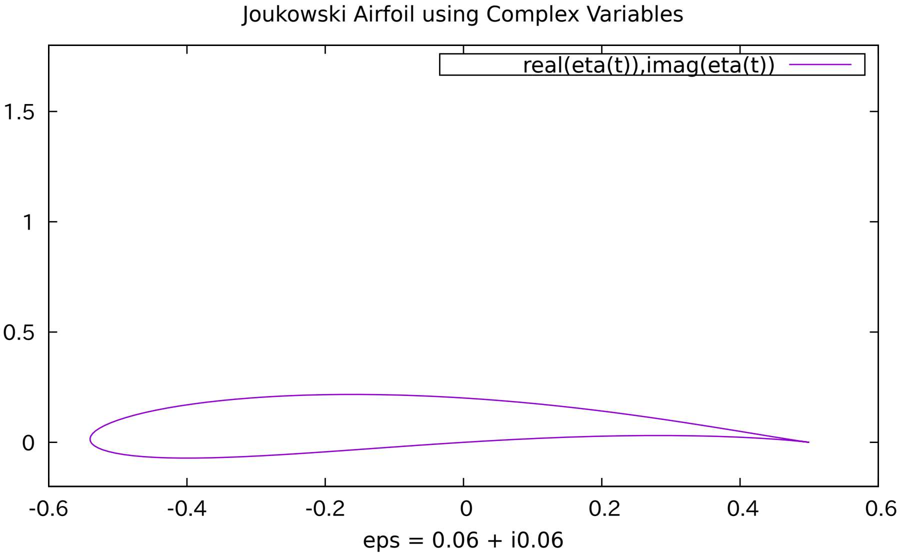



NACA6409 Airfoil
mean line
upper surface
lower surface
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.2
0.4
0.6
0.8
1
NACA6409 -- 9% thick, 40% max camber, 6% camber

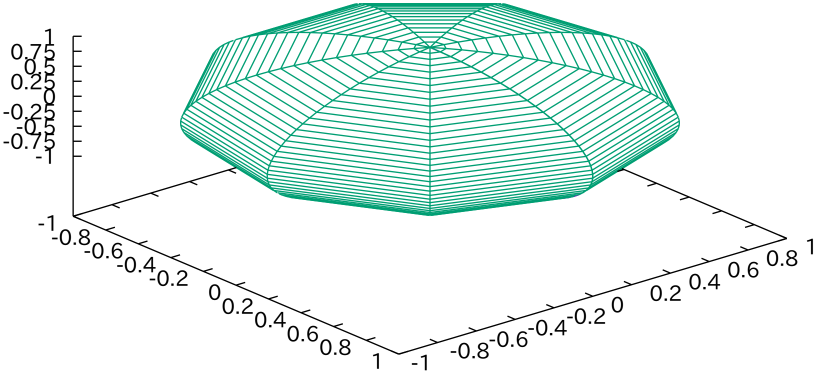

NACA0012 Airfoil
mean line
upper surface
lower surface
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.2
0.4
0.6
0.8
1
12% thick, no camber -- classical test case

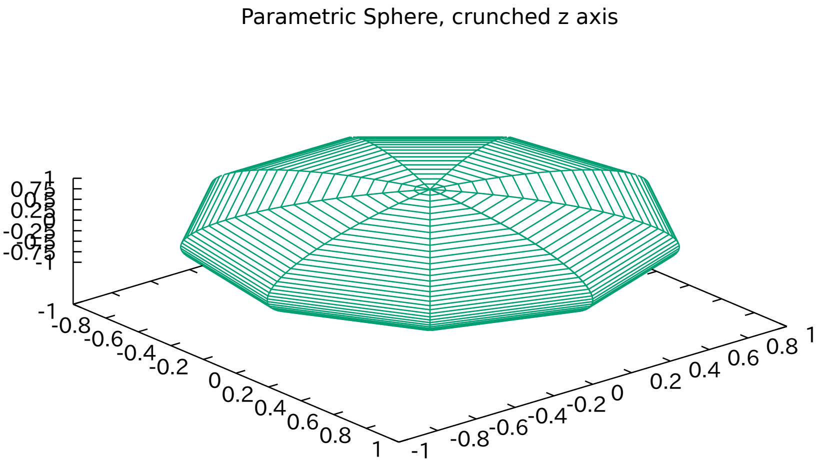


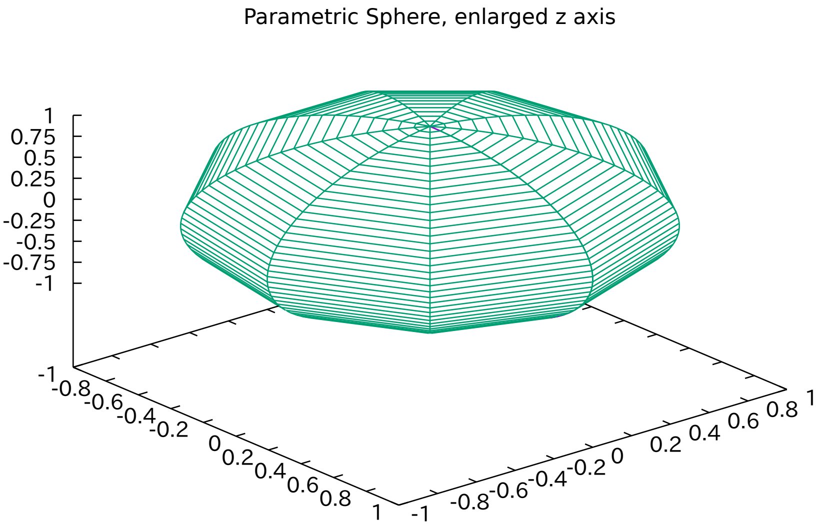
Joukowski Airfoil using Complex Variables
real(eta(t)),imag(eta(t))
1.5
1
0.5
0
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
eps = 0.06 real

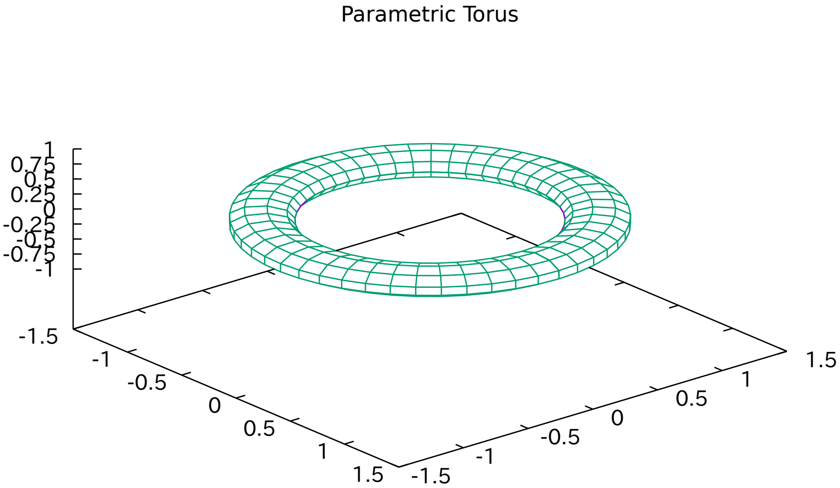

Joukowski Airfoil using Complex Variables
real(eta(t)),imag(eta(t))
1.5
1
0.5
0
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
eps = 0.06 + i0.06

Parametric Sphere

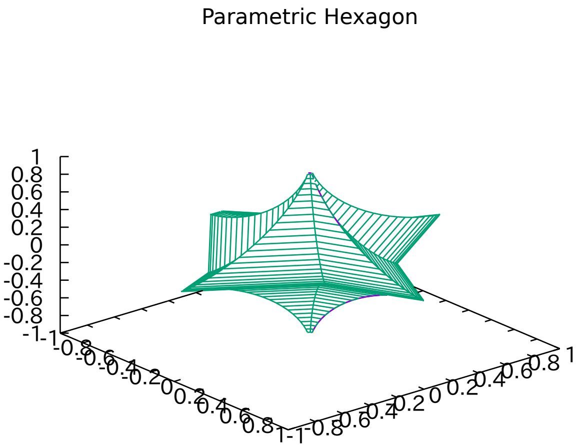



Parametric Sphere, crunched z axis
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

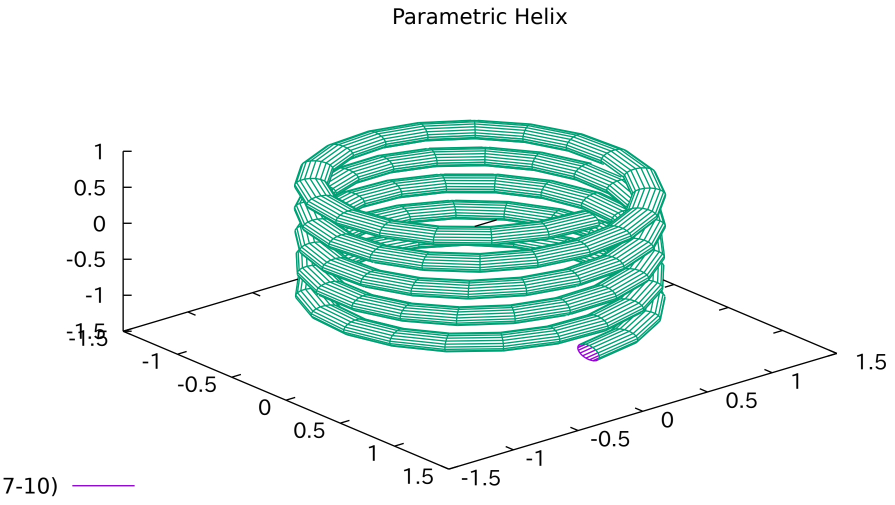

Parametric Sphere, enlarged z axis
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



Parametric Torus
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
-1.5
-1
-0.5
0
0.5
1
1.5
-1.5
-1
-0.5
0
0.5
1
1.5

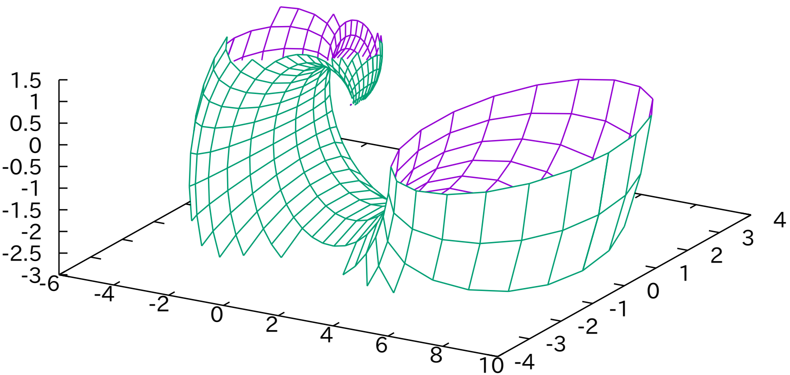


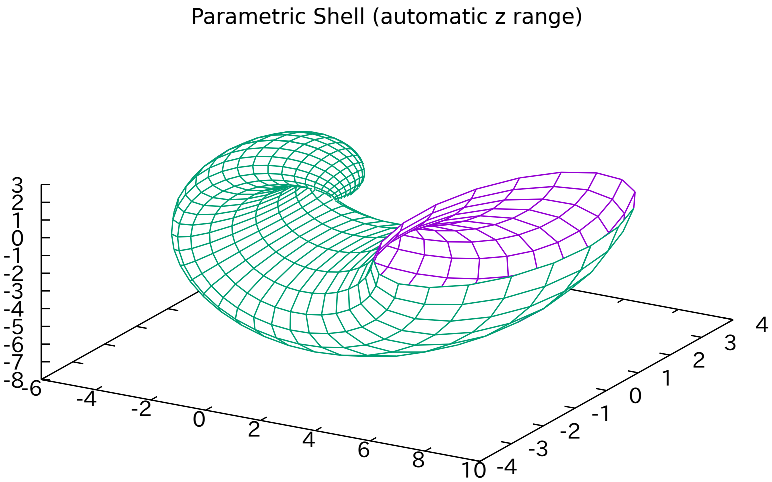
Parametric Hexagon
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1



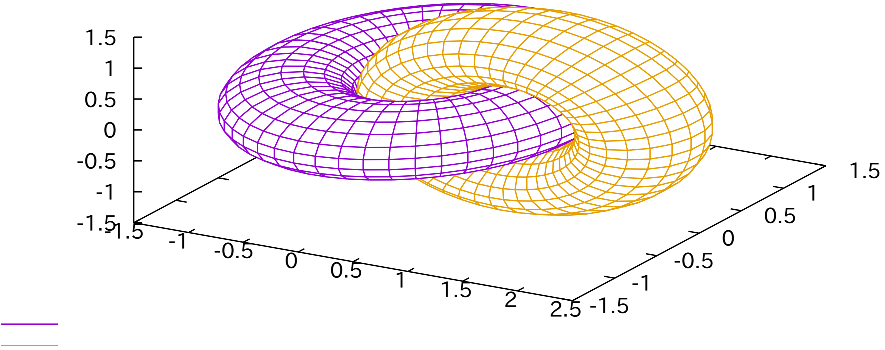
Parametric Helix
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5
-1.5
-1
-0.5
0
0.5
1
1.5
7-10)

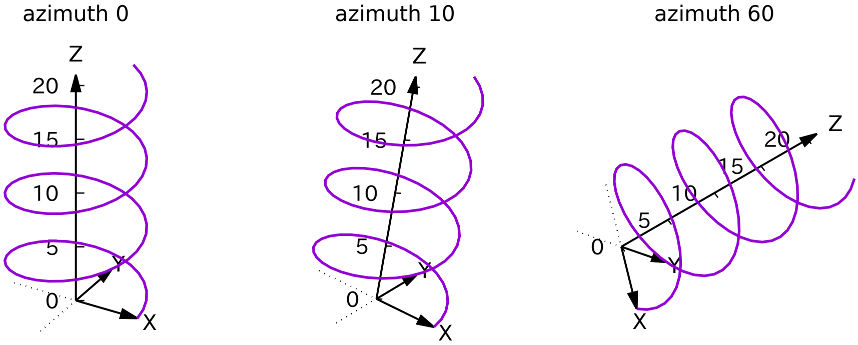
Parametric Shell (clipped to limited z range)



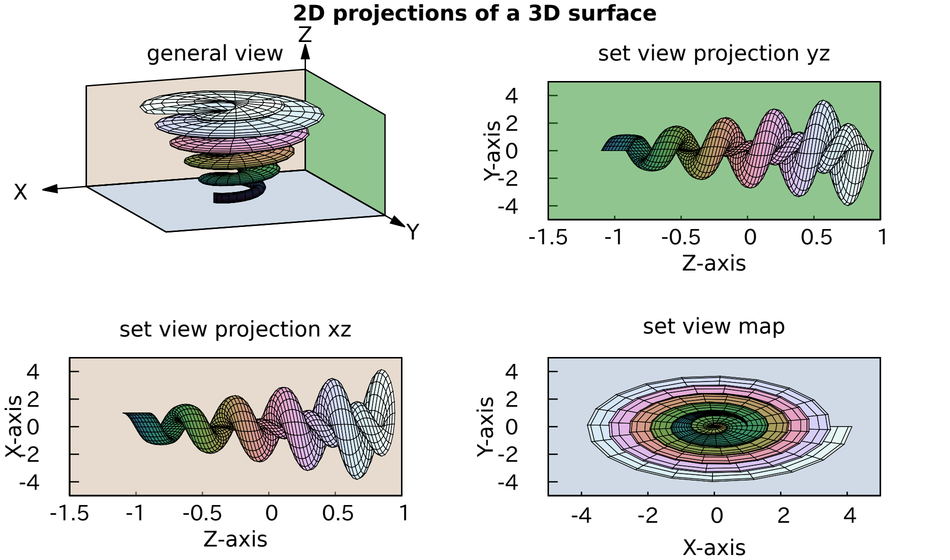

Parametric Shell (automatic z range)
3
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-6
-4
-2
0
2
4
6
8
10
-4
-3
-2
-1
0
1
2
3
4

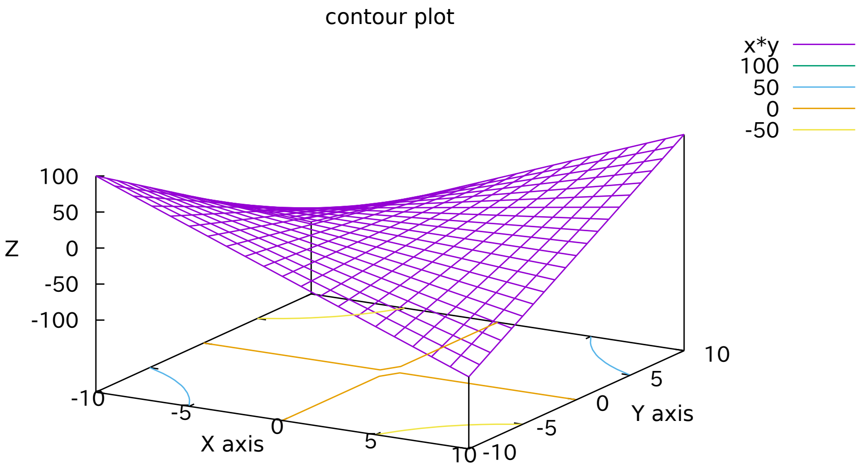
Interlocking Tori



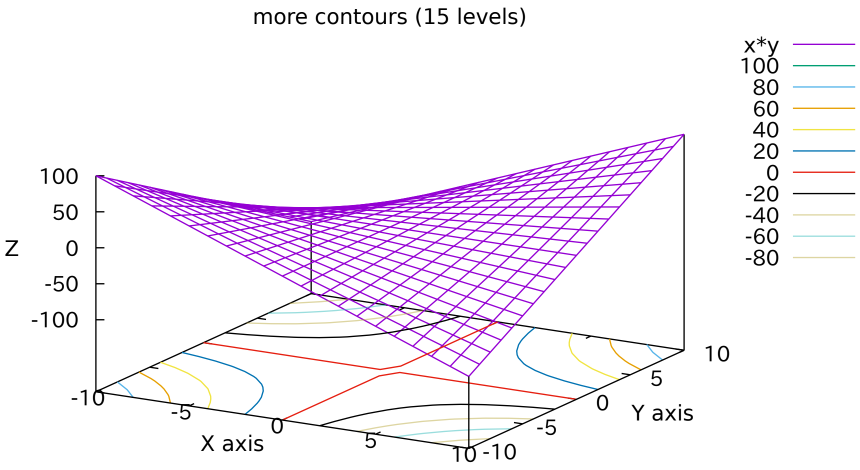


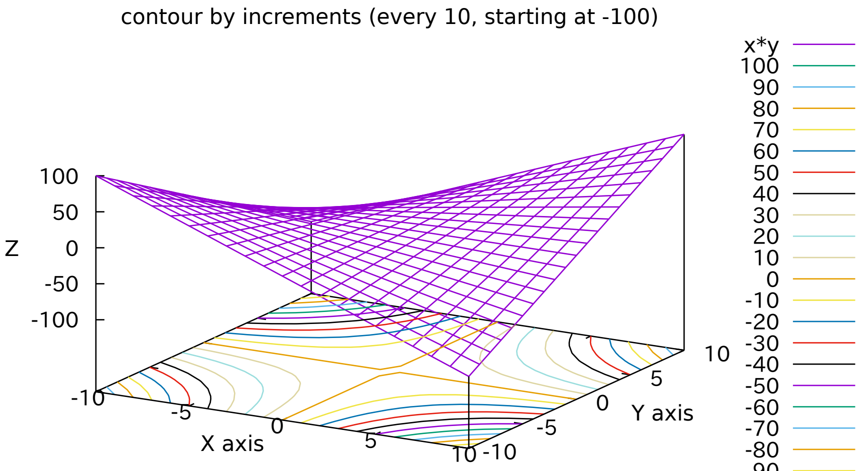
azimuth 0
Z
20
15
10
5
0
Y
X
azimuth 10
Z
20
15
10
5
0
Y
X
azimuth 60
Z
20
15
10
5
0
Y
X

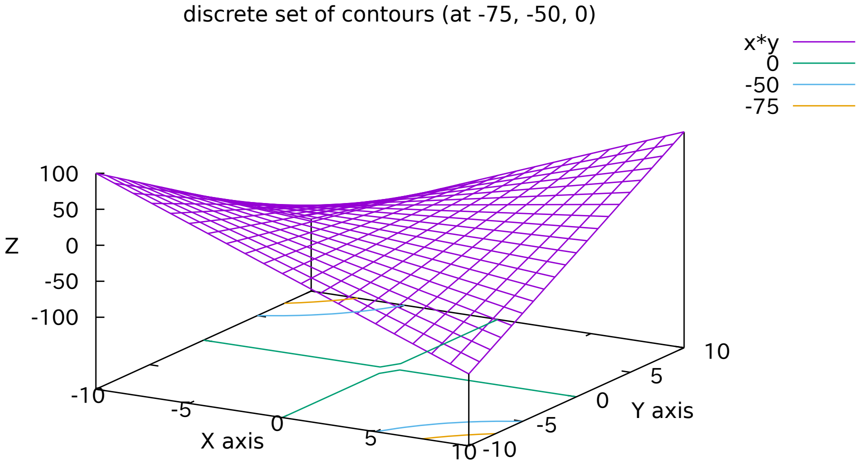

2D projections of a 3D surface
general view
Z
X
Y
set view projection yz
Y-axis
Z-axis
4
2
0
-2
-4
-1.5
-1
-0.5
0
0.5
1
set view projection xz
X-axis
Z-axis
4
2
0
-2
-4
-1.5
-1
-0.5
0
0.5
1
set view map
Y-axis
X-axis
4
2
0
-2
-4
-4
-2
0
2
4

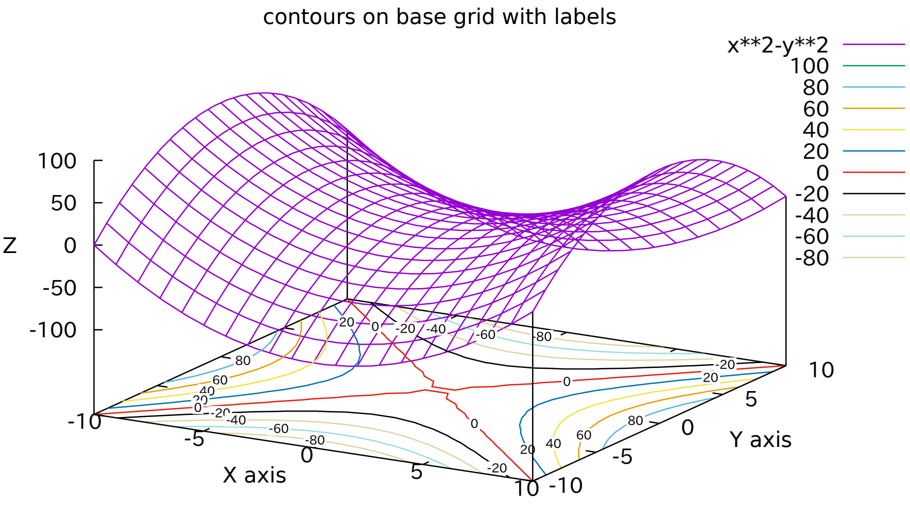


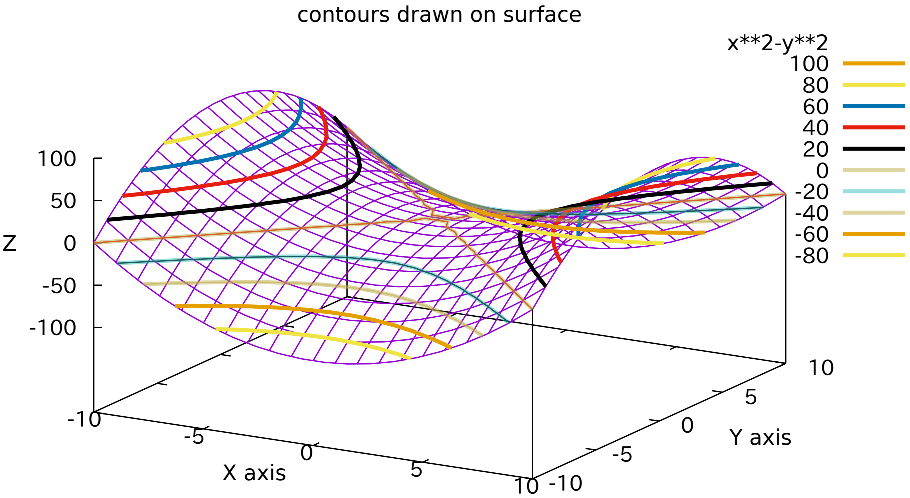
contour plot
x*y
100
50
0
-50
100
50
Z
0
-50
-100
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis


more contours (15 levels)
x*y
100
80
60
40
20
0
-20
-40
-60
-80
Z
100
50
0
-50
-100
X axis
-10
-5
0
5
10
Y axis
-10
-5
0
5
10

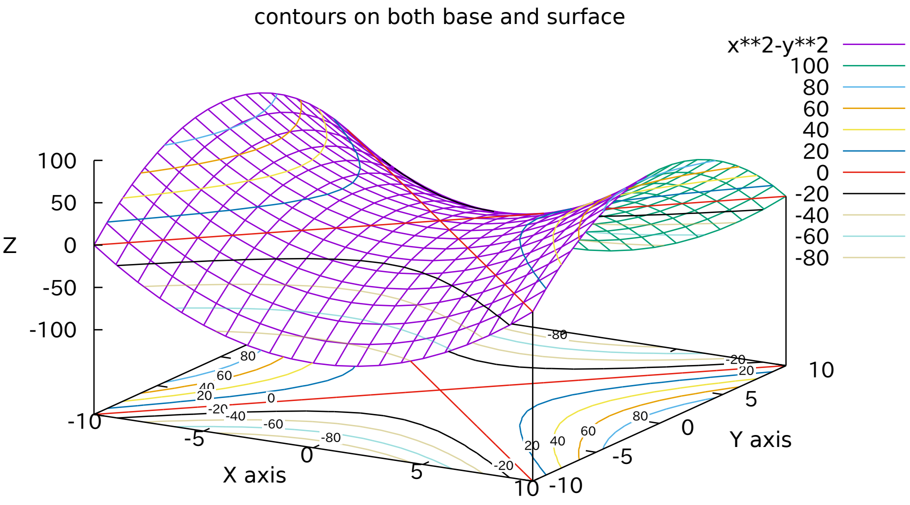

contour by increments (every 10, starting at -100)
Z
100
50
0
-50
-100
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis
x*y
100
90
80
70
60
50
40
30
20
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90

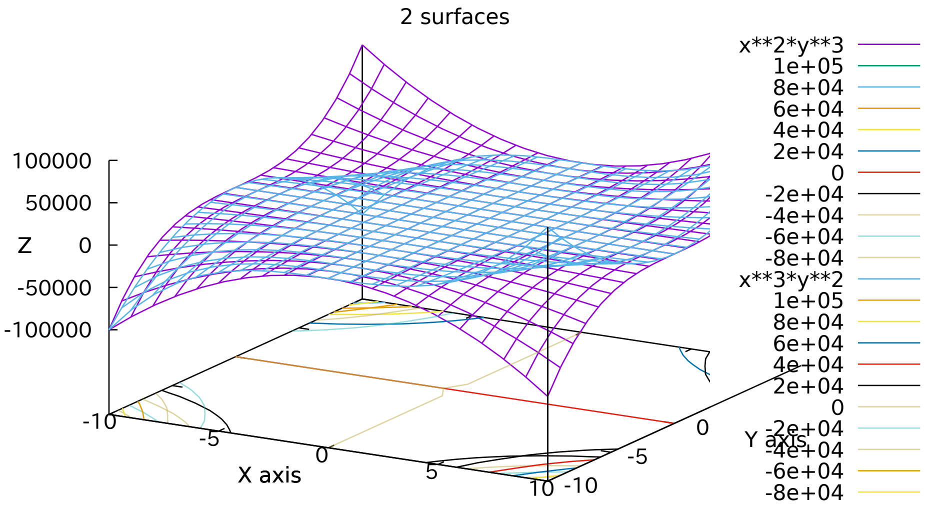

discrete set of contours (at -75, -50, 0)
x*y
0
-50
-75
100
50
Z
0
-50
-100
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis


contours on base grid with labels
x**2-y**2
100
80
60
40
20
0
-20
-40
-60
-80
Z
100
50
0
-50
-100
X axis
Y axis
-10
-5
0
5
10

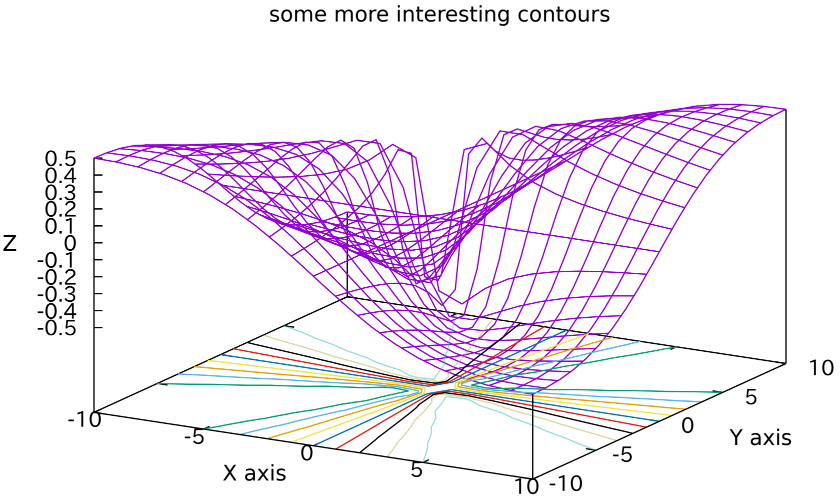


contours drawn on surface
x**2-y**2
100
80
60
40
20
0
-20
-40
-60
-80
Z
100
50
0
-50
-100
-10
-5
0
5
10
X axis
-10
-5
0
5
10
Y axis

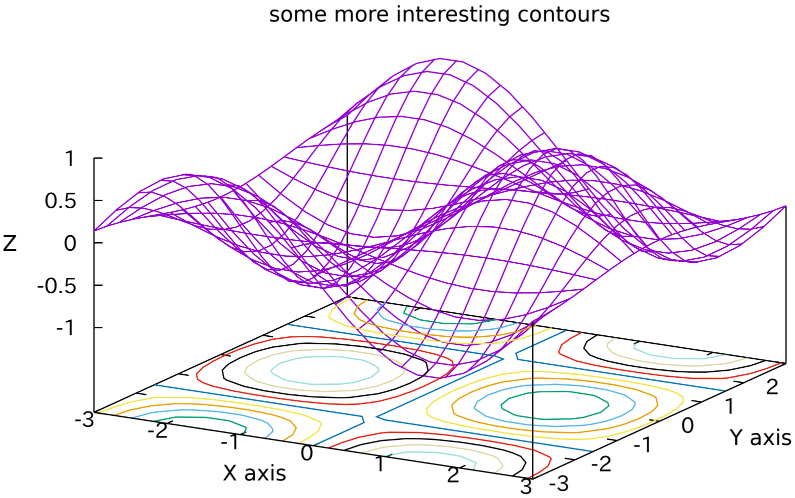

contours on both base and surface
x**2-y**2
100
80
60
40
20
0
-20
-40
-60
-80
Z
X axis
Y axis

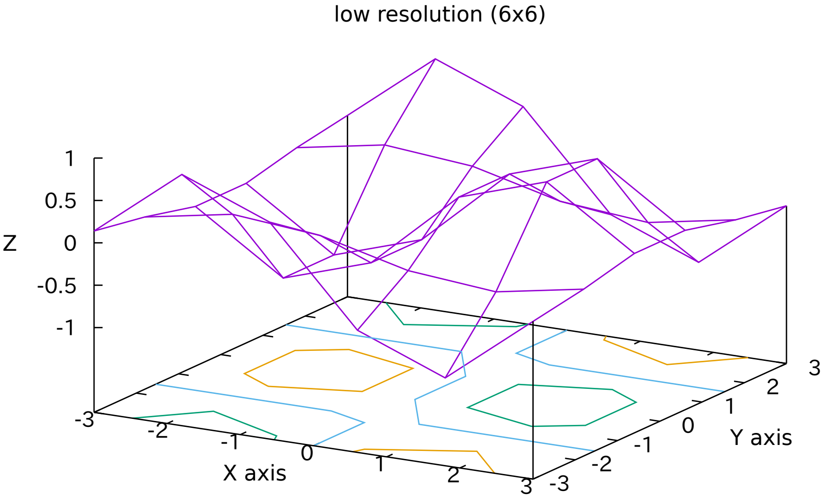


2 surfaces
x**2*y**3
1e+05
8e+04
6e+04
4e+04
2e+04
0
-2e+04
-4e+04
-6e+04
-8e+04
x**3*y**2
1e+05
8e+04
6e+04
4e+04
2e+04
0
-2e+04
-4e+04
-6e+04
-8e+04
100000
50000
Z
0
-50000
-100000
-10
-5
0
5
10
X axis
-10
-5
0
Y axis

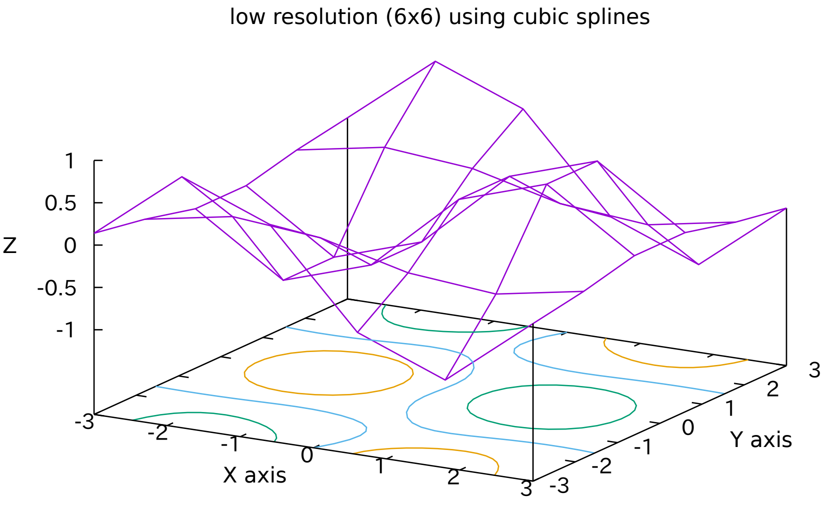


some more interesting contours
Z
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
X axis
-10
-5
0
5
10
Y axis
-10
-5
0
5
10

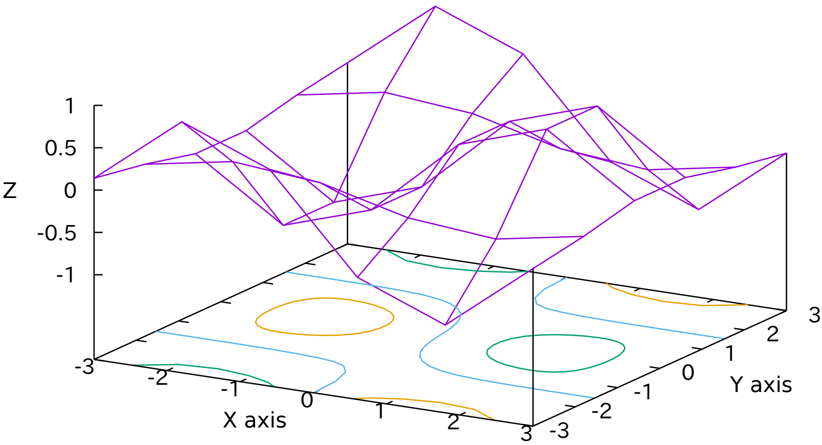

some more interesting contours
Z
1
0.5
0
-0.5
-1
X axis
-3
-2
-1
0
1
2
3
Y axis
-3
-2
-1
0
1
2
3

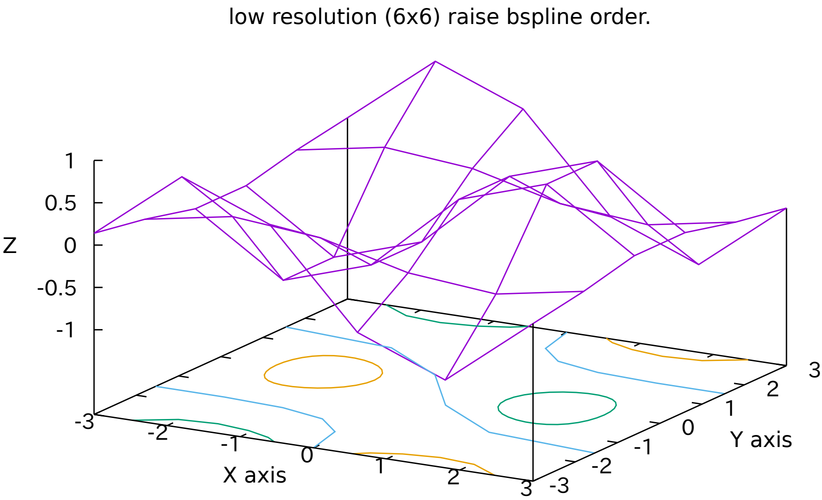


low resolution (6x6)
Z
1
0.5
0
-0.5
-1
X axis
-3
-2
-1
0
1
2
3
Y axis
-3
-2
-1
0
1
2
3

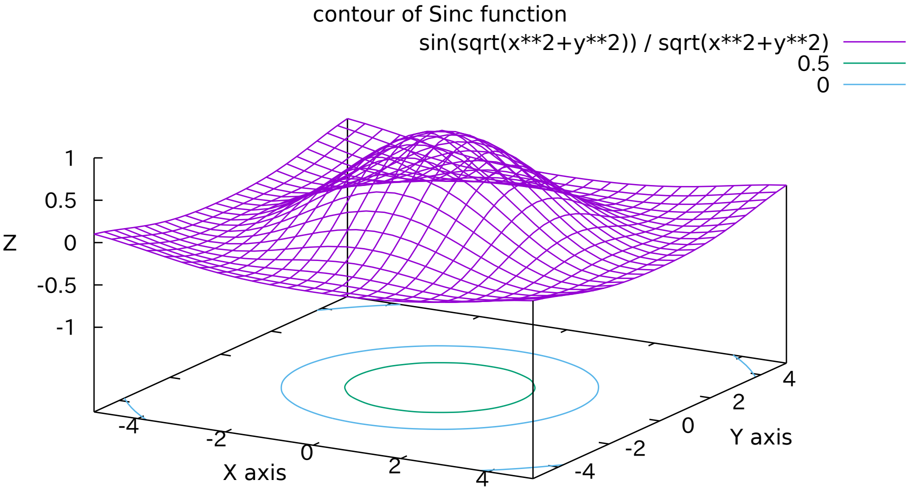

low resolution (6x6) using cubic splines
Z
1
0.5
0
-0.5
-1
X axis
-3
-2
-1
0
1
2
3
Y axis
-3
-2
-1
0
1
2
3

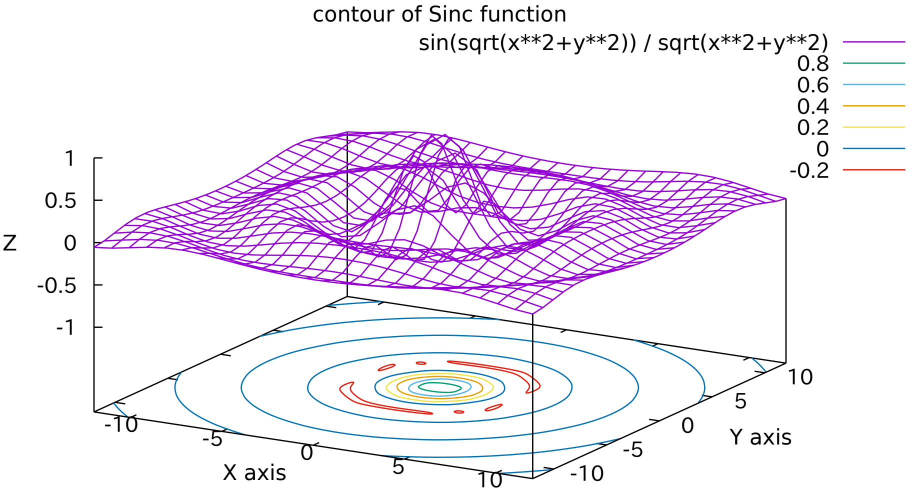
low resolution (6x6) using bspline approx.

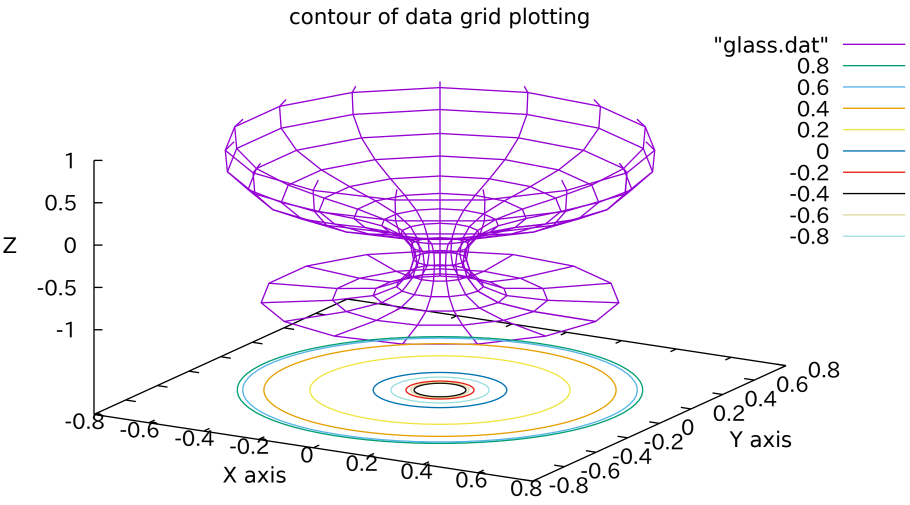


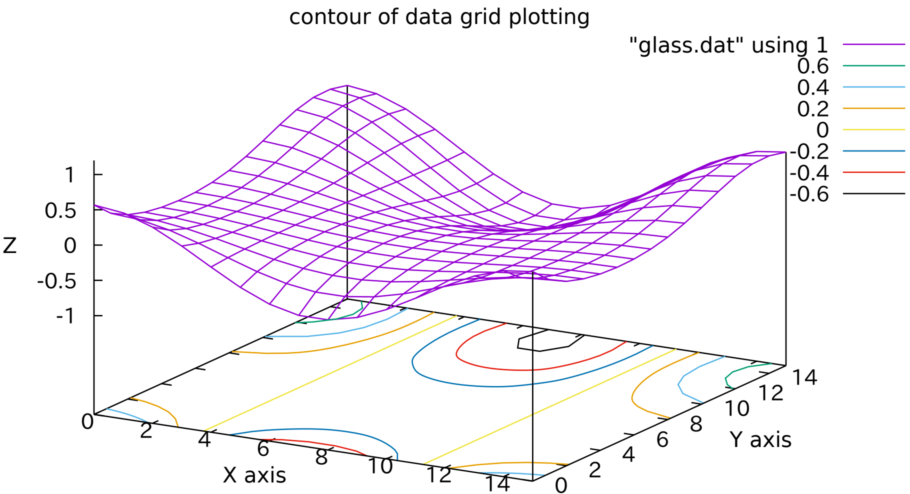


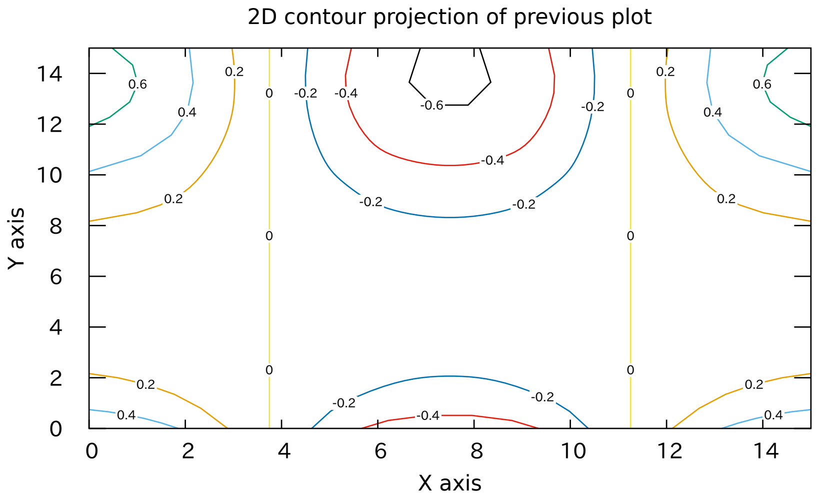
low resolution (6x6) raise bspline order.
Z
1
0.5
0
-0.5
-1
-3
-2
-1
0
1
2
3
X axis
-3
-2
-1
0
1
2
3
Y axis

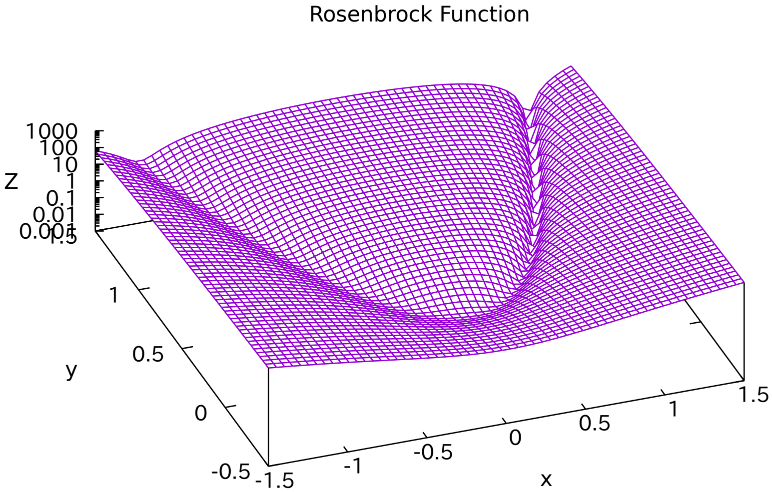

contour of Sinc function
sin(sqrt(x**2+y**2)) / sqrt(x**2+y**2)
0.5
0
Z
1
0.5
0
-0.5
-1
X axis
-4
-2
0
2
4
Y axis
-4
-2
0
2
4

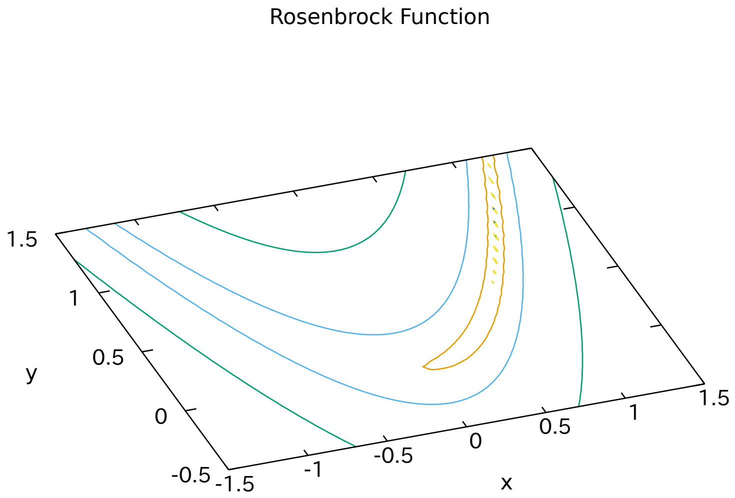

contour of Sinc function
sin(sqrt(x**2+y**2)) / sqrt(x**2+y**2)
0.8
0.6
0.4
0.2
0
-0.2
Z
1
0.5
0
-0.5
-1
X axis
-10
-5
0
5
10
Y axis
10
5
0
-5
-10


contour of data grid plotting
"glass.dat"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
Z
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
X axis
Y axis

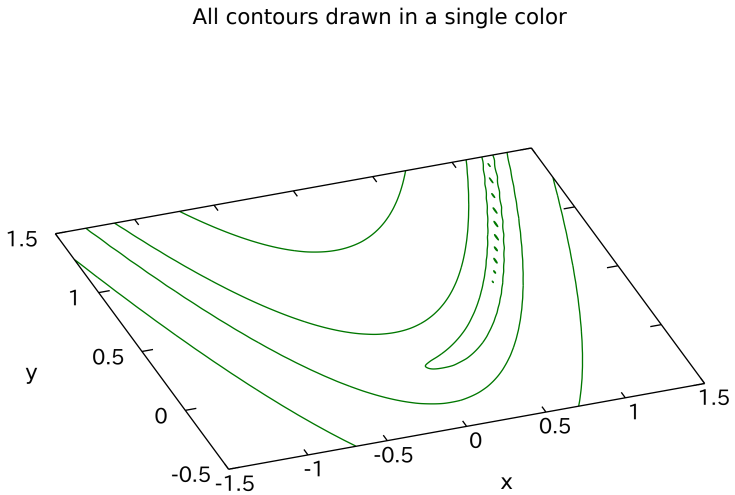

contour of data grid plotting
"glass.dat" using 1
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
Z
1
0.5
0
-0.5
-1
X axis
0
2
4
6
8
10
12
14
Y axis
0
2
4
6
8
10
12
14


2D contour projection of previous plot
X axis
Y axis
0
2
4
6
8
10
12
14
0.6
0.4
0.2
0
-0.2
-0.4
-0.6

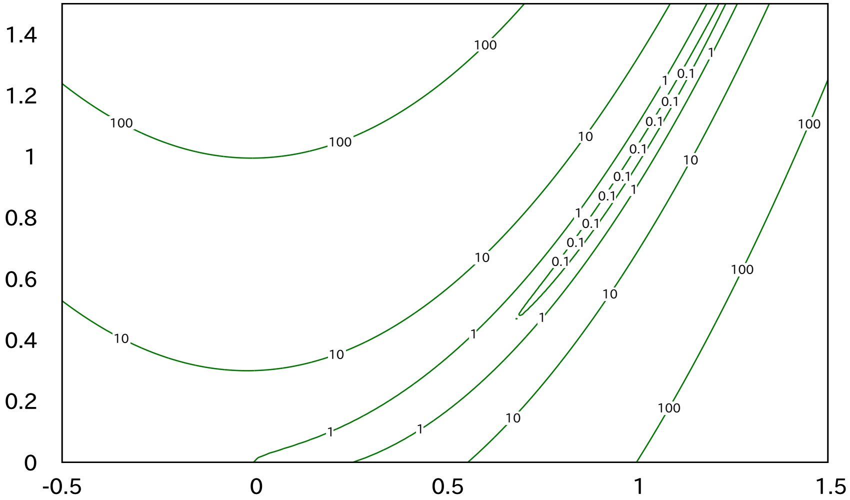

Rosenbrock Function
1000
100
10
1
0.1
0.01
0.001
1.5
Z
1
0.5
0
-0.5
y
-1.5
-1
-0.5
0
0.5
1
1.5
x

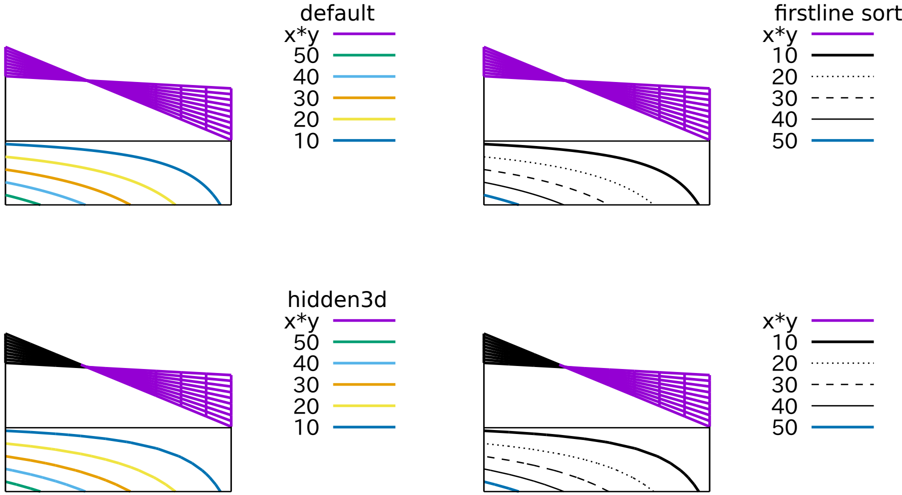


Rosenbrock Function
1.5
1
0.5
y
0
-0.5
-1.5
-1
-0.5
0
0.5
1
1.5
x


All contours drawn in a single color
1.5
1
0.5
y
0
-0.5
-1.5
-1
-0.5
0
0.5
1
1.5
x

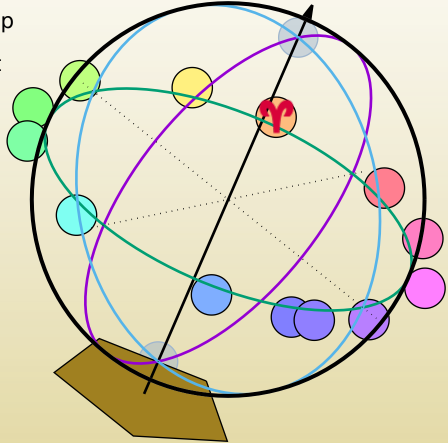
Sometimes it helps to use multiplot

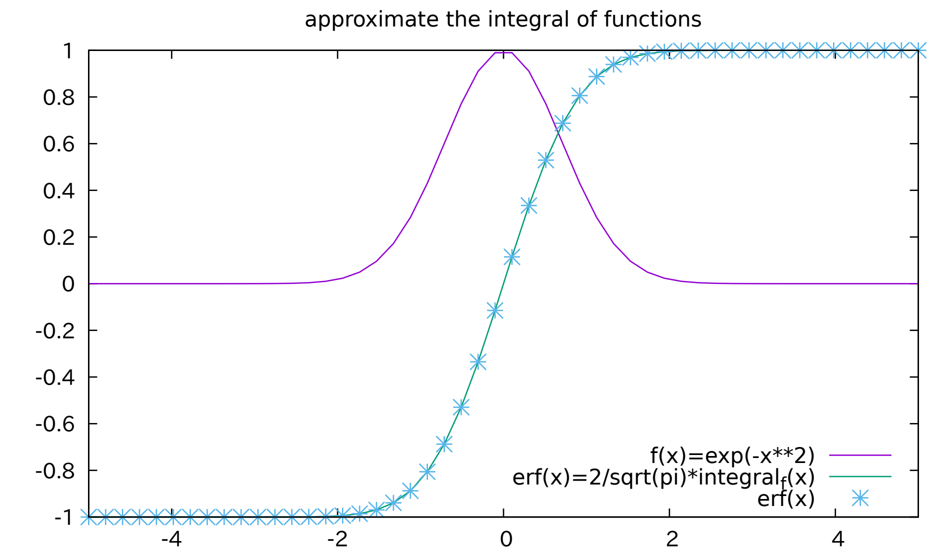


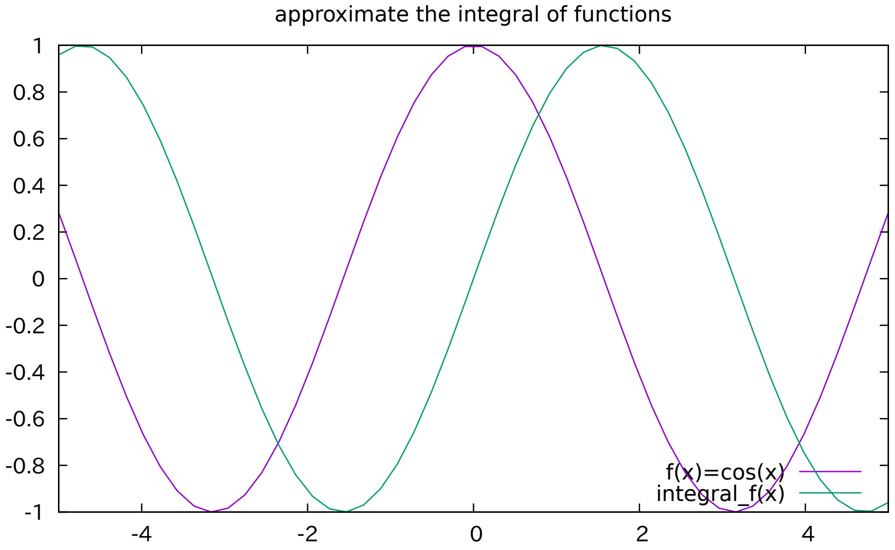

default
x*y
50
40
30
20
10
firstline sort
x*y
10
20
30
40
50
hidden3d
x*y
50
40
30
20
10
x*y
10
20
30
40
50

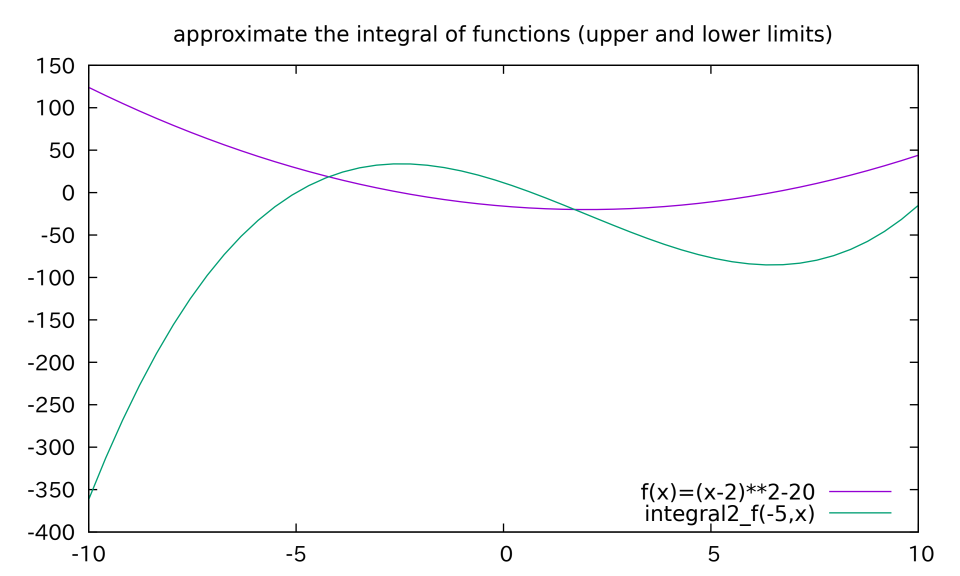
Circle and polygon objects in 3D

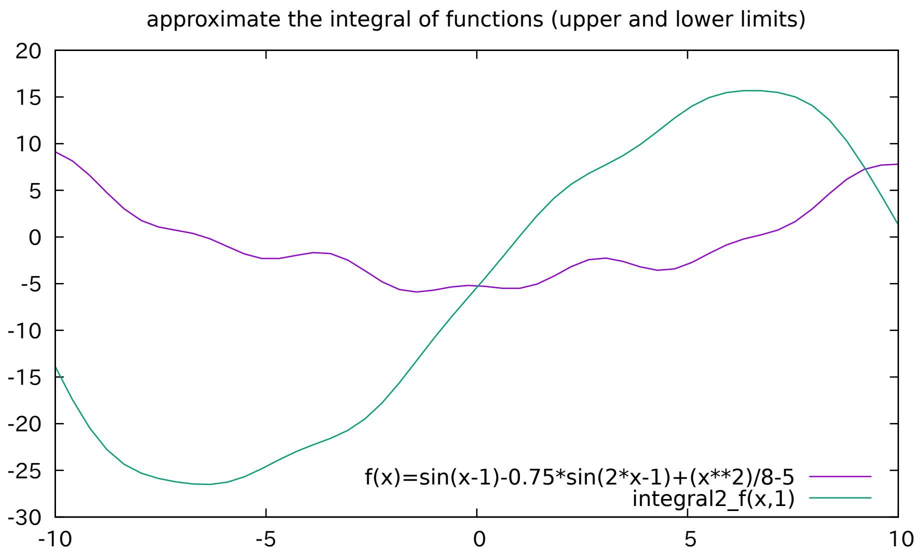
Pixmap use
- Gradient used as a backdrop
- Project logo bottom left
- Icon (Aries) as plot element



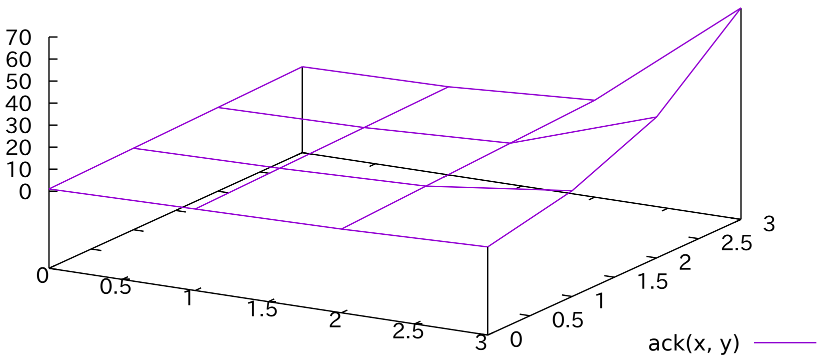

approximate the integral of functions
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-4
-2
0
2
4
f(x)=exp(-x**2)
erf(x)=2/sqrt(pi)*integral_f(x)
erf(x)

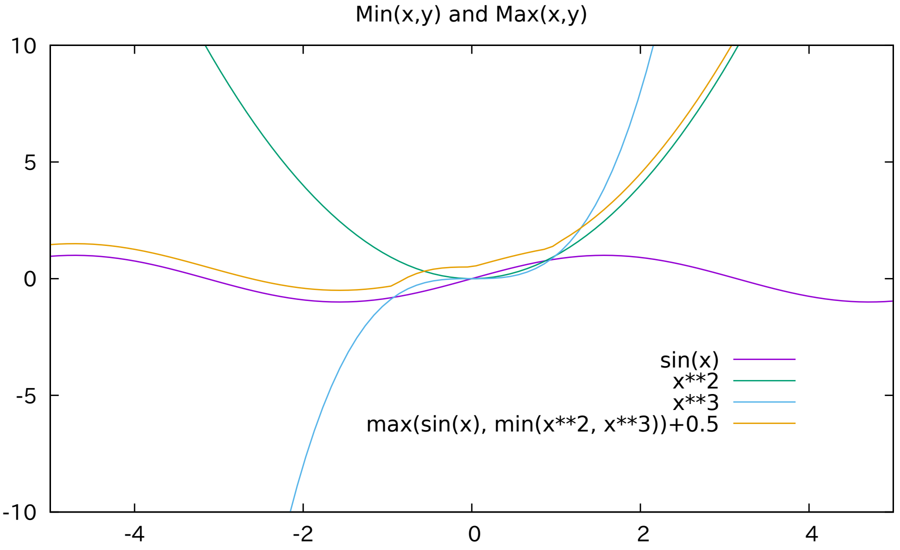

approximate the integral of functions
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-4
-2
0
2
4
f(x)=cos(x)
integral_f(x)


approximate the integral of functions (upper and lower limits)
150
100
50
0
-50
-100
-150
-200
-250
-300
-350
-400
-10
-5
0
5
10
f(x)=(x-2)**2-20
integral2_f(-5,x)

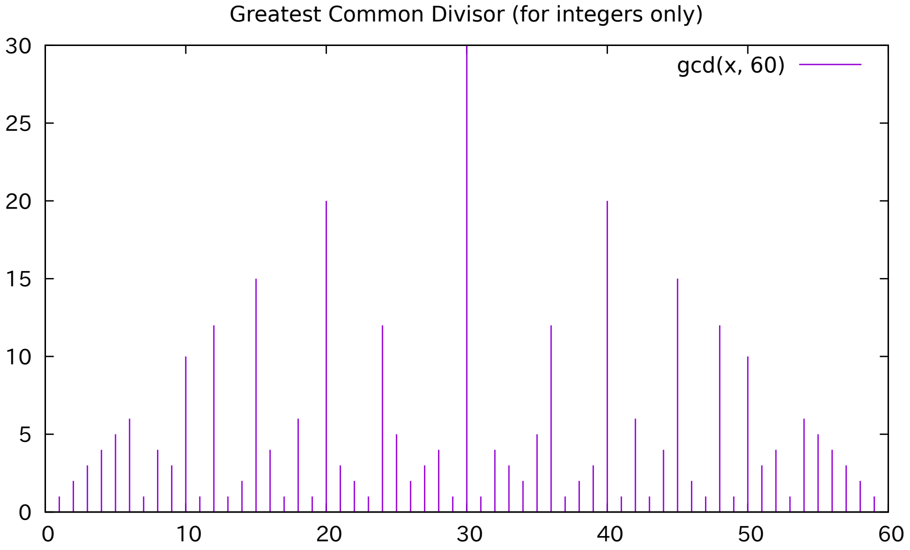

approximate the integral of functions (upper and lower limits)
20
15
10
5
0
-5
-10
-15
-20
-25
-30
-10
-5
0
5
10
f(x)=sin(x-1)-0.75*sin(2*x-1)+(x**2)/8-5
integral2_f(x,1)

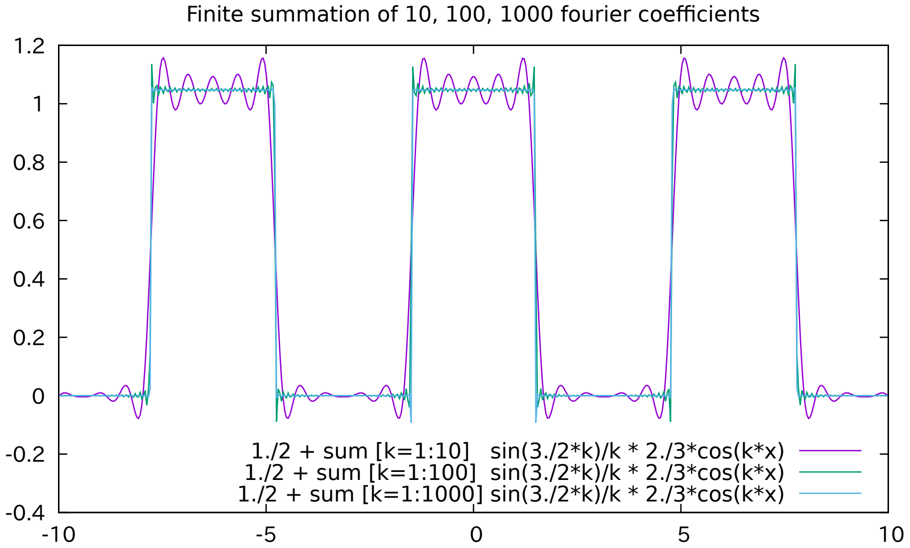
Plot of the ackermann function



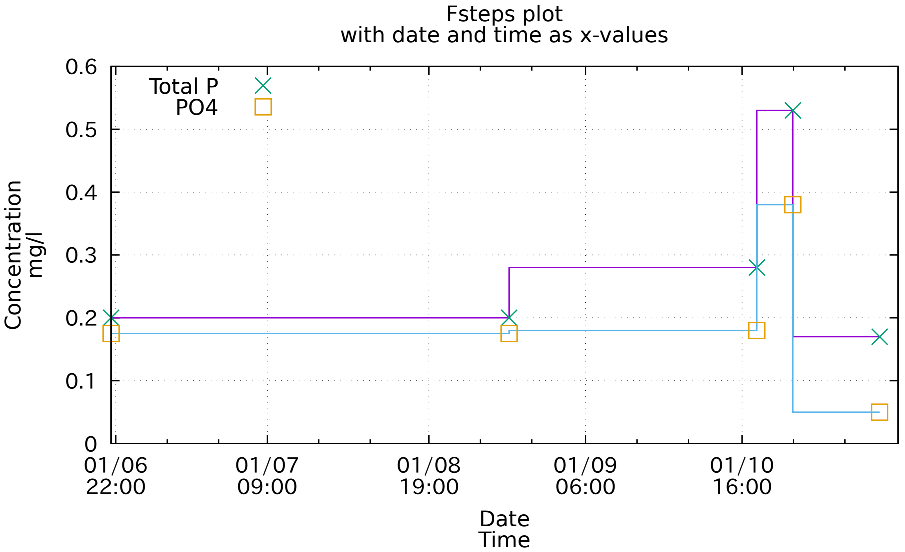


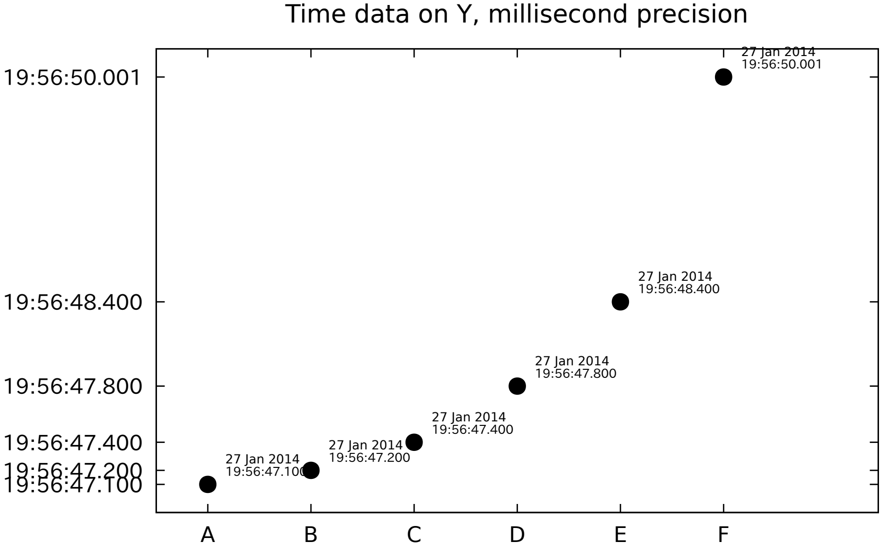
Min(x,y) and Max(x,y)
10
5
0
-5
-10
-4
-2
0
2
4
sin(x)
x**2
x**3
max(sin(x), min(x**2, x**3))+0.5


Greatest Common Divisor (for integers only)
gcd(x, 60)
0
5
10
15
20
25
30
0
10
20
30
40
50
60

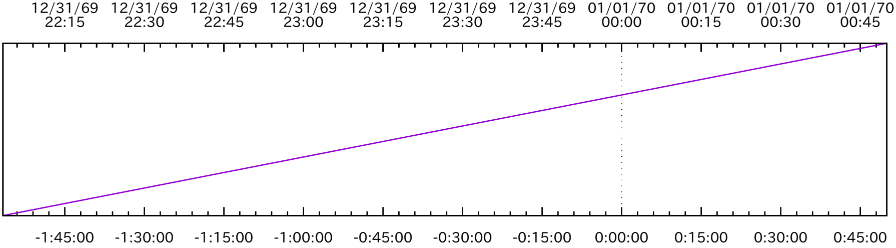


Finite summation of 10, 100, 1000 fourier coefficients
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
1./2 + sum [k=1:10] sin(3./2*k)/k * 2./3*cos(k*x)
1./2 + sum [k=1:100] sin(3./2*k)/k * 2./3*cos(k*x)
1./2 + sum [k=1:1000] sin(3./2*k)/k * 2./3*cos(k*x)


Fsteps plot
with date and time as x-values
Concentration
mg/l
0.6
0.5
0.4
0.3
0.2
0.1
0
Total P
PO4
01/06
22:00
01/07
09:00
01/08
19:00
01/09
06:00
01/10
16:00
Date
Time

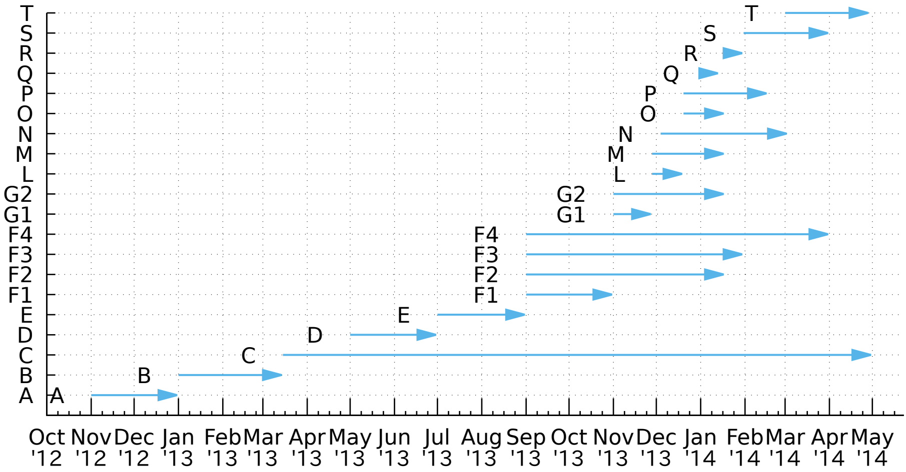

Time data on Y, millisecond precision
19:56:50.001
19:56:48.400
19:56:47.800
19:56:47.400
19:56:47.200
19:56:47.100
27 Jan 2014
19:56:47.100
27 Jan 2014
19:56:47.200
27 Jan 2014
19:56:47.400
27 Jan 2014
19:56:47.800
27 Jan 2014
19:56:48.400
27 Jan 2014
19:56:50.001
A
B
C
D
E
F

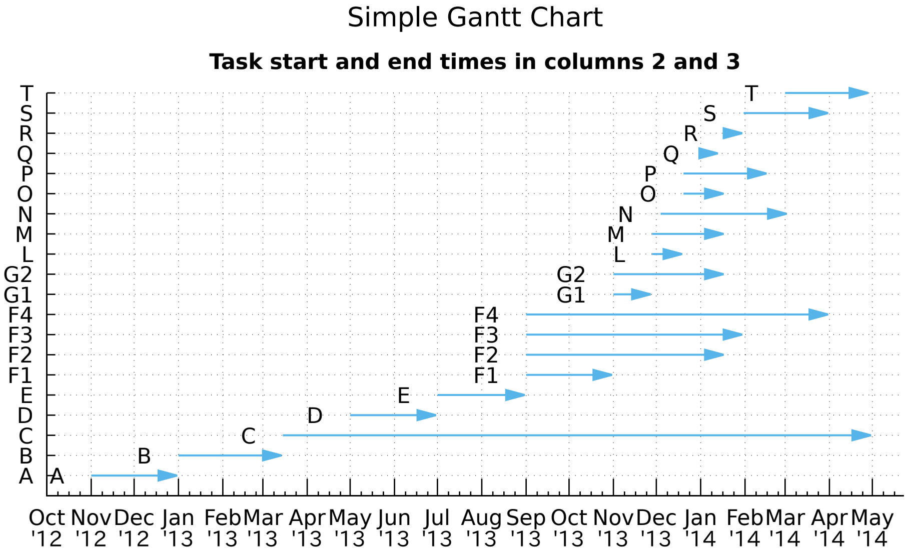
Date format (top) vs Time format (bottom)



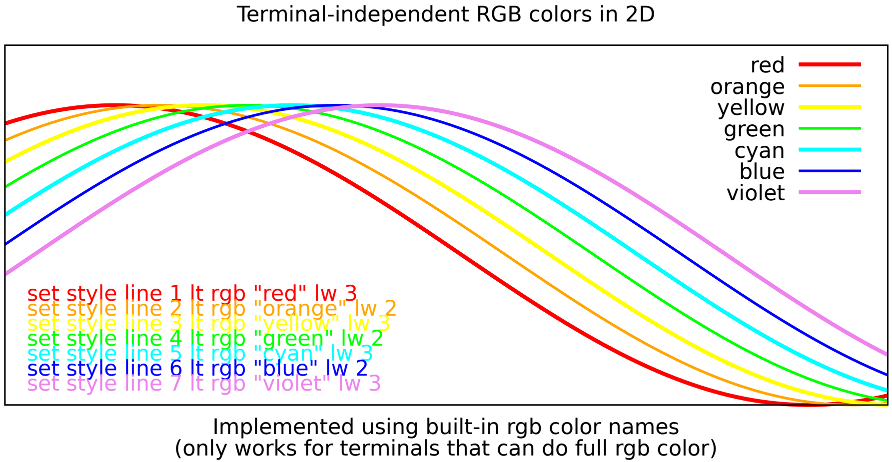
Simple Gantt Chart

Task start and end times in columns 2 and 3



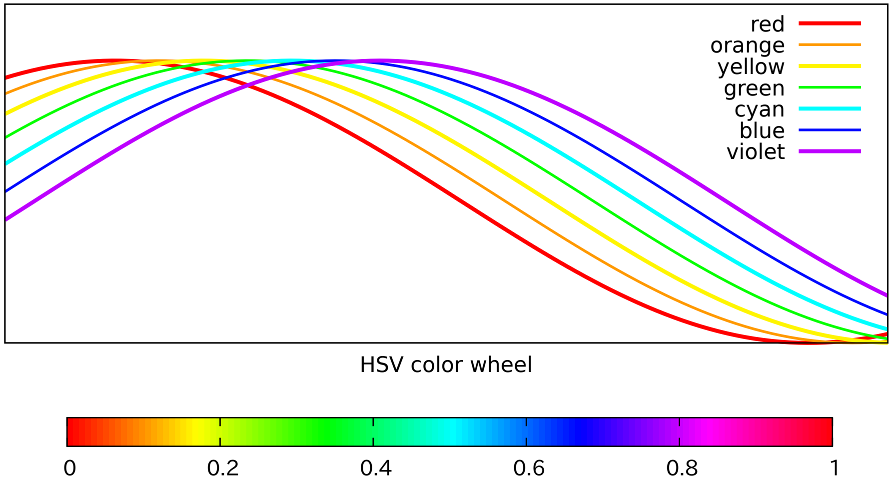


Simple Gantt Chart
Task start and end times in columns 2 and 3
T
S
R
Q
P
O
N
M
L
G2
G1
F4
F3
F2
F1
E
D
C
B
A
Oct '12
Nov '12
Dec '12
Jan '13
Feb '13
Mar '13
Apr '13
May '13
Jun '13
Jul '13
Aug '13
Sep '13
Oct '13
Nov '13
Dec '13
Jan '14
Feb '14
Mar '14
Apr '14
May '14

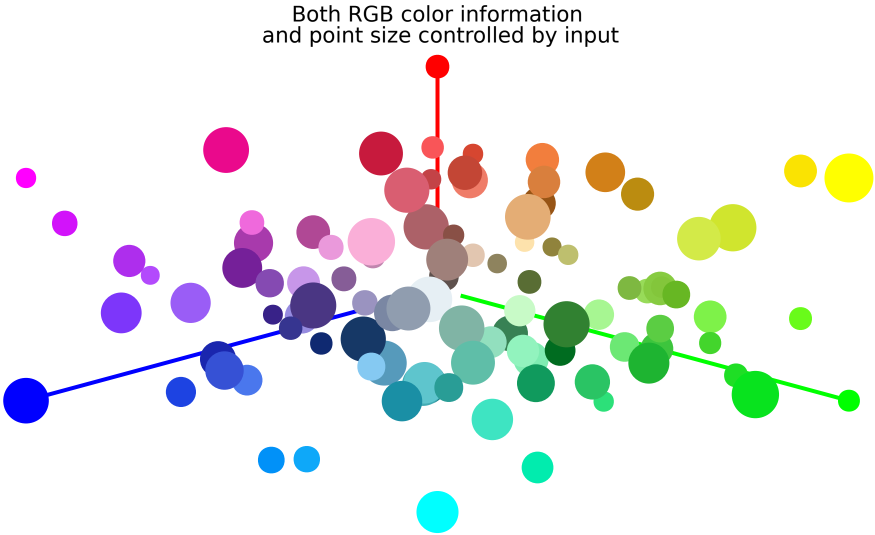


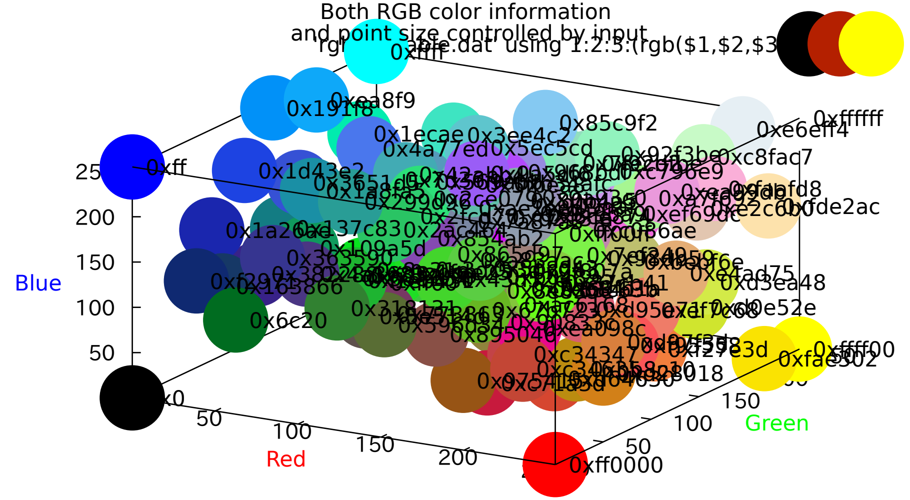
Terminal-independent RGB colors in 2D
red
orange
yellow
green
cyan
blue
violet
set style line 1 lt rgb "red" lw 3
set style line 2 lt rgb "orange" lw 2
set style line 3 lt rgb "yellow" lw 3
set style line 4 lt rgb "green" lw 2
set style line 5 lt rgb "cyan" lw 3
set style line 6 lt rgb "blue" lw 2
set style line 7 lt rgb "violet" lw 3
Implemented using built-in rgb color names
(only works for terminals that can do full rgb color)

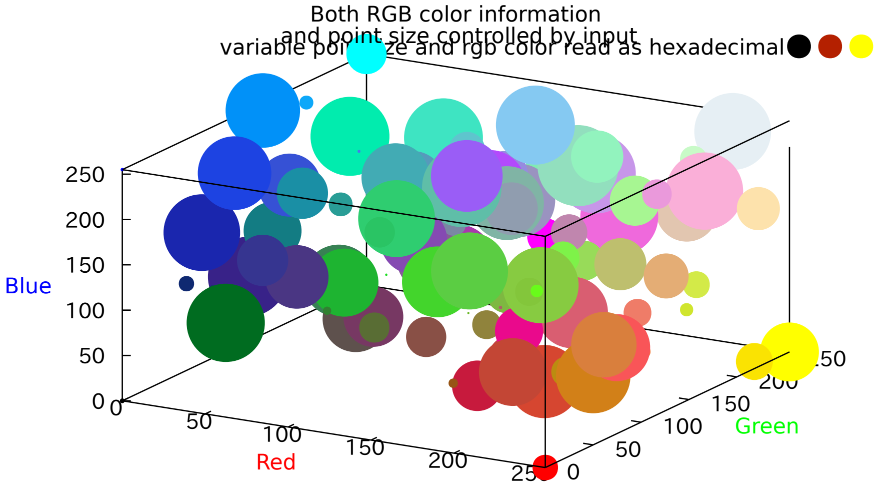
Terminal-independent palette colors in 2D
Implemented using command line macros referring to a fixed HSV palette



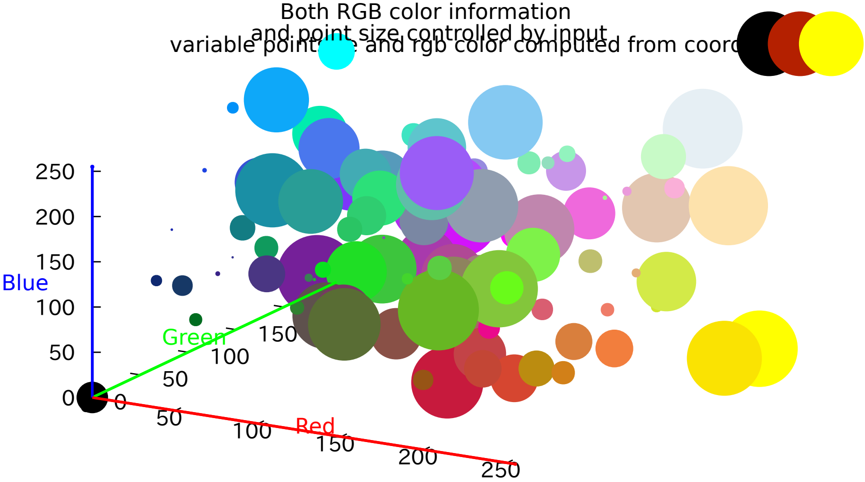

RGB color information read from data file

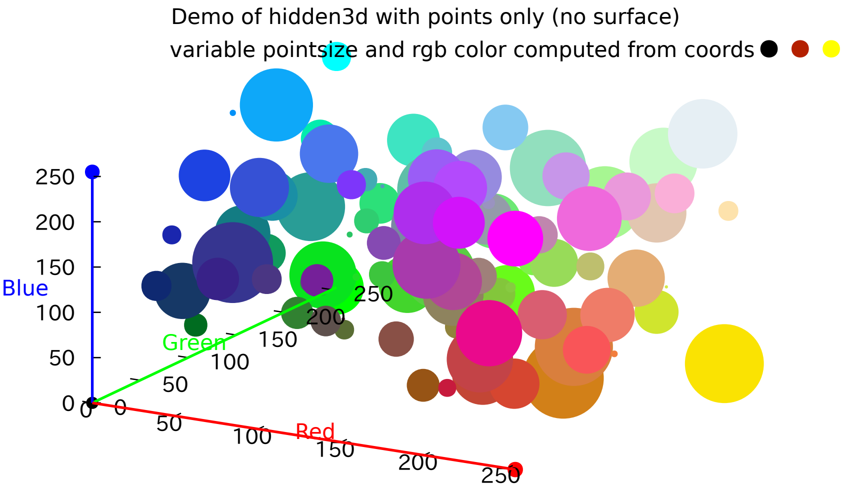

Both RGB color information
and point size controlled by input


Both RGB color information
and point size controlled by input
Blue
Red
Green
250
200
150
100
50
50
100
150
200
50
100
150
0xff
0x0
0xff0000
0xffffff
0xffff00
0xe6eff4
0x85c9f2
0xc8fac7
0xfde2ac
0xd3ea48
0xe4ad75
0x6c20
0xea8f9
0x191f8
0x1ecae
0x3ee4c2
0x4a77ed
0x5ec5cd
0x1d43e2
0x1a26ae
0x137c83
0x363590
0x163866
0xd0e52e
0xfae502
0xc34347
0xea098c



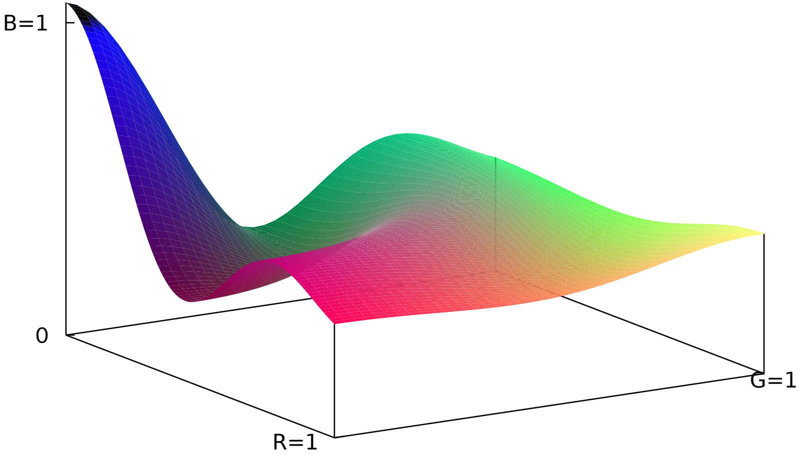
Both RGB color information
and point size controlled by input
variable point size and rgb color read as hexadecimal
Blue
Red
Green
0
50
100
150
200
250

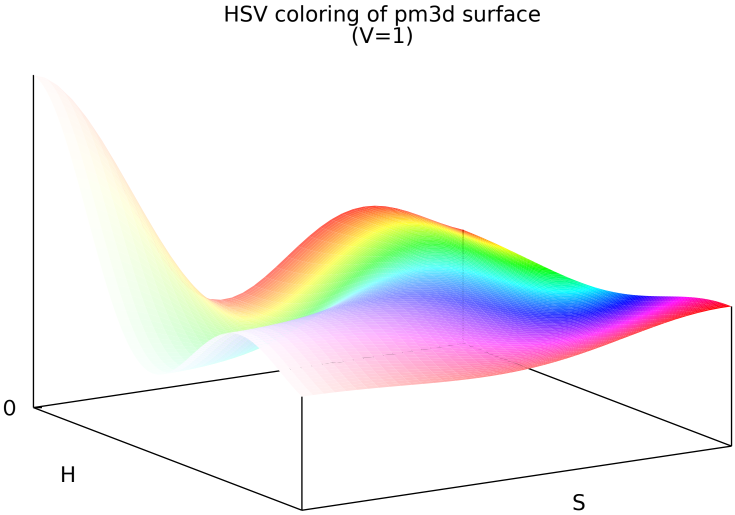

Both RGB color information
and point size controlled by input
Blue
0
50
100
150
200
250
Green
0
50
100
150
Red
0
50
100
150
200
250



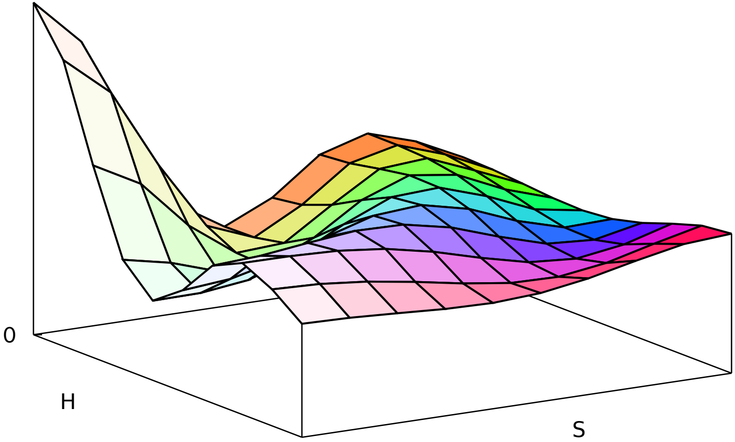
Demo of hidden3d with points only (no surface)
variable pointsize and rgb color computed from coords
Blue
250
200
150
100
50
0
Green
0
50
100
150
200
250
Red
0
50
100
150
200
250

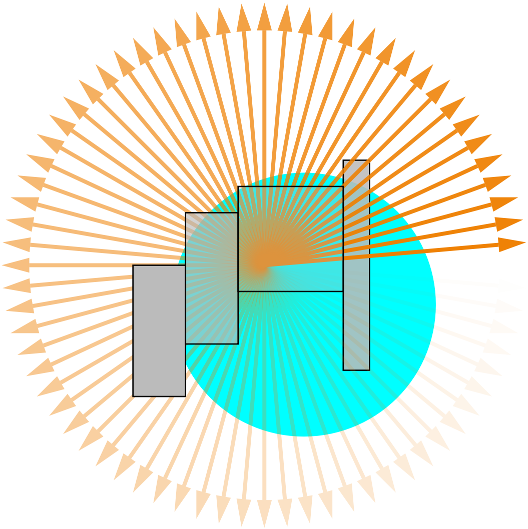
RGB coloring of pm3d surface

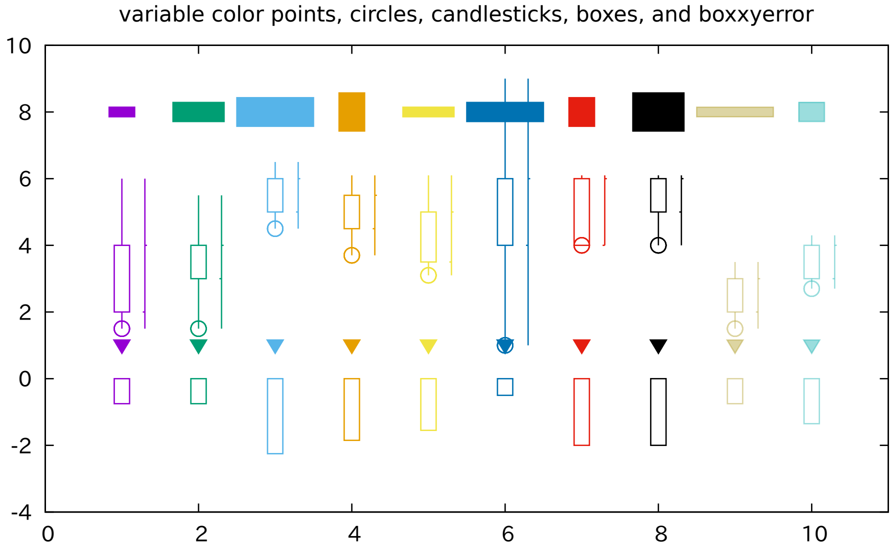


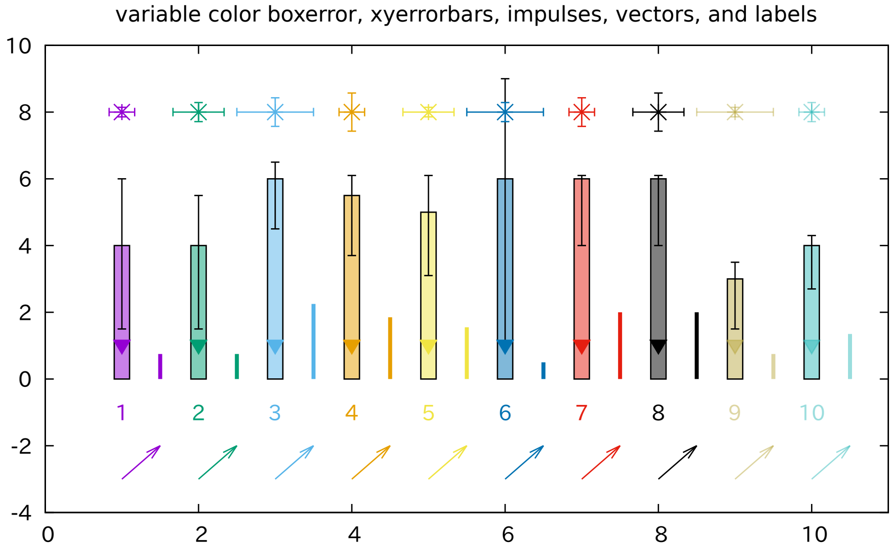

HSV coloring of pm3d surface
(V=1)
0
H
S

Explicit borders for pm3d tiling

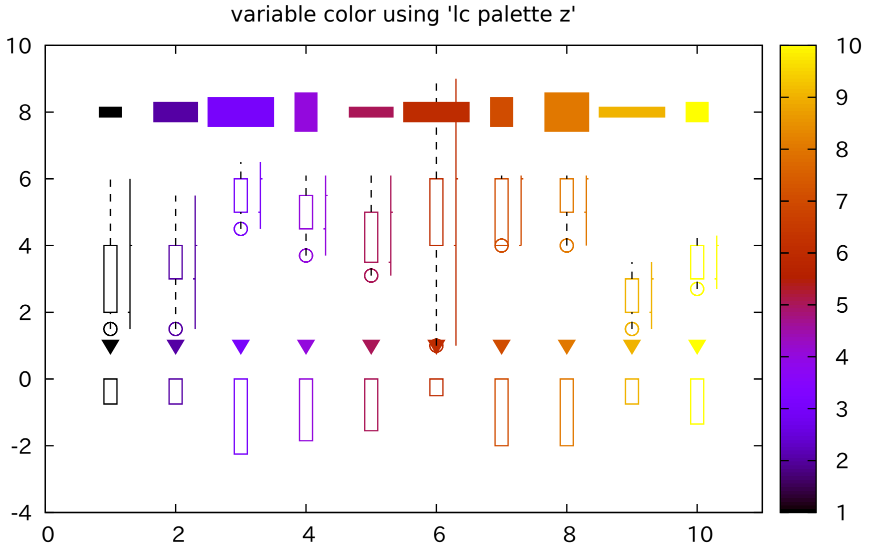


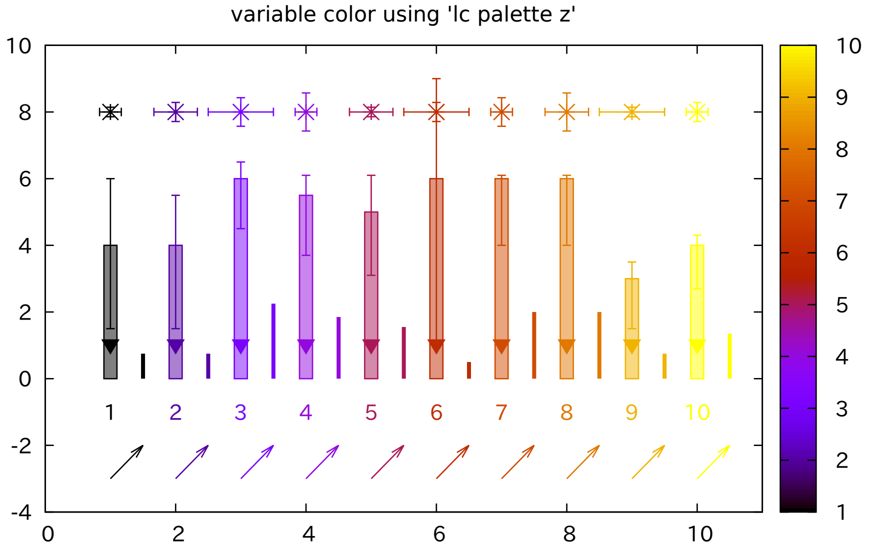



variable color points, circles, candlesticks, boxes, and boxxyerror
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10

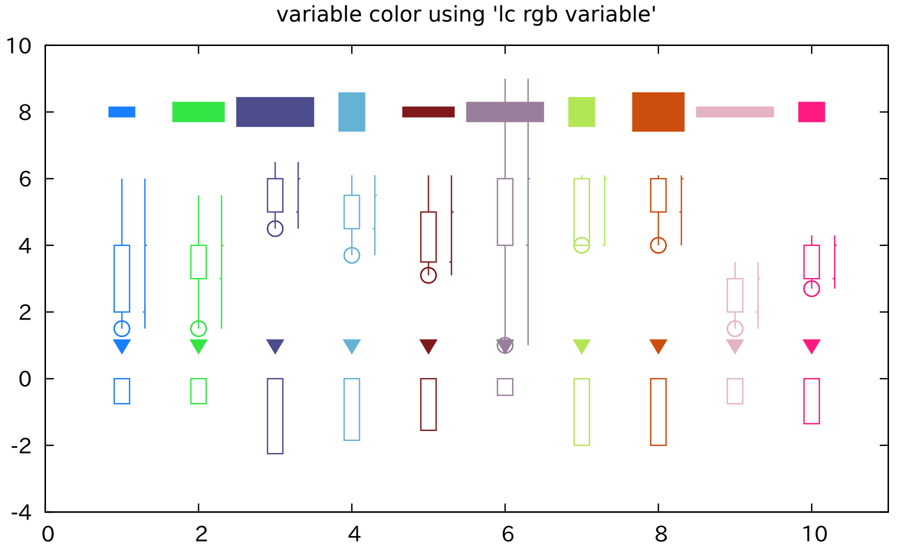


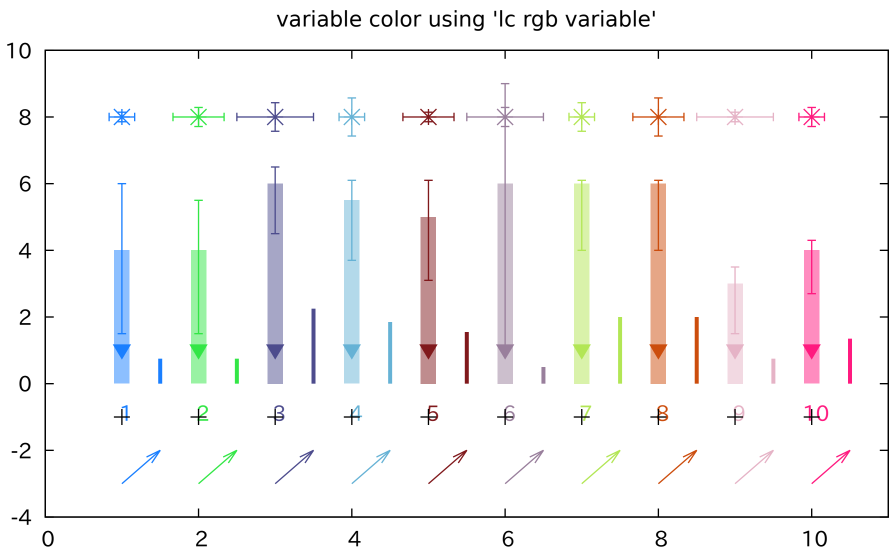
variable color boxerror, xyerrorbars, impulses, vectors, and labels
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10
1
2
3
4
5
6
7
8
9
10

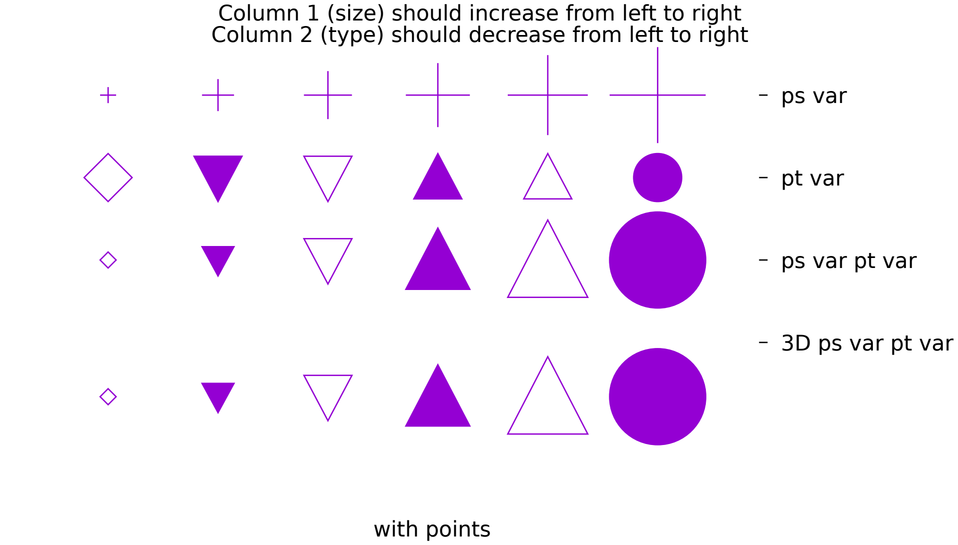

variable color using 'lc palette z'
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10
10
9
8
7
6
5
4
3
2
1

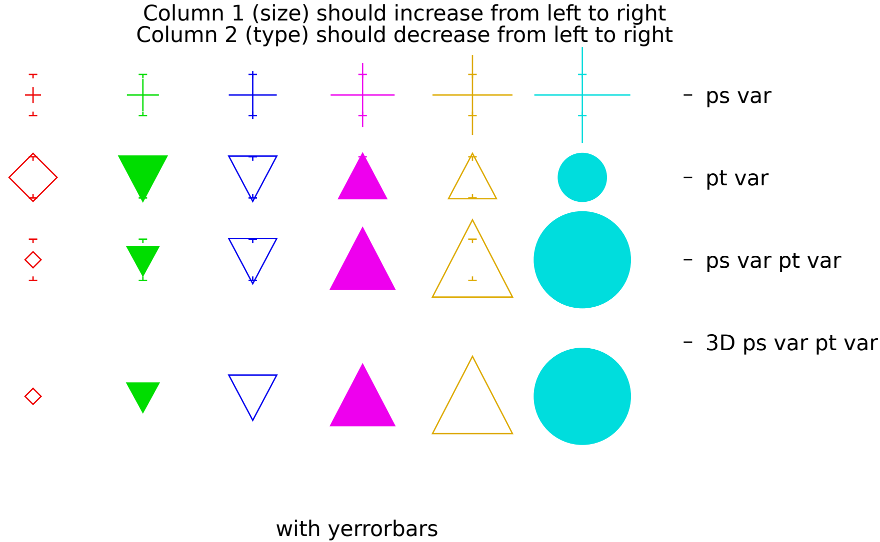


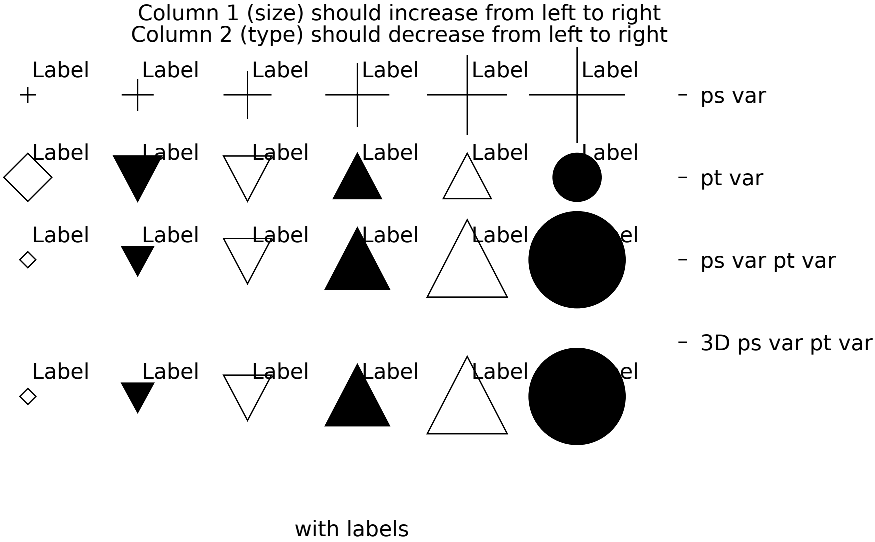
variable color using 'lc palette z'
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10
1
2
3
4
5
6
7
8
9
10
10
9
8
7
6
5
4
3
2
1

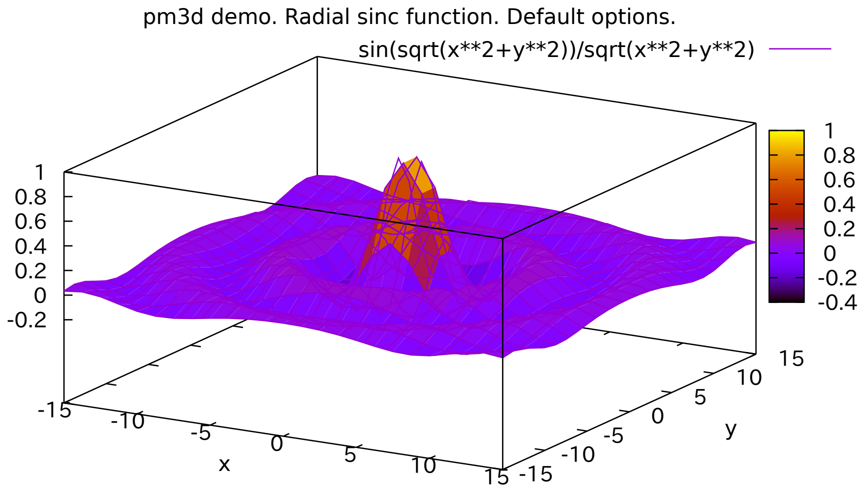

variable color using 'lc rgb variable'
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10

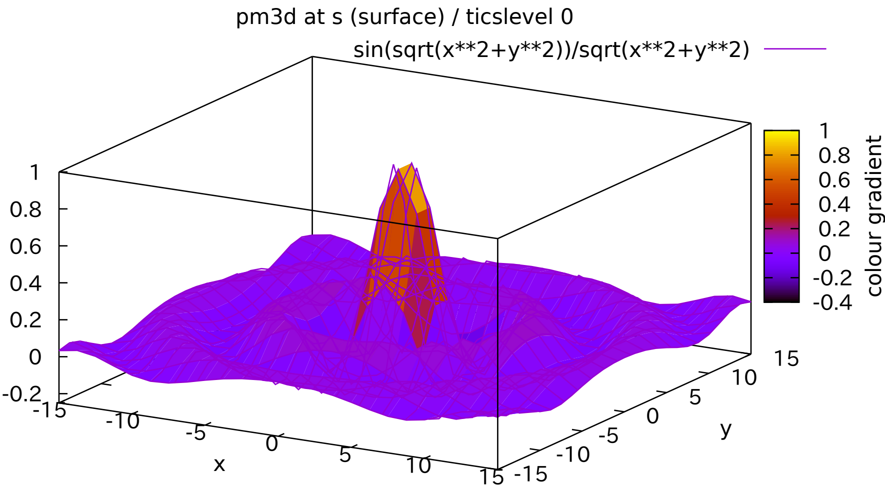

variable color using 'lc rgb variable'
10
8
6
4
2
0
-2
-4
0
2
4
6
8
10
1
2
3
4
5
6
7
8
9
10

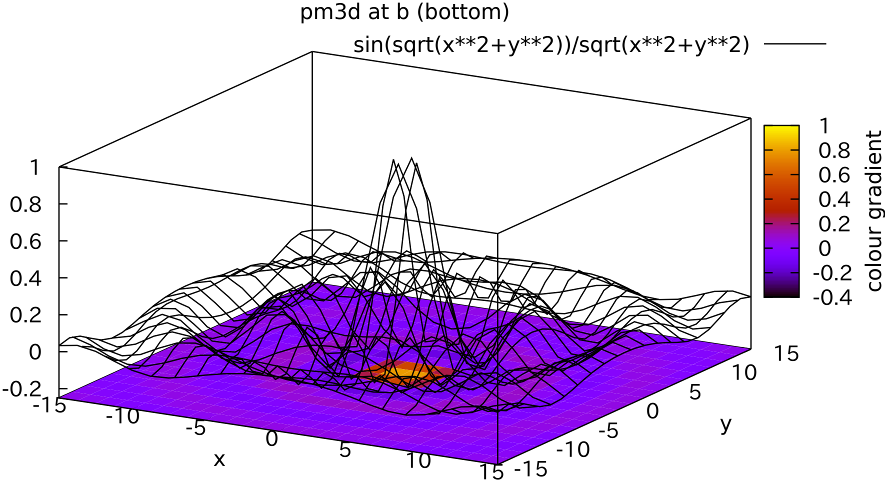

Column 1 (size) should increase from left to right
Column 2 (type) should decrease from left to right
ps var
pt var
ps var pt var
3D ps var pt var
with points

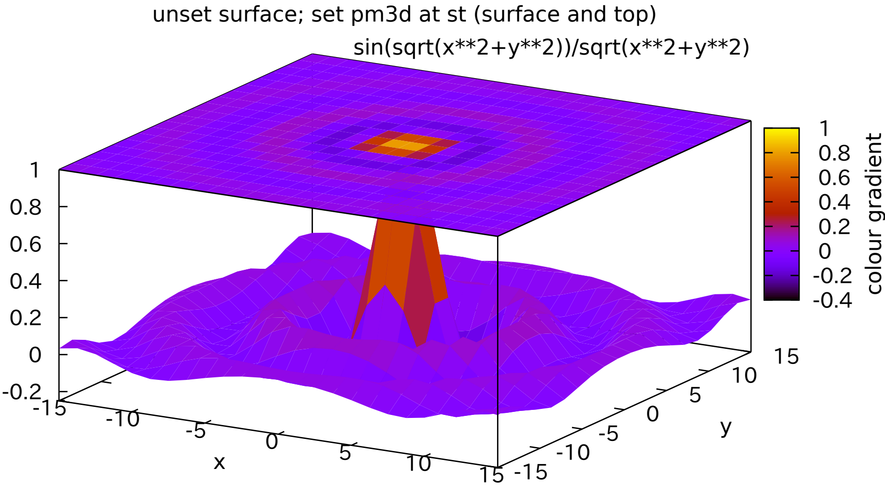

Column 1 (size) should increase from left to right
Column 2 (type) should decrease from left to right
ps var
pt var
ps var pt var
3D ps var pt var
with yerrorbars


Column 1 (size) should increase from left to right
Column 2 (type) should decrease from left to right
Label
ps var
pt var
ps var pt var
3D ps var pt var
with labels

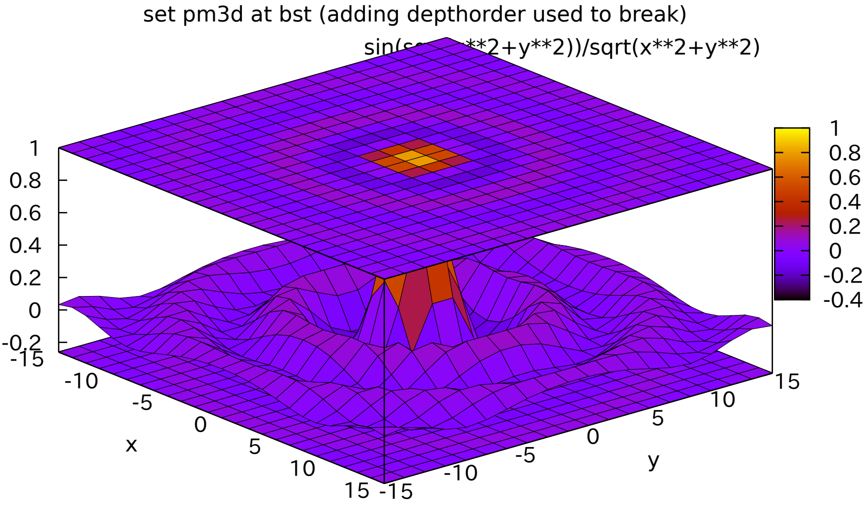

pm3d demo. Radial sinc function. Default options.
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15
x
y

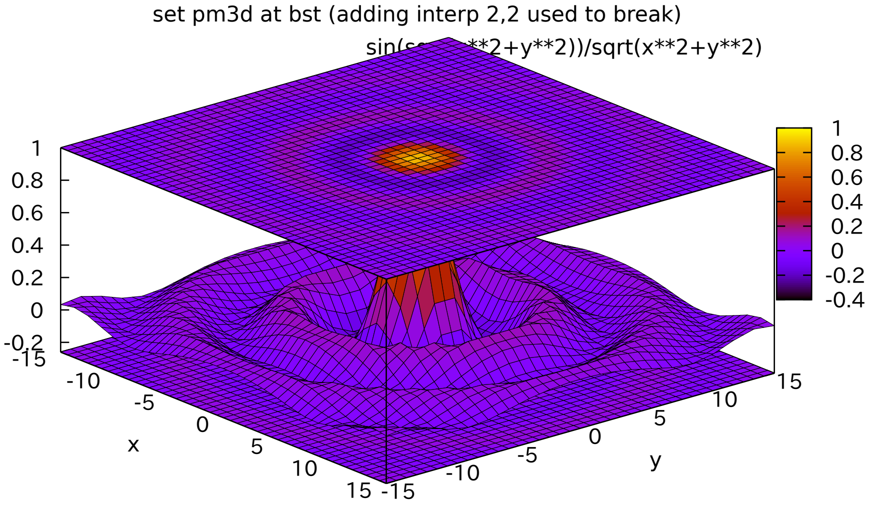

pm3d at s (surface) / ticslevel 0
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-15
-10
-5
0
5
10
15
x
-15
-10
-5
0
5
10
15
y
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
colour gradient

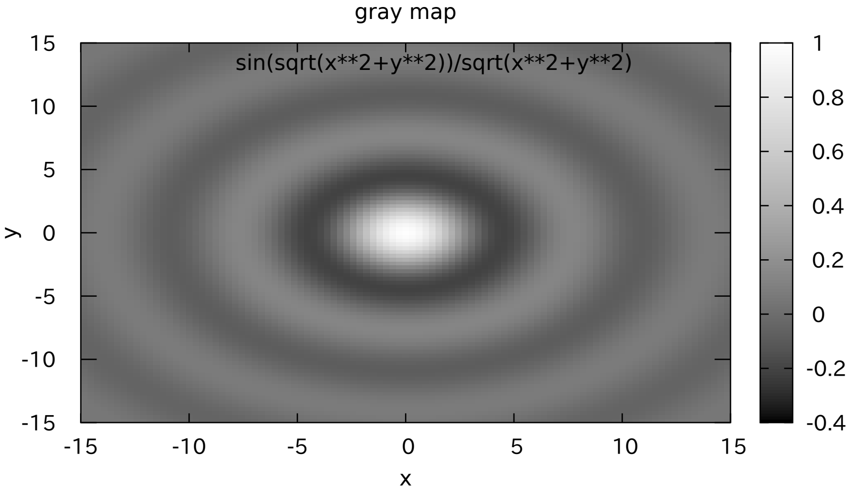

pm3d at b (bottom)
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-15
-10
-5
0
5
10
15
x
-15
-10
-5
0
5
10
15
y
colour gradient
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

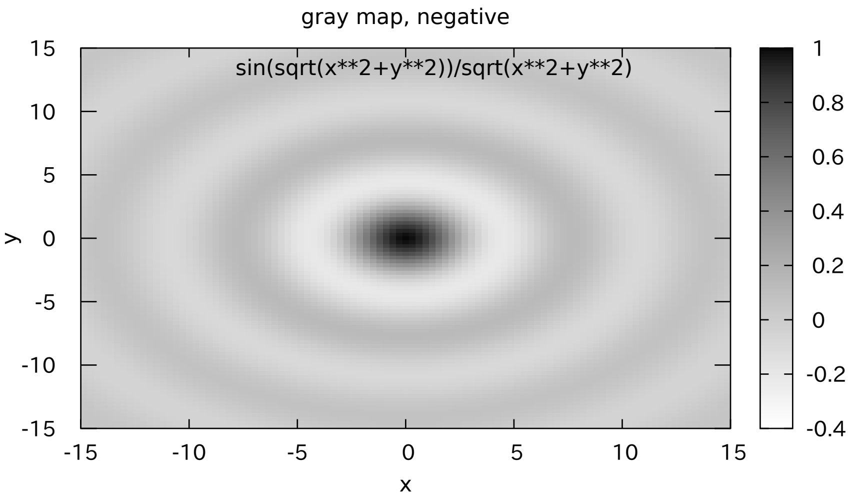


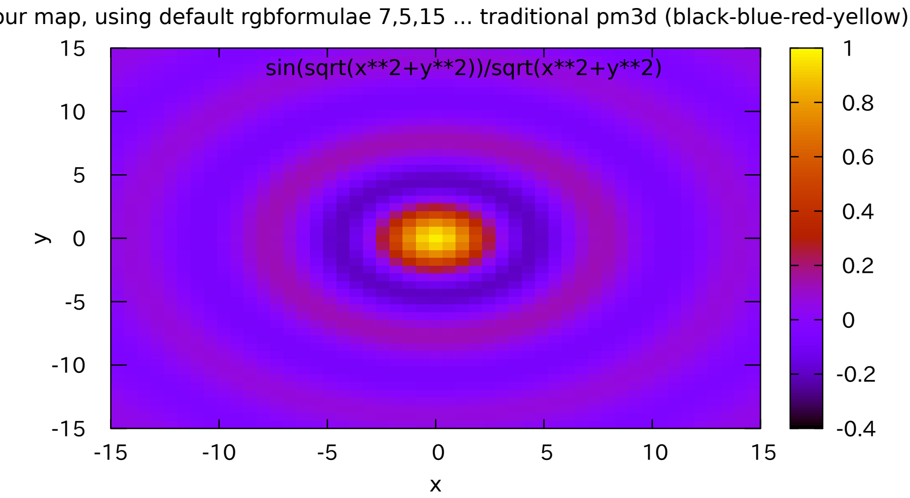
unset surface; set pm3d at st (surface and top)
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-15
-10
-5
0
5
10
15
x
-15
-10
-5
0
5
10
15
y
colour gradient
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

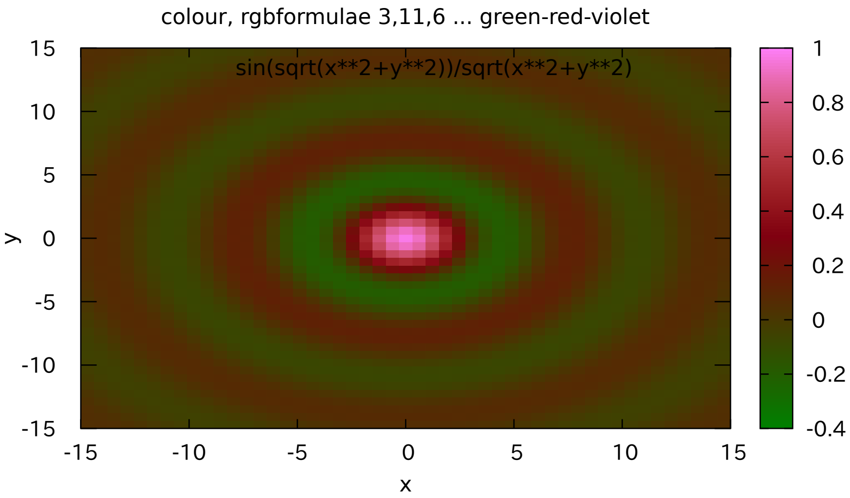

set pm3d at bst (adding depthorder used to break)
x
y
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
5
10
15

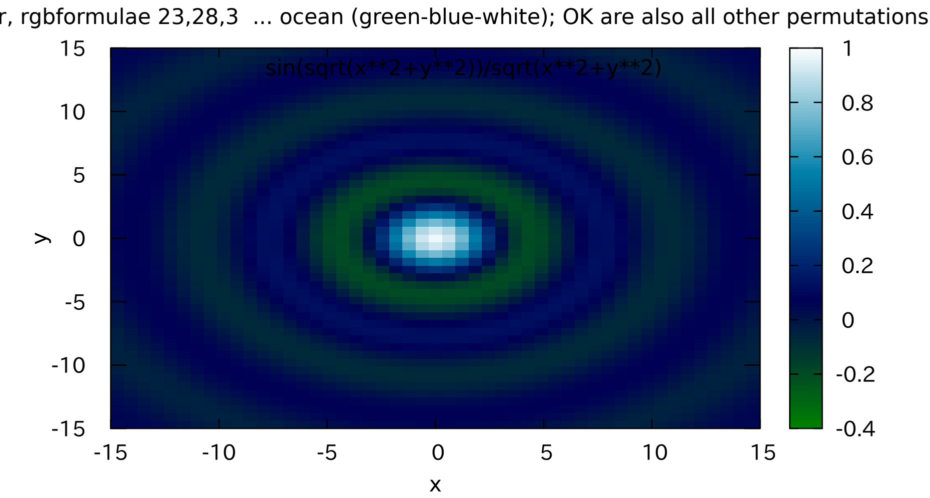

set pm3d at bst (adding interp 2,2 used to break)
2+y2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15
x
y


gray map
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

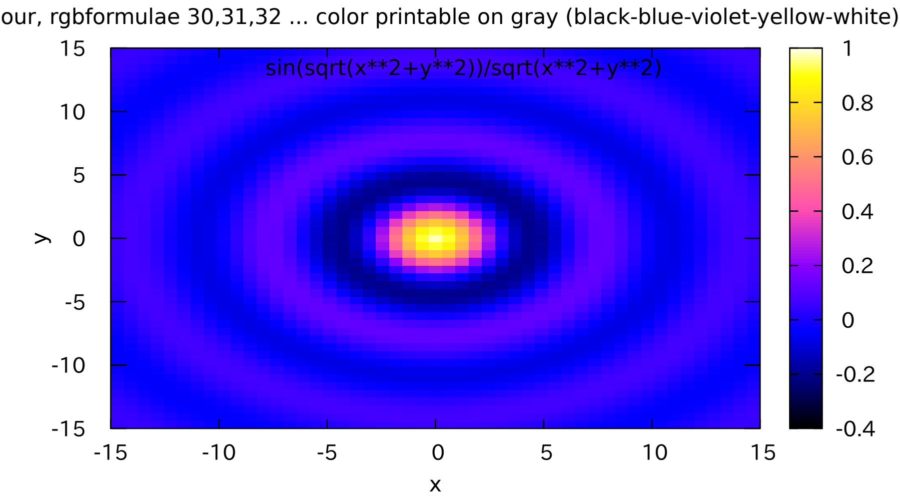

gray map, negative
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4


our map, using default rgbformulae 7,5,15 ... traditional pm3d (black-blue-red-yellow)
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

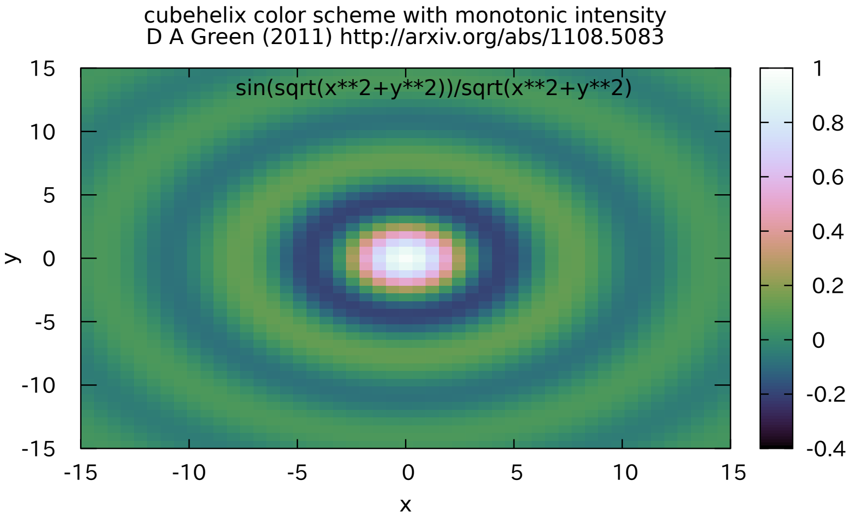

colour, rgbformulae 3,11,6 ... green-red-violet
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

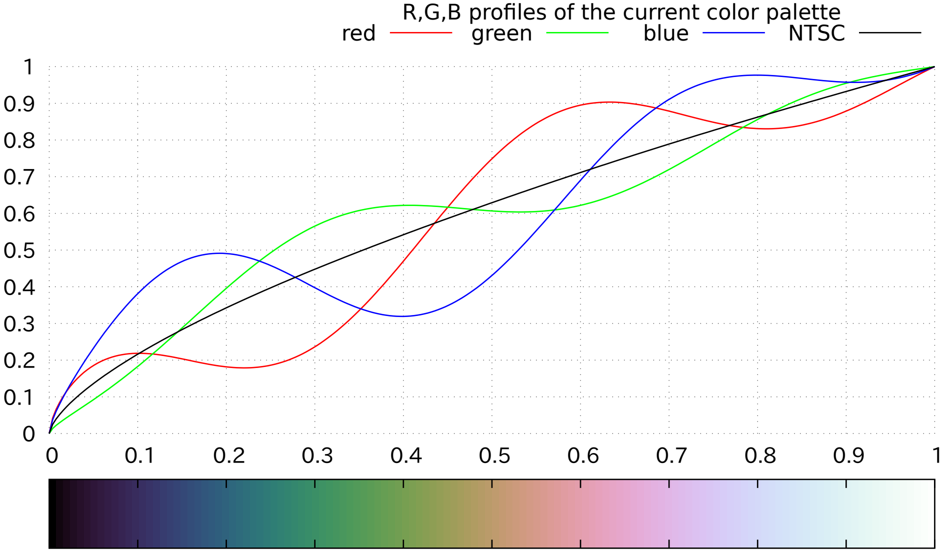


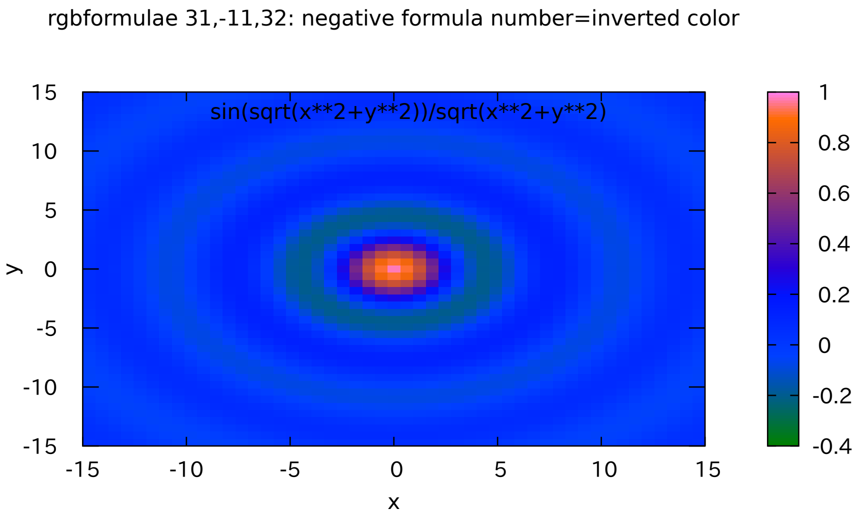
r, rgbformulae 23,28,3 ... ocean (green-blue-white); OK are also all other permutations
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

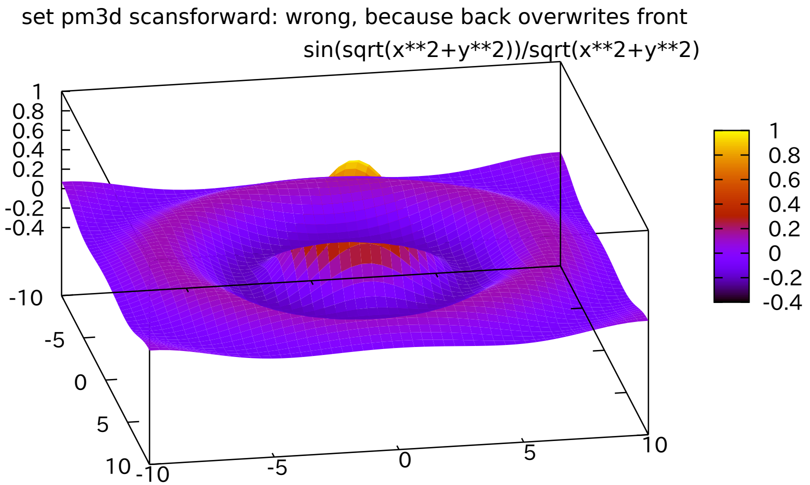


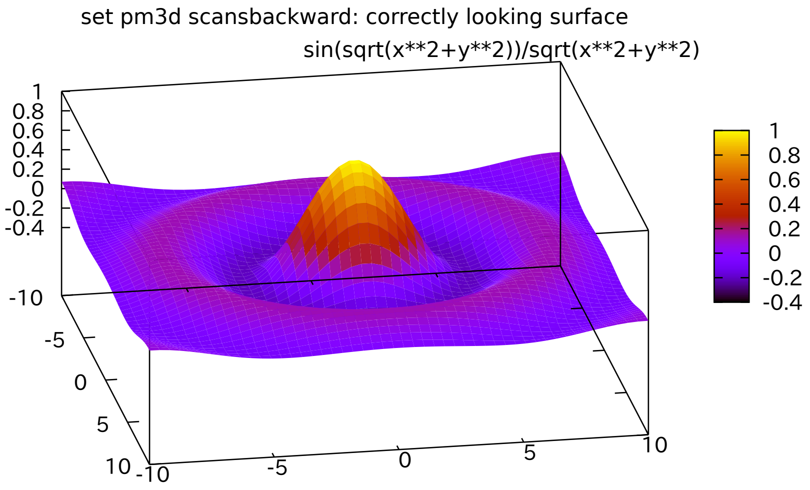
our, rgbformulae 30,31,32 ... color printable on gray (black-blue-violet-yellow-white)
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4


cubehelix color scheme with monotonic intensity
D A Green (2011) http://arxiv.org/abs/1108.5083
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

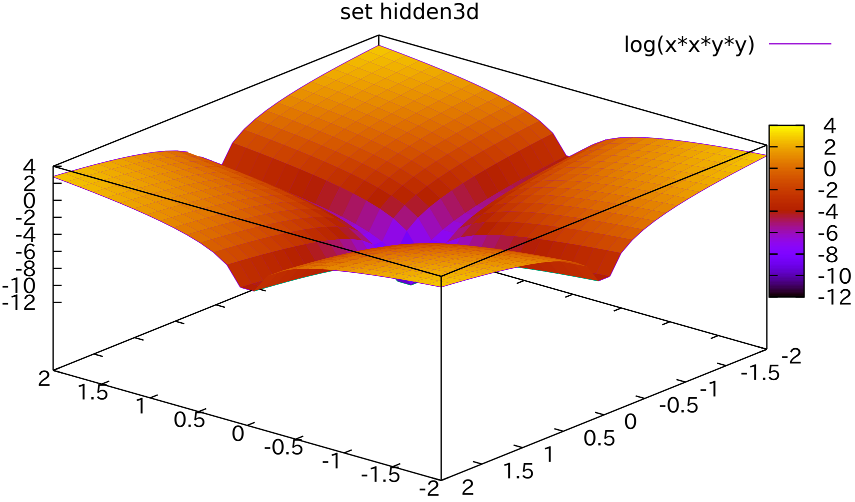

R,G,B profiles of the current color palette
red
green
blue
NTSC
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


rgbformulae 31,-11,32: negative formula number=inverted color
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
15
10
5
0
-5
-10
-15
y
-15
-10
-5
0
5
10
15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

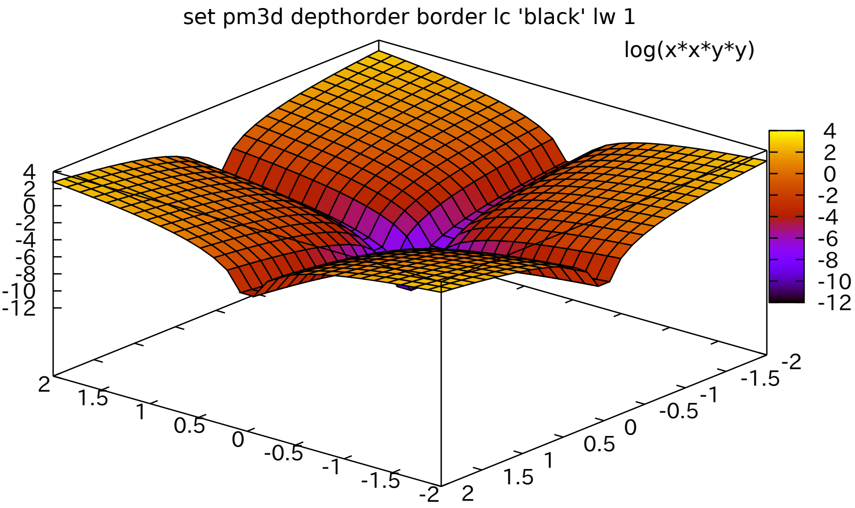

set pm3d scansforward: wrong, because back overwrites front
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
-10
-5
0
5
10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4



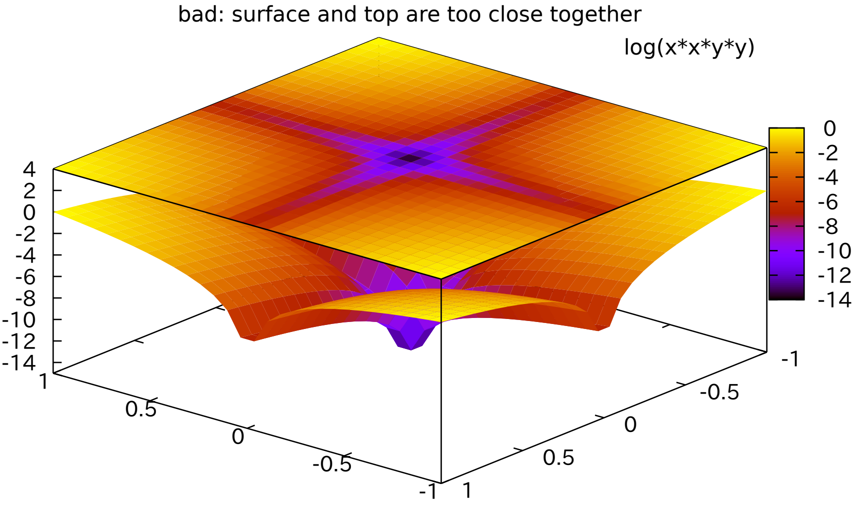
set pm3d scansbackward: correctly looking surface
sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
-10
-5
0
5
10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4



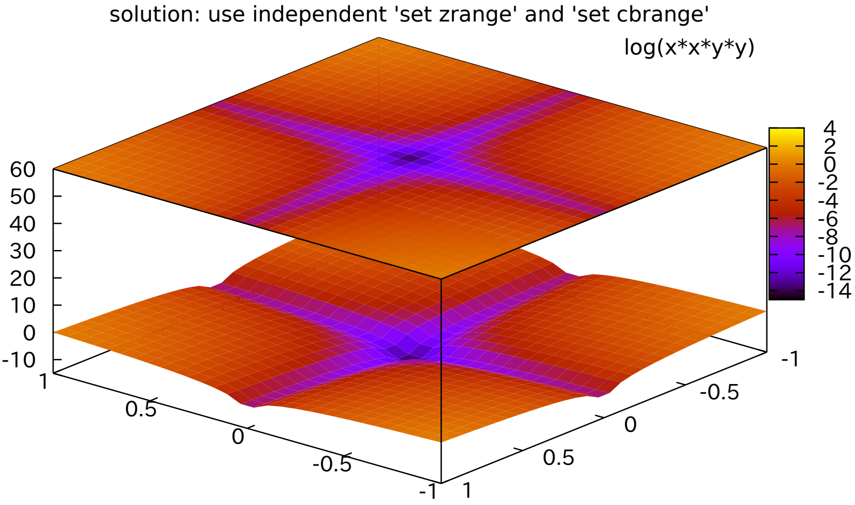
set hidden3d
log(x*x*y*y)
4
2
0
-2
-4
-6
-8
-10
-12
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2



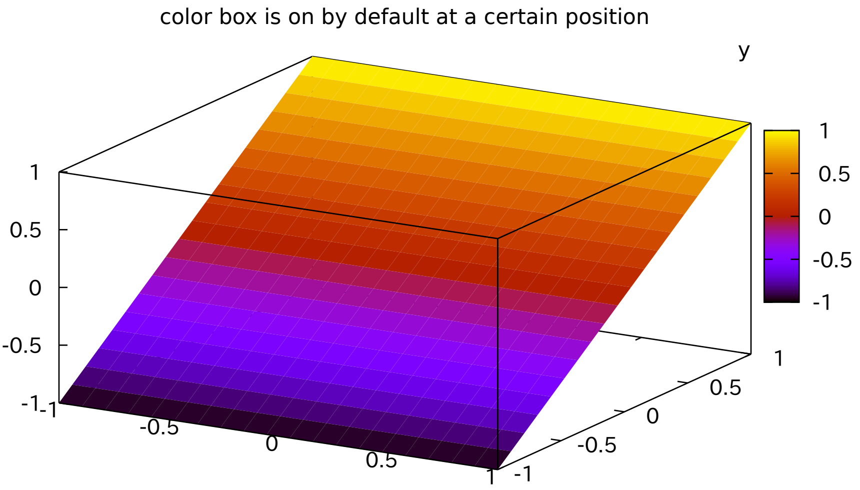
set pm3d depthorder border lc 'black' lw 1
log(x*x*y*y)
4
2
0
-2
-4
-6
-8
-10
-12
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2

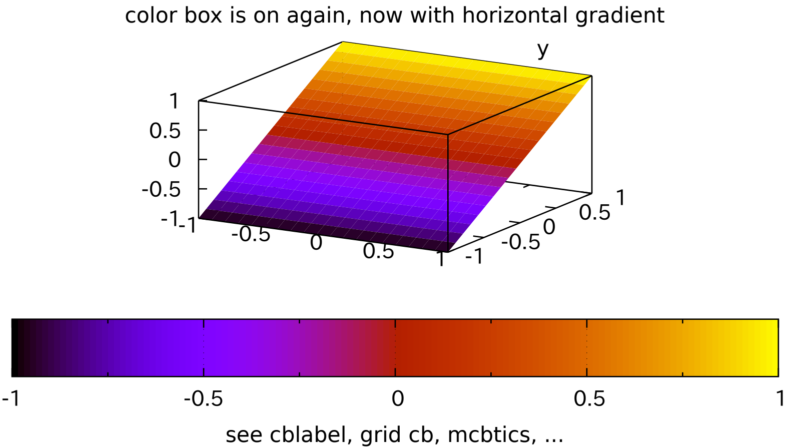


bad: surface and top are too close together
log(x*x*y*y)
4
2
0
-2
-4
-6
-8
-10
-12
-14
1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1
0
-2
-4
-6
-8
-10
-12
-14

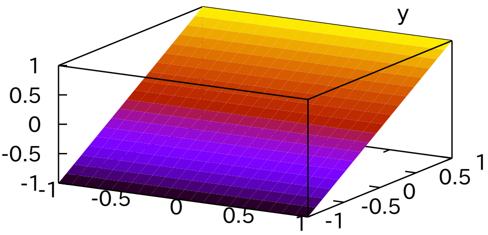


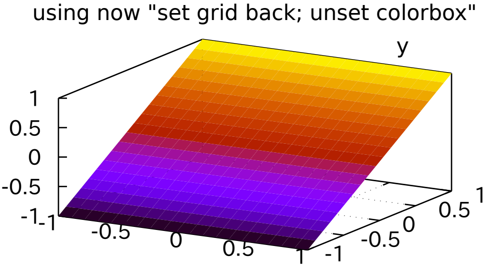
solution: use independent 'set zrange' and 'set cbrange'
log(x*x*y*y)
60
50
40
30
20
10
0
-10
1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1
4
2
0
-2
-4
-6
-8
-10
-12
-14

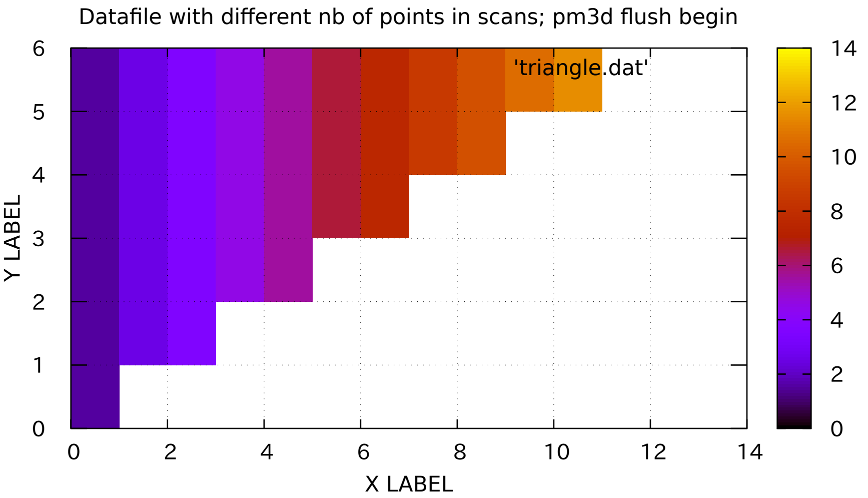

color box is on by default at a certain position
y
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1
1
0.5
0
-0.5
-1


color box is on again, now with horizontal gradient
y
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1
see cblabel, grid cb, mcbtics, ...

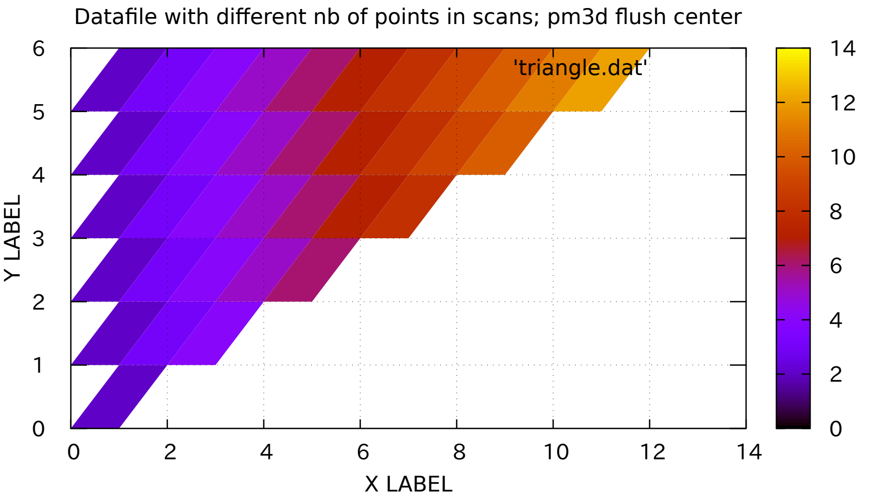
color box is switched off

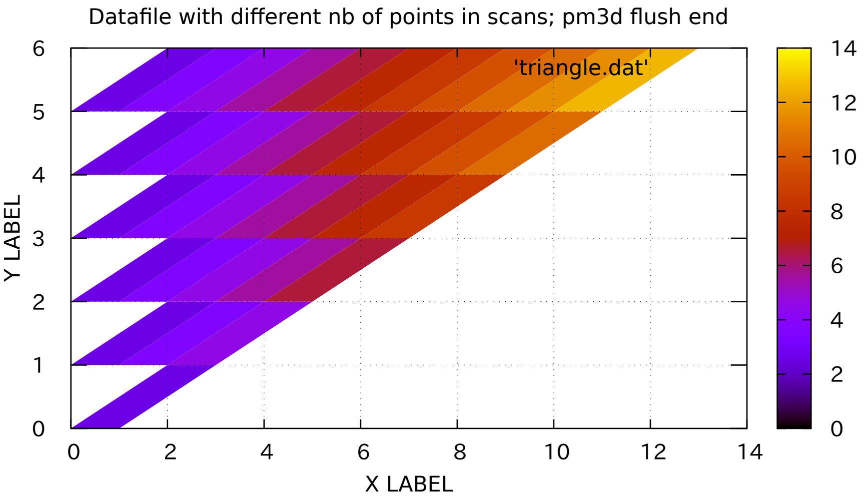


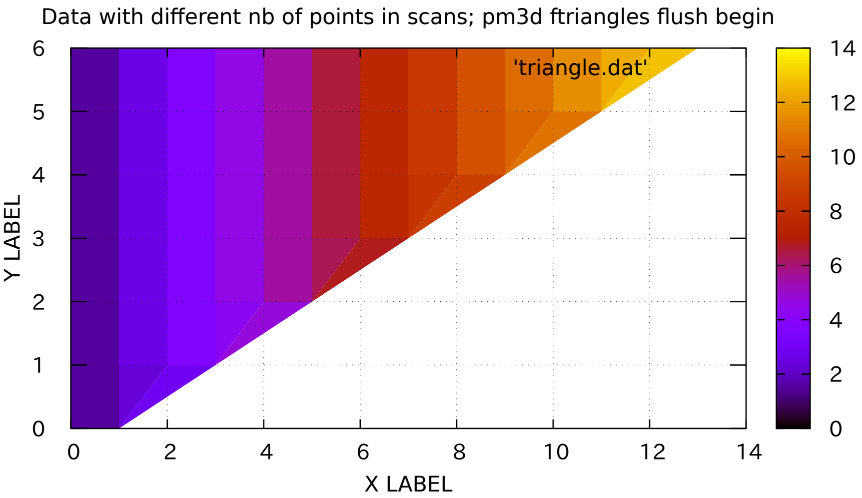

using now "set grid back; unset colorbox"
y
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1



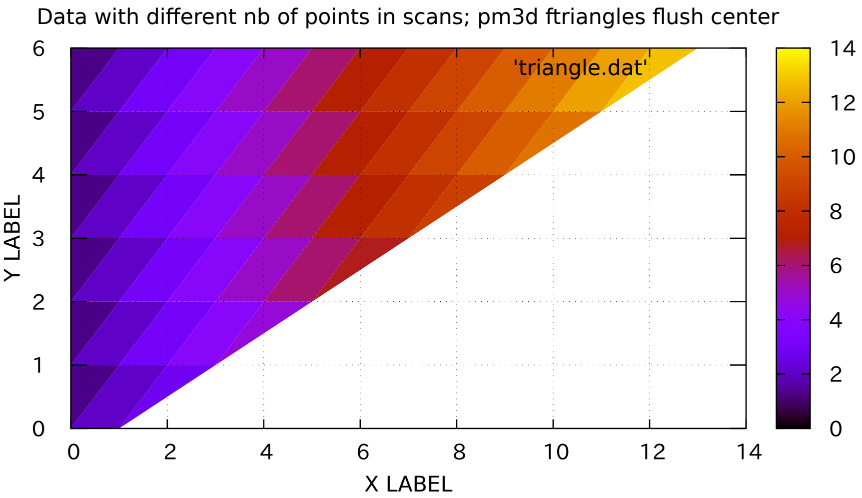
Datafile with different nb of points in scans; pm3d flush begin
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14


Datafile with different nb of points in scans; pm3d flush center
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

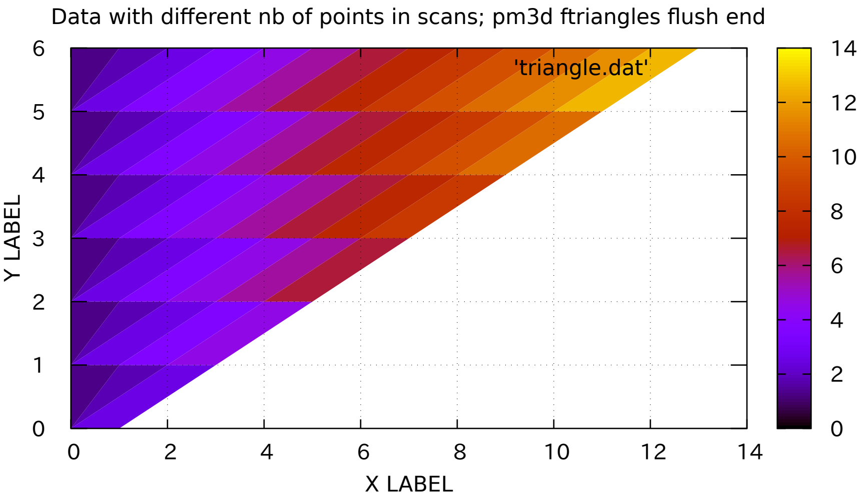

Datafile with different nb of points in scans; pm3d flush end
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

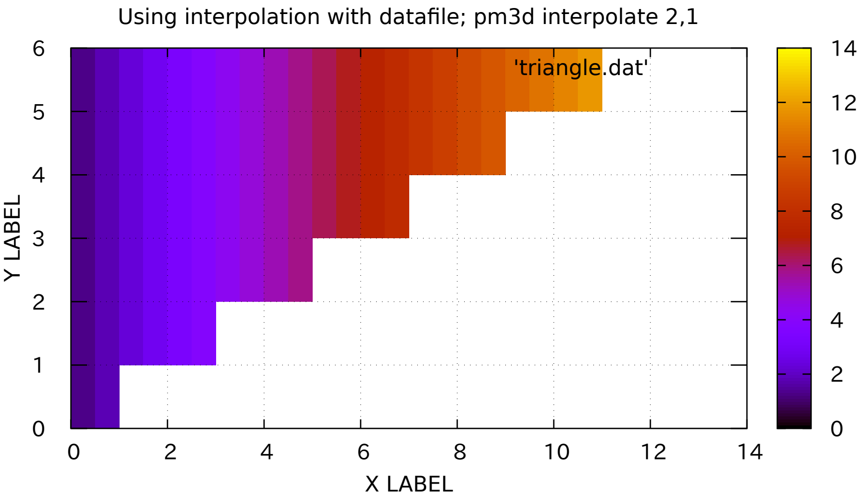


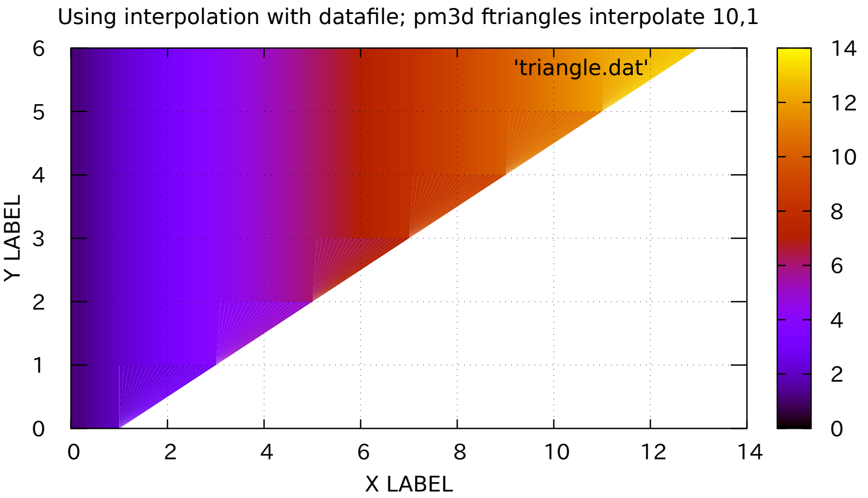
Data with different nb of points in scans; pm3d ftriangles flush begin
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

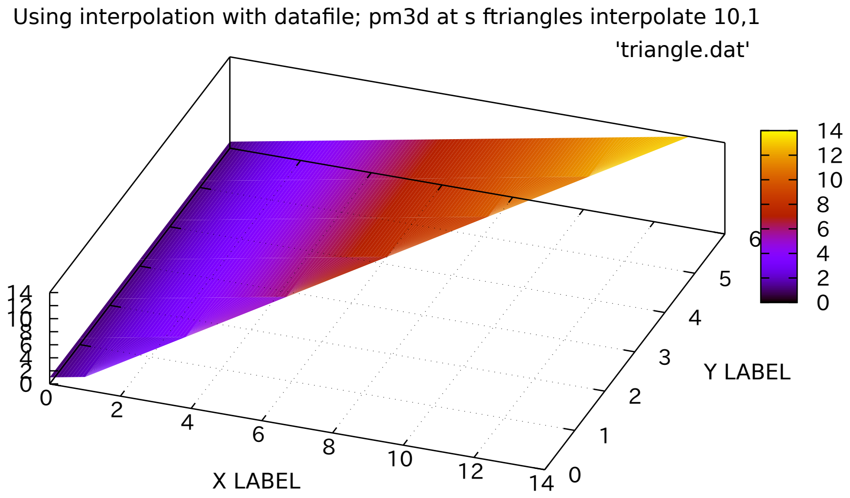


Data with different nb of points in scans; pm3d ftriangles flush center
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

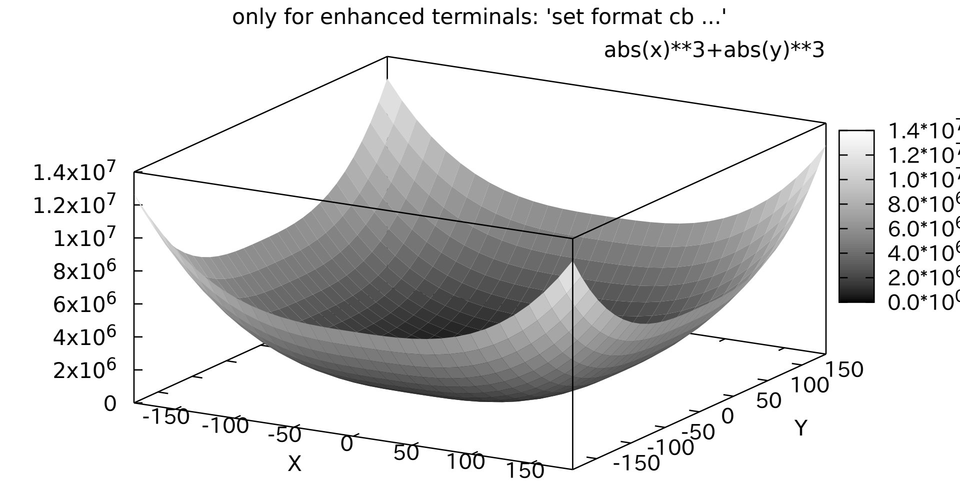

Data with different nb of points in scans; pm3d ftriangles flush end
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14


Using interpolation with datafile; pm3d interpolate 2,1
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14


Using interpolation with datafile; pm3d ftriangles interpolate 10,1
'triangle.dat'
Y LABEL
X LABEL
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14

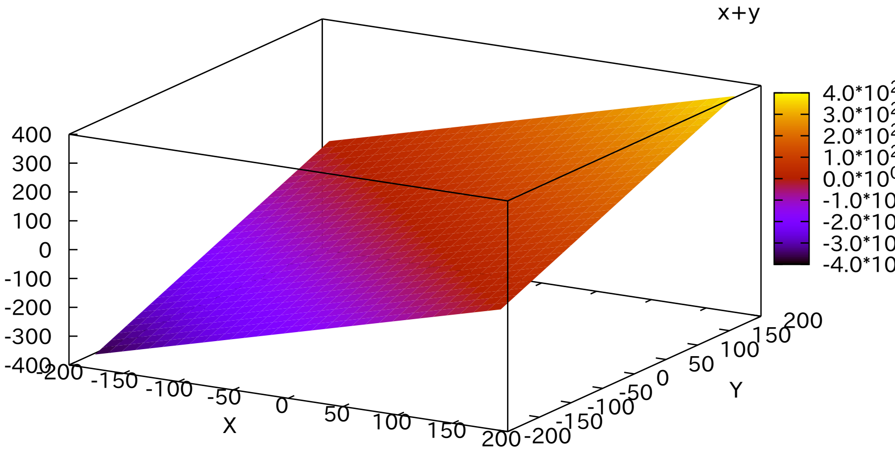


Using interpolation with datafile; pm3d at s ftriangles interpolate 10,1
'triangle.dat'
X LABEL
Y LABEL
0
2
4
6
8
10
12
14
0
1
2
3
4
5
6
0
2
4
6
8
10
12
14

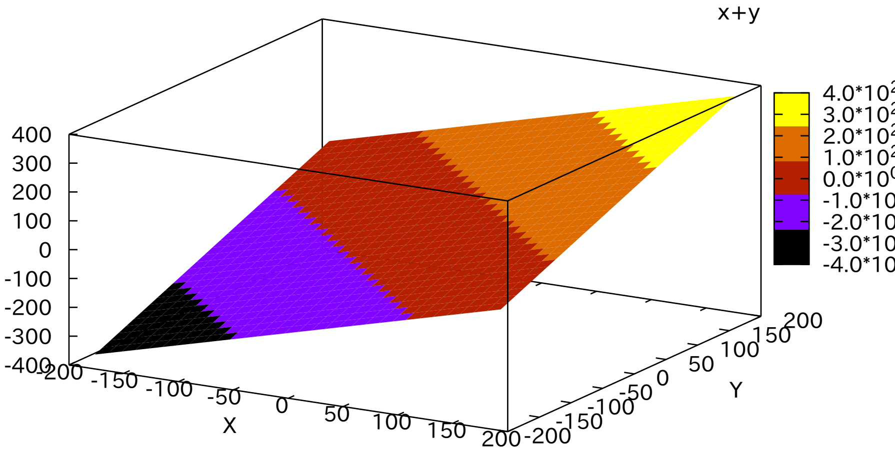


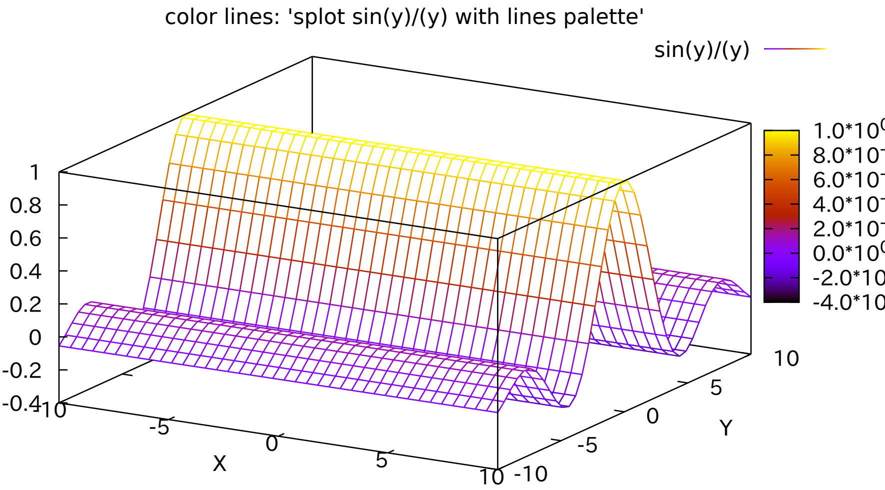
only for enhanced terminals: 'set format cb ...'
abs(x)**3+abs(y)**3
1.4x10^7
1.2x10^7
1x10^7
8x10^6
6x10^6
4x10^6
2x10^6
0
-150
-100
-50
0
50
100
150
X
Y
1.4*10^7
1.2*10^7
1.0*10^7
8.0*10^6
6.0*10^6
4.0*10^6
2.0*10^6
0.0*10^0

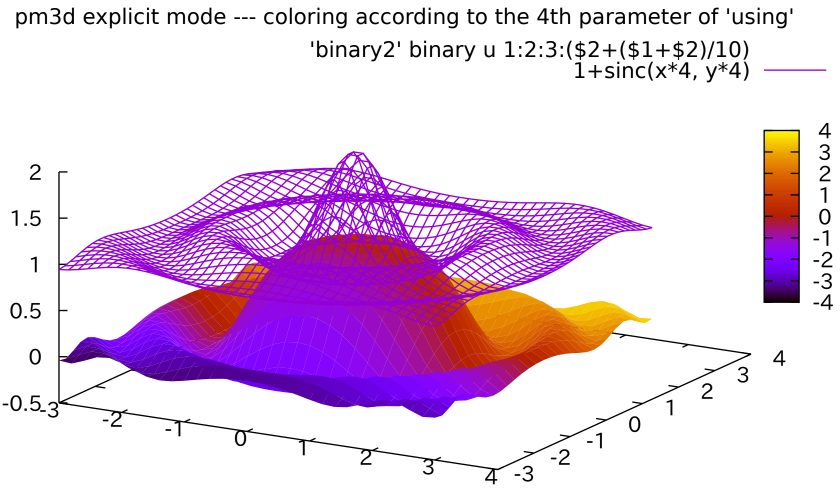
function 'x+y' using all colors available, 'set pal maxcolors 0'



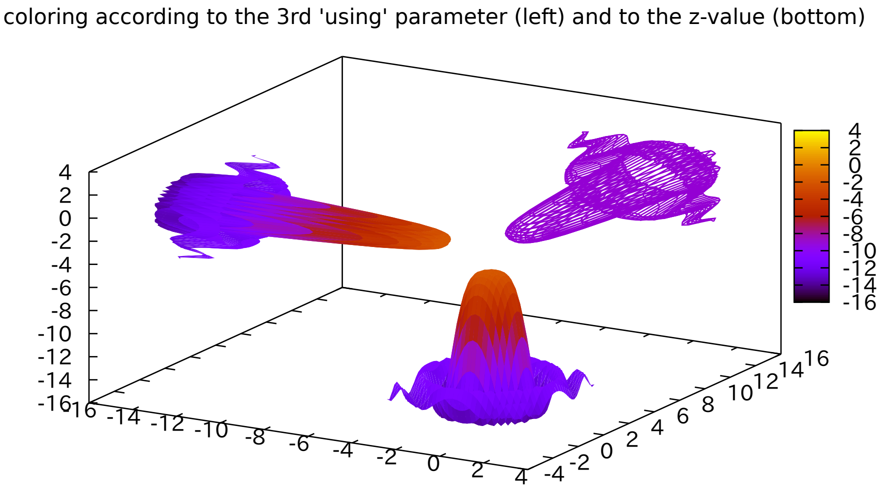
function 'x+y' using only 5 colors, 'set pal maxcolors 5'

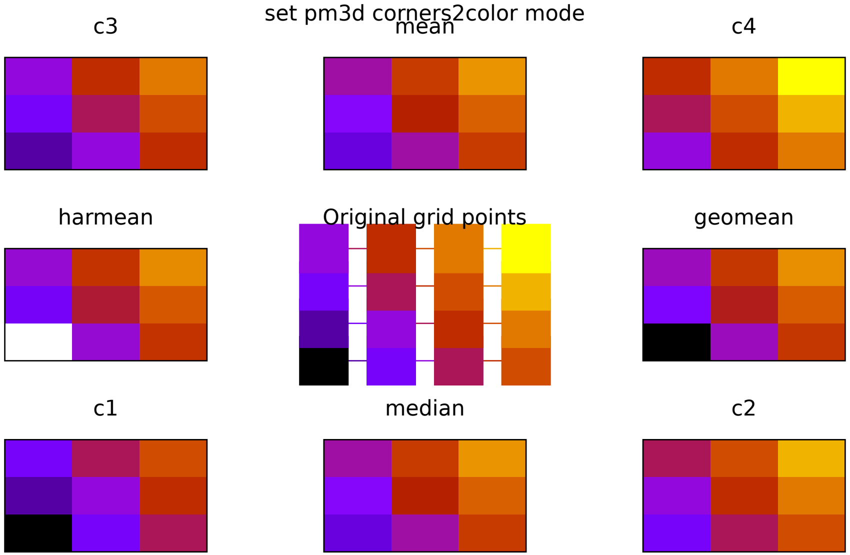


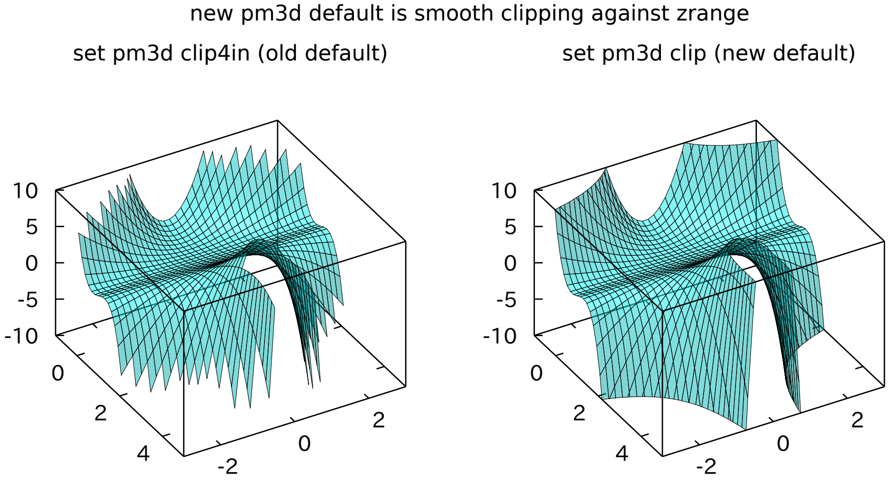

color lines: 'splot sin(y)/(y) with lines palette'
sin(y)/(y)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
X
-10
-5
0
5
10
Y
1.0*10^0
8.0*10^-1
6.0*10^-1
4.0*10^-1
2.0*10^-1
0.0*10^0
-2.0*10^-1
-4.0*10^-1

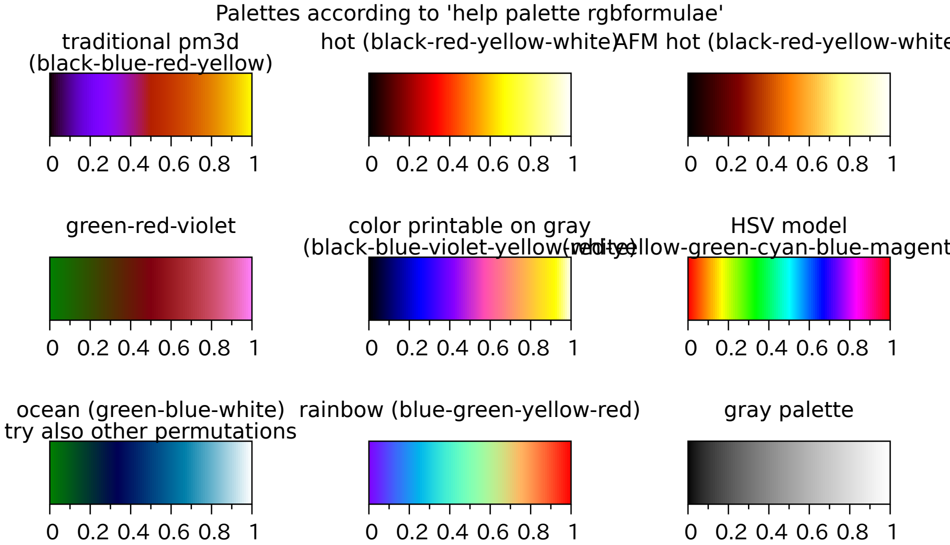


pm3d explicit mode --- coloring according to the 4th parameter of 'using'
'binary2' binary u 1:2:3:($2+($1+$2)/10)
1+sinc(x*4, y*4)
4
3
2
1
0
-1
-2
-3
-4
2
1.5
1
0.5
0
-0.5
-3
-2
-1
0
1
2
3
4
-3
-2
-1
0
1
2
3
4


coloring according to the 3rd 'using' parameter (left) and to the z-value (bottom)
4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
-16 -14 -12 -10 -8 -6 -4 -2 0 2 4
-4 -2 0 2 4 6 8 10 12 14 16

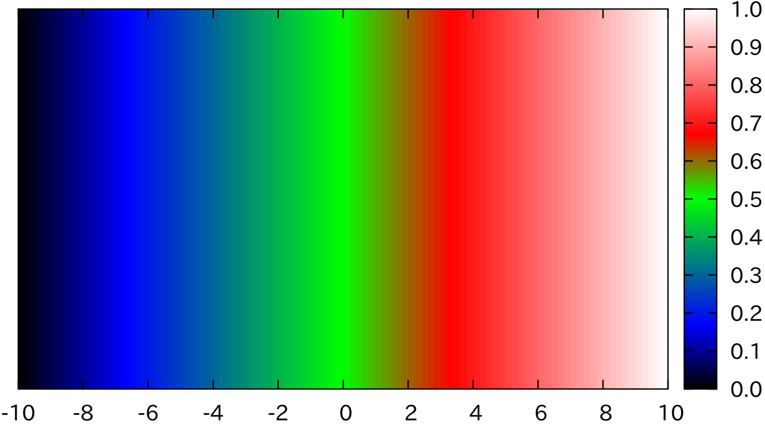

set pm3d corners2color mode
c3
mean
c4
harmean
Original grid points
geomean
c1
median
c2

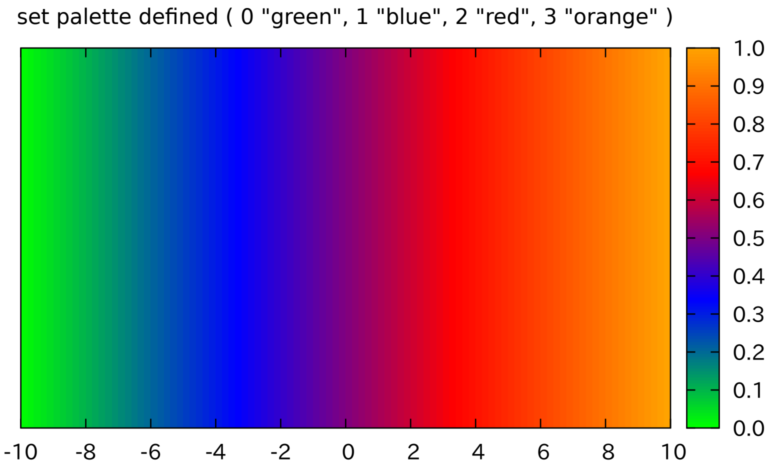

new pm3d default is smooth clipping against zrange
set pm3d clip4in (old default)
set pm3d clip (new default)
10
5
0
-5
-10
0
2
4
-2
0
2
10
5
0
-5
-10
0
2
4
-2
0
2


Palettes according to 'help palette rgbformulae'
traditional pm3d
(black-blue-red-yellow)
hot (black-red-yellow-white)
AFM hot (black-red-yellow-white
green-red-violet
color printable on gray
(black-blue-violet-yellow-white)
HSV model
(red-yellow-green-cyan-blue-magent
ocean (green-blue-white)
try also other permutations
rainbow (blue-green-yellow-red)
gray palette
0
0.2
0.4
0.6
0.8
1

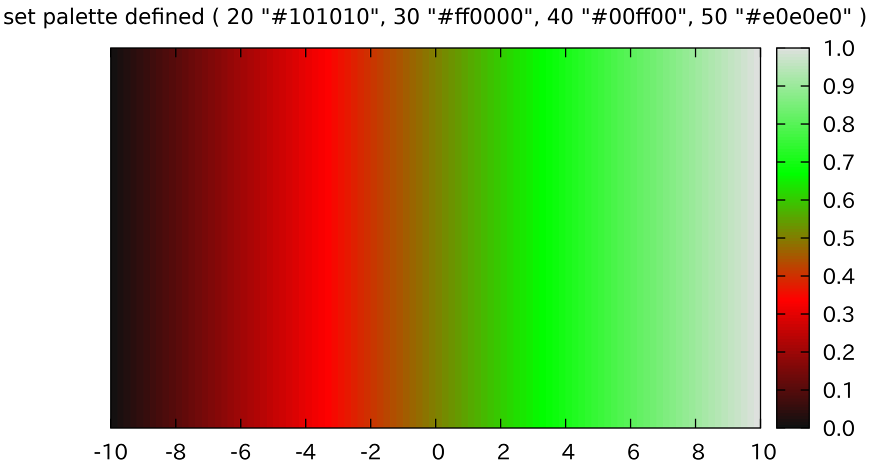

set palette defined
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

set palette defined (0 0 0 0, 1 0 0 1, 3 0 1 0, 4 1 0 0, 6 1 1 1)

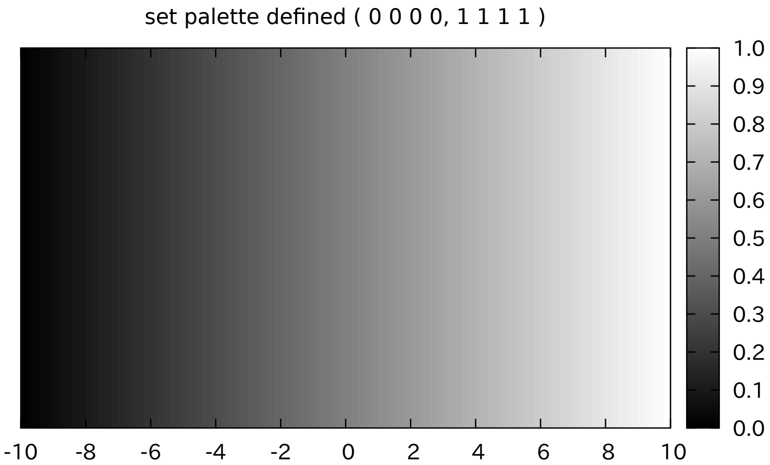


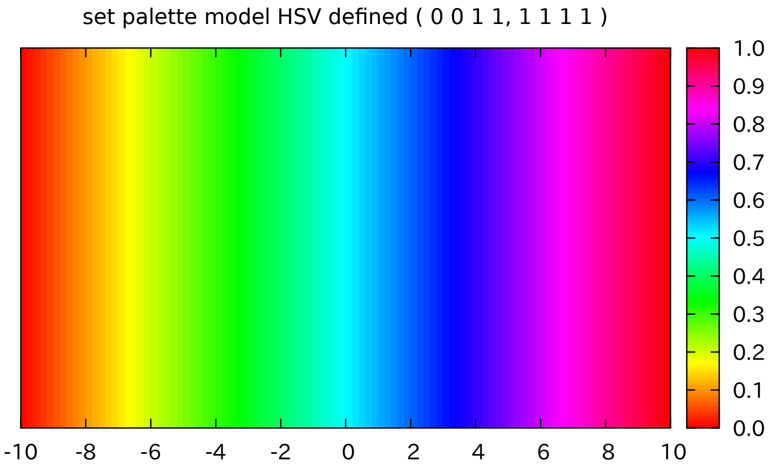


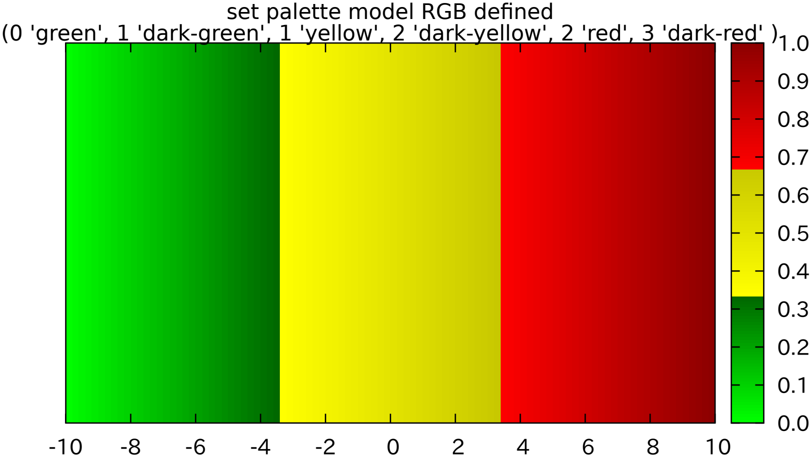
set palette defined (0 "green", 1 "blue", 2 "red", 3 "orange")
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0



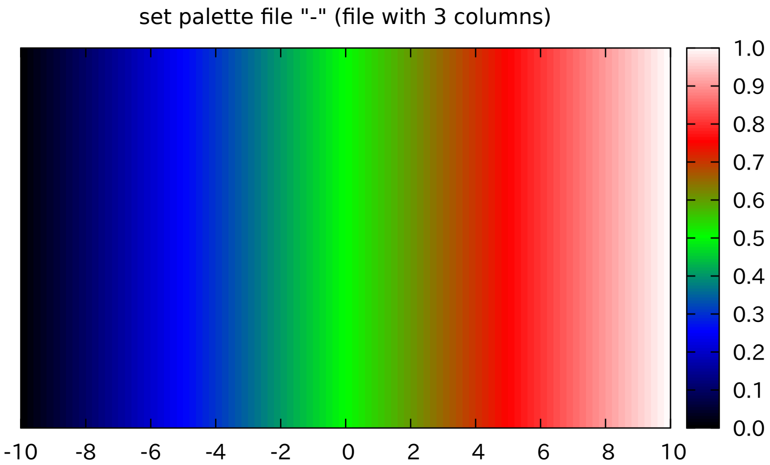
set palette defined (20 "#101010", 30 "#ff0000", 40 "#00ff00", 50 "#e0e0e0")
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

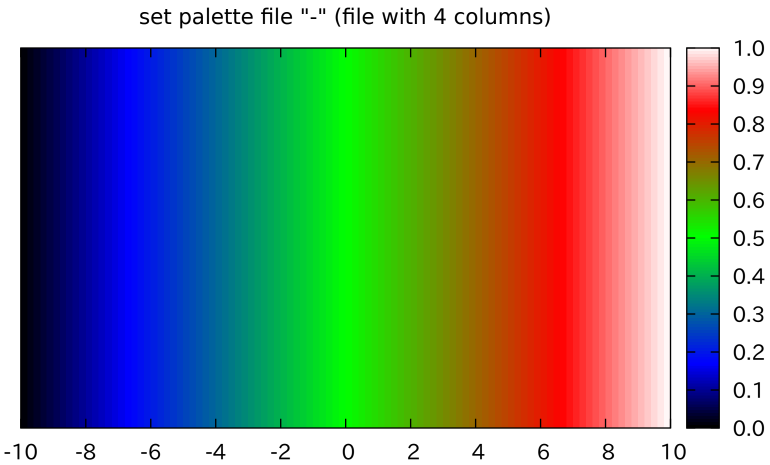

set palette defined (0 0 0 0, 1 1 1 1)
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

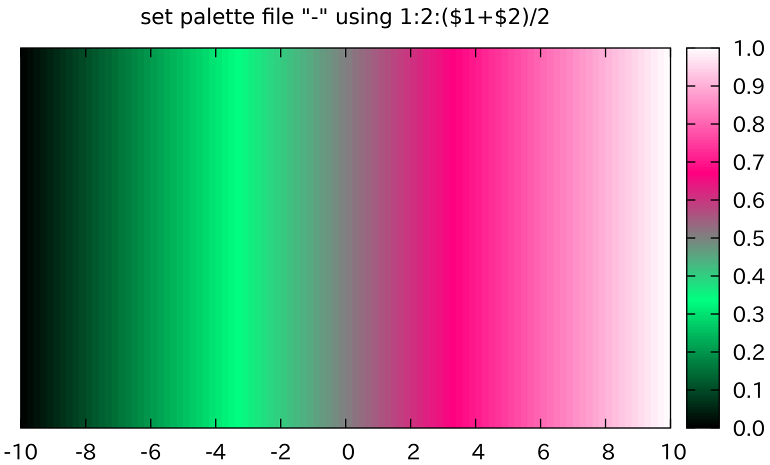

set palette model HSV defined (0 0 1 1, 1 1 1 1)
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

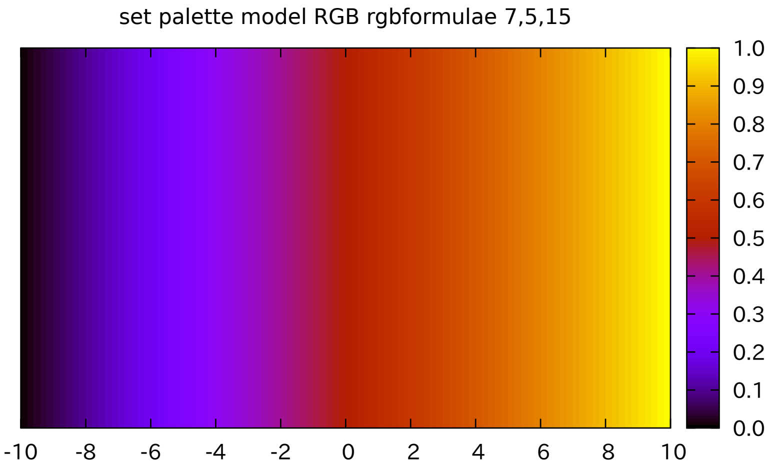


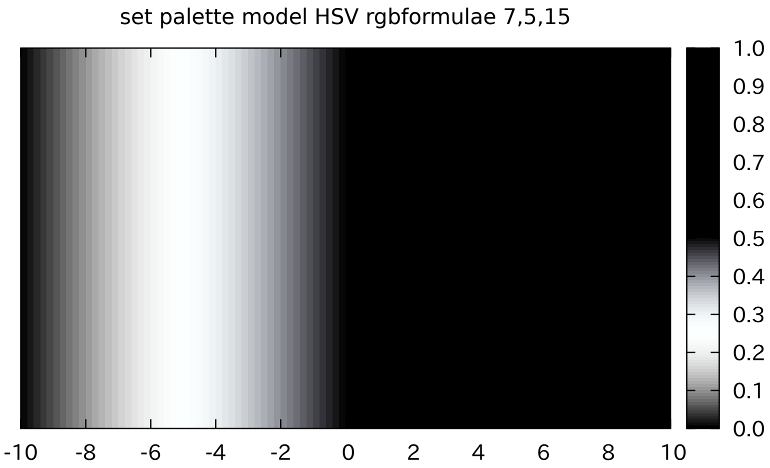
set palette model RGB defined
(0 'green', 1 'dark-green', 1 'yellow', 2 'dark-yellow', 2 'red', 3 'dark-red')
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-10
-8
-6
-4
-2
0
2
4
6
8
10


set palette file "-" (file with 3 columns)
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

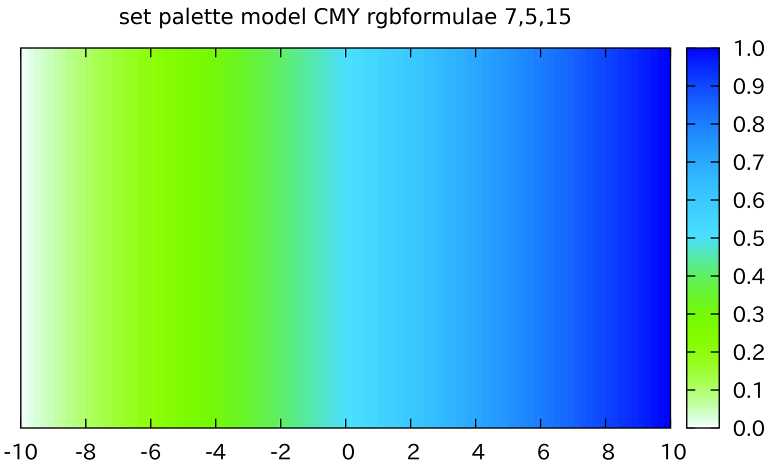

set palette file "-" (file with 4 columns)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
-10
-8
-6
-4
-2
0
2
4
6
8
10

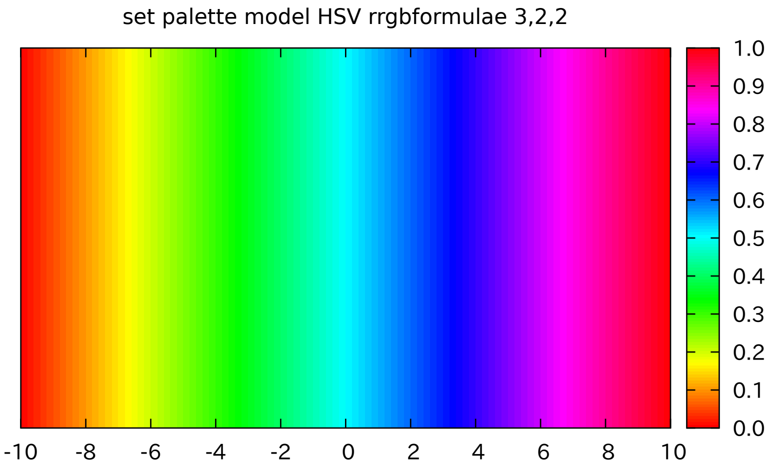

set palette file "-" using 1:2:($1+$2)/2
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


set palette model RGB rgbformulae 7,5,15
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

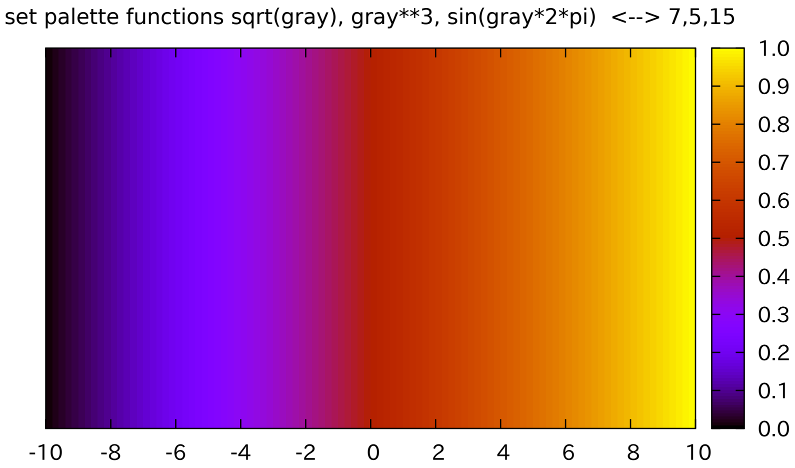


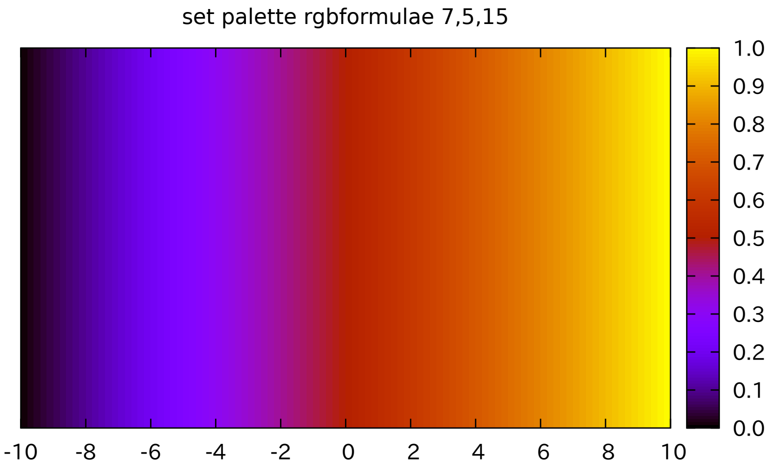
set palette model HSV rgbformulae 7,5,15
-10
-8
-6
-4
-2
0
2
4
6
8
10
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

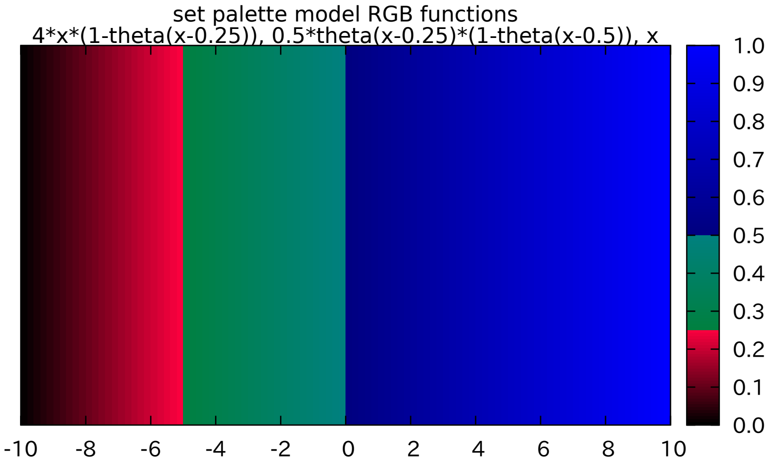

set palette model CMY rgbformulae 7,5,15
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

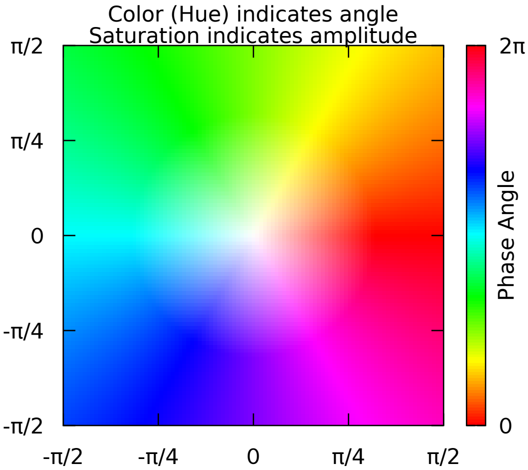

set palette model HSV rrgbformulae 3,2,2
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

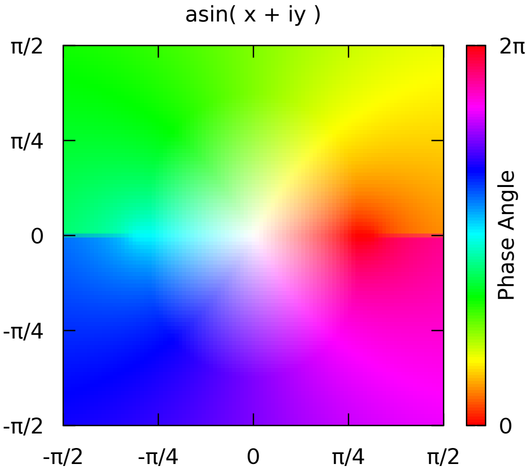

set palette functions gray, gray, gray
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

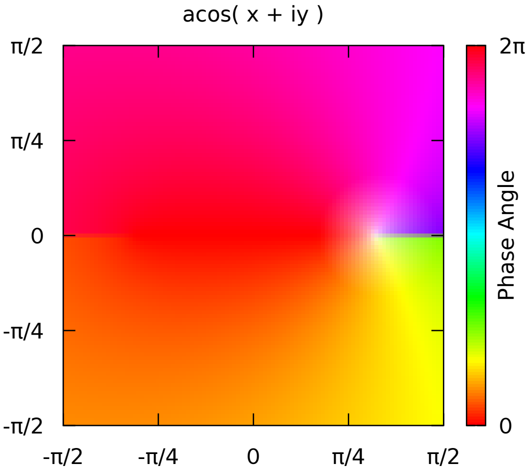

set palette functions sqrt(gray), gray**3, sin(gray*2*pi) <--> 7,5,15
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

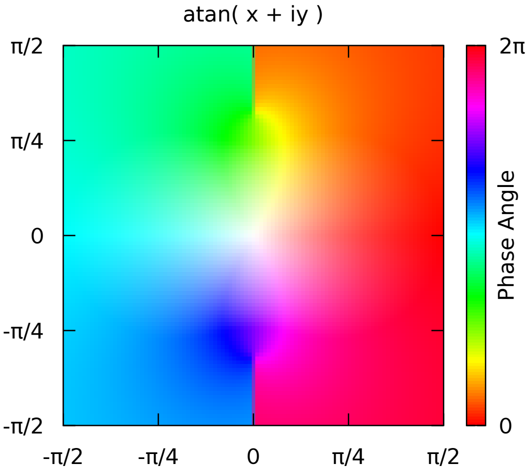

set palette rgbformulae 7,5,15
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


set palette model RGB functions
4*x*(1-theta(x-0.25)), 0.5*theta(x-0.25)*(1-theta(x-0.5)), x
-10
-8
-6
-4
-2
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

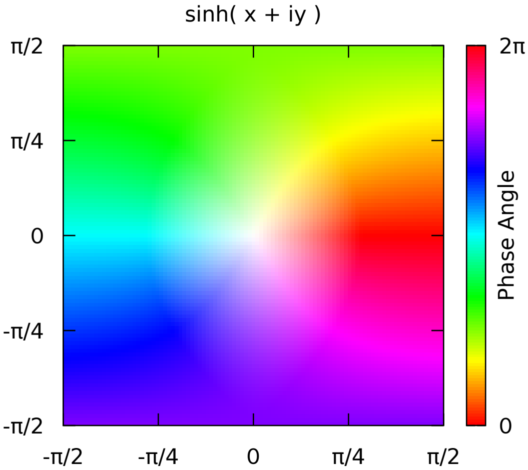

Color (Hue) indicates angle
Saturation indicates amplitude
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle

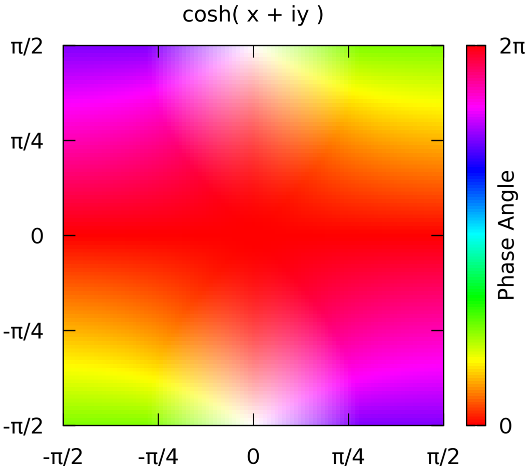

asin(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
Phase Angle
0

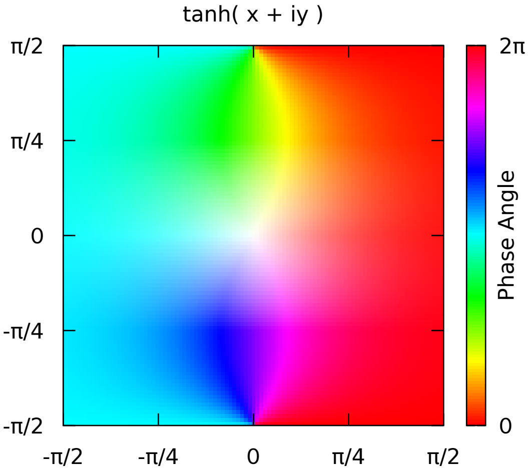

acos(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle

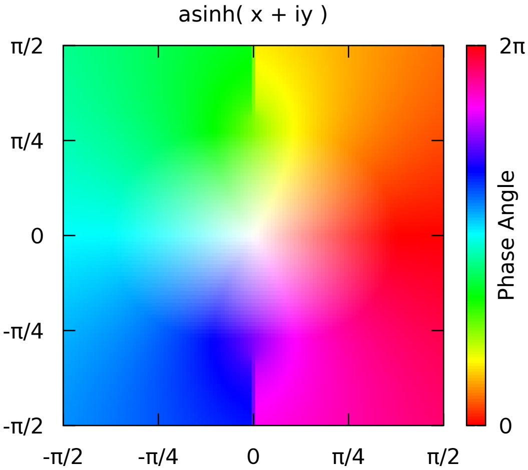


atan(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
Phase Angle
0

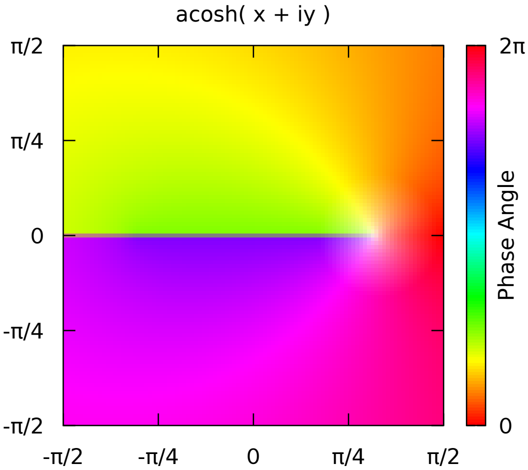

sinh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle

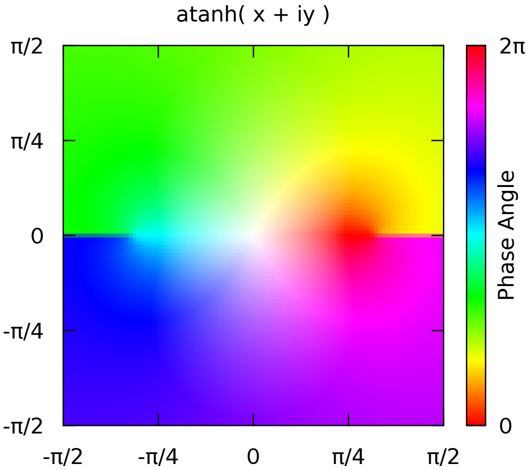

cosh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle


tanh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle

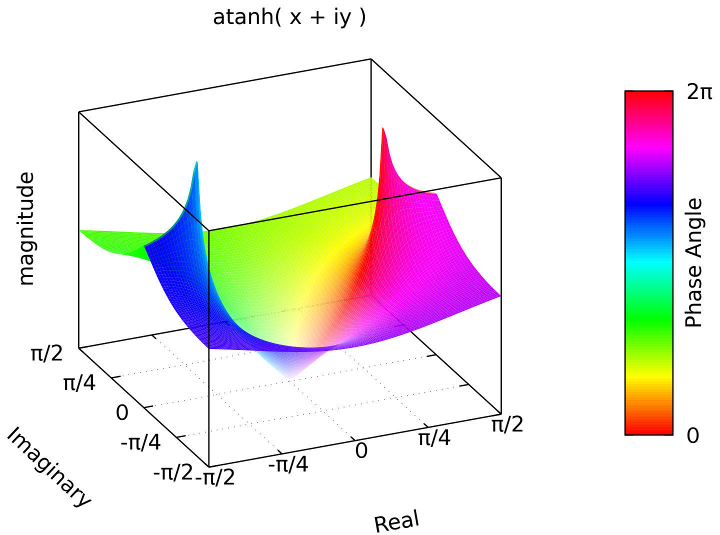


asinh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle

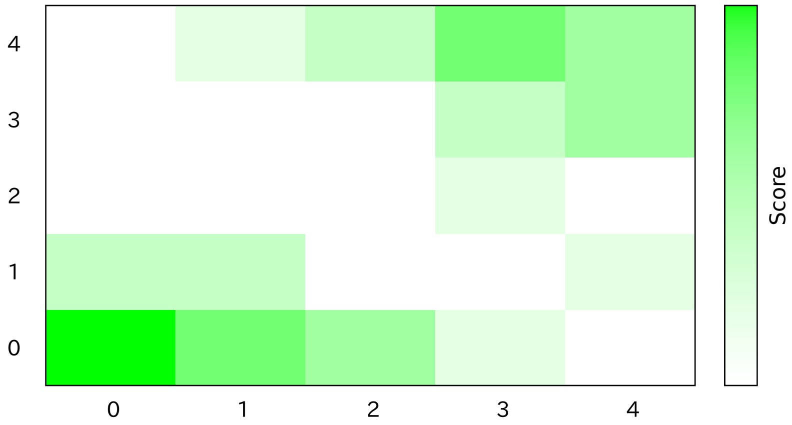

acosh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle



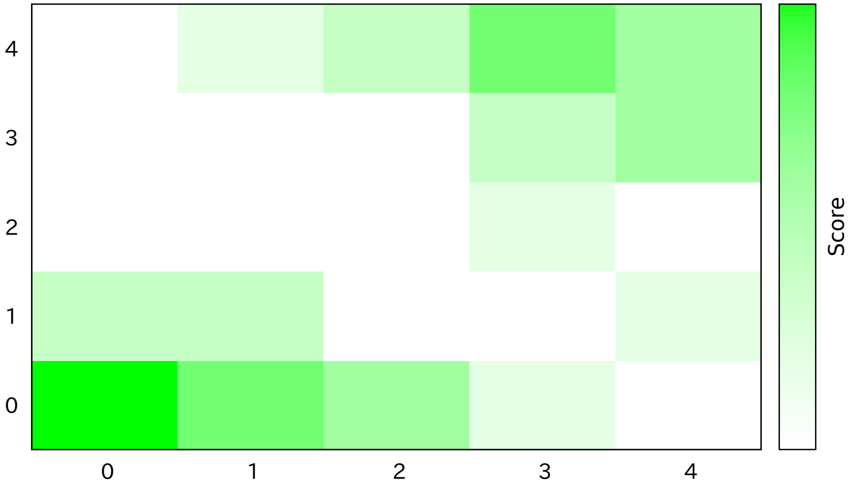
atanh(x + iy)
π/2
π/4
0
-π/4
-π/2
-π/2
-π/4
0
π/4
π/2
2π
0
Phase Angle


atanh(x + iy)
magnitude
π/2
π/4
0
-π/4
-π/2
Imaginary
-π/2
-π/4
0
π/4
π/2
Real
2π
Phase Angle
0

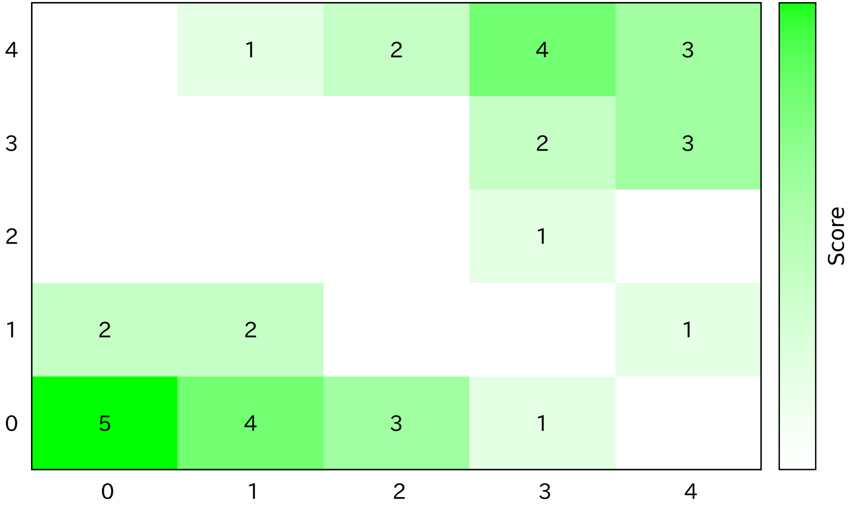
Heat Map generated from a file containing Z values only

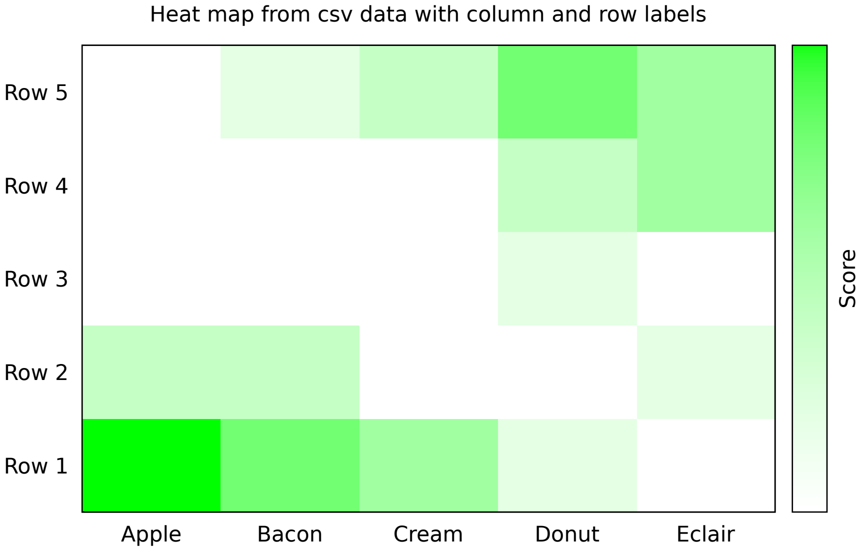


Heat Map generated by 'plot' from a stream of XYZ values
NB: Rows must be separated by blank lines!

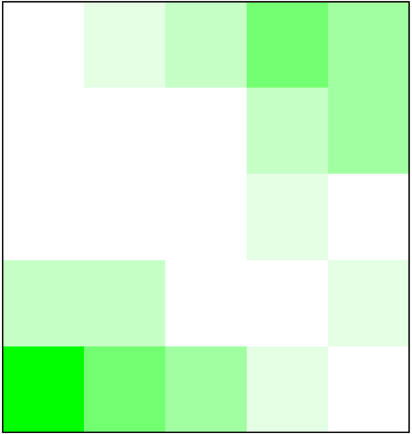


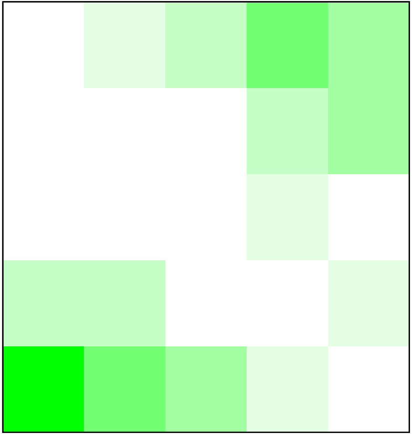
Heat map with non-zero pixel values written as labels

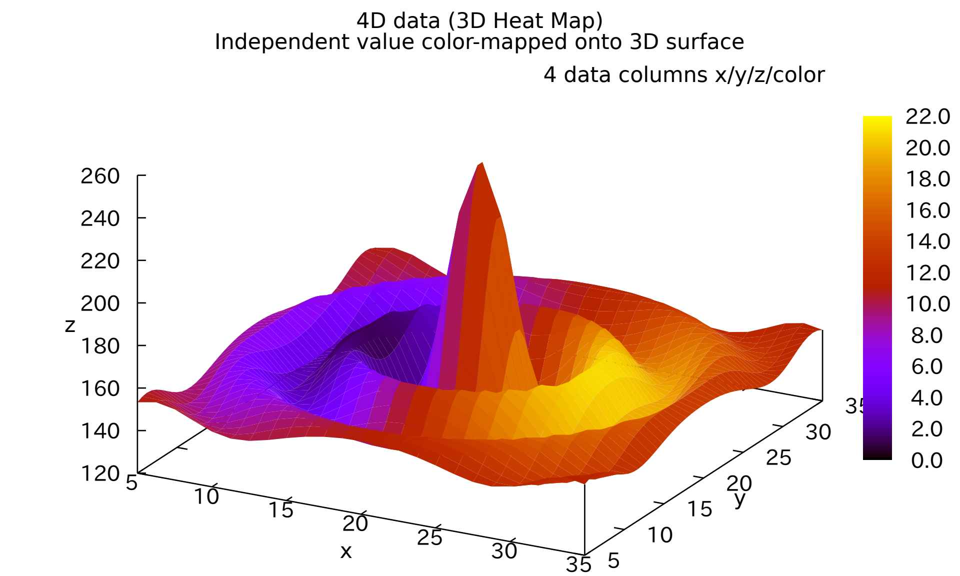


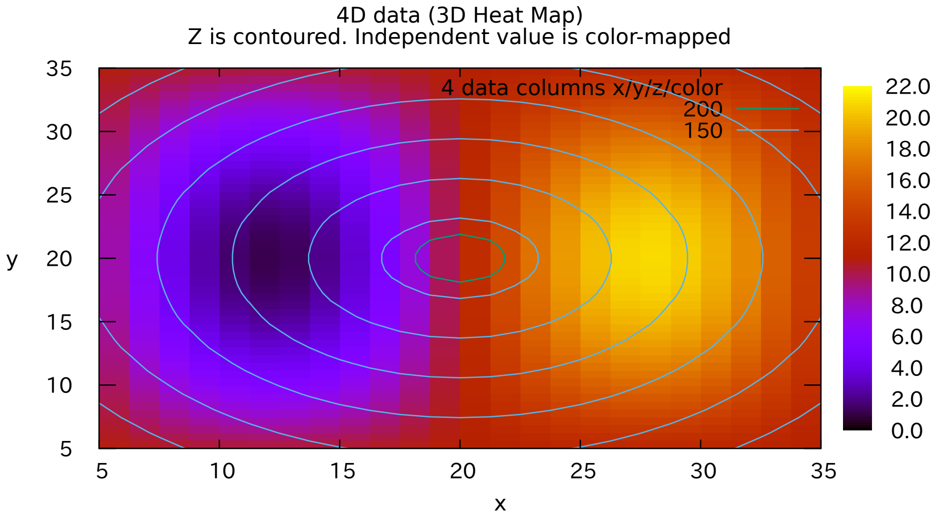

Heat map from csv data with column and row labels
Row 5
Row 4
Row 3
Row 2
Row 1
Apple
Bacon
Cream
Donut
Eclair
Score

Compare 'image' and 'image pixels' modes

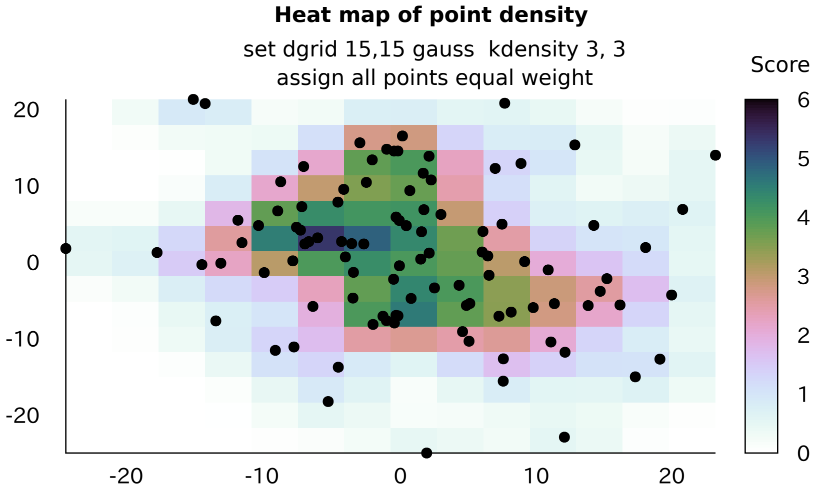




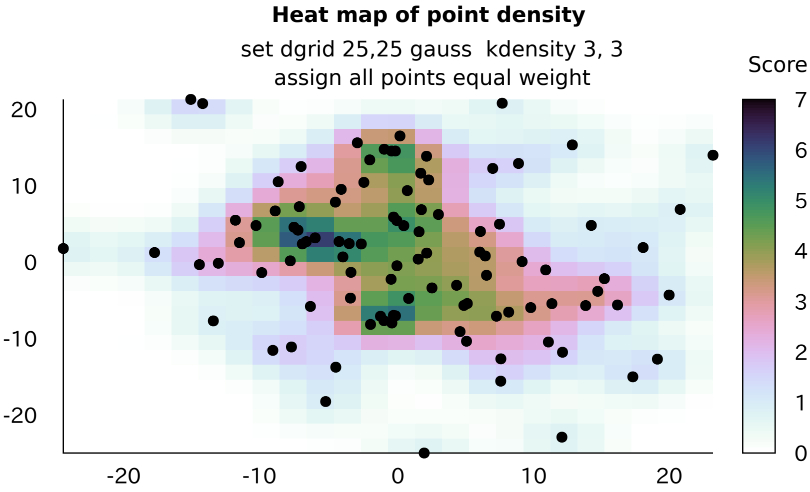

4D data (3D Heat Map)
Independent value color-mapped onto 3D surface
4 data columns x/y/z/color
z
260
240
220
200
180
160
140
120
x
5
10
15
20
25
30
35
y
5
10
15
20
25
30
35
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0



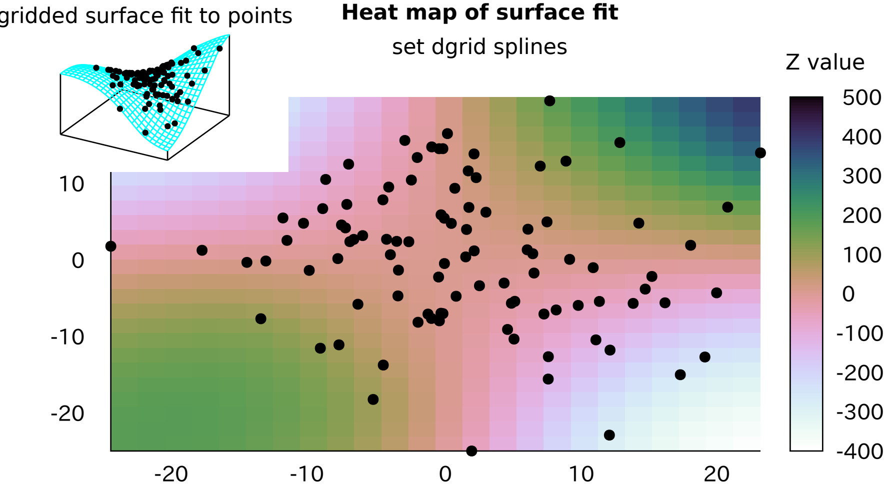
4D data (3D Heat Map)
Z is contoured. Independent value is color-mapped
4 data columns x/y/z/color
200
150
y
x
35
30
25
20
15
10
5
5
10
15
20
25
30
35
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0

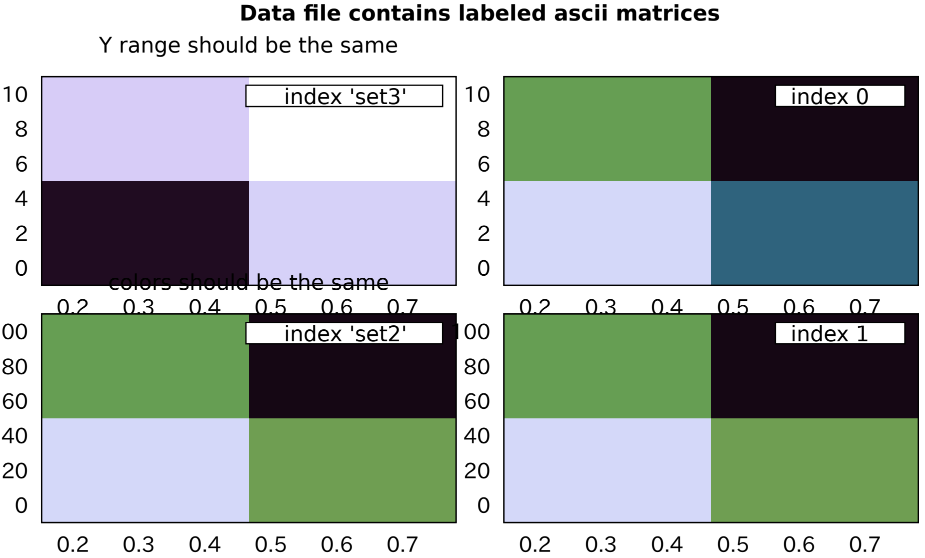


Heat map of point density
set dgrid 15,15 gauss kdensity 3, 3
assign all points equal weight
Score
20
10
0
-10
-20
-20
-10
0
10
20
6
5
4
3
2
1
0

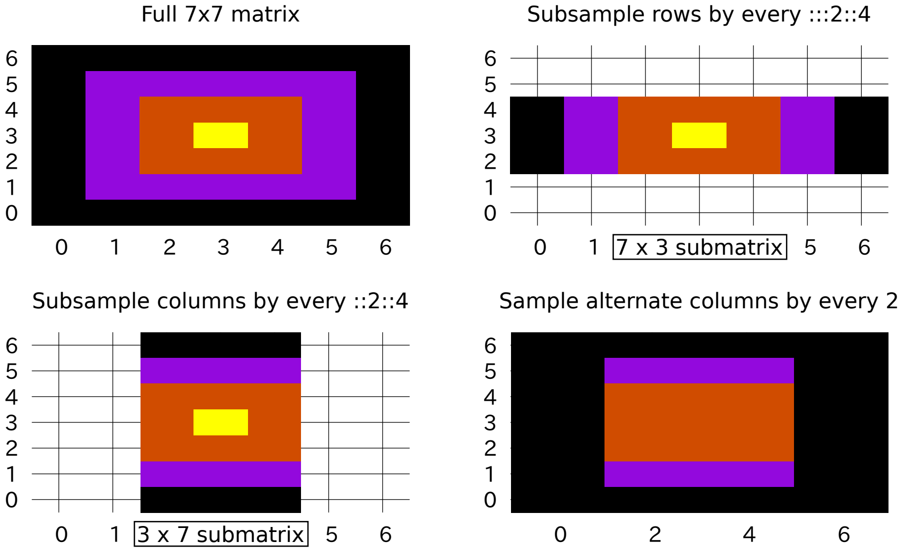


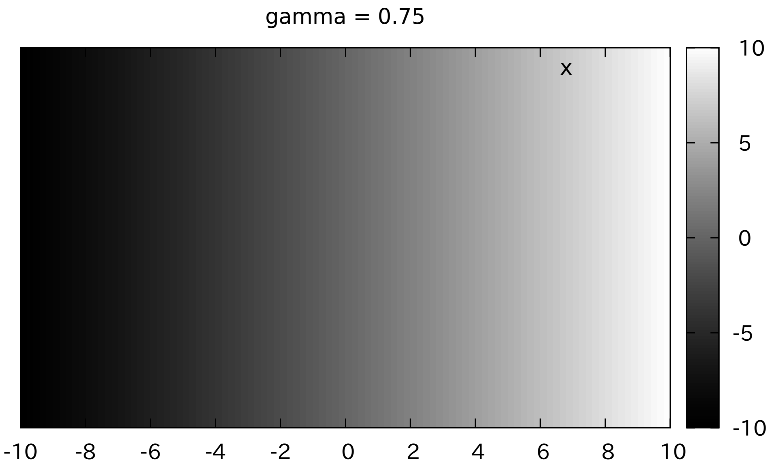
Heat map of point density
set dgrid 25,25 gauss kdensity 3, 3
assign all points equal weight
Score
20
10
0
-10
-20
-20
-10
0
10
20
7
6
5
4
3
2
1
0

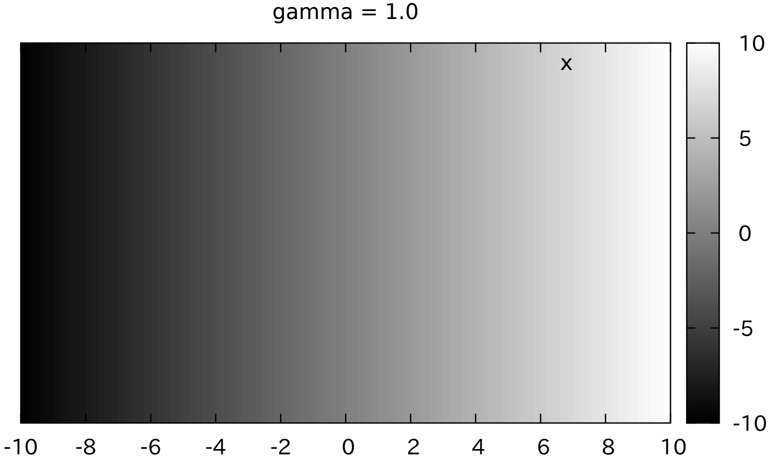

gridded surface fit to points
Heat map of surface fit
set dgrid splines
Z value
500
400
300
200
100
0
-100
-200
-300
-400
10
0
-10
-20
-20
-10
0
10
20



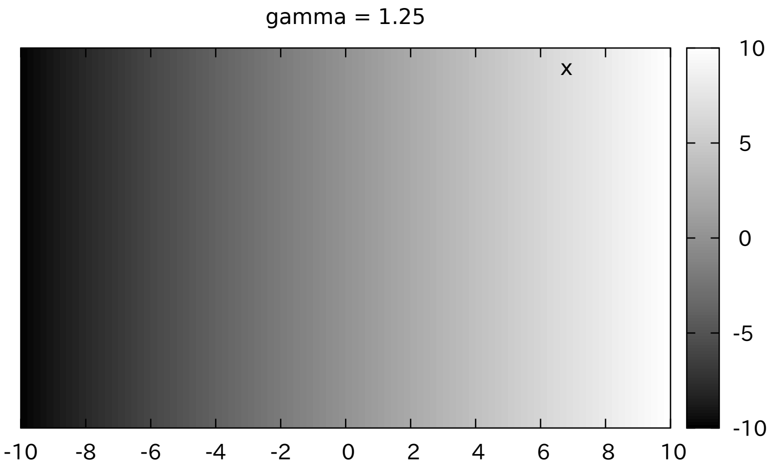
Data file contains labeled ascii matrices
Y range should be the same
index 'set3'
index 0
colors should be the same
index 'set2'
index 1
10
8
6
4
2
0
100
80
60
40
20
0
0.2
0.3
0.4
0.5
0.6
0.7


Full 7x7 matrix
Subsample rows by every :::2::4
7 x 3 submatrix
Subsample columns by every ::2::4
3 x 7 submatrix
Sample alternate columns by every 2

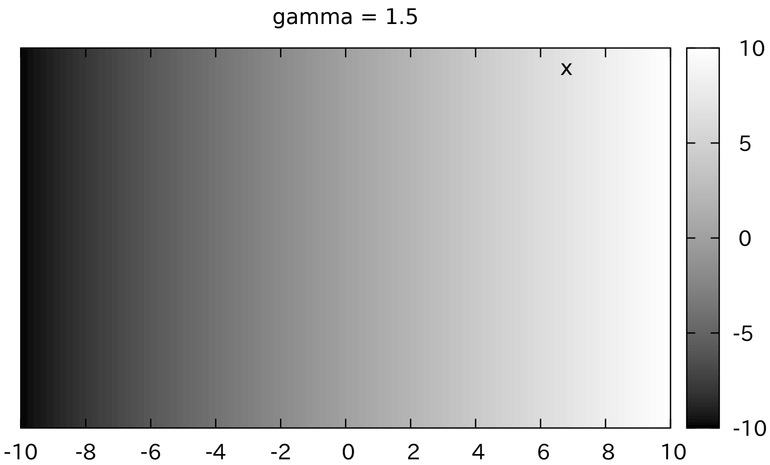

gamma = 0.75
x
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10



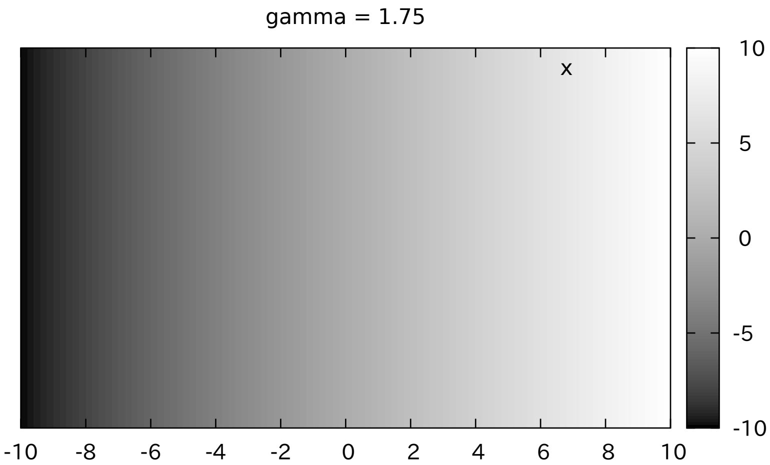
gamma = 1.0
x
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10



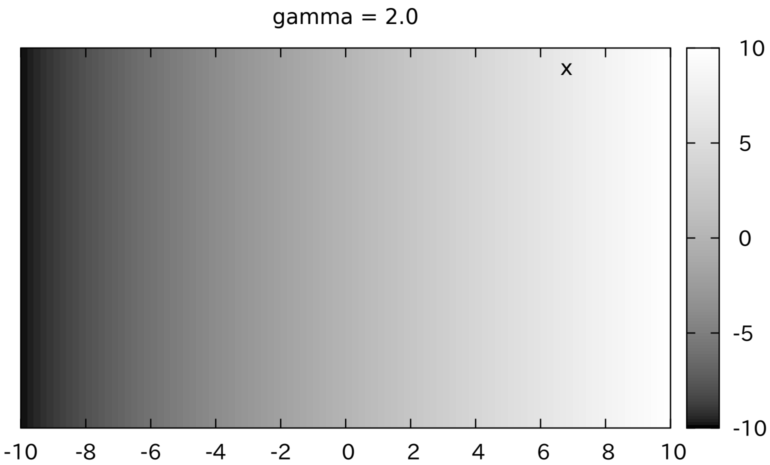
gamma = 1.25
x
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10



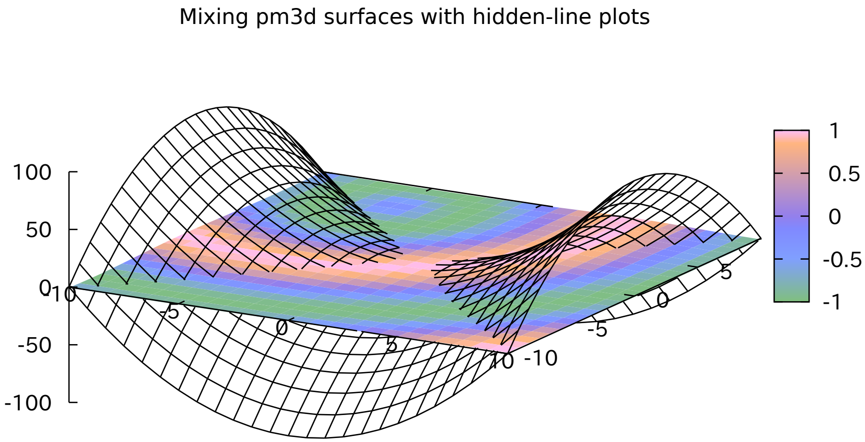
gamma = 1.5
x
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10



gamma = 1.75
x
-10
-8
-6
-4
-2
0
2
4
6
8
10
10
5
0
-5
-10

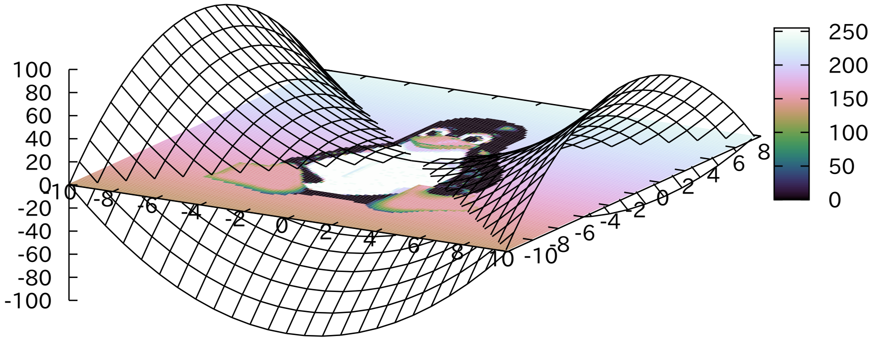


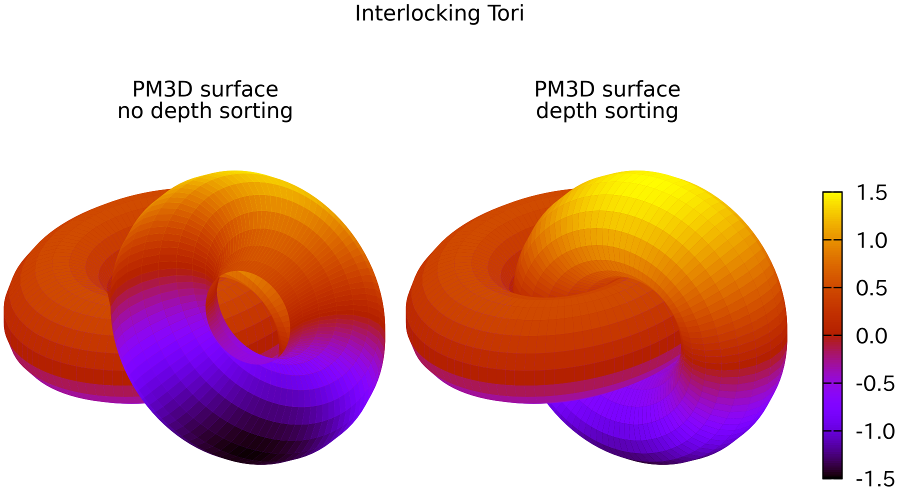
gamma = 2.0
x
-10
-8
-6
-4
-2
0
2
4
6
8
10
10
5
0
-5
-10



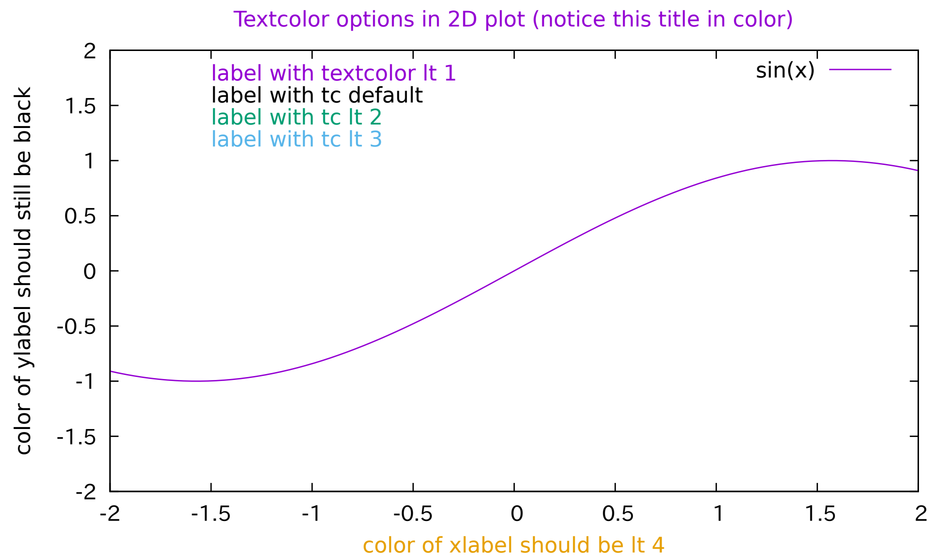
Mixing pm3d surfaces with hidden-line plots
100
50
0
-50
-100
-10
-5
0
5
10
-10
-5
0
5
1
0.5
0
-0.5
-1

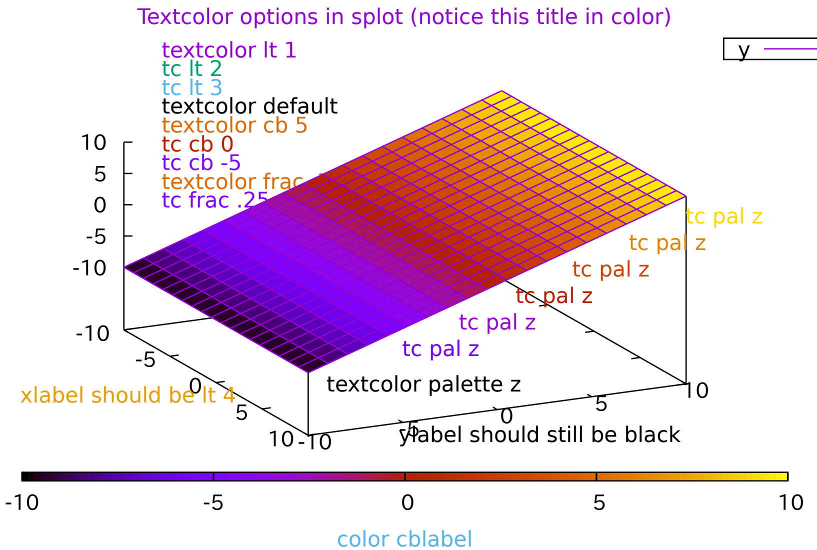
Mixing image surface with hidden-line plots



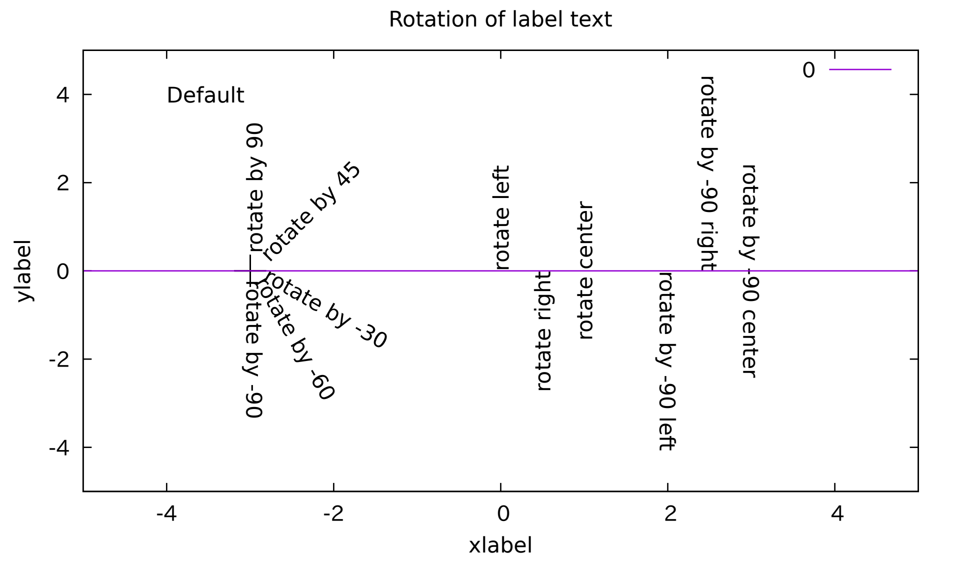


Interlocking Tori
PM3D surface
no depth sorting
PM3D surface
depth sorting
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5

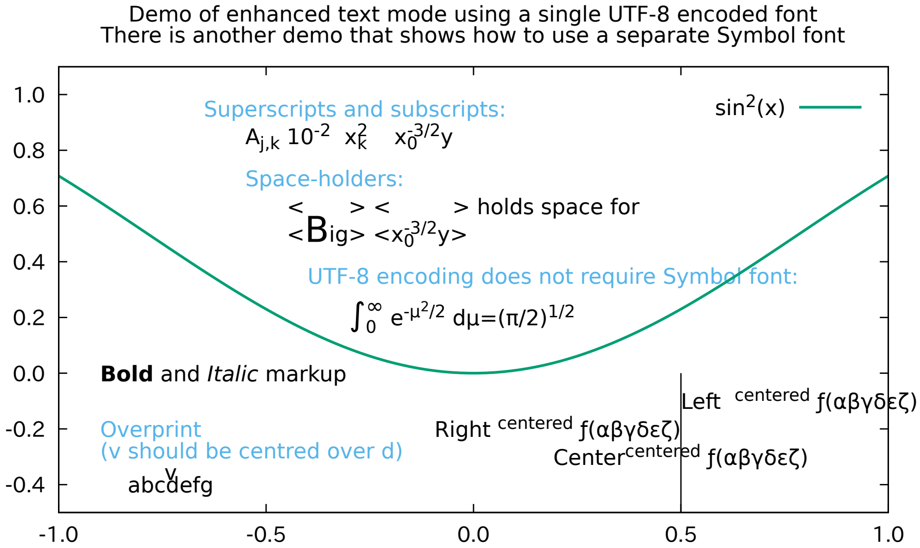

Textcolor options in 2D plot (notice this title in color)
label with textcolor lt 1
label with tc default
label with tc lt 2
label with tc lt 3
sin(x)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
color of ylabel should still be black
color of xlabel should be lt 4

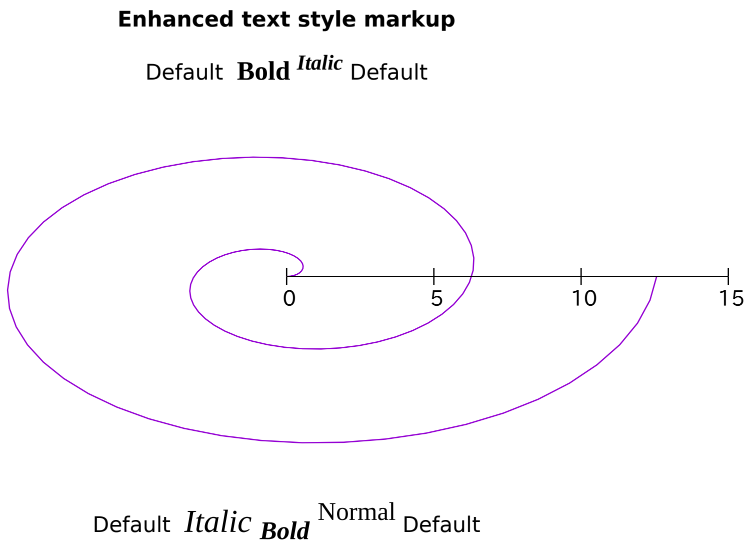


Textcolor options in splot (notice this title in color)
textcolor lt 1
tc lt 2
tc lt 3
textcolor default
textcolor cb 5
tc cb 0
tc cb -5
textcolor frac
tc frac .25
y
10
5
0
-5
-10
-10
-5
0
5
10
-10
-5
0
5
10
tc pal z
tc pal z
tc pal z
tc pal z
tc pal z
tc pal z
textcolor palette z
xlabel should be lt 4
ylabel should still be black
-10
-5
0
5
10
color cblabel



Rotation of label text
Default
rotate by 90
rotate by 45
rotate by -30
rotate by -60
rotate by -90
rotate left
rotate right
rotate center
rotate by -90 left
rotate by -90 right
rotate by -90 center
0
xlabel
ylabel
-4
-2
0
2
4


Demo of enhanced text mode using a single UTF-8 encoded font
There is another demo that shows how to use a separate Symbol font
Superscripts and subscripts:
A_{j,k} 10^{-2} x^2_k x^{-3/2}_0y
Space-holders:
< > < > holds space for
<Big> <x^{-3/2}_0y>
UTF-8 encoding does not require Symbol font:
∫_0^∞ e^{-μ^2/2} dμ=(π/2)^{1/2}
sin^2(x)
Bold and Italic markup
Overprint
(v should be centred over d)
abcdefg
Left centered ƒ(αβγδεζ)
Right centered ƒ(αβγδεζ)
Center centered ƒ(αβγδεζ)
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-1.0
-0.5
0.0
0.5
1.0


Enhanced text style markup
Default Bold Italic Default
0
5
10
15
Default Italic Bold Normal Default

Illustrate use of unicode escape sequences

unicode \U+221E : \U+221E   Infinity

unicode \U+210F : \U+210F   Planck constant h-bar

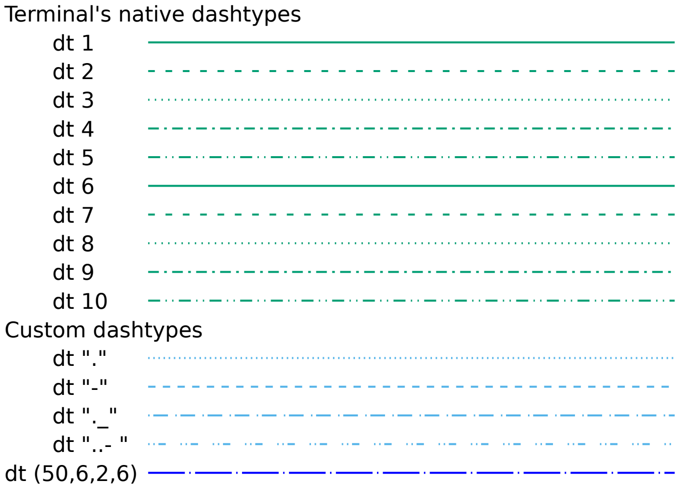
unicode \U+222C : \U+222C   Double integral

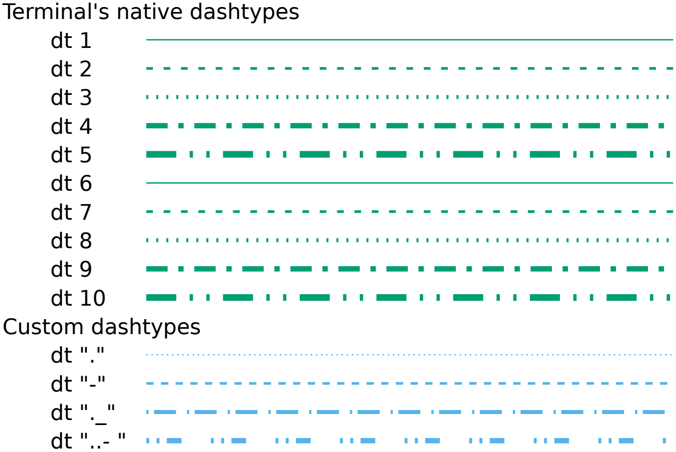
unicode \U+03F5 : \U+03F5   Greek lunate epsilon

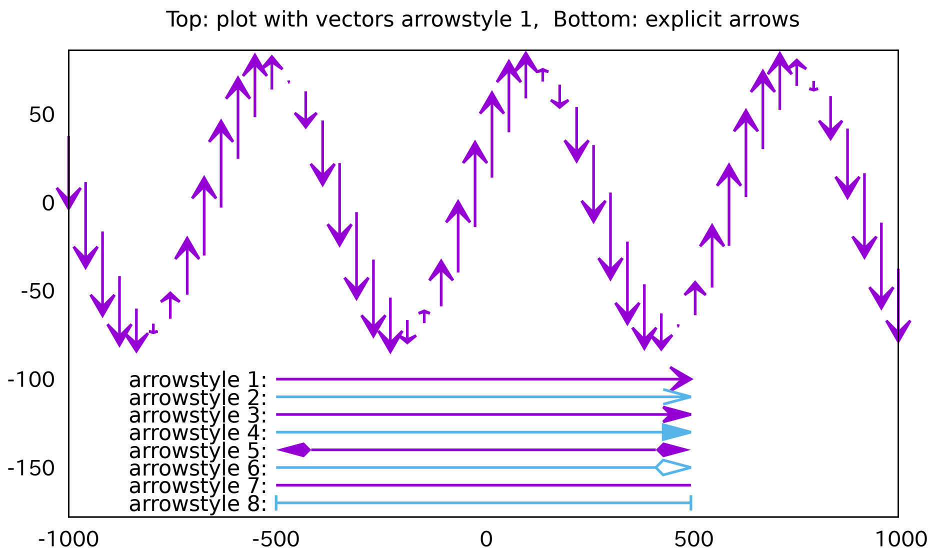
unicode \U+7403 : \U+7403   CJK unified ideograph 'sphere'

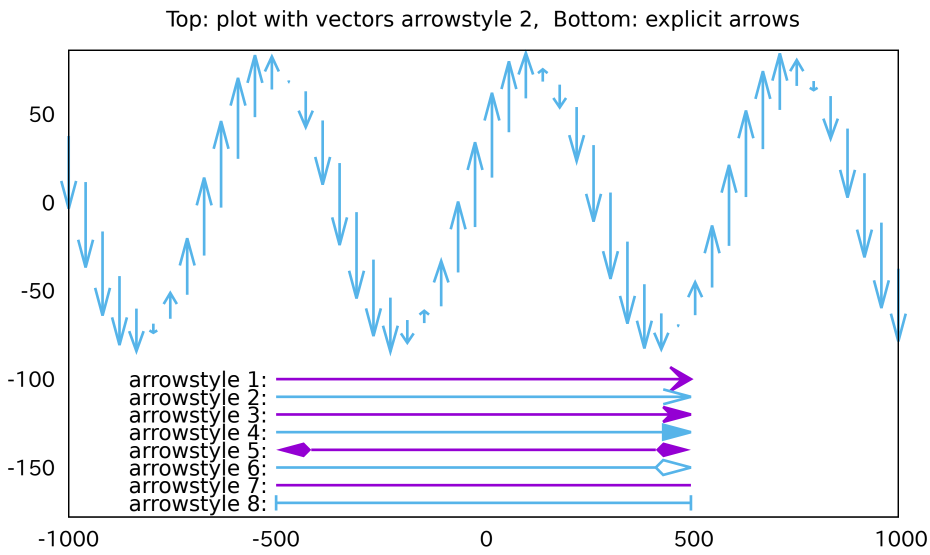
}a\U+0361}b   : a\U+0361b   Ligature tie (combining)

v\U+20D7   : v\U+20D7   Combining right arrow above

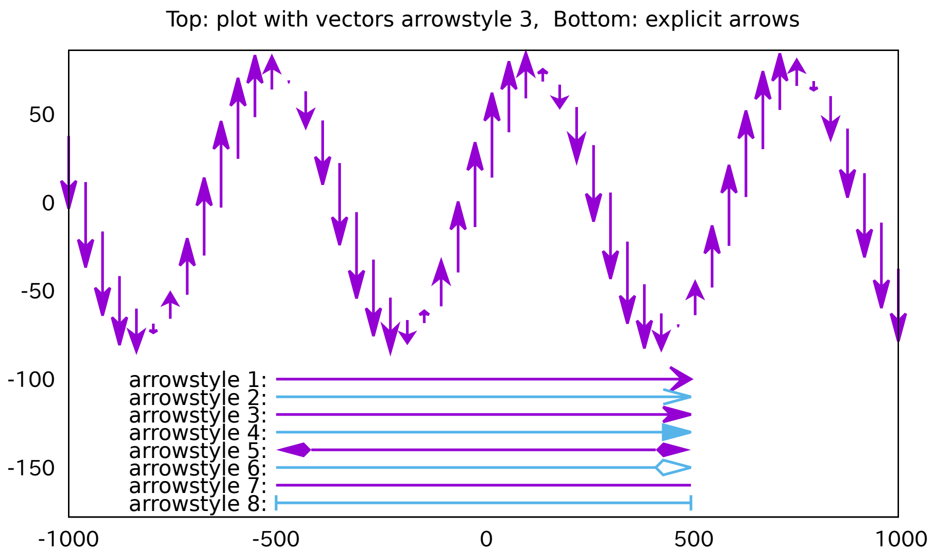

Terminal's native dashtypes
dt 1
dt 2
dt 3
dt 4
dt 5
dt 6
dt 7
dt 8
dt 9
dt 10
Custom dashtypes
dt "."
dt "-"
dt "._"
dt "..- "
dt (50,6,2,6)


Terminal's native dashtypes
dt 1
dt 2
dt 3
dt 4
dt 5
dt 6
dt 7
dt 8
dt 9
dt 10
Custom dashtypes
dt "."
dt "-"
dt "._"
dt "..- "

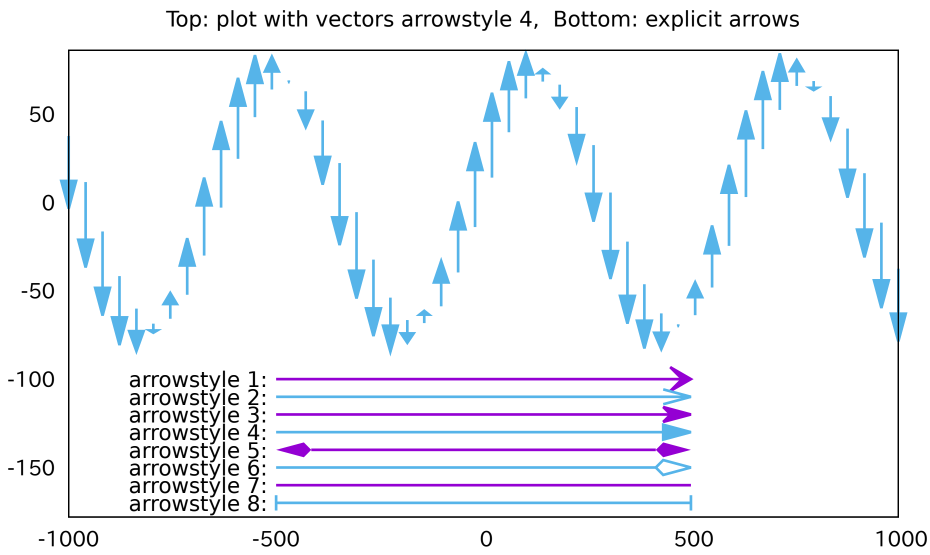

Top: plot with vectors arrowstyle 1, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:

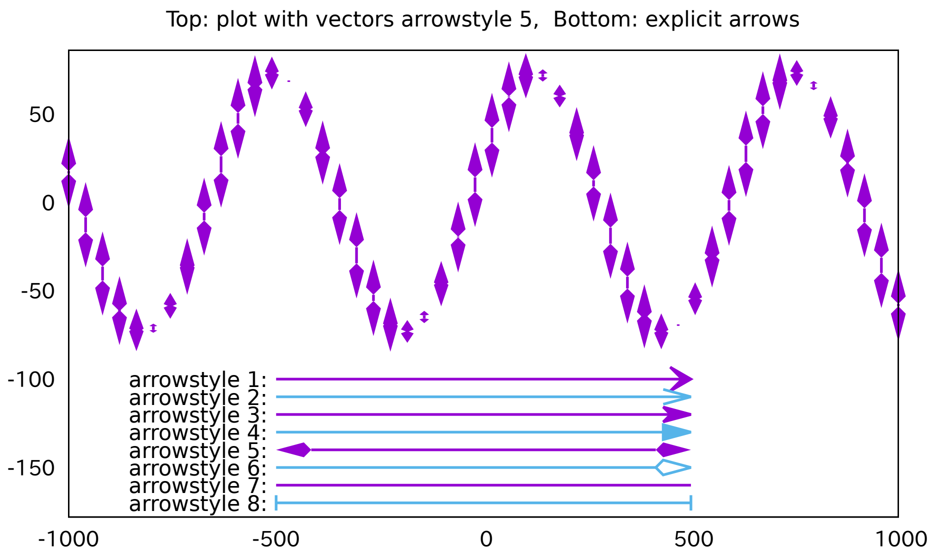


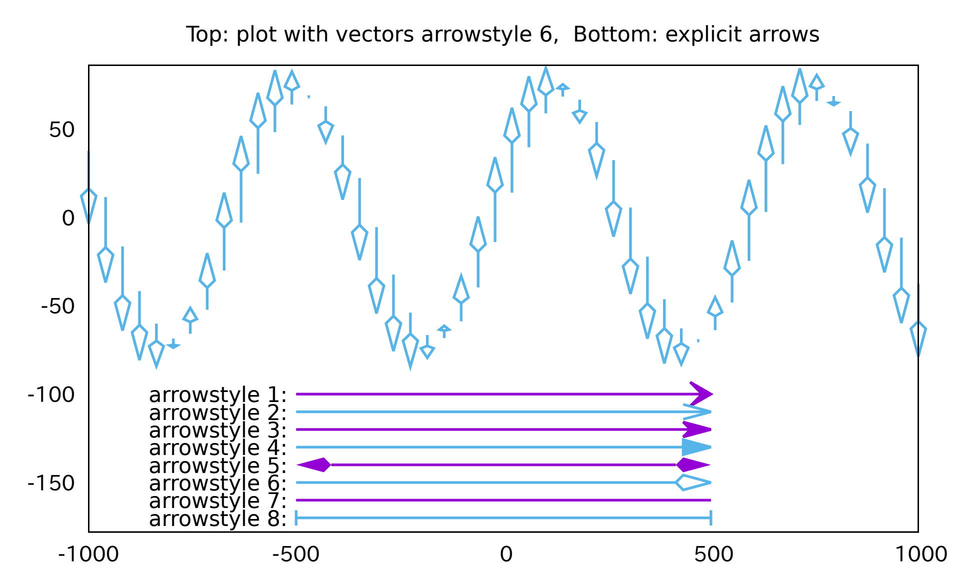
Top: plot with vectors arrowstyle 2, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:



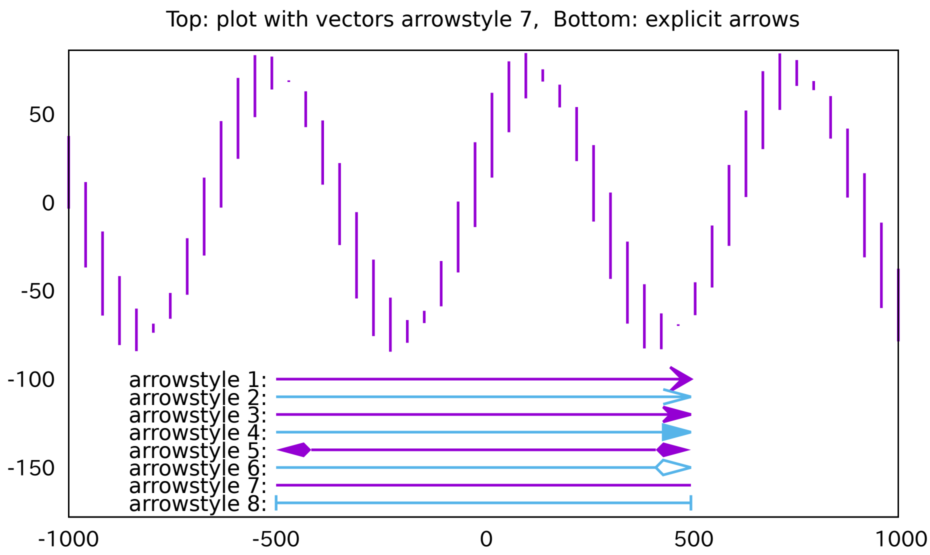
Top: plot with vectors arrowstyle 3, Bottom: explicit arrows
50
0
-50
-100
-150
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:
-1000
-500
0
500
1000


Top: plot with vectors arrowstyle 4, Bottom: explicit arrows
50
0
-50
-100
-150
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:
-1000
-500
0
500
1000

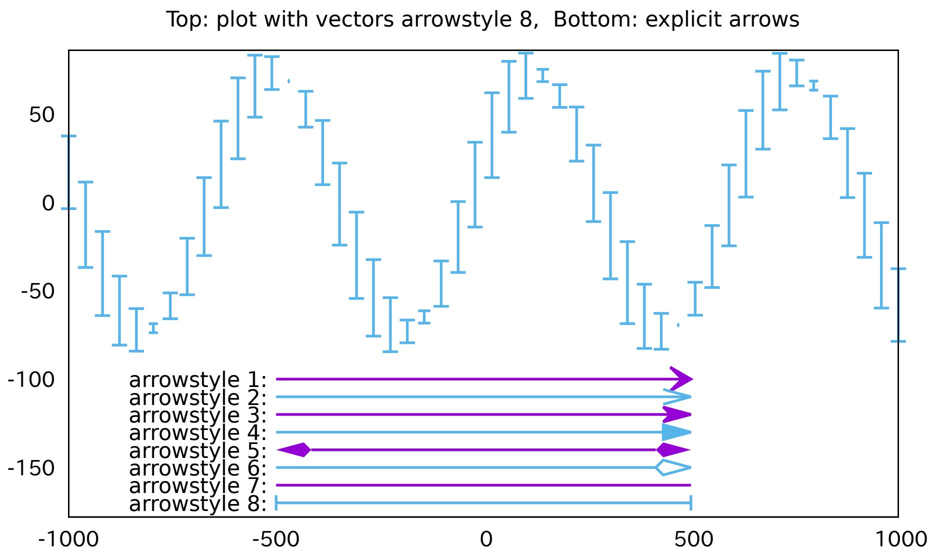

Top: plot with vectors arrowstyle 5, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:

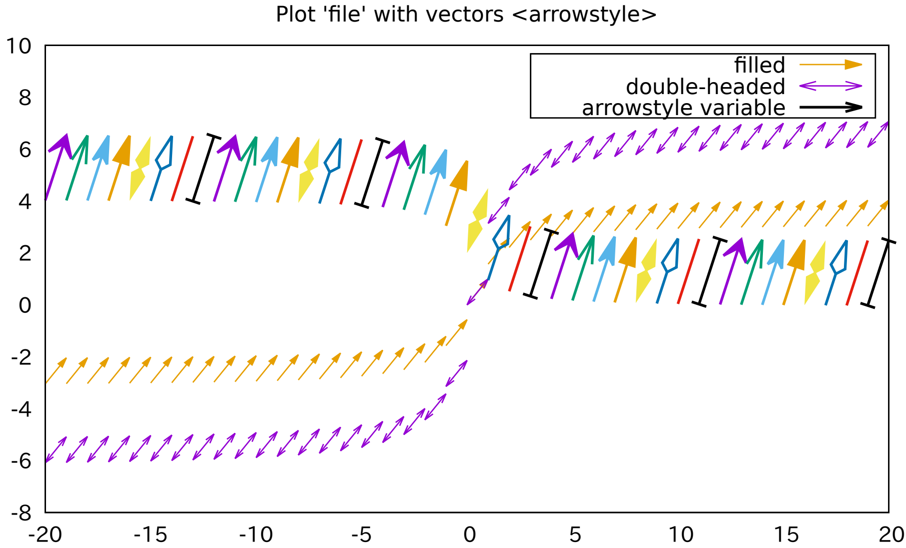


Top: plot with vectors arrowstyle 6, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:

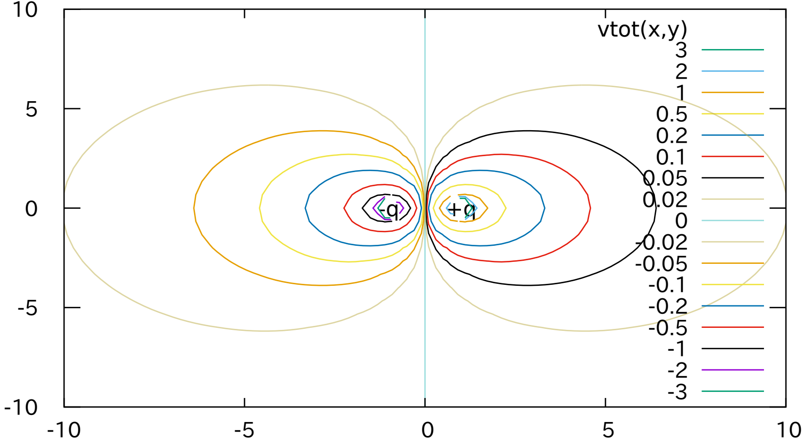


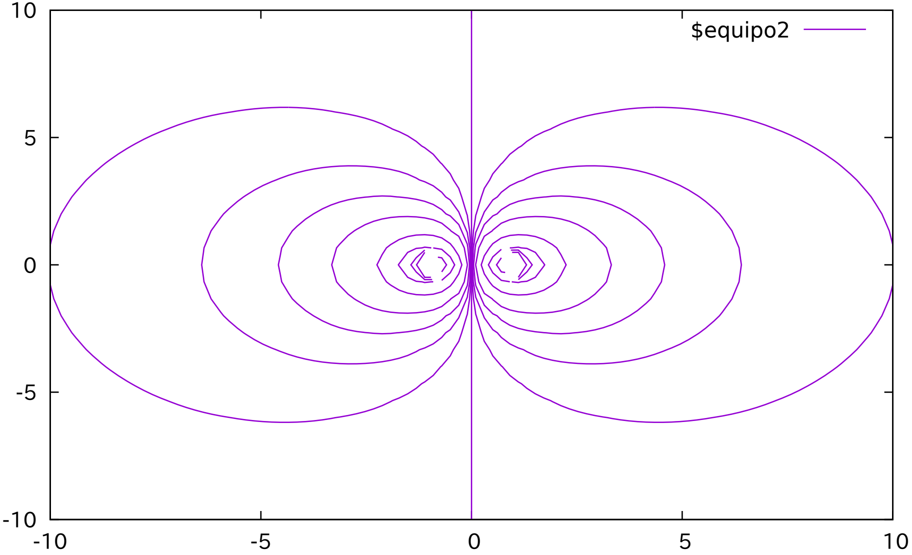
Top: plot with vectors arrowstyle 7, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:

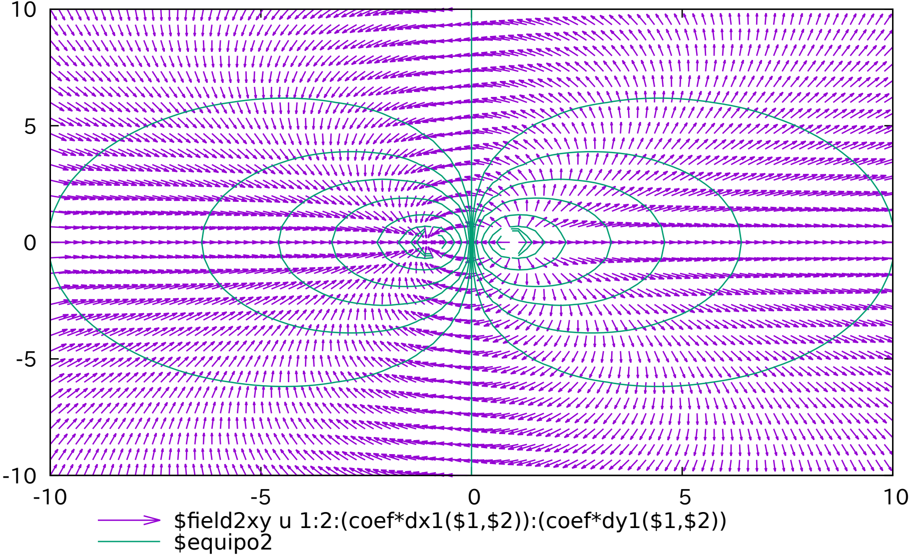

Top: plot with vectors arrowstyle 8, Bottom: explicit arrows
50
0
-50
-100
-150
-1000
-500
0
500
1000
arrowstyle 1:
arrowstyle 2:
arrowstyle 3:
arrowstyle 4:
arrowstyle 5:
arrowstyle 6:
arrowstyle 7:
arrowstyle 8:

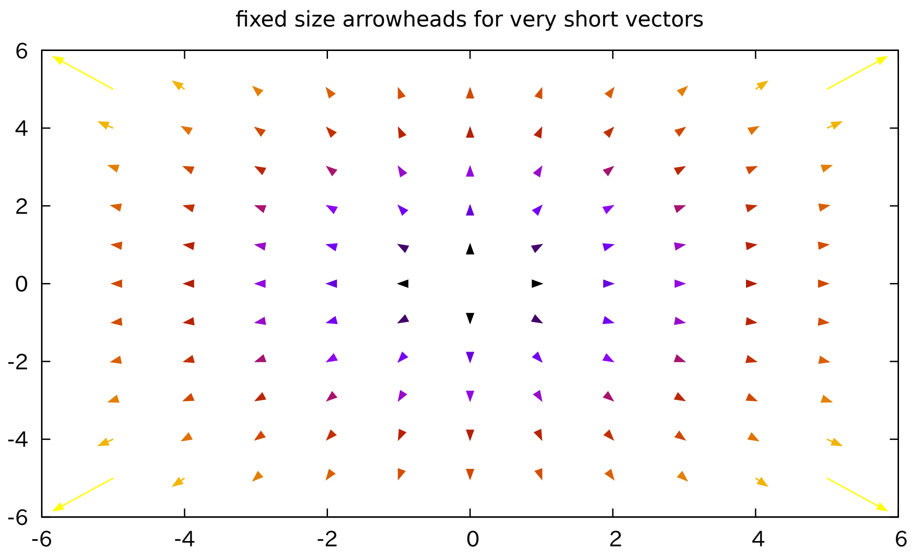


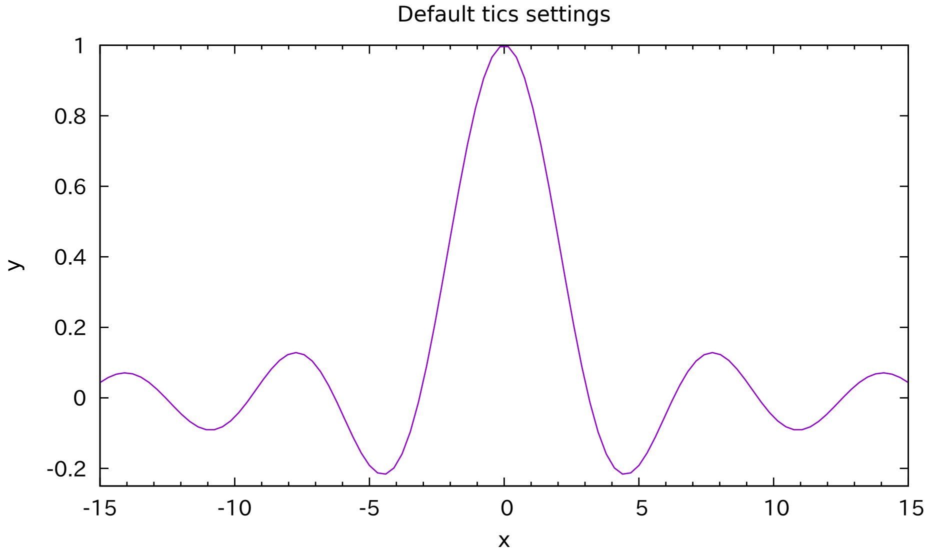
Plot 'file' with vectors <arrowstyle>
filled
double-headed
arrowstyle variable
10
8
6
4
2
0
-2
-4
-6
-8
-20
-15
-10
-5
0
5
10
15
20

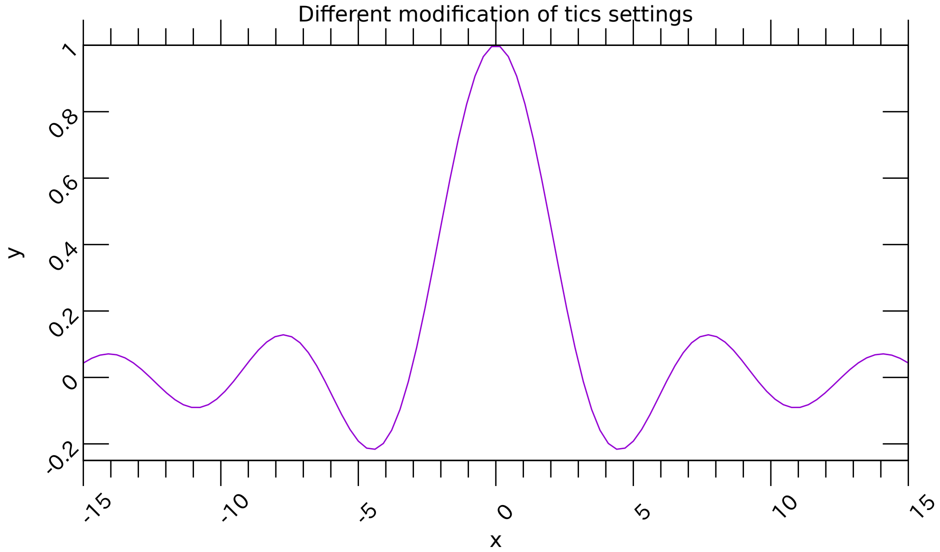

vtot(x,y)
3
2
1
0.5
0.2
0.1
0.05
0.02
0
-0.02
-0.05
-0.1
-0.2
-0.5
-1
-2
-3
-q
+q
10
5
0
-5
-10
-10
-5
0
5
10

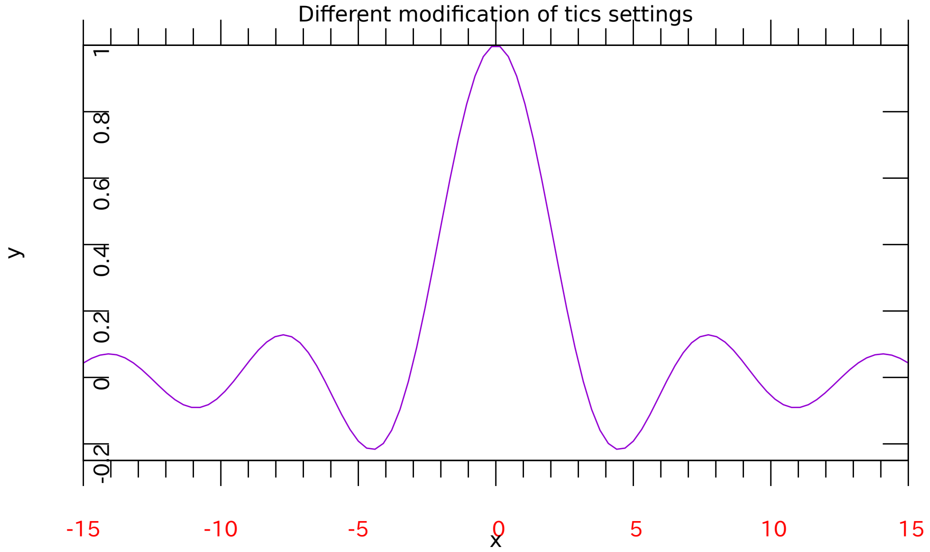

$equipo2
10
5
0
-5
-10
-10
-5
0
5
10

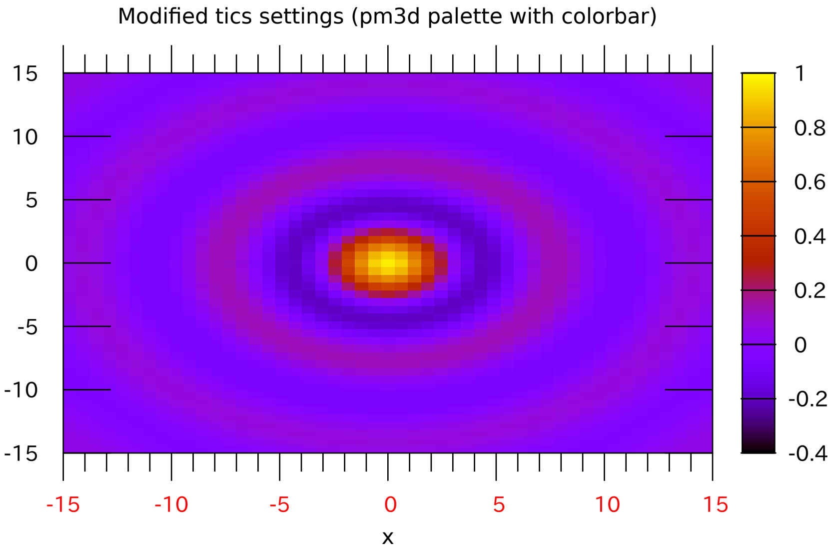

10
5
0
-5
-10
-10
-5
0
5
10
$field2xy u 1:2:(coef*dx1($1,$2)):(coef*dy1($1,$2))
$equipo2


fixed size arrowheads for very short vectors
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6


Default tics settings
1
0.8
0.6
0.4
0.2
0
-0.2
y
-15
-10
-5
0
5
10
15
x

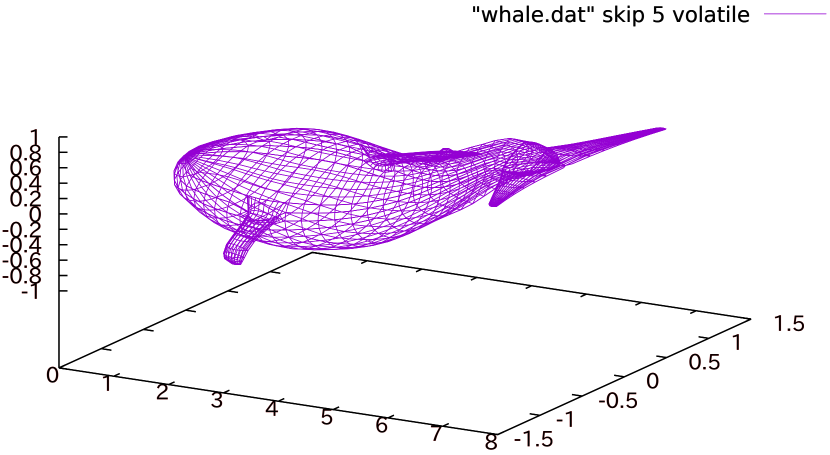

Different modification of tics settings
1
0.8
0.6
0.4
0.2
0
-0.2
y
-15
-10
-5
0
5
10
15
x

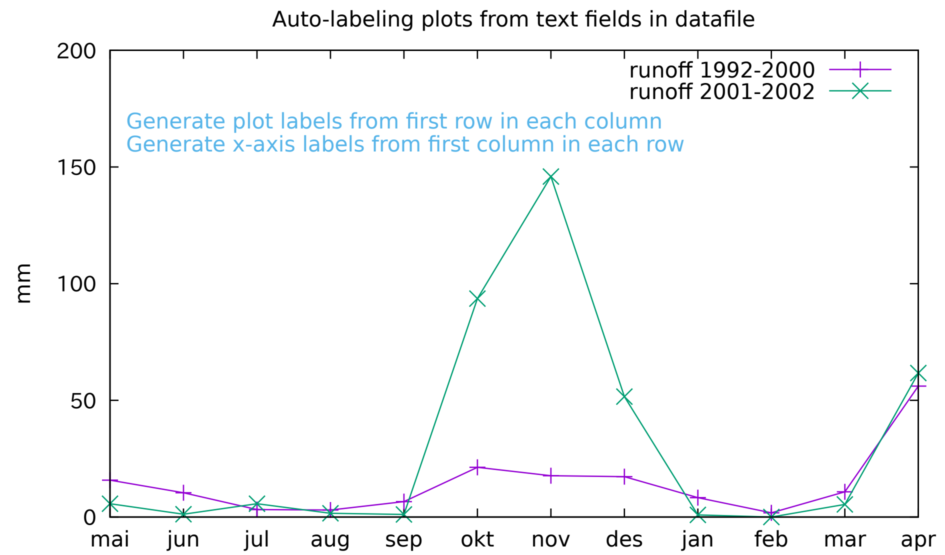

Different modification of tics settings
1
0.8
0.6
0.4
0.2
0
-0.2
y
-15
-10
-5
0
5
10
15
x



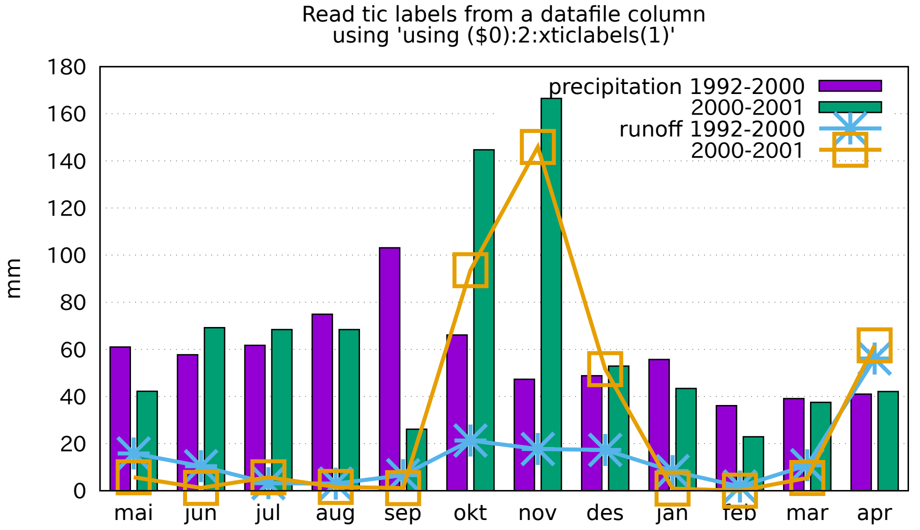
Modified tics settings (pm3d palette with colorbar)
15
10
5
0
-5
-10
-15
-15 -10 -5 0 5 10 15
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4

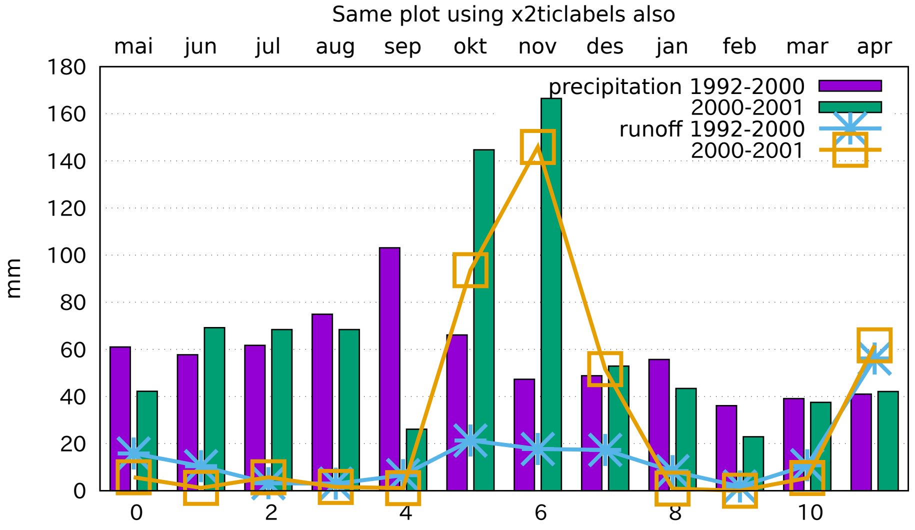
Nothing interesting here, just a unit test for volatile, skip, and refresh



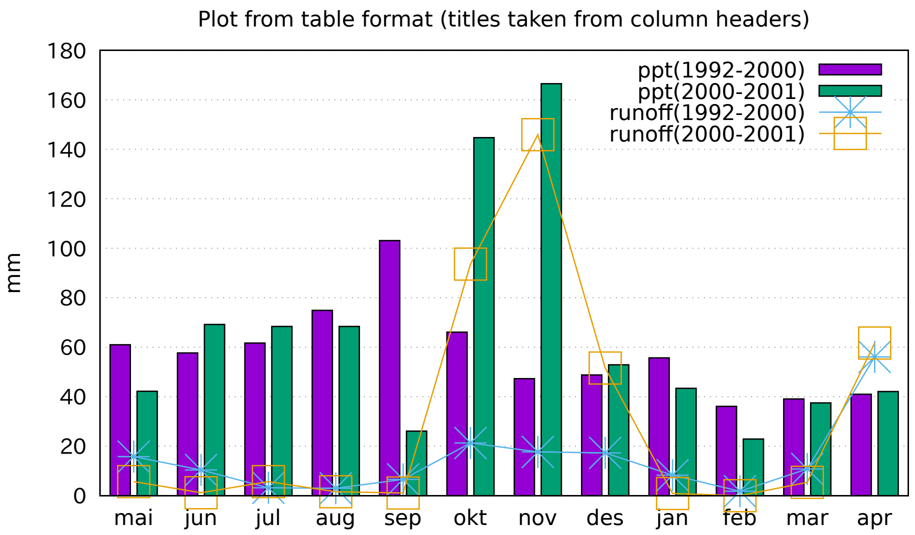


Auto-labeling plots from text fields in datafile
runoff 1992-2000
runoff 2001-2002
Generate plot labels from first row in each column
Generate x-axis labels from first column in each row
mm
200
150
100
50
0
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr

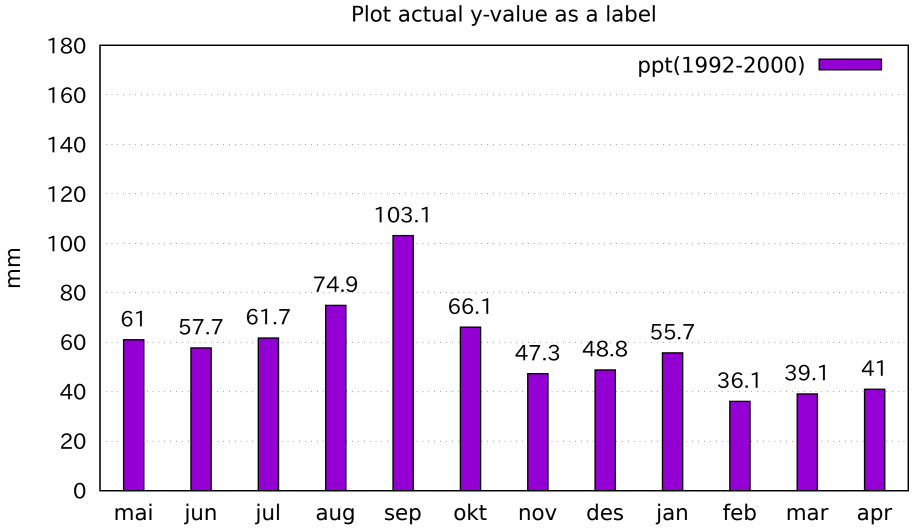

Read tic labels from a datafile column
using 'using ($0):2:xticlabels(1)'
mm
180
160
140
120
100
80
60
40
20
0
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr
precipitation 1992-2000
2000-2001
runoff 1992-2000
2000-2001

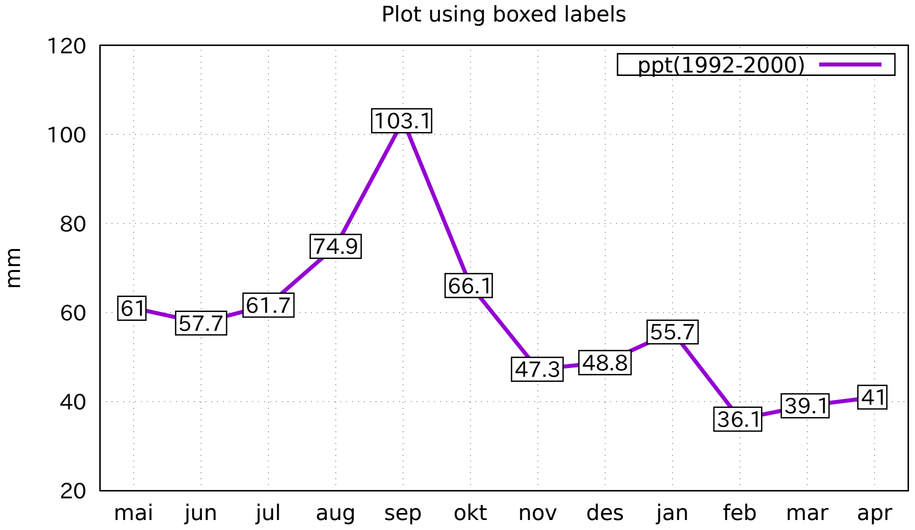


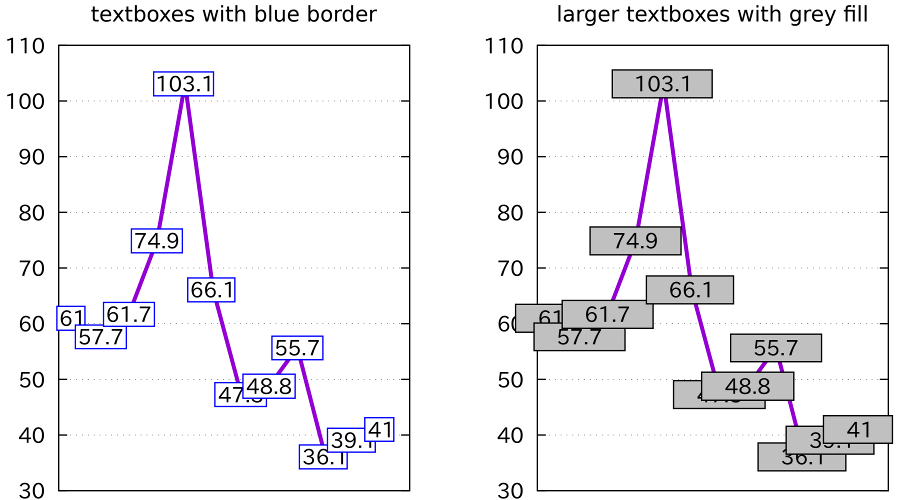
Same plot using x2ticlabels also
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr
180
160
140
120
100
80
60
40
20
0
mm
0
2
4
6
8
10
precipitation 1992-2000
2000-2001
runoff 1992-2000
2000-2001



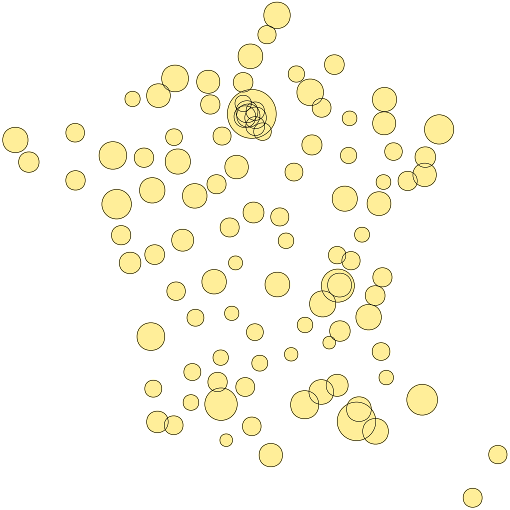
Plot from table format (titles taken from column headers)
mm
0
20
40
60
80
100
120
140
160
180
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr
ppt(1992-2000)
ppt(2000-2001)
runoff(1992-2000)
runoff(2000-2001)

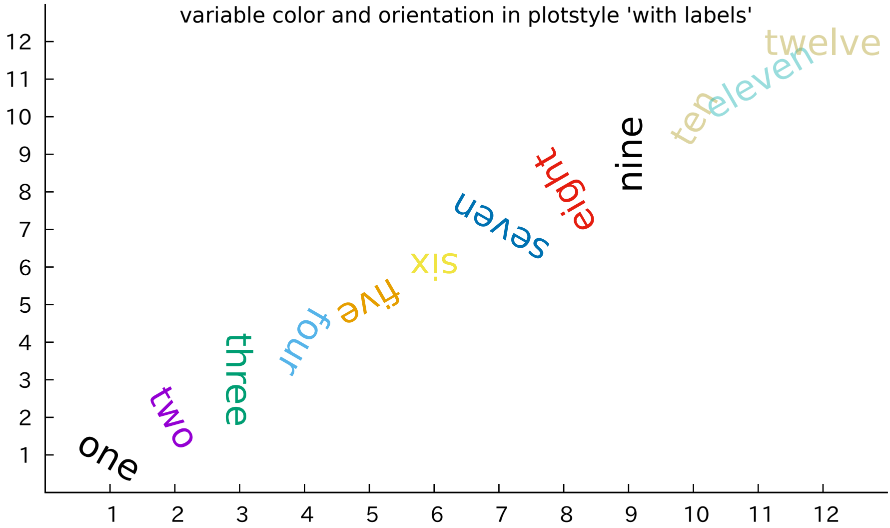

Plot actual y-value as a label
ppt(1992-2000)
mm
180
160
140
120
100
80
60
40
20
0
61
57.7
61.7
74.9
103.1
66.1
47.3
48.8
55.7
36.1
39.1
41
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr


Plot using boxed labels
ppt(1992-2000)
mm
120
100
80
60
40
20
mai
jun
jul
aug
sep
okt
nov
des
jan
feb
mar
apr
61
57.7
61.7
74.9
103.1
66.1
47.3
48.8
55.7
36.1
39.1
41

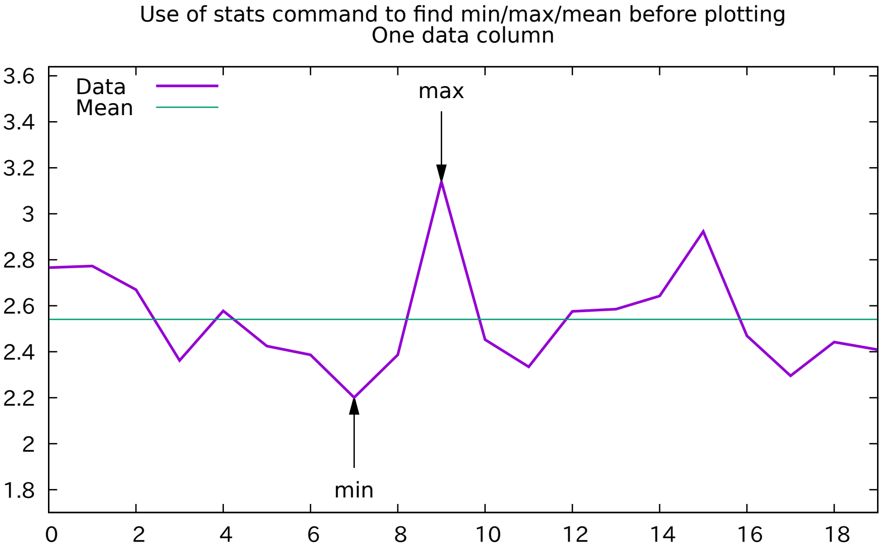

textboxes with blue border
larger textboxes with grey fill
110
100
90
80
70
60
50
40
30
103.1
74.9
66.1
61
61.7
57.7
55.7
48.8
41
36.1

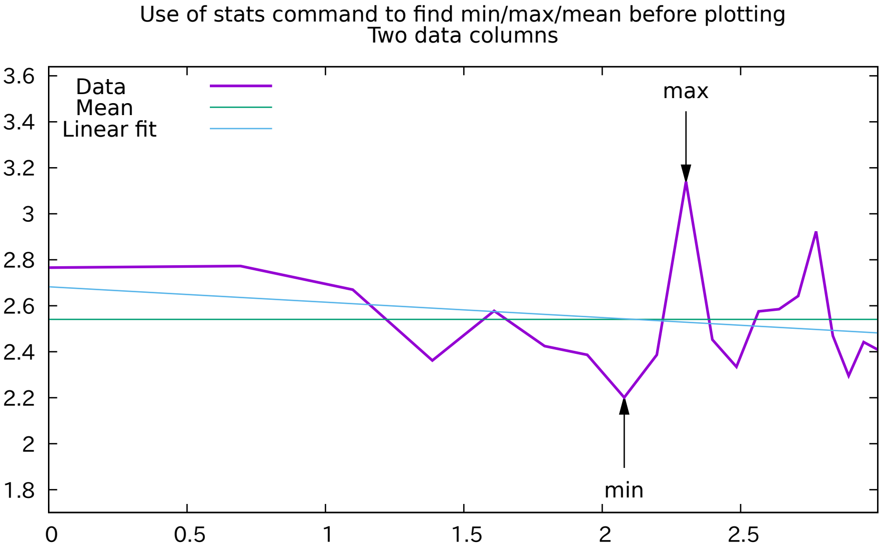
Hypertext is shown when the mouse is over a point

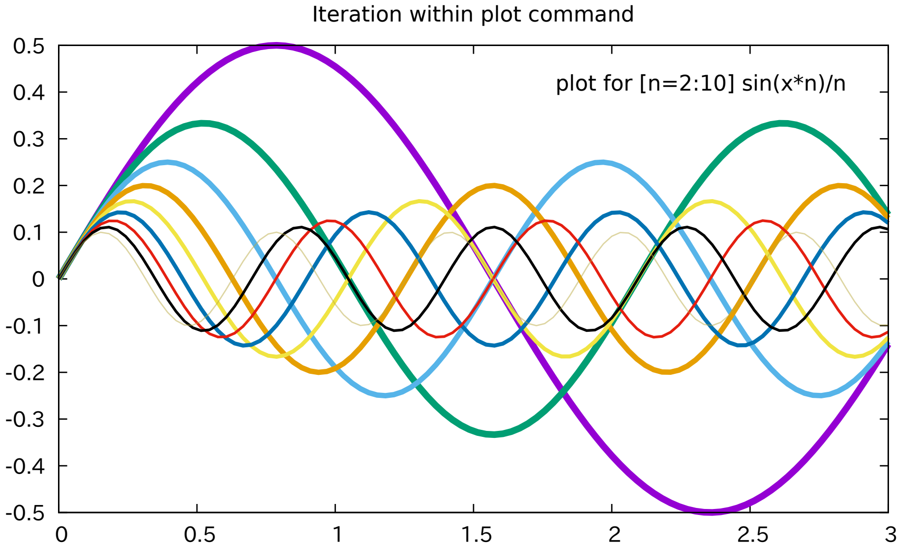


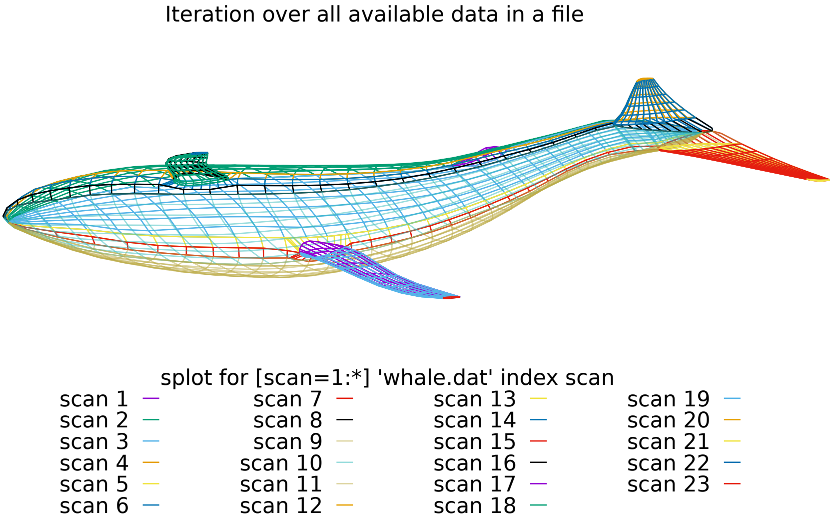


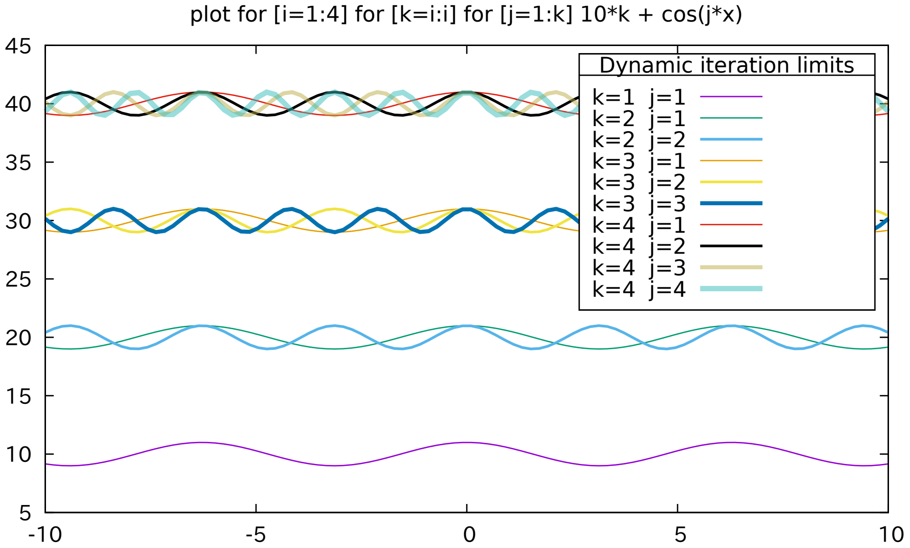
variable color and orientation in plotstyle 'with labels'
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
1
2
3
4
5
6
7
8
9
10
11
12

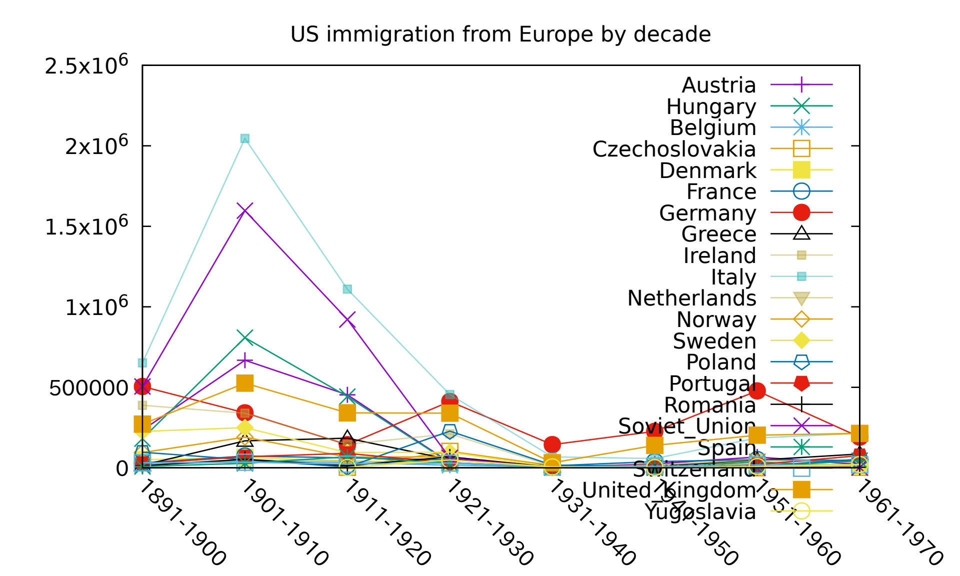

Use of stats command to find min/max/mean before plotting
One data column
Data
Mean
max
min
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
0
2
4
6
8
10
12
14
16
18

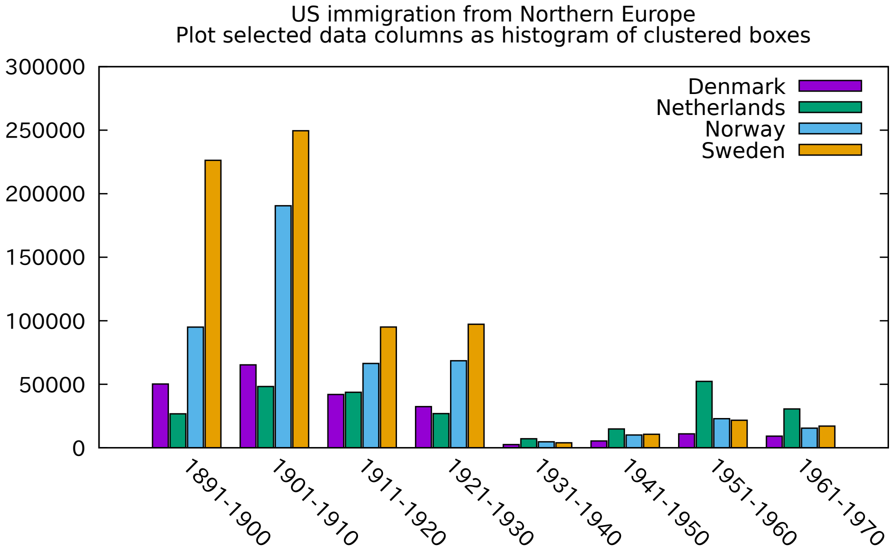

Use of stats command to find min/max/mean before plotting
Two data columns
Data
Mean
Linear fit
max
min
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
0
0.5
1
1.5
2
2.5

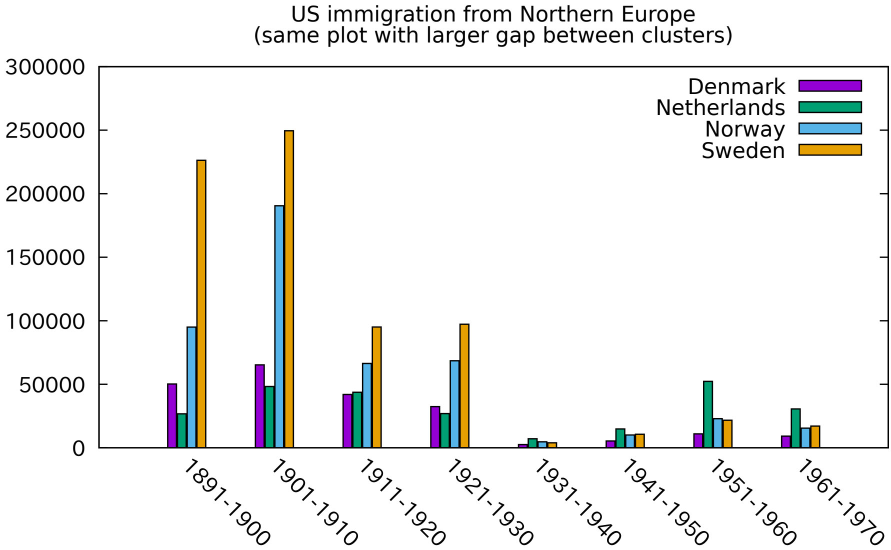

Iteration within plot command
plot for [n=2:10] sin(x*n)/n
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
0.5
1
1.5
2
2.5
3

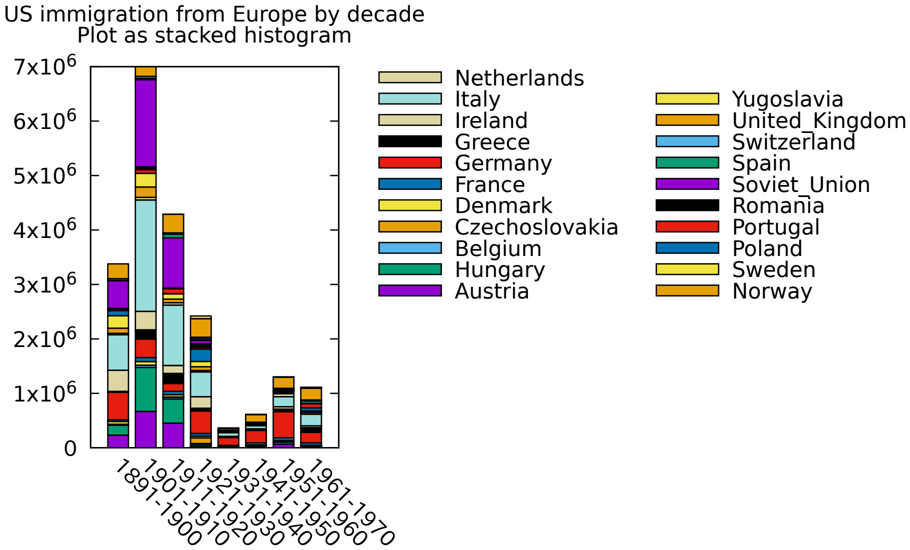

Iteration over all available data in a file
splot for [scan=1:*] 'whale.dat' index scan
scan 1
scan 2
scan 3
scan 4
scan 5
scan 6
scan 7
scan 8
scan 9
scan 10
scan 11
scan 12
scan 13
scan 14
scan 15
scan 16
scan 17
scan 18
scan 19
scan 20
scan 21
scan 22
scan 23

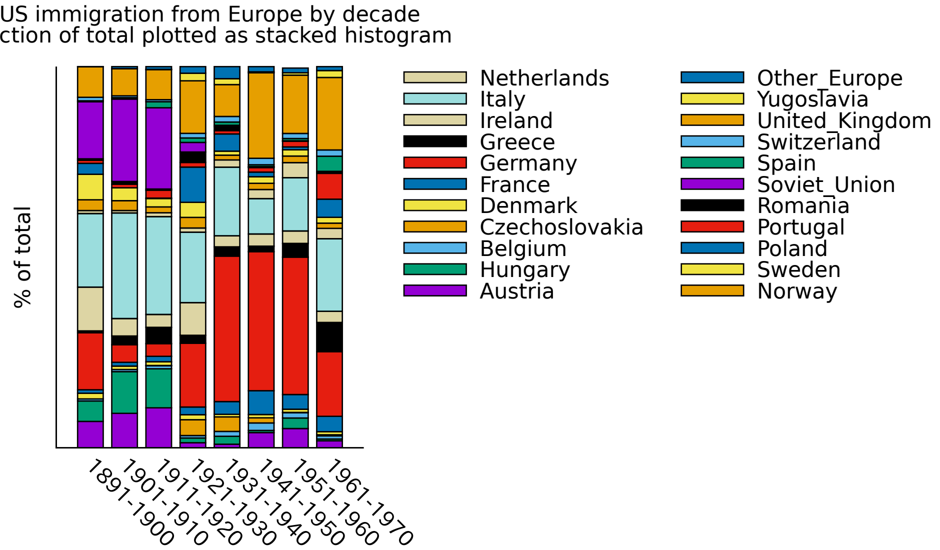

plot for [i=1:4] for [k=i:i] for [j=1:k] 10*k + cos(j*x)
Dynamic iteration limits
k=1 j=1
k=2 j=1
k=2 j=2
k=3 j=1
k=3 j=2
k=3 j=3
k=4 j=1
k=4 j=2
k=4 j=3
k=4 j=4
45
40
35
30
25
20
15
10
5
-10
-5
0
5
10

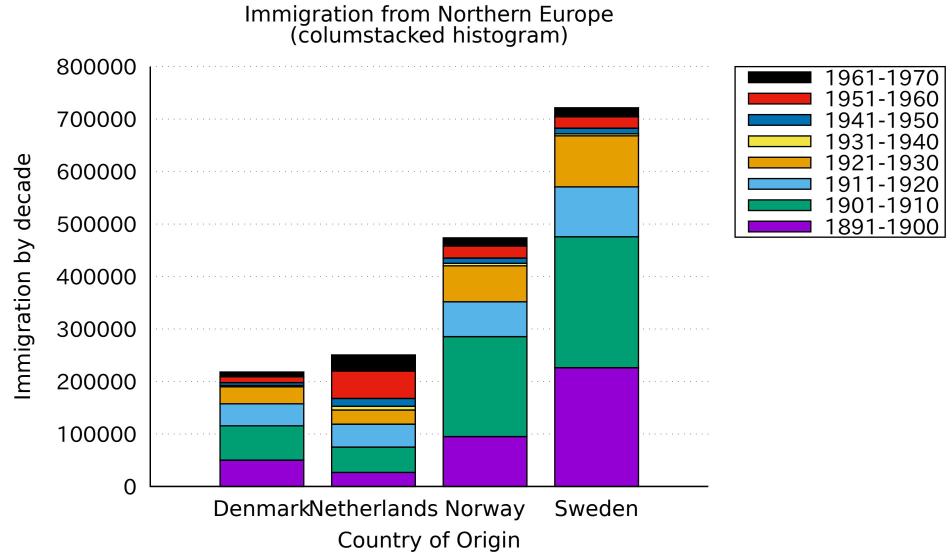

US immigration from Europe by decade
2.5x10^6
2x10^6
1.5x10^6
1x10^6
500000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Austria
Hungary
Belgium
Czechoslovakia
Denmark
France
Germany
Greece
Ireland
Italy
Netherlands
Norway
Sweden
Poland
Portugal
Romania
Soviet_Union
Spain
Switzerland
United_Kingdom
Yugoslavia

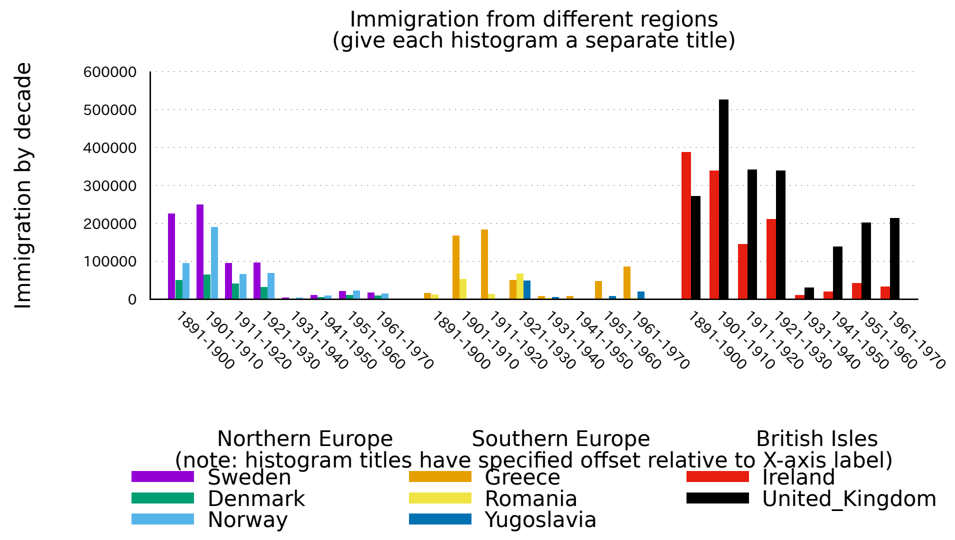

US immigration from Northern Europe
Plot selected data columns as histogram of clustered boxes
300000
250000
200000
150000
100000
50000
0
Denmark
Netherlands
Norway
Sweden
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970

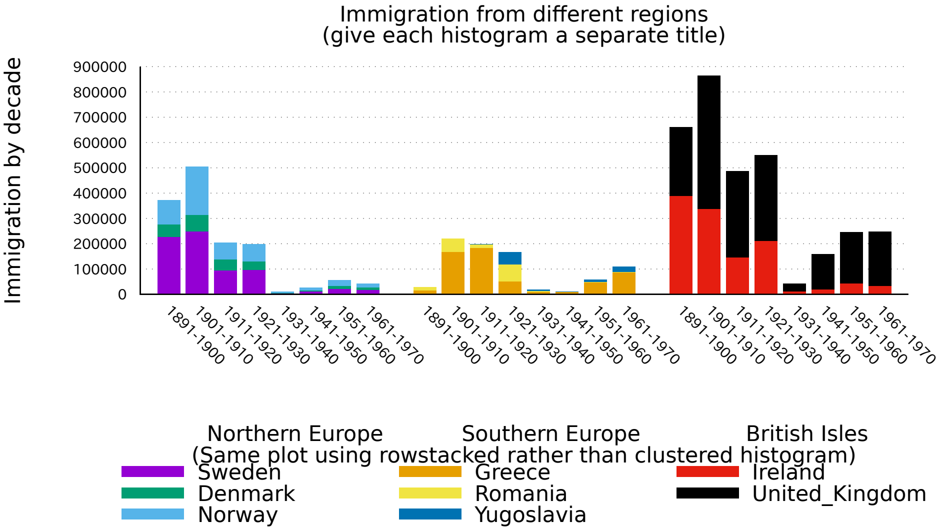

US immigration from Northern Europe
(same plot with larger gap between clusters)
300000
250000
200000
150000
100000
50000
0
Denmark
Netherlands
Norway
Sweden
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970


US immigration from Europe by decade
Plot as stacked histogram
7x10^6
6x10^6
5x10^6
4x10^6
3x10^6
2x10^6
1x10^6
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Netherlands
Italy
Ireland
Greece
Germany
France
Denmark
Czechoslovakia
Belgium
Hungary
Austria
Yugoslavia
United_Kingdom
Switzerland
Spain
Soviet_Union
Romania
Portugal
Poland
Sweden
Norway

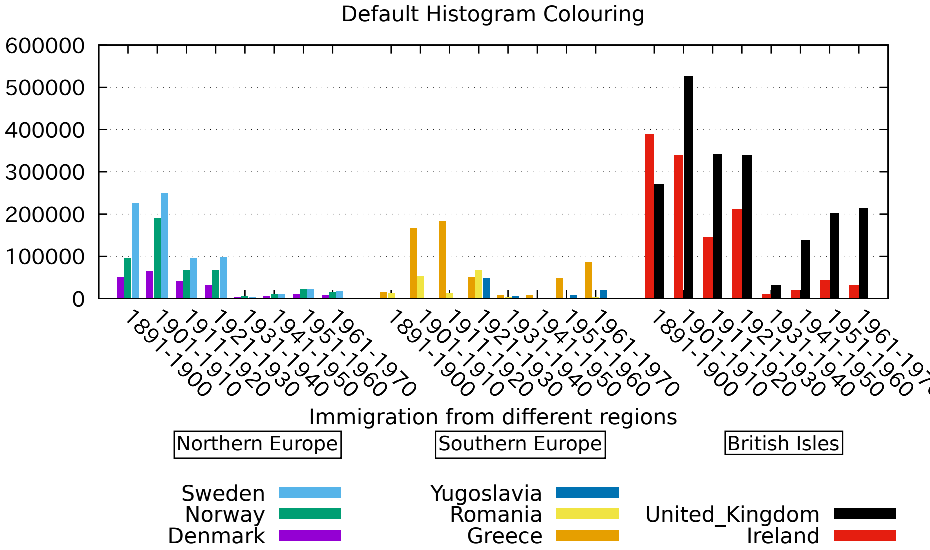

US immigration from Europe by decade
ction of total plotted as stacked histogram
% of total
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Netherlands
Italy
Ireland
Greece
Germany
France
Denmark
Czechoslovakia
Belgium
Hungary
Austria
Other_Europe
Yugoslavia
United_Kingdom
Switzerland
Spain
Soviet_Union
Romania
Portugal
Poland
Sweden
Norway

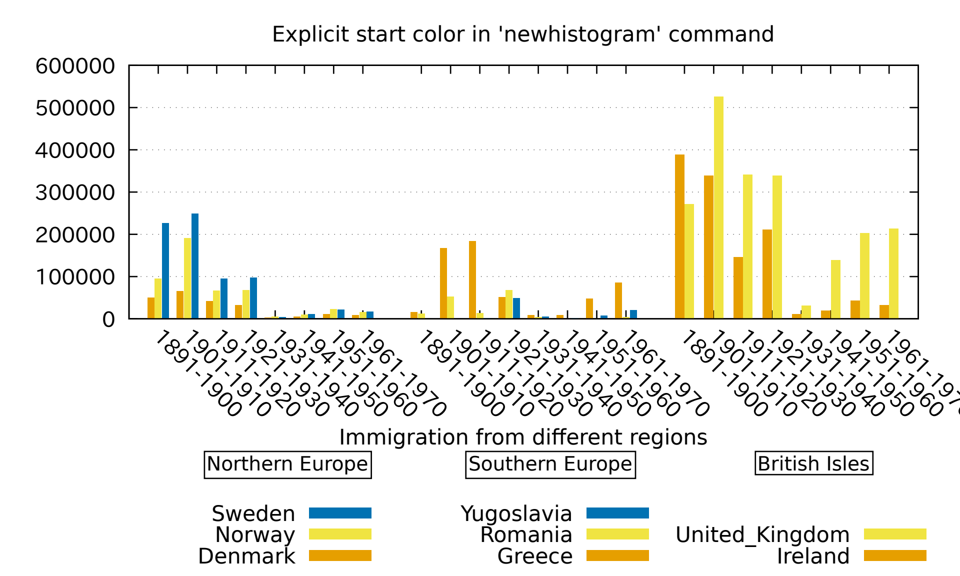

Immigration from Northern Europe
(columstacked histogram)
800000
700000
600000
500000
400000
300000
200000
100000
0
Immigration by decade
Denmark
Netherlands
Norway
Sweden
Country of Origin
1961-1970
1951-1960
1941-1950
1931-1940
1921-1930
1911-1920
1901-1910
1891-1900

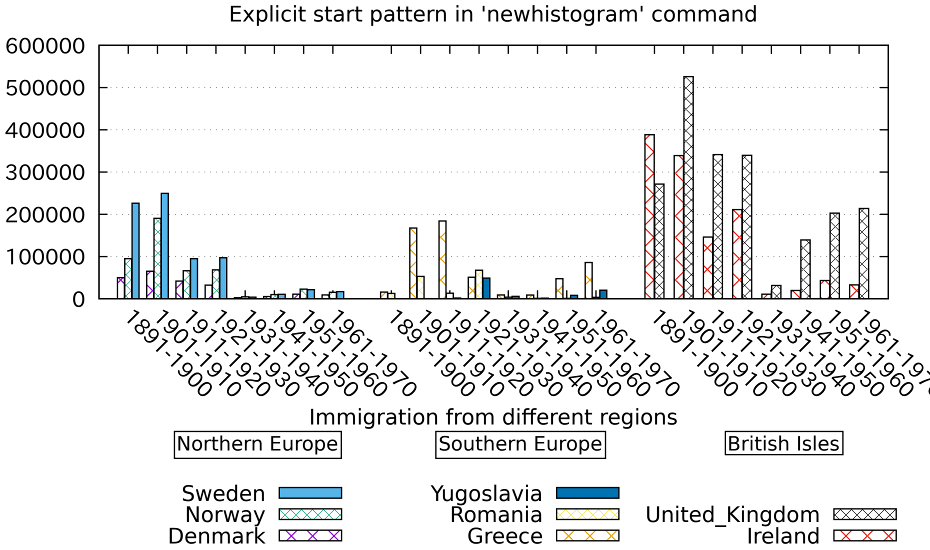

Immigration from different regions
(give each histogram a separate title)
Immigration by decade
600000
500000
400000
300000
200000
100000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Northern Europe
Southern Europe
British Isles
(note: histogram titles have specified offset relative to X-axis label)
Sweden
Denmark
Norway
Greece
Romania
Yugoslavia
Ireland
United_Kingdom



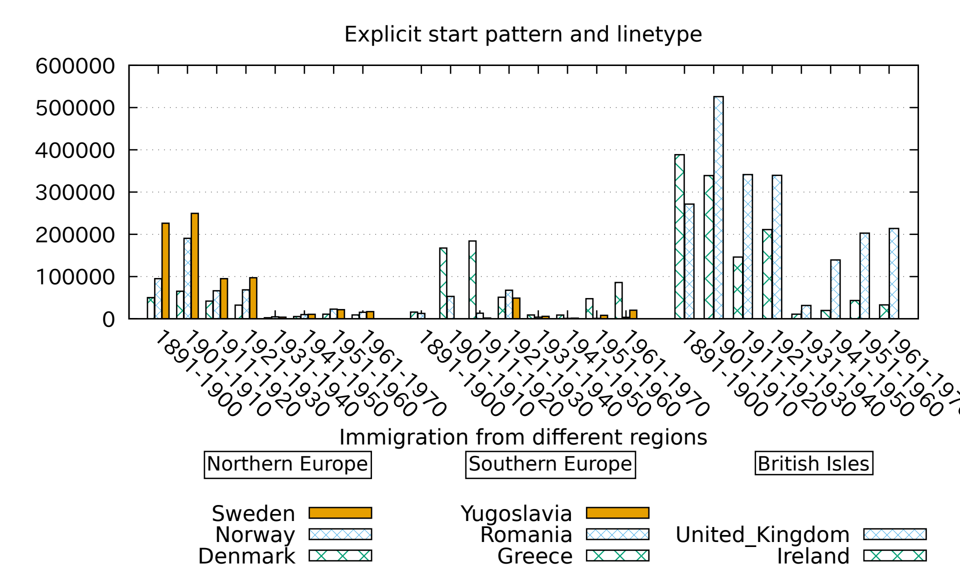
Immigration from different regions
(give each histogram a separate title)
Immigration by decade
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Northern Europe
Southern Europe
British Isles
(Same plot using rowstacked rather than clustered histogram)
Sweden
Denmark
Norway
Greece
Romania
Yugoslavia
Ireland
United_Kingdom


Default Histogram Colouring
600000
500000
400000
300000
200000
100000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration from different regions
Northern Europe
Southern Europe
British Isles
Sweden
Norway
Denmark
Yugoslavia
Romania
Greece
United_Kingdom
Ireland

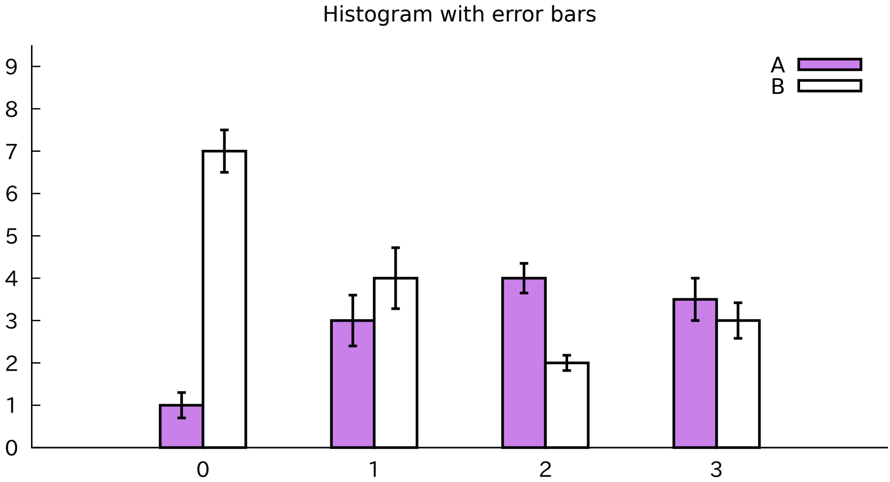

Explicit start color in 'newhistogram' command
600000
500000
400000
300000
200000
100000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration from different regions
Northern Europe
Southern Europe
British Isles
Sweden
Norway
Denmark
Yugoslavia
Romania
Greece
United_Kingdom
Ireland

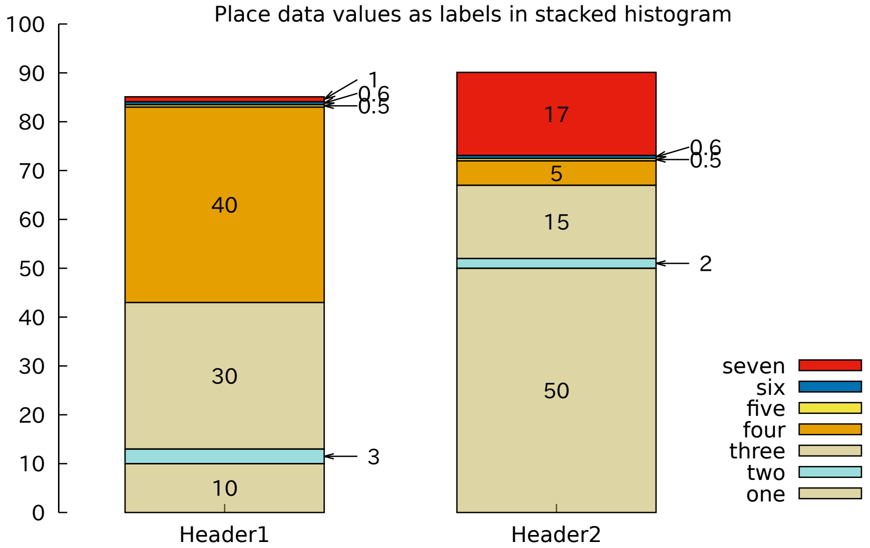


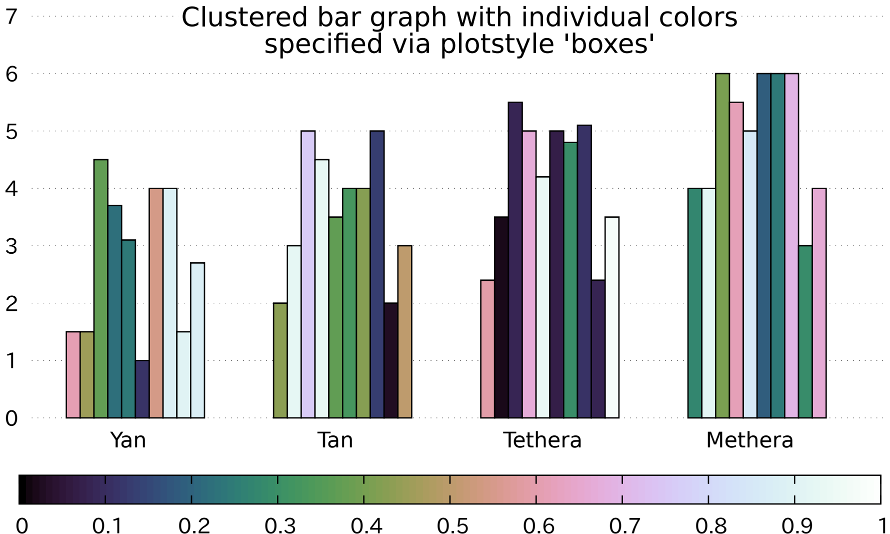
Explicit start pattern in 'newhistogram' command
600000
500000
400000
300000
200000
100000
0
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration from different regions
Northern Europe
Southern Europe
British Isles
Sweden
Norway
Denmark
Yugoslavia
Romania
Greece
United_Kingdom
Ireland

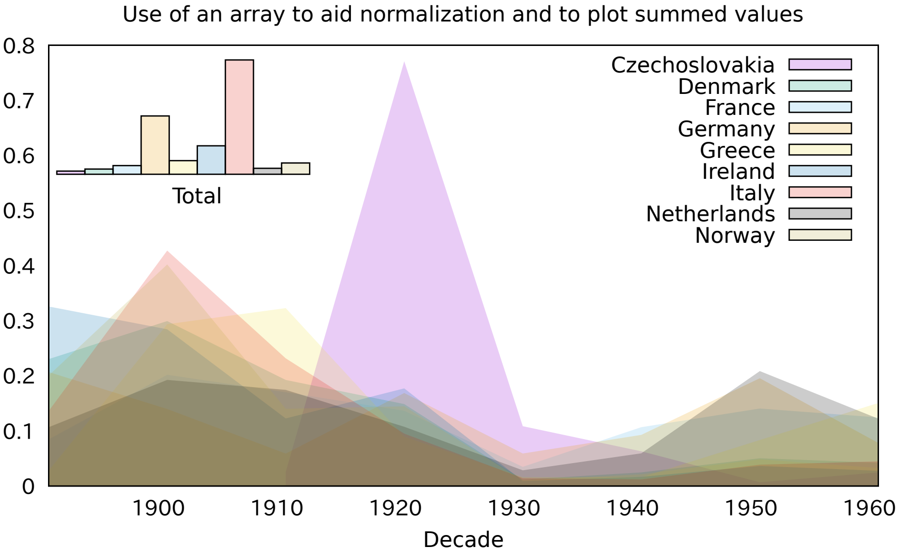


Explicit start pattern and linetype
0
100000
200000
300000
400000
500000
600000
1891-1900
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
Immigration from different regions
Northern Europe
Southern Europe
British Isles
Sweden
Norway
Denmark
Yugoslavia
Romania
Greece
United_Kingdom
Ireland

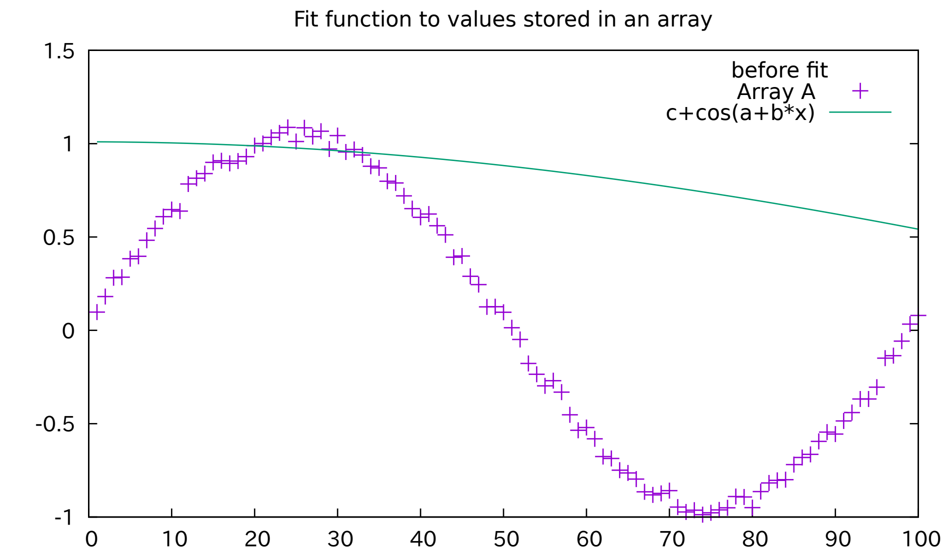

Histogram with error bars
A
B
0
1
2
3
4
5
6
7
8
9
0
1
2
3


Place data values as labels in stacked histogram
0
10
20
30
40
50
60
70
80
90
100
10
3
30
40
0.5
0.6
1
50
2
15
5
0.5
0.6
17
Header1
Header2
seven
six
five
four
three
two
one

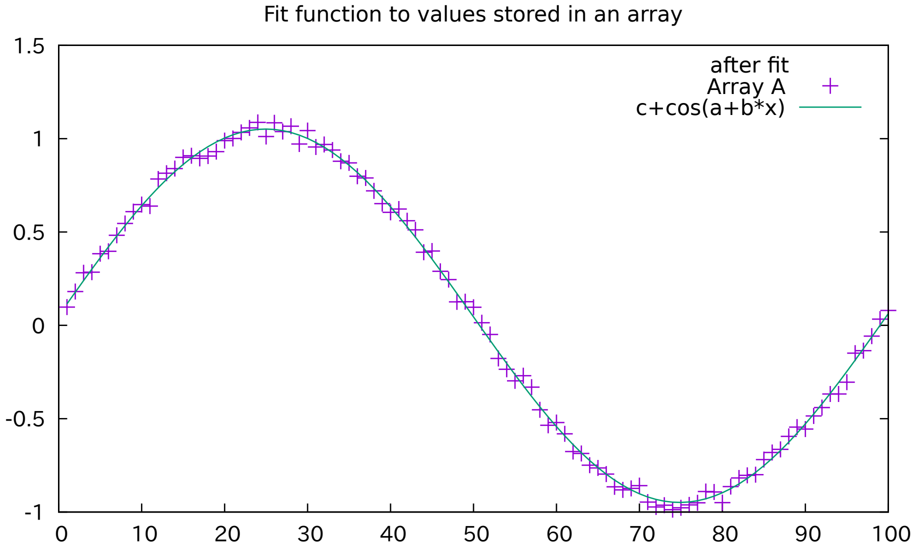

Clustered bar graph with individual colors
specified via plotstyle 'boxes'
7
6
5
4
3
2
1
0
Yan
Tan
Tethera
Methera
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

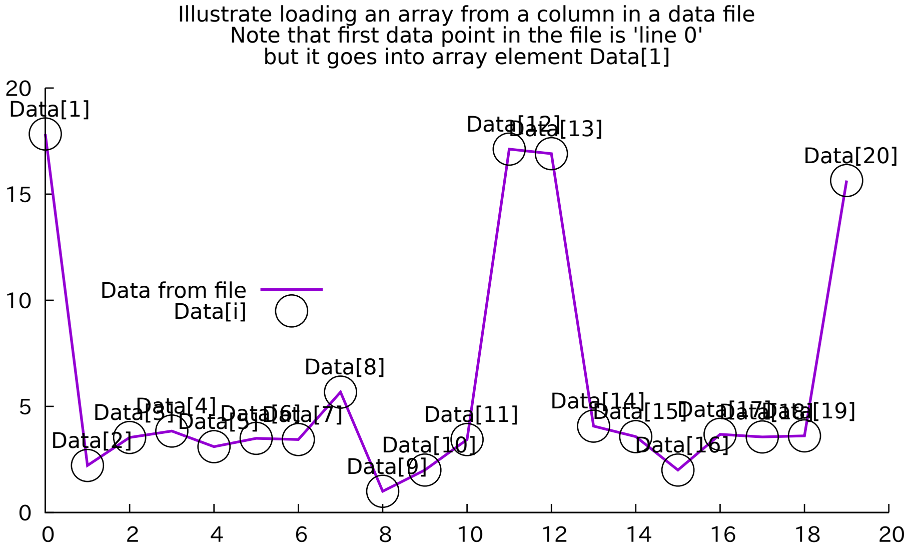


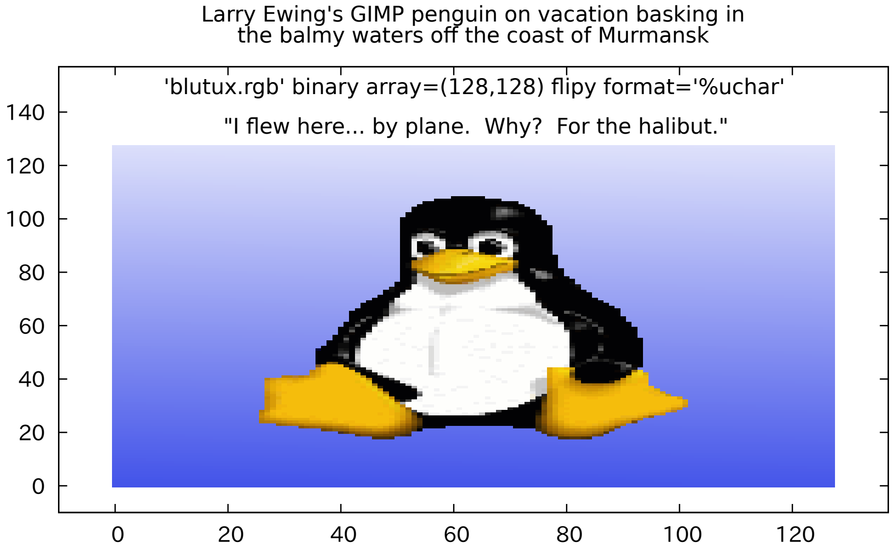
Use of an array to aid normalization and to plot summed values
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1900
1910
1920
1930
1940
1950
1960
Decade
Total
Czechoslovakia
Denmark
France
Germany
Greece
Ireland
Italy
Netherlands
Norway

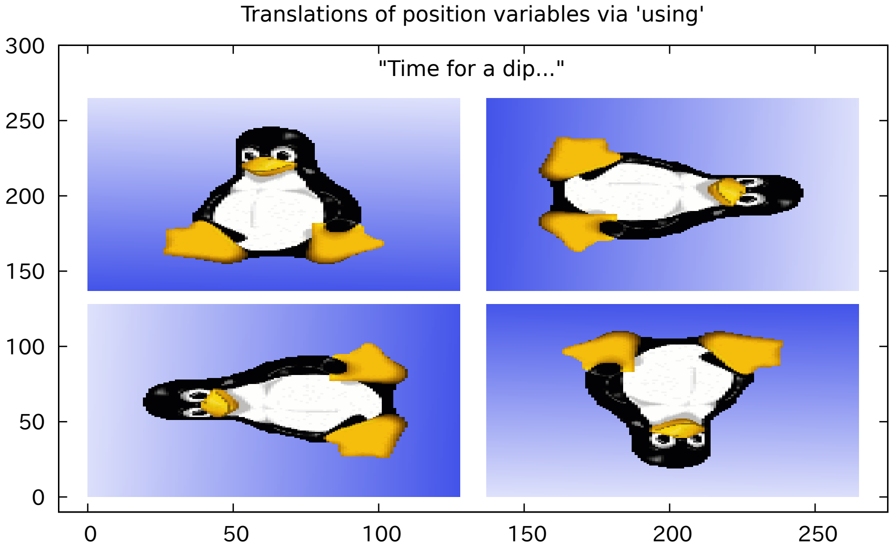


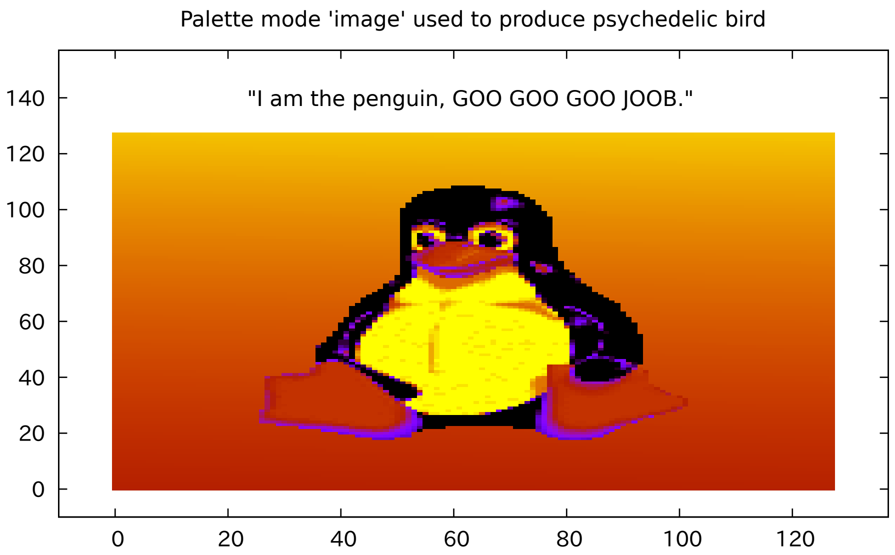
Fit function to values stored in an array
before fit
Array A
c+cos(a+b*x)
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
80
90
100

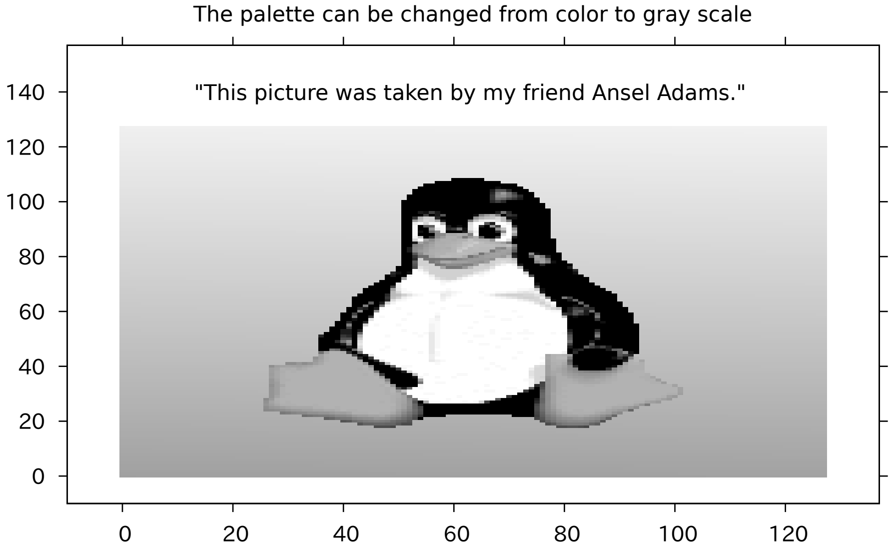

Fit function to values stored in an array
after fit
Array A
c+cos(a+b*x)
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
80
90
100

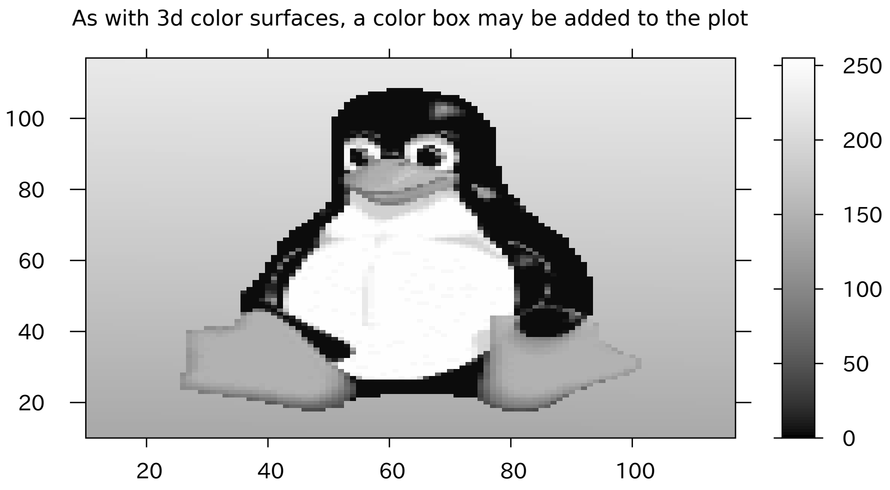

Illustrate loading an array from a column in a data file
Note that first data point in the file is 'line 0'
but it goes into array element Data[1]
Data from file
Data[i]
Data[1]
Data[2]
Data[3]
Data[4]
Data[5]
Data[6]
Data[7]
Data[8]
Data[9]
Data[10]
Data[11]
Data[12]
Data[13]
Data[14]
Data[15]
Data[16]
Data[17]
Data[18]
Data[19]
Data[20]
0
5
10
15
20
0
2
4
6
8
10
12
14
16
18
20

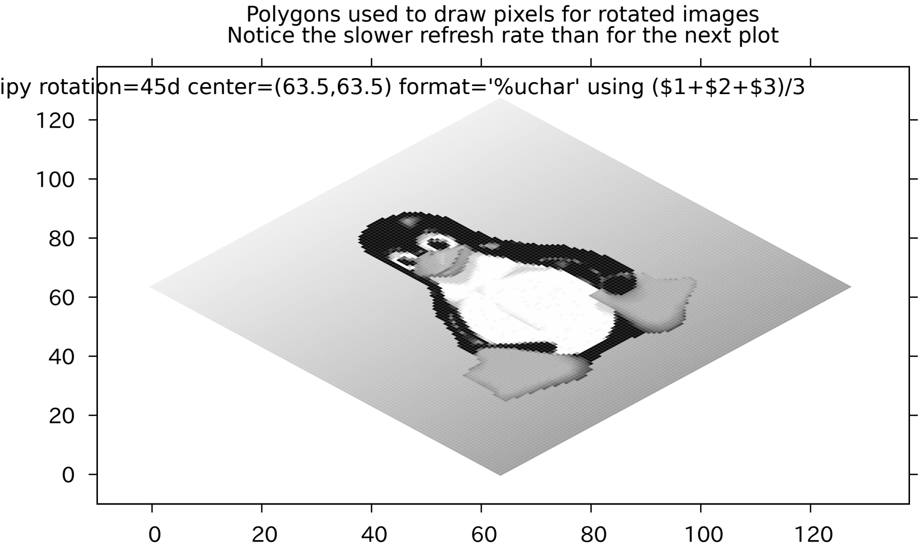

Larry Ewing's GIMP penguin on vacation basking in
the balmy waters off the coast of Murmansk
'blutux.rgb' binary array=(128,128) flipy format='%uchar'
"I flew here... by plane. Why? For the halibut."
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120


Translations of position variables via 'using'
"Time for a dip..."
300
250
200
150
100
50
0
0
50
100
150
200
250


Palette mode 'image' used to produce psychedelic bird
"I am the penguin, GOO GOO GOO JOOB."
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120

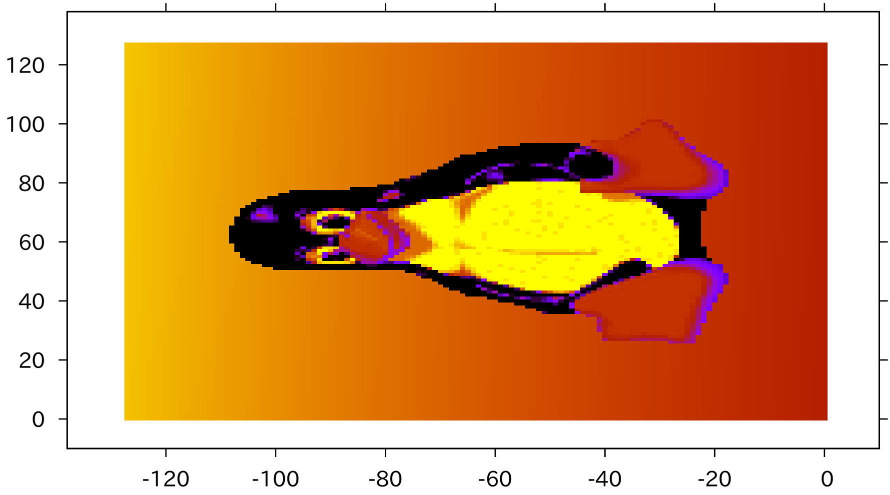

The palette can be changed from color to gray scale
"This picture was taken by my friend Ansel Adams."
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120

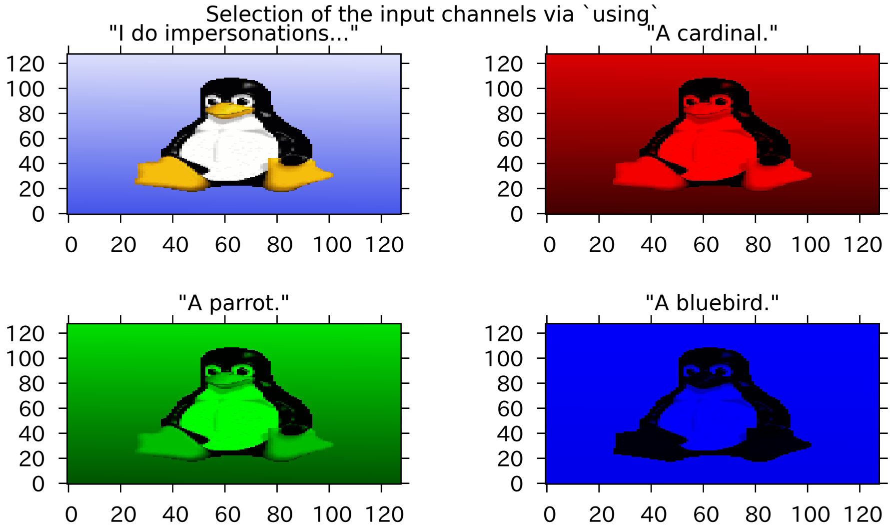

As with 3d color surfaces, a color box may be added to the plot
100
80
60
40
20
20
40
60
80
100
250
200
150
100
50
0

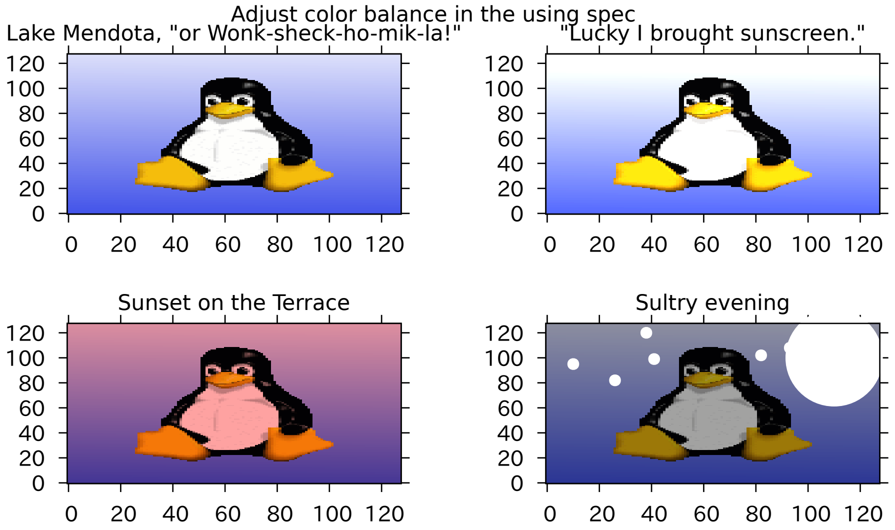


Polygons used to draw pixels for rotated images
Notice the slower refresh rate than for the next plot
ipy rotation=45d center=(63.5,63.5) format='%uchar' using ($1+$2+$3)/3
0
20
40
60
80
100
120
0
20
40
60
80
100
120

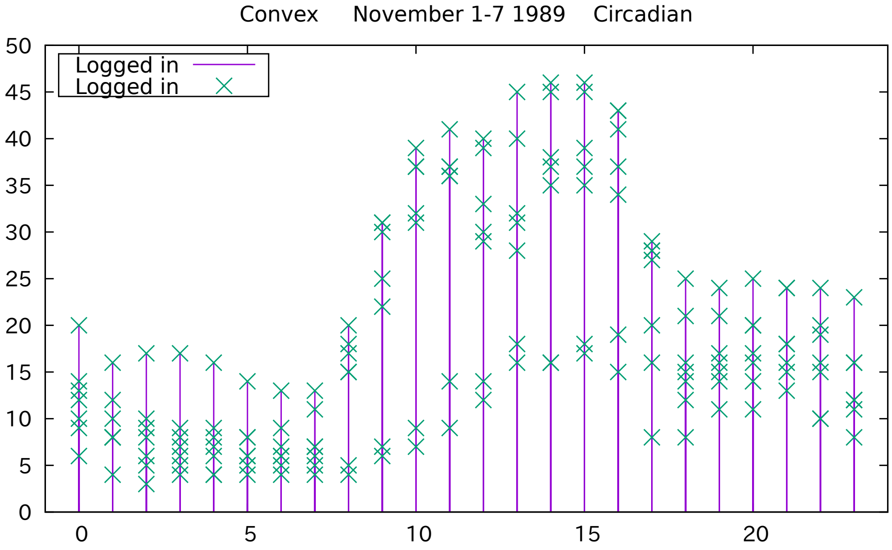
Terminal image routine used to draw plot rotated about origin
Notice the faster refresh rate than for the previous plot

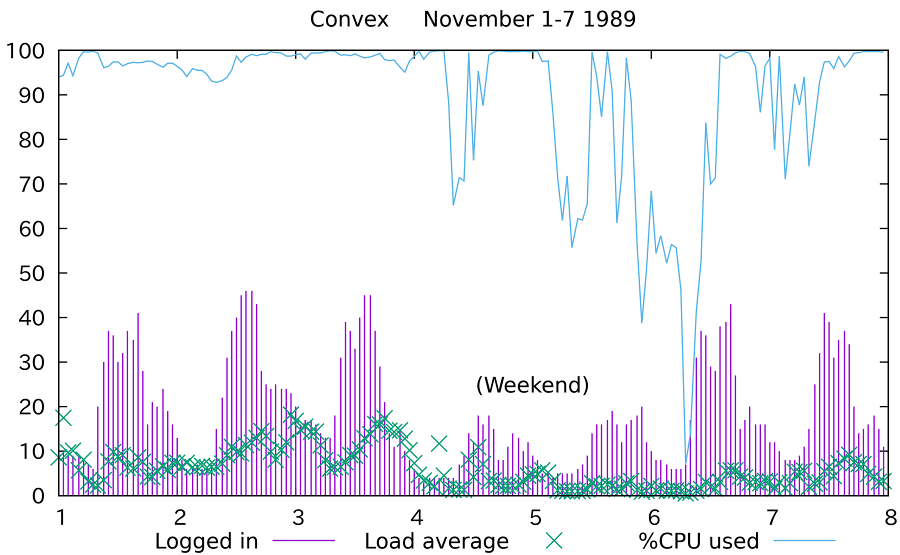


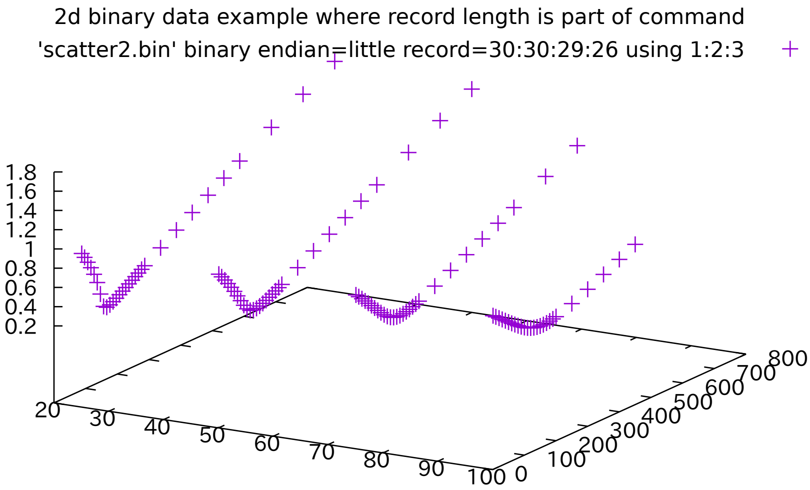

Selection of the input channels via `using`
"I do impersonations..."
"A cardinal."
"A parrot."
"A bluebird."
0 20 40 60 80 100 120
0 20 40 60 80 100 120

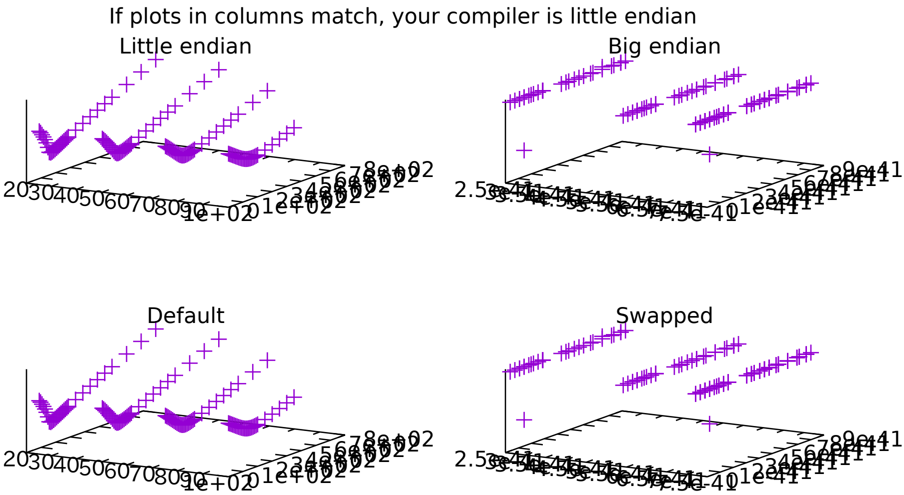


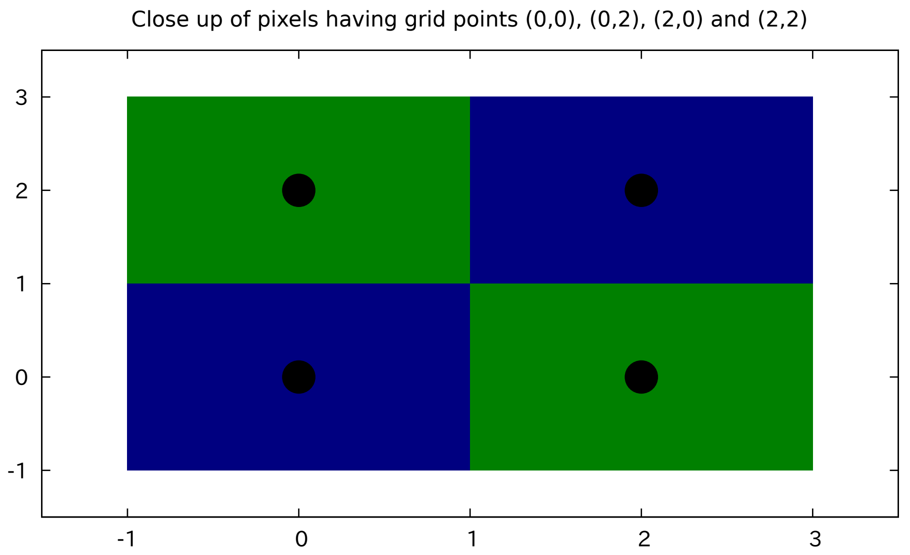
Adjust color balance in the using spec
Lake Mendota, "or Wonk-sheck-ho-mik-la!"
"Lucky I brought sunscreen."
Sunset on the Terrace
Sultry evening
0
20
40
60
80
100
120


Convex November 1-7 1989 Circadian
Logged in
Logged in
0
5
10
15
20
25
30
35
40
45
50
0
5
10
15
20


Convex November 1-7 1989
100
90
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
(Weekend)
Logged in
Load average
%CPU used

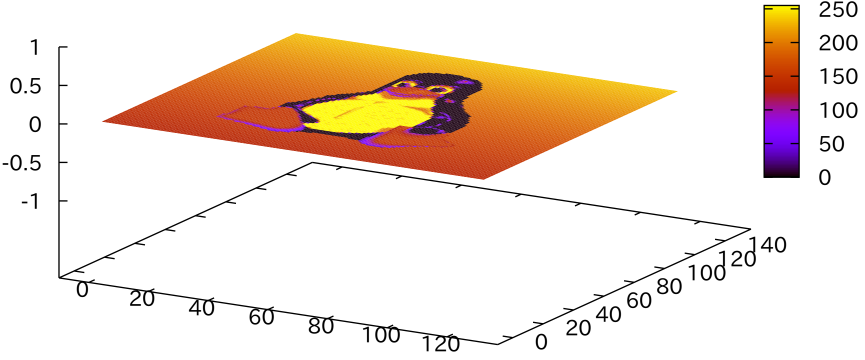

2d binary data example where record length is part of command
'scatter2.bin' binary endian=little record=30:30:29:26 using 1:2:3
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800


If plots in columns match, your compiler is little endian
Little endian
Big endian
Default
Swapped


Close up of pixels having grid points (0,0), (0,2), (2,0) and (2,2)
3
2
1
0
-1
-1
0
1
2
3

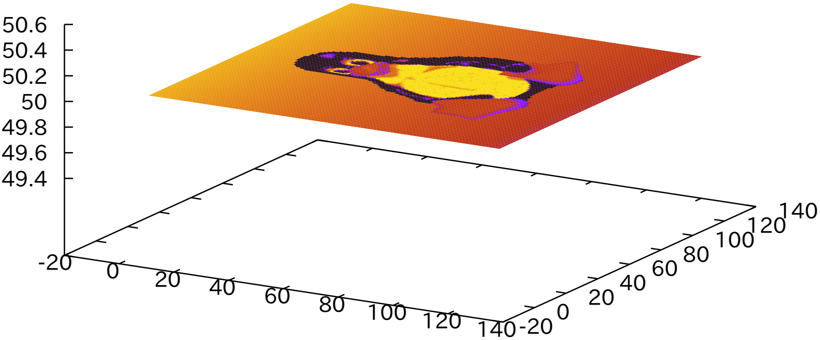
Simple extension of a two dimensional image into three dimensions

array=(128,128) flip=y format='%uchar%uchar%uchar' using ($1+$2+$3)/3



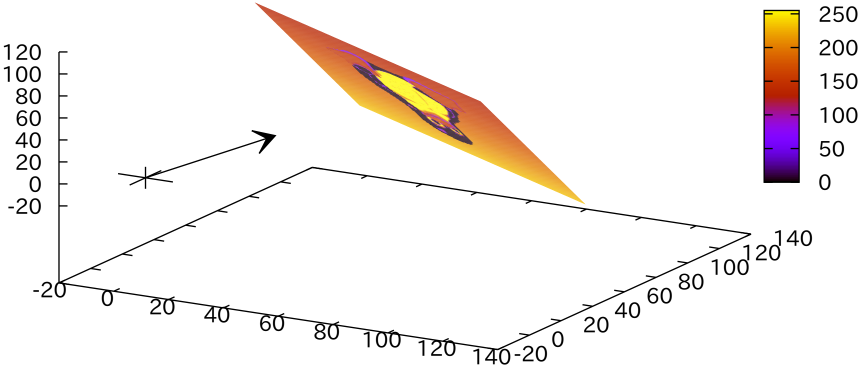
Orientation operations from 'plot' also apply to to 'splot'

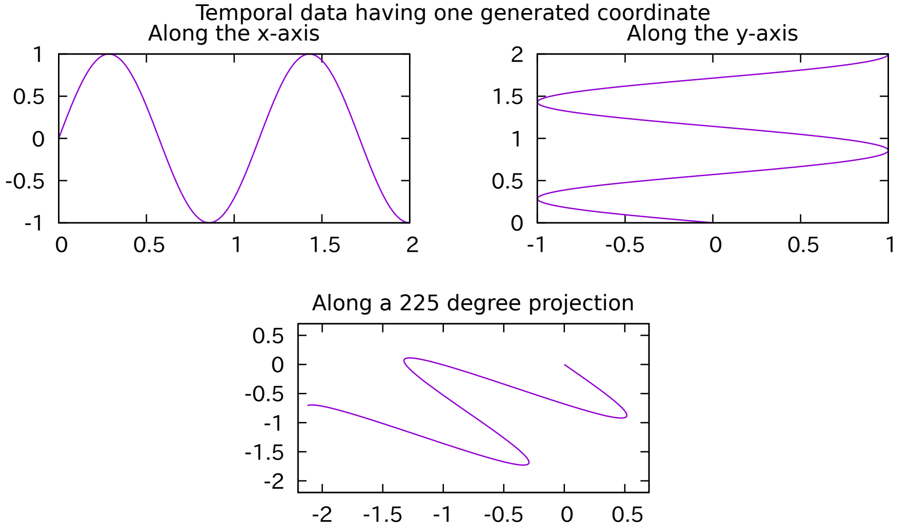
d center = (63.5,63.5,50) format='%uchar%uchar%uchar' using ($1+$2+$3)



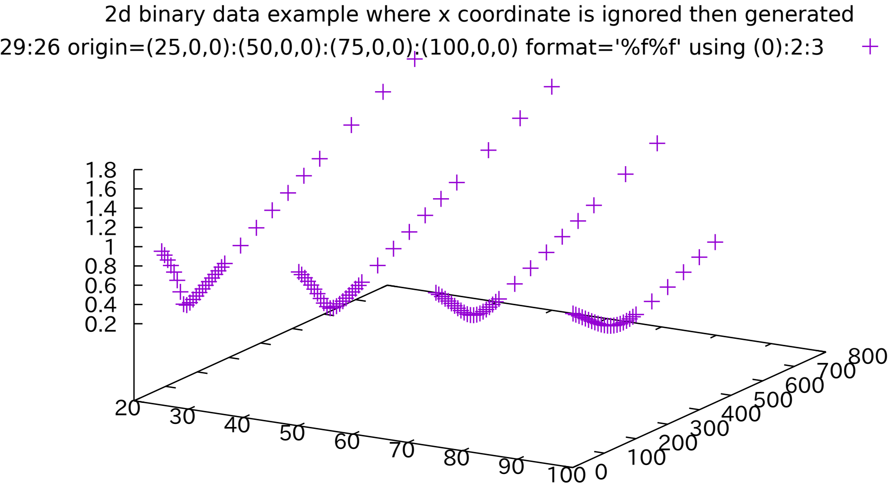
The key word 'perpendicular' applies only to 'splot'

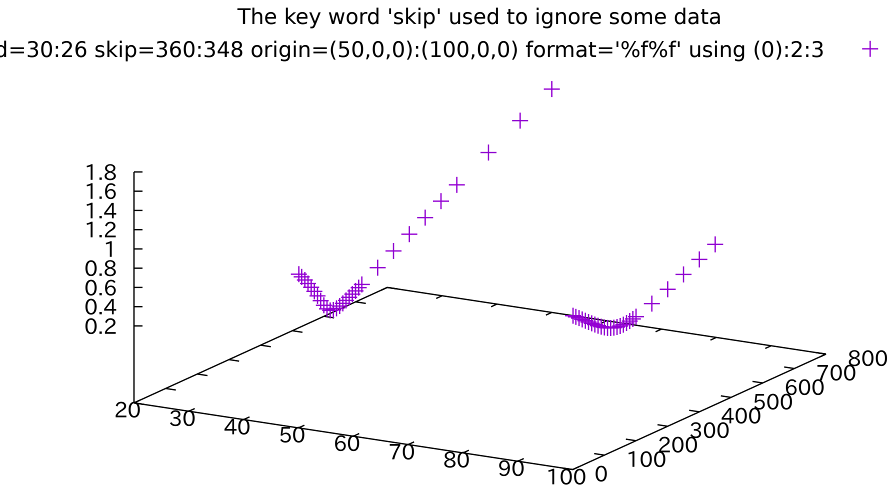
5,63.5,50) perp=(1,1,1) format='%uchar%uchar%uchar' using ($1+$2+$3)/3



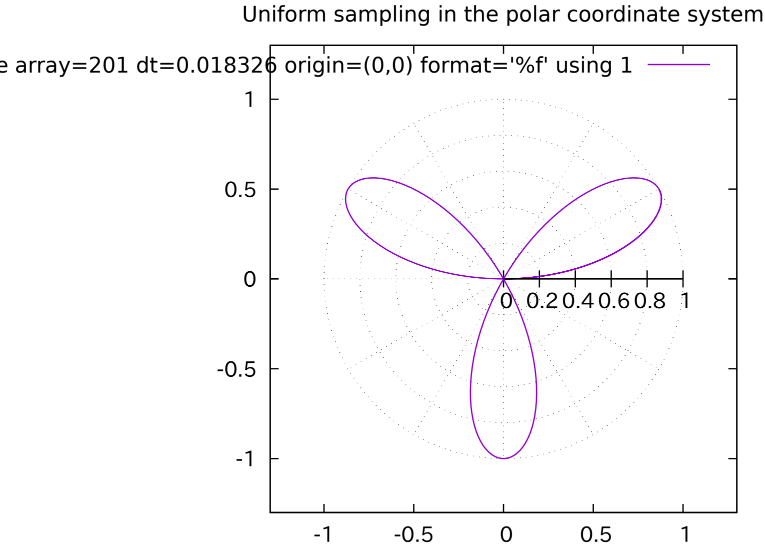


Temporal data having one generated coordinate
Along the x-axis
Along the y-axis
Along a 225 degree projection
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2
1.5
1
0.5
0
-1
-0.5
0
0.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5

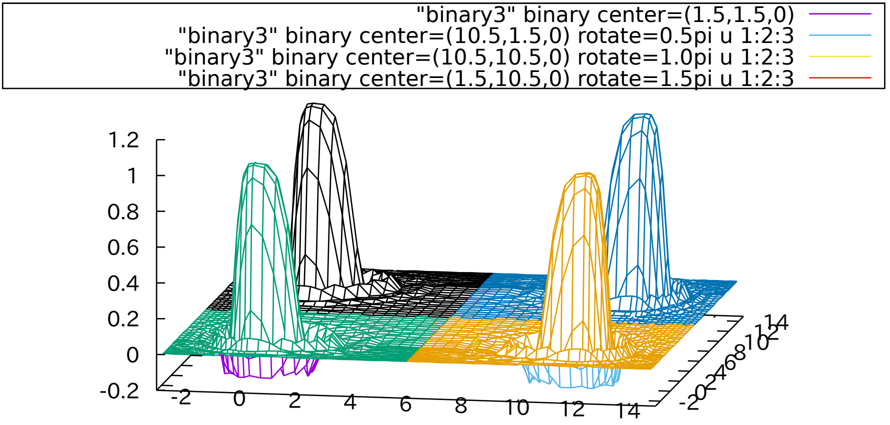

2d binary data example where x coordinate is ignored then generated
29:26 origin=(25,0,0):(50,0,0):(75,0,0):(100,0,0) format='%f%f' using (0):2:3
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800

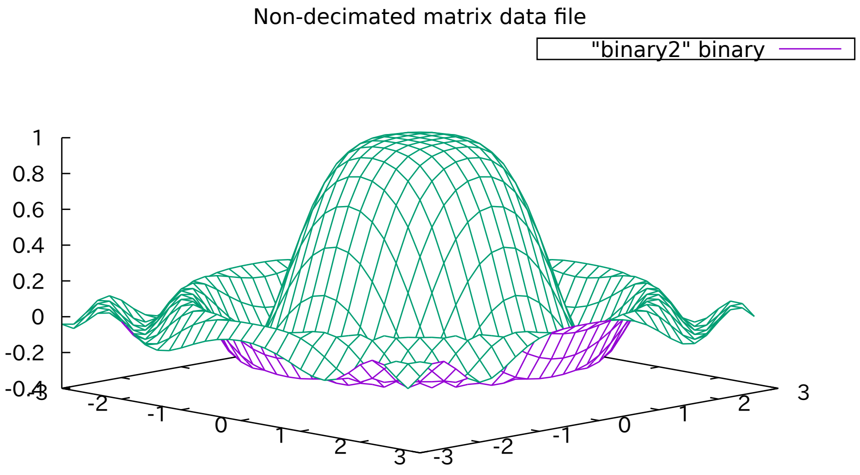


The key word 'skip' used to ignore some data
d=30:26 skip=360:348 origin=(50,0,0):(100,0,0) format='%f%f' using (0):2:3
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800

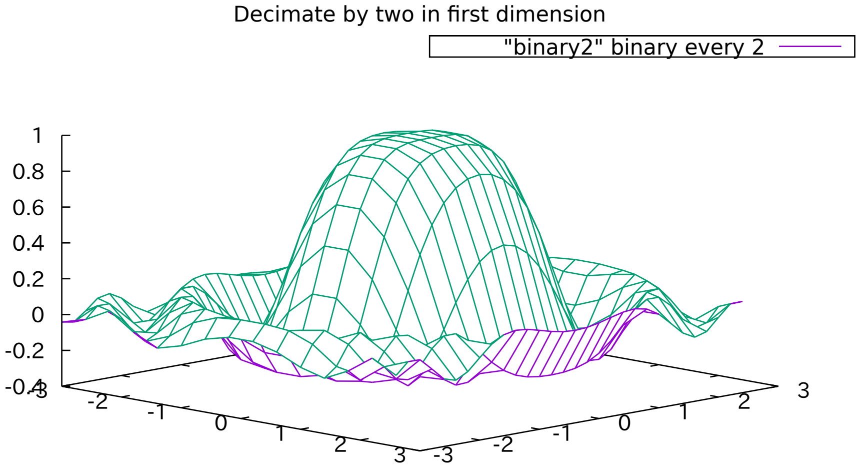

Uniform sampling in the polar coordinate system
e array=201 dt=0.018326 origin=(0,0) format='%f' using 1
1
0.5
0
-0.5
-1
0 0.2 0.4 0.6 0.8 1
-1 -0.5 0 0.5 1

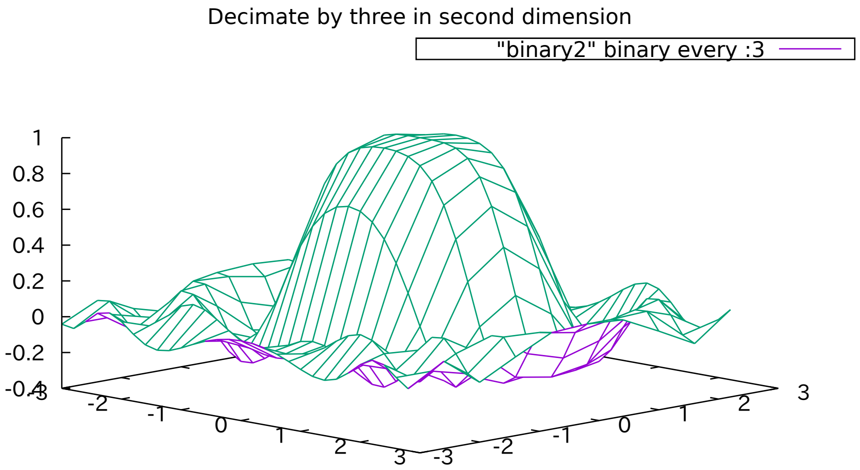
Matrix binary data (gnuplot binary) translated



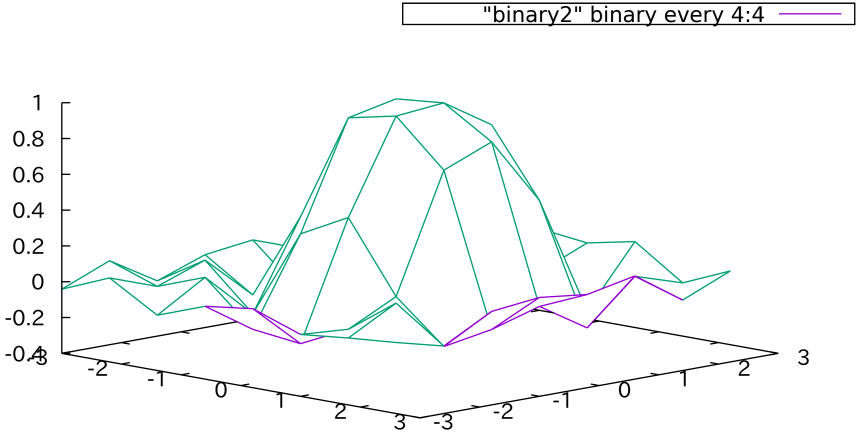


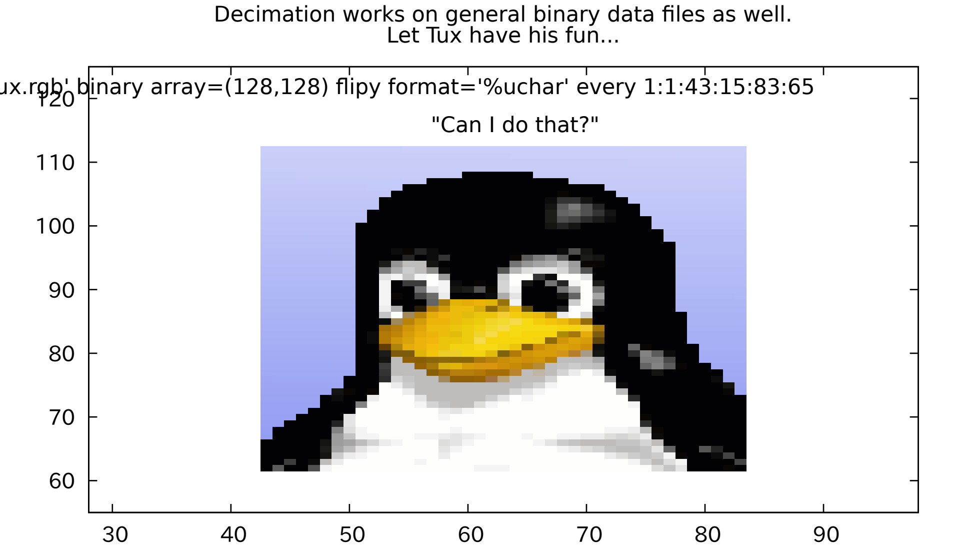
Non-decimated matrix data file
"binary2" binary
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-3
-2
-1
0
1
2
3
-3
-2
-1
0
1
2
3


Decimate by two in first dimension
"binary2" binary every 2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-3
-2
-1
0
1
2
3
-3
-2
-1
0
1
2
3

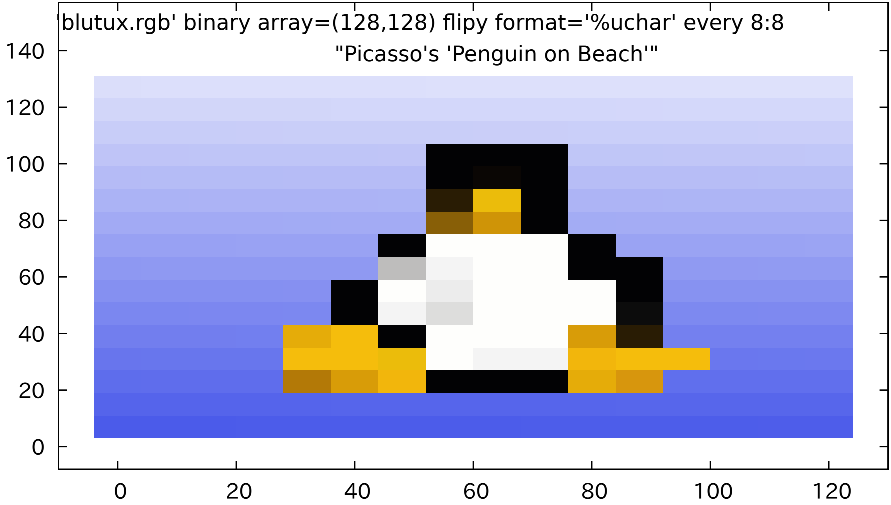

Decimate by three in second dimension
"binary2" binary every :3
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-3
-2
-1
0
1
2
3
-3
-2
-1
0
1
2
3

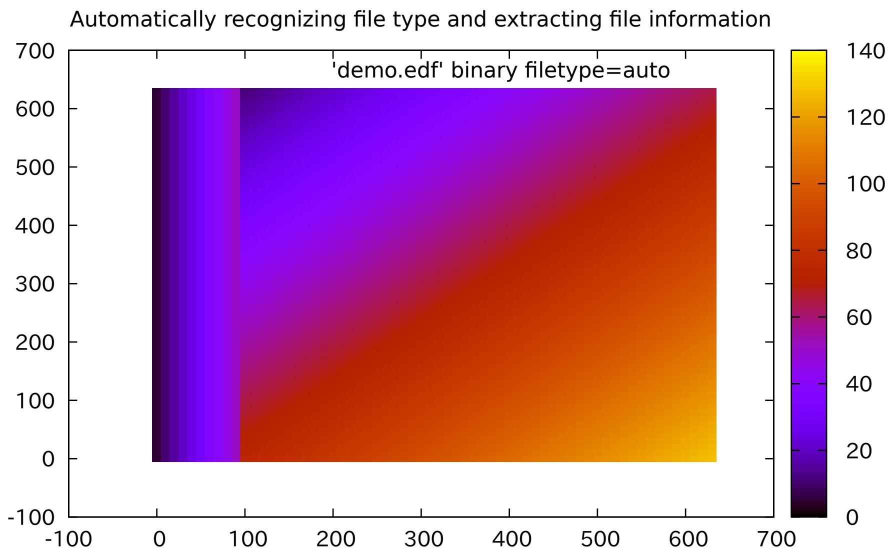
Decimate by four in both dimensions

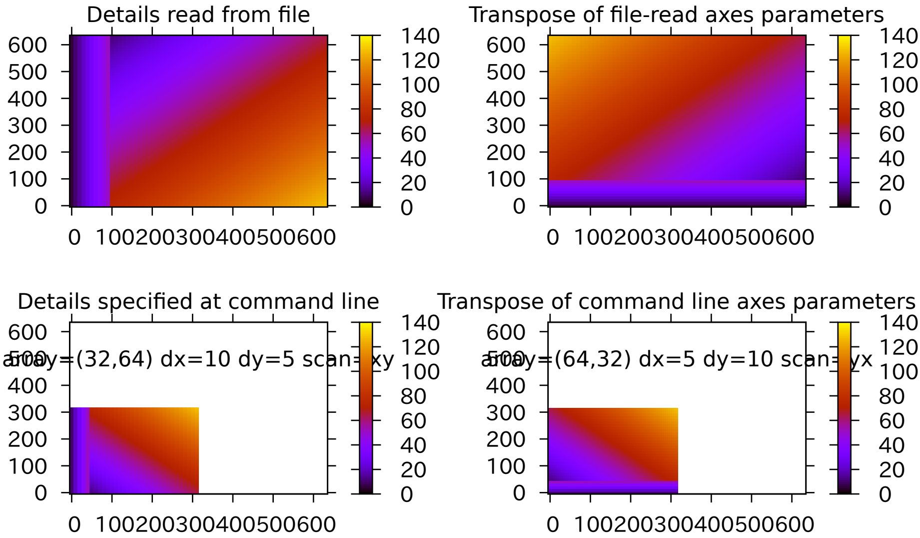


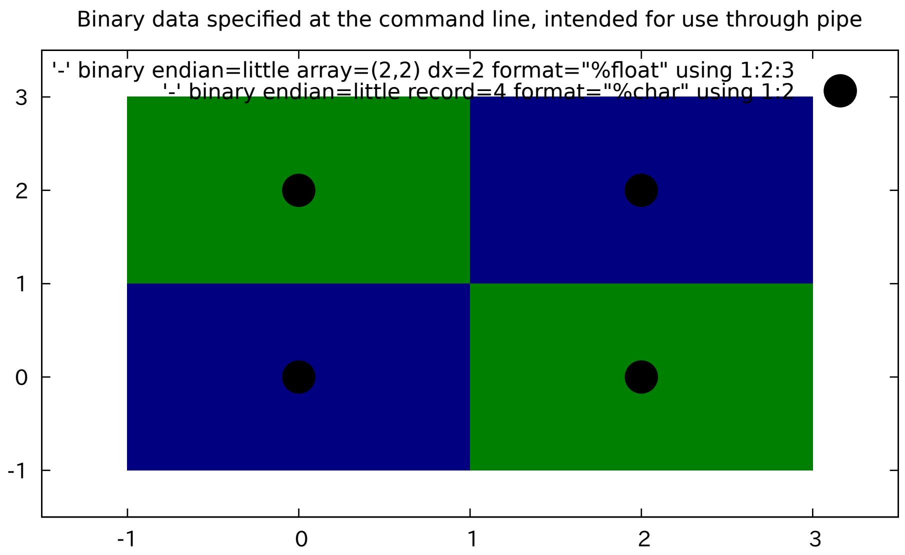

Decimation works on general binary data files as well.
Let Tux have his fun...
ux.rgb' binary array=(128,128) flipy format='%uchar' every 1:1:43:15:83:65
"Can I do that?"
120
110
100
90
80
70
60
30
40
50
60
70
80
90

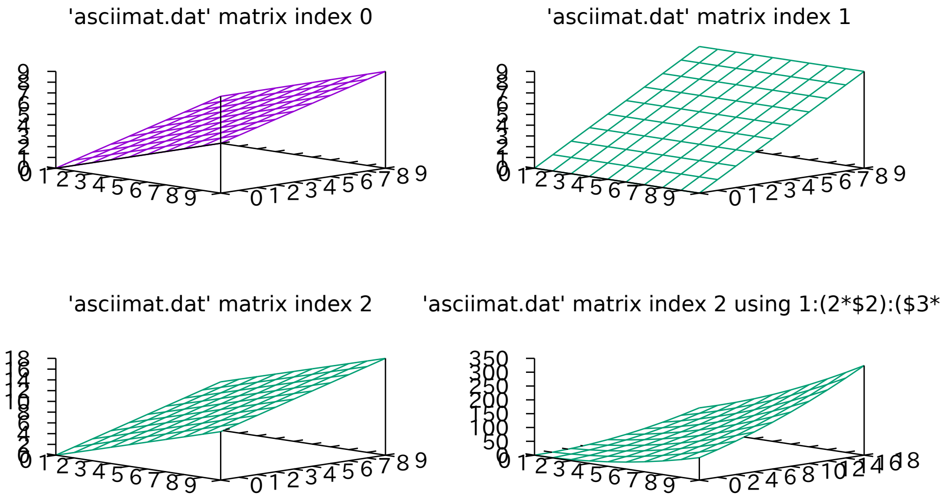
... Sure, go ahead.

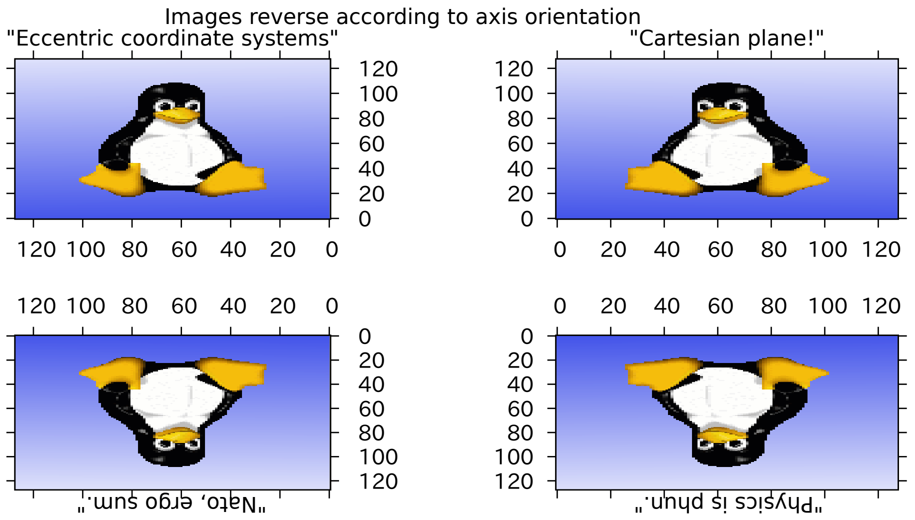



Automatically recognizing file type and extracting file information
'demo.edf' binary filetype=auto
700
600
500
400
300
200
100
0
-100
-100
0
100
200
300
400
500
600
700
140
120
100
80
60
40
20
0

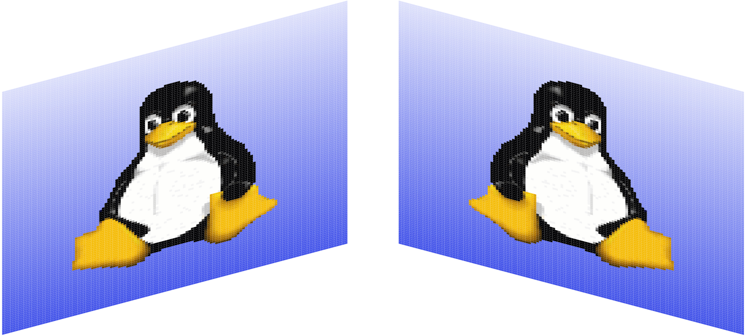

Details read from file
Transpose of file-read axes parameters
Details specified at command line
array=(32,64) dx=10 dy=5 scan=xy
Transpose of command line axes parameters
array=(64,32) dx=5 dy=10 scan=yx
0
100
200
300
400
500
600
0
20
40
60
80
100
120
140


Binary data specified at the command line, intended for use through pipe
'-' binary endian=little array=(2,2) dx=2 format="%float" using 1:2:3
'-' binary endian=little record=4 format="%char" using 1:2
3
2
1
0
-1
-1
0
1
2
3

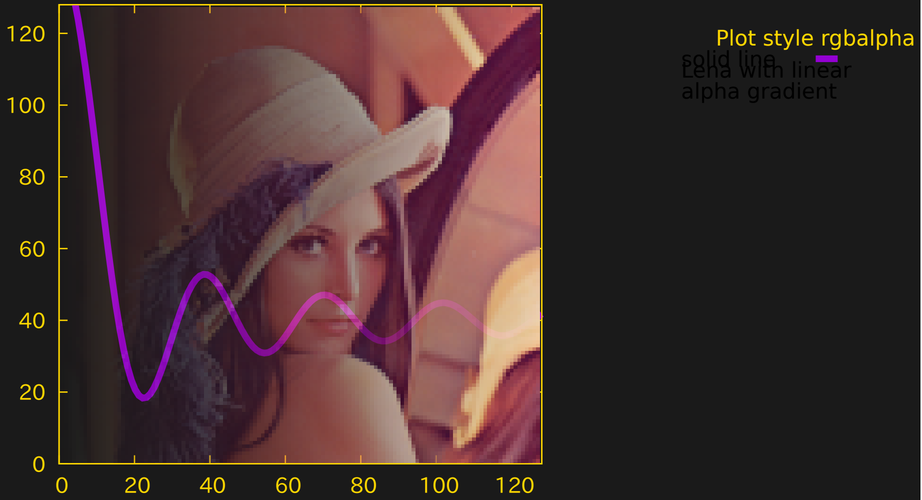

'asciimat.dat' matrix index 0
'asciimat.dat' matrix index 1
'asciimat.dat' matrix index 2
'asciimat.dat' matrix index 2 using 1:(2*$2):($3*

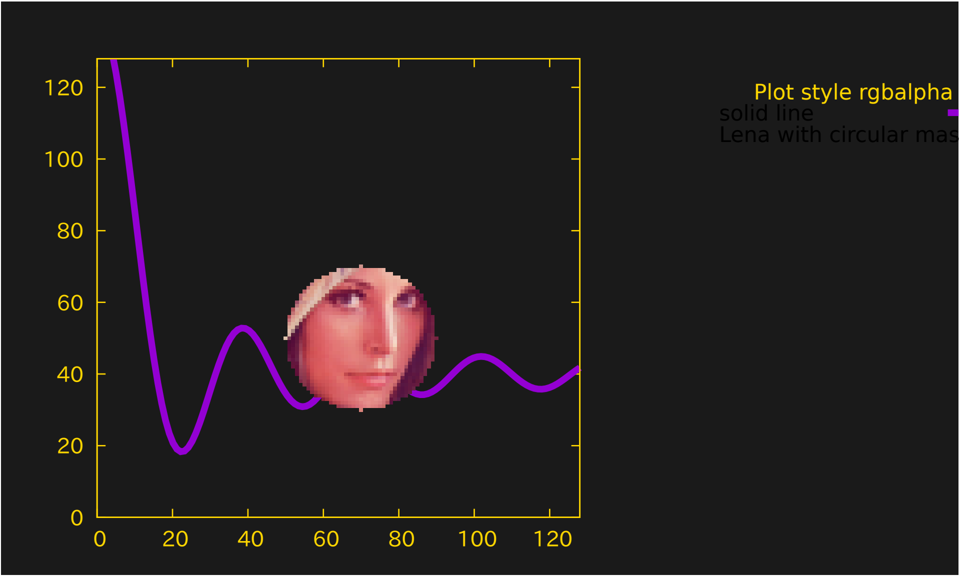


Images reverse according to axis orientation
"Eccentric coordinate systems"
"Cartesian plane!"
"Nato, ergo sum."
"Physics is phun."

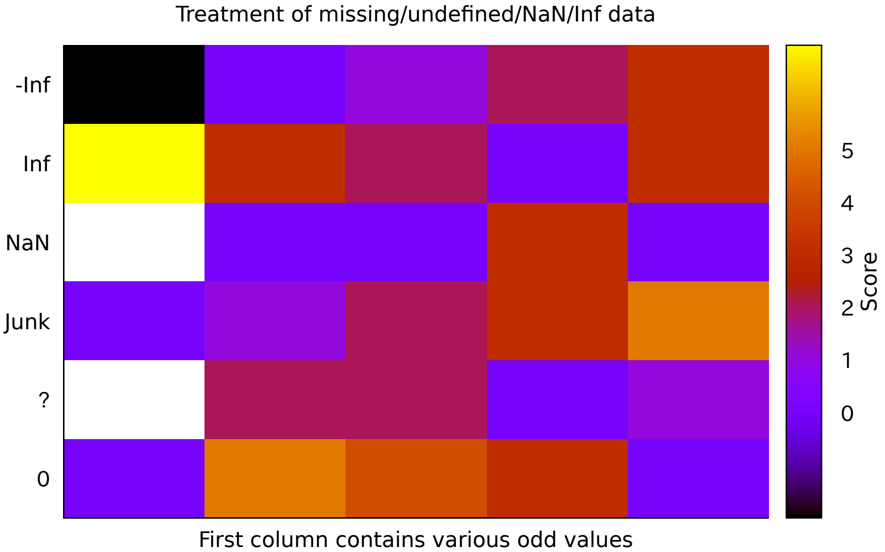


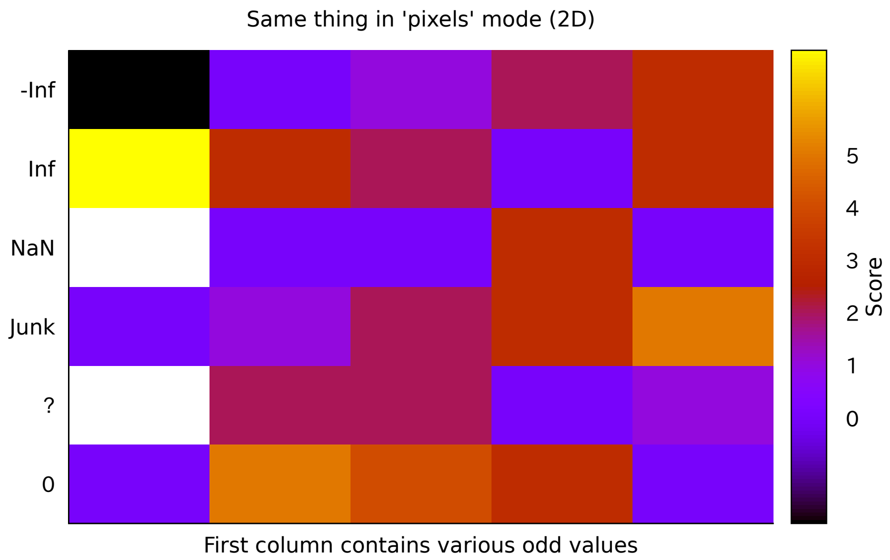
Tux in a reflective mood


Plot style rgbalpha
solid line
Lena with linear alpha gradient
0
20
40
60
80
100
120
0
20
40
60
80
100
120

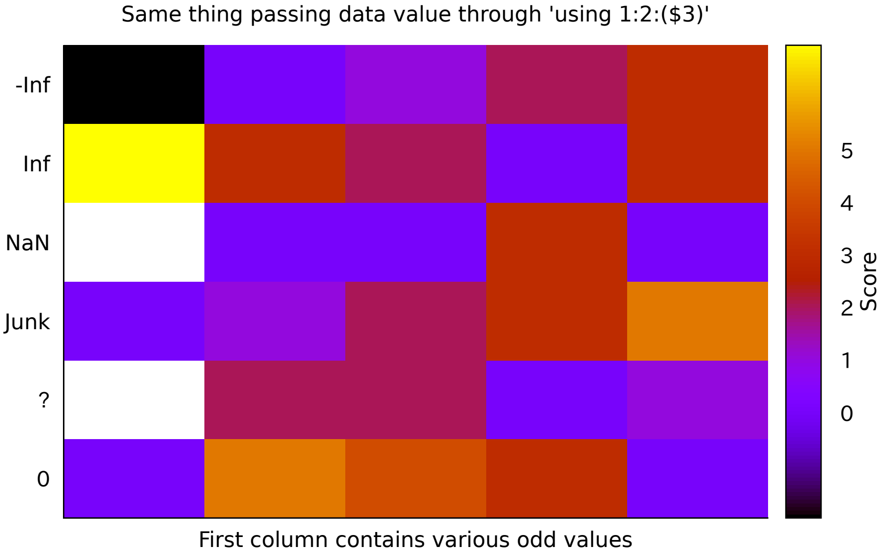


Plot style rgbalpha
solid line
Lena with circular mas
120
100
80
60
40
20
0
0
20
40
60
80
100
120

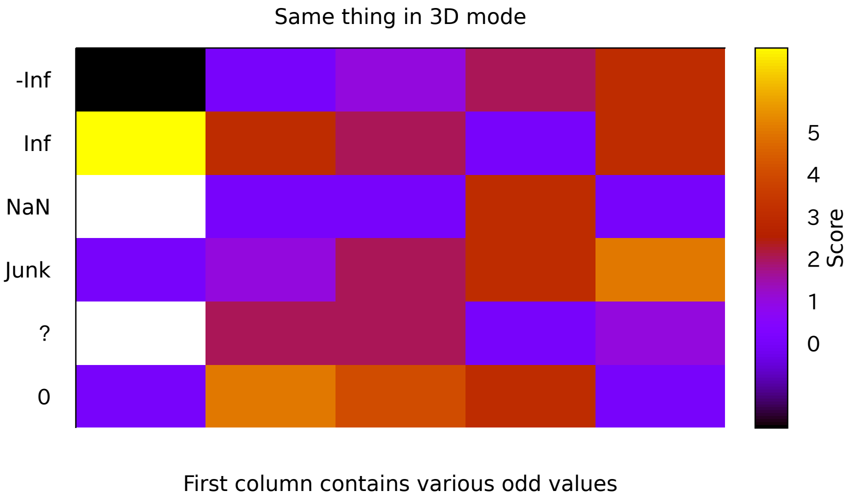

Treatment of missing/undefined/NaN/Inf data
-Inf
Inf
NaN
Junk
?
0
First column contains various odd values
5
4
3
2
1
0
Score

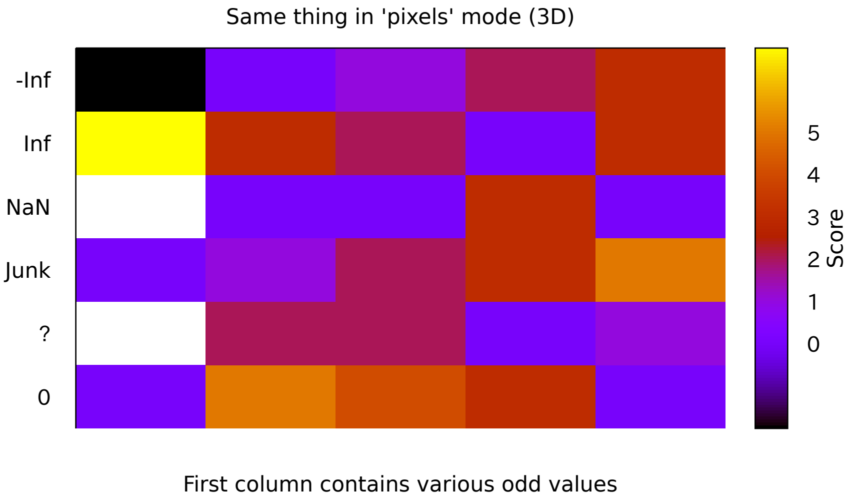


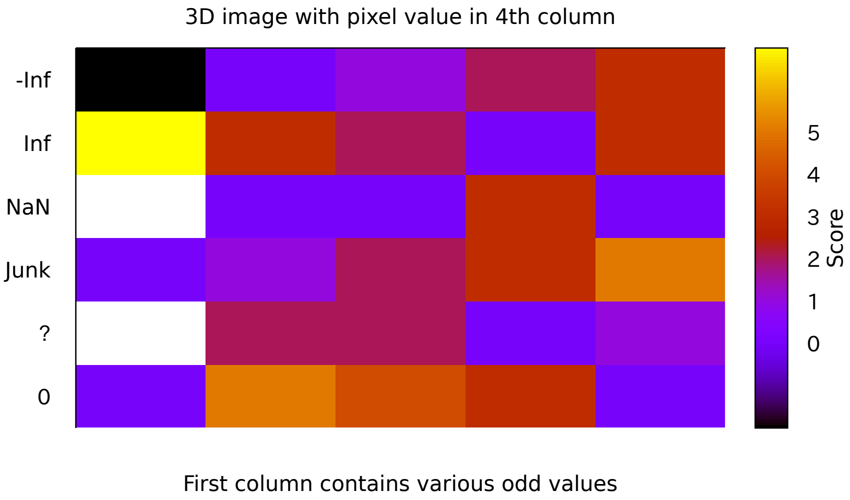
Same thing in 'pixels' mode (2D)
-Inf
Inf
NaN
Junk
?
0
5
4
3
2
1
0
Score
First column contains various odd values



Same thing passing data value through 'using 1:2:($3)'
-Inf
Inf
NaN
Junk
?
0
5
4
3
2
1
0
Score
First column contains various odd values


Same thing in 3D mode
-Inf
Inf
NaN
Junk
?
0
5
4
3
2
1
0
Score
First column contains various odd values


Same thing in 'pixels' mode (3D)
-Inf
Inf
NaN
Junk
?
0
5
4
3
2
1
0
Score
First column contains various odd values


3D image with pixel value in 4th column
-Inf
Inf
NaN
Junk
?
0
5
4
3
2
1
0
Score
First column contains various odd values

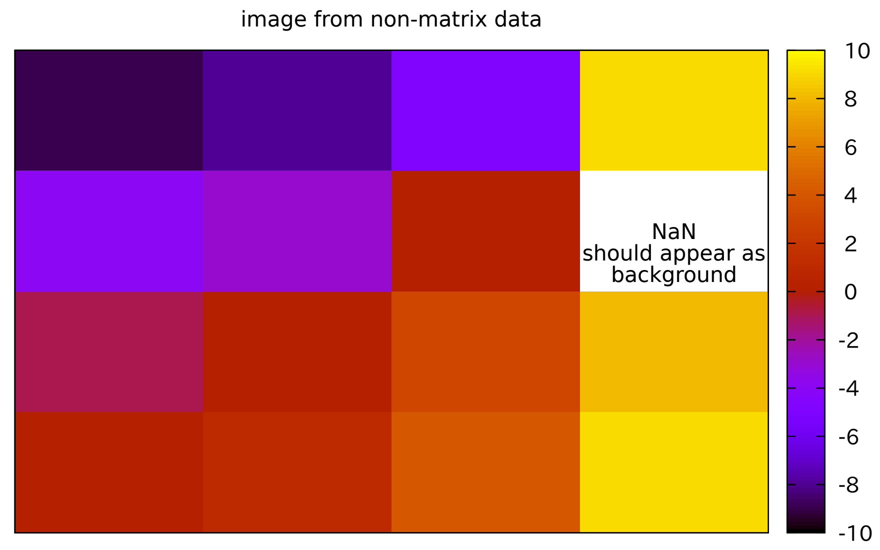
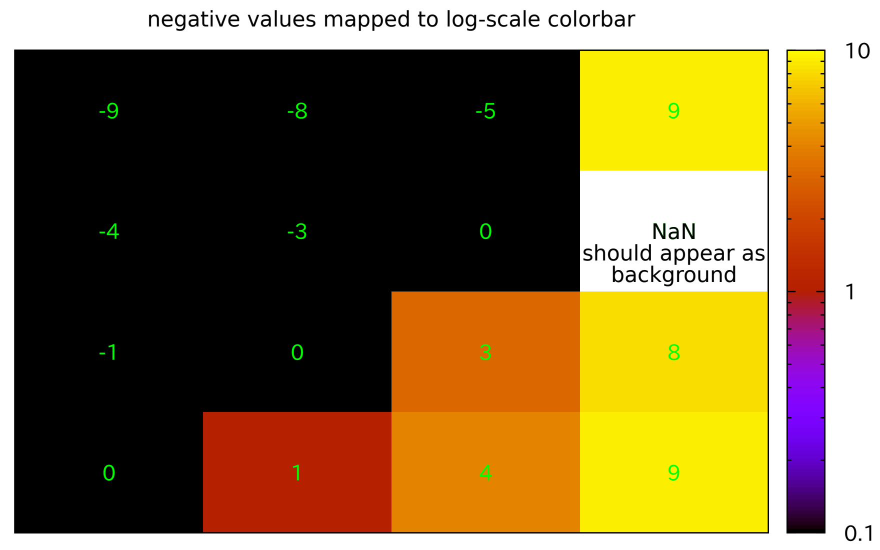
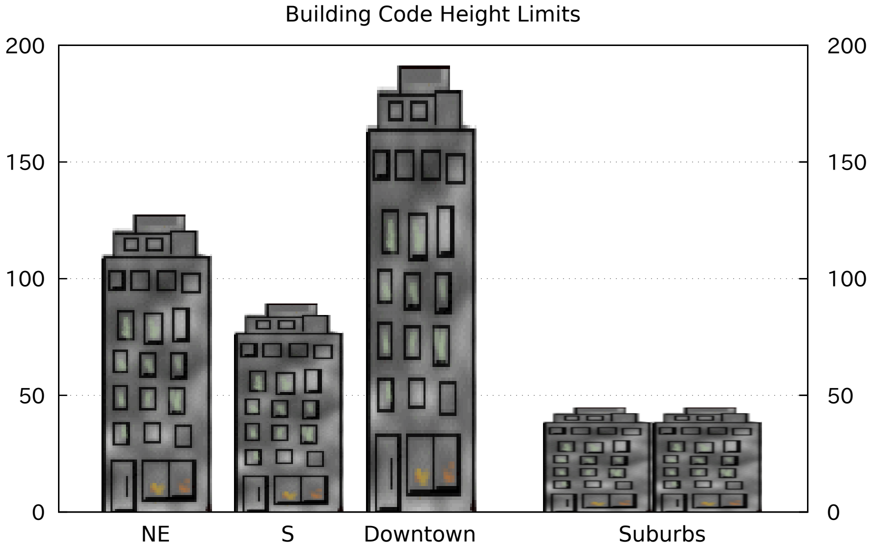


image from non-matrix data
NaN
should appear as
background
10
8
6
4
2
0
-2
-4
-6
-8
-10



negative values mapped to log-scale colorbar
-9
-8
-5
9
-4
-3
0
NaN
should appear as
background
-1
0
3
8
0
1
4
9
10
1
0.1


Building Code Height Limits
200
150
100
50
0
NE
S
Downtown
Suburbs

Exercise substring handling

```
beg =  2  end =  4
foo           =  ABCDEF
foo[3:5]      =  CDE
foo[1:1]      =  A
foo[5:3]      =
foo[beg:end]  =  BCD
foo[end:beg]  =
foo[5:]       =  EF
foo[5:*]      =  EF
foo[:]        =  ABCDEF
foo[*:*]      =  ABCDEF
foo.foo[2:2]  =  ABCDEFB
(foo.foo)[2:2]=  B

foo[1:1] eq 'A'  foo[2:2] ne 'X' =  true
```

Exercise string handling functions

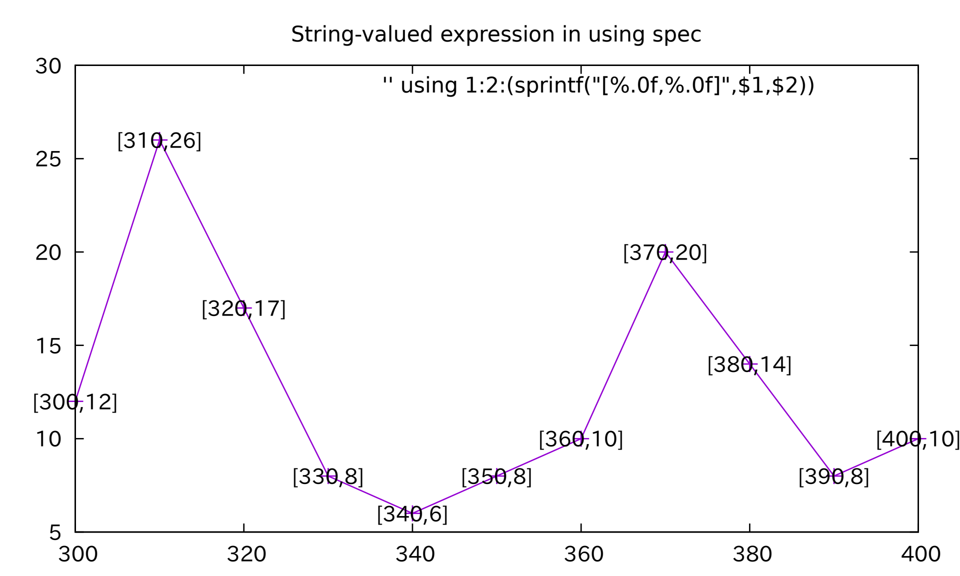
```
foo          =  ABCDEF
strlen(foo)  =  6
substr(foo,3,4) =  CD

haystack     = `date`
haystack     =  2023年 5月24日 水曜日 14時22分46秒 JST
needle       =  :
S = strstrt(haystack,needle) =  0
haystack[S-2:S+2] =  20
It is now 20

words(haystack)   =  5
word(haystack,5)  =  JST

sprintf output of long strings works OK
```

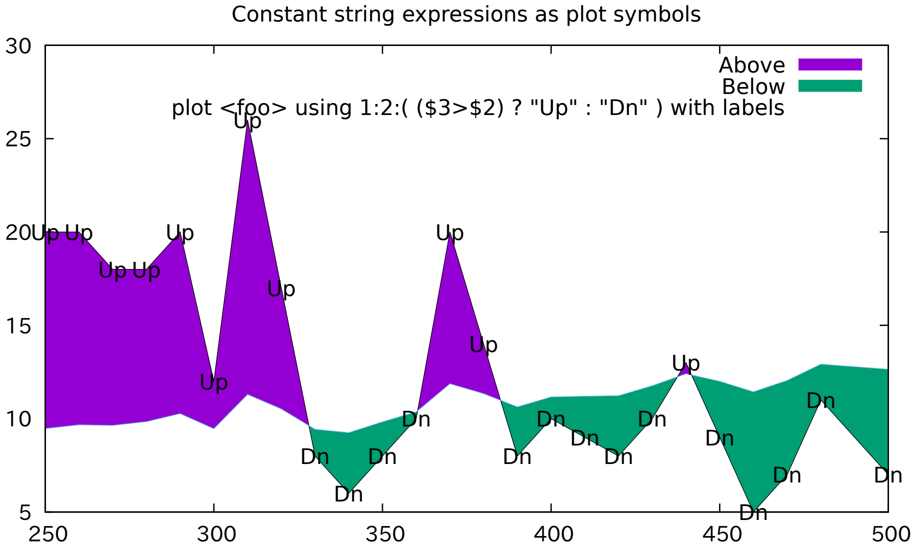
Exercise word and words functions

foo = word and words can handle 'quoted string'
words(foo) = 6
word(foo, 6) = quoted string

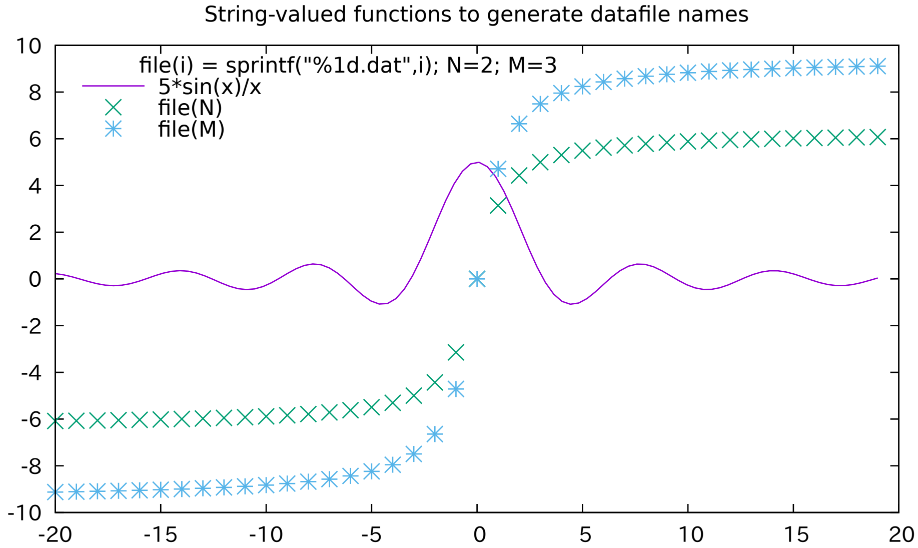
foo = "double quotes" or 'single quotes'
words(foo) = 3

foo = Apostrophes inside words don't matter
word(foo, 4) = don't

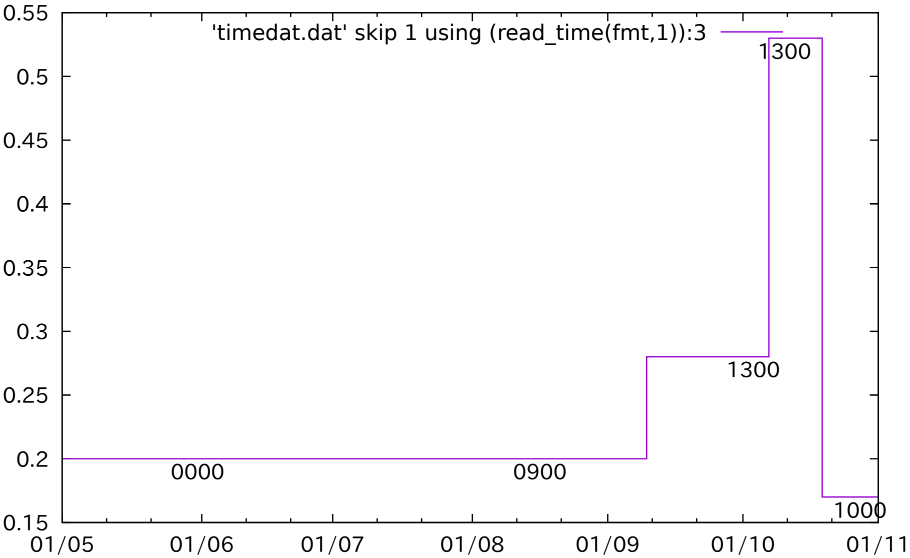

String-valued expression in using spec
'' using 1:2:(sprintf("[%.0f,%.0f]",$1,$2))
[300,12]
[310,26]
[320,17]
[330,8]
[340,6]
[350,8]
[360,10]
[370,20]
[380,14]
[390,8]
[400,10]
30
25
20
15
10
5
300
320
340
360
380
400



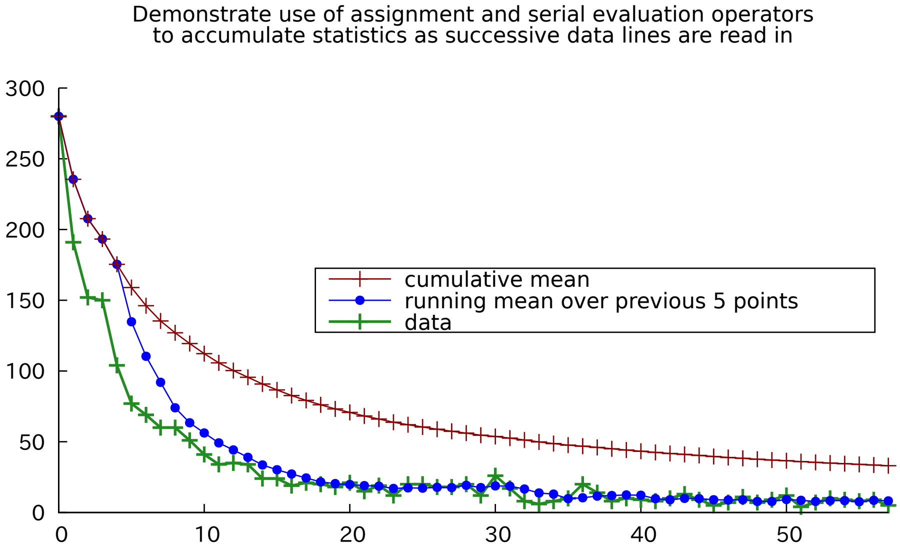
Constant string expressions as plot symbols
Above
Below
plot <foo> using 1:2:(($3>$2) ? "Up" : "Dn") with labels
30
25
20
15
10
5
250
300
350
400
450
500
Up
Dn



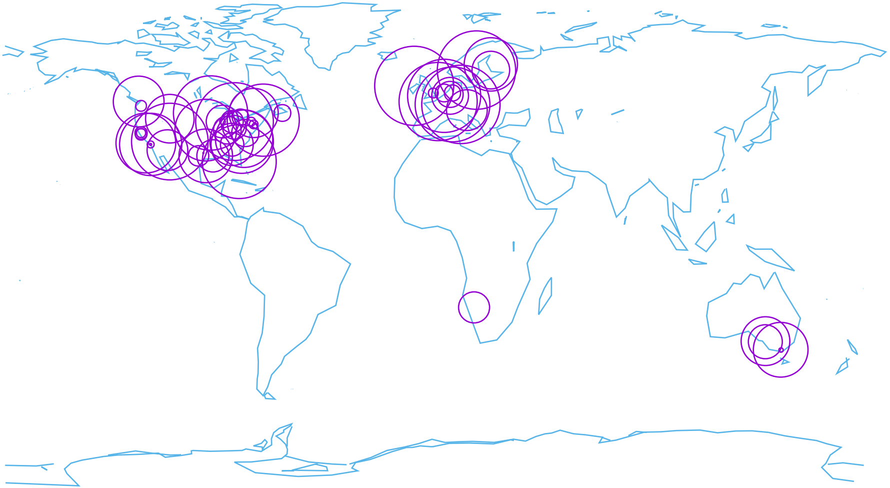
String-valued functions to generate datafile names
file(i) = sprintf("%1d.dat",i); N=2; M=3
5*sin(x)/x
file(N)
file(M)
10
8
6
4
2
0
-2
-4
-6
-8
-10
-20
-15
-10
-5
0
5
10
15
20



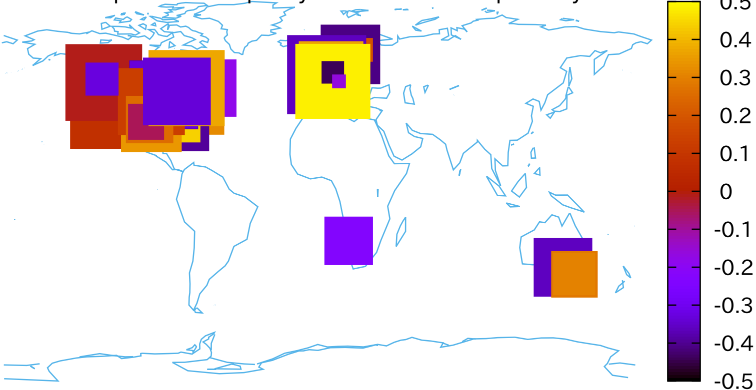
'timedat.dat' skip 1 using (read_time(fmt,1)):3
1300
1300
0000
0900
1000
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
01/05
01/06
01/07
01/08
01/09
01/10
01/11


Demonstrate use of assignment and serial evaluation operators
to accumulate statistics as successive data lines are read in
cumulative mean
running mean over previous 5 points
data
300
250
200
150
100
50
0
0
10
20
30
40
50

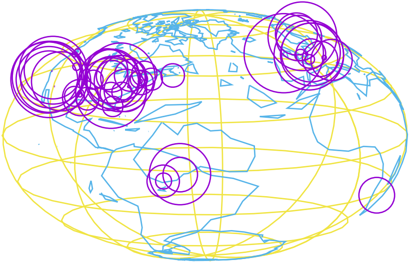
plot with variable size points



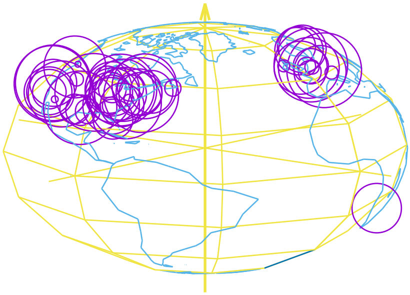
splot with variable size points
it is possible to specify size and color separately

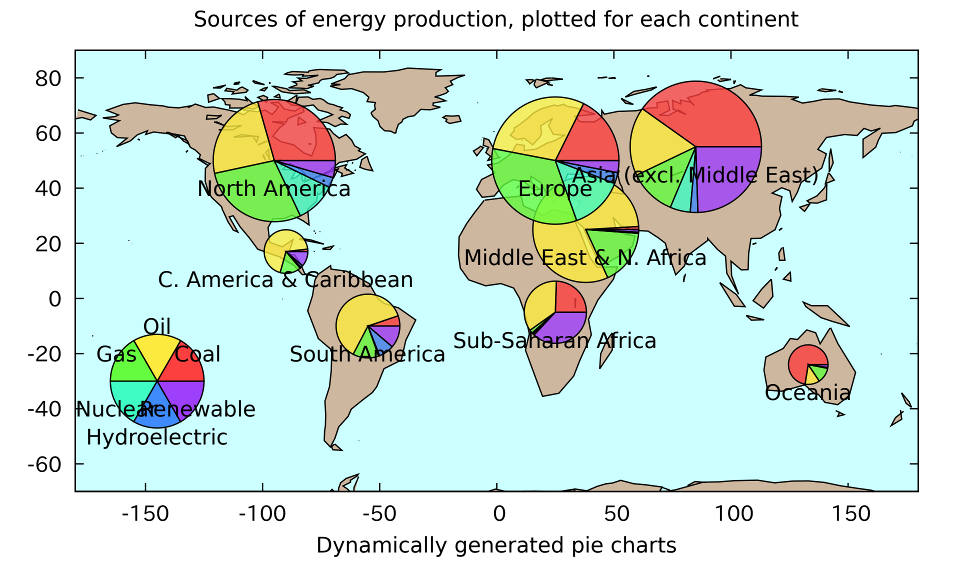


3D version using spherical coordinate system

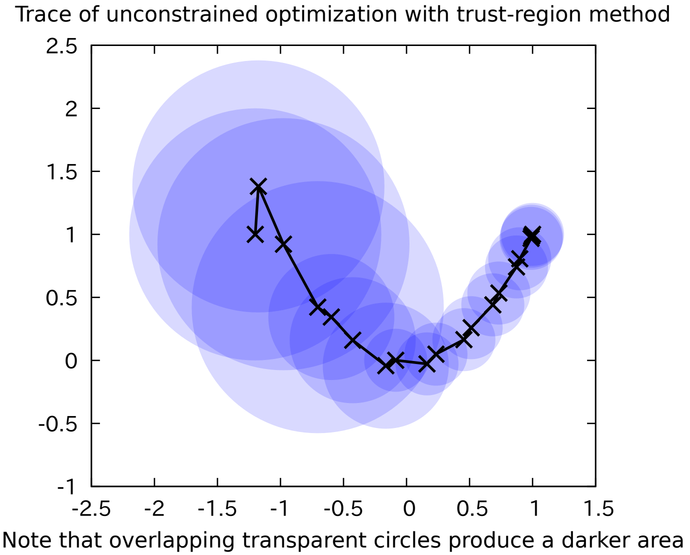


3D solid version through hiddenlining



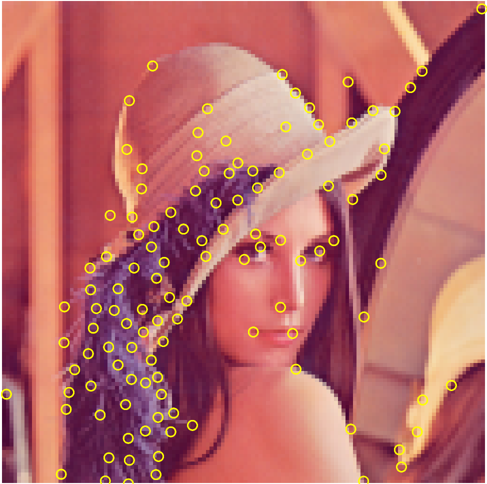


Sources of energy production, plotted for each continent
80
60
40
20
0
-20
-40
-60
-150
-100
-50
0
50
100
150
North America
C. America & Caribbean
South America
Europe
Asia (excl. Middle East)
Middle East & N. Africa
Sub-Saharan Africa
Oceania
Oil
Gas
Coal
Nuclear
Renewable
Hydroelectric
Dynamically generated pie charts

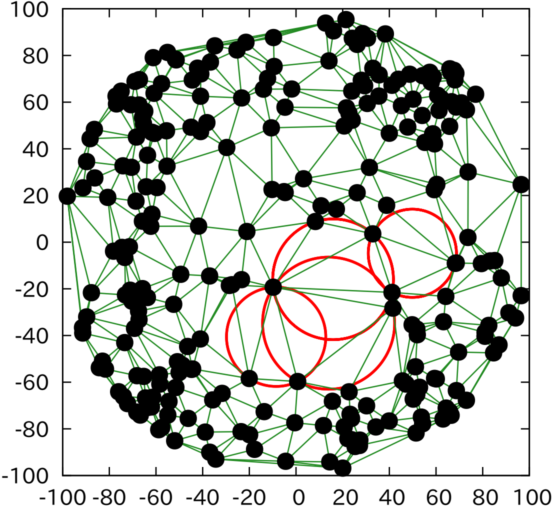


Trace of unconstrained optimization with trust-region method
2.5
2
1.5
1
0.5
0
-0.5
-1
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
Note that overlapping transparent circles produce a darker area

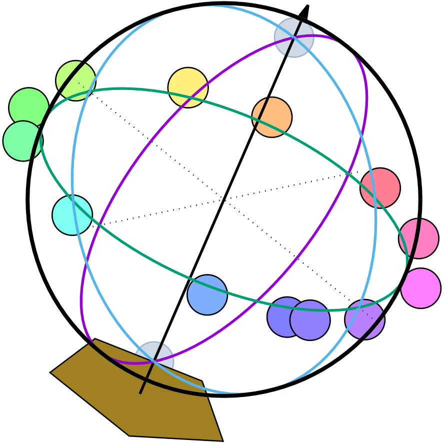
Lena's key points

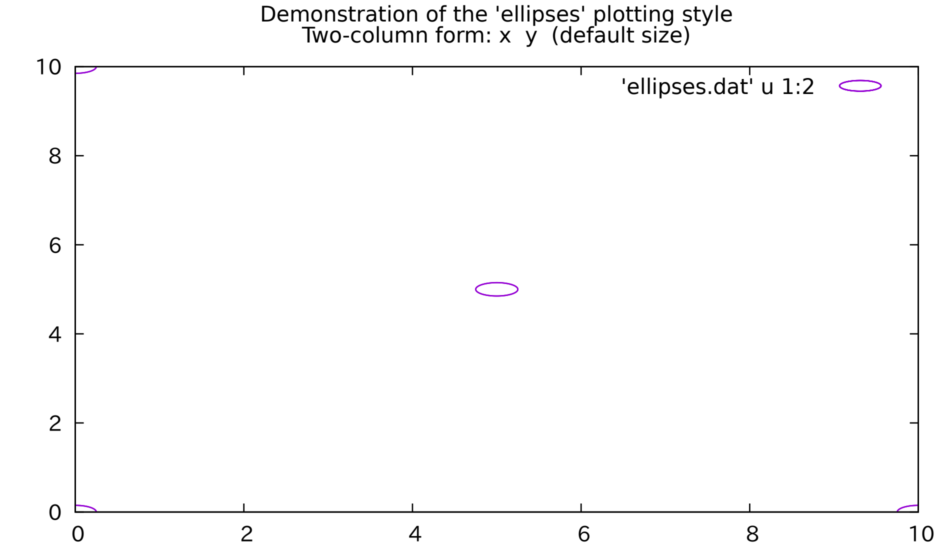


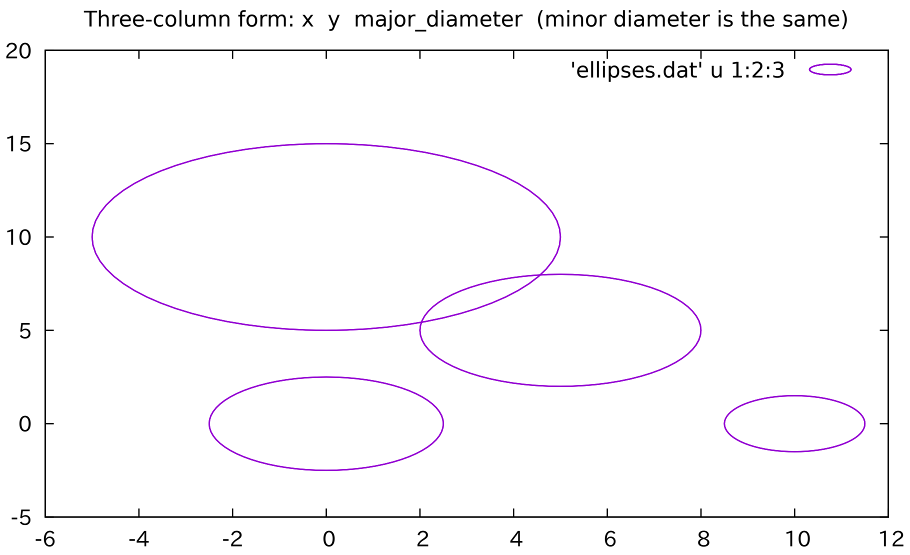
Delaunay triangulation of Hemisphere points, some empty circles in red

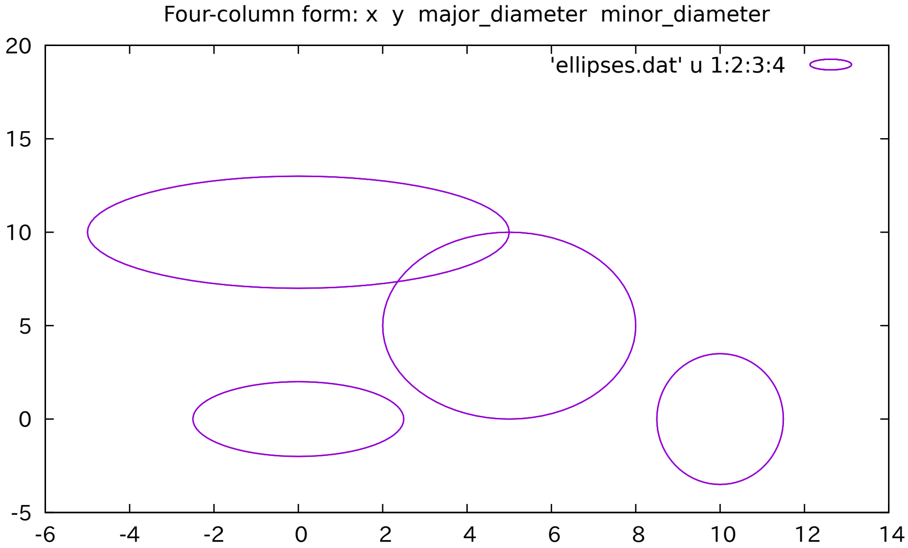


Circle and polygon objects in 3D

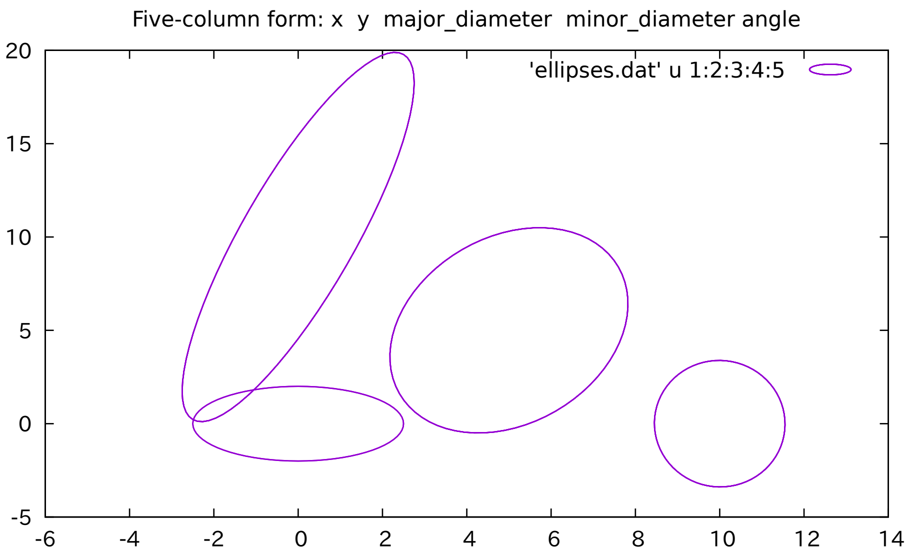


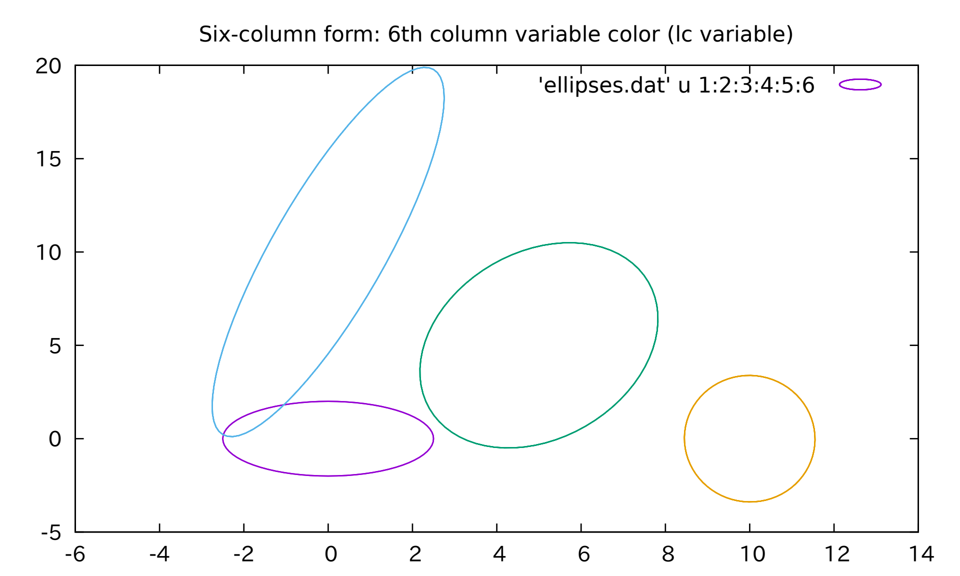

Demonstration of the 'ellipses' plotting style
Two-column form: x y (default size)
'ellipses.dat' u 1:2
0
2
4
6
8
10
0
2
4
6
8
10

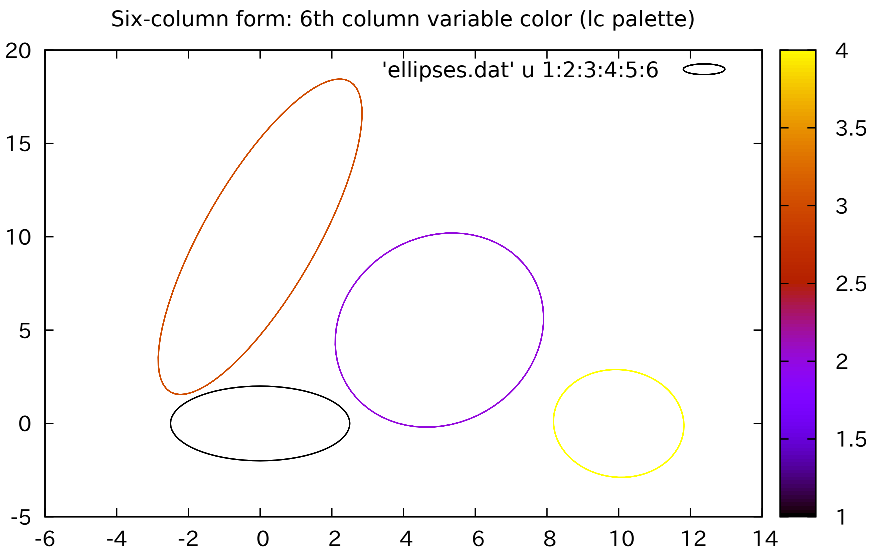

Three-column form: x y major_diameter (minor diameter is the same)
'ellipses.dat' u 1:2:3
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12



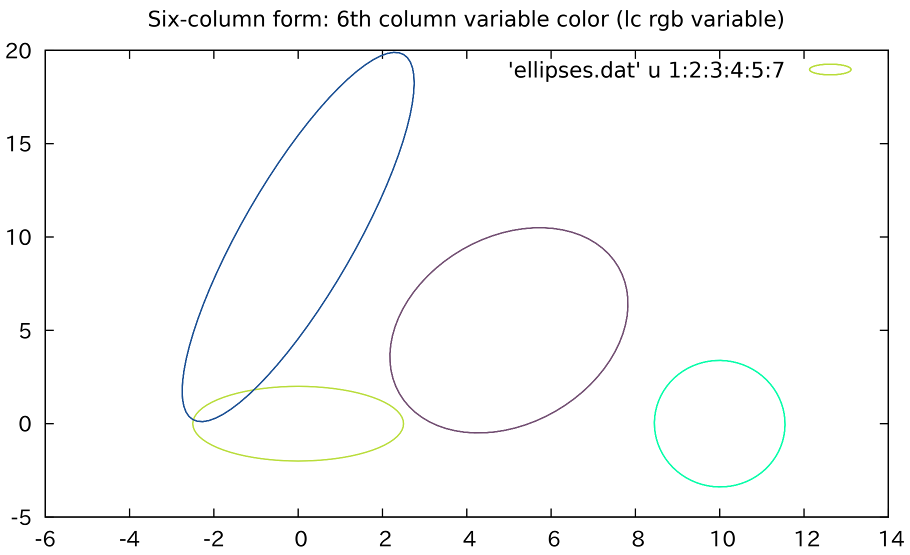
Four-column form: x y major_diameter minor_diameter
'ellipses.dat' u 1:2:3:4
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12
14


Five-column form: x y major_diameter minor_diameter angle
'ellipses.dat' u 1:2:3:4:5
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12
14

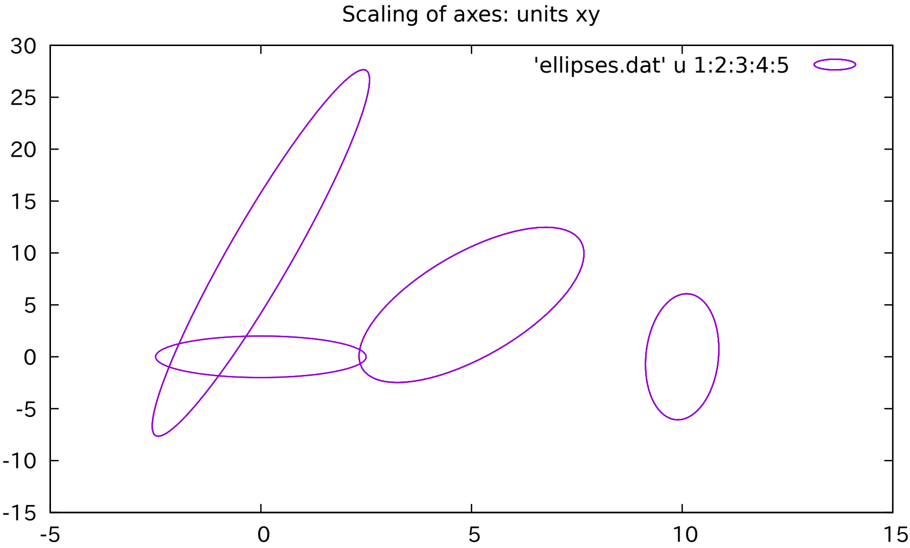

Six-column form: 6th column variable color (lc variable)
'ellipses.dat' u 1:2:3:4:5:6
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12
14

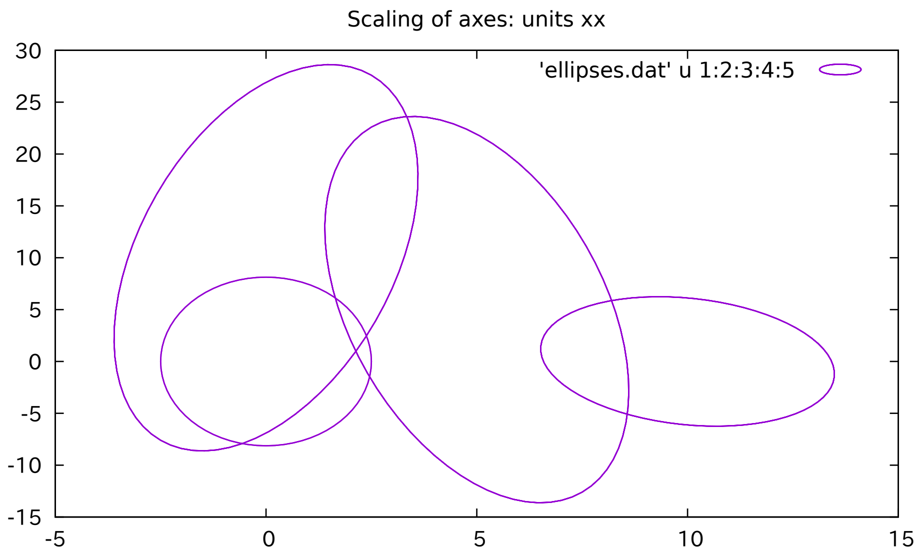


Six-column form: 6th column variable color (lc palette)
'ellipses.dat' u 1:2:3:4:5:6
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12
14
4
3.5
3
2.5
2
1.5
1

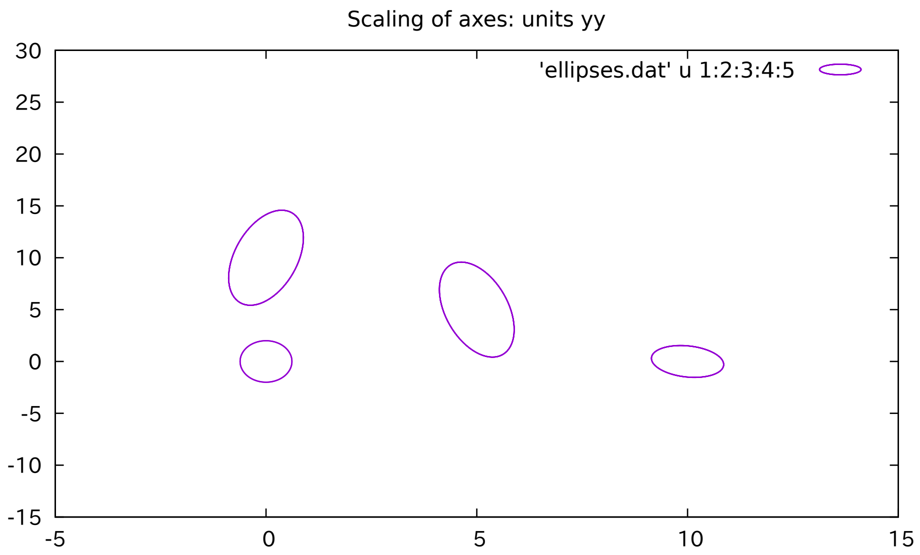


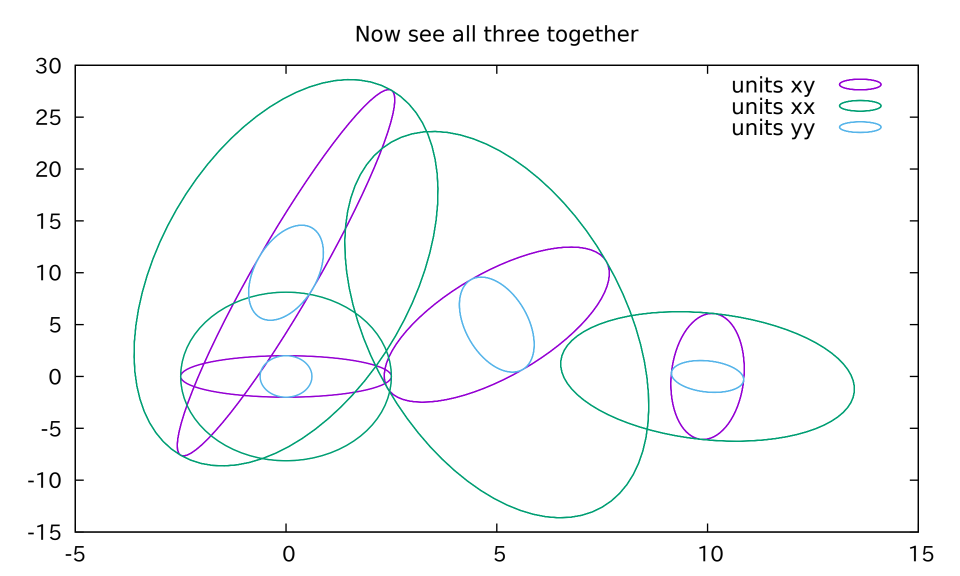
Six-column form: 6th column variable color (lc rgb variable)
'ellipses.dat' u 1:2:3:4:5:7
20
15
10
5
0
-5
-6
-4
-2
0
2
4
6
8
10
12
14

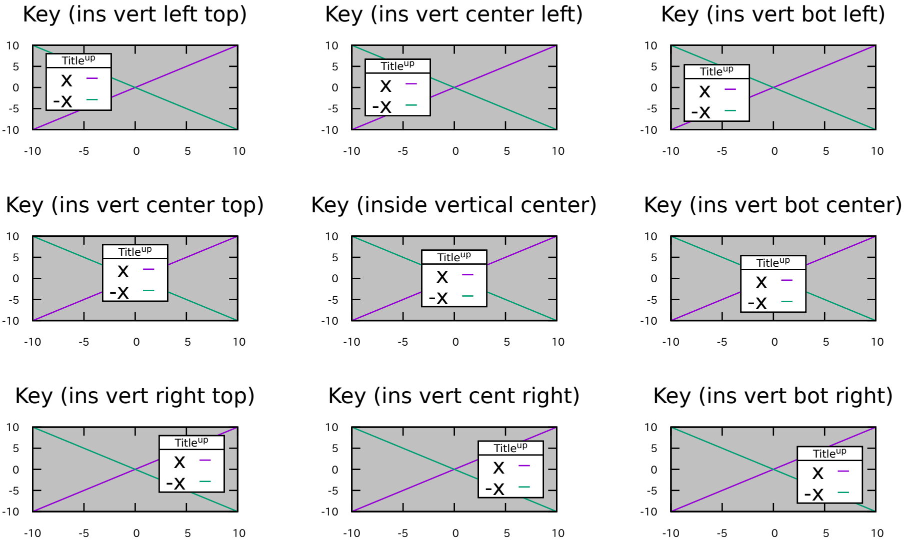

Scaling of axes: units xy
'ellipses.dat' u 1:2:3:4:5
30
25
20
15
10
5
0
-5
-10
-15
-5
0
5
10
15



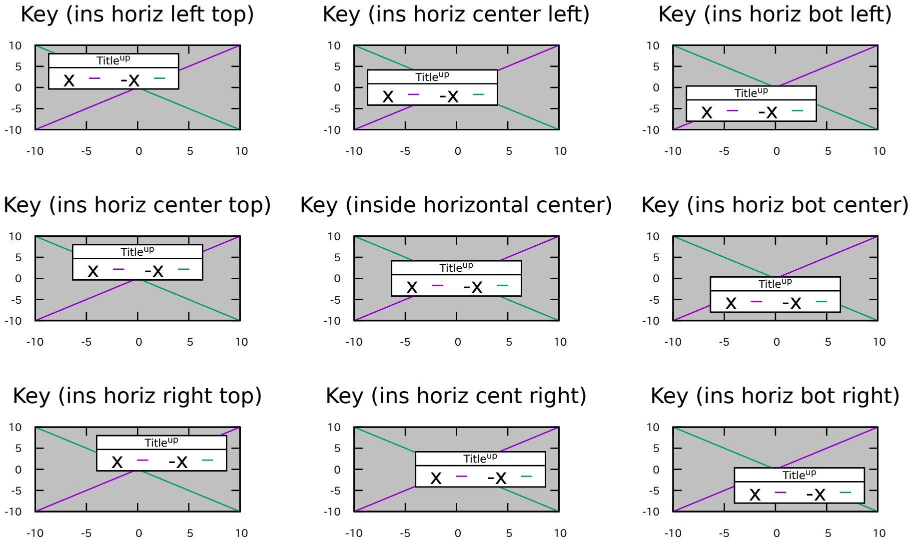
Scaling of axes: units xx
'ellipses.dat' u 1:2:3:4:5
30
25
20
15
10
5
0
-5
-10
-15
-5
0
5
10
15

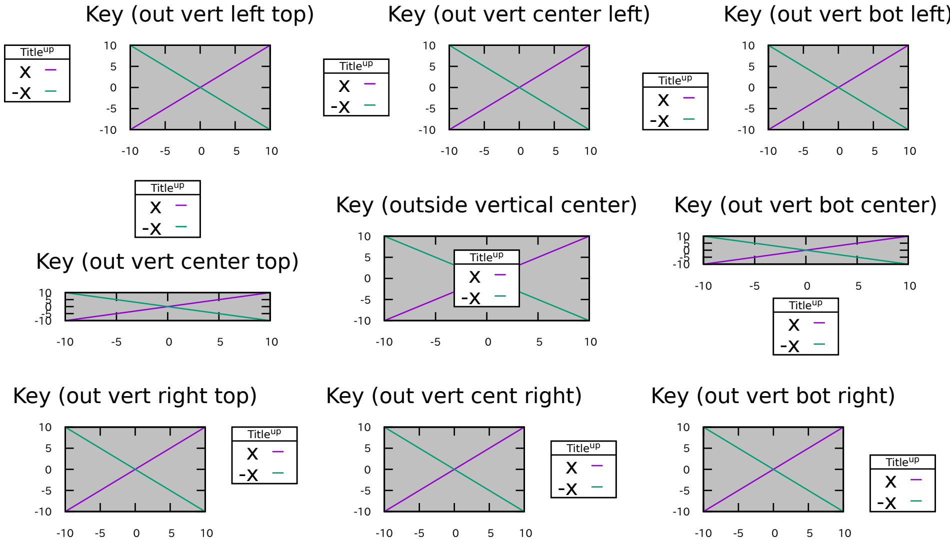

Scaling of axes: units yy
'ellipses.dat' u 1:2:3:4:5
30
25
20
15
10
5
0
-5
-10
-15
-5
0
5
10
15


Now see all three together
units xy
units xx
units yy
30
25
20
15
10
5
0
-5
-10
-15
-5
0
5
10
15

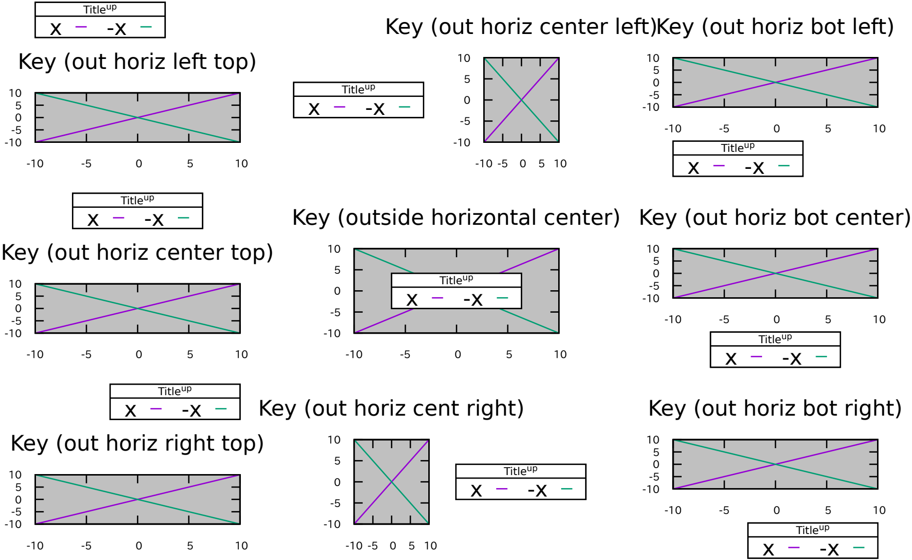


Key (ins vert left top)
Key (ins vert center left)
Key (ins vert bot left)
Key (ins vert center top)
Key (inside vertical center)
Key (ins vert bot center)
Key (ins vert right top)
Key (ins vert cent right)
Key (ins vert bot right)
Title^up
x
-x
10
5
0
-5
-10

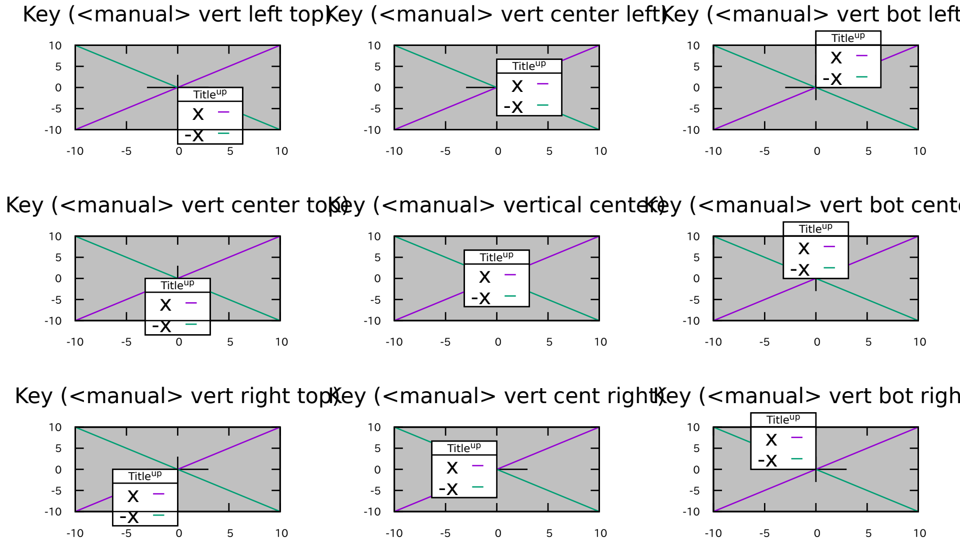

Key (ins horiz left top)
Key (ins horiz center left)
Key (ins horiz bot left)
Key (ins horiz center top)
Key (inside horizontal center)
Key (ins horiz bot center)
Key (ins horiz right top)
Key (ins horiz cent right)
Key (ins horiz bot right)
Title^up
x
-x
10
5
0
-5
-10

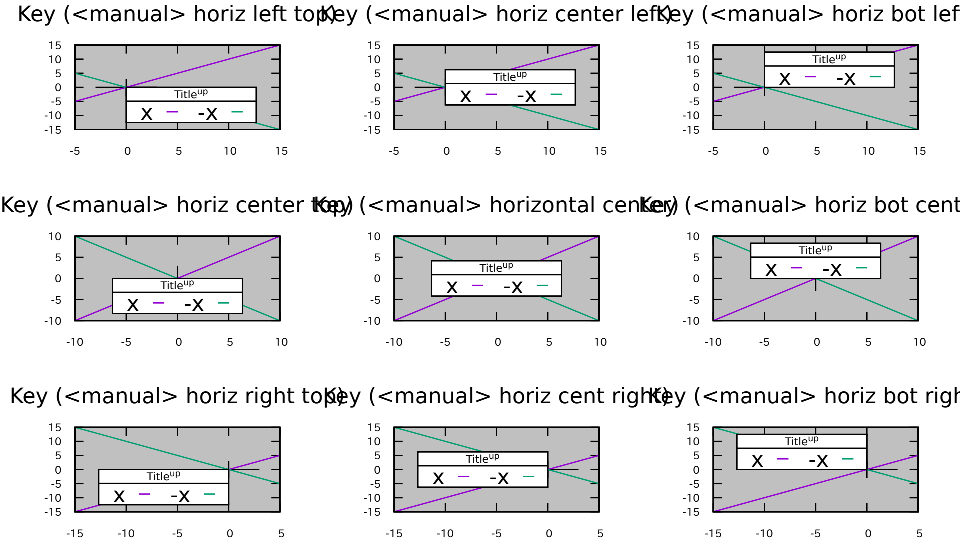


Key (out vert left top)
Key (out vert center left)
Key (out vert bot left)
Key (outside vertical center)
Key (out vert bot center)
Key (out vert center top)
Key (out vert right top)
Key (out vert cent right)
Key (out vert bot right)
Title^{up}
x
-x
10
5
0
-5
-10

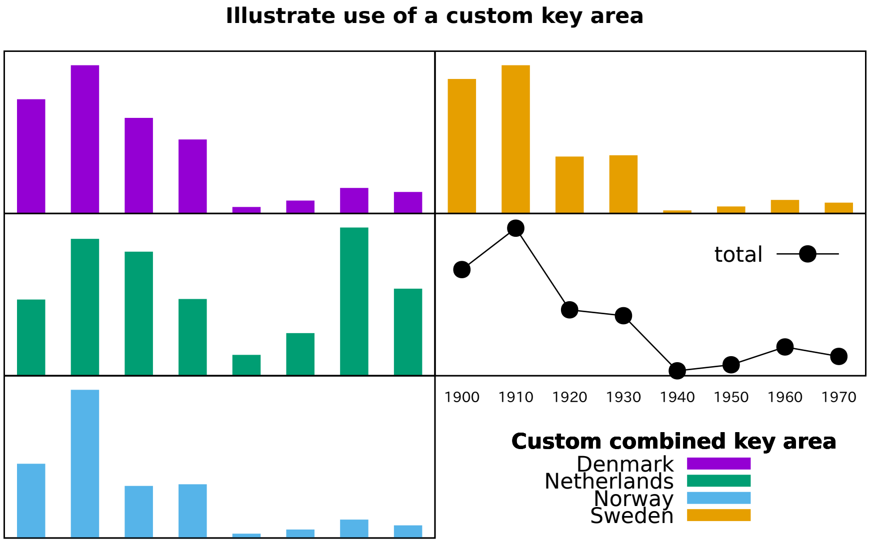


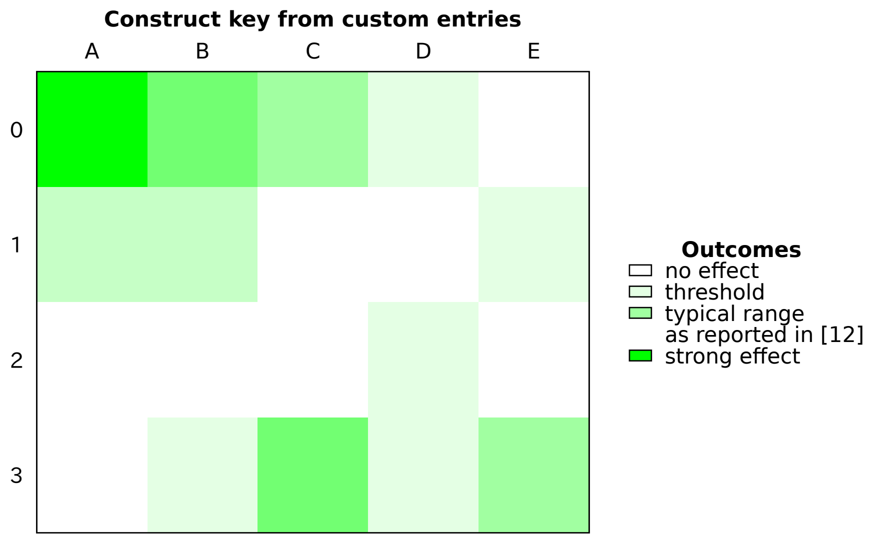
Titleup
x
-x
Key (out horiz left top)
Key (out horiz center left)
Key (out horiz bot left)
Key (out horiz center top)
Key (outside horizontal center)
Key (out horiz bot center)
Key (out horiz right top)
Key (out horiz cent right)
Key (out horiz bot right)
10
5
0
-5
-10

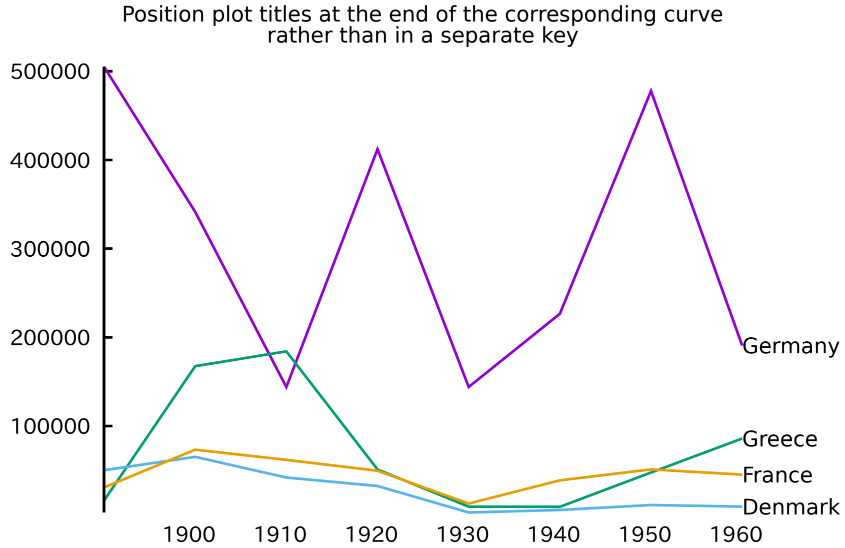

Key (<manual> vert left top)
Key (<manual> vert center left)
Key (<manual> vert bot left
Key (<manual> vert center top)
Key (<manual> vertical center)
Key (<manual> vert bot cente
Key (<manual> vert right top)
Key (<manual> vert cent right)
Key (<manual> vert bot righ
Title^up
x
-x
10
5
0
-5
-10

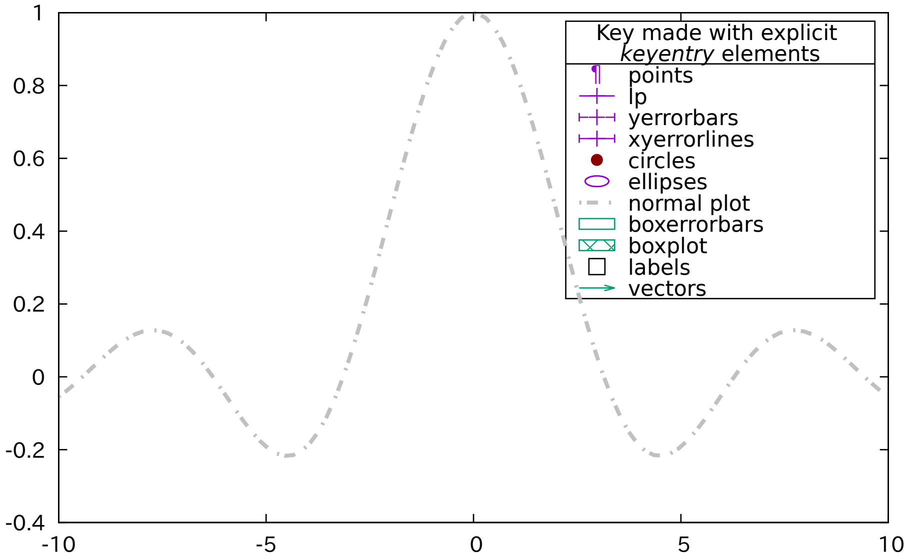


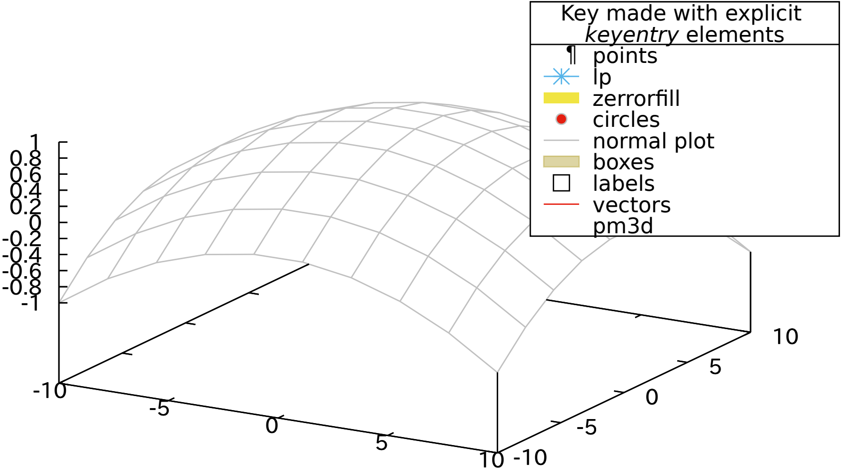
Key (<manual> horiz left top)
Key (<manual> horiz center left)
Key (<manual> horiz bot lef
Key (<manual> horiz center top)
Key (<manual> horizontal center)
Key (<manual> horiz bot cent
Key (<manual> horiz right top)
Key (<manual> horiz cent right)
Key (<manual> horiz bot righ
Title^up
x
-x
15
10
5
0
-5
-10
-15


Illustrate use of a custom key area
total
1900
1910
1920
1930
1940
1950
1960
1970
Custom combined key area
Denmark
Netherlands
Norway
Sweden

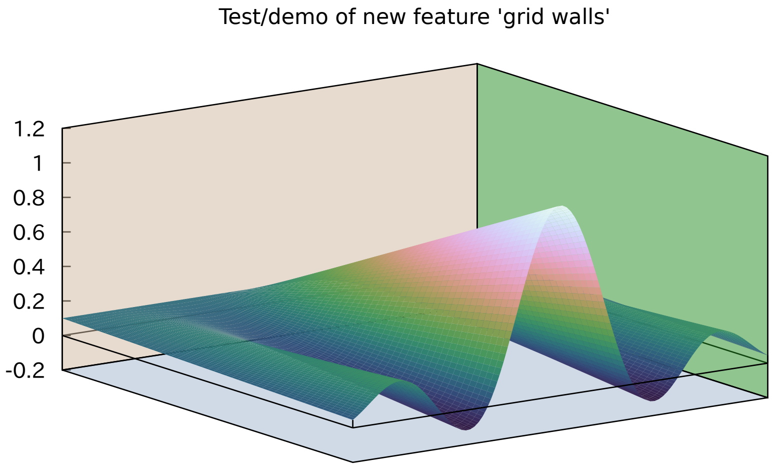

Construct key from custom entries
A
B
C
D
E
0
1
2
3
Outcomes
no effect
threshold
typical range
as reported in [12]
strong effect


Position plot titles at the end of the corresponding curve
rather than in a separate key
500000
400000
300000
200000
100000
1900
1910
1920
1930
1940
1950
1960
Germany
Greece
France
Denmark

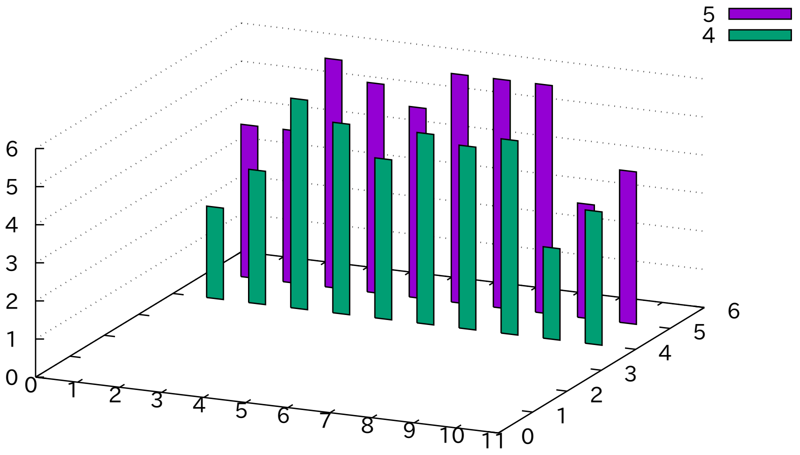

Key made with explicit
keyentry elements
points
lp
yerrorbars
xyerrorlines
circles
ellipses
normal plot
boxerrorbars
boxplot
labels
vectors
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10

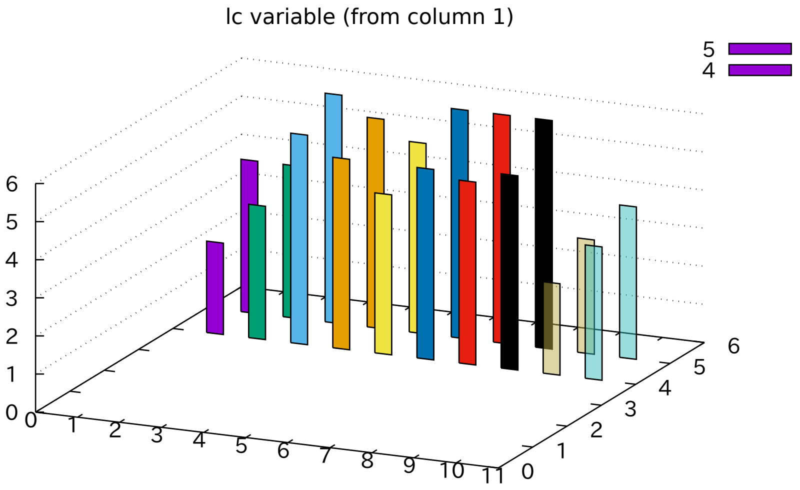


Key made with explicit
keyentry elements
points
lp
zerrorfill
circles
normal plot
boxes
labels
vectors
pm3d
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-10
-5
0
5
10
-10
-5
0
5
10

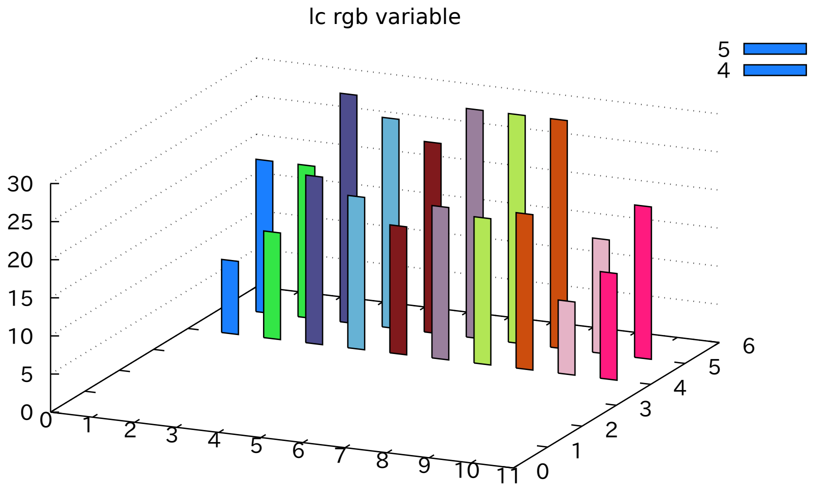

Test/demo of new feature 'grid walls'
1.2
1
0.8
0.6
0.4
0.2
0
-0.2

3D Boxes

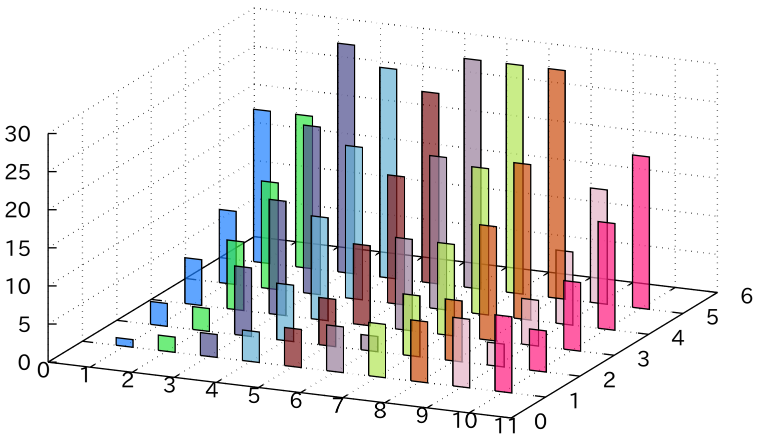




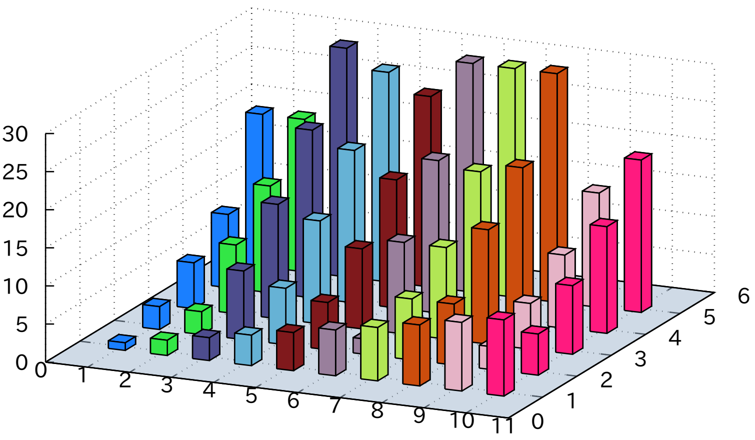
lc variable (from column 1)
5
4
0
1
2
3
4
5
6
0
1
2
3
4
5
6
7
8
9
10
11
0
1
2
3
4
5
6

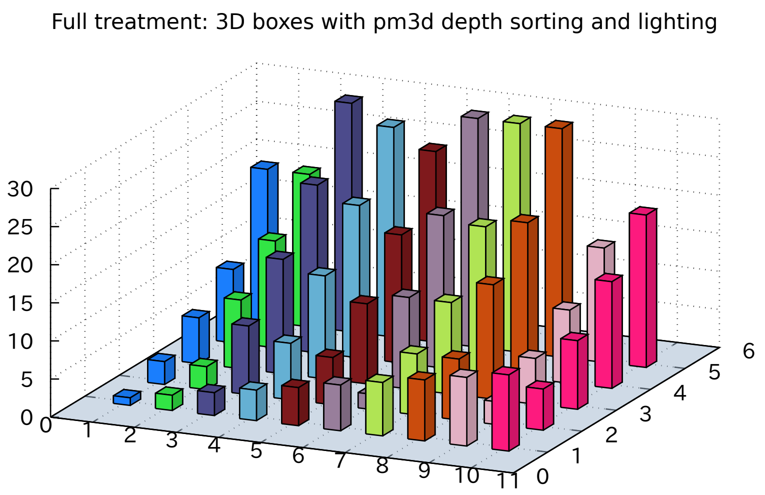

lc rgb variable
5
4
30
25
20
15
10
5
0
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6

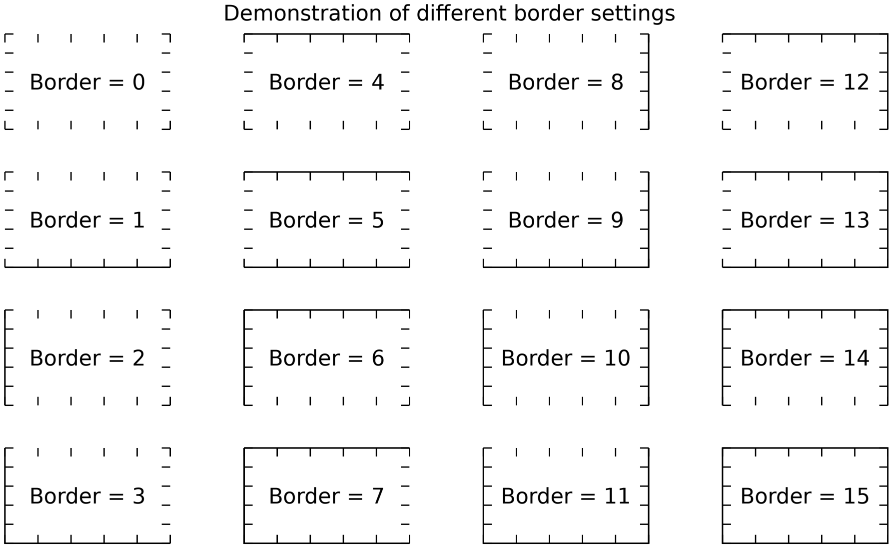
transparent boxes with imperfect depth sorting

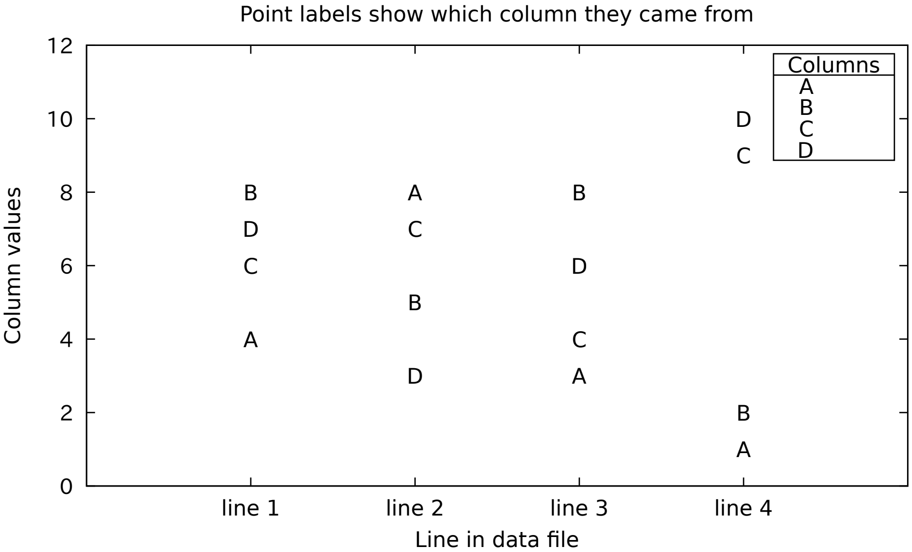


give the boxes a 3D depth and correct depth sorting

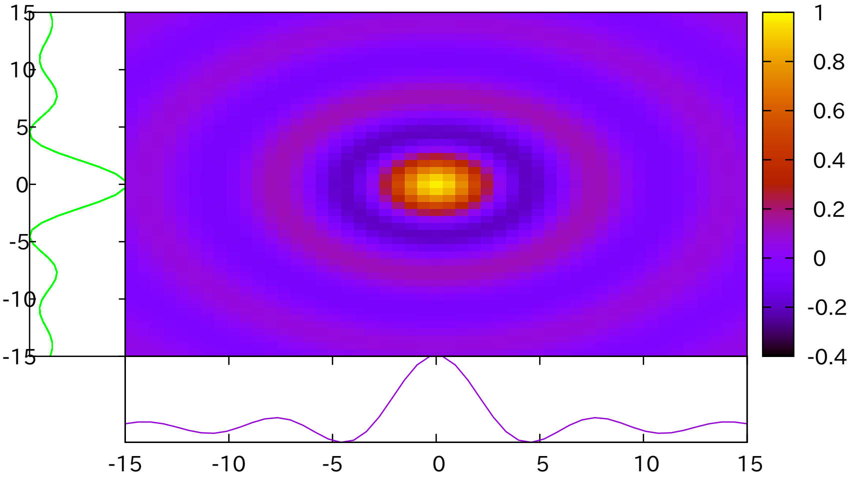


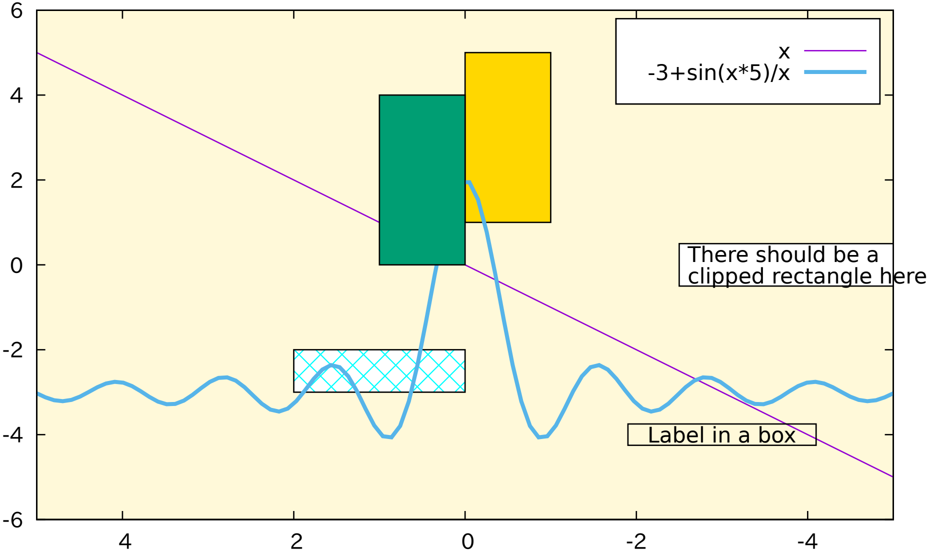

Full treatment: 3D boxes with pm3d depth sorting and lighting
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8
9
10
11
0
1
2
3
4
5
6


Demonstration of different border settings
Border = 0
Border = 4
Border = 8
Border = 12
Border = 1
Border = 5
Border = 9
Border = 13
Border = 2
Border = 6
Border = 10
Border = 14
Border = 3
Border = 7
Border = 11
Border = 15

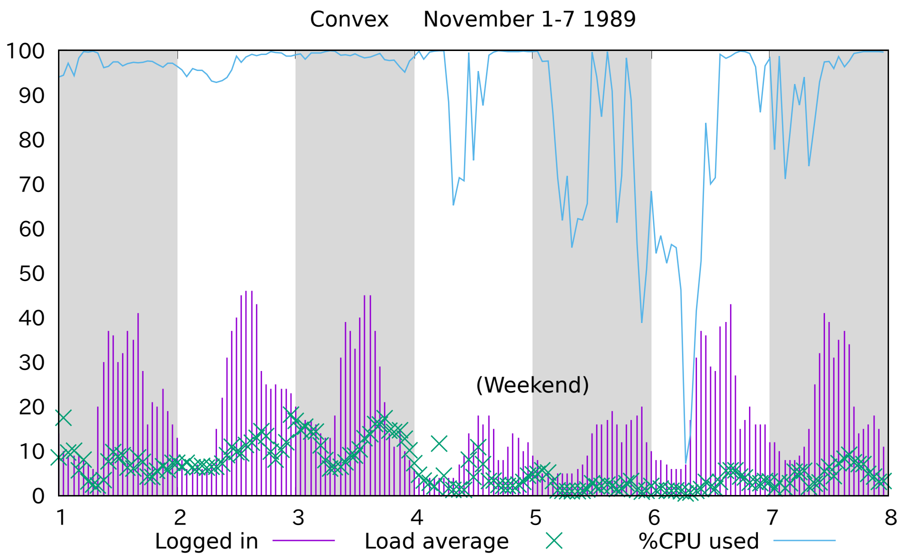

Point labels show which column they came from
Columns
A
B
C
D
0
2
4
6
8
10
12
Column values
line 1
line 2
line 3
line 4
Line in data file

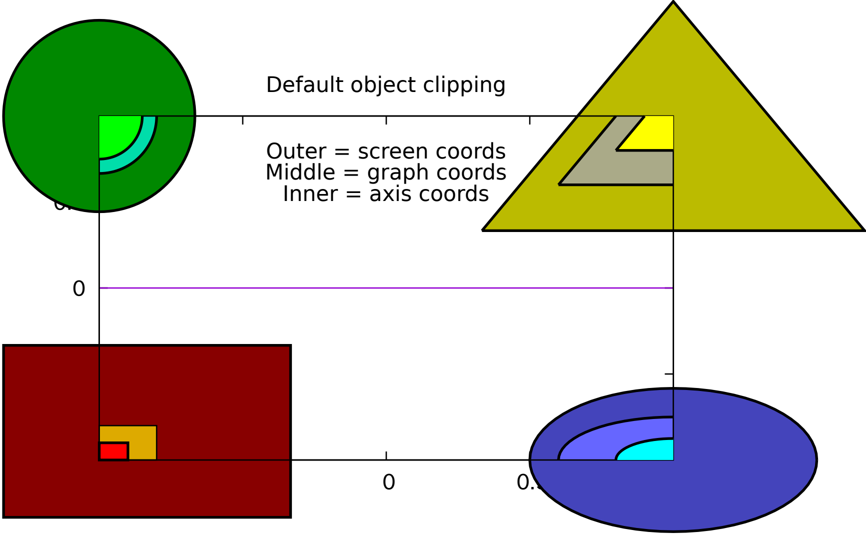
Demo of placing multiple plots (2D and 3D)
with explicit alignment of plot borders



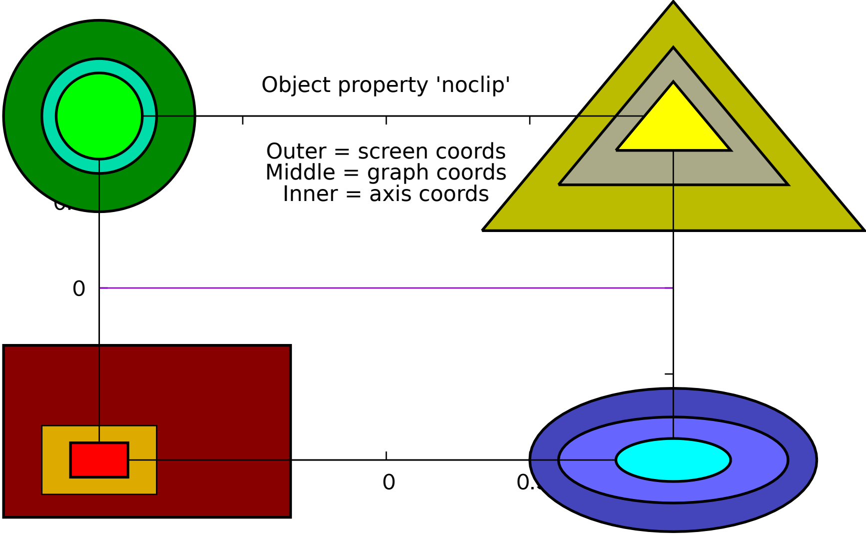


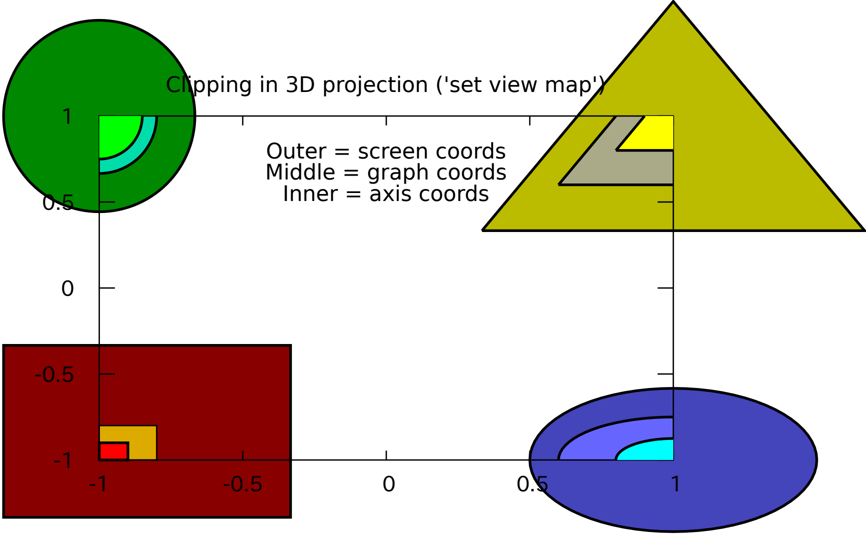
6
4
2
0
-2
-4
-6
4
2
0
-2
-4
x
-3+sin(x*5)/x
There should be a
clipped rectangle here
Label in a box


Convex November 1-7 1989
100
90
80
70
60
50
40
30
20
10
0
(Weekend)
1
2
3
4
5
6
7
8
Logged in
Load average
%CPU used

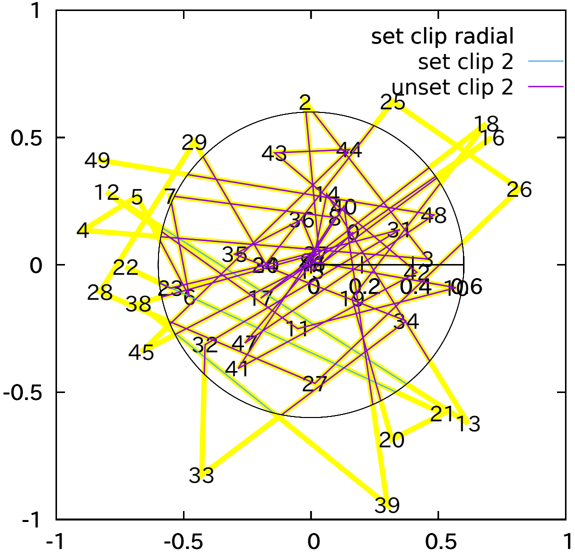


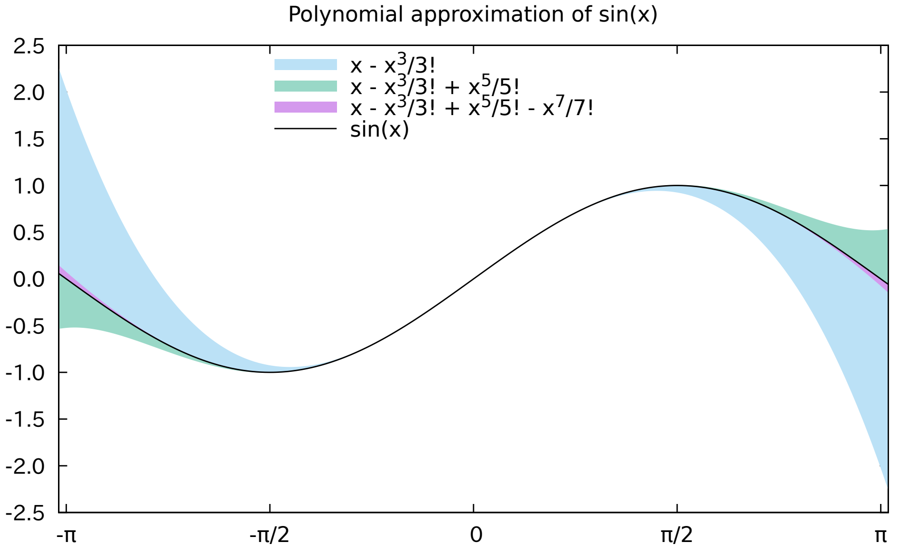
Default object clipping
Outer = screen coords
Middle = graph coords
Inner = axis coords
0
0

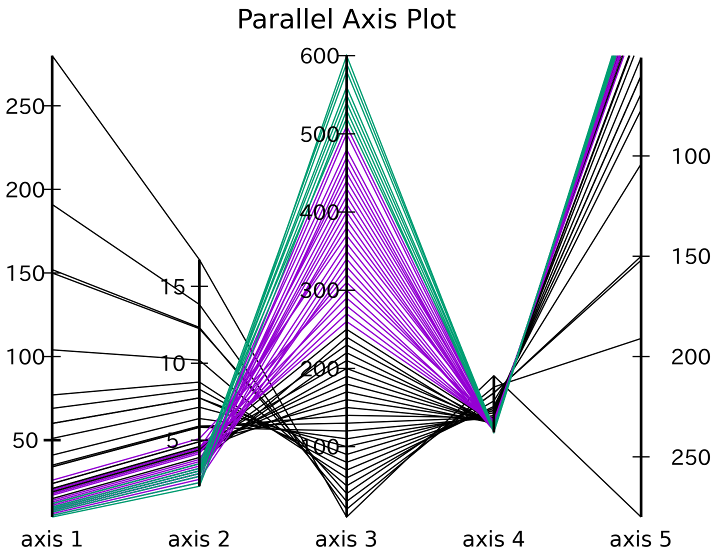

Object property 'noclip'
Outer = screen coords
Middle = graph coords
Inner = axis coords
0
0



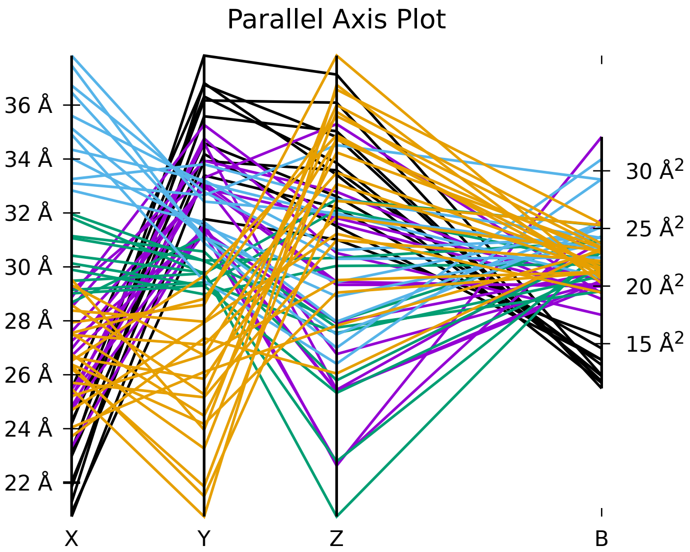
Clipping in 3D projection ('set view map')
Outer = screen coords
Middle = graph coords
Inner = axis coords
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

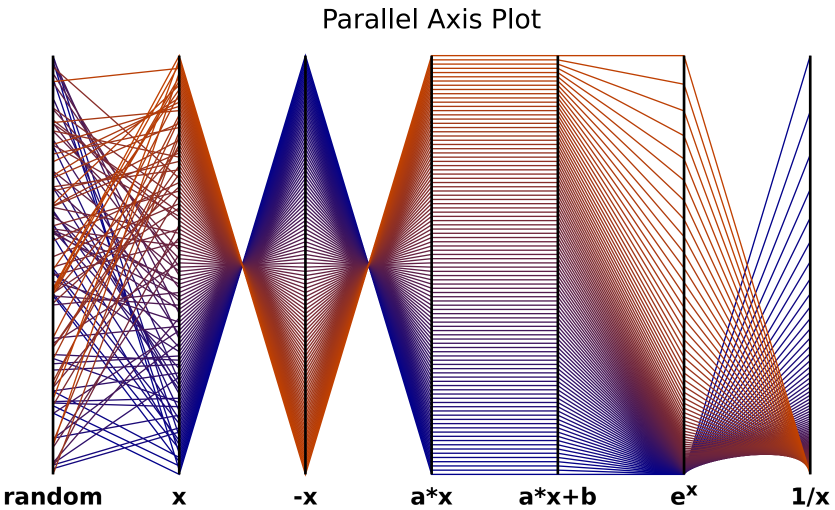

set clip radial
set clip 2
unset clip 2
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

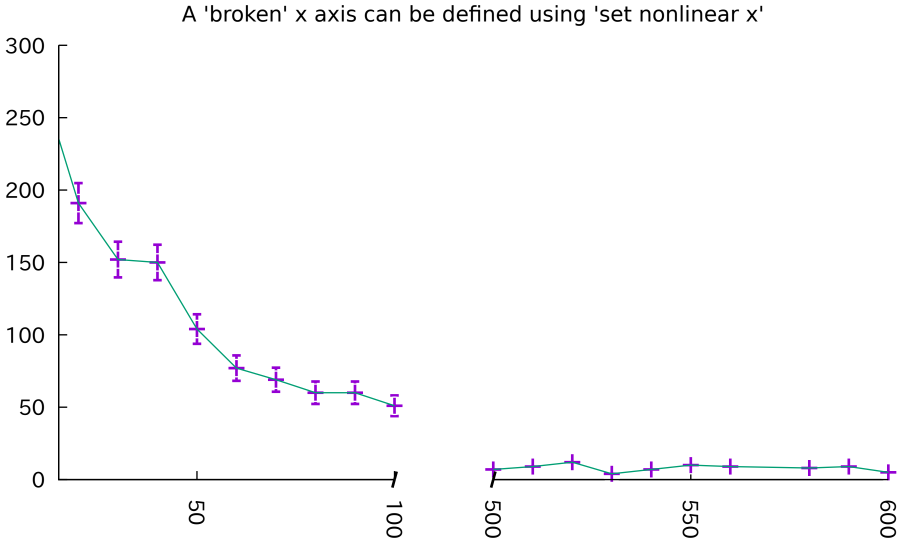

Polynomial approximation of sin(x)
x - x3/3!
x - x3/3! + x5/5!
x - x3/3! + x5/5! - x7/7!
sin(x)
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-π
-π/2
0
π/2
π



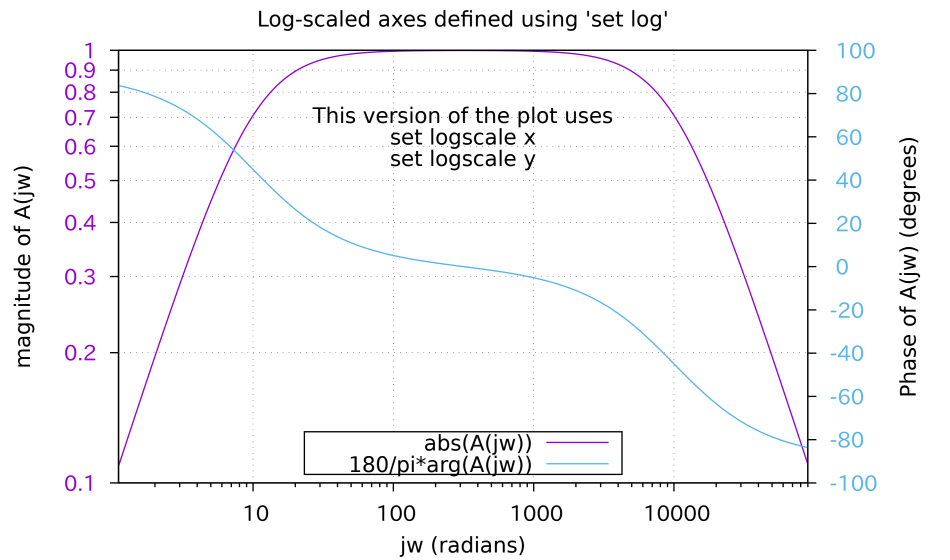
Parallel Axis Plot
250
200
150
100
50
15
10
5
600
500
400
300
200
100
100
150
200
250
axis 1
axis 2
axis 3
axis 4
axis 5

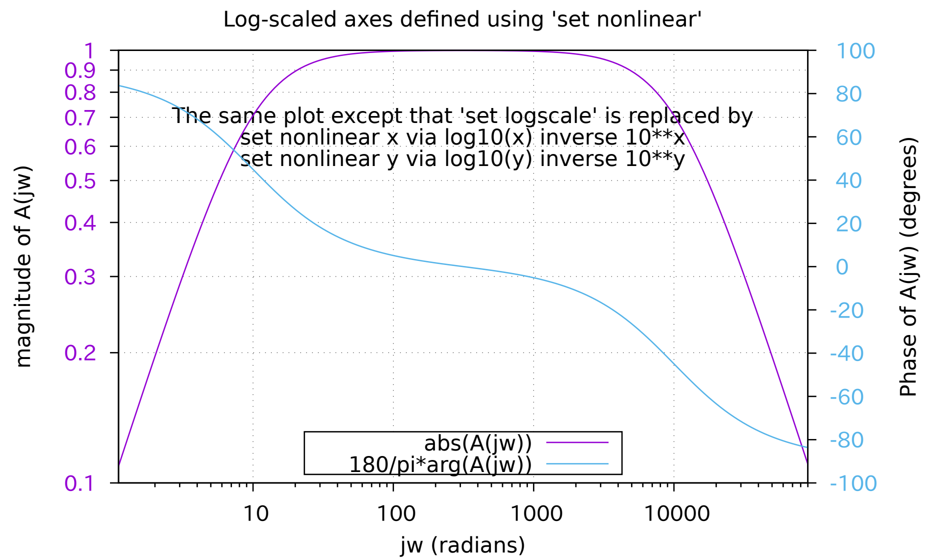

Parallel Axis Plot
36 Å
34 Å
32 Å
30 Å
28 Å
26 Å
24 Å
22 Å
30 Å^2
25 Å^2
20 Å^2
15 Å^2
X
Y
Z
B


Parallel Axis Plot
random
x
-x
a*x
a*x+b
e^x
1/x

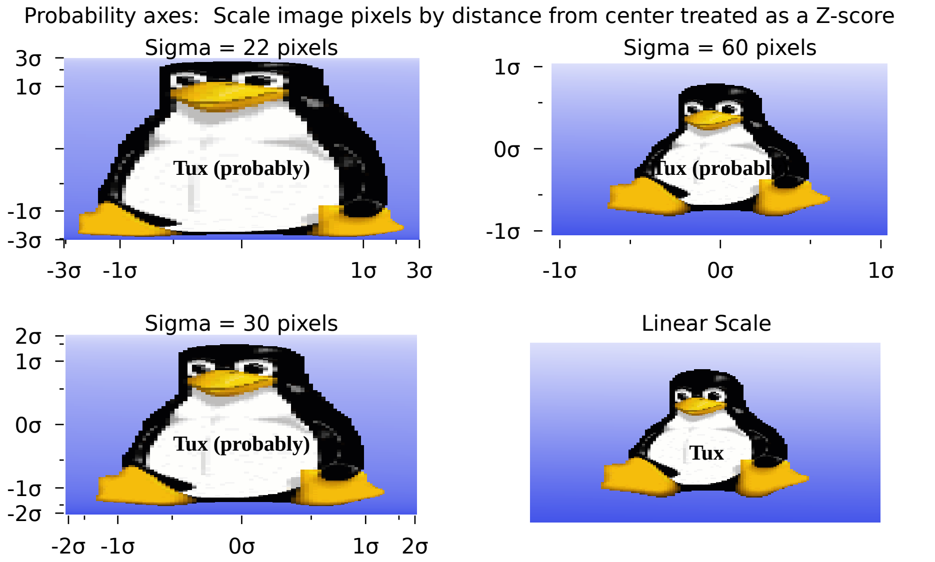


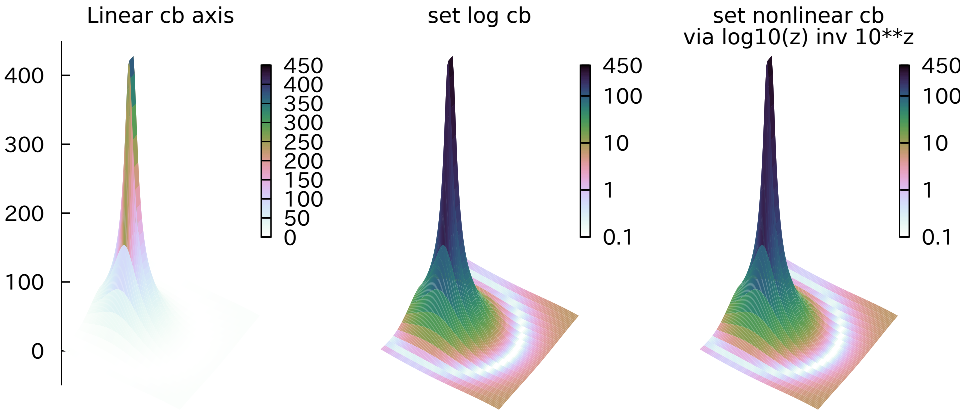
A 'broken' x axis can be defined using 'set nonlinear x'
300
250
200
150
100
50
0
50
100
500
550
600

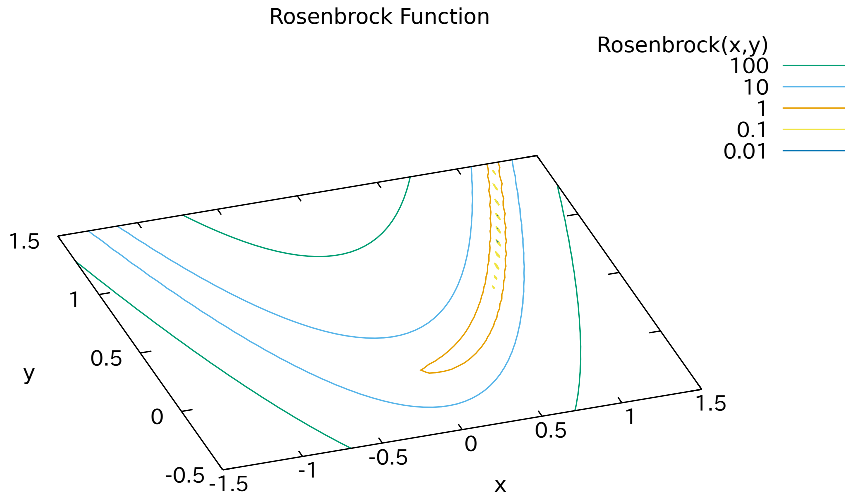

Log-scaled axes defined using 'set log'
This version of the plot uses
set logscale x
set logscale y
abs(A(jw))
180/pi*arg(A(jw))
magnitude of A(jw)
Phase of A(jw) (degrees)
jw (radians)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
100
80
60
40
20
0
-20
-40
-60
-80
-100
10
100
1000
10000

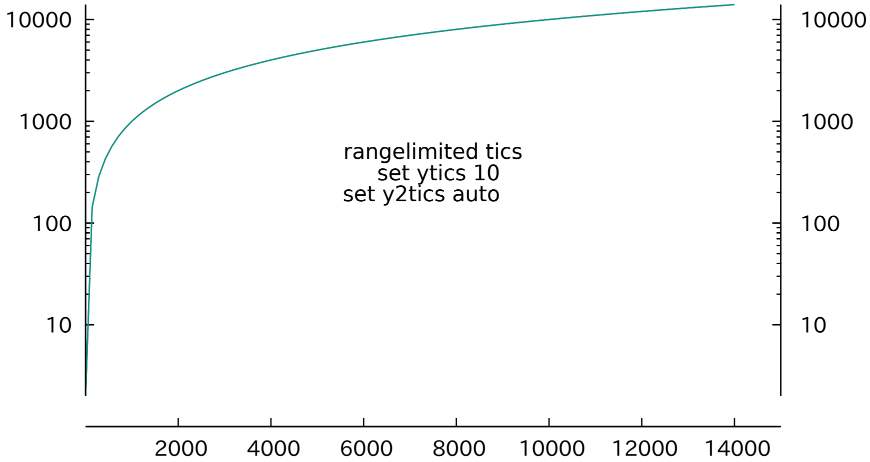


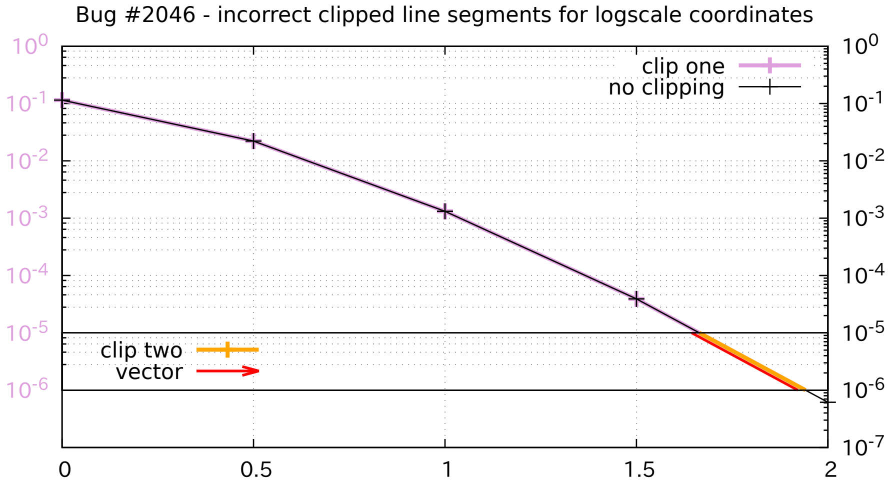
Log-scaled axes defined using 'set nonlinear'
The same plot except that 'set logscale' is replaced by
set nonlinear x via log10(x) inverse 10**x
set nonlinear y via log10(y) inverse 10**y
abs(A(jw))
180/pi*arg(A(jw))
magnitude of A(jw)
Phase of A(jw) (degrees)
jw (radians)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
100
80
60
40
20
0
-20
-40
-60
-80
-100
10
100
1000
10000


Probability axes: Scale image pixels by distance from center treated as a Z-score
Sigma = 22 pixels
3σ
1σ
-1σ
-3σ
Tux (probably)
-3σ
-1σ
1σ
3σ
Sigma = 60 pixels
1σ
0σ
-1σ
Tux (probabl
-1σ
0σ
1σ
Sigma = 30 pixels
2σ
1σ
0σ
-1σ
-2σ
Tux (probably)
-2σ
-1σ
0σ
1σ
2σ
Linear Scale
Tux

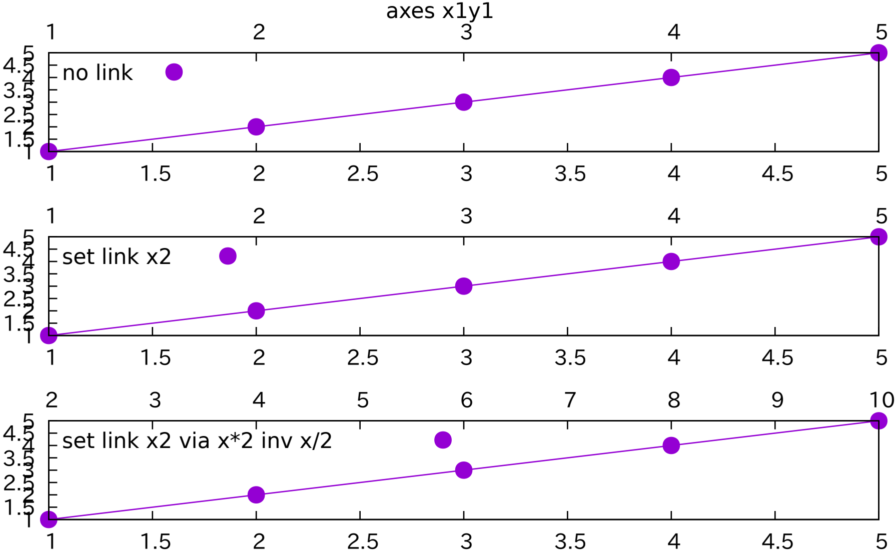

Linear cb axis
set log cb
set nonlinear cb
via log10(z) inv 10**z
400
300
200
100
0
450
400
350
300
250
200
150
100
50
0
450
100
10
1
0.1
450
100
10
1
0.1

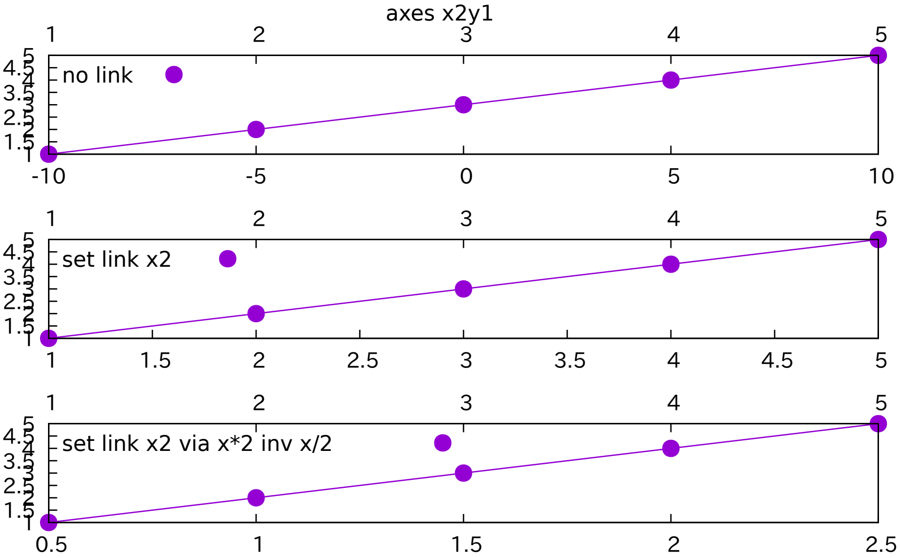

Rosenbrock Function
Rosenbrock(x,y)
100
10
1
0.1
0.01
1.5
1
0.5
0
-0.5
y
-1.5
-1
-0.5
0
0.5
1
1.5
x

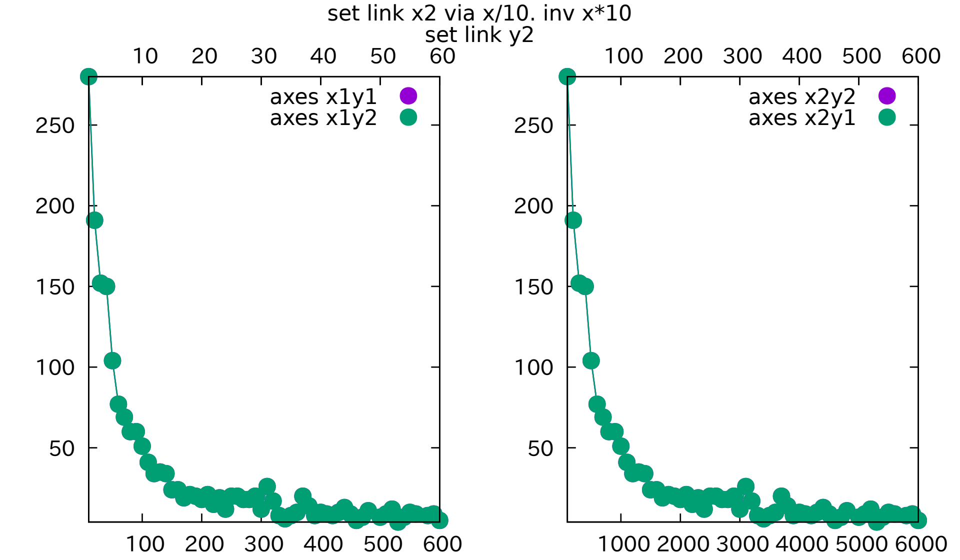

rangelimited tics
set ytics 10
set y2tics auto
10000
1000
100
10
10000
1000
100
10
2000
4000
6000
8000
10000
12000
14000

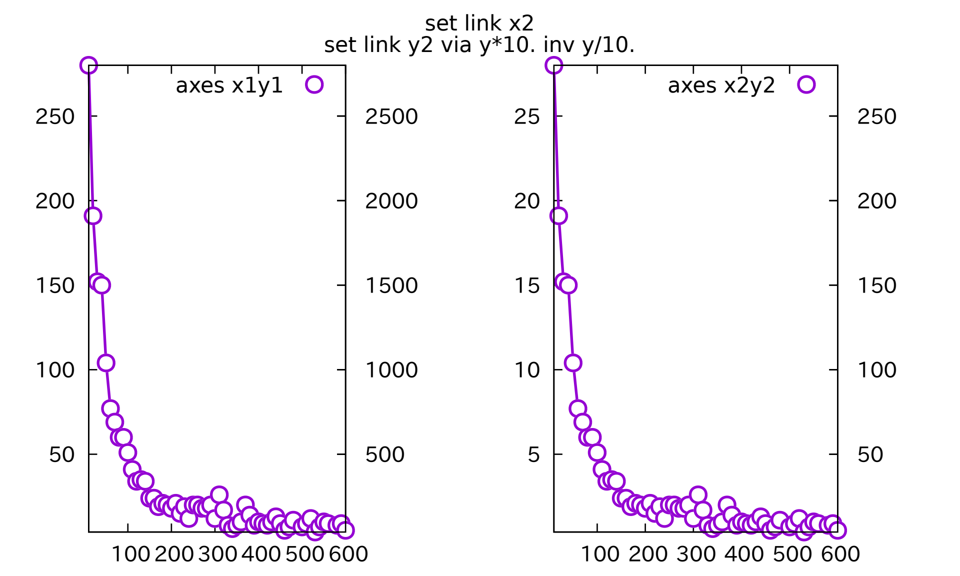


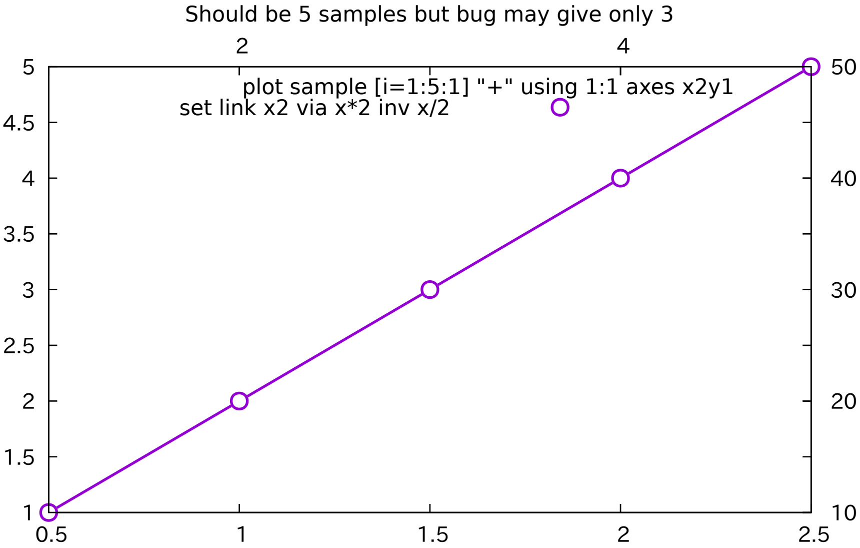
Bug #2046 - incorrect clipped line segments for logscale coordinates
clip one
no clipping
clip two
vector
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
0.5
1
1.5
2


axes x1y1
1
2
3
4
5
no link
1.5
2
2.5
3
3.5
4
4.5
5
set link x2
set link x2 via x*2 inv x/2
6
7
8
9
10


axes x2y1
no link
set link x2
set link x2 via x*2 inv x/2
1
2
3
4
5
1.5
2.5
3.5
4.5
-10
-5
0
10
0.5


set link x2 via x/10. inv x*10
set link y2
axes x1y1
axes x1y2
axes x2y2
axes x2y1

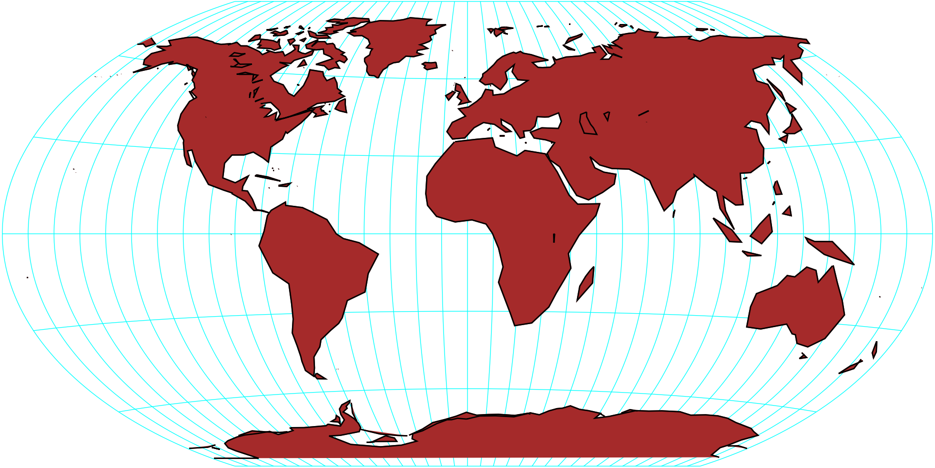

set link x2
set link y2 via y*10. inv y/10.
axes x1y1
axes x2y2
250
200
150
100
50
2500
2000
1500
1000
500
25
20
15
10
5
100 200 300 400 500 600

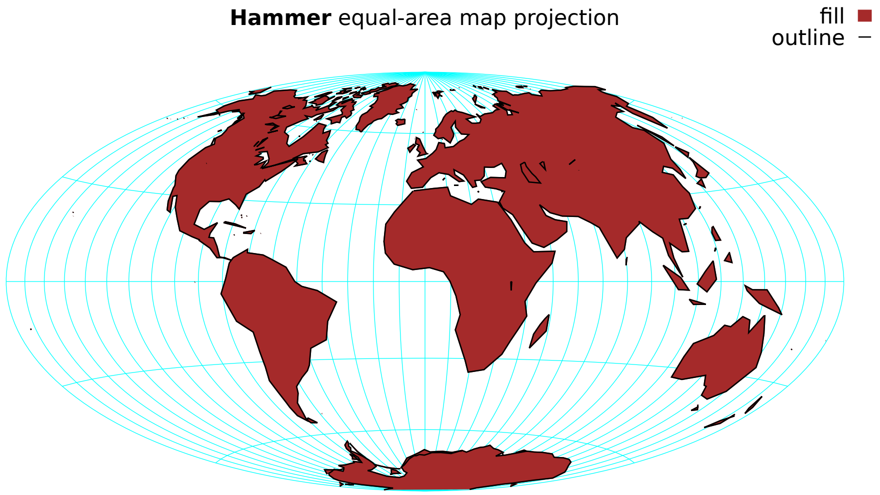


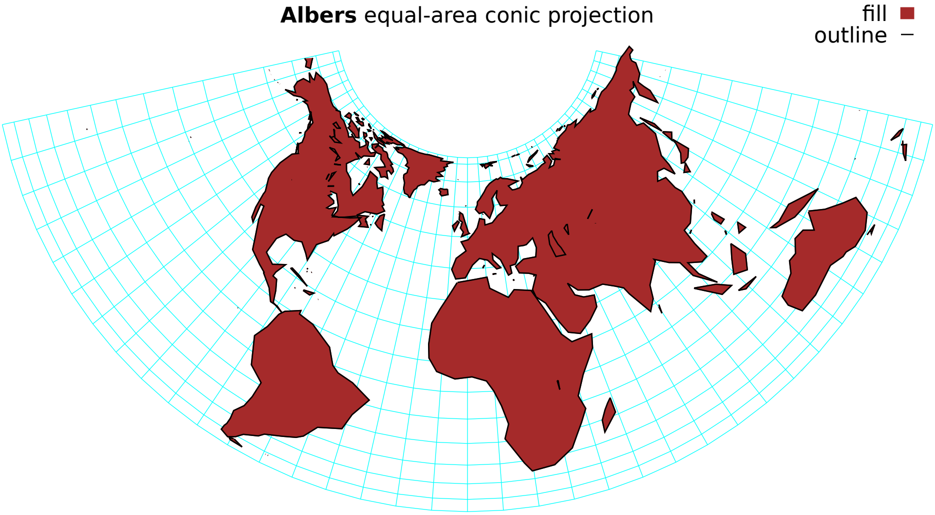
Should be 5 samples but bug may give only 3
2
4
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
1
1.5
2
2.5
50
40
30
20
10
plot sample [i=1:5:1] "+" using 1:1 axes x2y1
set link x2 via x*2 inv x/2

'**Winkel tripel**' map projection

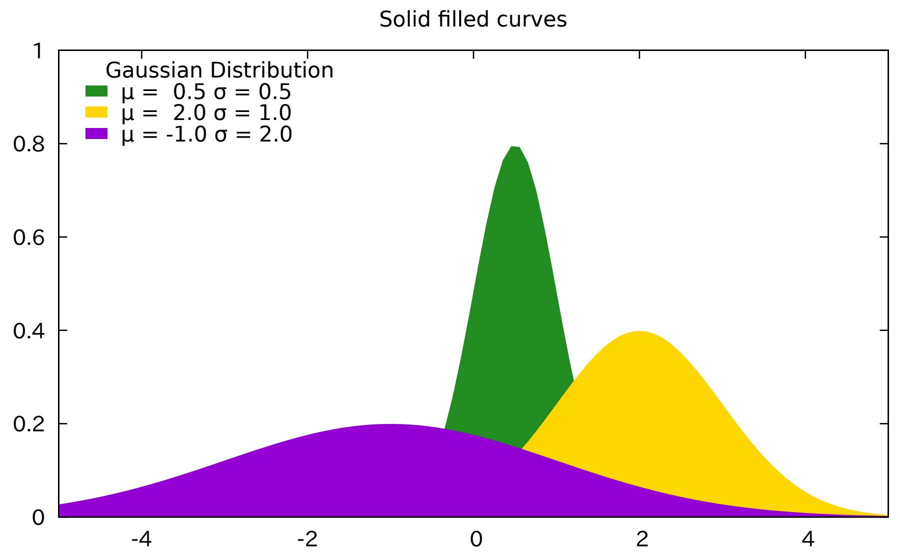
fill
outline

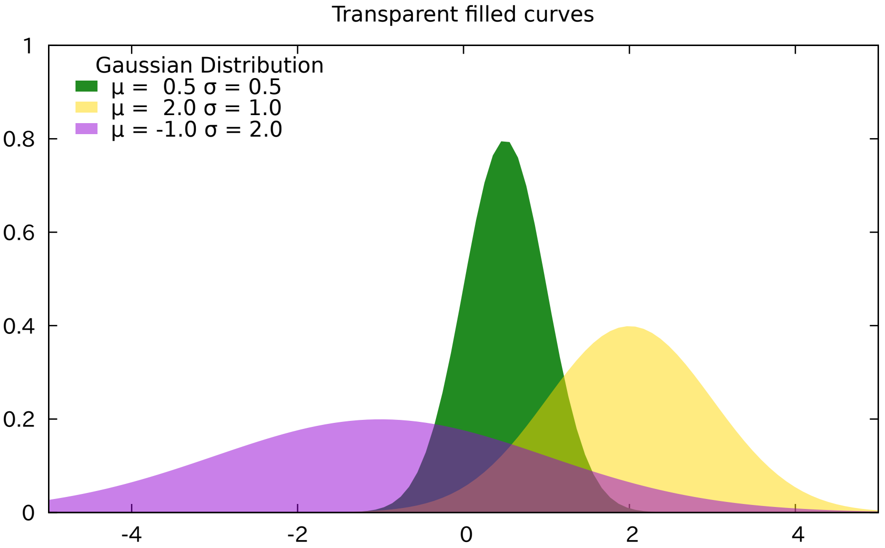



Hammer equal-area map projection
fill
outline

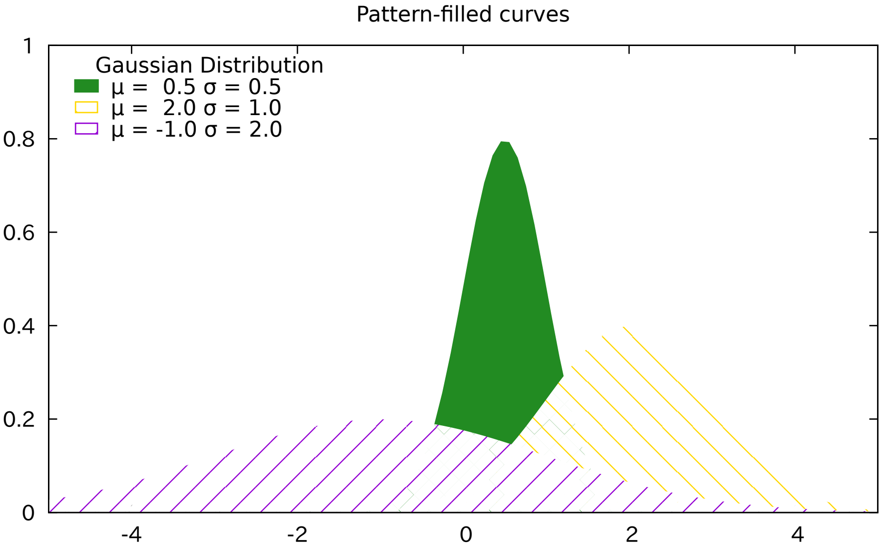


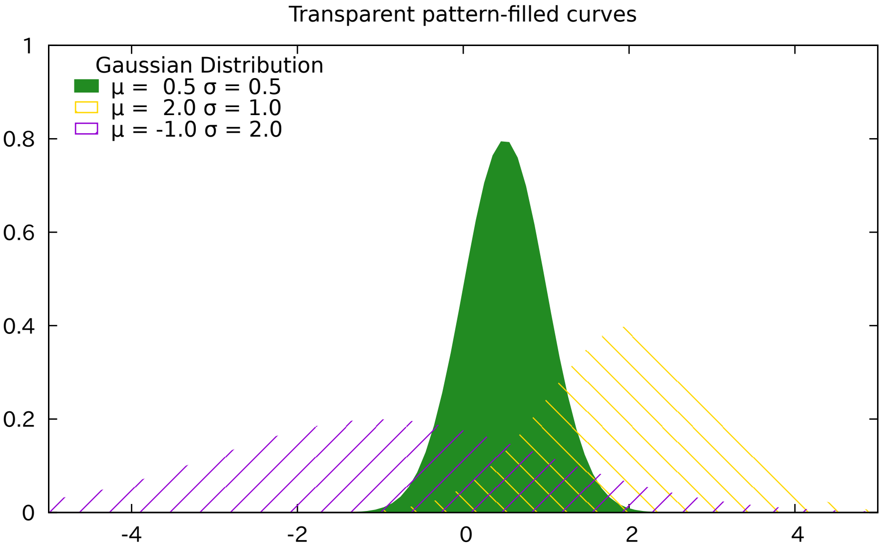
Albers equal-area conic projection
fill
outline


Solid filled curves
Gaussian Distribution
μ = 0.5 σ = 0.5
μ = 2.0 σ = 1.0
μ = -1.0 σ = 2.0
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4

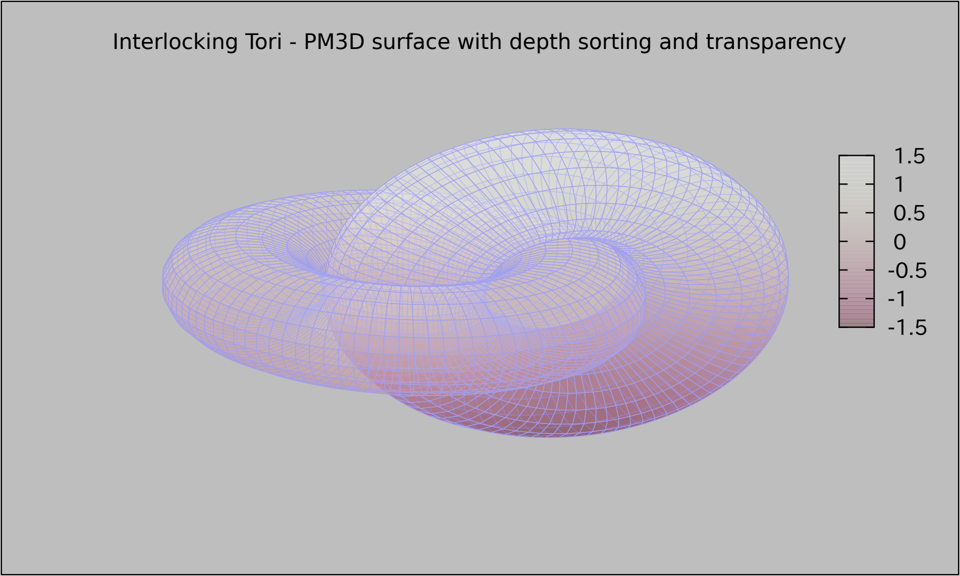


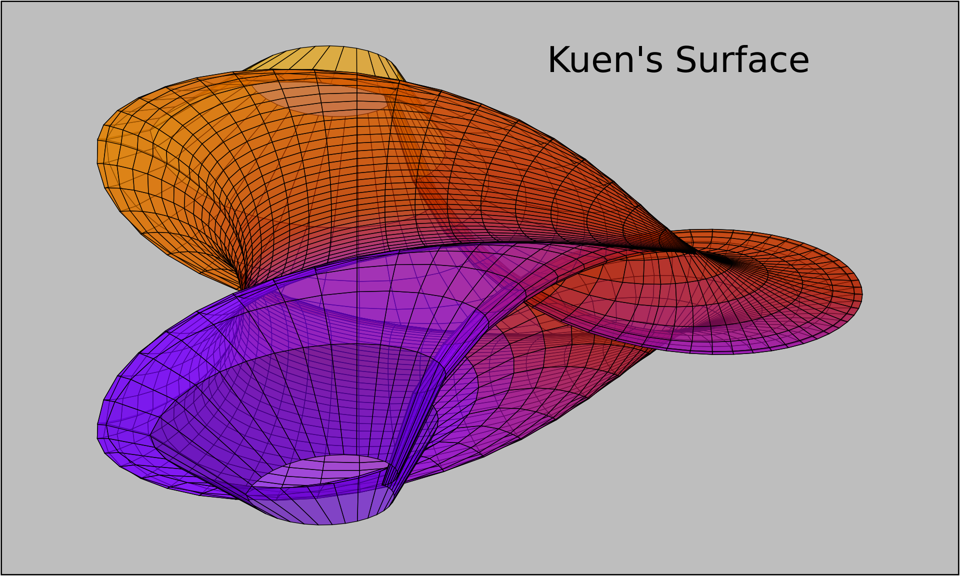
Transparent filled curves
Gaussian Distribution
μ = 0.5 σ = 0.5
μ = 2.0 σ = 1.0
μ = -1.0 σ = 2.0
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4


Pattern-filled curves
Gaussian Distribution
μ = 0.5 σ = 0.5
μ = 2.0 σ = 1.0
μ = -1.0 σ = 2.0
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4

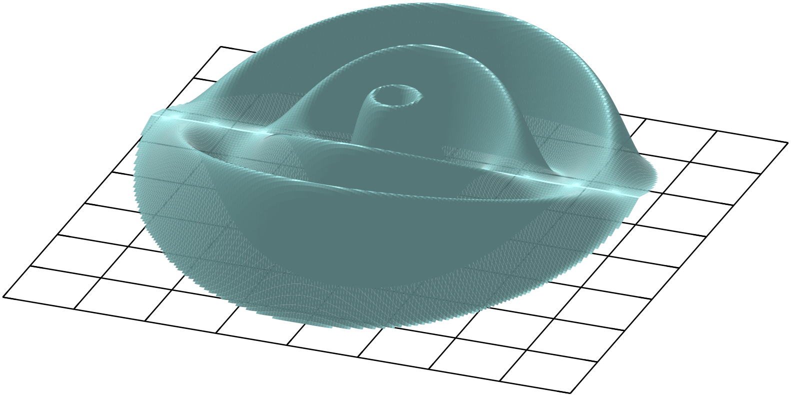


Transparent pattern-filled curves
Gaussian Distribution
μ = 0.5 σ = 0.5
μ = 2.0 σ = 1.0
μ = -1.0 σ = 2.0
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4

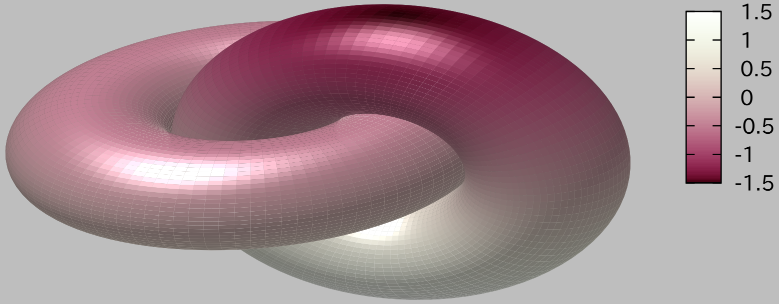

Interlocking Tori - PM3D surface with depth sorting and transparency
1.5
1
0.5
0
-0.5
-1
-1.5


Kuen's Surface

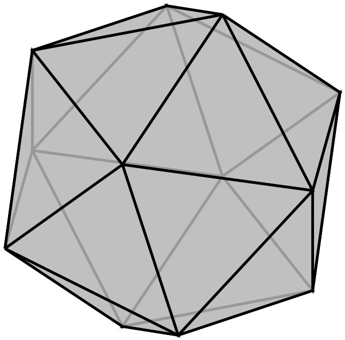
pm3d lighting model with specular highlighting



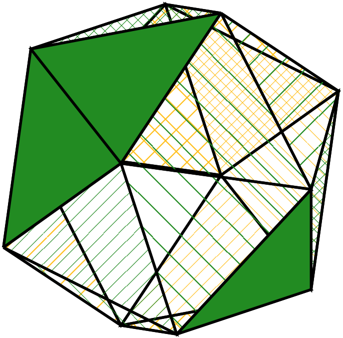
PM3D surfaces with specular highlighting



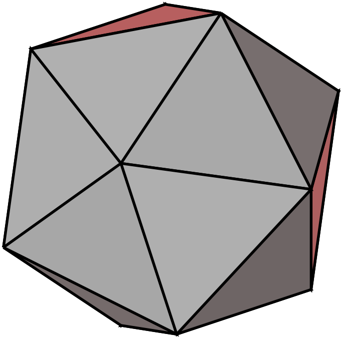
Faces of an icosahedron drawn as 20 individual objects



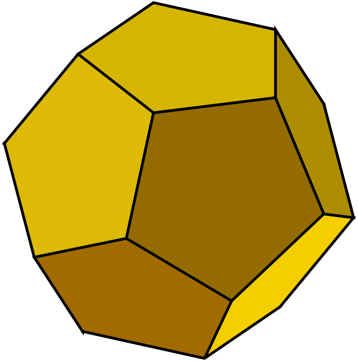
2-sided coloring
green outside, yellow inside



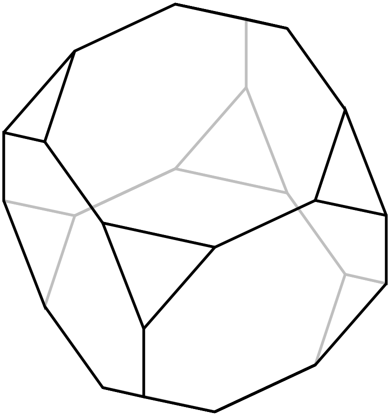
splot icosahedron.dat with polygons



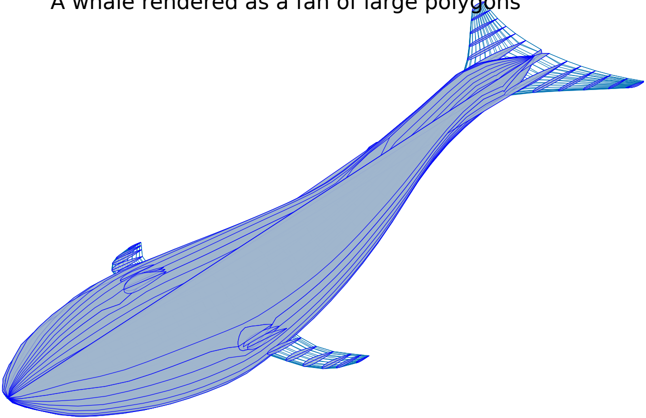
splot dodecahedron.dat with polygons

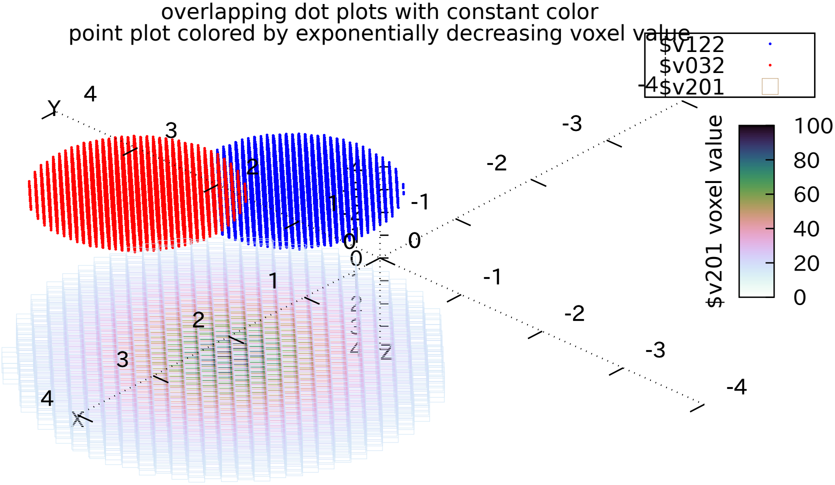


splot truncated_cube with polygons

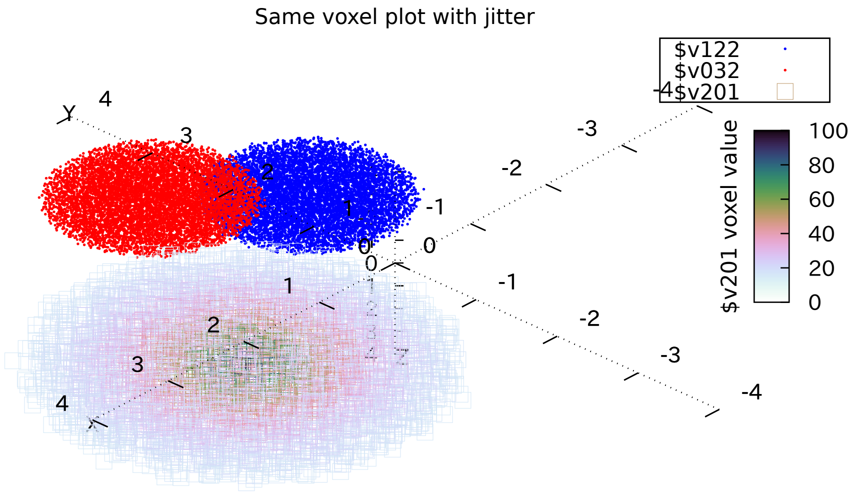


A whale rendered as a fan of large polygons

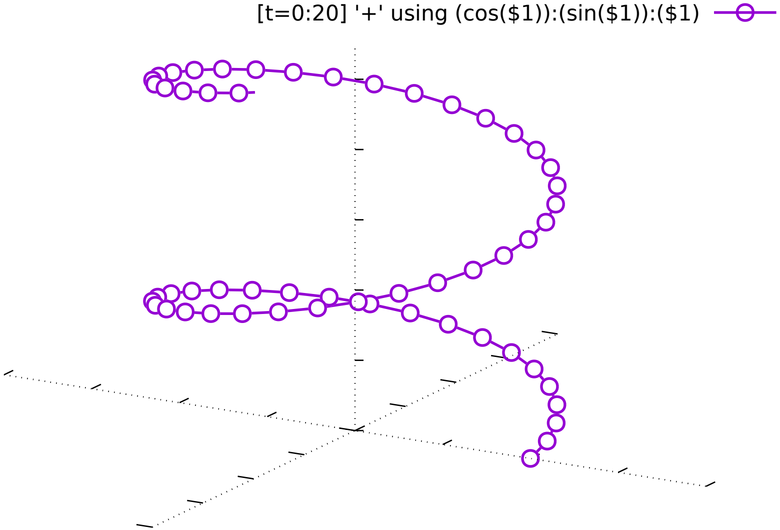


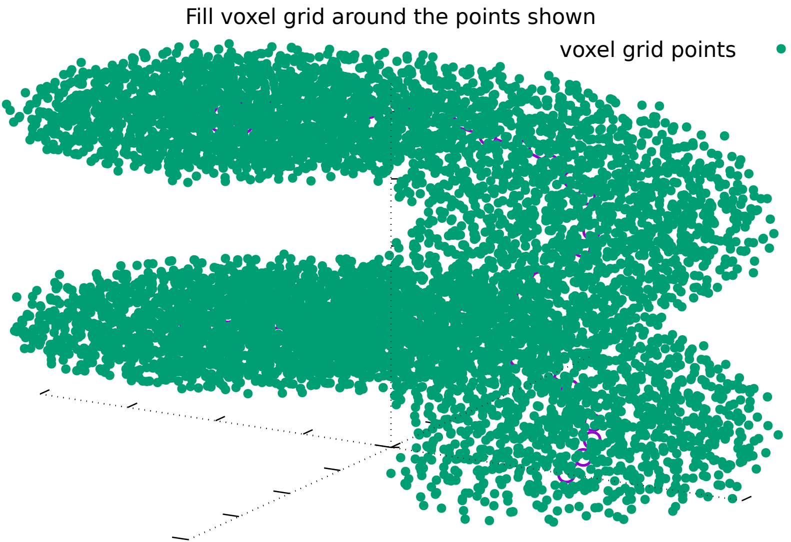


overlapping dot plots with constant color
point plot colored by exponentially decreasing voxel value
$v122
$v032
$v201
$v201 voxel value
100
80
60
40
20
0
X
Y
Z

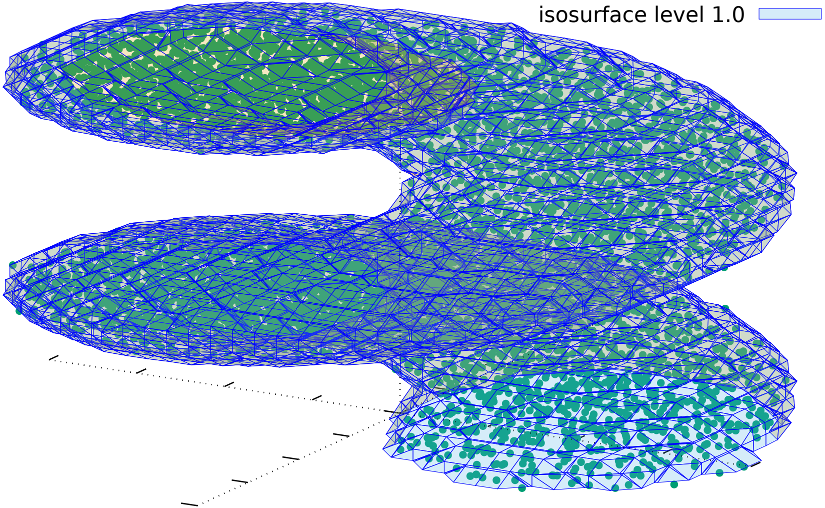


Same voxel plot with jitter
$v122
$v032
$v201
$v201 voxel value
100
80
60
40
20
0
Y
X
Z

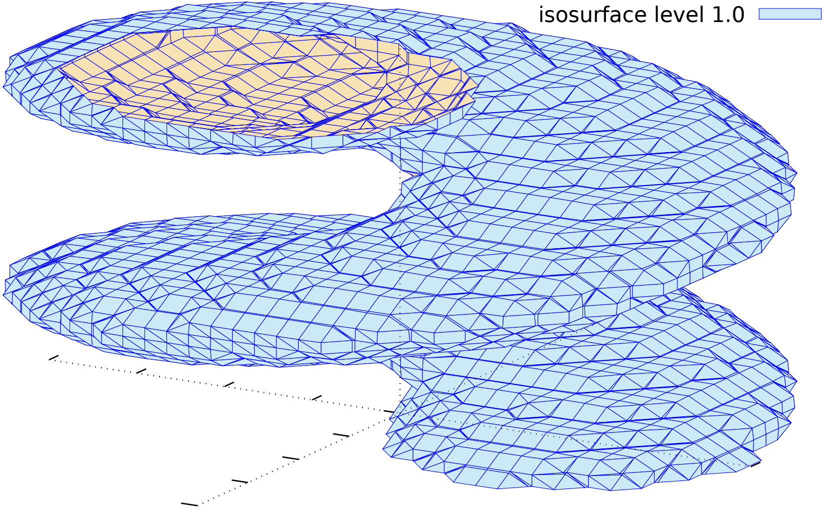

[t=0:20] '+' using (cos($1)):(sin($1)):($1)


Fill voxel grid around the points shown
voxel grid points

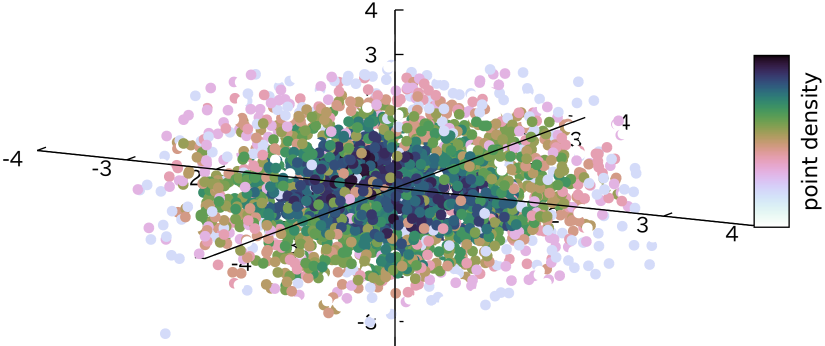
Draw isosurface enclosing all points

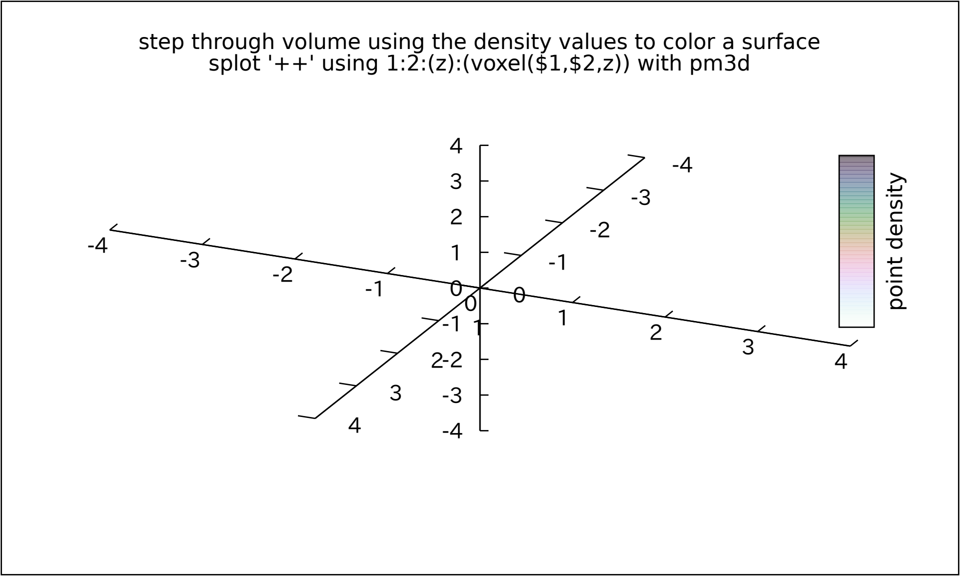


isosurface alone

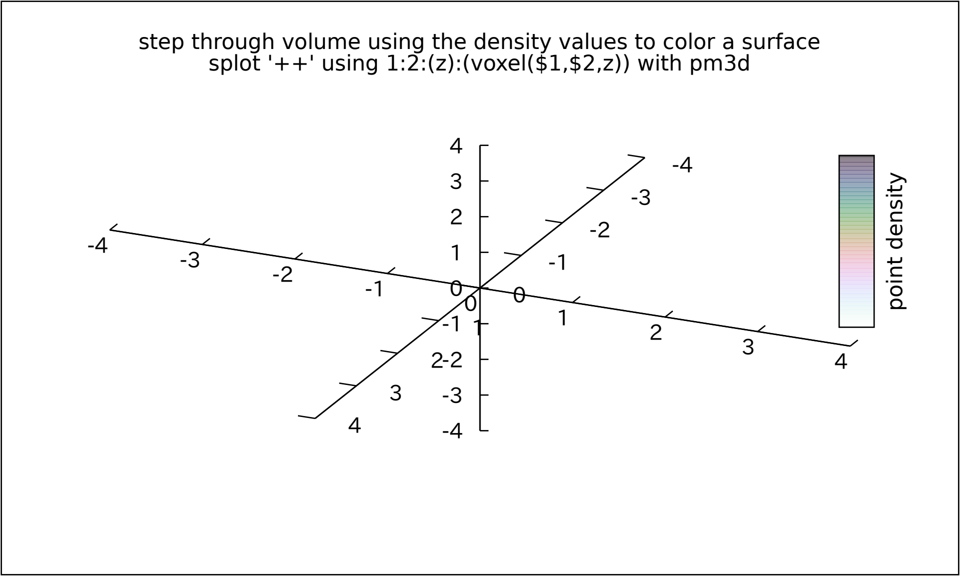


Gaussian 3D cloud of 3000 random samples
colored by local point density

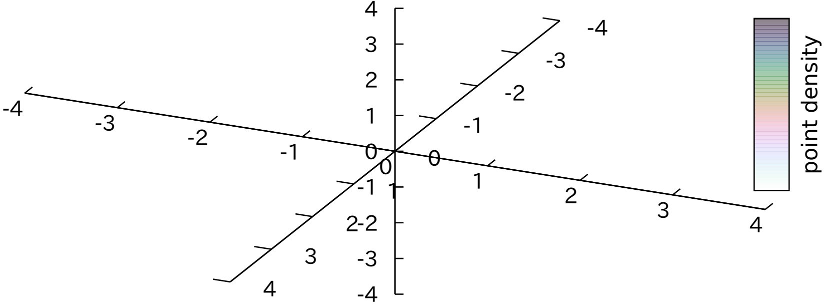


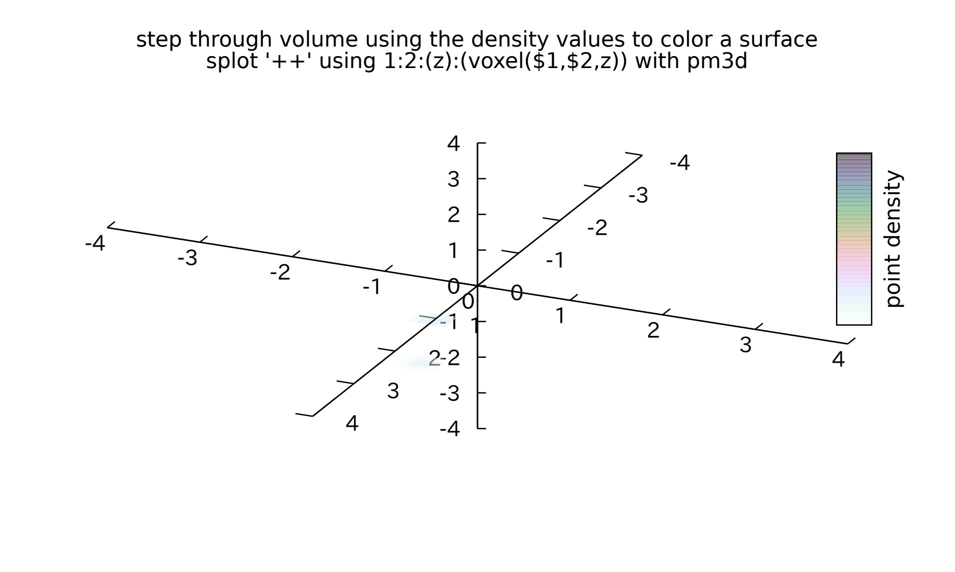


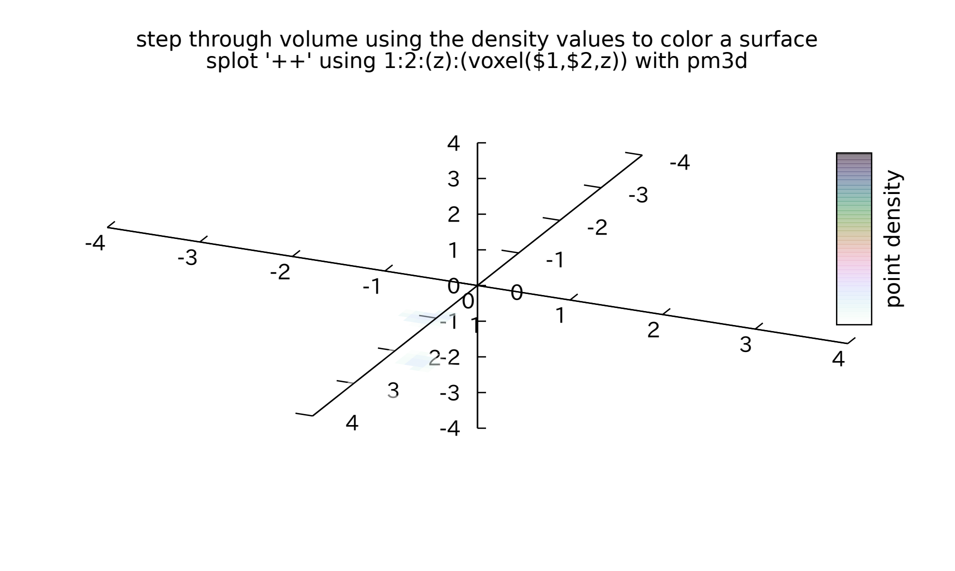
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

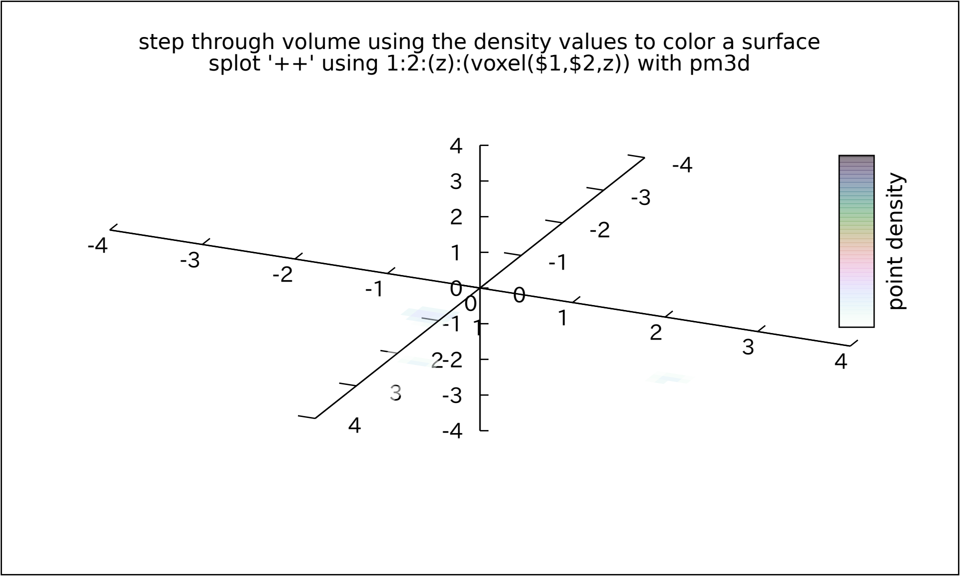

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

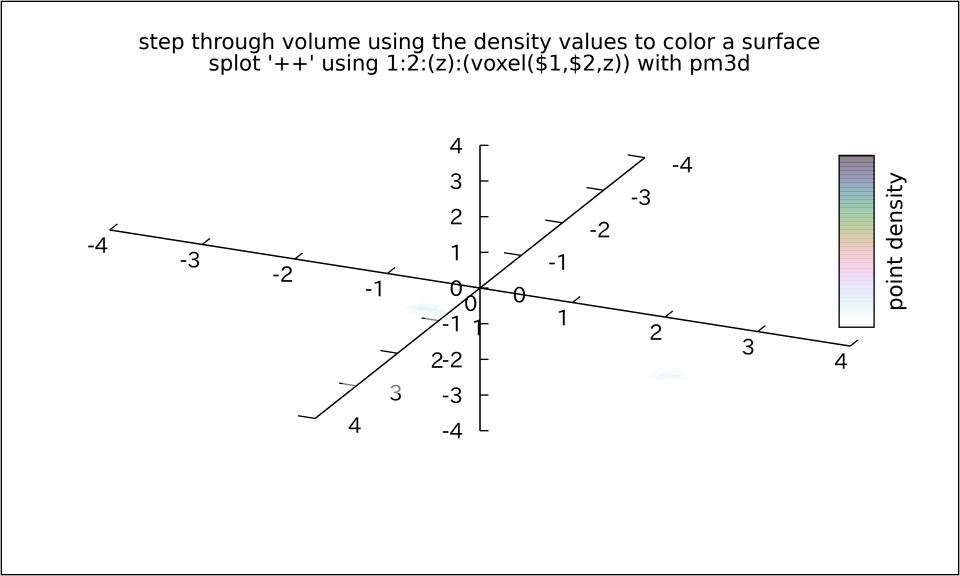


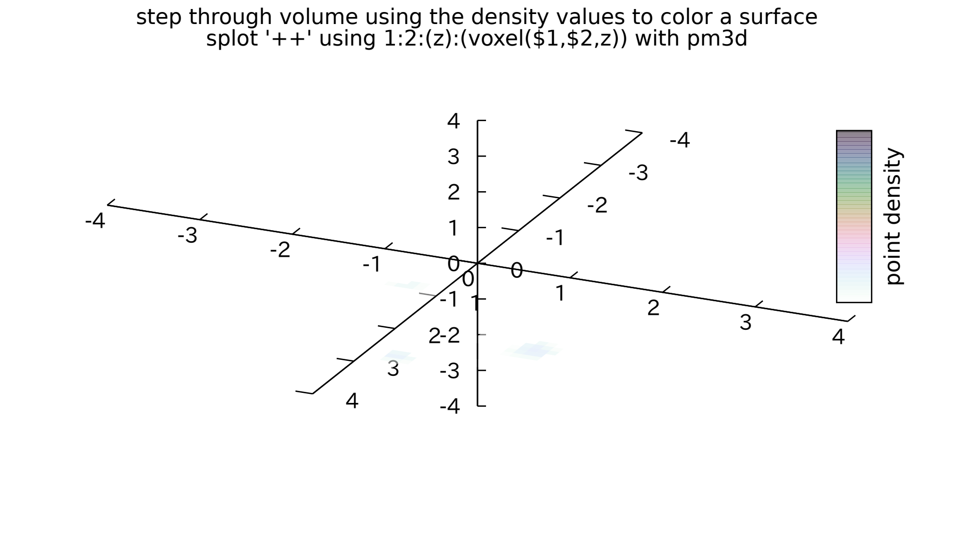

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4



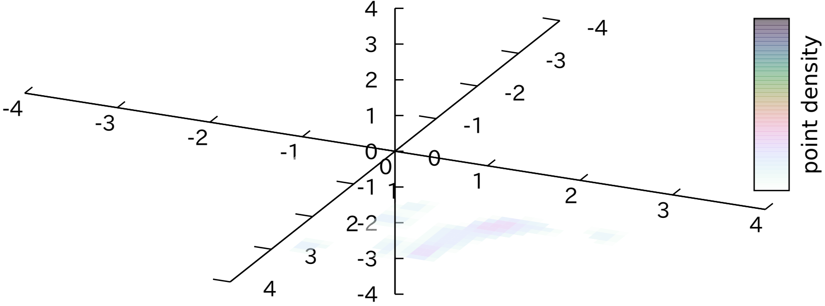
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

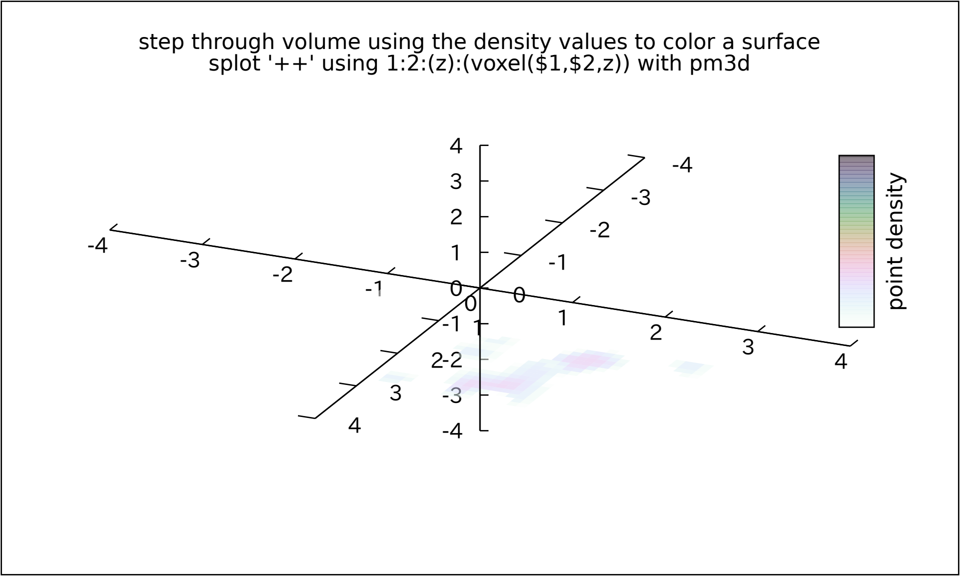

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

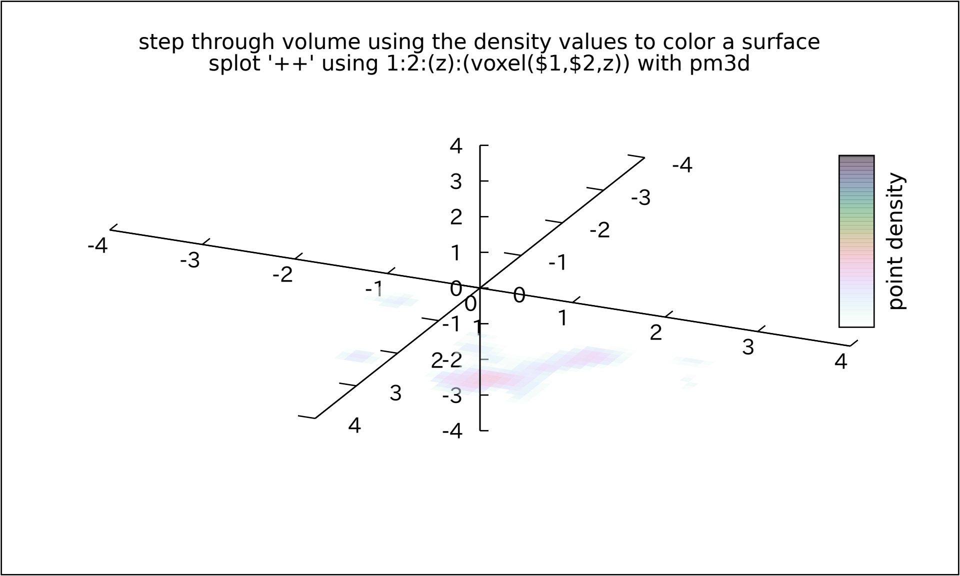


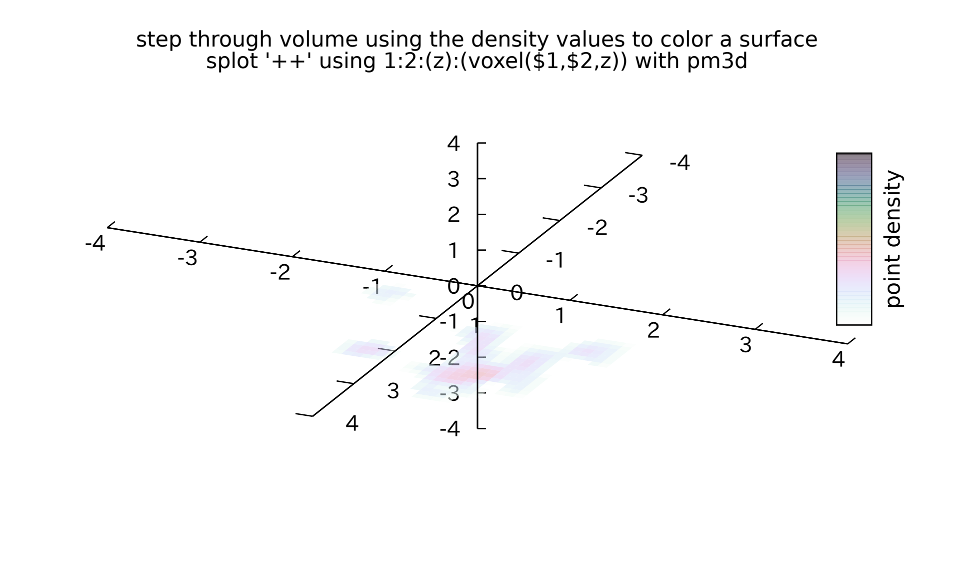
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

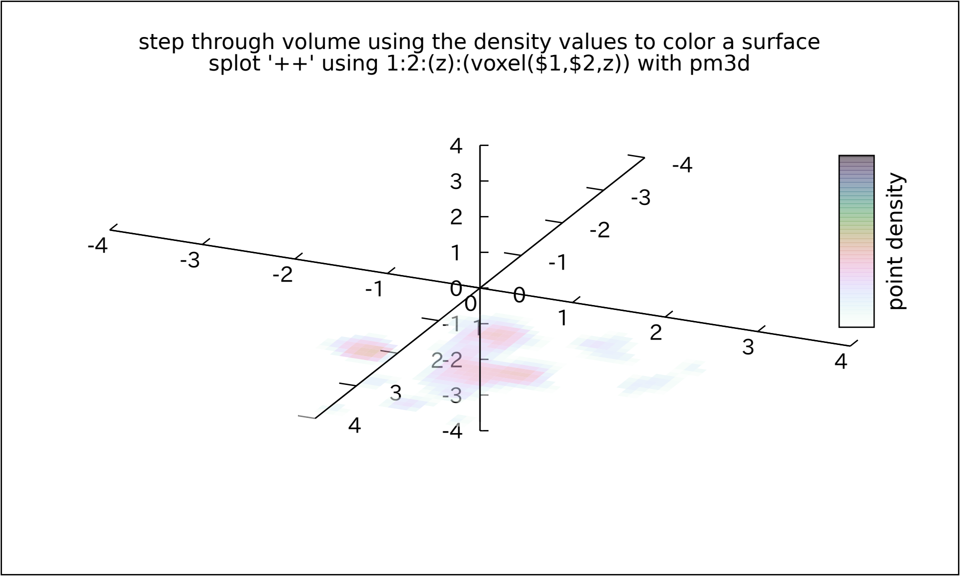
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

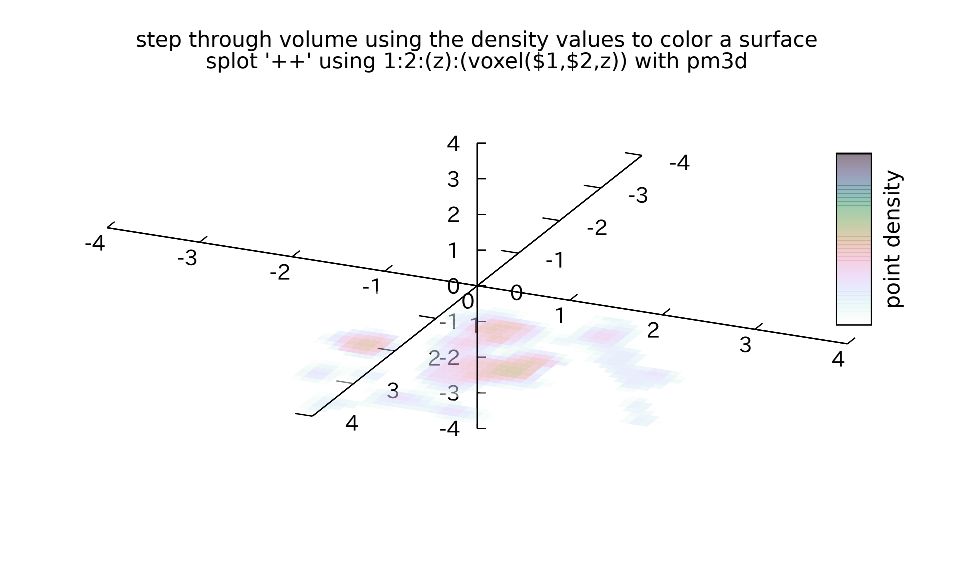


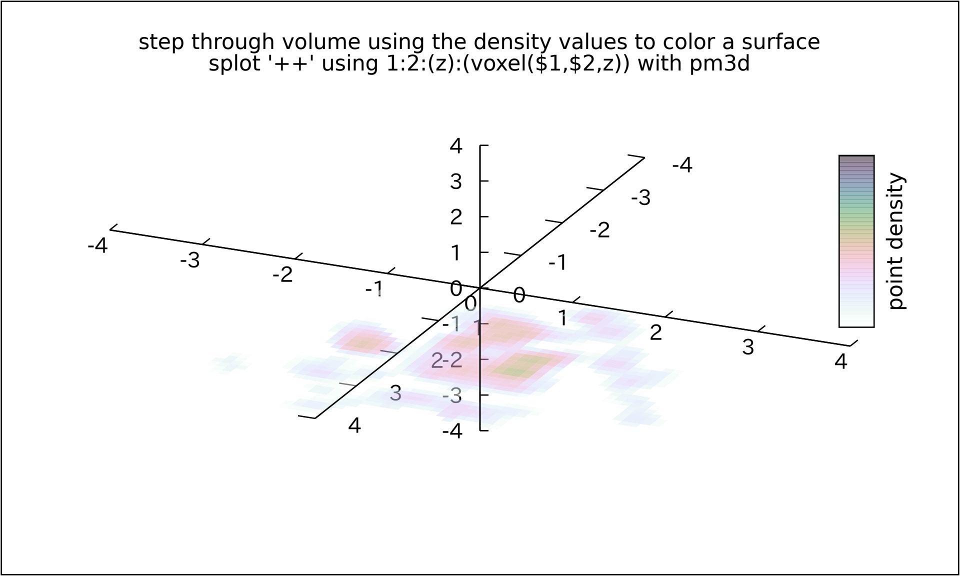

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
4
3
2
1
0
-1
-2
-3
-4

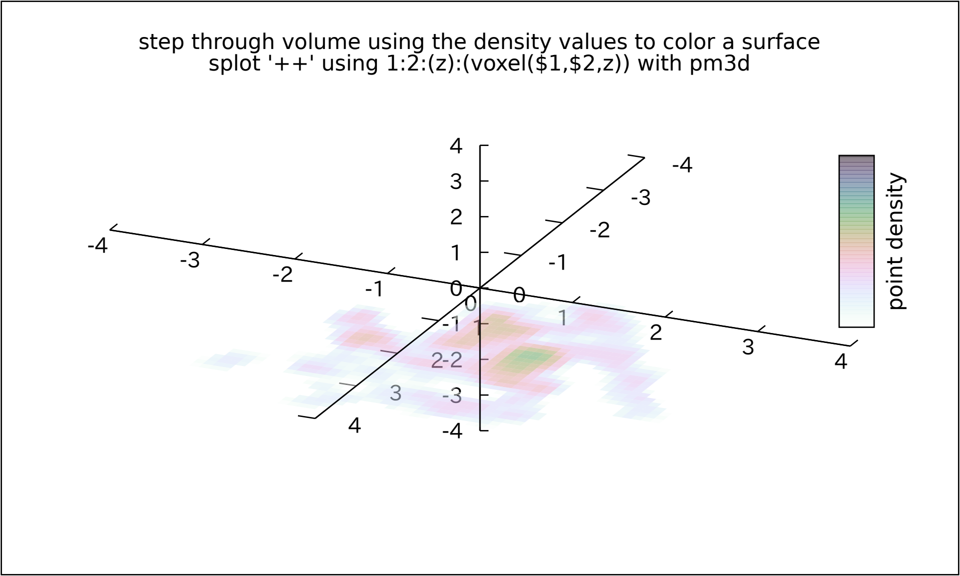

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

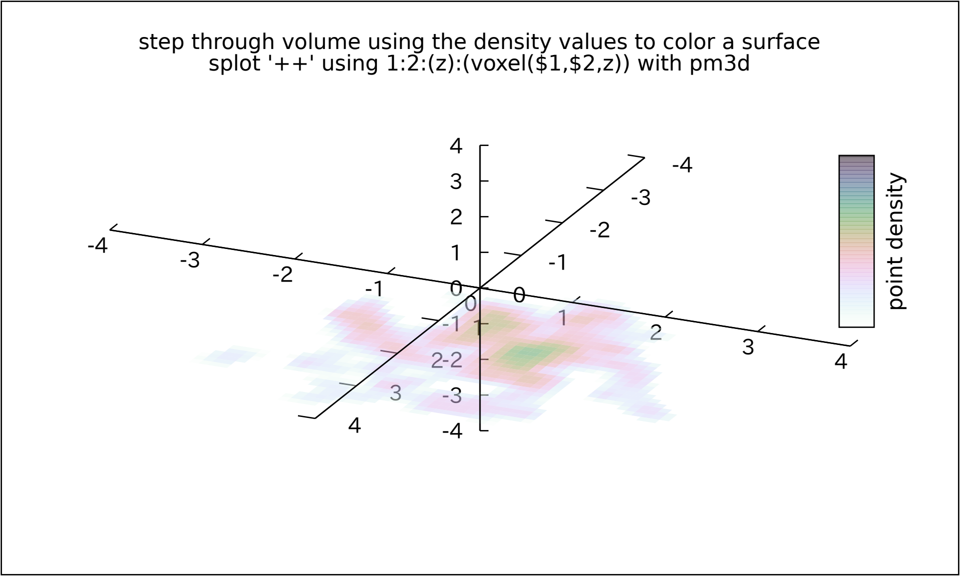

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

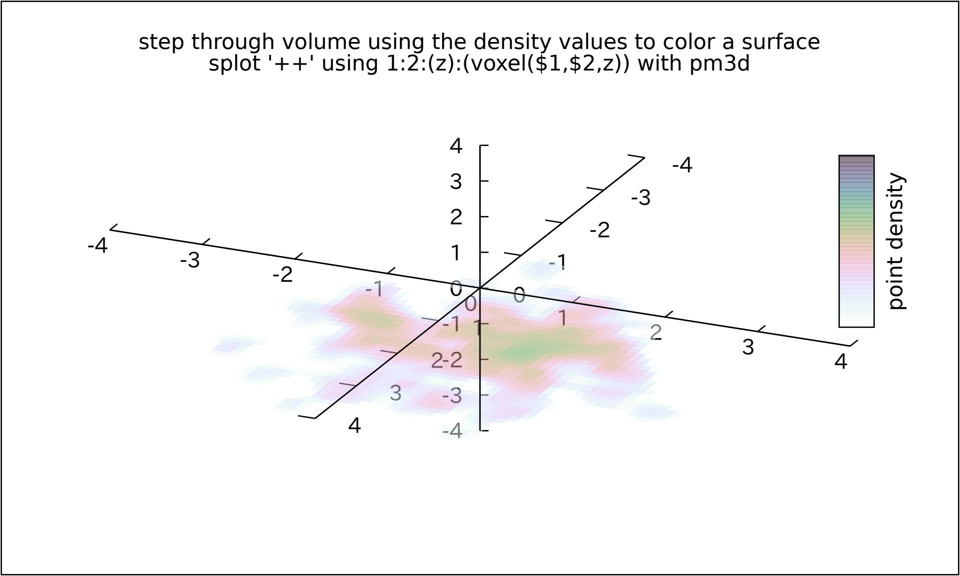

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

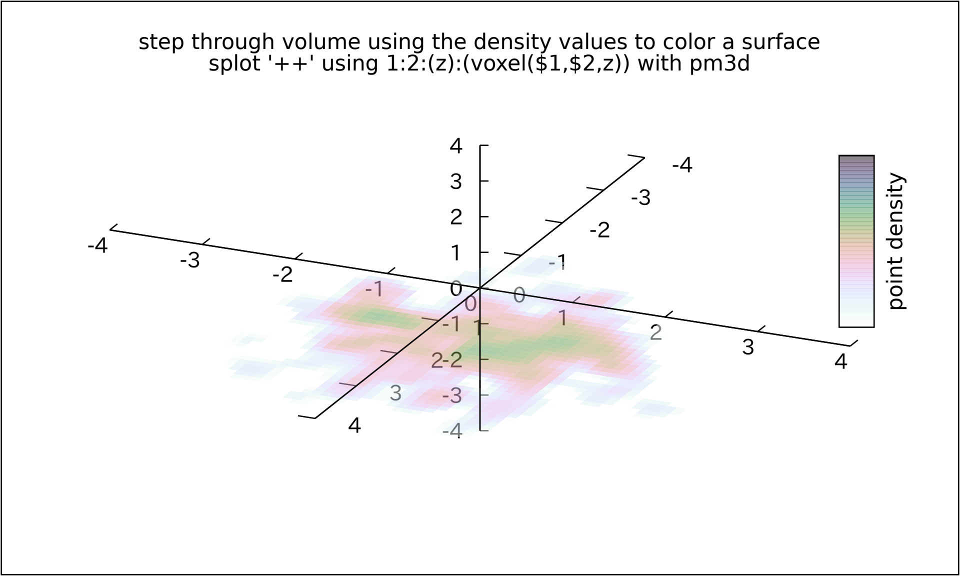

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

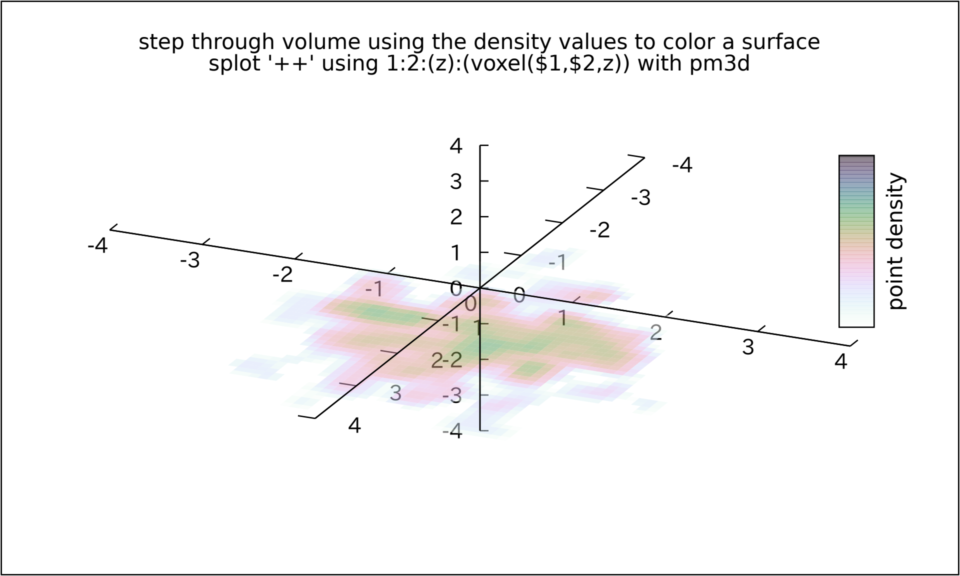

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

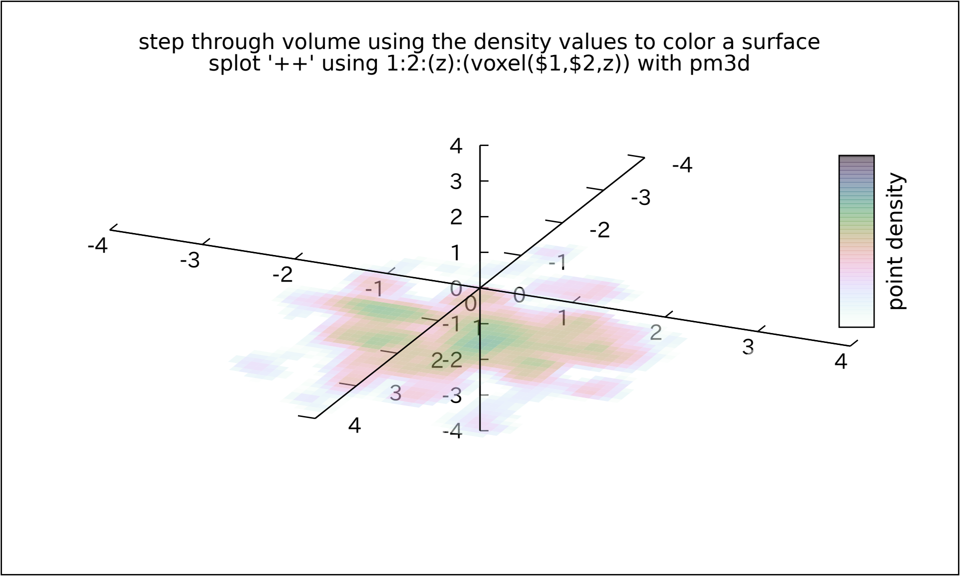

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4



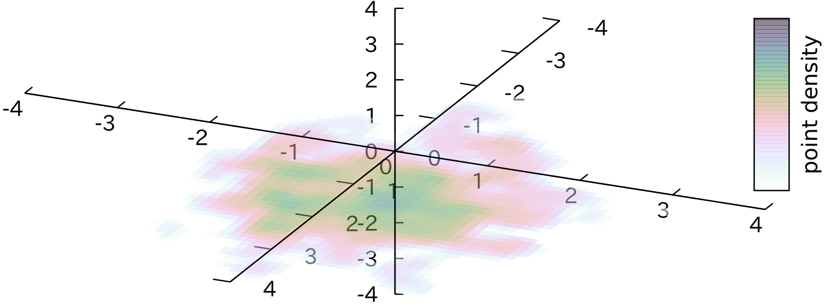
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

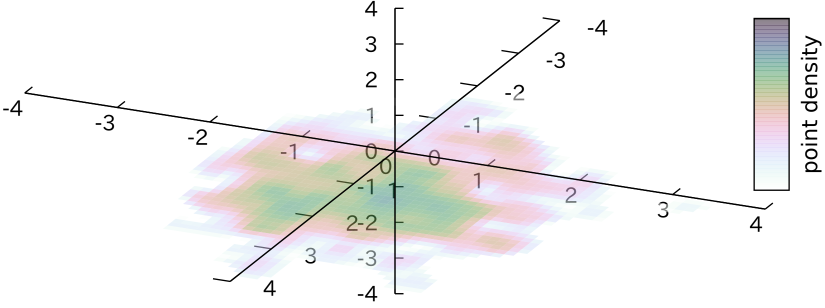

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

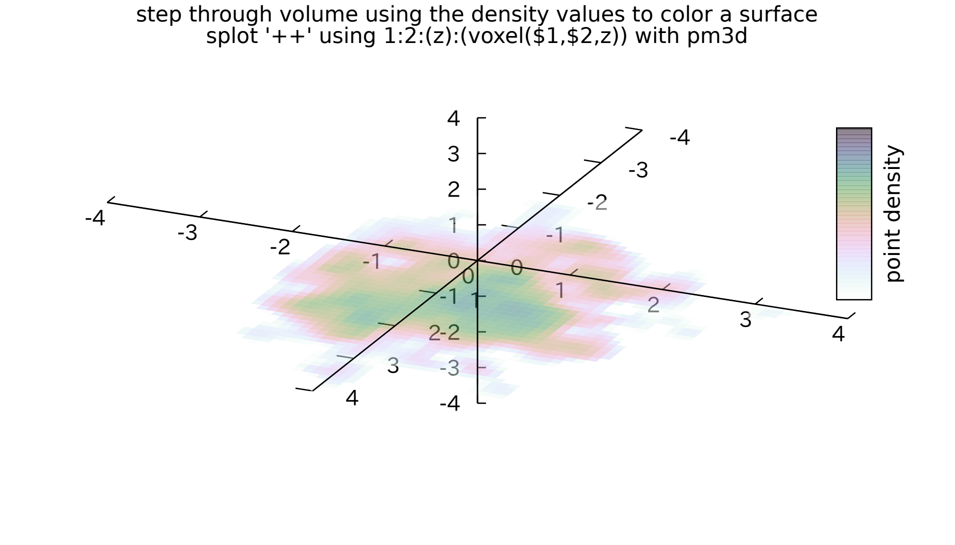

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

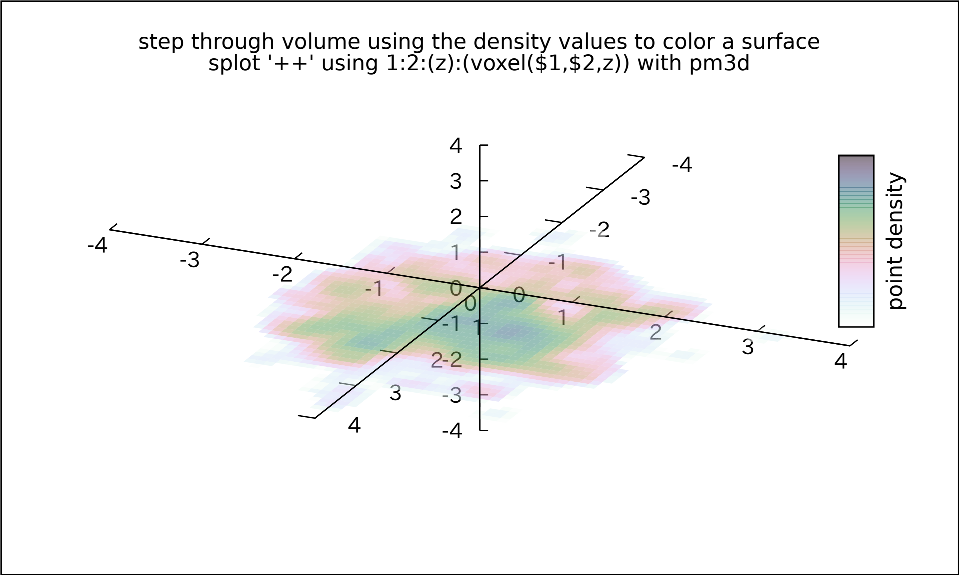
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d



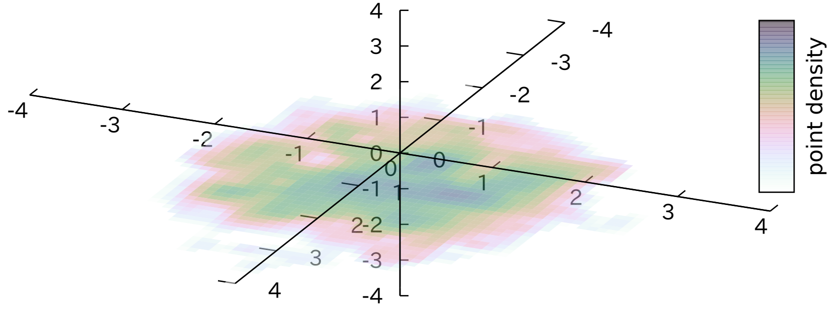
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

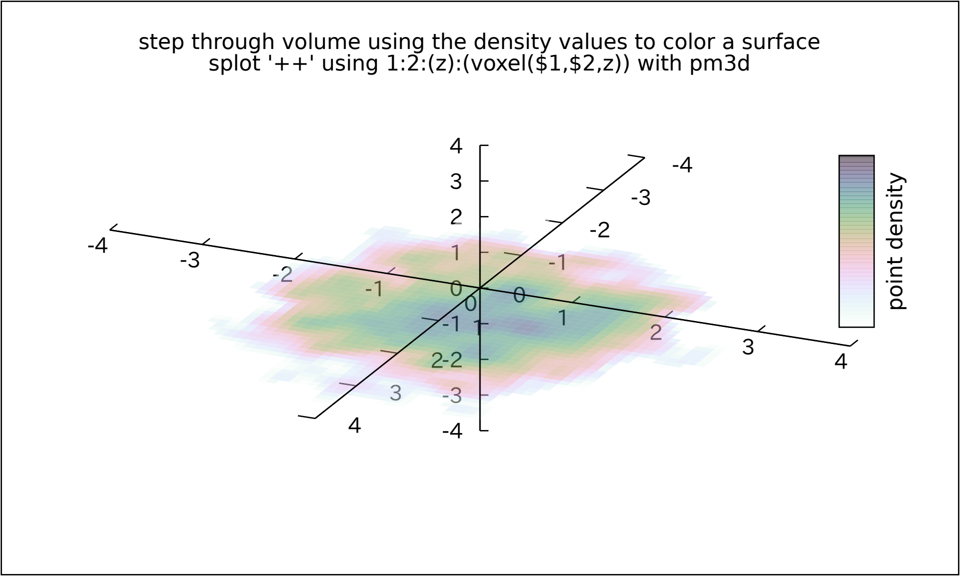


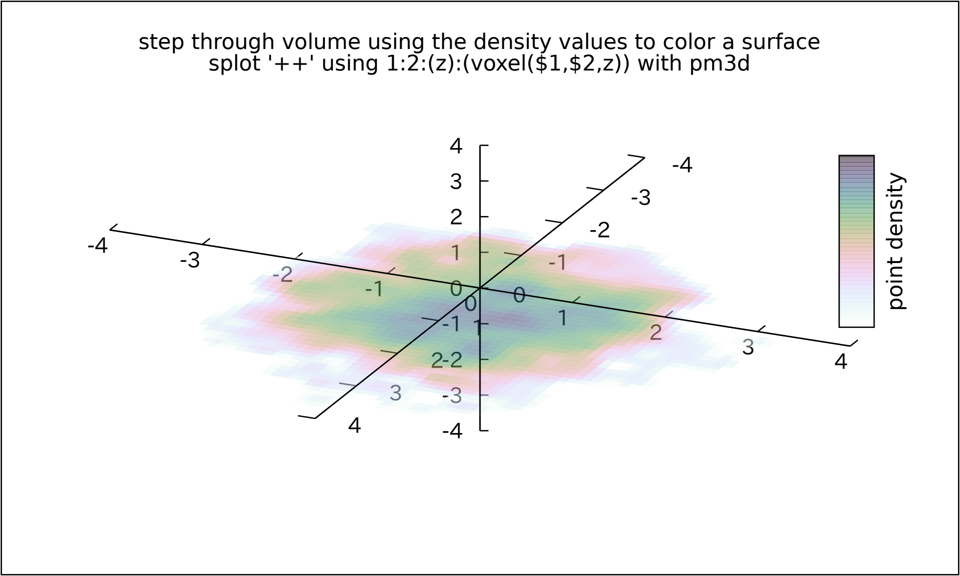

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

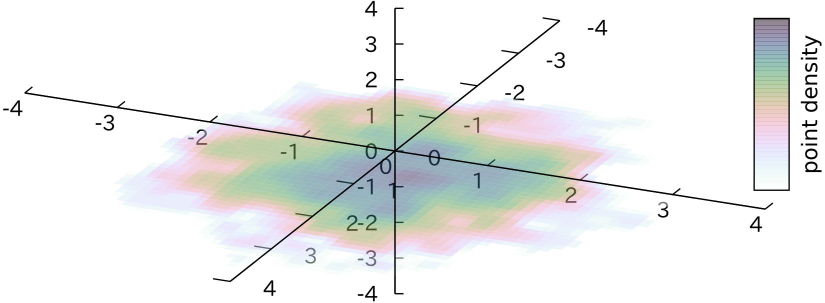
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

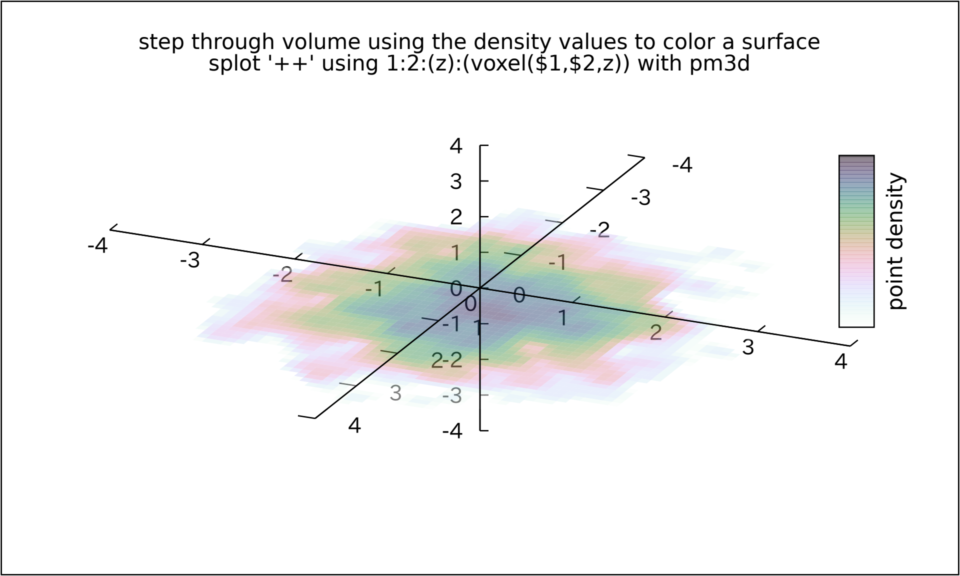



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

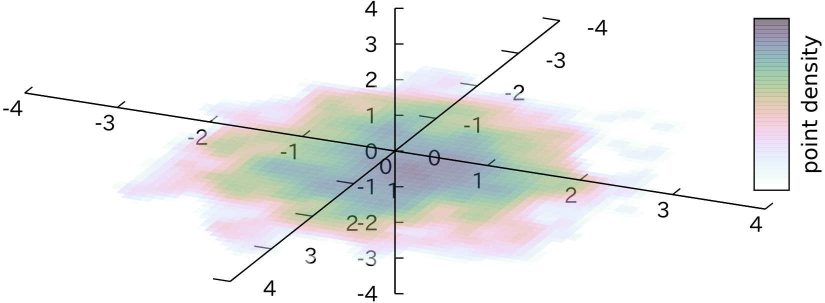
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

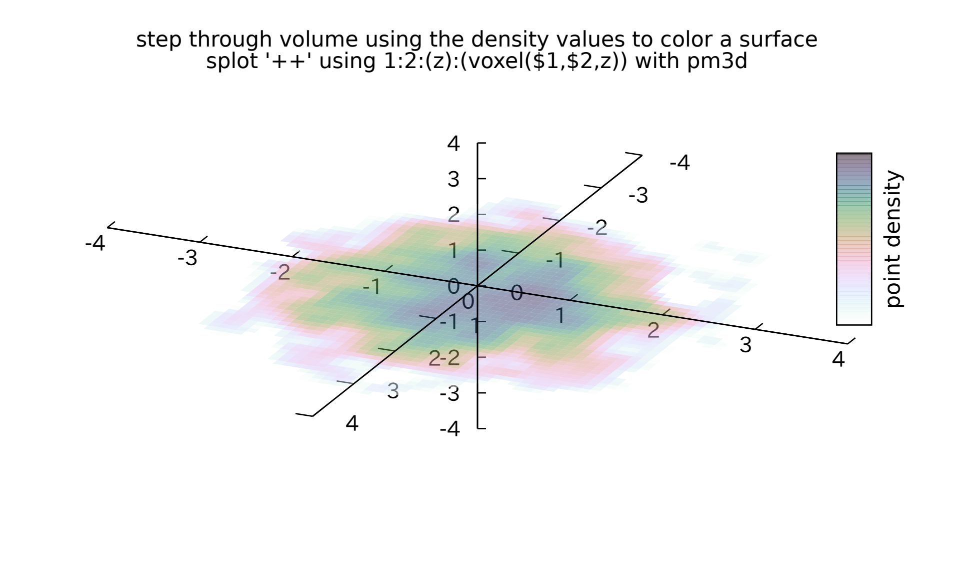


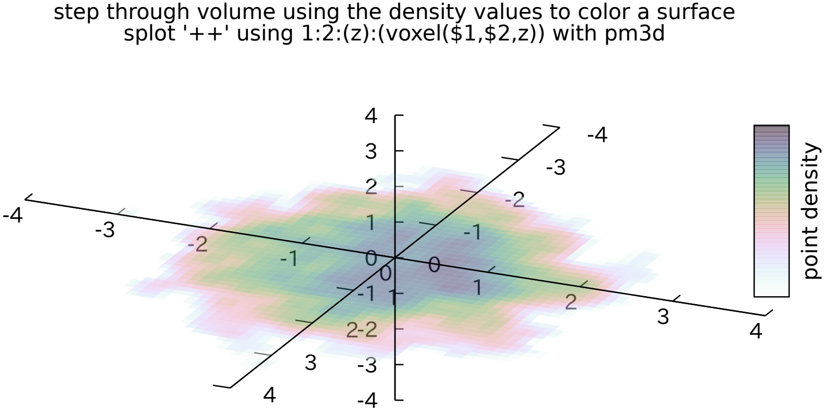


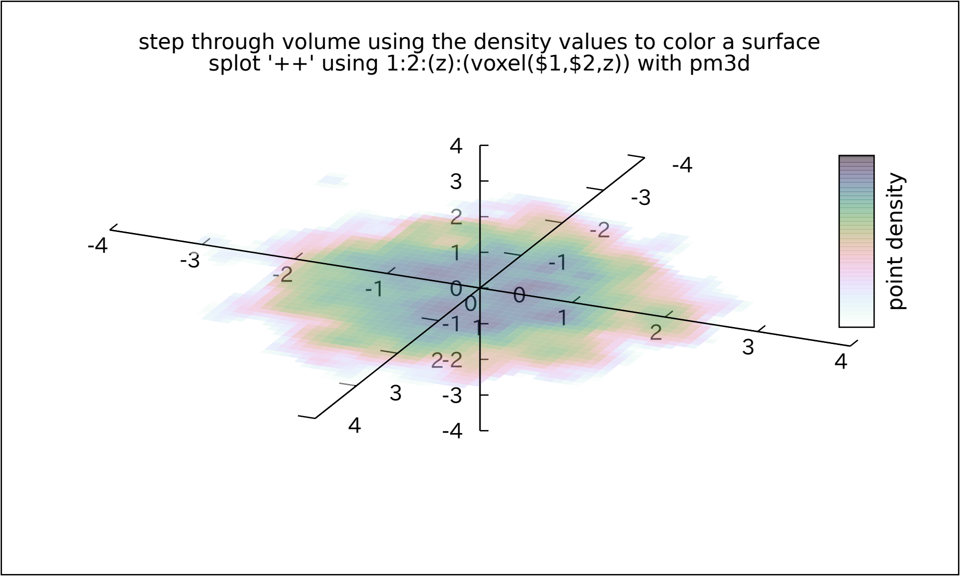
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

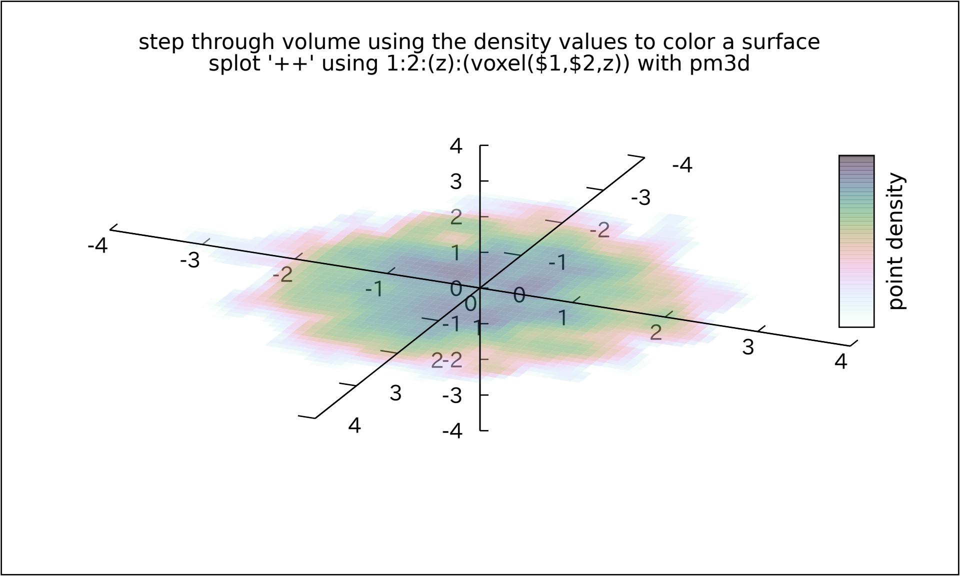
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

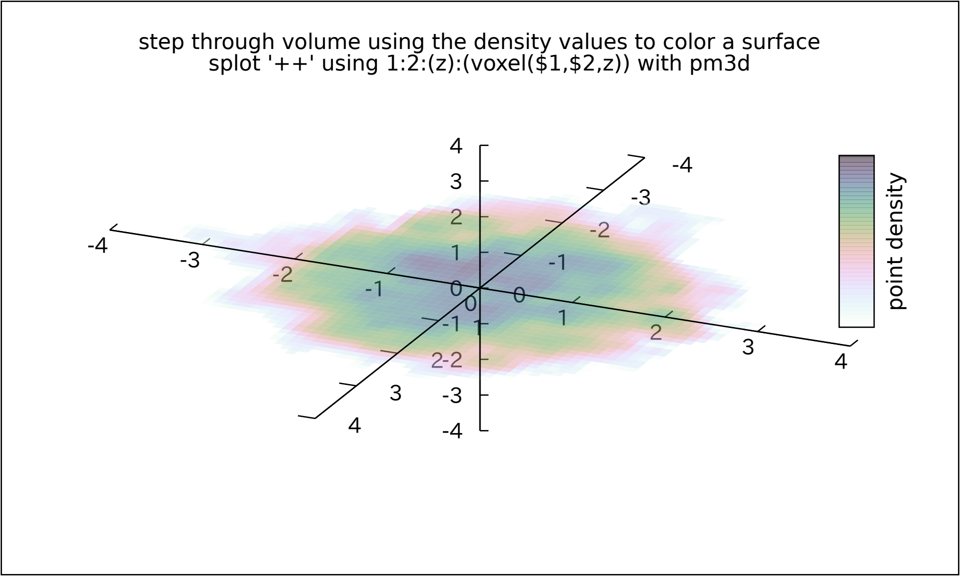



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

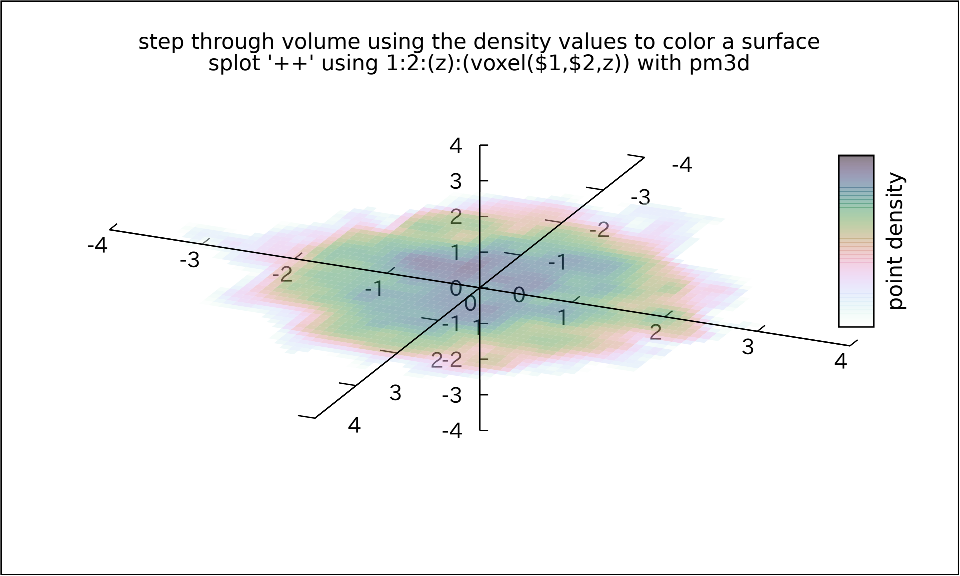

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

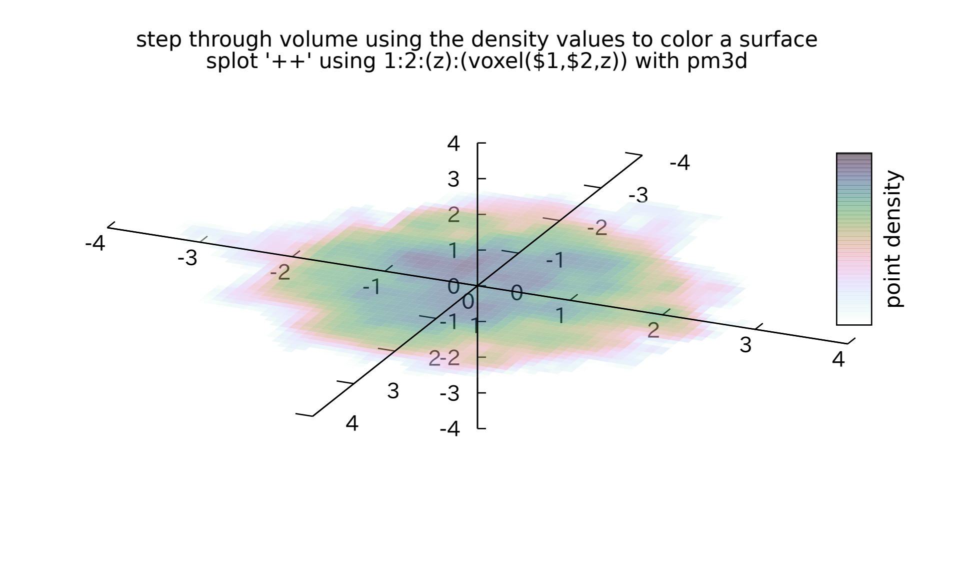

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

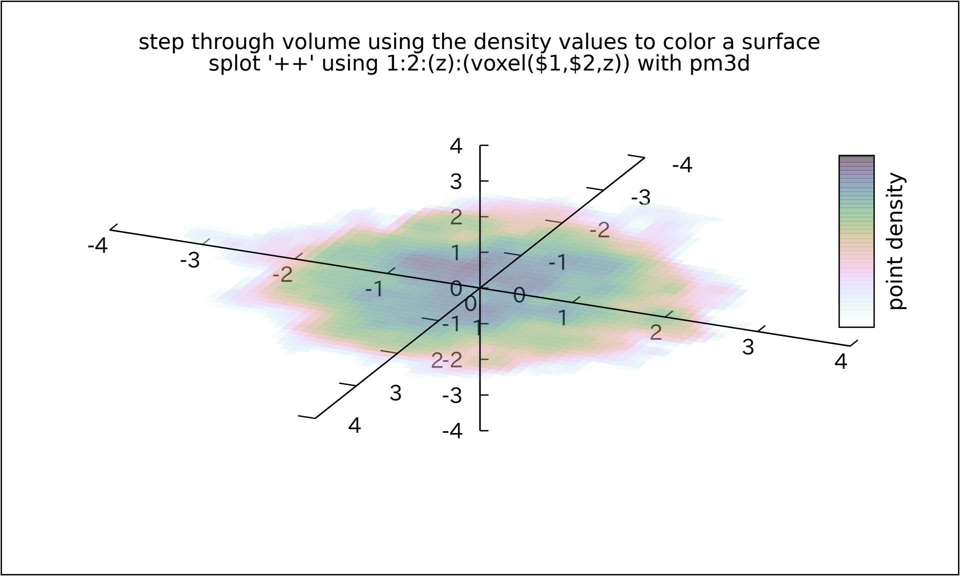

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

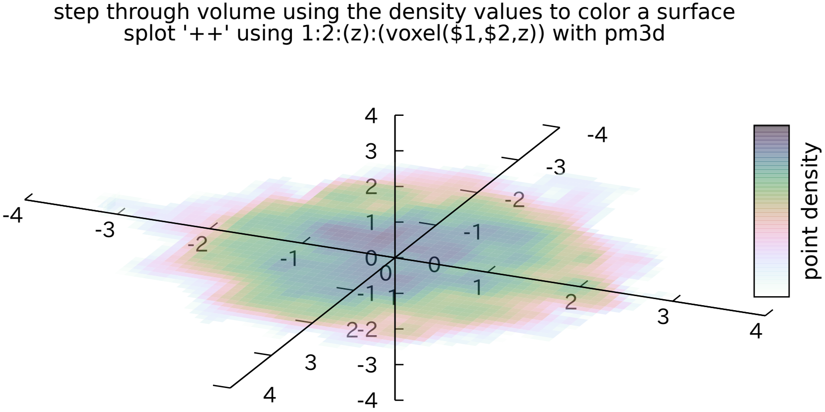


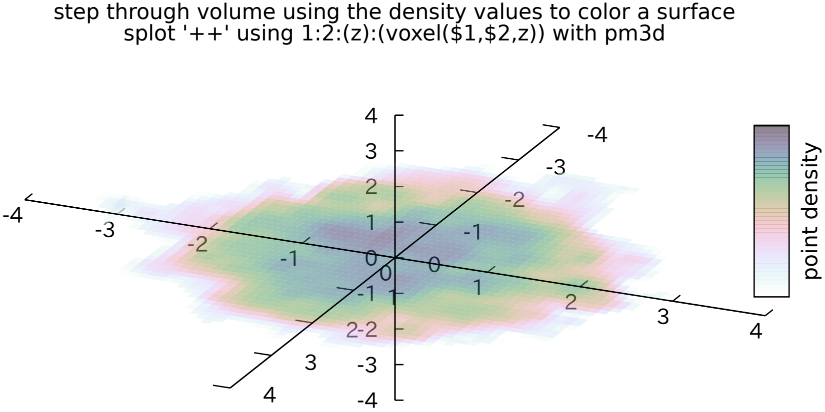
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
point density


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

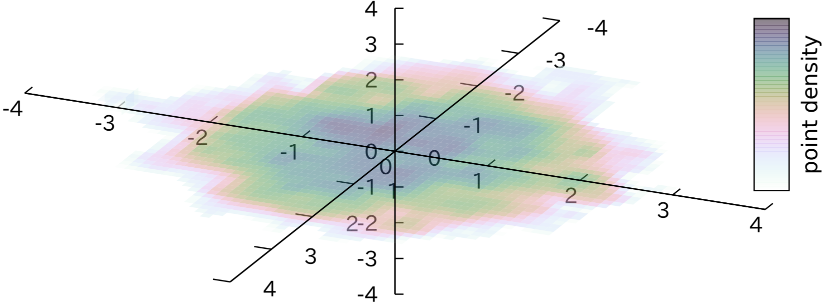

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

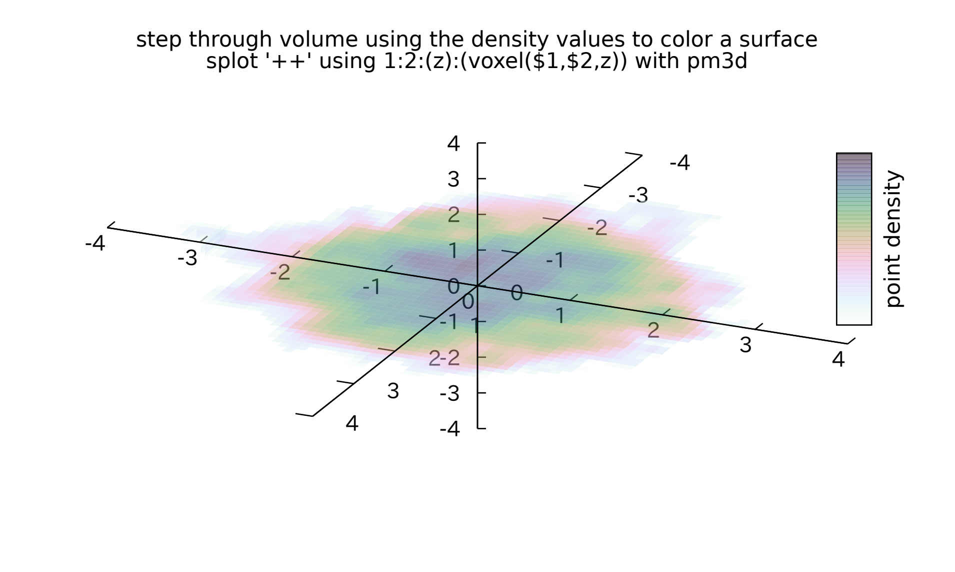

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

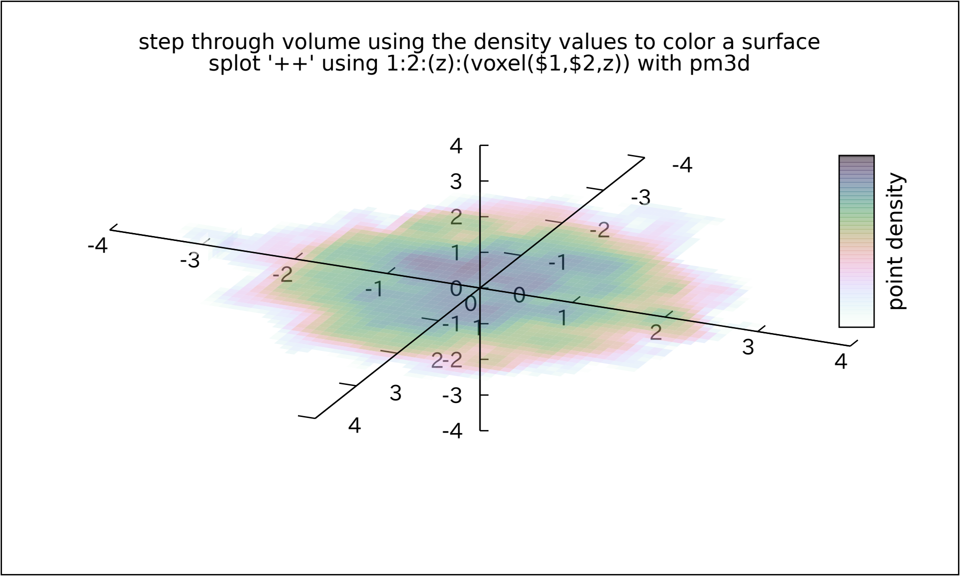

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

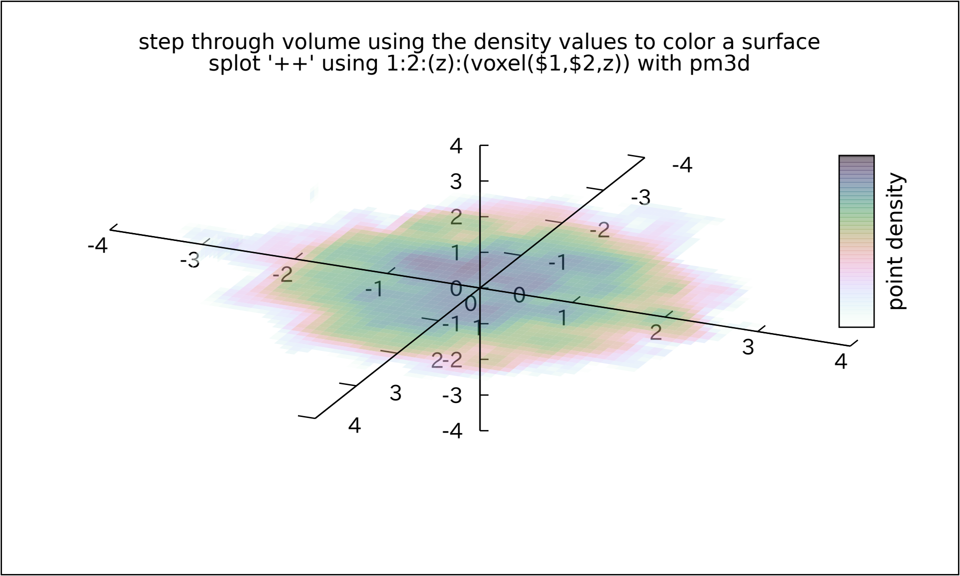

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

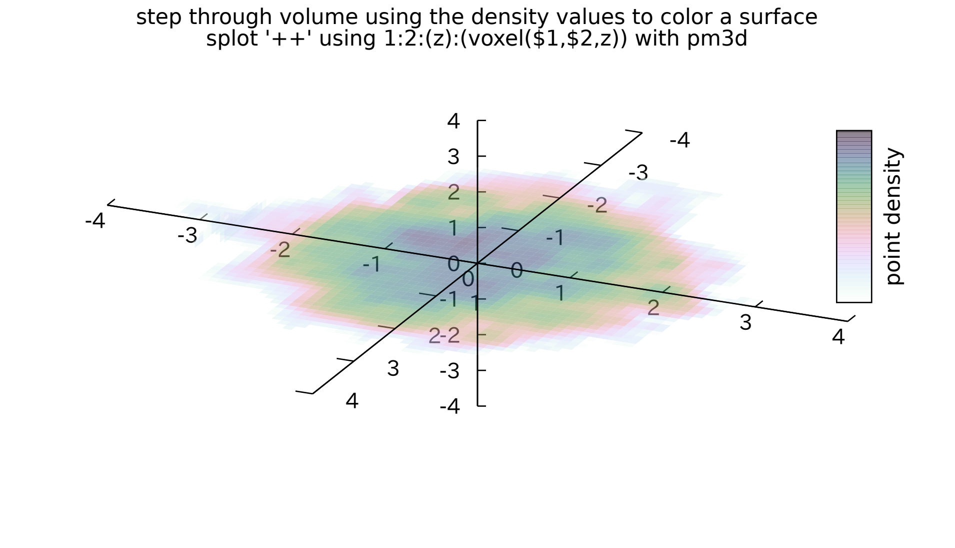
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

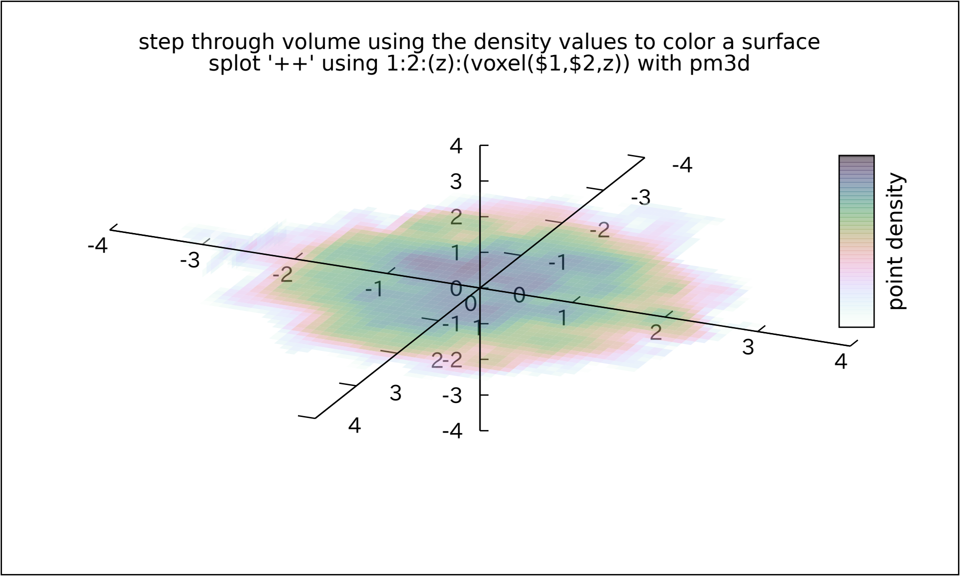



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

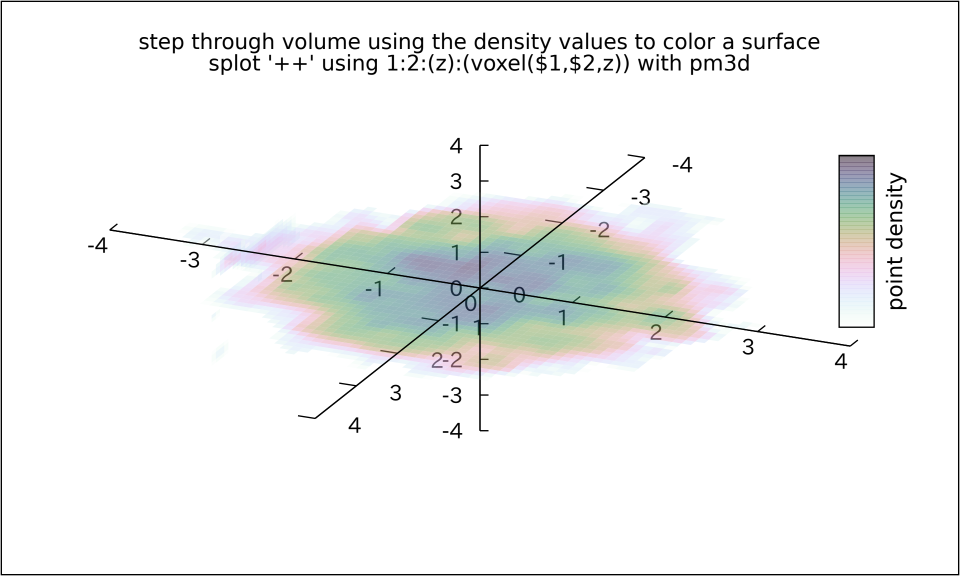

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

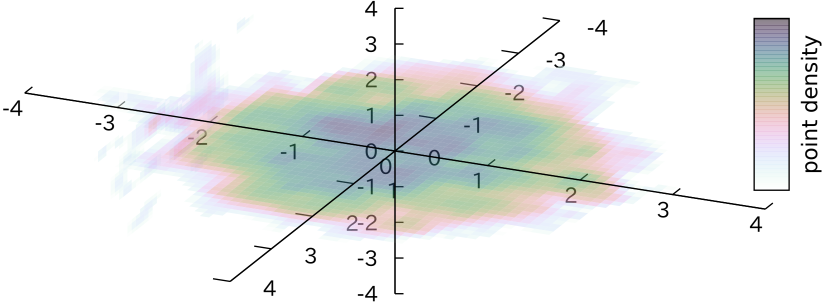

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

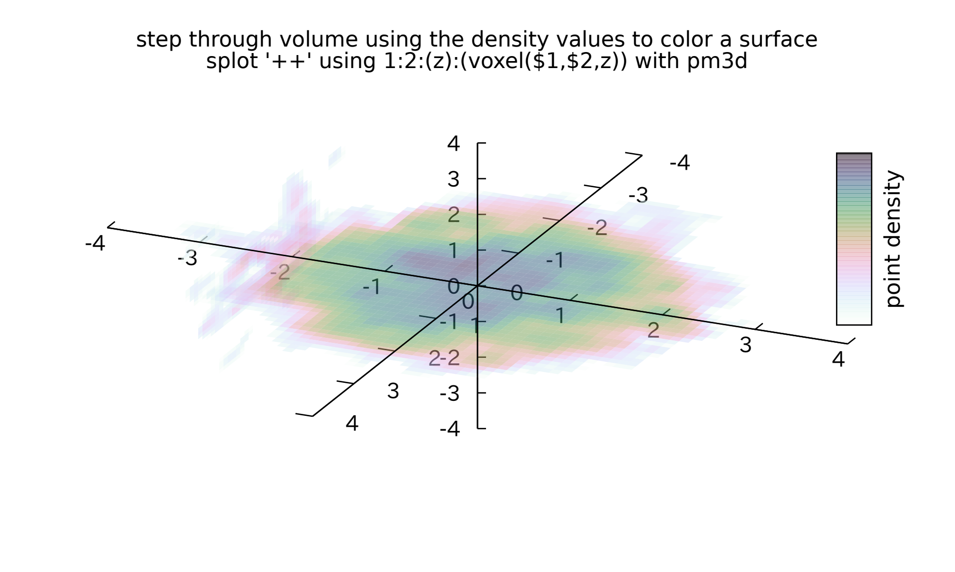


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

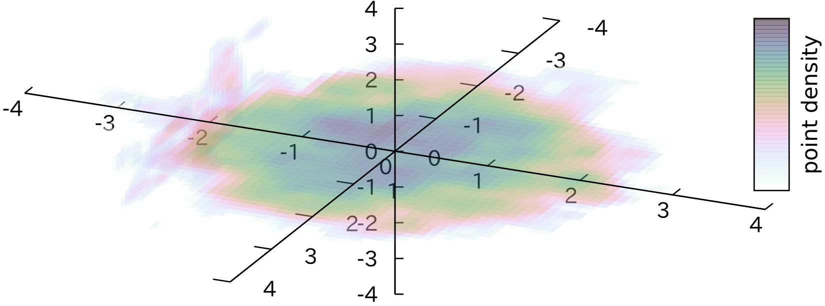
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

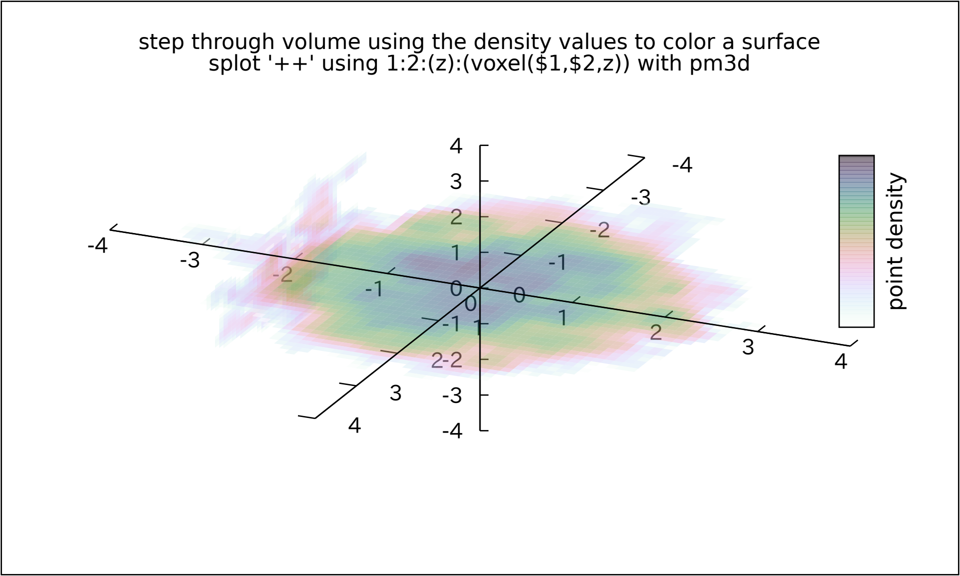




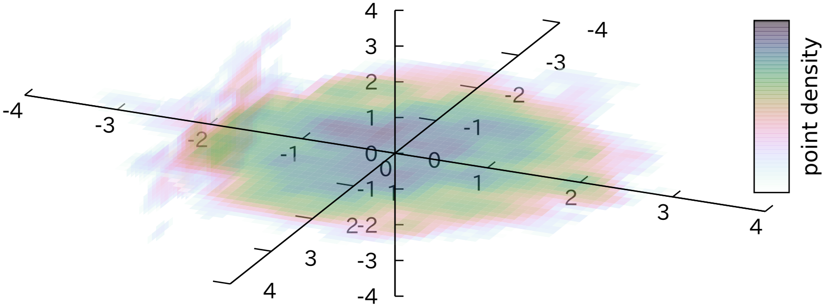
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
4
3
2
1
0
-1
-2
-3
-4

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

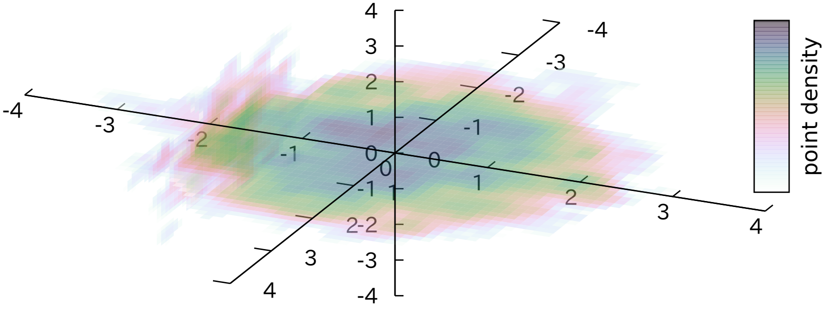



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

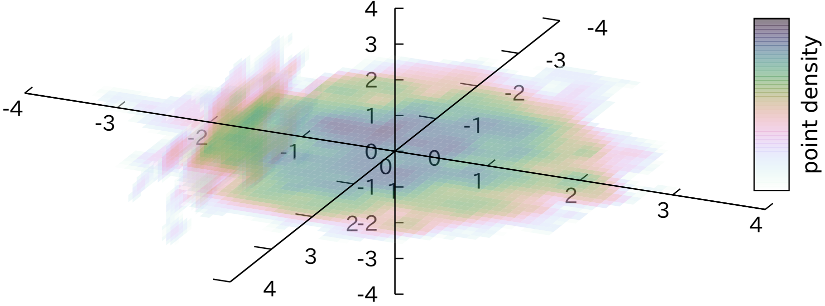
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d



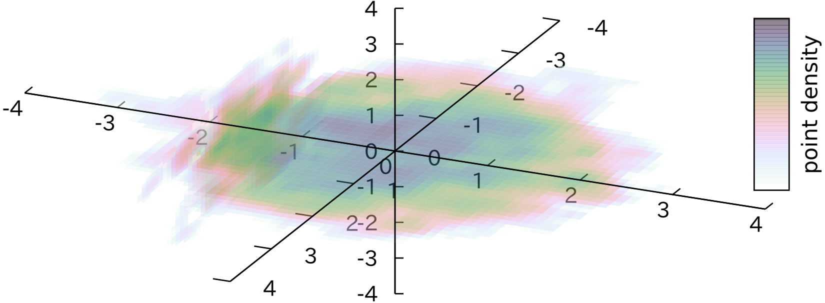
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d



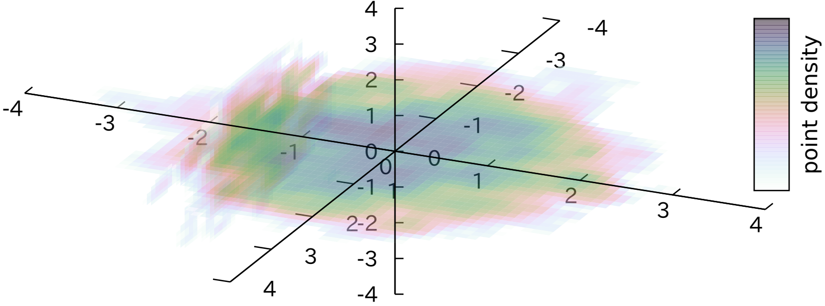
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

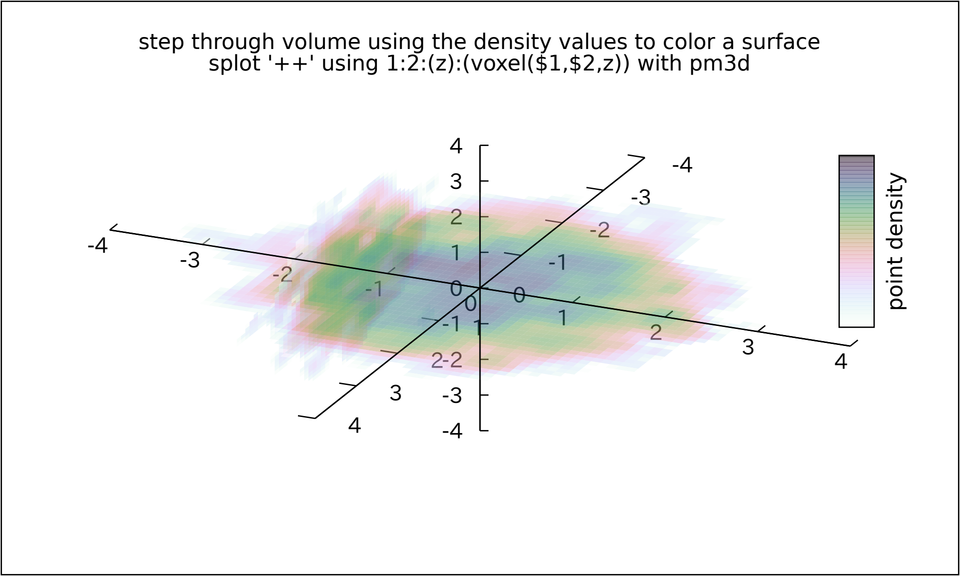


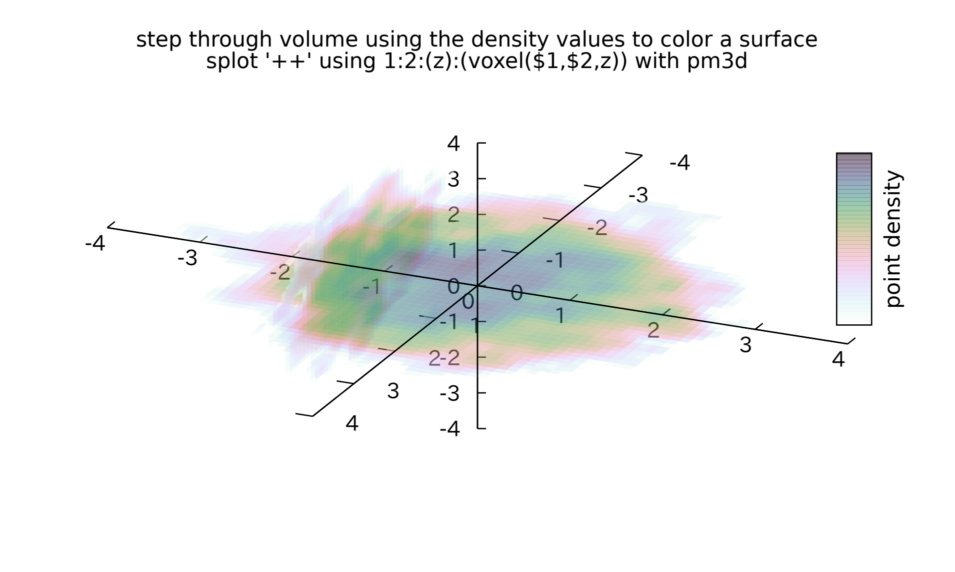
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

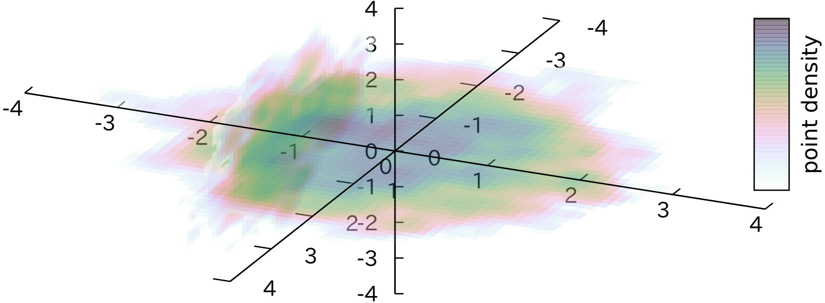


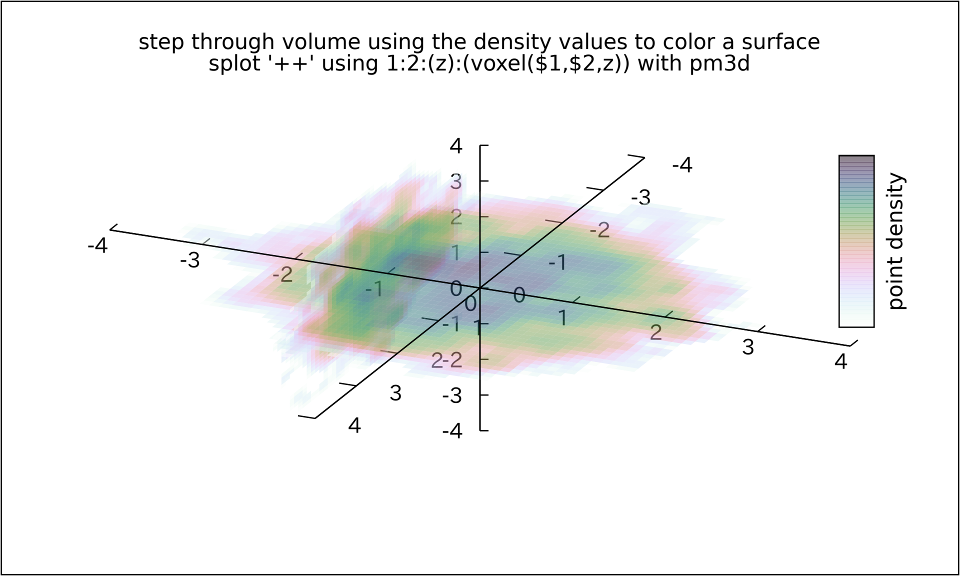

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density



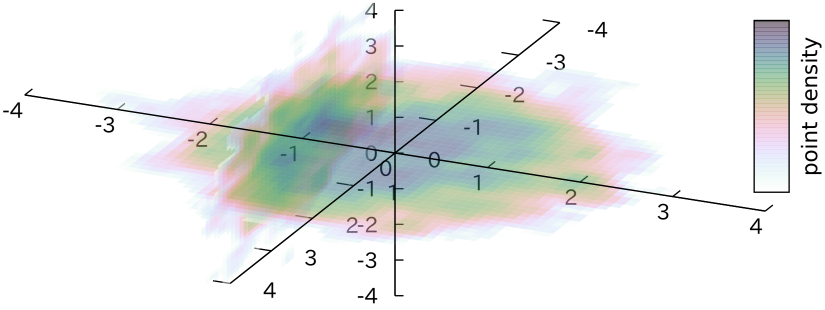
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

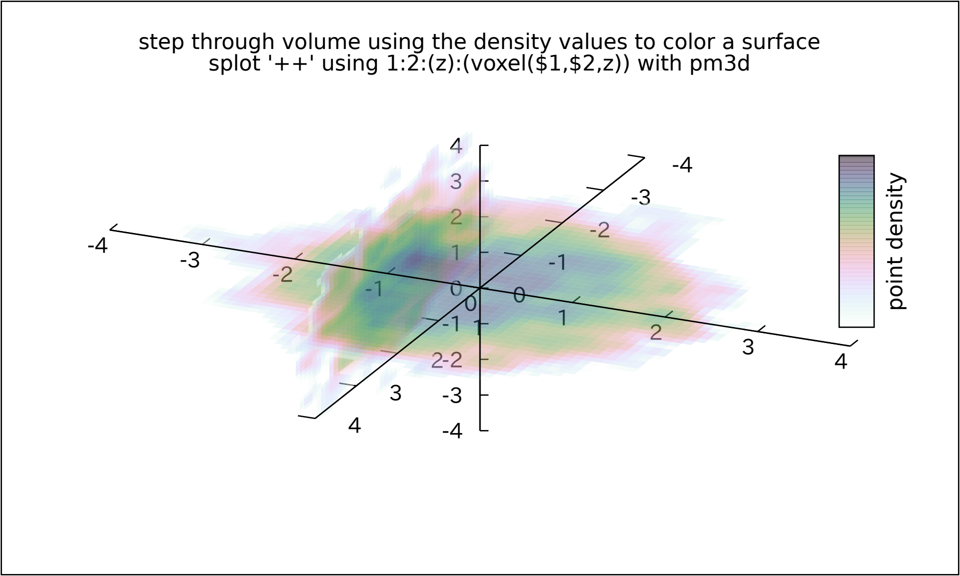
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

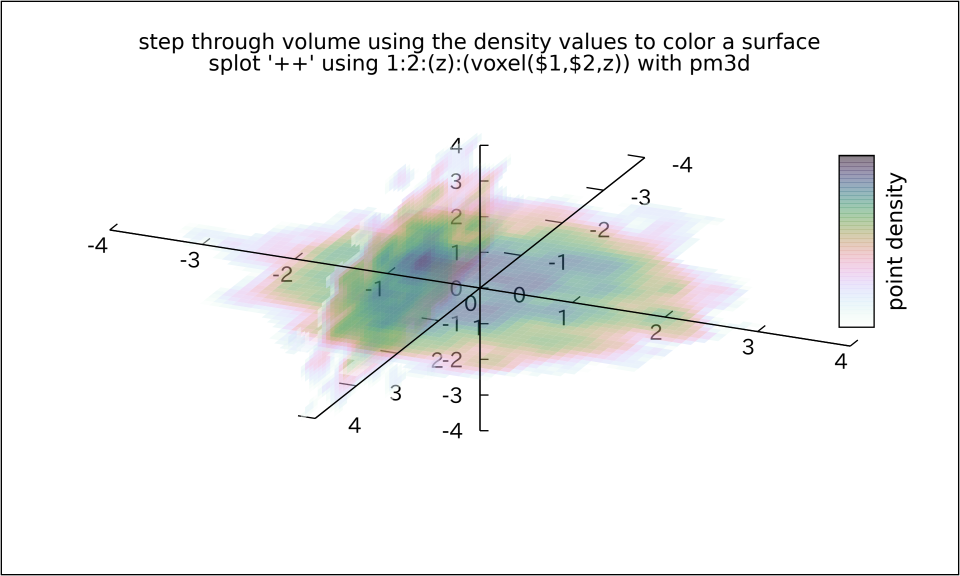


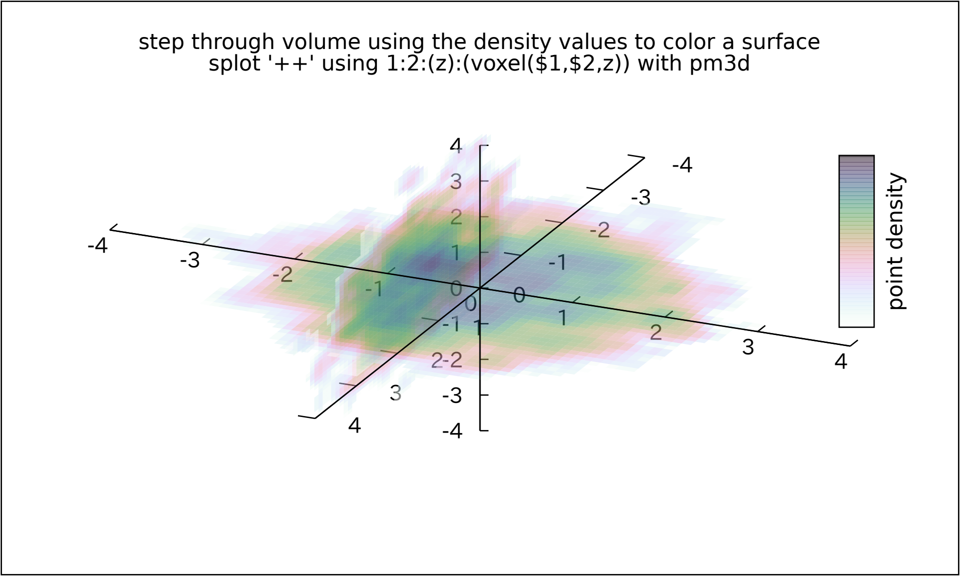

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

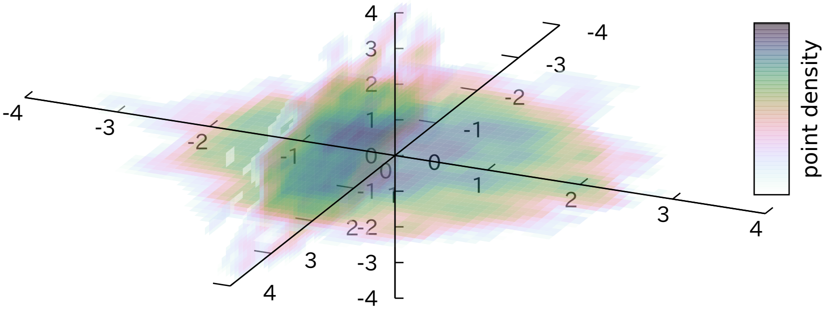


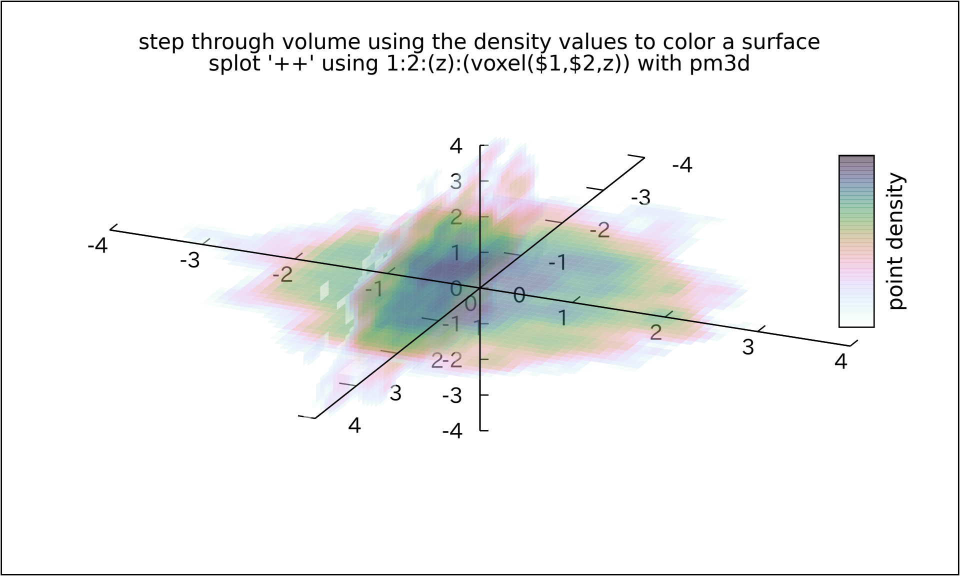

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

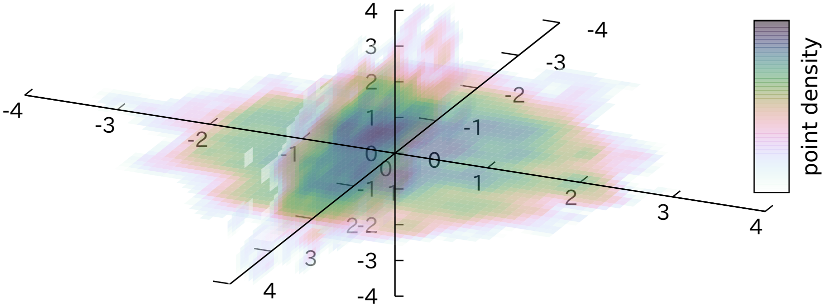

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

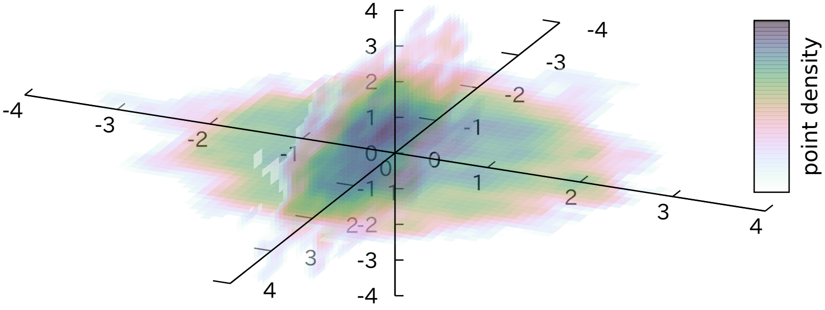



step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

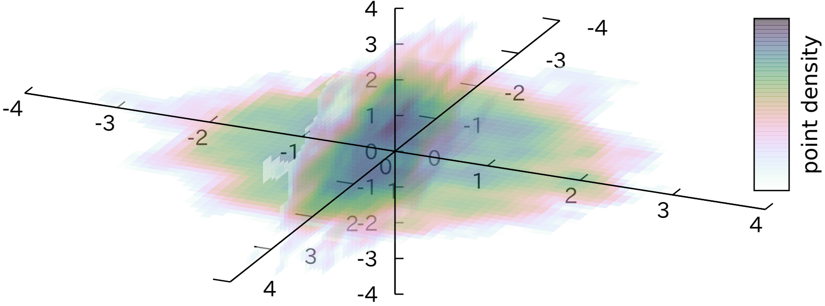
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

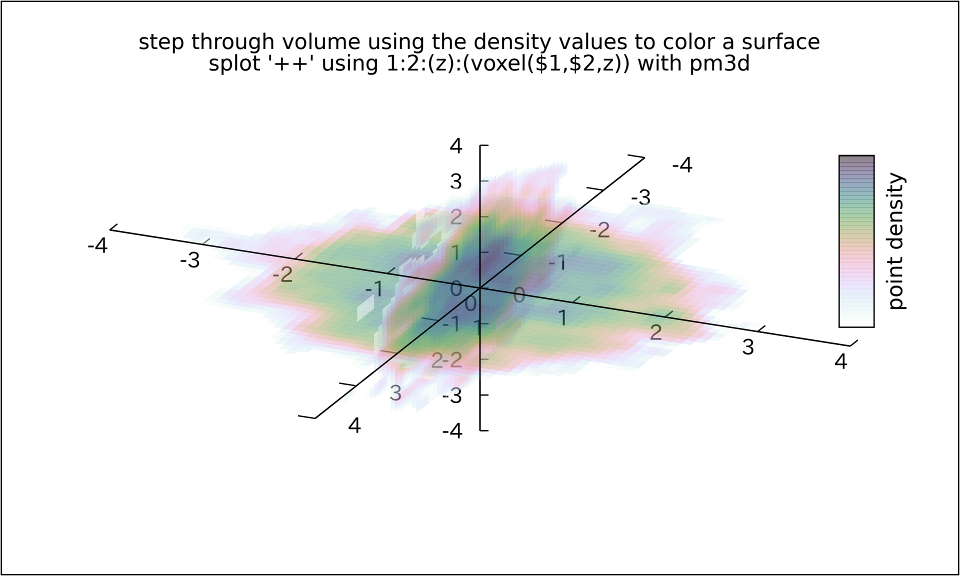


step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d

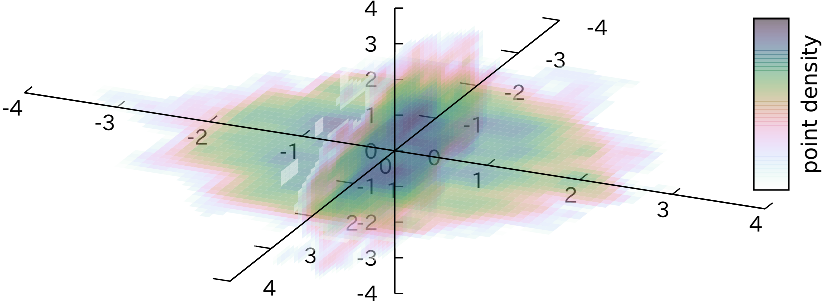


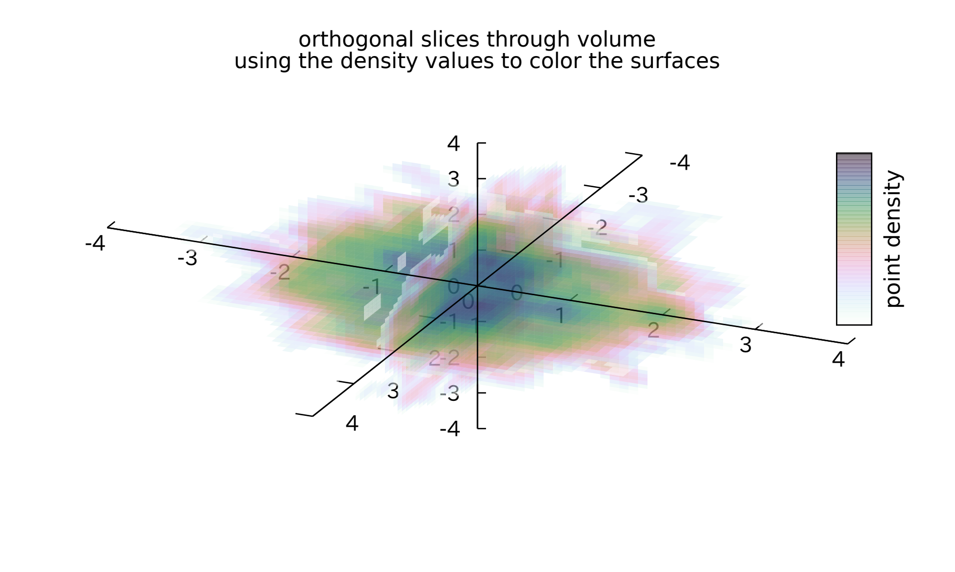
step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d




step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d
point density
-4
-3
-2
-1
0
1
2
3
4

step through volume using the density values to color a surface
splot '++' using 1:2:(z):(voxel($1,$2,z)) with pm3d




orthogonal slices through volume
using the density values to color the surfaces
point density
-4
-3
-2
-1
0
1
2
3
4