Identitas

Nama : Intan Nur Setyarini
NIM : 24030130001
Kelas : Pendidikan Matematika A

Menggambar Plot 3D dengan EMT

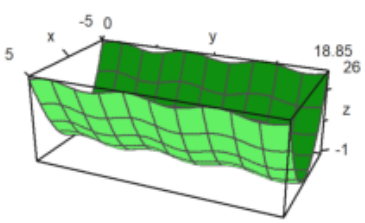
Pendahuluan Plot 3D di Euler. Kita memerlukan plot 3D untuk memvisualisasikan suatu fungsi dengan dua variabel.

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

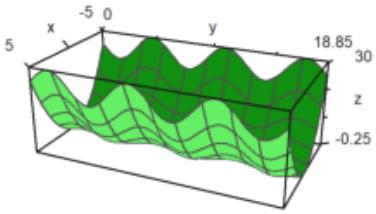
Euler dapat membuat plot:
- permukaan dengan arsiran dan garis kontur atau rentang kontur,
- kumpulan titik,
- kurva parametrik,
- permukaan implisit.

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

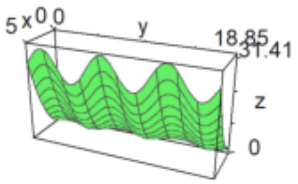
```
>aspect(1.5); plot3d("x^2+sin(y)",-5,5,0,6*pi):
```




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

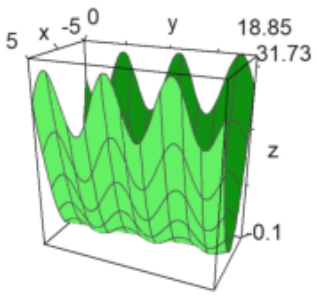



```
>plot3d("x^2+x*sin(y)+0.1*(y)",0,5,0,6*pi,zoom=1):
```



Gambar tersebut adalah permukaan 3D yang berupa parabola dalam arah x, bergelombang sinusoidal dalam arah y, dan sekaligus miring naik karena adanya komponen linear 0.1y.

```
>plot3d("x^2+x*sin(y)+0.1*(y)",-5,5,0,6*pi,zoom=1):
```



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 dengan Dua Variabel

Untuk grafik suatu fungsi, kita dapat menggunakan:

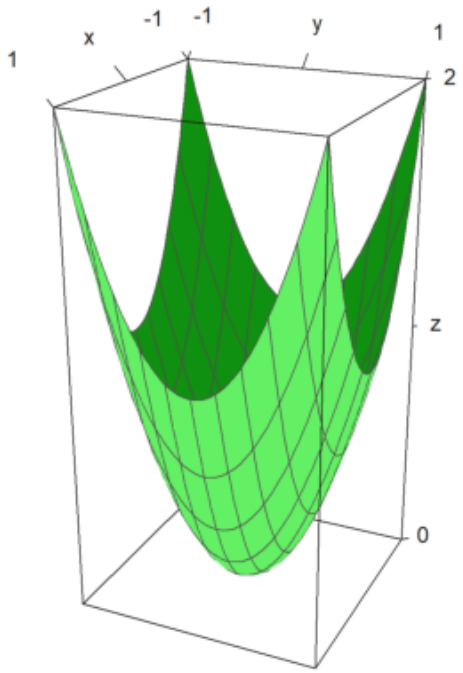
- ekspresi sederhana dalam variabel X dan y,
- nama suatu fungsi dengan dua variabel,
- atau matriks data.

Secara default, grafik ditampilkan sebagai wire grid (jaring kawat) yang terisi dengan warna berbeda pada kedua sisinya. Perlu diperhatikan bahwa jumlah interval grid bawaan adalah 10, tetapi plot menggunakan jumlah default 40x40 persegi panjang untuk membangun permukaan. Nilai ini bisa diubah.

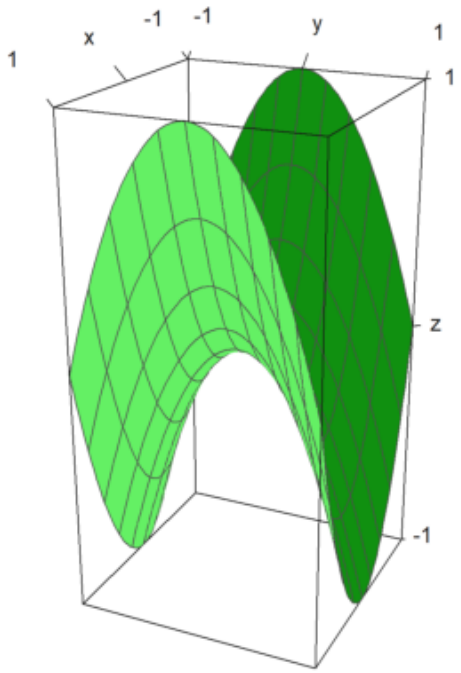
- n=40, n=[40,40]: jumlah titik grid (interval) dalam tiap arah.
- grid=10, grid=[10,10]: jumlah garis grid dalam tiap arah.

Dalam contoh ini, kita menggunakan nilai default n=40 dan grid=10.

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




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

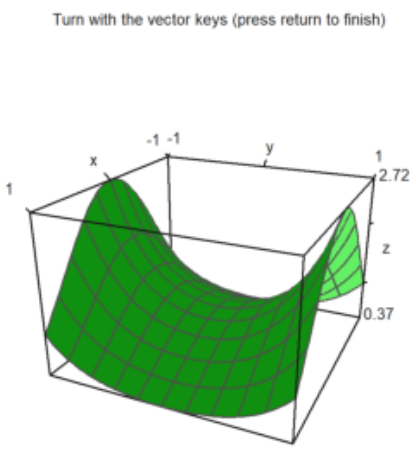


gambar tersebut adalah potongan kecil dari permukaan pelana z=x^2-y^2 yang melengkung ke atas sepanjang sumbu x dan melengkung ke bawah sepanjang sumbu y.

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

- left, right, up, down: mengubah sudut pandang
- + , -: memperbesar atau memperkecil tampilan (zoom in/out)
- a: menghasilkan anaglyph (lihat penjelasan di bawah)
- l: menyalakan/mematikan sumber cahaya (lihat penjelasan di bawah)
- space: mengembalikan ke tampilan default
- return (Enter): mengakhiri interaksi

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



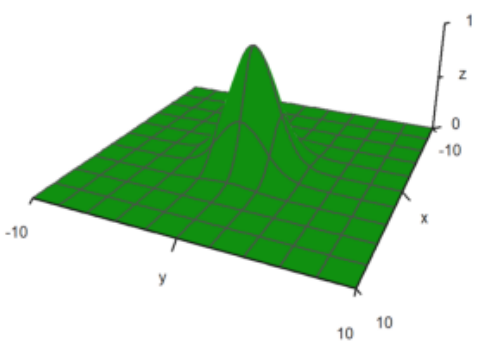
Rentang plot untuk fungsi dapat ditentukan dengan:

- a,b: rentang sumbu-x
- c,d: rentang sumbu-y
- r: sebuah persegi simetris di sekitar titik
- n: jumlah subinterval untuk plot.

Terdapat beberapa parameter untuk melakukan skala pada fungsi atau mengubah tampilan grafik:

fscale: menskalakan terhadap nilai fungsi (default adalah <fscale>).
scale: sebuah angka atau vektor 1x2 untuk skala pada arah x dan y.
frame: tipe bingkai (default = 1)

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



Tampilan (View) dapat diubah dengan berbagai cara.

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

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

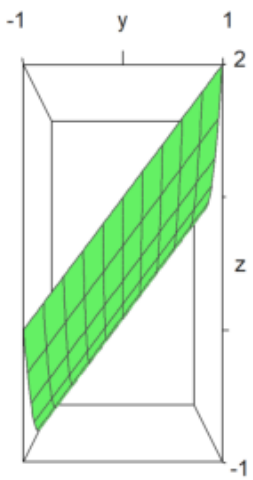
```
>view
```

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

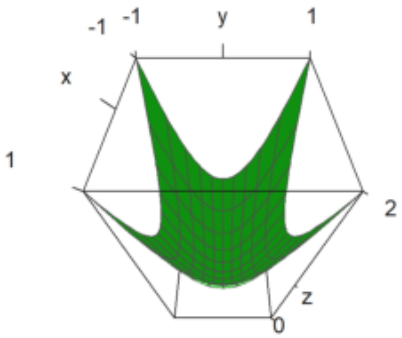
Jarak pandang yang lebih dekat membutuhkan zoom yang lebih kecil. Efeknya mirip dengan penggunaan lensa sudut lebar (wide angle lens).

Dalam contoh berikut, angle=0 dan height=0 membuat tampilan berasal dari arah sumbu-y negatif. Pada kasus ini, label sumbu-y akan tersembunyi.

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

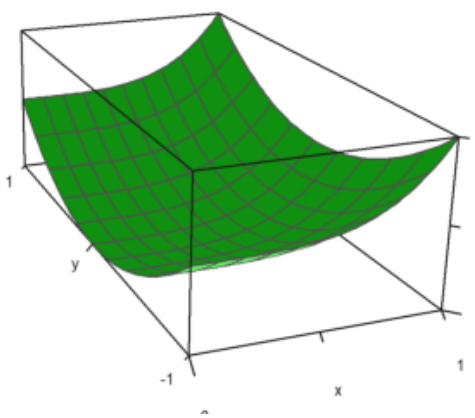



```
>plot3d("x^2+y^2",distance=2,zoom=0.5,angle=pi/2,height=1):
```



Plot selalu diarahkan ke pusat (center) dari kubus plot. Anda dapat memindahkan pusat tersebut dengan menggunakan parameter center.

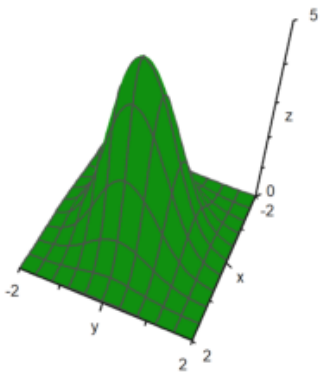
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>  center=[0.4,0,0],zoom=5):
```



Plot diskalakan agar muat ke dalam sebuah unit cube untuk dilihat. Jadi, tidak perlu mengubah distance atau zoom tergantung pada ukuran plot. Namun, label sumbu tetap merujuk pada ukuran sebenarnya.

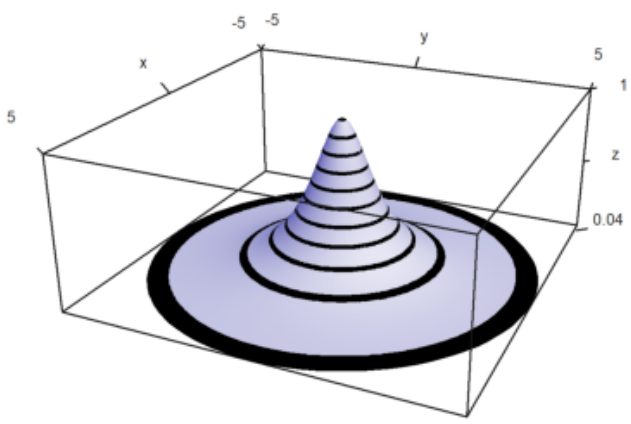
Jika Anda mematikan fitur ini dengan scale=false, maka Anda perlu memastikan sendiri bahwa plot tetap pas di jendela tampilan, dengan cara mengubah jarak pandang (distance) atau zoom, serta memindahkan center.

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

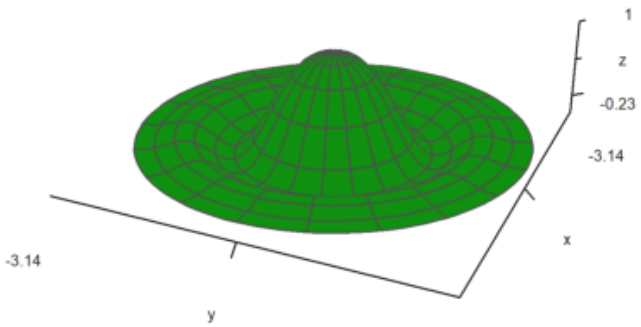


Plot polar juga tersedia. Parameter polar=true akan menggambar plot polar. Fungsi yang digunakan tetap harus berupa fungsi dari x dan y. Parameter fscale akan menskalakan fungsi dengan skala tersendiri. Jika tidak, fungsi akan diskalakan agar pas ke dalam sebuah kubus.

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



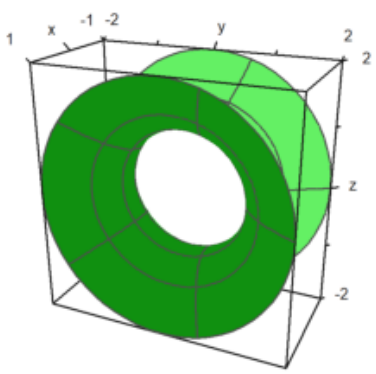

```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



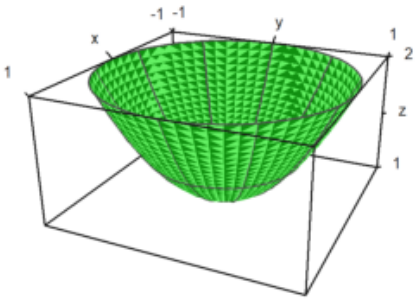
Parameter rotate berfungsi untuk memutar suatu fungsi terhadap sumbu tertentu.

- rotate=1: memutar fungsi terhadap sumbu-x
- rotate=2: memutar fungsi terhadap sumbu-z

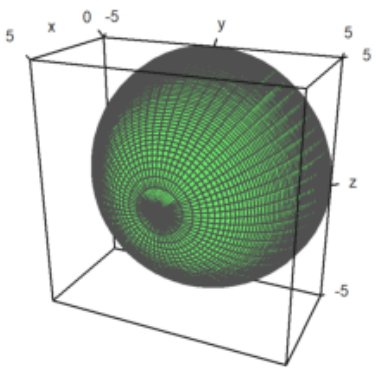
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



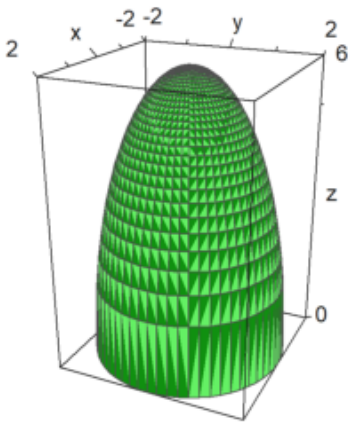

```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




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




```
>plot3d("sqrt(max(36-9x^2-4y^2))",-2,2,-3,3,rotate=2,zoom=1.5):
```

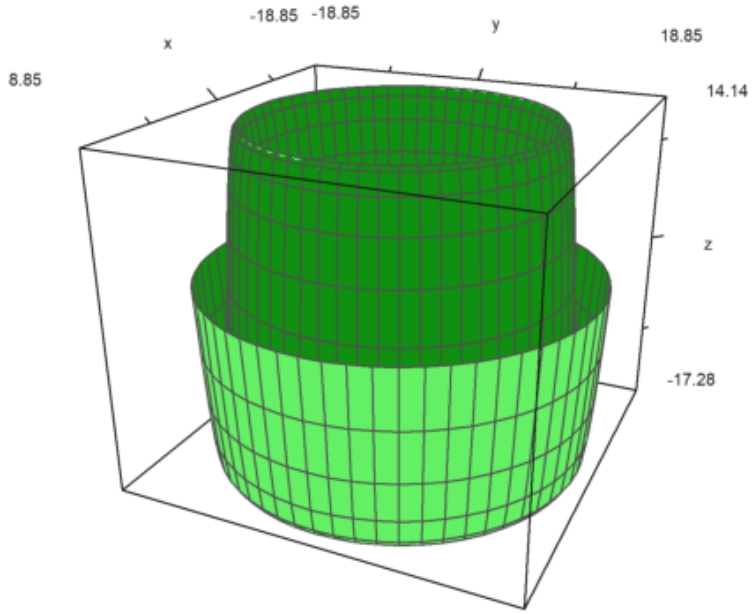


Gambar di atas adalah grafik setengah ellipsoid (bagian atasnya saja) yang didefinisikan oleh persamaan

$$z^2 + 9x^2 + 4y^2 = 36$$

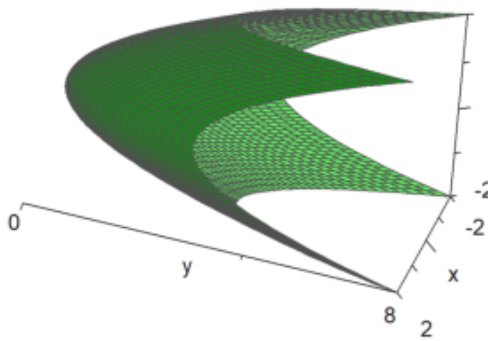
divisualisasikan dalam rentang tertentu pada sumbu x dan y, dengan sudut pandang diputar dan diperbesar agar lebih jelas terlihat. max(...) digunakan agar nilai di dalam akar tidak menjadi negatif. Jika hasil 36-9x^2-4y^2 < 0, maka akan diganti dengan 0 sehingga grafik tetap dapat digambar.

```
>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=2,frame=3):
```



Plot Kontur

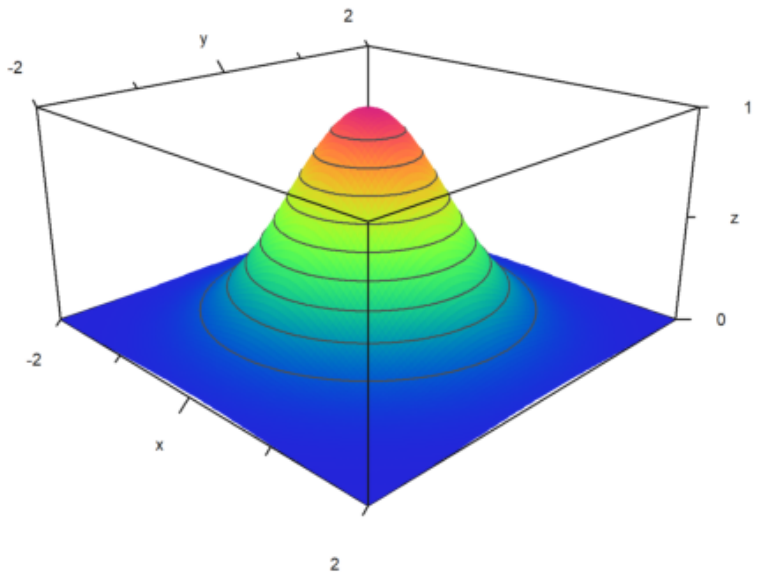
Untuk plot, Euler biasanya menambahkan garis grid. Namun, sebagai gantinya dimungkinkan untuk menggunakan garis level (level lines) serta pewarnaan satu warna (one-color hue) atau pewarnaan spektral. Euler juga dapat menggambarkan ketinggian fungsi pada plot dengan shading. Pada semua plot 3D, Euler dapat menghasilkan anaglyph merah/sian.

- >hue: menyalakan shading cahaya sebagai pengganti garis kawat (wire).
- >contour: menggambar garis kontur otomatis pada plot.
- level=... (atau levels): sebuah vektor nilai untuk garis kontur.

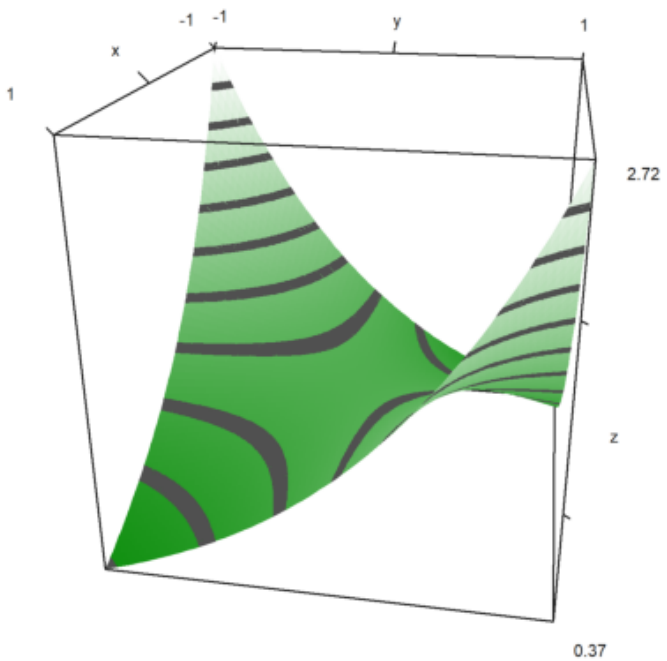
Secara default, level="auto", yang secara otomatis menghitung beberapa garis kontur. Seperti terlihat pada plot, level ini sebenarnya berupa rentang level.

Gaya default dapat diubah. Untuk plot kontur berikut, kita menggunakan grid yang lebih rapat (100x100 titik), melakukan penskalaan pada fungsi dan plot, serta mengubah sudut pandang.

```
>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):
```

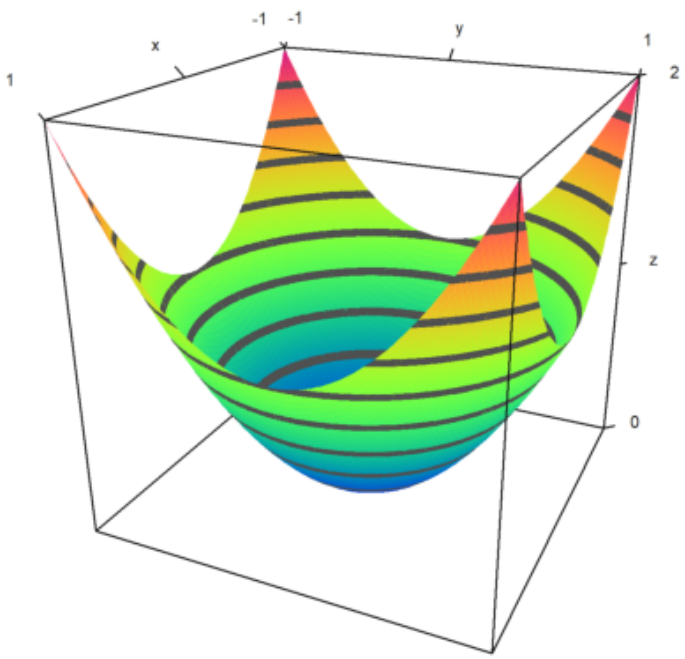


Shading (arsiran) default menggunakan warna abu-abu. Namun, rentang warna spektral juga tersedia.

- >spectral: menggunakan skema spektral default.
- color=...: menggunakan warna khusus atau skema spektral tertentu.

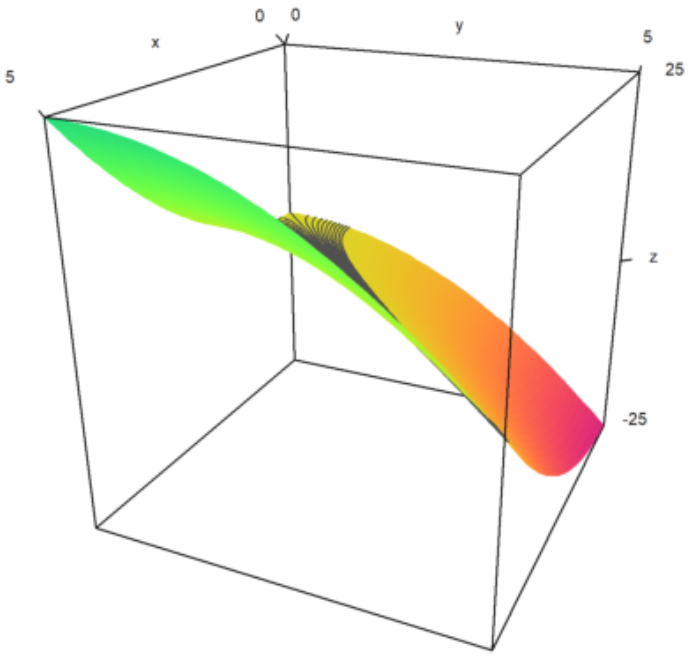
Untuk plot berikut, kita menggunakan skema spektral default dan menambah jumlah titik agar tampilannya menjadi sangat halus.

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



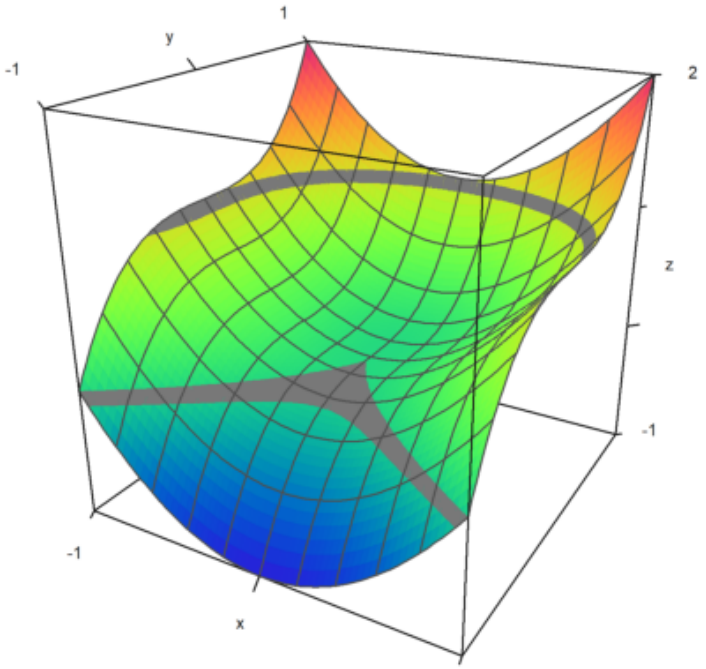
Alih-alih menggunakan garis level otomatis, kita juga dapat menetapkan sendiri nilai-nilai untuk garis level. Hal ini akan menghasilkan garis level tipis, bukan rentang level.

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



Dalam plot berikut, kita menggunakan dua pita level (level bands) yang sangat lebar, yaitu dari -0.1 hingga 1, dan dari 0.9 hingga 1. Hal ini dimasukkan sebagai sebuah matriks dengan batas level sebagai kolom.
Selain itu, kita menambahkan grid dengan 10 interval pada setiap arah.

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

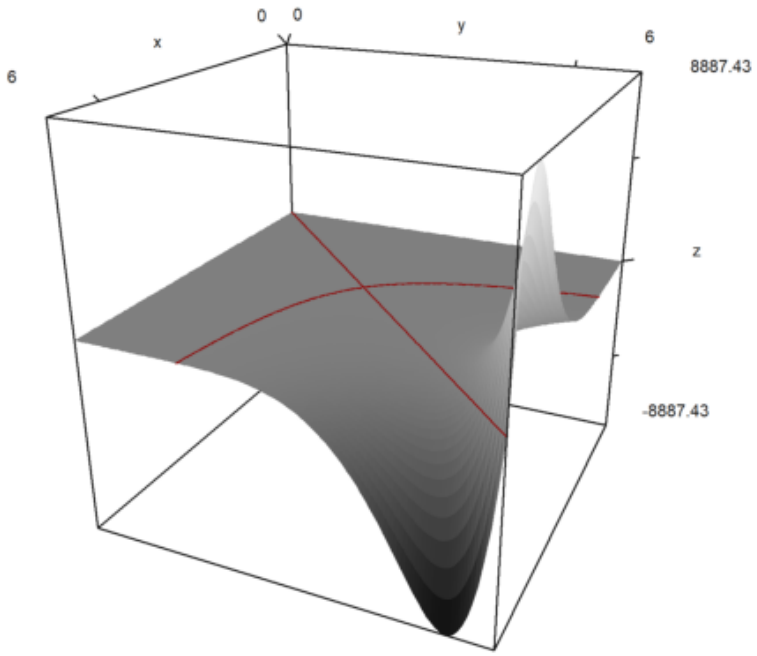


Dalam contoh berikut, kita memplot himpunan, di mana

$$f(x,y) = x^y - y^x = 0$$

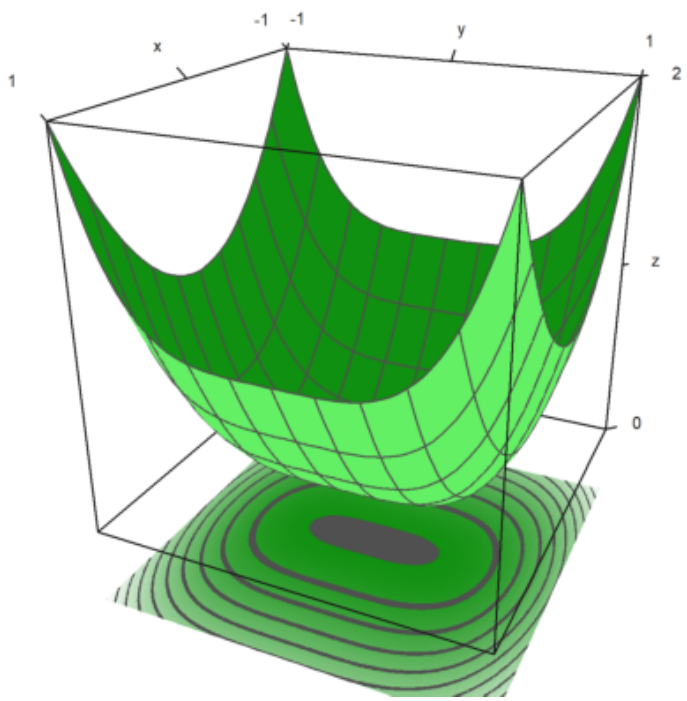
Kita menggunakan satu garis tipis sebagai garis level (level line).

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



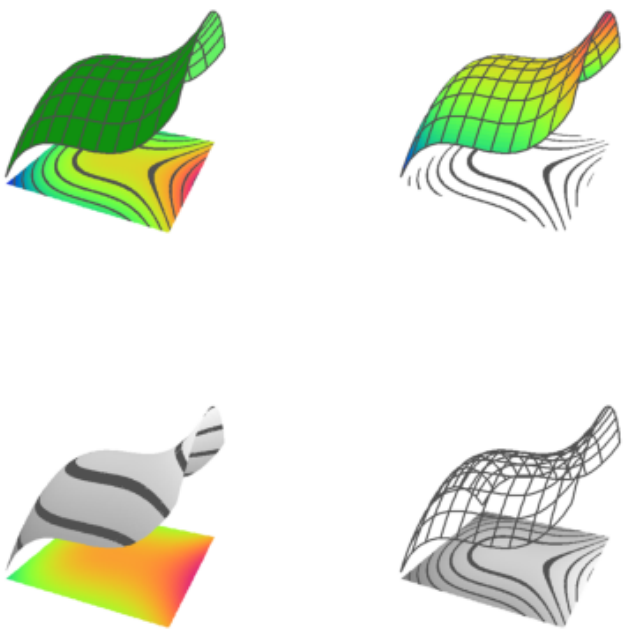
Dimungkinkan untuk menampilkan bidang kontur (contour plane) di bawah plot. Warna serta jaraknya terhadap plot dapat ditentukan.

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



Berikut ini beberapa gaya tambahan. Pada semua contoh, bingkai (frame) dimatikan, dan digunakan berbagai skema warna untuk plot serta grid.

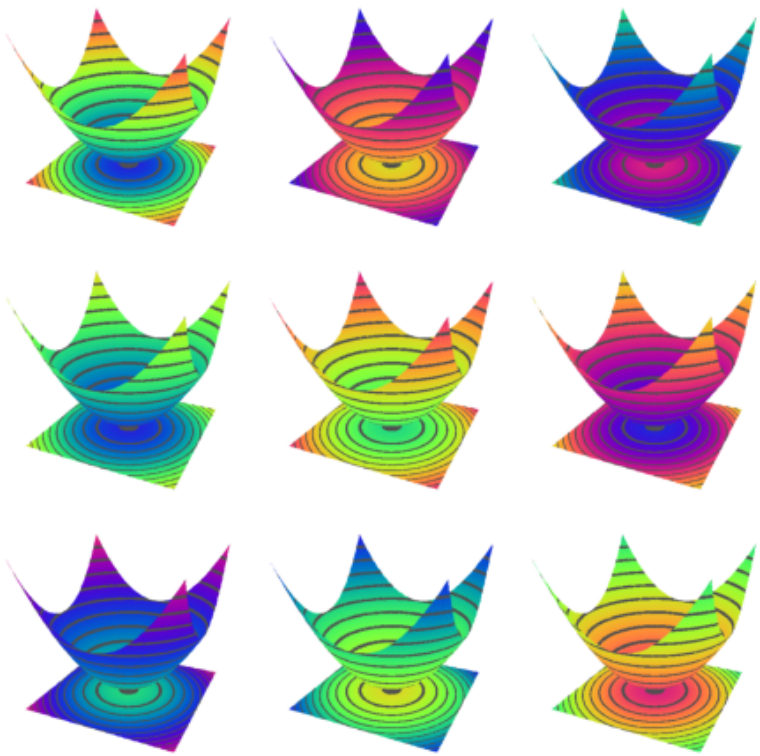
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1);  ...
>  plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2);  ...
>  plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3);  ...
>  plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4);  ...
>  plot3d(expr,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



Ada beberapa skema spektral lain, diberi nomor dari 1 sampai 9. Tetapi Anda juga bisa menggunakan color=value, di mana value dapat berupa:

- spectral: untuk rentang dari biru hingga merah
- white: untuk rentang yang lebih redup
- 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):
```

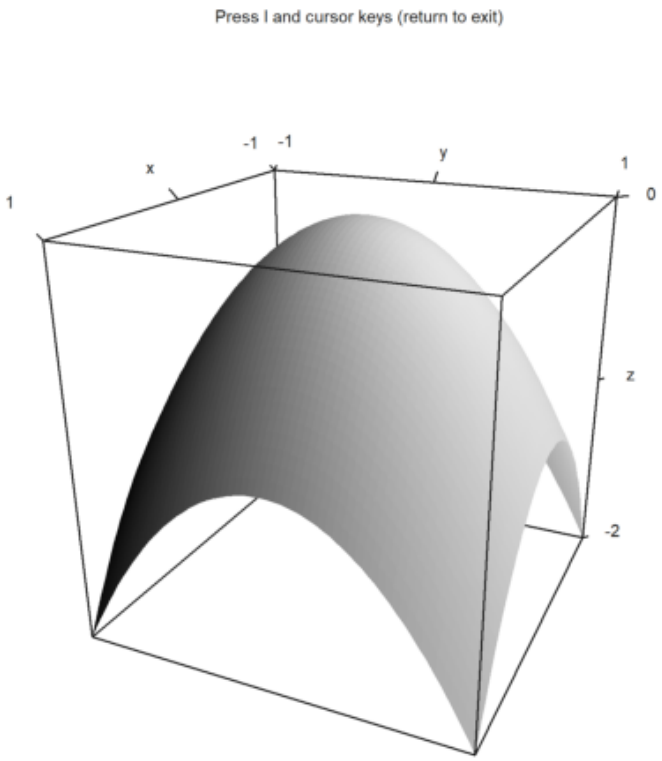


Sumber cahaya dapat diubah dengan l dan tombol panah saat interaksi pengguna. Itu juga bisa diatur dengan parameter.

- light: arah untuk cahaya
- amb: cahaya ambient antara 0 dan 1

Perlu dicatat bahwa program ini tidak membedakan sisi-sisi plot. Tidak ada bayangan. Untuk mendapatkan bayangan, 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)"):
```

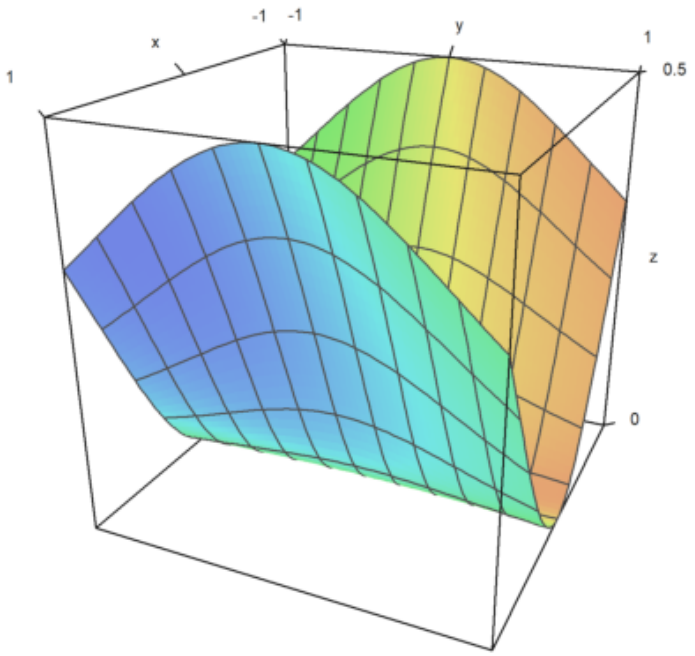


Parameter color digunakan untuk mengubah warna permukaan. Selain itu, warna level lines (garis kontur) juga bisa diubah sesuai kebutuhan.

```
>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):
```

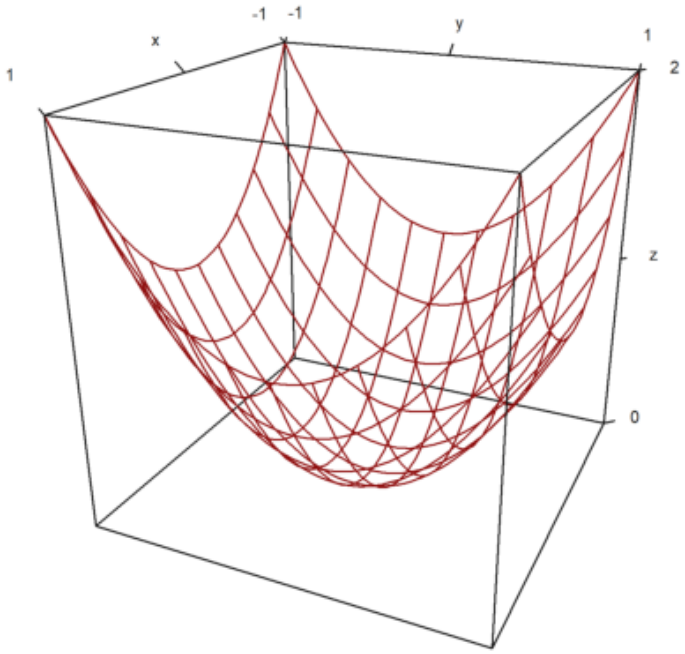
Warna 0 memberikan efek khusus berupa pelangi (rainbow effect).

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



Permukaan juga dapat dibuat transparan.

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



Plot Implisit

Selain plot eksplisit, tersedia juga plot implisit dalam tiga dimensi. Euler akan menghasilkan potongan (cuts) melalui objek. Fitur plot3d mendukung plot implisit, yang menampilkan himpunan nol (zero set) dari suatu fungsi dalam tiga variabel.
Dengan kata lain, solusi dari

$$f(x,y,z)=0$$

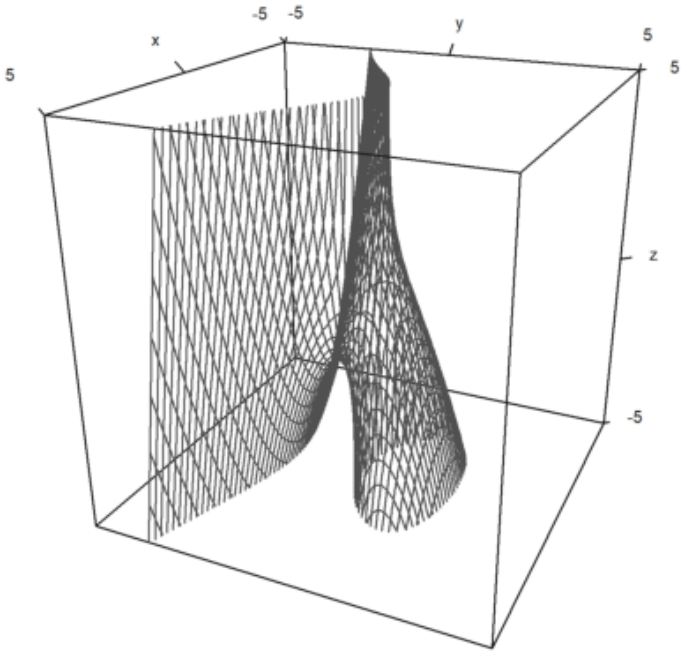
dapat divisualisasikan dalam bentuk potongan yang sejajar dengan bidang koordinat:

- implicit=1 ? potongan sejajar dengan bidang y-z
- implicit=2 ? potongan sejajar dengan bidang x-z
- implicit=4 ? potongan sejajar dengan bidang x-y

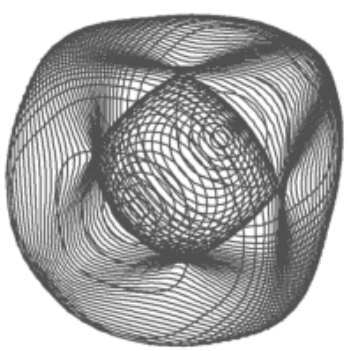
Nilai-nilai tersebut bisa dijumlahkan untuk menampilkan beberapa potongan sekaligus.

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

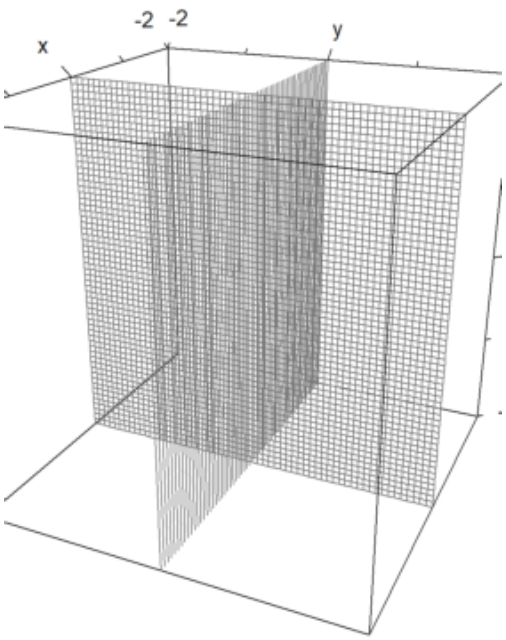
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```



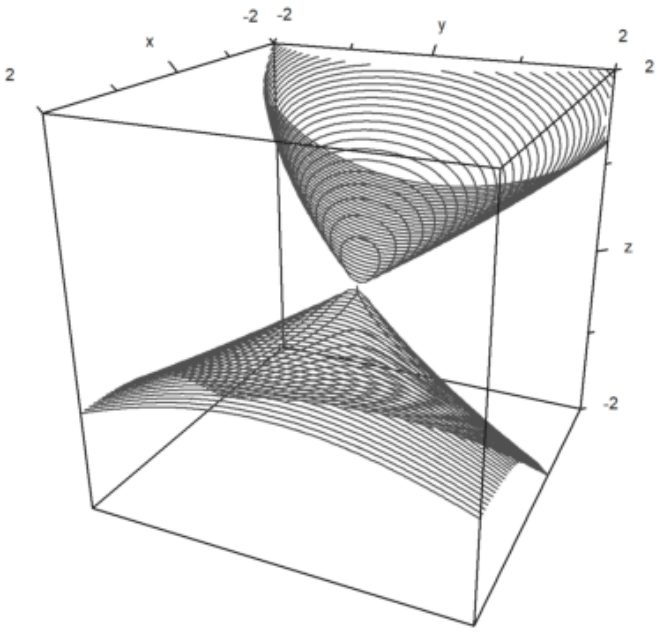

```
>c=1; d=1;
>plot3d("((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^
```



```
>plot3d("2*y^2*sin(2x)",>implicit,r=2,zoom=2):
```




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



Plotting Data 3D

Sama seperti plot2d, perintah plot3d juga dapat menerima data. Untuk objek 3D, kita perlu menyediakan matriks nilai x, y, dan z, atau tiga fungsi/ekspresi fx(x,y), fy(x,y), fz(x,y).

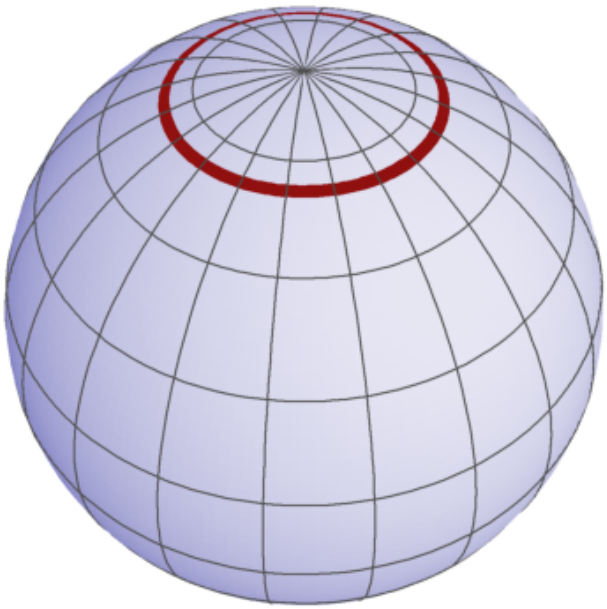
$$\gamma(t,s) = (x(t,s), y(t,s), z(t,s))$$

Karena
x,y,z berbentuk matriks, maka kita anggap pasangan (t,s) berjalan pada grid berbentuk persegi. Hasilnya, kita dapat memplot citra berupa persegi panjang di ruang 3D.

Dengan bahasa matriks di Euler, koordinat dapat dihasilkan dengan sangat efektif.

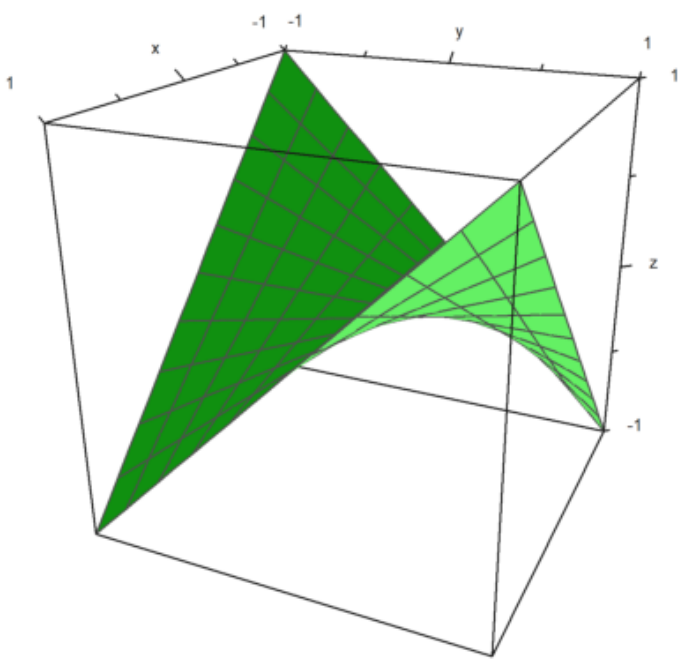
Contoh Kita ambil sebuah vektor nilai, Kita ambil vektor kolom nilai, Dengan kombinasi keduanya, kita bisa memparametrisasi permukaan bola. Pada gambar, kita juga bisa menandai daerah tertentu, misalnya daerah kutub (polar region) dari bola tersebut.

```
>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.

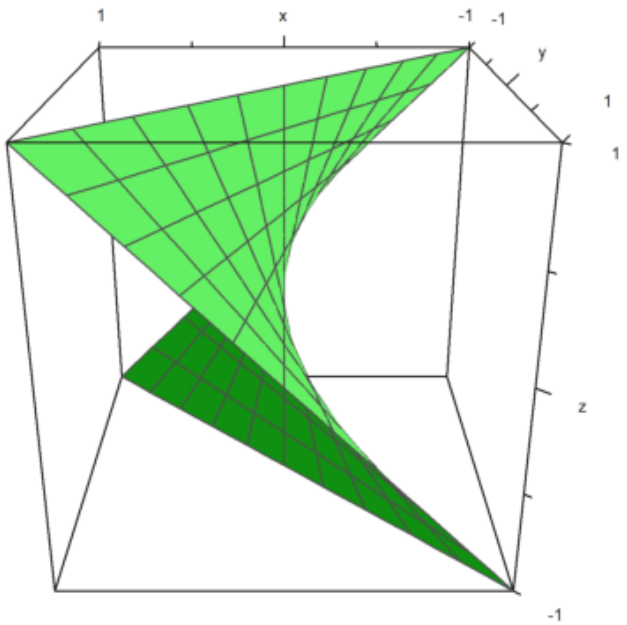
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



Namun, kita bisa membuat berbagai macam permukaan. Berikut adalah permukaan yang sama dalam bentuk fungsi

$$x = y\,z$$

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



Dengan sedikit usaha lebih, kita dapat menghasilkan banyak permukaan.

Pada contoh berikut, kita membuat tampilan berbayang dari sebuah bola yang terdistorsi. Koordinat bola biasanya dinyatakan sebagai

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

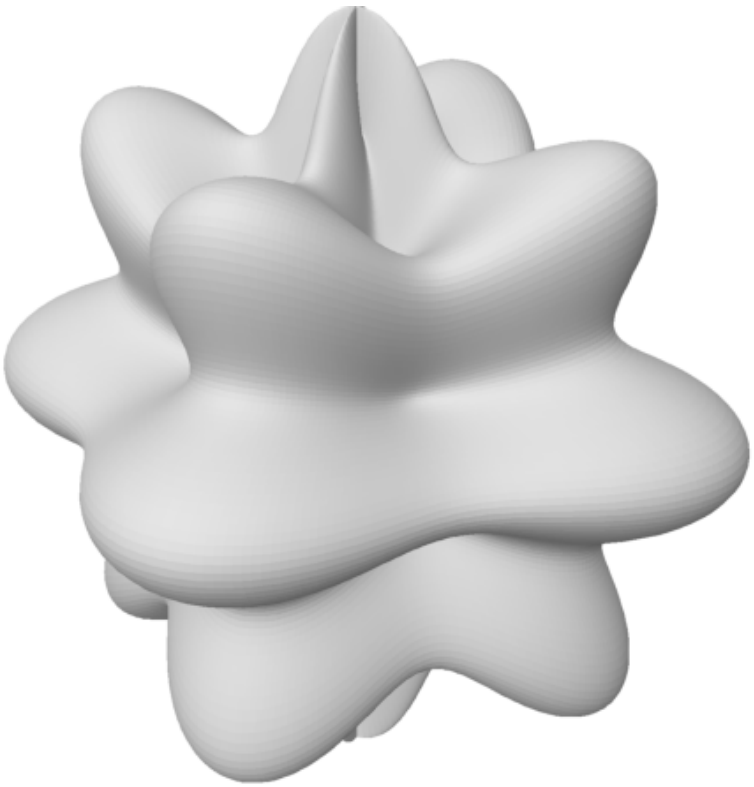
dengan

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

Kemudian, kita mendistorsi permukaan ini dengan faktor

$$d(t,s) = \frac{\cos(4t)+\cos(8s)}{4}.$$

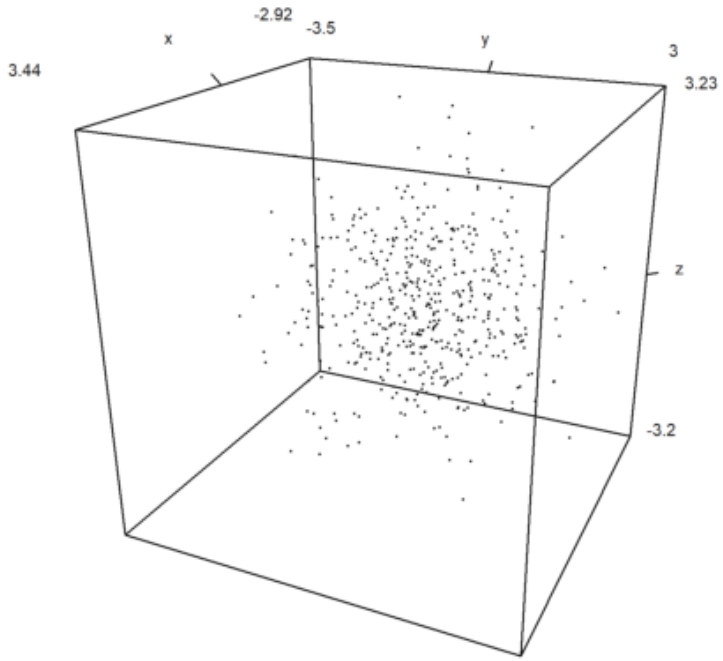
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
>  light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, plot titik (point cloud) juga dimungkinkan. Untuk menggambar data titik dalam ruang 3D, kita membutuhkan tiga vektor yang mewakili koordinat titik-titik tersebut, yaitu vektor untuk koordinat x, vektor untuk koordinat y, vektor untuk koordinat z.

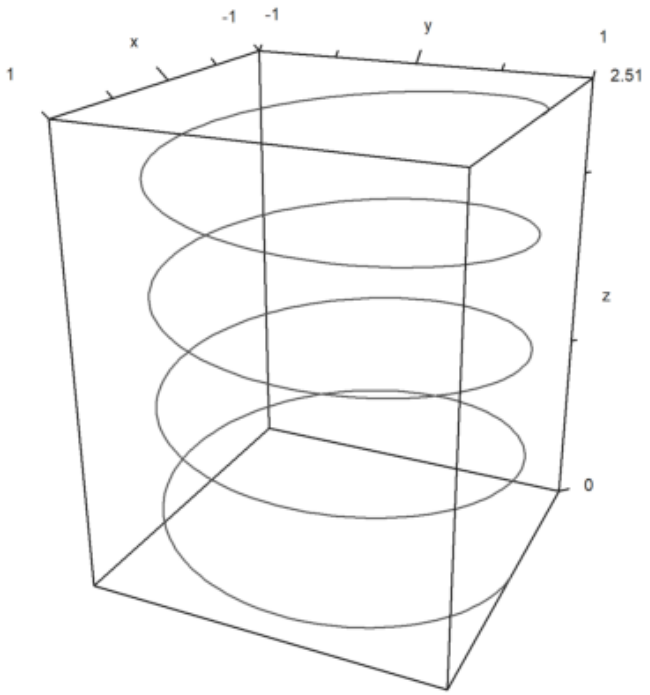
Gaya (style) plot sama seperti pada plot2d dengan parameter points=true.

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

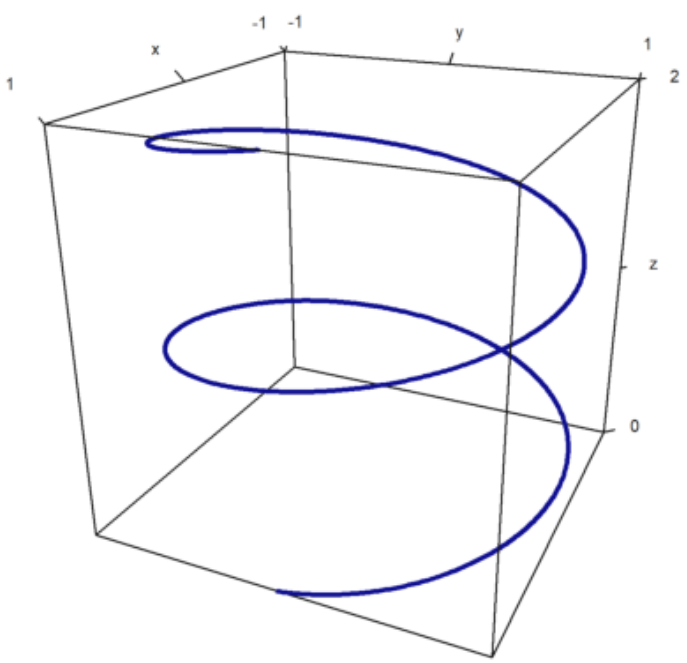


Kita juga bisa menggambar kurva dalam 3D. Dalam kasus ini, lebih mudah jika titik-titik kurva dihitung terlebih dahulu.
ntuk kurva pada bidang, kita menggunakan urutan koordinat dan menambahkan parameter wire=true.

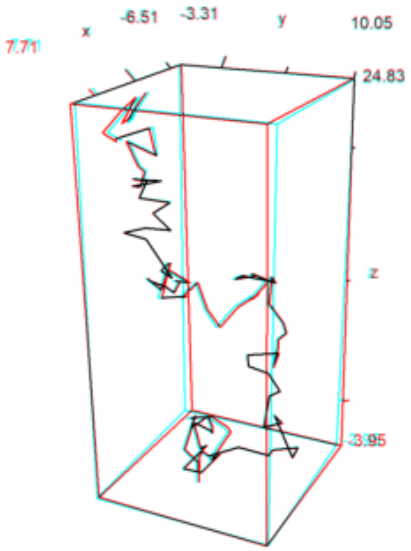
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

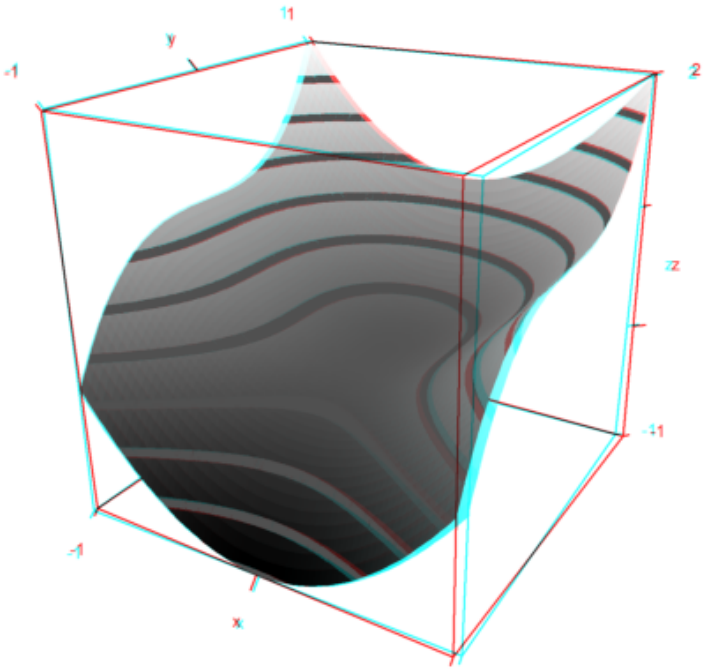


```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



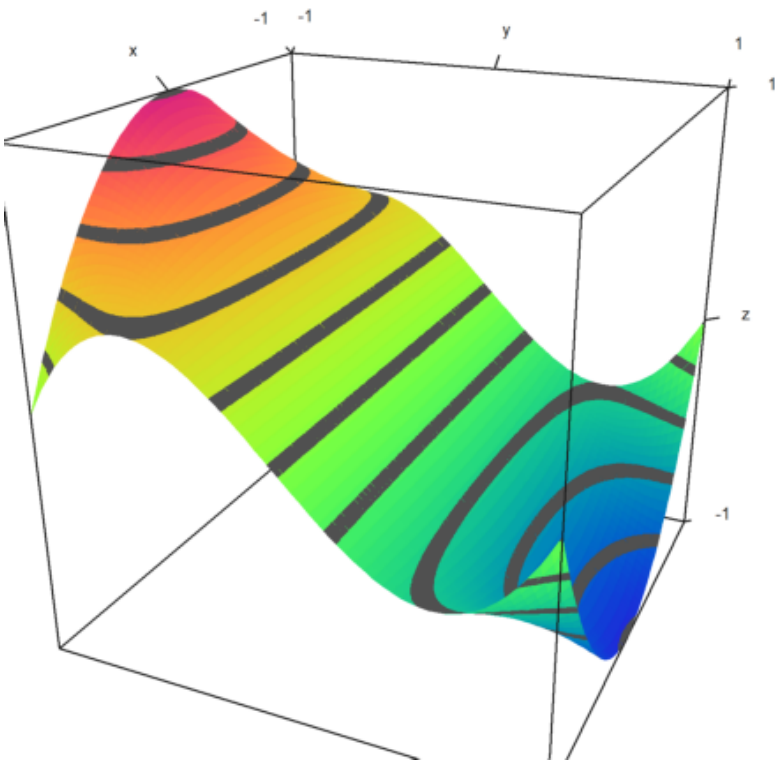
EMT juga dapat melakukan plot dalam mode anaglyph. Untuk melihat plot jenis ini, Anda memerlukan kacamata merah/sian (red/cyan glasses).

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



Sering kali, skema warna spektral digunakan untuk plot. Hal ini berfungsi untuk menekankan tinggi/rendahnya nilai fungsi dalam visualisasi.

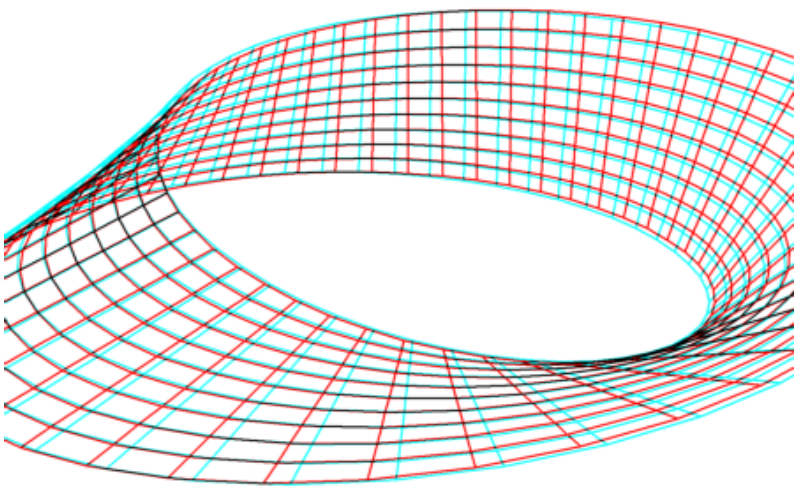
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler juga bisa memplot permukaan terparameterisasi, ketika parameter yang digunakan adalah nilai x-, y-, dan z- dari citra suatu grid persegi panjang dalam ruang.

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

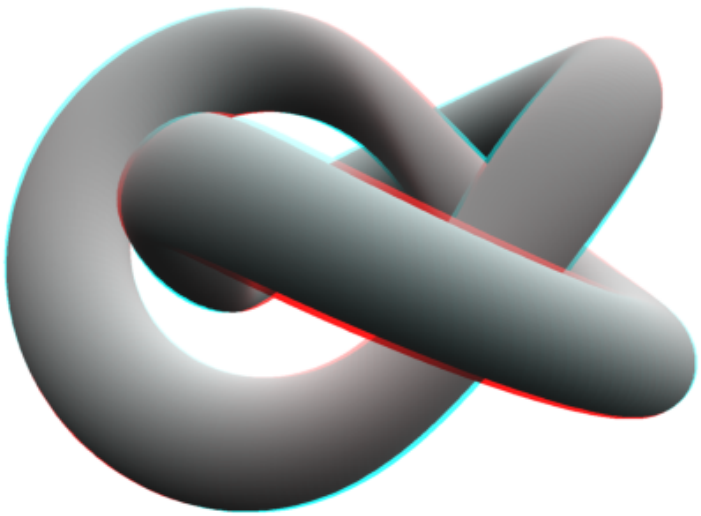
```
>u=linspace(-1,1,10); v=linspace(0,2*pi,50)'; ...
>X=(3+u*cos(v/2))*cos(v); Y=(3+u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3):
```



Berikut ini adalah contoh yang lebih rumit, yang akan tampak spektakuler jika dilihat dengan kacamata merah/sian (red/cyan glasses).

Artinya, plot tersebut menggunakan mode anaglyph 3D sehingga menghasilkan efek kedalaman (depth) pada gambar, dan bisa memberikan pengalaman visual yang lebih nyata dibandingkan plot 3D biasa.

```
>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 Statistik

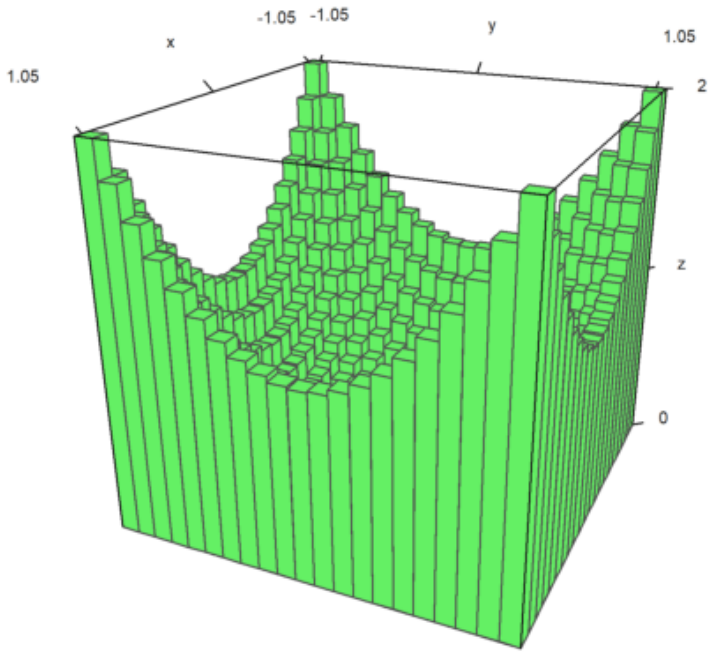
Plot statistik seperti diagram batang (bar plots) juga dapat dibuat di Euler. Untuk membuatnya, kita perlu menyediakan:

- x: vektor baris dengan n+1 elemen,
- y: vektor kolom dengan n+1 elemen,
- z: matriks n×n berisi nilai-nilai.

Jika z lebih besar, hanya nilai n×n yang akan dipakai.

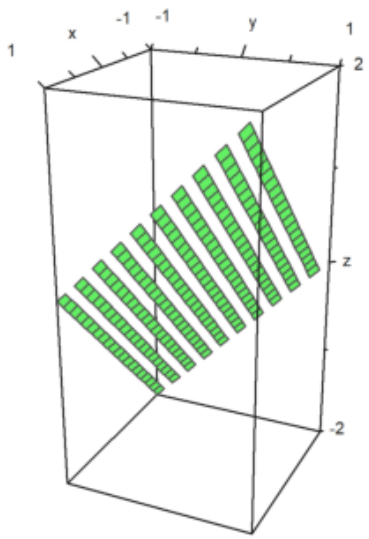
Dalam contoh, pertama kita menghitung nilai-nilai yang akan diplot. Kemudian kita menyesuaikan x dan y, sehingga vektor-vektor tersebut berpusat pada nilai yang digunakan.

```
>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):
```



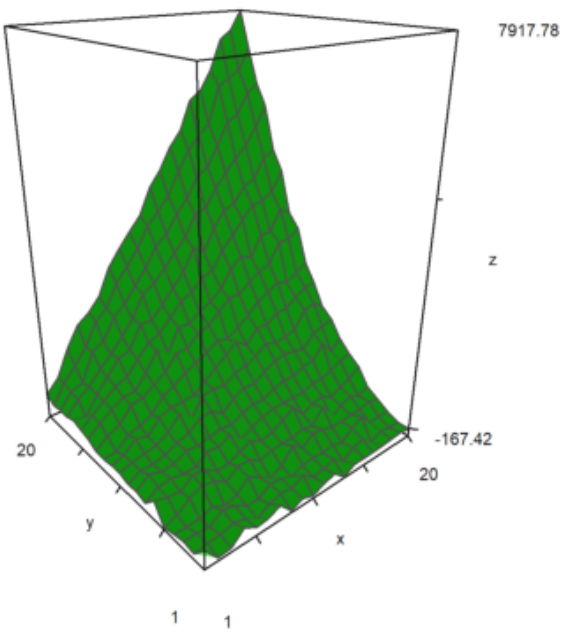
Adalah mungkin untuk membagi plot suatu permukaan menjadi dua atau lebih bagian.

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

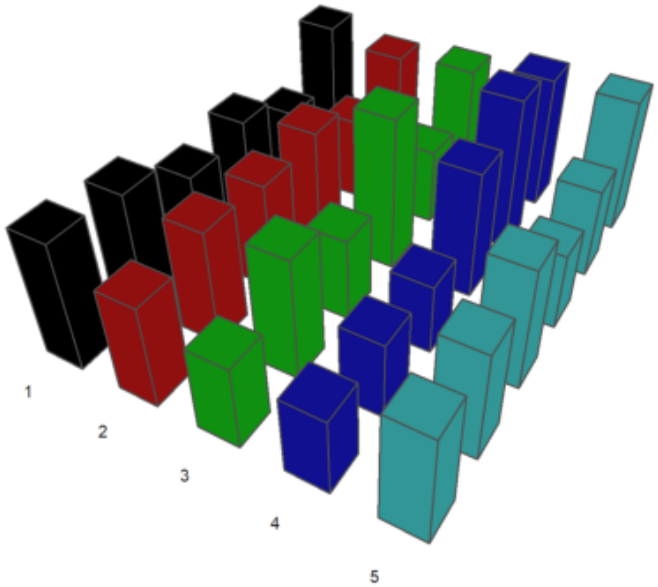


Jika Anda memuat atau menghasilkan sebuah matriks data M dari sebuah file dan perlu memplotnya dalam bentuk 3D, Anda dapat: men-skala matriks ke rentang [-1,1] dengan perintah scale(M), atau men-skala matriks dengan menggunakan parameter >zscale. Hal ini juga bisa dikombinasikan dengan faktor skala individual yang diterapkan secara tambahan.

```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8):
```

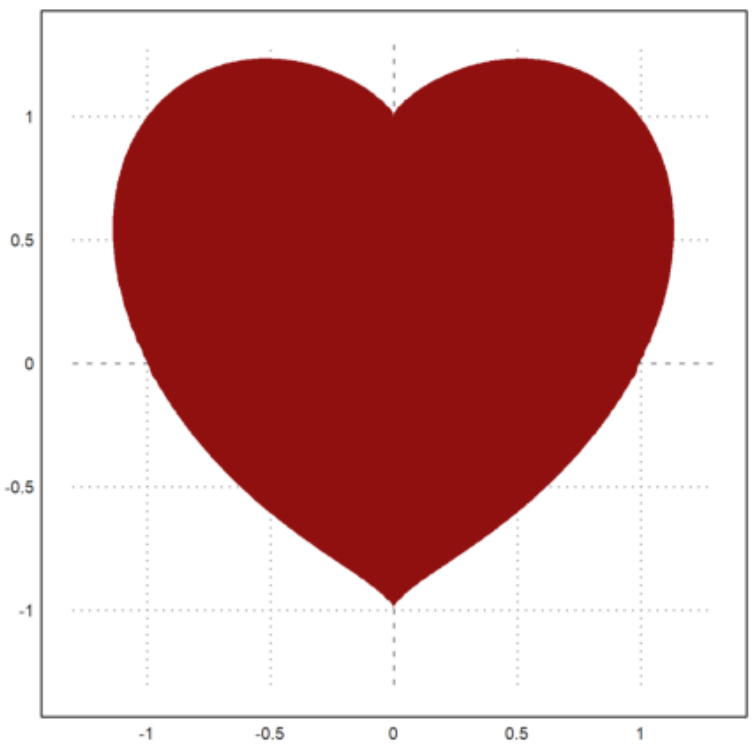



```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



Permukaan Benda Putar

```
>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
```

$$\left(y^2+x^2-1\right)^3-x^2\,y^3$$

Kita ingin memutar kurva hati (heart curve) mengelilingi sumbu-y. Berikut adalah ekspresi yang mendefinisikan kurva hati:

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

Selanjutnya kita tetapkan:

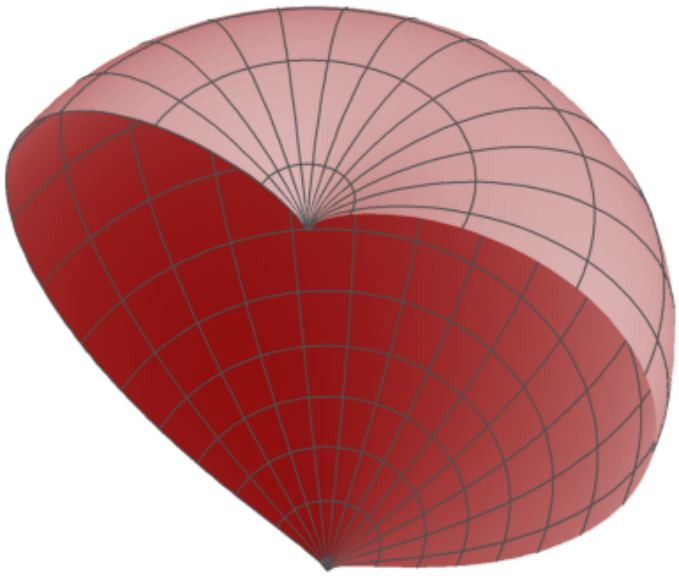
$$x=r.cos(a),\quad y=r.sin(a).$$

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

$$\left(r^2-1\right)^3+\frac{\left(\sin\left(5\,a\right)-\sin\left(3\,a\right)-2\,\sin a\right)\,r^5}{16}$$

Hal ini memungkinkan kita untuk mendefinisikan sebuah fungsi numerik, yang menyelesaikan nilai r jika nilai a sudah diberikan. Dengan fungsi tersebut, kita kemudian dapat memplot bentuk hati yang diputar sebagai sebuah permukaan parametrik.

```
>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°):
```

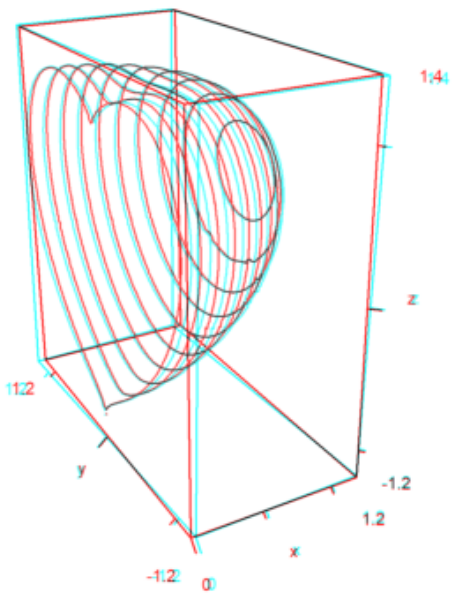


Berikut ini adalah plot 3D dari gambar hati di atas yang telah diputar mengelilingi sumbu-z. Kita mendefinisikan sebuah fungsi yang menggambarkan objek tersebut, lalu memvisualisasikannya dalam bentuk permukaan parametrik.

```
>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 sangat berguna, tetapi tidak selalu dapat memenuhi semua kebutuhan. Selain rutin dasar, dimungkinkan juga untuk mendapatkan plot dengan bingkai (framed plot) dari objek apa pun yang diinginkan.

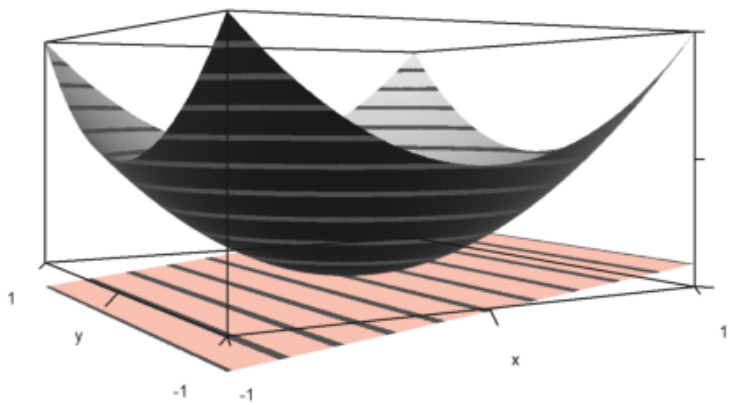
Meskipun Euler bukanlah program khusus 3D, namun Euler dapat mengombinasikan beberapa objek dasar. Sebagai contoh, kita bisa mencoba memvisualisasikan sebuah paraboloid beserta bidang singgungnya (tangent).

```
>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
```

Sekarang, fungsi framedplot() digunakan untuk menyediakan bingkai (frames) serta mengatur sudut pandang (views) dari plot tersebut.

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

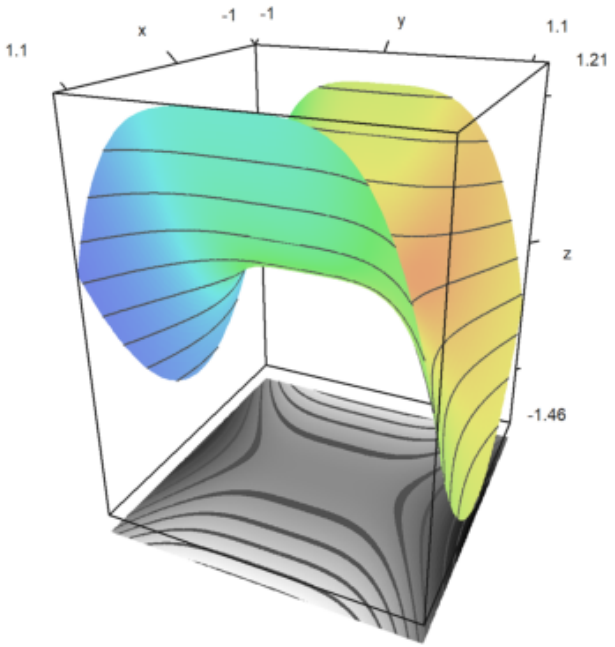


Dengan cara yang sama, Anda juga bisa memplot bidang kontur (contour plane) secara manual. Perlu dicatat bahwa plot3d() secara bawaan (default) akan mengatur jendela ke fullwindow(), sedangkan plotcontourplane() mengasumsikan pengaturan tersebut.

```
>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):
```



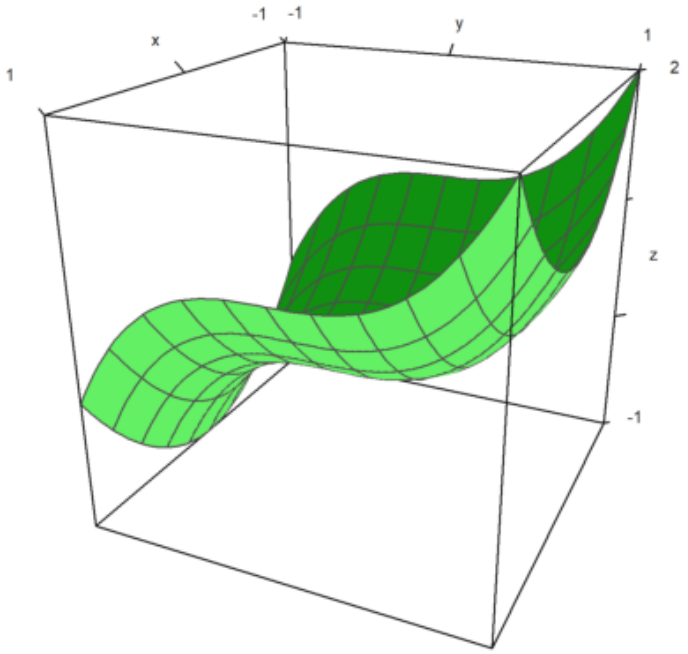
Animasi

Euler dapat menggunakan frame untuk menghitung terlebih dahulu (pre-compute) sebuah animasi.

Salah satu fungsi yang memanfaatkan teknik ini adalah rotate. Fungsi ini dapat mengubah sudut pandang dan menggambar ulang sebuah plot 3D. Setiap kali membuat plot baru, fungsi ini akan memanggil addpage(). Akhirnya, rangkaian plot tersebut dianimasikan.

Untuk lebih detail, Anda dapat mempelajari kode sumber (source) dari fungsi rotate.

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



Menggambar Povray

Dengan bantuan file Euler povray.e, Euler dapat menghasilkan file Povray. Hasilnya terlihat sangat menarik.
Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/

```
dan menambahkan sub-direktori "bin" dari Povray ke environment path,
```

atau mengatur variabel defaultpovray dengan path lengkap yang menunjuk ke pvengine.exe.

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

Fungsi pov3d memiliki konsep yang sama seperti plot3d. Fungsi ini dapat menghasilkan grafik dari fungsi f(x,y), atau 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(), ada banyak fungsi lain yang menghasilkan objek Povray. Fungsi-fungsi ini mengembalikan string yang berisi kode Povray untuk objek tersebut. Untuk menggunakan fungsi-fungsi ini, mulailah file Povray dengan povstart(). Kemudian gunakan writeln(...) untuk menulis objek ke file scene. Akhiri file dengan povend(). Secara default, raytracer akan dijalankan, dan PNG akan dimasukkan ke dalam notebook Euler.

Fungsi objek memiliki parameter bernama look, yang membutuhkan string berisi kode Povray untuk tekstur dan finish objek. Fungsi povlook() dapat digunakan untuk membuat string ini. Fungsi ini memiliki parameter untuk warna, transparansi, Phong Shading, dll.

Perlu dicatat bahwa sistem koordinat Povray berbeda. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray. Jadi Anda tetap bisa berpikir dalam sistem koordinat Euler, dengan sumbu z menunjuk ke atas, dan sumbu x, y, z mengikuti aturan tangan kanan.
Anda perlu memuat file Povray untuk memulai.

```
>load povray;
```

Pastikan direktori bin Povray sudah ada di path. Jika belum, edit variabel berikut agar berisi path lengkap menuju file eksekusi Povray.

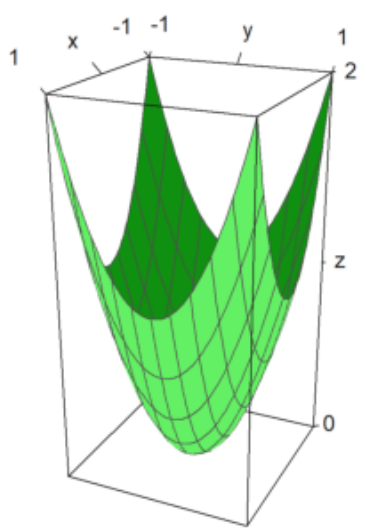
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

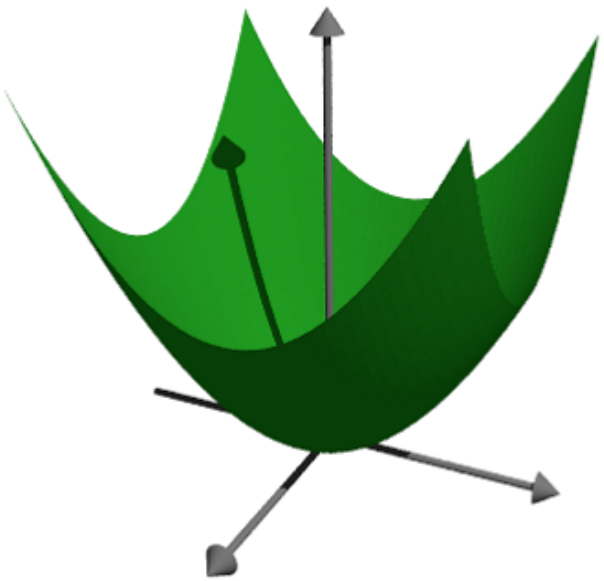
Untuk percobaan awal, kita akan memplot sebuah fungsi sederhana. Perintah berikut akan menghasilkan file Povray di direktori pengguna Anda, dan menjalankan Povray untuk ray tracing file tersebut.

Jika Anda menjalankan perintah ini, GUI Povray seharusnya terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan diminta konfirmasi apakah ingin mengizinkan file .exe dijalankan. Anda dapat menekan Cancel untuk menghentikan pertanyaan selanjutnya. Anda mungkin juga perlu menekan OK pada jendela Povray untuk mengakui dialog start-up Povray.

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

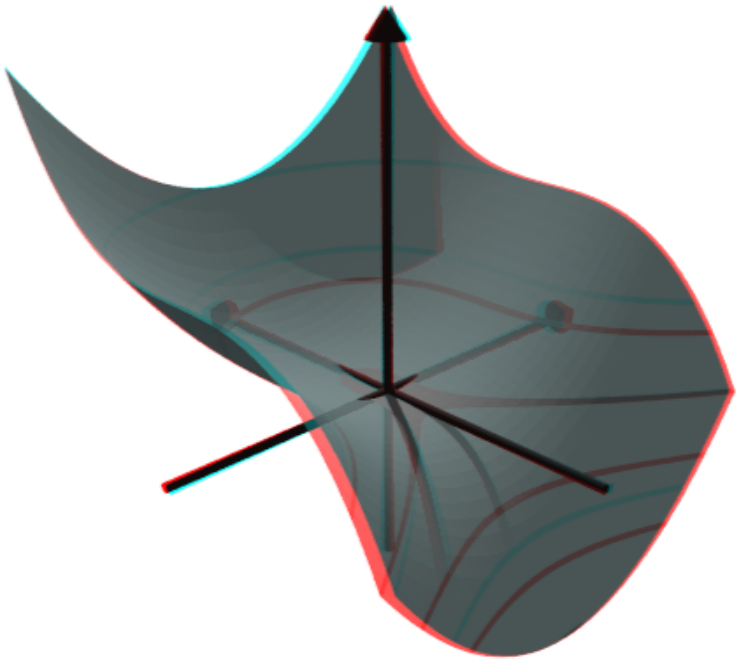


```
>pov3d("x^2+y^2",zoom=3);
```



Kita dapat membuat fungsi menjadi transparan dan menambahkan finish lain. Kita juga bisa menambahkan garis level pada plot fungsi tersebut.

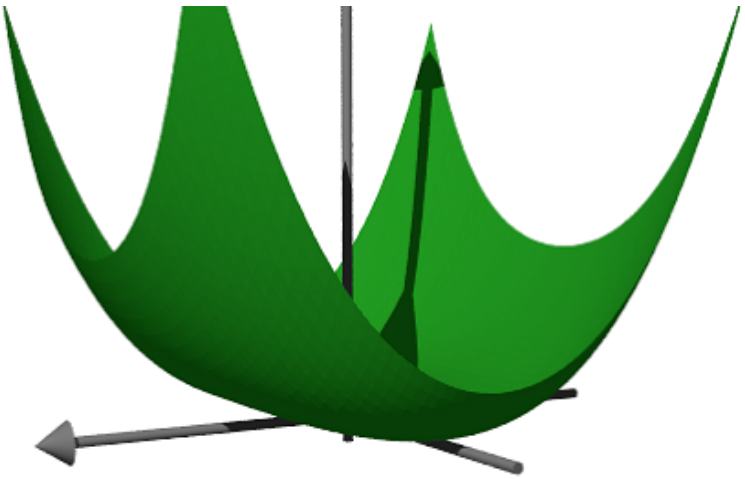
```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
>  look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



Terkadang perlu untuk mencegah penskalaan otomatis pada fungsi, dan melakukan penskalaan fungsi secara manual.

Kita akan memplot himpuan titik di bidang kompleks, di mana hasil kali jarak ke 1 dan -1 sama dengan 1.

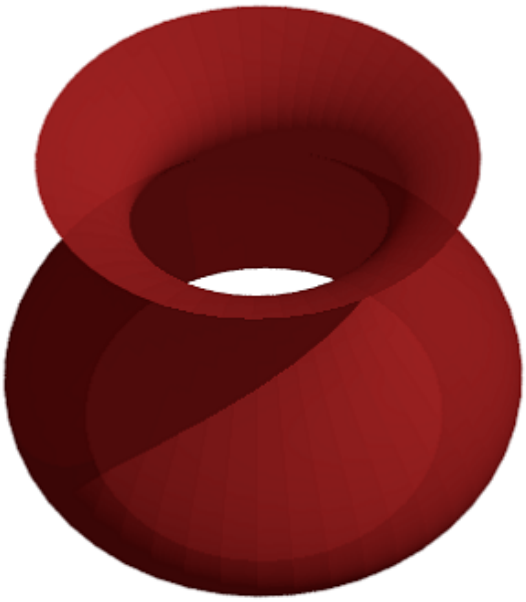
```
>pov3d("((x-1)^2+y^2)*((x+1)^2+y^2)/40",r=2, ...
>  angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=10°,n=50, ...
>  <fscale,zoom=3.8);
```



Plotting dengan Koordinat

Alih-alih menggunakan fungsi, kita bisa memplot menggunakan koordinat.
eperti pada plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek.
Dalam contoh ini, kita memutar sebuah fungsi mengelilingi sumbu z.

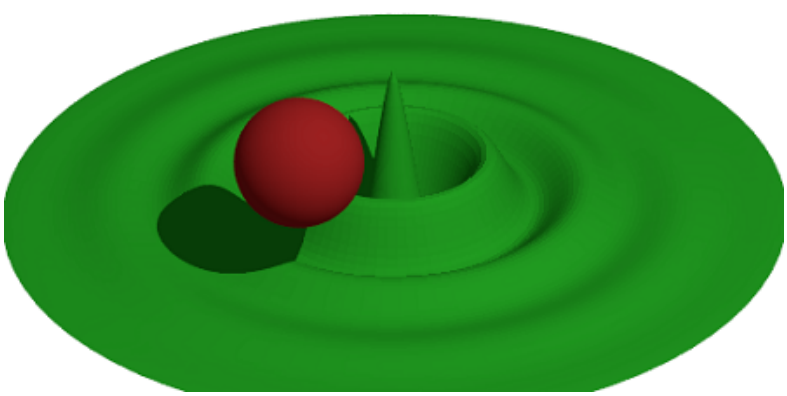
```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,axis=0,zoom=4,light=[10,5,15]);
```



Dalam contoh berikut, kita memplot gelombang yang meredam. Gelombang ini dihasilkan menggunakan bahasa matriks Euler.

Kita juga menunjukkan cara menambahkan objek tambahan ke dalam scene pov3d. Untuk pembuatan objek, lihat contoh-contoh berikut. Perlu dicatat bahwa plot3d melakukan pensskalaan otomatis, sehingga plot akan pas dengan 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);
```



Dengan metode shading lanjutan dari Povray, hanya sedikit titik saja sudah dapat menghasilkan permukaan yang sangat halus. Hanya pada batas-batas dan di bayangan trik ini mungkin terlihat jelas.

Untuk ini, kita perlu menambahkan vektor normal pada setiap titik matriks.

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

$$x^2 y^3$$

Persamaan permukaannya adalah [x,y,Z]. Kita menghitung turunan terhadap x dan y dari persamaan ini, lalu mengambil cross product sebagai vektor normal.

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

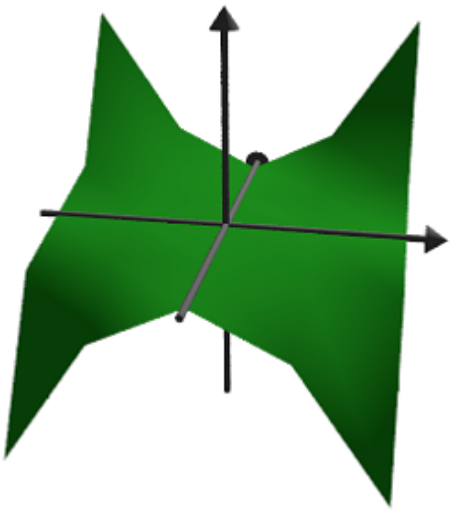
Kita mendefinisikan vektor normal sebagai cross product dari turunan-turunan tersebut, dan mendefinisikan fungsi koordinat.

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

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

Kita hanya menggunakan 25 titik saja.

```
>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);
```



Berikut adalah Trefoil Knot yang dibuat oleh A. Busser di Povray. Ada versi yang ditingkatkan dari ini dalam contoh-contoh.

See: Examples\Trefoil Knot | Trefoil Knot

Untuk tampilan yang baik tanpa terlalu banyak titik, kita menambahkan vektor normal di sini. Kita menggunakan Maxima untuk menghitung normalnya. Pertama, dibuat tiga fungsi koordinat sebagai ekspresi simbolik.

```
>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);
```

Kemudian dibuat dua vektor turunan terhadap x dan y.

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

Sekarang kita buat vektor normal, yaitu cross product dari kedua vektor turunan tersebut.

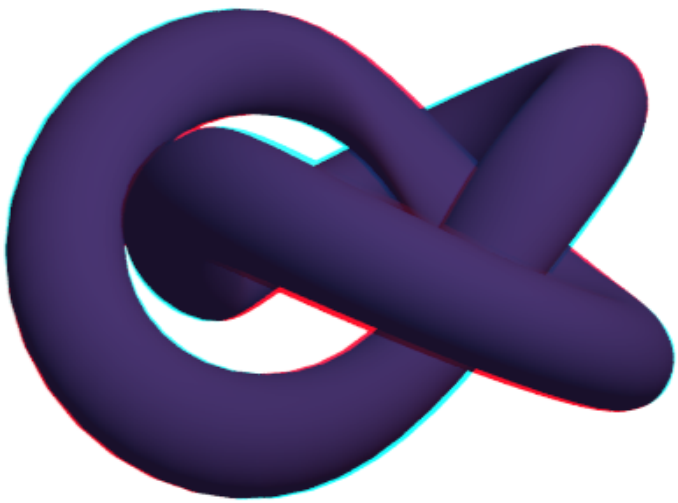
```
>dn &= crossproduct(dx,dy);
```

Sekarang kita mengevaluasi semuanya secara numerik.

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

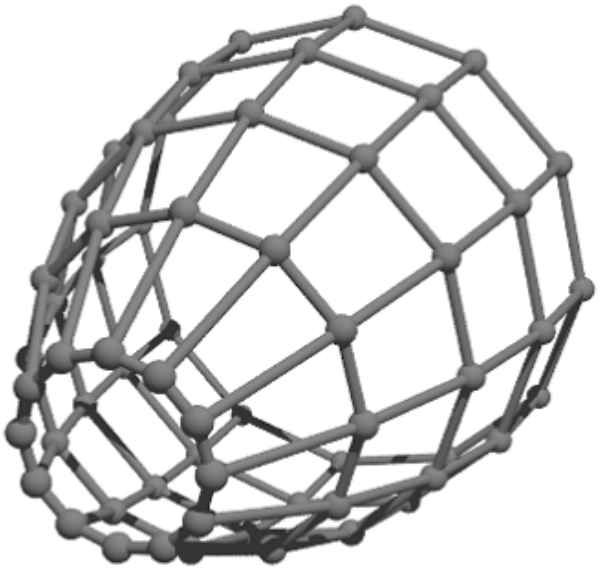
Vektor normal adalah hasil evaluasi ekspresi simbolik dn[i] untuk i=1,2,3. Sintaks untuk ini adalah &"expression"(parameters). Ini merupakan alternatif dari metode pada contoh sebelumnya, di mana kita terlebih dahulu mendefinisikan ekspresi simbolik NX, NY, NZ.

```
>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));
```



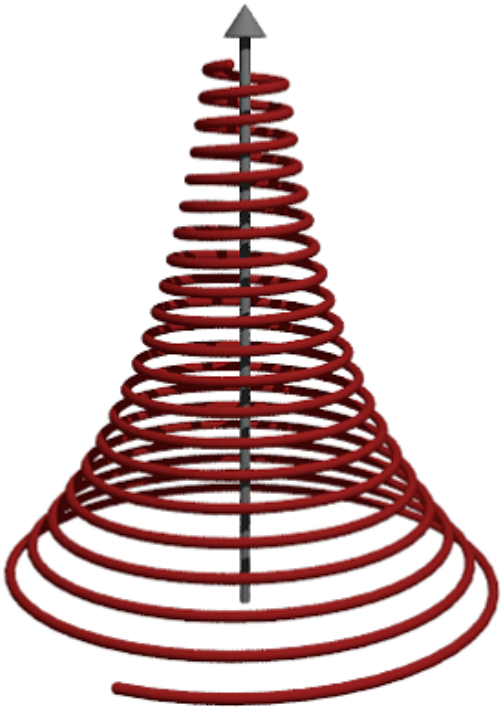
Kita juga dapat menghasilkan grid dalam 3D.

```
>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();
```



Dengan povgrid(), kurva juga dimungkinkan.

```
>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();
```



Objek Povray

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);
```

Pertama, kita mendefinisikan tiga silinder, dan menyimpannya sebagai string di Euler.
Fungsi povx() dan sejenisnya hanya mengembalikan vektor [1,0,0], yang bisa digunakan sebagai alternatif.

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

String-string tersebut berisi kode Povray, yang pada tahap ini tidak perlu kita pahami.

```
>c2
```

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

Seperti yang Anda lihat, kita menambahkan tekstur pada objek dengan tiga warna berbeda.

Hal ini dilakukan oleh povlook(), yang mengembalikan string berisi kode Povray yang relevan. Kita bisa menggunakan warna default Euler, atau mendefinisikan warna sendiri. Kita juga bisa menambahkan transparansi, atau mengubah cahaya ambient.

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

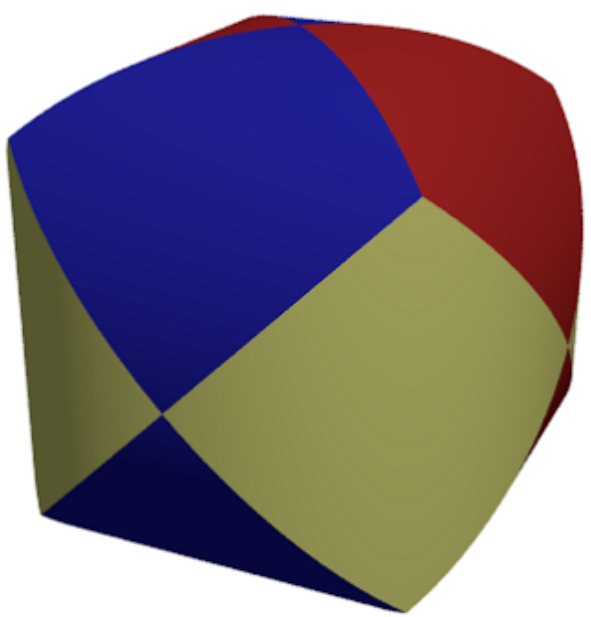
```
   texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> }  }
   finish { ambient 0.5 }
```

Sekarang kita mendefinisikan objek irisan (intersection), dan menulis hasilnya ke dalam file.

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

Irisan dari tiga silinder sulit divisualisasikan jika Anda belum pernah melihatnya sebelumnya.

```
>povend;
```



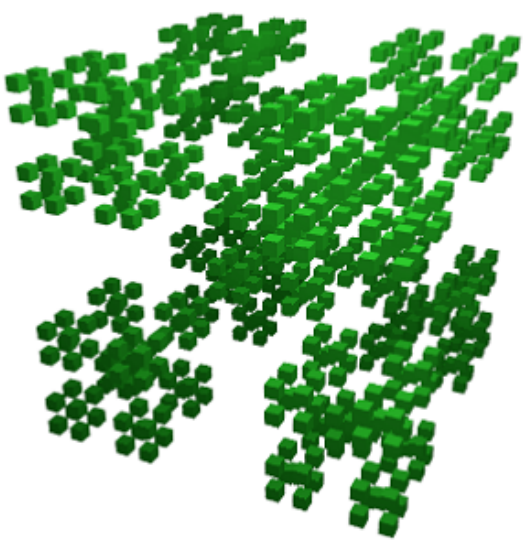
Fungsi-fungsi berikut menghasilkan fraktal secara rekursif.

Fungsi pertama menunjukkan cara Euler menangani objek Povray sederhana. Fungsi povbox() mengembalikan string yang berisi koordinat kotak, tekstur, dan 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();
```



Difference memungkinkan memotong satu objek dari objek lain. Seperti intersection, ini merupakan bagian dari CSG (Constructive Solid Geometry) objects di Povray.

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

Untuk demonstrasi ini, kita mendefinisikan objek langsung di Povray, alih-alih menggunakan string di Euler. Definisi akan langsung ditulis ke file.

Koordinat kotak -1 berarti[-1,-1,-1].

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

Kita bisa menggunakan objek ini di povobject(), yang seperti biasa akan mengembalikan string.

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

Kita membuat kubus kedua, lalu memutar dan menskalakan sedikit.

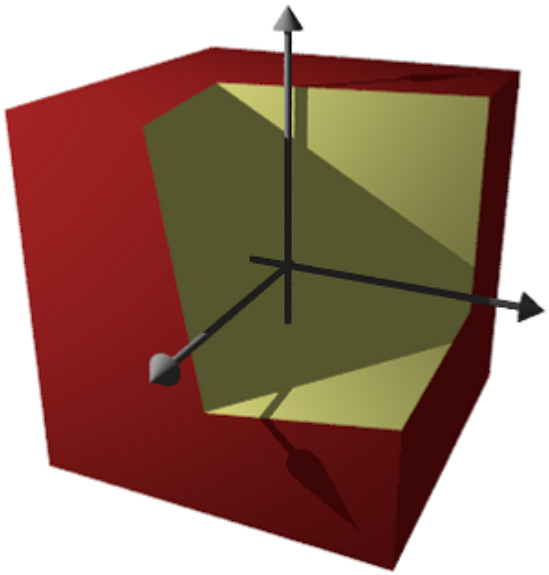
```
>c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...
>  rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Kemudian kita mengambil difference dari kedua objek tersebut.

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

Sekarang tambahkan tiga sumbu (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 titik di mana f(x,y,z)=0, sama seperti parameter implisit di plot3d. Namun, hasilnya biasanya lebih baik.

Sintaks untuk fungsi ini sedikit berbeda. Anda tidak bisa menggunakan output Maxima atau ekspresi Euler secara langsung.

$$((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^2)=d$$

```
>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)+p
>povend();
```

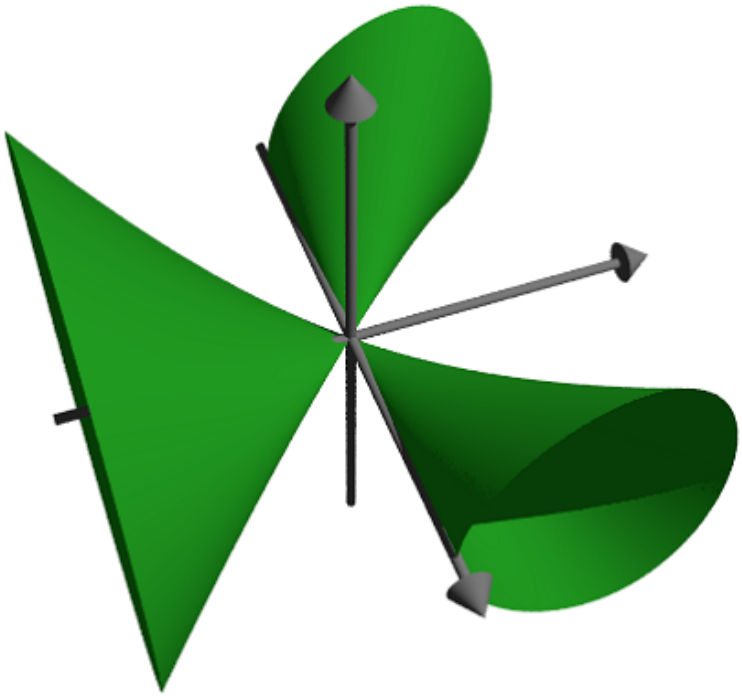
```
  Error : Povray error!

  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);
```

Buatlah permukaan implisit. Perhatikan bahwa sintaks ekspresi berbeda dari biasa.

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



Objek Mesh

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

Kita ingin memaksimalkan xy dengan kondisi x+y=1 dan mendemonstrasikan sentuhan tangensial garis level.

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

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

Berikut ini mendefinisikan objek mesh dan menulisnya langsung ke 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);
```

Sekarang kita mendefinisikan dua cakram (discs), yang akan diiris (intersected) dengan permukaan.

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

Tulis permukaan tersebut dikurangi kedua cakram.

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

Tulis kedua objek irisan (intersections) tersebut.

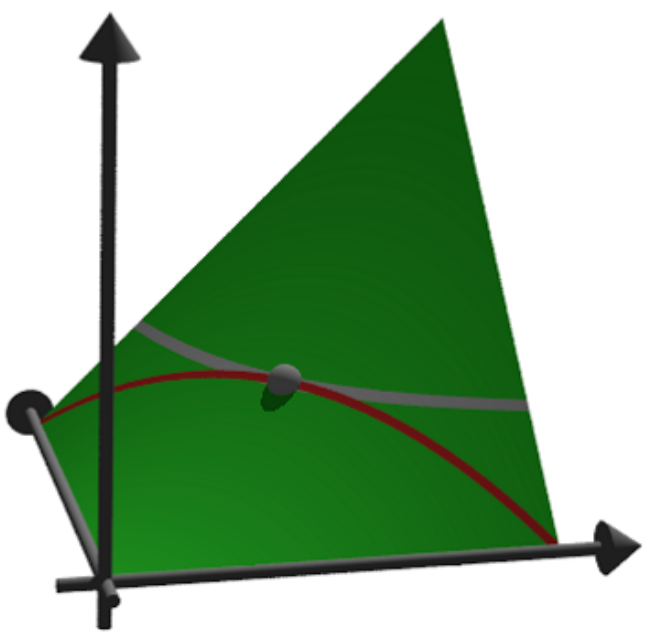
```
>writeln(povintersection([mesh,cl],povlook(red)); ...
>writeln(povintersection([mesh,ll],povlook(gray)));
```

Tulis sebuah titik pada posisi maksimum.

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

Tambahkan sumbu-sumbu (axes) dan selesaikan (finish).

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



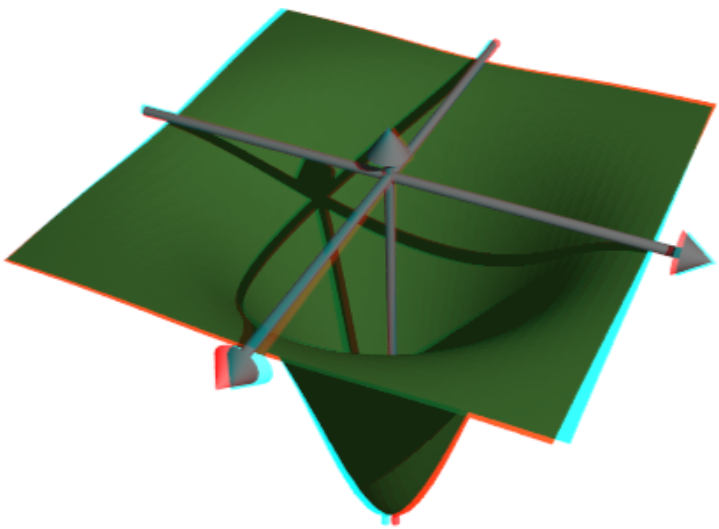
Anaglif di Povray

Untuk menghasilkan anaglif untuk kacamata merah/teal, Povray harus dijalankan dua kali dari posisi kamera berbeda. Povray akan menghasilkan dua file Povray dan dua file PNG, yang dapat dimuat menggunakan fungsi loadanaglyph().

Tentu saja, Anda membutuhkan kacamata merah/teal untuk melihat contoh berikut dengan benar.

Fungsi pov3d() memiliki switch sederhana untuk menghasilkan anaglif.

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



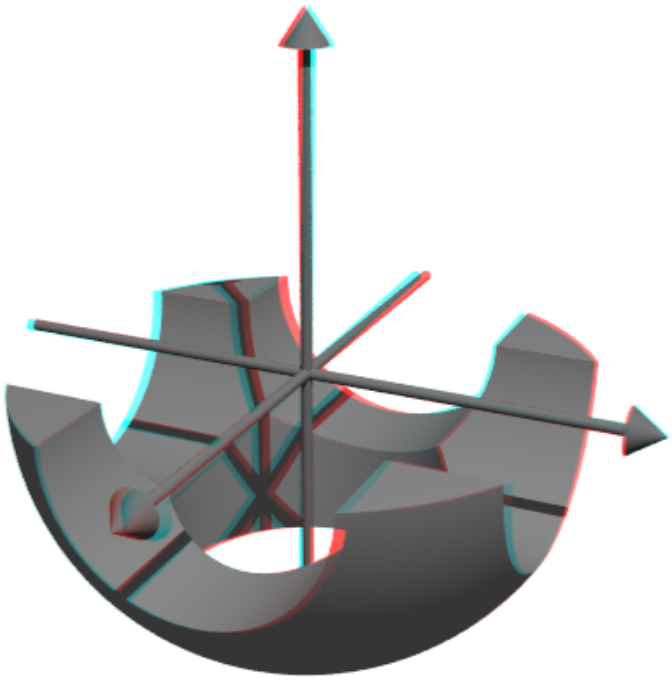
Jika Anda membuat sebuah scene dengan objek, Anda perlu memasukkan pembuatan scene ke dalam sebuah fungsi, lalu menjalankannya dua kali dengan nilai parameter anaglif yang berbeda.

```
>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
```

Fungsi povanaglyph() melakukan semua langkah ini. Parameternya sama seperti gabungan dari povstart() dan povend().

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



Mendefinisikan Objek Sendiri

Antarmuka Povray di Euler sudah menyediakan banyak objek. Namun, Anda tidak terbatas pada objek-objek tersebut. Anda bisa membuat objek sendiri, yang dapat menggabungkan objek lain atau menjadi objek baru sepenuhnya.

Contohnya kita mendemonstrasikan torus. Perintah Povray untuk ini adalah "torus". Jadi, kita mengembalikan string yang berisi perintah ini beserta parameternya. Perlu dicatat bahwa torus selalu berpusat di origin.

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

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

Berikut adalah torus pertama kita.

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

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

Mari kita gunakan objek ini untuk membuat torus kedua, yang ditranslasi dan diputar.

```
>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>
 }
```

Sekarang kita menempatkan objek-objek ini ke dalam sebuah scene. Untuk look, kita menggunakan 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();
```

akan memanggil program Povray. Namun, jika terjadi error, Povray tidak menampilkan pesan kesalahan. Oleh karena itu, sebaiknya gunakan:

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

jika ada sesuatu yang tidak berjalan dengan benar. Ini akan menjaga jendela Povray tetap terbuka.

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

Berikut adalah contoh yang lebih kompleks. Kita menyelesaikan:

$$Ax \leq b, \quad x \geq 0, \quad c.x \to \text{Max}.$$

dan menampilkan titik-titik layak (feasible points) serta titik optimum dalam plot 3D.

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

Pertama, mari kita periksa apakah contoh ini memiliki solusi sama sekali.

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

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

Ya, memiliki solusi.
Selanjutnya, kita mendefinisikan dua objek. Yang pertama adalah bidang (plane):

$$a \cdot x \leq b$$

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

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

Kemudian kita mendefinisikan irisan dari semua half-space dengan sebuah kubus.

```
>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
```

Sekarang kita dapat memplot scene tersebut.

```
>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); ...
```

Berikut adalah lingkaran di sekitar titik optimum.

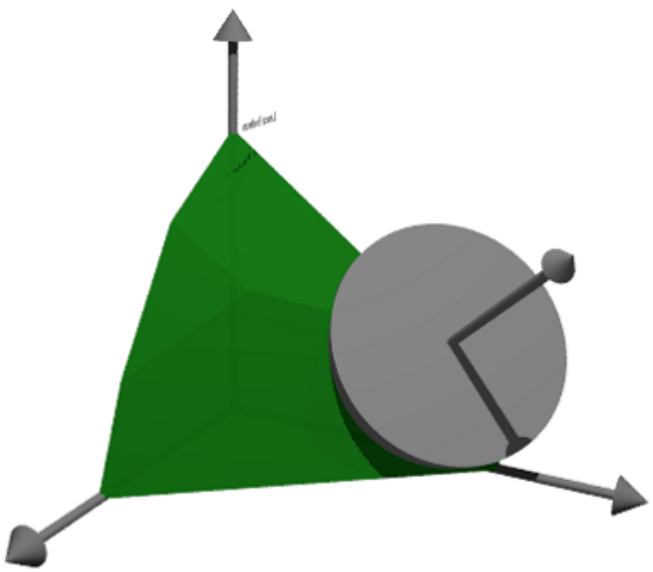
```
>writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')], ...
>  povlook(red,0.9)));
```

Dan sebuah error (kesalahan) dalam arah menuju titik optimum.

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

Kita menambahkan teks ke layar. Teks hanyalah sebuah objek 3D. Kita perlu menempatkan dan memutarnya sesuai dengan sudut pandang kita.

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



Contoh Lainnya

Anda dapat menemukan beberapa contoh tambahan untuk Povray di Euler pada file-file berikut:

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

```
>load povray;
>defaultpovray="C:/Program Files/POV-Ray/v3.7/bin/pvengine.exe";
>povstart("dandelin",zoom=2,angle=25);
>povcone([0,0,0],[0,0,5],3,look=texture(pigment(color Yellow transmit 0.5)));
```

```
  Closing bracket missing in function call!
  Error in:
  ... one([0,0,0],[0,0,5],3,look=texture(pigment(color Yellow transm ...
                                                    ^
```

```
>povplane([0,0,1],-3,texture="Blue",transparency=0.5);
```

```
  Argument texture not in parameter list of function povplane.

  Parameters:
  P, d, look, clippedby

  Error in:
  ... plane([0,0,1],-3,texture="Blue",transparency=0.5); ...
                                                       ^
```

```
>povsphere([0,0,1.5],1.0,texture="Red",transparency=0.3);
```

```
  Argument texture not in parameter list of function povsphere.

  Parameters:
  P, r, look

  Error in:
  ... ere([0,0,1.5],1.0,texture="Red",transparency=0.3); ...
                                                       ^
```

```
>povsphere([0,0,3.5],1.0,texture="Green",transparency=0.3);
```

```
  Argument texture not in parameter list of function povsphere.

  Parameters:
  P, r, look

  Error in:
  ... e([0,0,3.5],1.0,texture="Green",transparency=0.3); ...
                                                       ^
```

```
>povend();
```