Menggambar Plot 3D dengan EMT *

Nama : Sabrina Royani Winahyu
NIM : 24030130013

Ini adalah pengenalan untuk plot 3D di Euler. Kita membutuhkan plot 3D untuk memvisualisasikan fungsi dari dua variabel.

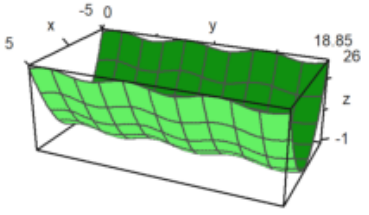
Euler menggambar fungsi-fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan bagian-bagian yang berada di latar belakang. Secara umum, Euler menggunakan proyeksi sentral. Proyeksi default berasal dari kuadran x-y positif menuju titik asal x=y=z=0, tetapi sudut=0° berarti pandangan mengarah ke arah sumbu y. Sudut pandang dan ketinggian dapat diubah.

Euler dapat memplot:

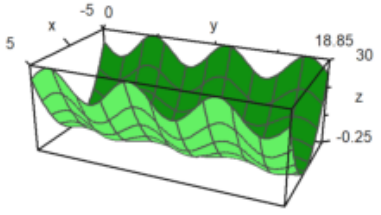
- permukaan dengan pencahayaan (shading) dan garis kontur atau rentang kontur,
- kumpulan titik (clouds of points),
- kurva parametrik,
- permukaan implisit.

Plot 3D dari suatu fungsi menggunakan ‘plot3d‘. Cara termudah adalah memplot sebuah ekspresi dalam x dan y. Parameter ‘r‘ mengatur jangkauan plot di sekitar titik (0,0).

```
>aspect(1.5); plot3d("x^2+sin(y)",-5,5,0,6*pi):
```




```
>plot3d("x^2+x*sin(y)",-5,5,0,6*pi):
```



Silakan lakukan modifikasi agar gambar "talang bergelombang" tersebut tidak lurus melainkan melengkung/melingkar, baik melingkar secara mendatar maupun melingkar turun/naik (seperti papan peluncur pada kolam renang. Temukan rumusnya.

Fungsi dari Dua Variabel *

Untuk grafik dari sebuah fungsi, gunakan:

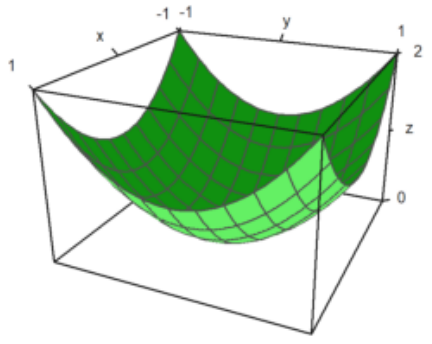
- ekspresi sederhana dalam x dan y,
- nama dari fungsi dengan dua variabel,
- atau matriks data.

Default-nya adalah kisi kawat (wire grid) yang diisi warna dengan warna berbeda di kedua sisinya. Perlu dicatat bahwa jumlah interval kisi secara default adalah 10, tetapi plot menggunakan jumlah default 40x40 persegi panjang untuk membentuk permukaan. Nilai ini bisa diubah.

- ‘n=40‘, ‘n=[40,40]‘: jumlah garis kisi (grid lines) di setiap arah
- ‘grid=10‘, ‘grid=[10,10]‘: jumlah garis kisi dalam setiap arah.

Kita menggunakan nilai default ‘n=40‘ dan ‘grid=10‘.

```
>plot3d("x^2+y^2"):
```



Interaksi Pengguna dimungkinkan dengan parameter ‘>user‘. Pengguna dapat menekan tombol-tombol berikut:

- left, right, up, down: mengubah sudut pandang
- + , -: memperbesar atau memperkecil (zoom in/out)
- a: menghasilkan anaglif (lihat penjelasan di bawah)
- l: mengaktifkan/menonaktifkan perputaran sumber cahaya (lihat penjelasan di bawah)
- spasi: mengatur ulang ke tampilan default
- enter/return: mengakhiri interaksi

```
>plot3d("exp(-x^2+y^2)",>user, ...
>  title="Turn with the vector keys (press return to finish)"):
```

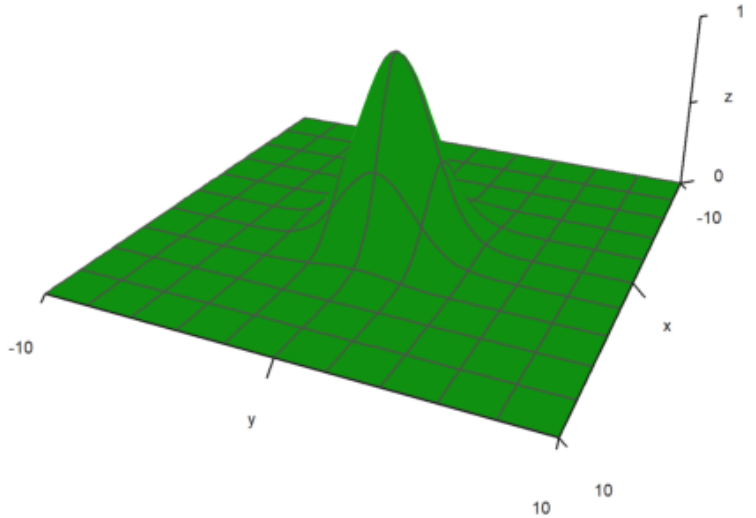
Rentang plot untuk fungsi dapat ditentukan dengan:

- a, b: rentang sumbu-x
- c, d: rentang sumbu-y
- r: area simetris berbentuk persegi di sekitar titik (0,0)
- n: jumlah subinterval untuk plot

Terdapat beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik:

- fscale: menskalakan terhadap nilai fungsi (default adalah ‘<fscale‘)
- scale: angka atau vektor 1x2 untuk skala arah x dan y
- frame: jenis bingkai (default adalah 1)

```
>plot3d("exp(-(x^2+y^2)/5)",r=10,n=80,fscale=4,scale=1.2,frame=3):
```



Tampilan dapat diubah dengan berbagai cara.

- distance: jarak pandang ke plot.
- zoom: nilai perbesaran (zoom).
- angle: sudut terhadap sumbu y negatif dalam radian.
- height: ketinggian tampilan dalam radian.

Nilai default dapat diperiksa atau diubah dengan fungsi ‘view()‘. Fungsi ini mengembalikan parameter dalam urutan seperti di atas.

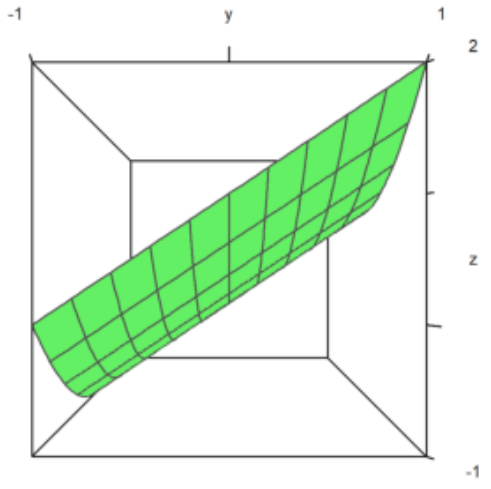
```
>view
```

```
  [5,  2.6,  2,  0.4]
```

Jarak yang lebih dekat membutuhkan zoom yang lebih kecil. Efeknya lebih mirip seperti lensa sudut lebar (wide angle).

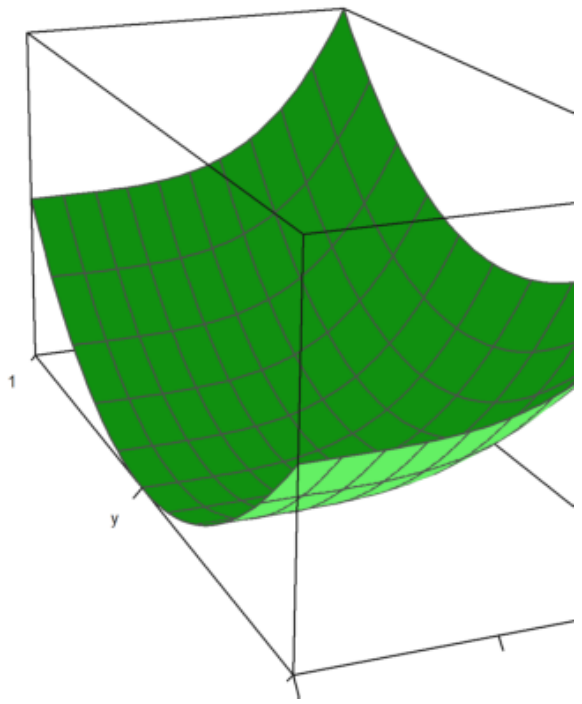
Dalam contoh berikut, ‘angle=0‘ dan ‘height=0‘ berarti melihat dari arah sumbu y negatif. Label sumbu untuk y disembunyikan dalam kasus ini.

```
>plot3d("x^2+y",distance=3,zoom=1,angle=pi/2,height=0):
```



Plot selalu mengarah ke pusat kubus plot. Anda dapat memindahkan pusat tersebut dengan parameter ‘center‘.

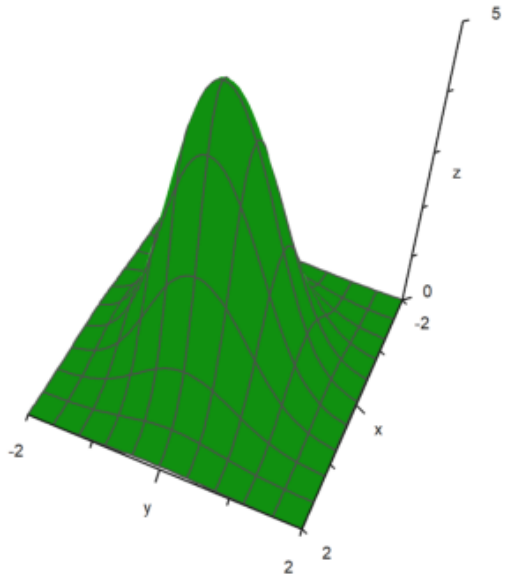
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>  center=[0.4,0,0],zoom=5):
```



Plot diskalakan agar muat dalam kubus satuan untuk tampilan. Jadi, tidak perlu mengubah jarak atau zoom tergantung pada ukuran plot. Namun, label tetap mengacu pada ukuran sebenarnya.

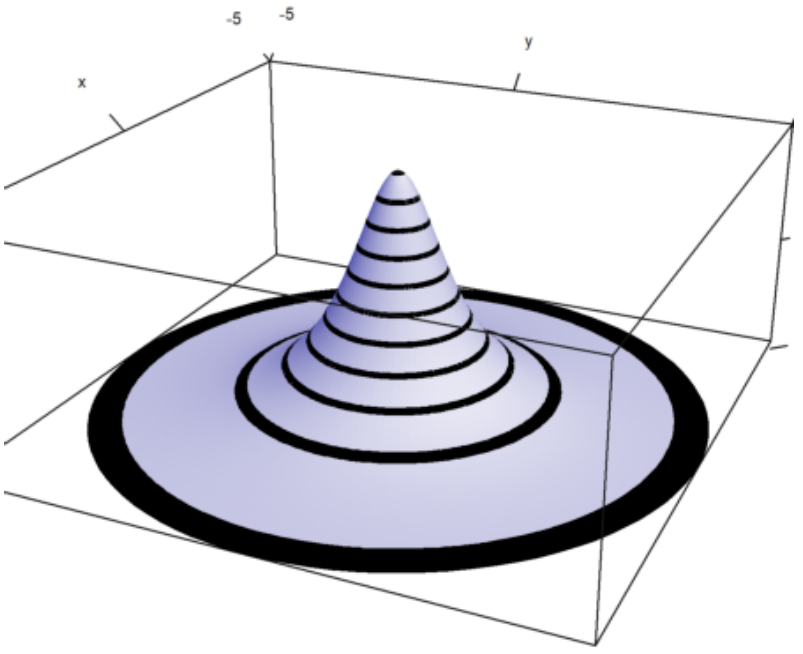
Jika Anda mematikan fitur ini dengan ‘scale=false‘, Anda perlu memastikan bahwa plot tetap muat di dalam jendela tampilan dengan cara mengubah jarak pandang atau zoom, serta memindahkan pusat tampilan.

```
>plot3d("5*exp(-x^2-y^2)",r=2,<fscale,<scale,distance=13,height=50°, ...
>  center=[0,0,-2],frame=3):
```

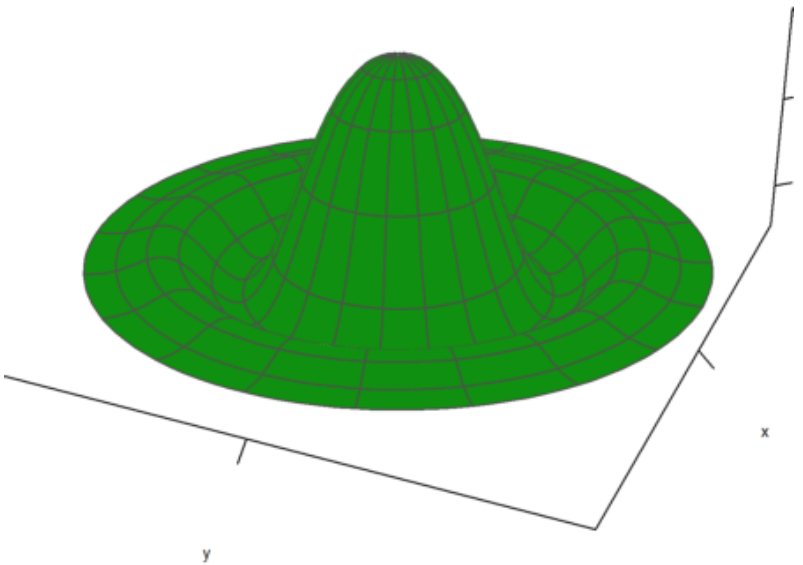


Plot polar juga tersedia. Parameter ‘polar=true‘ akan menggambar plot polar. Fungsi tetap harus berupa fungsi dari x dan y. Parameter ‘fscale‘ digunakan untuk menskalakan fungsi dengan skala tersendiri. Jika tidak, fungsi akan diskalakan agar muat di dalam sebuah kubus.

```
>plot3d("1/(x^2+y^2+1)",r=5,>polar, ...
>fscale=2,>hue,n=100,zoom=4,>contour,color=blue):
```



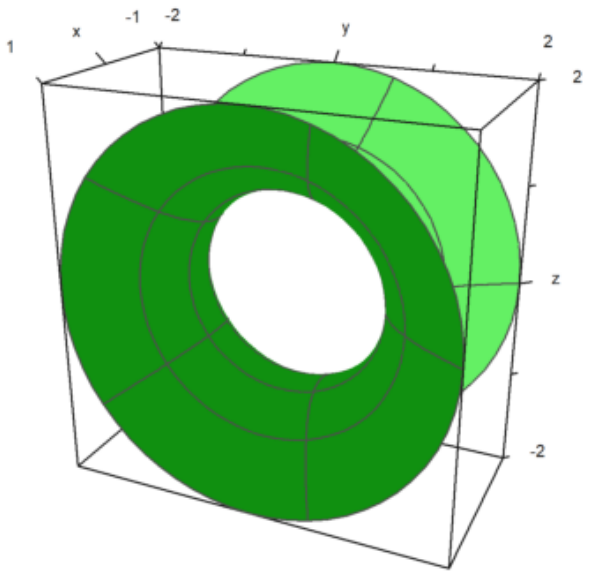
```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



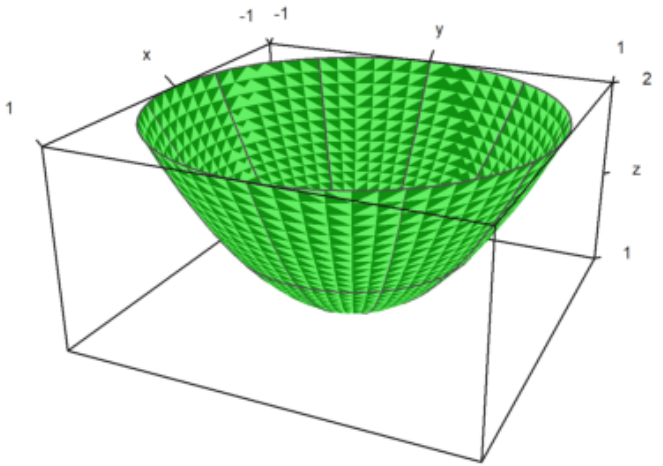
Parameter ‘rotate‘ memutar sebuah fungsi terhadap x mengelilingi sumbu tertentu.

- rotate=1: Menggunakan sumbu x
- rotate=2: Menggunakan sumbu z

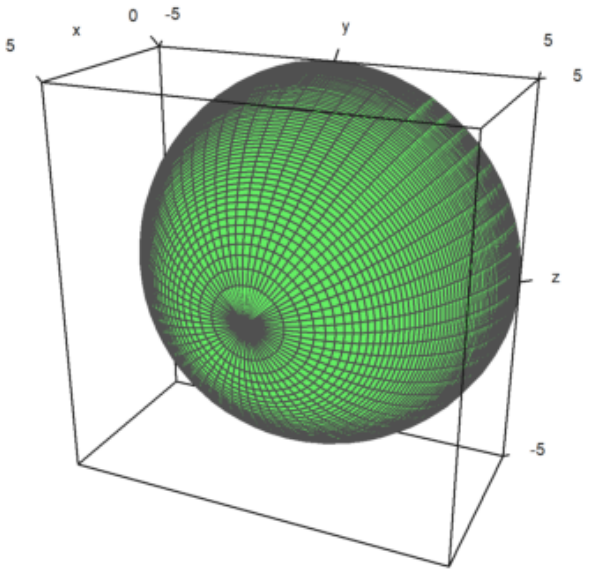
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



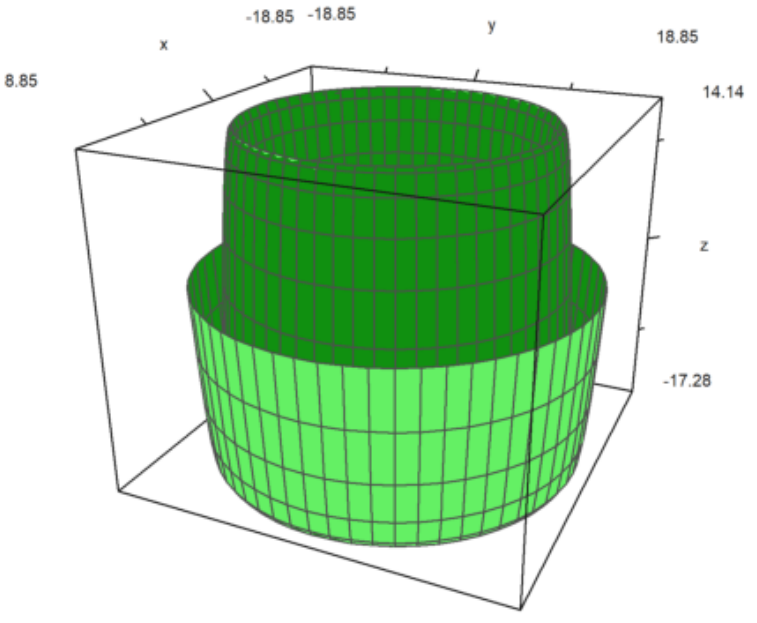
```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




```
>plot3d("sqrt(25-x^2)",a=0,b=5,rotate=1):
```

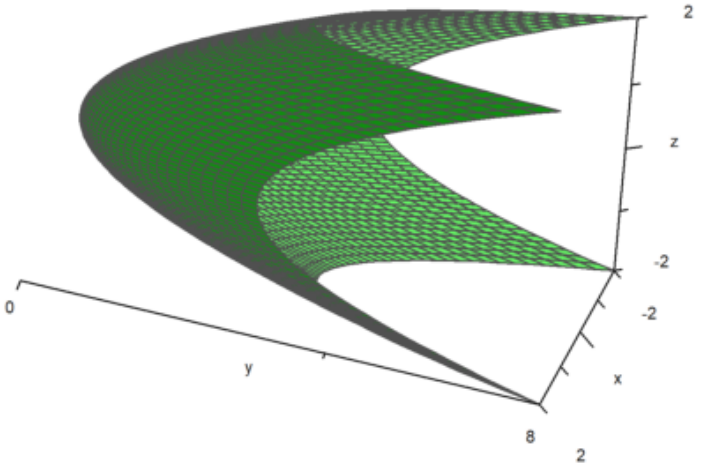



```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Berikut adalah sebuah plot dengan tiga fungsi.

```
>plot3d("x","x^2+y^2","y",r=2,zoom=3.5,frame=3):
```


2
z
-2
-2
x
8
2
0
y

Plot Kontur

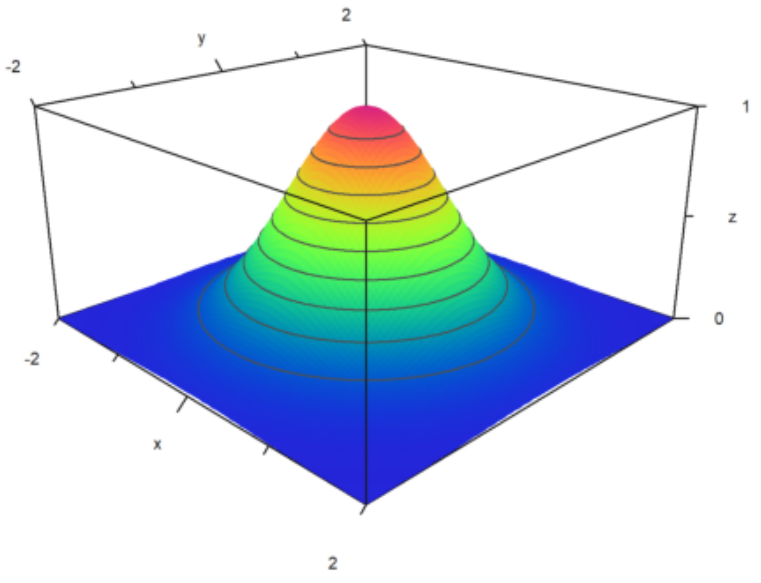
Untuk plot, Euler menambahkan garis grid. Sebagai gantinya, bisa digunakan garis level dan gradasi satu warna atau gradasi spektral berwarna. Euler dapat menggambarkan ketinggian fungsi pada plot dengan pewarnaan (shading). Dalam semua plot 3D, Euler dapat menghasilkan anaglif merah/sian.

- ‘>hue‘: Mengaktifkan shading cahaya sebagai pengganti garis kawat.
- ‘>contour‘: Menggambar garis kontur otomatis pada plot.
- ‘level=...‘ (atau ‘levels‘): Sebuah vektor nilai untuk garis kontur.

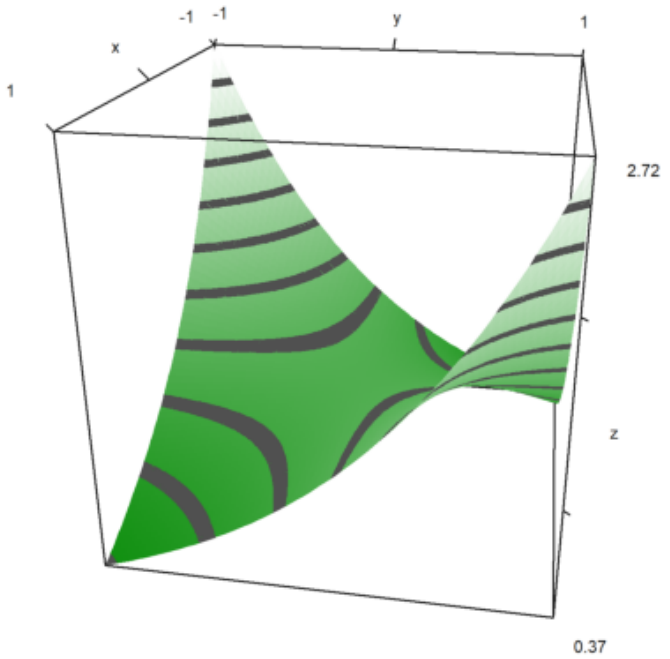
Default-nya adalah ‘level=”auto”‘, yang secara otomatis menghitung beberapa garis kontur. Seperti yang terlihat pada plot, level sebenarnya adalah rentang dari beberapa level.

Gaya default ini dapat diubah. Untuk plot kontur berikut, kita menggunakan grid yang lebih halus sebanyak 100x100 titik, menskalakan fungsi dan plotnya, serta menggunakan sudut pandang yang berbeda.

```
>plot3d("exp(-x^2-y^2)",r=2,n=100,level="thin", ...
> >contour,>spectral,fscale=1,scale=1.1,angle=45°,height=20°):
```




```
>plot3d("exp(x*y)",angle=100°,>contour,color=green):
```

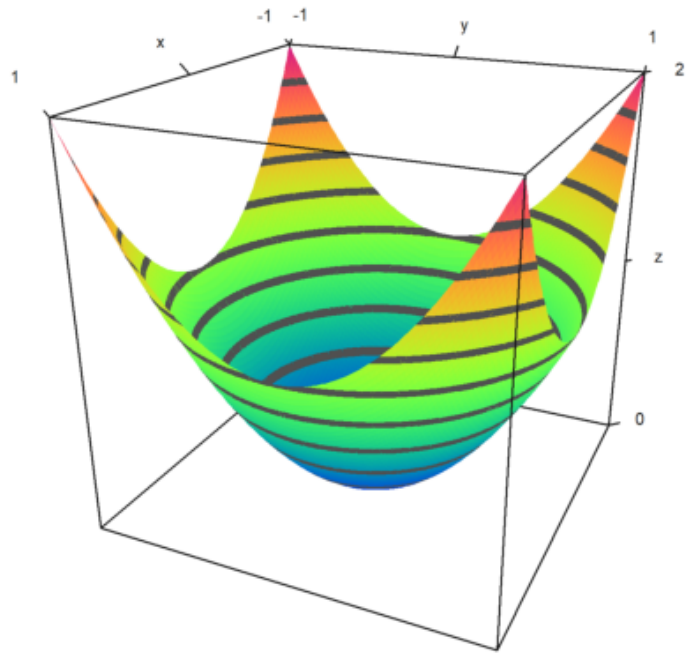


Shading (pewarnaan) default menggunakan warna abu-abu. Namun, rentang warna spektral juga tersedia.

`‘>spectral‘: Menggunakan skema spektral default`
`‘color=...‘: Menggunakan warna khusus atau skema spektral tertentu`

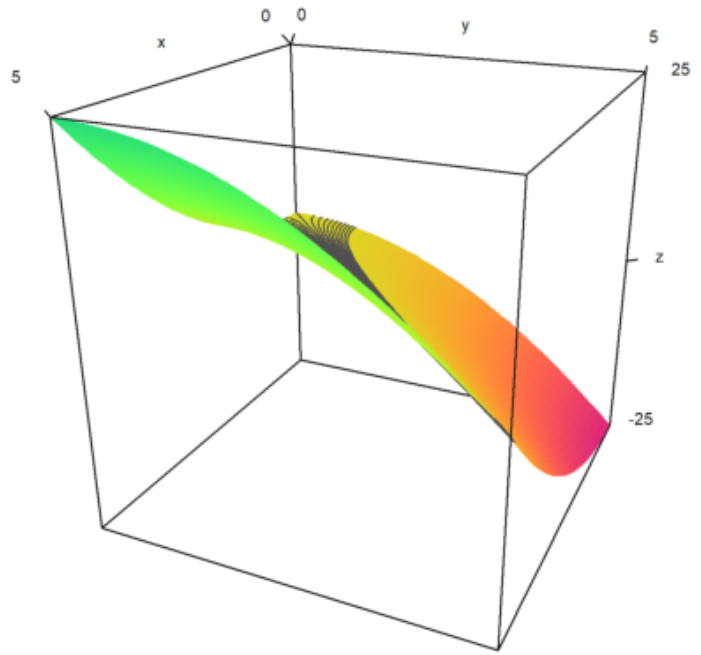
Untuk plot berikut, kita menggunakan skema spektral default dan meningkatkan jumlah titik untuk mendapatkan tampilan yang sangat halus.

```
>plot3d("x^2+y^2",>spectral,>contour,n=100):
```



Alih-alih menggunakan garis level otomatis, kita juga dapat menetapkan nilai-nilai untuk garis level tersebut. Ini akan menghasilkan garis level tipis, bukan rentang level.

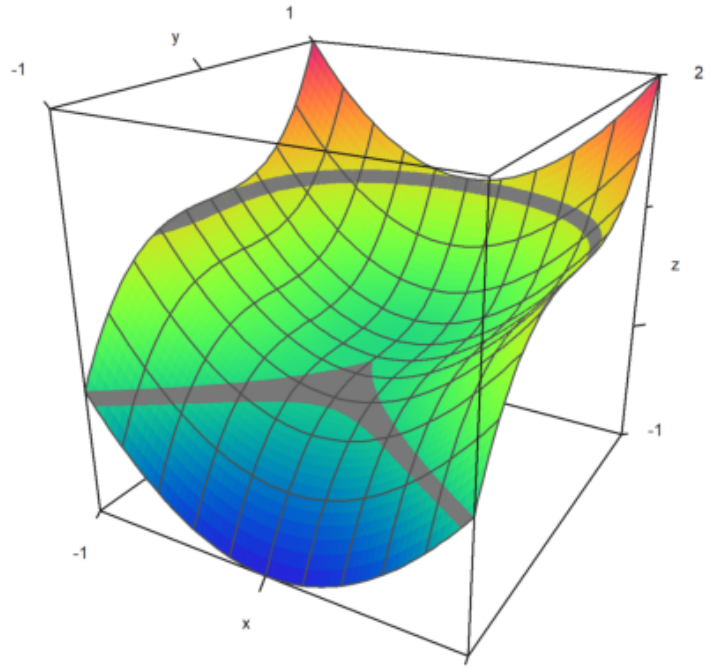
```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



Dalam plot berikut, kita menggunakan dua pita level yang sangat lebar, dari -0,1 hingga 1, dan dari 0,9 hingga 1. Ini dimasukkan sebagai matriks dengan batas level sebagai kolom.

Selain itu, kita menambahkan grid dengan 10 interval di setiap arah.

```
>plot3d("x^2+y^3",level=[-0.1,0.9;0,1], ...
>  >spectral,angle=30°,grid=10,contourcolor=gray):
```

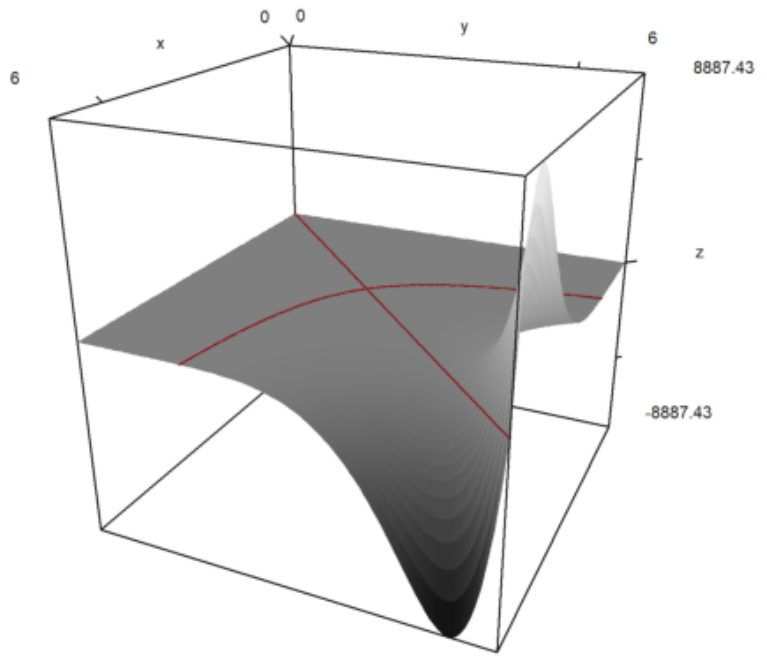


Dalam contoh berikut, kita memplot himpunan di mana

f(x,y) = xˆy - yˆx = 0

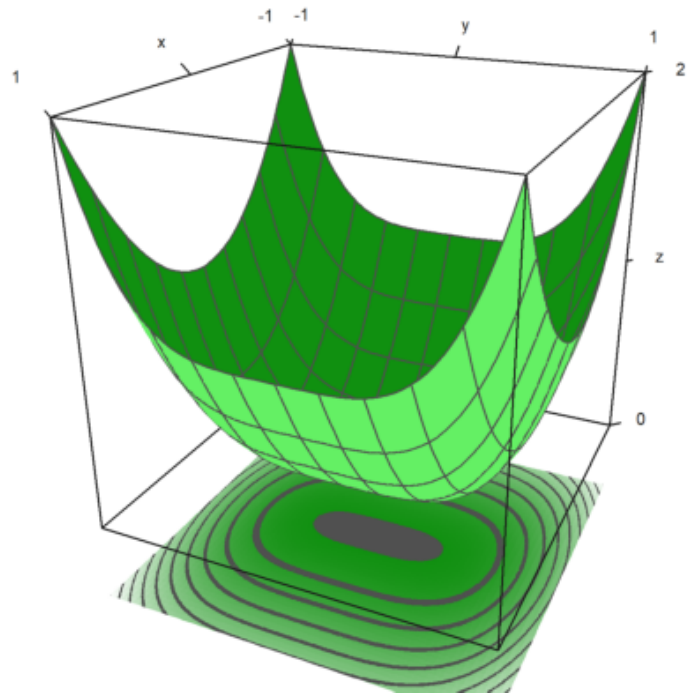
Kita menggunakan satu garis tipis untuk garis level tersebut.

```
>plot3d("xˆy-yˆx",level=0,a=0,b=6,c=0,d=6,contourcolor=red,n=100):
```



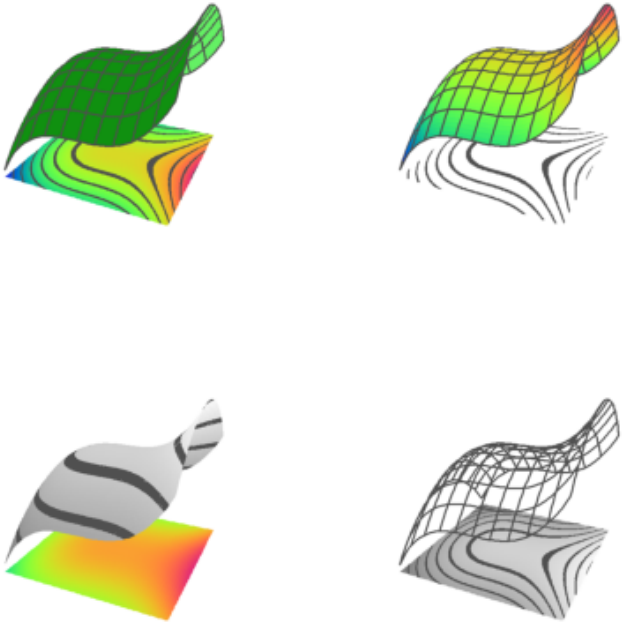
Dimungkinkan untuk menampilkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

```
>plot3d("x^2+y^4",>cp,cpcolor=green,cpdelta=0.2):
```



Berikut beberapa gaya lainnya. Kami selalu mematikan bingkai (frame), dan menggunakan berbagai skema warna untuk plot dan grid.

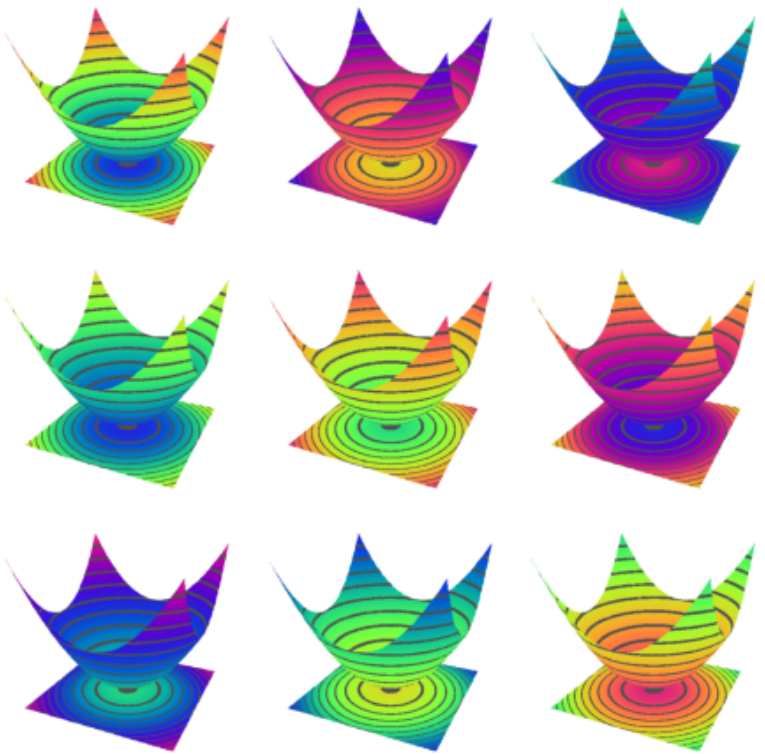
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1);  ...
>  plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2);  ...
>  plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3);  ...
>  plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4);  ...
>  plot3d(expr,<frame,>hue,grid=10,>transparent,>cp,cpcolor=gray); ...
>figure(0):
```



Terdapat beberapa skema spektral lainnya, yang diberi nomor dari 1 hingga 9. Namun, Anda juga dapat menggunakan ‘color=value‘, di mana nilai tersebut bisa berupa:

- spectral: untuk rentang warna dari biru ke merah
- white: untuk rentang warna yang lebih lembut
- yellowblue, purplegreen, blueyellow, greenred
- blueyellow, greenpurple, yellowblue, redgreen

```
>figure(3,3); ...
>for i=1:9;  ...
>  figure(i); plot3d("x^2+y^2",spectral=i,>contour,>cp,<frame,zoom=4);  ...
>end; ...
>figure(0):
```



Sumber cahaya dapat diubah dengan tombol l dan tombol panah (cursor keys) saat interaksi pengguna. Sumber cahaya juga dapat diatur menggunakan parameter.

- light: arah sumber cahaya
- amb: cahaya ambient dengan nilai antara 0 dan 1

Perlu diperhatikan bahwa program ini tidak membedakan sisi-sisi plot. Tidak ada bayangan yang dihasilkan. Untuk itu, Anda memerlukan Povray.

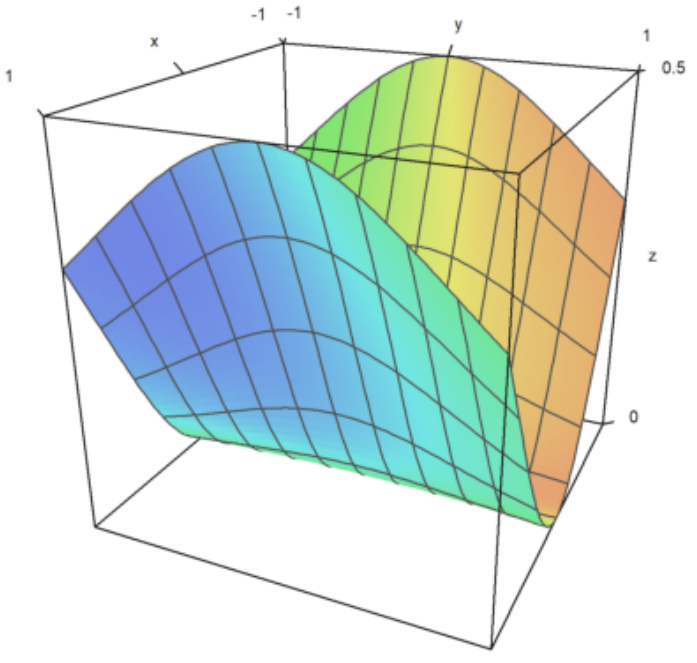
```
>plot3d("-x^2-y^2", ...
>  hue=true,light=[0,1,1],amb=0,user=true, ...
>  title="Press l and cursor keys (return to exit)"):
```

Parameter color mengubah warna permukaan. Warna garis level juga dapat diubah.

```
>plot3d("-x^2-y^2",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>  zoom=3,contourcolor=red,level=-2:0.1:1,dl=0.01):
```

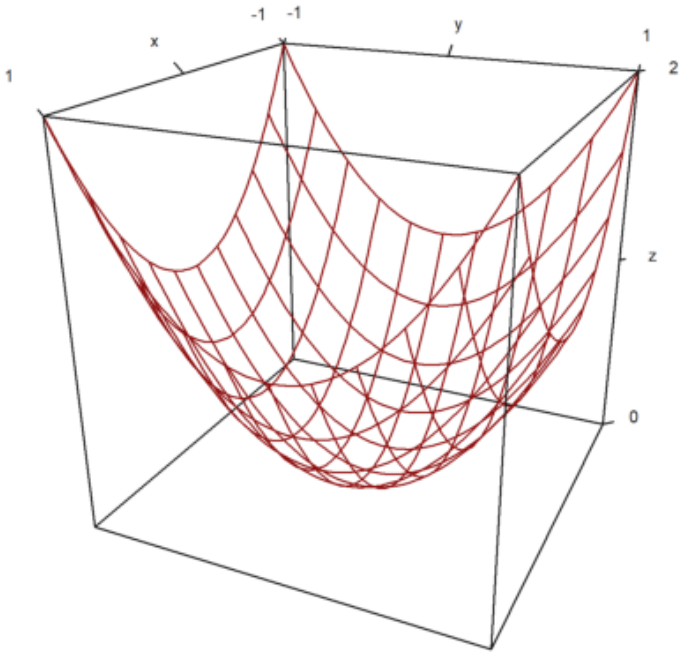
Warna 0 memberikan efek pelangi khusus.

```
>plot3d("x^2/(x^2+y^2+1)",color=0,hue=true,grid=10):
```



The surface can also be transparent.

```
>plot3d("x^2+y^2",>transparent,grid=10,wirecolor=red):
```


-1
-1
x
y
1
1
2
z
0

Plot Implisit

Terdapat juga plot implisit dalam tiga dimensi. Euler menghasilkan potongan melalui objek-objek tersebut. Fitur dari ‘plot3d‘ mencakup plot implisit. Plot ini menunjukkan himpunan nol dari sebuah fungsi dengan tiga variabel.

Penyelesaian dari

f(x,y,z) = 0

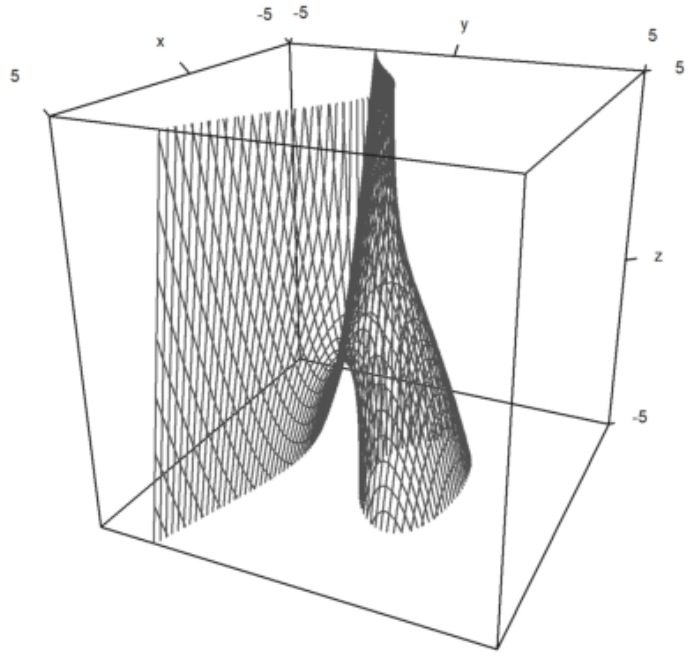
dapat divisualisasikan dalam potongan yang sejajar dengan bidang x-y, x-z, dan y-z.

- ‘implicit=1‘: potongan sejajar bidang y-z
- ‘implicit=2‘: potongan sejajar bidang x-z
- ‘implicit=4‘: potongan sejajar bidang x-y

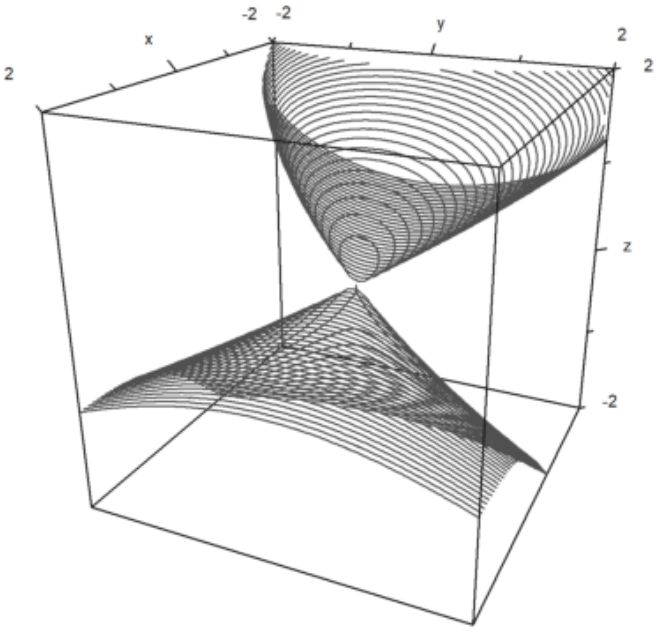
Nilai-nilai ini dapat dijumlahkan sesuai kebutuhan. Dalam contoh, kita memplot

M = { (x,y,z) : x^2 + y^3 + zy = 1 }

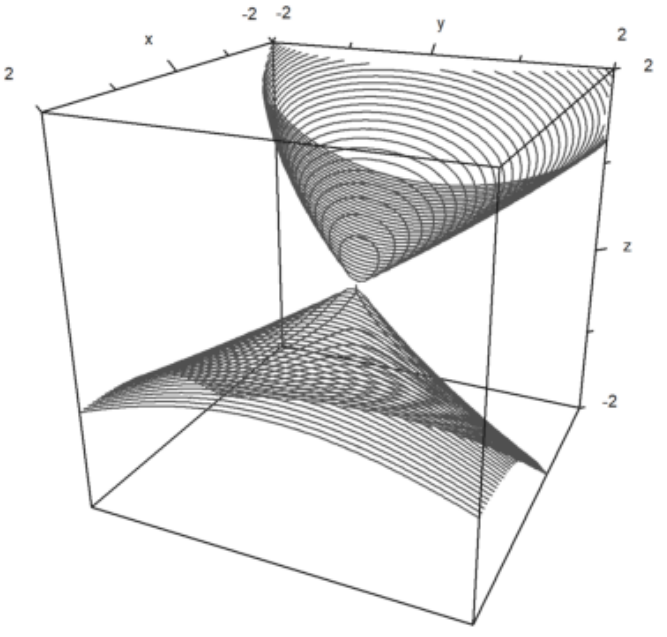
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```




```
>c=1; d=1;
>plot3d("((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^2)-d",r=2,
>plot3d("x^2+y^2+4*x*z+z^3",>implicit,r=2,zoom=2.5):
```




```
>plot3d("x^2+y^2+4*x*z+z^3",>implicit,r=2,zoom=2.5):
```


-2
-2
x
y
2
2
2
z
-2

Plot Data 3D

Seperti pada ‘plot2d‘, ‘plot3d‘ juga menerima data. Untuk objek 3D, Anda perlu menyediakan matriks nilai x, y, dan z, atau tiga fungsi atau ekspresi (f_x(x,y), f_y(x,y), f_z(x,y)).

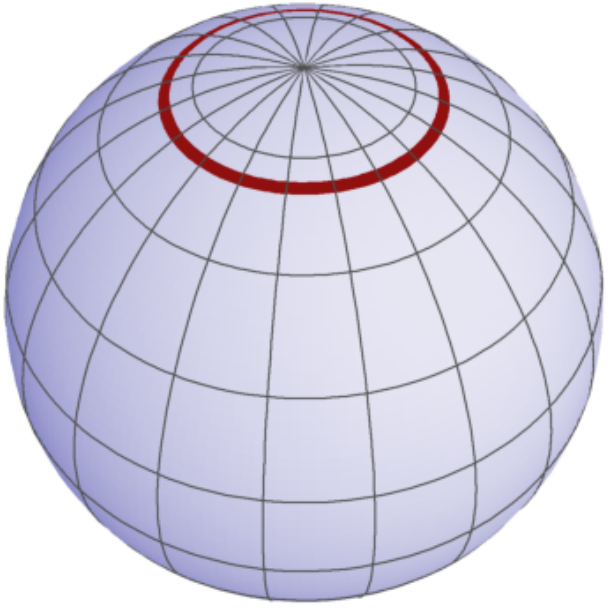
\gamma(t,s) = (x(t,s), y(t,s), z(t,s))

Karena x, y, dan z adalah matriks, diasumsikan bahwa ((t,s)) berjalan melalui grid persegi. Dengan demikian, Anda dapat memplot gambar persegi panjang di ruang.

Anda dapat menggunakan bahasa matriks Euler untuk menghasilkan koordinat secara efektif.

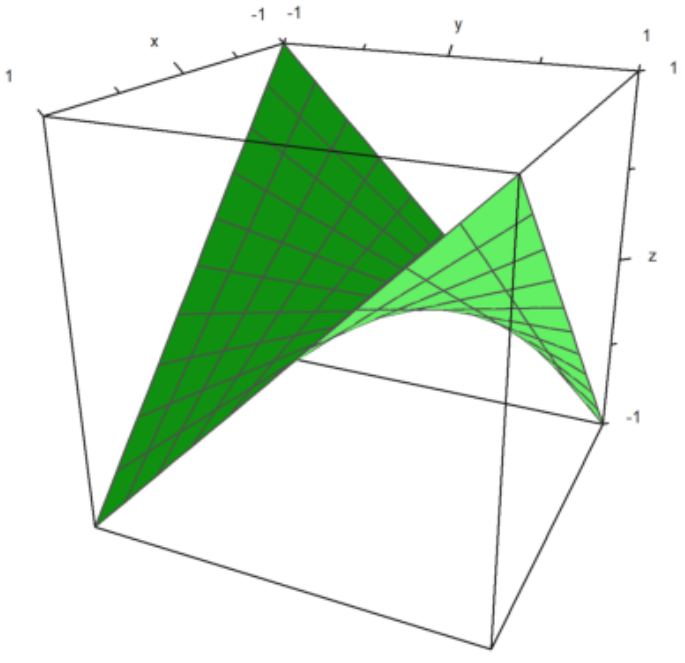
Dalam contoh berikut, kita menggunakan vektor nilai t dan vektor kolom nilai s untuk memparametrisasi permukaan bola. Dalam gambar, kita dapat menandai area tertentu, dalam kasus ini adalah daerah kutub (polar region).

```
>t=linspace(0,2pi,180); s=linspace(-pi/2,pi/2,90)’; ...
>x=cos(s)*cos(t); y=cos(s)*sin(t); z=sin(s); ...
>plot3d(x,y,z,>hue, ...
>color=blue,<frame,grid=[10,20], ...
>values=s,contourcolor=red,level=[90°-24°;90°-22°], ...
>scale=1.4,height=50°):
```



Berikut adalah sebuah contoh, yaitu grafik dari suatu fungsi.

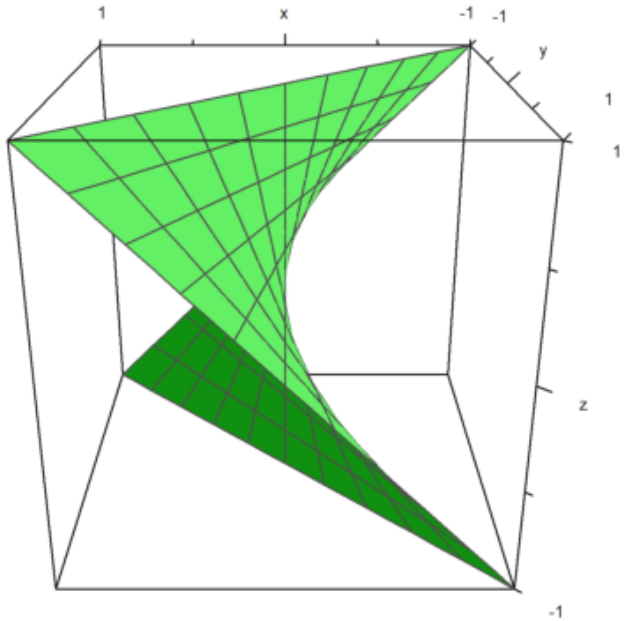
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



Namun, kita dapat membuat berbagai macam permukaan. Berikut adalah permukaan yang sama sebagai sebuah fungsi

$$x = y\,z$$

```
>plot3d(t*s,t,s,angle=180°,grid=10):
```



Dengan usaha lebih, kita dapat menghasilkan banyak permukaan.

Pada contoh berikut kita membuat tampilan berarsir dari sebuah bola yang terdistorsi. Koordinat biasa untuk bola adalah**

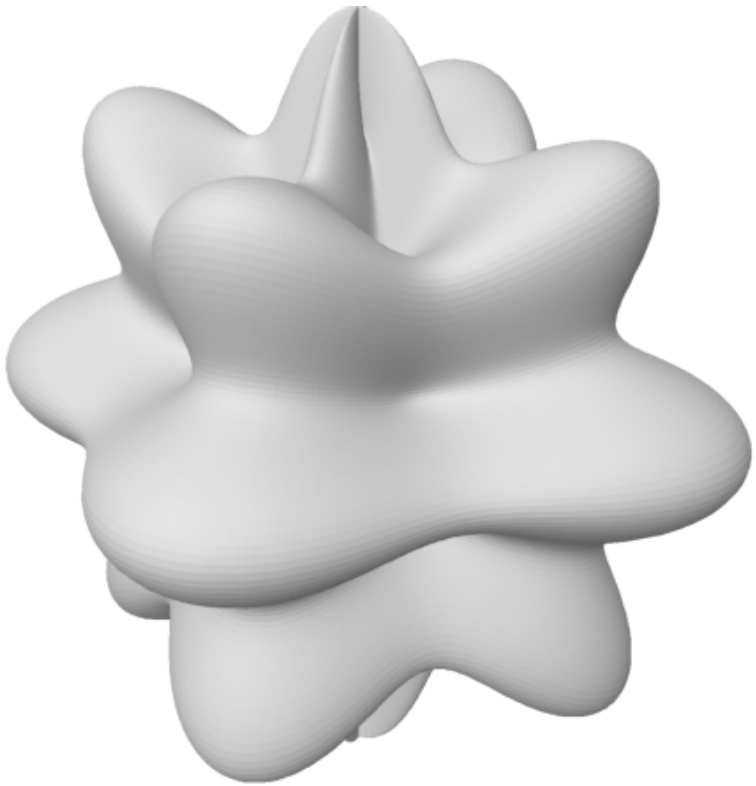
\gamma(t,s) = (\cos(t)\cos(s), \sin(t)\sin(s), \cos(s))

dengan

0 \le t \le 2\pi, \quad \frac{-\pi}{2} \le s \le \frac{\pi}{2}.

Kita mendistorsi bola ini dengan sebuah faktor
d(t,s) = \frac{\cos(4t)+\cos(8s)}{4}.

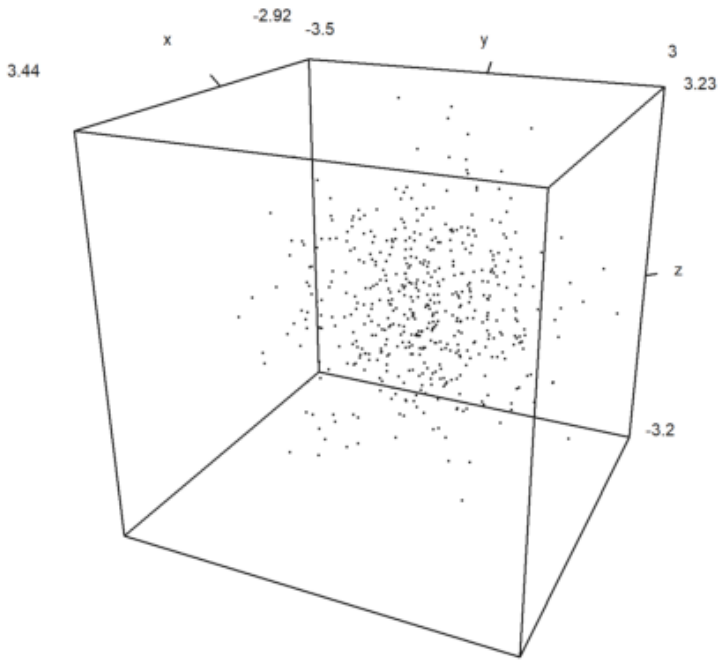
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
>  light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, sebuah point cloud juga dimungkinkan. Untuk memplot data titik dalam ruang, kita memerlukan tiga vektor untuk koordinat titik-titiknya.

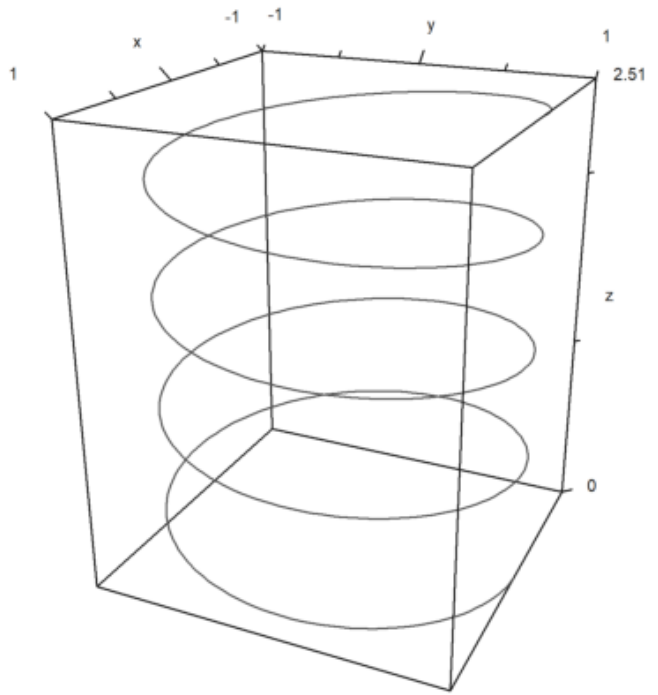
Gaya (styles) sama seperti pada plot2d dengan points=true;

```
>n=500;  ...
>  plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style="."):
```

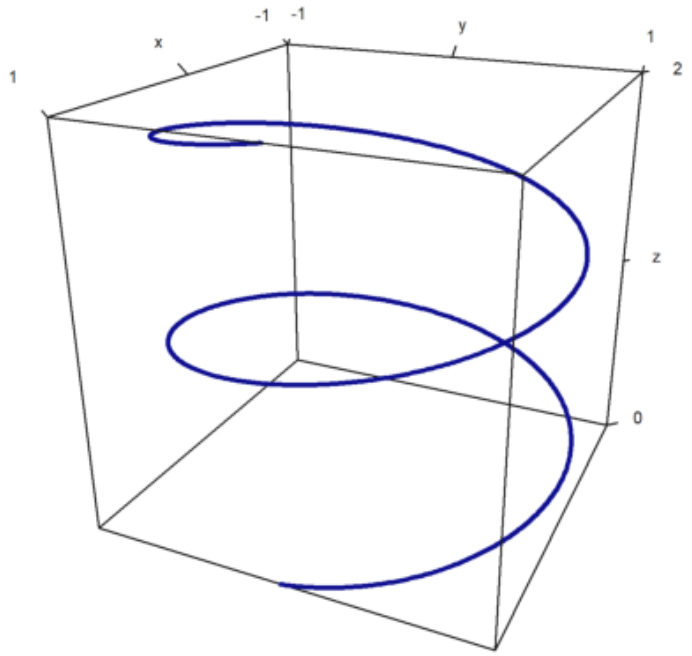


Kita juga dapat memplot sebuah kurva dalam 3D. Dalam hal ini, lebih mudah untuk menghitung terlebih dahulu titik-titik dari kurva tersebut. Untuk kurva dalam bidang, kita menggunakan suatu urutan koordinat dan parameter wire=true.

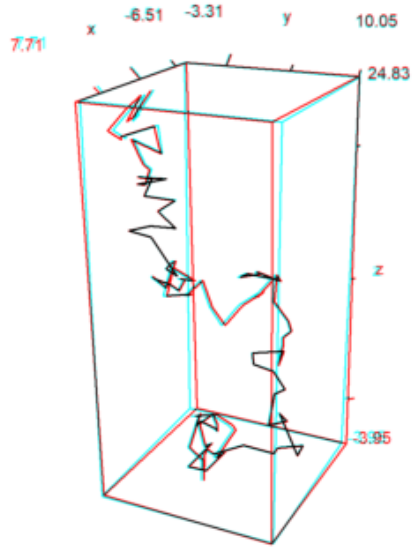
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```



```
>t=linspace(0,4pi,1000); plot3d(cos(t),sin(t),t/2pi,>wire, ...
>linewidth=3,wirecolor=blue):
```

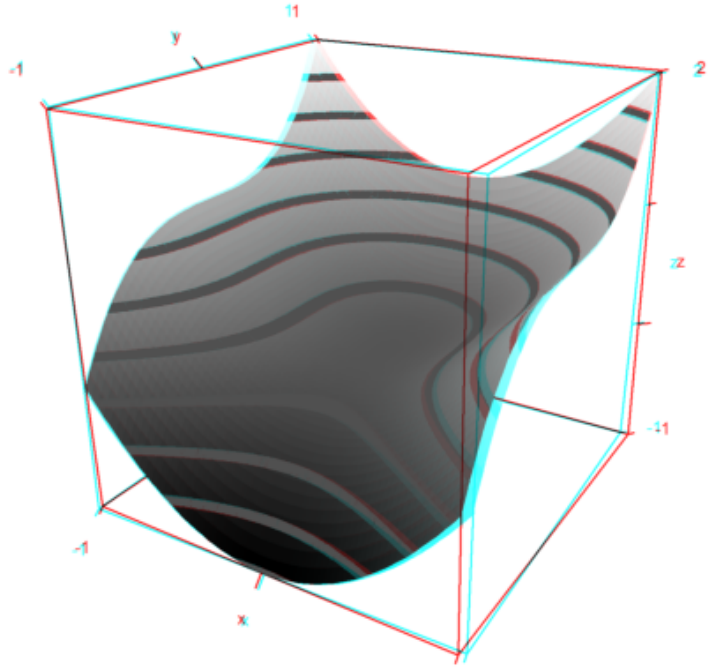



```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



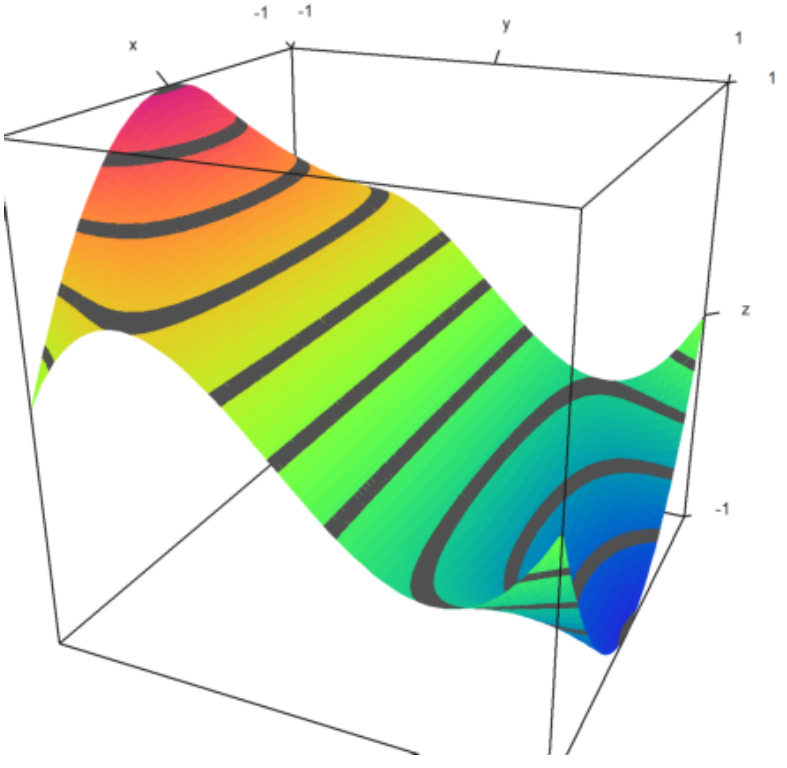
EMT juga dapat melakukan plot dalam mode anaglif. Untuk melihat plot seperti itu, Anda memerlukan kacamata merah/sian.

```
> plot3d("x^2+y^3",>anaglyph,>contour,angle=30°):
```



Sering kali, skema warna spektral digunakan untuk plot. Hal ini menekankan ketinggian dari fungsi.

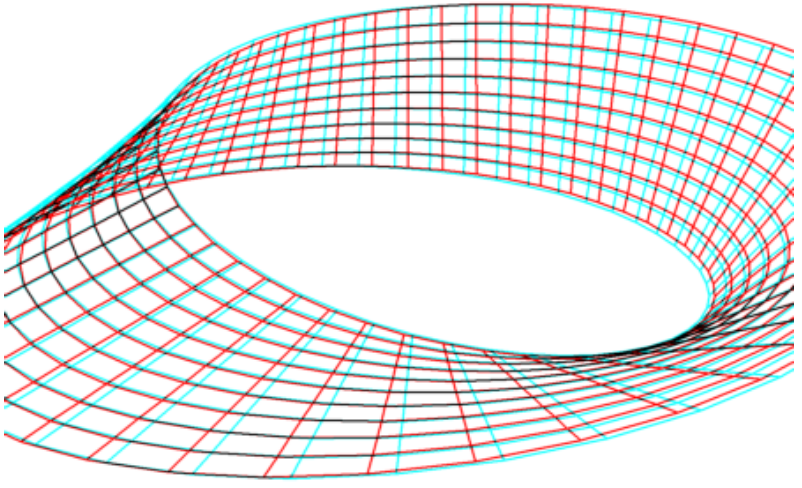
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler juga dapat memplot permukaan terparametrisasi, ketika parameter-parameter tersebut merupakan nilai x-, y-, dan z- dari citra sebuah grid persegi panjang dalam ruang.

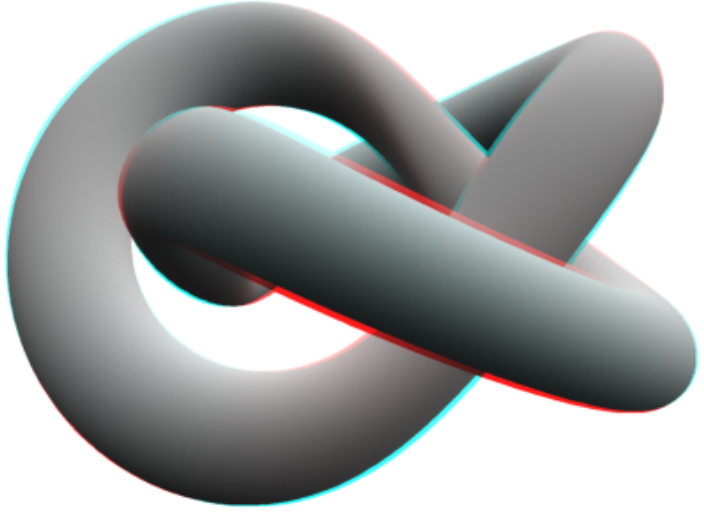
Untuk demo berikut, kita menyiapkan parameter u dan v, lalu menghasilkan koordinat ruang dari parameter-parameter tersebut.

```
>u=linspace(-1,1,10); v=linspace(0,2*pi,50)'; ...
>X=(3+u*cos(v/2))*cos(v); Y=(3+u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3):
```



Here is a more complicated example, which is majestic with red/cyan glasses.

```
>u:=linspace(-pi,pi,160); v:=linspace(-pi,pi,400)';  ...
>x:=(4*(1+.25*sin(3*v))+cos(u))*cos(2*v); ...
>y:=(4*(1+.25*sin(3*v))+cos(u))*sin(2*v); ...
> z=sin(u)+2*cos(3*v); ...
>plot3d(x,y,z,frame=0,scale=1.5,hue=1,light=[1,0,-1],zoom=2.8,>anaglyph):
```



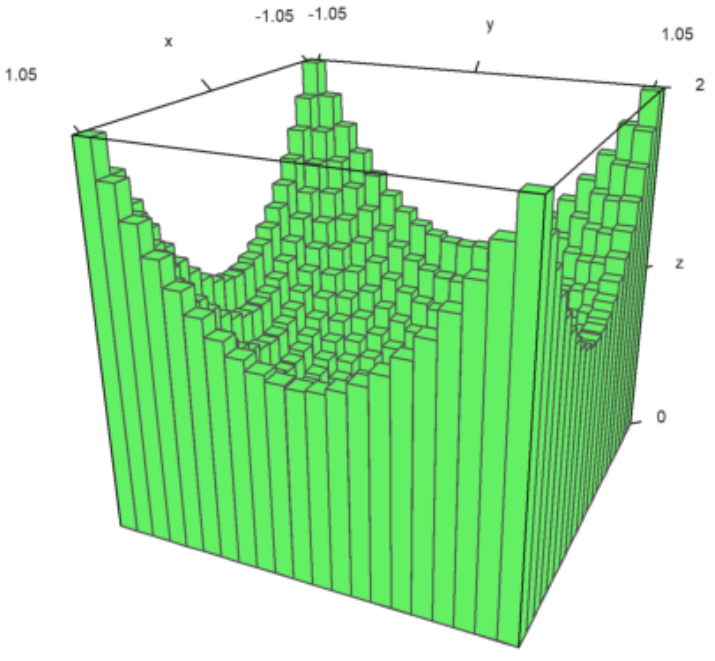
Plot batang (bar plots) juga memungkinkan. Untuk ini, kita harus menyediakan:

- x: vektor baris dengan (n+1) elemen
- y: vektor kolom dengan (n+1) elemen
- z: matriks berukuran (n \times n) yang berisi nilai-nilai

Ukuran matriks z bisa lebih besar, tetapi hanya nilai sebanyak (n \times n) yang akan digunakan.

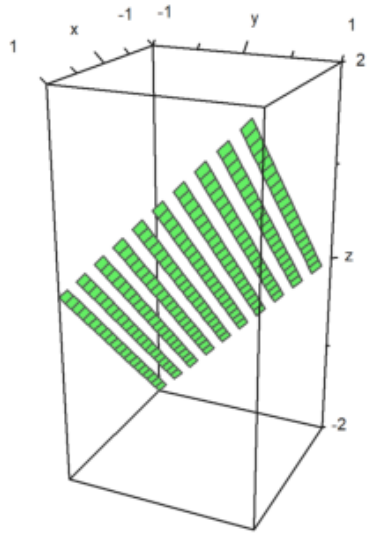
Dalam contoh, kita pertama-tama menghitung nilainya. Kemudian kita sesuaikan x dan y, sehingga vektor-vektor tersebut berada di tengah nilai-nilai yang digunakan.

```
>x=-1:0.1:1; y=x'; z=x^2+y^2; ...
>xa=(x|1.1)-0.05; ya=(y_1.1)-0.05; ...
>plot3d(xa,ya,z,bar=true):
```



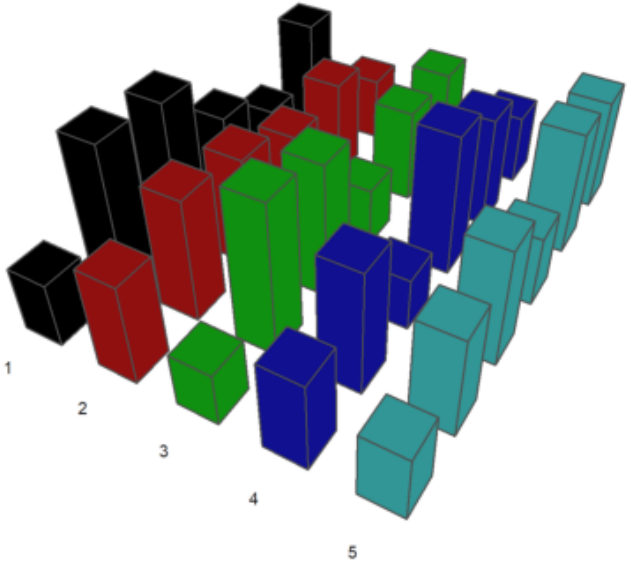
It is possible to split the plot of a surface in two or more parts.

```
>x=-1:0.1:1; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:20):
```

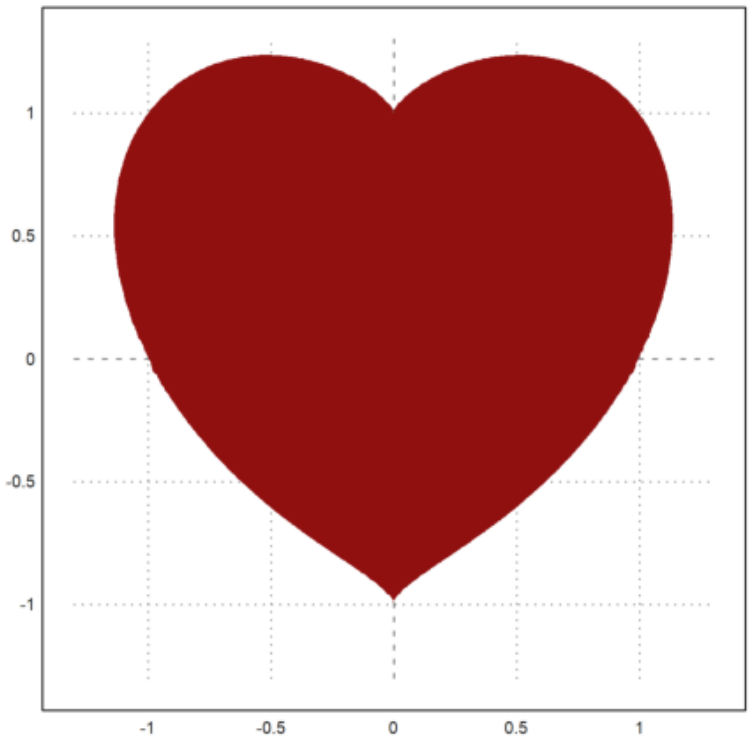


If load or generate a data matrix M from a file and need to plot it in 3D you can either scale the matrix to [-1,1] with scale(M), or scale the matrix with >zscale. This can be combined with individual scaling factors which are applied additionally.

```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8):
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...
>style="#",color=red,<outline, ...
>level=[-2;0],n=100):
```



```
>ekspresi &= (x^2+y^2-1)^3-x^2*y^3; $ekspresi
```

We wish to turn the heart curve around the y-axis. Here is the expression, which defines the heart:

$$f(x,y) = (x^2+y^2-1)^3 - x^2.y^3.$$

Next we set

$$x = r.cos(a), \quad y = r.sin(a).$$

```
>function fr(r,a) &= ekspresi with [x=r*cos(a),y=r*sin(a)] | trigreduce; $fr(r,a)
```

This allows to define a numerical function, which solves for r, if a is given. With that function we can plot the turned heart as a parametric surface.

```
>function map f(a) := bisect("fr",0,2;a); ...
>t=linspace(-pi/2,pi/2,100); r=f(t);  ...
>s=linspace(pi,2pi,100)'; ...
>plot3d(r*cos(t)*sin(s),r*cos(t)*cos(s),r*sin(t), ...
>>hue,<frame,color=red,zoom=4,amb=0,max=0.7,grid=12,height=50°):
```

The following is a 3D plot of the figure above rotated around the z-axis. We define the function, which describes the object.

```
>function f(x,y,z) ...
```

```
  r=x^2+y^2;
  return (r+z^2-1)^3-r*z^3;
   endfunction
```

```
>plot3d("f(x,y,z)", ...
>xmin=0,xmax=1.2,ymin=-1.2,ymax=1.2,zmin=-1.2,zmax=1.4, ...
>implicit=1,angle=-30°,zoom=2.5,n=[10,100,60],>anaglyph):
```

Plot 3D Khusus

Fungsi ‘plot3d‘ memang berguna, tetapi tidak memenuhi semua kebutuhan. Selain beberapa rutin dasar, dimungkinkan untuk membuat plot berbingkai dari objek apa pun yang Anda inginkan.

Meskipun Euler bukan program 3D, ia dapat menggabungkan beberapa objek dasar. Kita mencoba memvisualisasikan sebuah paraboloid beserta garis singgungnya.

```
>function myplot ...
```

```
    y=-1:0.01:1; x=(-1:0.01:1)';
    plot3d(x,y,0.2*(x-0.1)/2,<scale,<frame,>hue, ..
      hues=0.5,>contour,color=orange);
    h=holding(1);
    plot3d(x,y,(x^2+y^2)/2,<scale,<frame,>contour,>hue);
    holding(h);
  endfunction
```

Now framedplot() provides the frames, and sets the views.

```
>framedplot("myplot",[-1,1,-1,1,0,1],height=0,angle=-30°, ...
>  center=[0,0,-0.7],zoom=3):
```

In the same way, you can plot the contour plane manually. Note that plot3d() sets the window to fullwindow() by default, but plotcontourplane() assumes that.

```
>x=-1:0.02:1.1; y=x'; z=x^2-y^4;
>function myplot (x,y,z) ...
```

```
    zoom(2);
    wi=fullwindow();
    plotcontourplane(x,y,z,level="auto",<scale);
    plot3d(x,y,z,>hue,<scale,>add,color=white,level="thin");
    window(wi);
    reset();
  endfunction
```

```
>myplot(x,y,z):
```

Animasi

Euler dapat menggunakan frame untuk melakukan pra-komputasi animasi.

Salah satu fungsi yang memanfaatkan teknik ini adalah rotate. Fungsi ini dapat mengubah sudut pandang dan menggambar ulang plot 3D. Fungsi tersebut memanggil addpage() untuk setiap plot baru. Akhirnya, fungsi ini akan menganimasikan plot-plot tersebut.

Silakan pelajari kode sumber fungsi rotate untuk melihat detail lebih lanjut.

```
>function testplot () := plot3d("x^2+y^3"); ...
>rotate("testplot"); testplot():
```

Menggambar Povray

Dengan bantuan file Euler ‘povray.e‘, Euler dapat menghasilkan file-file Povray. Hasilnya sangat indah untuk dilihat.

Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/, dan memasukkan sub-direktori ”bin” dari Povray ke dalam environment path, atau mengatur variabel ‘defaultpovray dengan path lengkap yang mengarah ke ‘pvengine.exe‘.

Antarmuka Povray di Euler menghasilkan file-file Povray di direktori home pengguna, dan memanggil Povray untuk memproses file-file tersebut. Nama file default adalah ‘current.pov‘, dan direktori default adalah ‘eulerhome()‘, biasanya terletak di ‘c:\Users\Username\Euler‘. Povray menghasilkan file PNG, yang dapat dimuat oleh Euler ke dalam notebook. Untuk membersihkan file-file ini, gunakan fungsi ‘povclear()‘.

Fungsi ‘pov3d‘ memiliki tujuan yang sama dengan ‘plot3d‘. Fungsi ini dapat menghasilkan grafik dari fungsi (f(x,y)), atau sebuah permukaan dengan koordinat X, Y, Z dalam bentuk matriks, termasuk garis level opsional. Fungsi ini secara otomatis memulai raytracer, dan memuat scene ke dalam notebook Euler.

Selain ‘pov3d()‘, terdapat banyak fungsi lain yang menghasilkan objek Povray. Fungsi-fungsi ini mengembalikan string yang berisi kode Povray untuk objek tersebut. Untuk menggunakan fungsi-fungsi ini, mulai file Povray dengan ‘povstart(). Kemudian gunakan ‘writeln(...)‘ untuk menulis objek-objek ke file scene. Akhirnya, tutup file dengan ‘povend()‘. Secara default, raytracer akan mulai berjalan, dan file PNG akan dimasukkan ke dalam notebook Euler.

Fungsi objek memiliki parameter bernama ‘look‘, yang membutuhkan string berisi kode Povray untuk tekstur dan finishing objek. Fungsi ‘povlook()‘ dapat digunakan untuk menghasilkan string ini. Fungsi ini memiliki parameter untuk warna, transparansi, Phong shading, dan lain-lain.

Perlu diperhatikan bahwa sistem koordinat di dunia Povray berbeda. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray. Jadi, Anda tetap dapat berpikir menggunakan sistem koordinat Euler dengan sumbu z mengarah vertikal ke atas, dan sumbu x, y, z mengikuti aturan tangan kanan.

Anda perlu memuat file Povray tersebut untuk menggunakannya.

```
>load povray;
```

Make sure, the Povray bin directory is in the path. If it is not edit the following variable so that it contains the path to the povray executable.

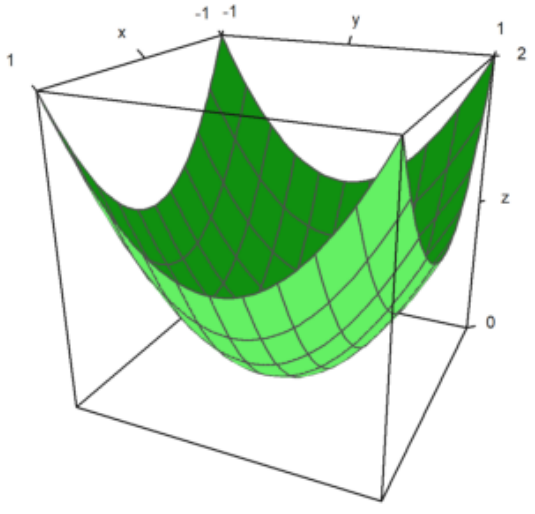
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

```
  C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe
```

For a first impression, we plot a simple function. The following command generates a povray file in your user directory, and runs Povray for ray tracing this file.

If you start the following command, the Povray GUI should open, run the file, and close automatically. Due to security reasons, you will be asked, if you want to allow the exe file to run. You can press cancel to stop further questions. You may have to press OK in the Povray window to acknowledge the start-up dialog of Povray.

```
>plot3d("x^2+y^2",zoom=2):
```




```
>pov3d("x^2+y^2",zoom=3);
```

We can make the function transparent and add another finish. We can also add level lines to the function plot.

```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
>  look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```

Sometimes it is necessary to prevent the scaling of the function, and scale the function by hand.

We plot the set of points in the complex plane, where the product of the distances to 1 and -1 is equal to 1.

```
>pov3d("((x-1)^2+y^2)*((x+1)^2+y^2)/40",r=2, ...
>  angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=10°,n=50, ...
>  <fscale,zoom=3.8);
```

Plotting with Coordinates

Instead of functions, we can plot with coordinates. As in plot3d, we need three matrices to define the object.

In the example we turn a function around the z-axis.

```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,axis=0,zoom=4,light=[10,5,15]);
```

In the following example, we plot a damped wave. We generate the wave with the matrix language of Euler.

We also show, how an additional object can be added to a pov3d scene. For the generation of objects, see the following examples. Note that plot3d scales the plot, so that it fits into the unit cube.

```
>r=linspace(0,1,80); phi=linspace(0,2pi,80)'; ...
>x=r*cos(phi); y=r*sin(phi); z=exp(-5*r)*cos(8*pi*r)/3;  ...
>pov3d(x,y,z,zoom=6,axis=0,height=30°,add=povsphere([0.5,0,0.25],0.15,povlook(red)), ...
>  w=500,h=300);
```

With the advanced shading method of Povray, very few points can produce very smooth surfaces. Only at the boundaries and in shadows the trick might become obvious.

For this, we need to add normal vectors in each matrix point.

```
>Z &= x^2*y^3
```

$$x^2 y^3$$

The equation of the surface is [x,y,Z]. We compute the two derivatives to x and y of this and take the cross product as the normal.

```
>dx &= diff([x,y,Z],x); dy &= diff([x,y,Z],y);
```

We define the normal as the cross product of these derivatives, and define coordinate functions.

```
>N &= crossproduct(dx,dy); NX &= N[1]; NY &= N[2]; NZ &= N[3]; N,
```

$$\left[-2 x y^3, -3 x^2 y^2, 1\right]$$

We use only 25 points.

```
>x=-1:0.5:1; y=x';
>pov3d(x,y,Z(x,y),angle=10°, ...
>  xv=NX(x,y),yv=NY(x,y),zv=NZ(x,y),<shadow);
```

The following is the Trefoil knot done by A. Busser in Povray. There is an improved version of this in the examples.

See: Examples\Trefoil Knot | Trefoil Knot

For a good look with not too many points, we add normal vectors here. We use Maxima to compute the normals for us. First, the three functions for the coordinates as symbolic expressions.

```
>X &= ((4+sin(3*y))+cos(x))*cos(2*y); ...
>Y &= ((4+sin(3*y))+cos(x))*sin(2*y); ...
>Z &= sin(x)+2*cos(3*y);
```

Then the two derivative vectors to x and y.

```
>dx &= diff([X,Y,Z],x); dy &= diff([X,Y,Z],y);
```

Now the normal, which is the cross product of the two derivatives.

```
>dn &= crossproduct(dx,dy);
```

We now evaluate all this numerically.

```
>x:=linspace(-%pi,%pi,40); y:=linspace(-%pi,%pi,100)';
```

Vektor normal adalah hasil evaluasi dari ekspresi simbolik ‘dn[i]‘ untuk (i=1,2,3). Sintaksnya adalah ‘&”expression”(parameters)‘. Ini merupakan alternatif dari metode pada contoh sebelumnya, di mana kita terlebih dahulu mendefinisikan ekspresi simbolik ‘NX‘, ‘NY‘, dan ‘NZ‘.

```
>pov3d(X(x,y),Y(x,y),Z(x,y),>anaglyph,axis=0,zoom=5,w=450,h=350, ...
>  <shadow,look=povlook(blue), ...
>  xv=&"dn[1]"(x,y), yv=&"dn[2]"(x,y), zv=&"dn[3]"(x,y));
```

We can also generate a grid in 3D.

```
>povstart(zoom=4); ...
>x=-1:0.5:1; r=1-(x+1)^2/6; ...
>t=(0°:30°:360°)'; y=r*cos(t); z=r*sin(t); ...
>writeln(povgrid(x,y,z,d=0.02,dballs=0.05)); ...
>povend();
```

With povgrid(), curves are possible.

```
>povstart(center=[0,0,1],zoom=3.6); ...
>t=linspace(0,2,1000); r=exp(-t); ...
>x=cos(2*pi*10*t)*r; y=sin(2*pi*10*t)*r; z=t; ...
>writeln(povgrid(x,y,z,povlook(red))); ...
>writeAxis(0,2,axis=3); ...
>povend();
```

Di atas, kita menggunakan ‘pov3d‘ untuk memplot permukaan. Antarmuka Povray di Euler juga dapat menghasilkan objek Povray. Objek-objek ini disimpan sebagai string di Euler, dan perlu ditulis ke dalam file Povray.

Kita memulai output dengan ‘povstart()‘.

```
>povstart(zoom=4);
```

First we define the three cylinders, and store them in strings in Euler.

The functions povx() etc. simply returns the vector [1,0,0], which could be used instead.

```
>c1=povcylinder(-povx,povx,1,povlook(red)); ...
>c2=povcylinder(-povy,povy,1,povlook(yellow)); ...
>c3=povcylinder(-povz,povz,1,povlook(blue)); ...
```

The strings contain Povray code, which we need not understand at that point.

```
>c2
```

```
cylinder { <0,0,-1>, <0,0,1>, 1
 texture { pigment { color rgb <0.941176,0.941176,0.392157> }  }
 finish { ambient 0.2 }
 }
```

As you see, we added texture to the objects in three different colors.

That is done by povlook(), which returns a string with the relevant Povray code. We can use the default Euler colors, or define our own color. We can also add transparency, or change the ambient light.

```
>povlook(rgb(0.1,0.2,0.3),0.1,0.5)
```

```
 texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> }  }
 finish { ambient 0.5 }
```

Now we define an intersection object, and write the result to the file.

```
>writeln(povintersection([c1,c2,c3]));
```

The intersection of three cylinders is hard to visualize, if you never saw it before.

```
>povend;
```

The following functions generate a fractal recursively.

The first function shows, how Euler handles simple Povray objects. The function povbox() returns a string, containing the box coordinates, the texture and the finish.

```
>function onebox(x,y,z,d) := povbox([x,y,z],[x+d,y+d,z+d],povlook());
>function fractal (x,y,z,h,n) ...
```

```
  if n==1 then writeln(onebox(x,y,z,h));
  else
    h=h/3;
    fractal(x,y,z,h,n-1);
    fractal(x+2*h,y,z,h,n-1);
    fractal(x,y+2*h,z,h,n-1);
    fractal(x,y,z+2*h,h,n-1);
    fractal(x+2*h,y+2*h,z,h,n-1);
    fractal(x+2*h,y,z+2*h,h,n-1);
    fractal(x,y+2*h,z+2*h,h,n-1);
    fractal(x+2*h,y+2*h,z+2*h,h,n-1);
    fractal(x+h,y+h,z+h,h,n-1);
  endif;
endfunction
```

```
>povstart(fade=10,<shadow);
>fractal(-1,-1,-1,2,4);
>povend();
```

Differences allow cutting off one object from another. Like intersections, there are part of the CSG objects of Povray.

```
>povstart(light=[5,-5,5],fade=10);
```

For this demonstration, we define an object in Povray, instead of using a string in Euler. Definitions are written to the file immediately.

A box coordinate of -1 just means [-1,-1,-1].

```
>povdefine("mycube",povbox(-1,1));
```

We can use this object in povobject(), which returns a string as usual.

```
>c1=povobject("mycube",povlook(red));
```

We generate a second cube, and rotate and scale it a bit.

```
>c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...
>  rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Then we take the difference of the two objects.

```
>writeln(povdifference(c1,c2));
```

Now add three axes.

```
>writeAxis(-1.2,1.2,axis=1); ...
>writeAxis(-1.2,1.2,axis=2); ...
>writeAxis(-1.2,1.2,axis=4); ...
>povend();
```

Fungsi Implisit

Povray dapat memplot himpunan di mana (f(x,y,z) = 0), sama seperti parameter implisit pada ‘plot3d‘. Namun, hasilnya terlihat jauh lebih baik.

Sintaks untuk fungsi-fungsi tersebut sedikit berbeda. Anda tidak bisa menggunakan keluaran dari ekspresi Maxima atau Euler.

((x2+y2-c2)2+(z2-1)2)×((y2+z2-c2)2+(x2-1)2)×((z2+x2-c2)2+(y2-1)2)=d

```
>povstart(angle=70°,height=50°,zoom=4);
>c=0.1; d=0.1; ...
>writeln(povsurface("(pow(pow(x,2)+pow(y,2)-pow(c,2),2)+pow(pow(z,2)-1,2))*(pow(pow(y,2)+pow(z,2)-po
>povend();
```

```
Error : Povray error!

Error generated by error() command

povray:
    error("Povray error!");
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

```
>povstart(angle=25°,height=10°);
>writeln(povsurface("pow(x,2)+pow(y,2)*pow(z,2)-1",povlook(blue),povbox(-2,2,"")));
>povend();
>povstart(angle=70°,height=50°,zoom=4);
```

Create the implicit surface. Note the different syntax in the expression.

```
>writeln(povsurface("pow(x,2)*y-pow(y,3)-pow(z,2)",povlook(green))); ...
>writeAxes(); ...
>povend();
```

Objek Mesh

Dalam contoh ini, kami menunjukkan cara membuat objek mesh, dan menggambarnya dengan informasi tambahan.

Kami ingin memaksimalkan (xy) dengan syarat (x + y = 1) dan mendemonstrasikan sentuhan tangensial dari garis-garis level.

```
>povstart(angle=-10°,center=[0.5,0.5,0.5],zoom=7);
```

Kita tidak bisa menyimpan objek dalam sebuah string seperti sebelumnya, karena ukurannya terlalu besar. Jadi, kita mendefinisikan objek tersebut dalam sebuah file Povray menggunakan declare. Fungsi ‘povtriangle()‘ melakukan ini secara otomatis. Fungsi ini dapat menerima vektor normal seperti halnya ‘pov3d()‘.

Berikut ini mendefinisikan objek mesh dan langsung menuliskannya ke dalam file.

```
>x=0:0.02:1; y=x'; z=x*y; vx=-y; vy=-x; vz=1;
>mesh=povtriangles(x,y,z,"",vx,vy,vz);
```

Now we define two discs, which will be intersected with the surface.

```
>cl=povdisc([0.5,0.5,0],[1,1,0],2); ...
>ll=povdisc([0,0,1/4],[0,0,1],2);
```

Write the surface minus the two discs.

```
>writeln(povdifference(mesh,povunion([cl,ll]),povlook(green)));
```

Write the two intersections.

```
>writeln(povintersection([mesh,cl],povlook(red)); ...
>writeln(povintersection([mesh,ll],povlook(gray)));
```

Write a point at the maximum.

```
>writeln(povpoint([1/2,1/2,1/4],povlook(gray),size=2*defaultpointsize));
```

Add axes and finish.

```
>writeAxes(0,1,0,1,0,1,d=0.015); ...
>povend();
```

Anaglif di Povray

Untuk menghasilkan anaglif bagi kacamata merah/sian, Povray harus dijalankan dua kali dari posisi kamera yang berbeda. Povray akan menghasilkan dua file Povray dan dua file PNG, yang kemudian dapat dimuat dengan fungsi ‘loadanaglyph()‘.

Tentu saja, Anda memerlukan kacamata merah/sian agar dapat melihat contoh berikut dengan benar.

Fungsi ‘pov3d()‘ memiliki saklar sederhana untuk menghasilkan anaglif.

```
>pov3d("-exp(-x^2-y^2)/2",r=2,height=45°,>anaglyph, ...
>  center=[0,0,0.5],zoom=3.5);
```

If you create a scene with objects, you need to put the generation of the scene into a function, and run it twice with different values for the anaglyph parameter.

```
>function myscene ...
```

```
    s=povsphere(povc,1);
    cl=povcylinder(-povz,povz,0.5);
    clx=povobject(cl,rotate=xrotate(90°));
    cly=povobject(cl,rotate=yrotate(90°));
    c=povbox([-1,-1,0],1);
    un=povunion([cl,clx,cly,c]);
    obj=povdifference(s,un,povlook(red));
    writeln(obj);
    writeAxes();
  endfunction
```

The function povanaglyph() does all this. The parameters are like in povstart() and povend() combined.

```
>povanaglyph("myscene",zoom=4.5);
```

Mendefinisikan Objek Sendiri

Antarmuka Povray dari Euler berisi banyak objek. Namun, Anda tidak terbatas pada objek-objek tersebut. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau benar-benar objek baru.

Kita akan mendemonstrasikan sebuah torus. Perintah Povray untuk ini adalah "torus". Jadi, kita mengembalikan sebuah string dengan perintah ini beserta parameternya. Perlu diperhatikan bahwa torus selalu dipusatkan di titik asal (origin).

```
>function povdonat (r1,r2,look="") ...
```

```
    return "torus {"+r1+","+r2+look+"}";
  endfunction
```

Here is our first torus.

```
>t1=povdonat(0.8,0.2)
```

```
  torus {0.8,0.2}
```

Let us use this object to create a second torus, translated and rotated.

```
>t2=povobject(t1,rotate=xrotate(90°),translate=[0.8,0,0])
```

```
object { torus {0.8,0.2}
 rotate 90 *x
 translate <0.8,0,0>
 }
```

Now we place these objects into a scene. For the look, we use Phong Shading.

```
>povstart(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); ...
>writeln(povobject(t1,povlook(green,phong=1))); ...
>writeln(povobject(t2,povlook(green,phong=1))); ...
```

```
>povend();
```

calls the Povray program. However, in case of errors, it does not display the error. You should therefore use

```
>povend(<exit);
```

if anything did not work. This will leave the Povray window open.

```
>povend(h=320,w=480);
```

```
Function povstart not found.
Try list ... to find functions!
Error in:
povstart(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); writeln(povobject(t1,povlook(green,phong=
                                                                ^
```

Here is a more elaborate example. We solve

$$Ax \le b, \quad x \ge 0, \quad c.x \to \text{Max}.$$

and show the feasible points and the optimum in a 3D plot.

```
>A=[10,8,4;5,6,8;6,3,2;9,5,6];
>b=[10,10,10,10]';
>c=[1,1,1];
```

First, let us check, if this example has a solution at all.

```
>x=simplex(A,b,c,>max,>check)'
```

```
[0,  1,  0.5]
```

Yes, it has.

Next we define two objects. The first is the plane

$$a \cdot x \le b$$

```
>function oneplane (a,b,look="") ...
```

```
    return povplane(a,b,look)
  endfunction
```

Then we define the intersection of all half spaces and a cube.

```
>function adm (A, b, r, look="") ...
```

```
    ol=[];
    loop 1 to rows(A); ol=ol|oneplane(A[#],b[#]); end;
    ol=ol|povbox([0,0,0],[r,r,r]);
    return povintersection(ol,look);
  endfunction
```

We can now plot the scene.

```
>povstart(angle=120°,center=[0.5,0.5,0.5],zoom=3.5); ...
>writeln(adm(A,b,2,povlook(green,0.4))); ...
>writeAxes(0,1.3,0,1.6,0,1.5); ...
```

The following is a circle around the optimum.

```
>writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')], ...
>  povlook(red,0.9)));
```

And an error in the direction of the optimum.

```
>writeln(povarrow(x,c*0.5,povlook(red)));
```

We add text to the screen. Text is just a 3D object. We need to place and turn it according to our view.

```
>writeln(povtext("Linear Problem",[0,0.2,1.3],size=0.05,rotate=5°)); ...
>povend();
```

More Examples

You can find some more examples for Povray in Euler in the following files.

See: Examples/Dandelin Spheres
See: Examples/Donat Math
See: Examples/Trefoil Knot
See: Examples/Optimization by Affine Scaling