Nama : Latifah Nurfitriyani
Kelas : Matematika E
NIM : 22305141010

Menggambar Plot 3D dengan EMT

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

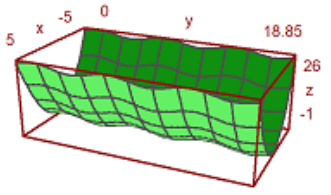
Euler menggambar fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan bagian-bagian di latar belakang. Secara umum, Euler menggunakan proyeksi pusat. Defaultnya adalah dari kuadran positif x-y menuju asal x=y=z=0, tetapi sudut=0° terlihat dari arah sumbu y. Sudut pandang dan tinggi dapat diubah.

Euler dapat membuat plot

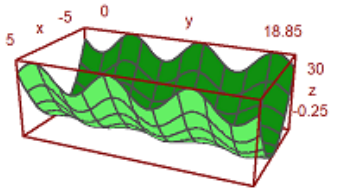
- permukaan dengan bayangan dan garis tingkat atau rentang tingkat,
- clouds of points,
- kurva parametrik,
- permukaan implisit.

Plot 3D dari sebuah fungsi menggunakan plot3d. Cara termudah adalah merencanakan sebuah ekspresi dalam x dan y. Parameter r mengatur jangkauan plot sekitar (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 fungsi, gunakan

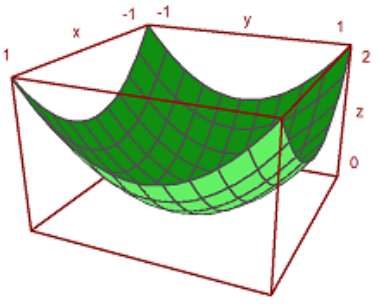
- ekspresi sederhana dalam x dan y,
- nama dari sebuah fungsi dari dua variabel
- atau matriks data.

Standarnya adalah kisi kawat yang diisi dengan warna yang berbeda di kedua sisi. Perhatikan bahwa jumlah default interval grid adalah 10, tetapi plot menggunakan jumlah default 40x40 persegi panjang untuk membangun permukaan. Hal ini dapat diubah.

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

Kami menggunakan default n=40 dan grid=10.

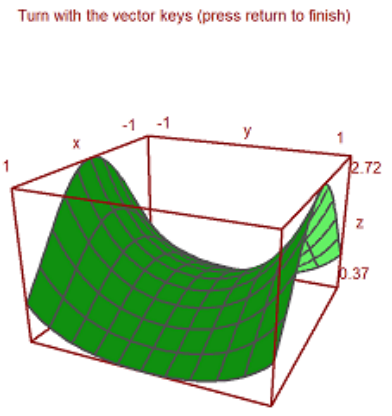
```
> plot3d("x^2+y^2"):
```



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

- left,right,up,down: putar sudut pandang
- +,-: memperbesar atau memperkecil
- a: menghasilkan anaglyph (lihat di bawah)
- l: ubah mengubah sumber cahaya (lihat di bawah)
- space: reset ke default
- return: mengakhiri interaksi

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



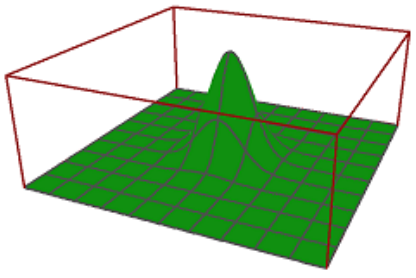
Jangkauan plot untuk fungsi dapat dispesifikasikan dengan

- a,b: the x-range
- c,d: the y-range
- r: sebuah persegi simetris sekitar (0,0).
- n: jumlah subinterval untuk plot.

Ada beberapa parameter untuk meningkatkan fungsi atau mengubah tampilan grafik.

fscale: skala ke nilai fungsi (default adalah <fscale).
scale: angka atau 1x2 vektor untuk skala ke arah x dan y.
frame: tipe frame (default 1).

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



Pandangan dapat diubah dalam berbagai cara.

- distance: jarak pandang ke plot.
- zoom: nilai zoom.
- angle: sudut ke sumbu y negatif dalam radian.
- height: tinggi tampilan dalam radian.

Nilai default dapat diinspeksi atau diubah dengan fungsi view(). Ini mengembalikan parameter sesuai urutan di atas.

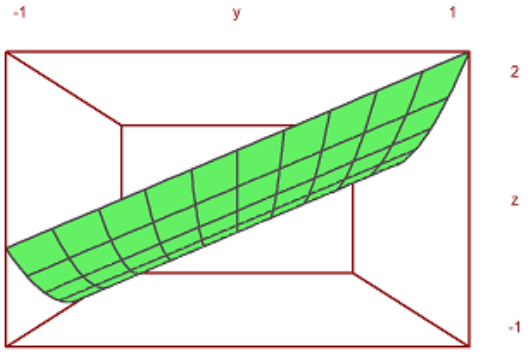
```
> view
```

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

Jarak yang lebih dekat membutuhkan sedikit zoom. Efeknya lebih seperti lensa sudut lebar.

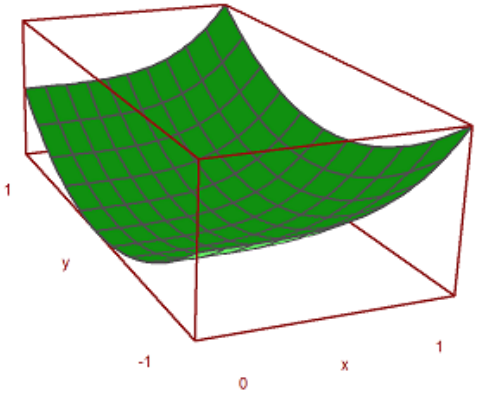
Dalam contoh berikut, angle=0 dan height=0 terlihat dari sumbu y negatif. Label sumbu y tersembunyi dalam kasus ini.

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



Plot selalu tampak ke tengah dari kubus plot. Anda dapat memindahkan pusat dengan parameter pusat.

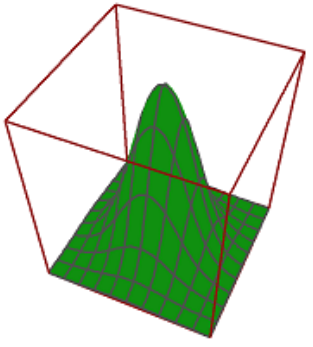
```
> plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>   center=[0.4,0,0],zoom=5.5):
```



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

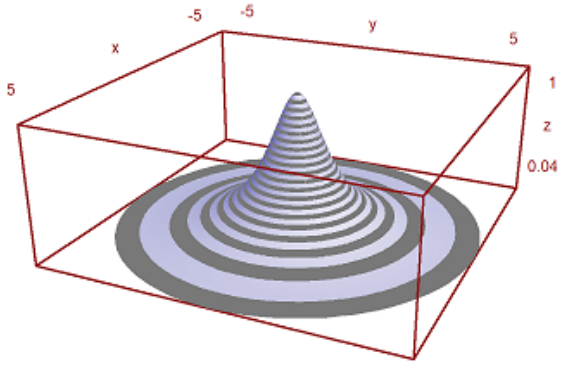
Jika Anda mematikan ini dengan scale=false, Anda perlu berhati-hati, bahwa plot masih cocok ke jendela plot, dengan mengubah jarak pandang atau zoom, dan memindahkan pusat.

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

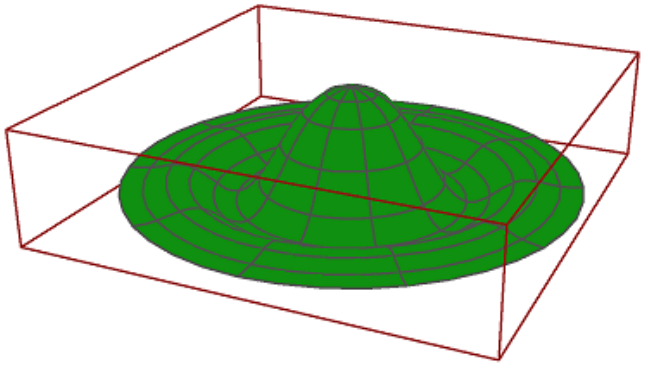


Plot kutub juga tersedia. Parameter polar=true menggambar plot polar. Fungsi harus tetap merupakan fungsi dari x dan y. Parameter "fscale" skala fungsi dengan skala sendiri. Jika tidak, fungsi ini diperbesar agar muat dalam kubus.

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



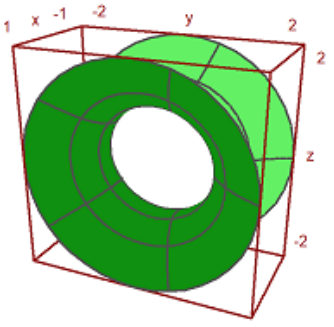

```
> function f(r) := exp(-r/2)*cos(r); ...
> plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=5):
```



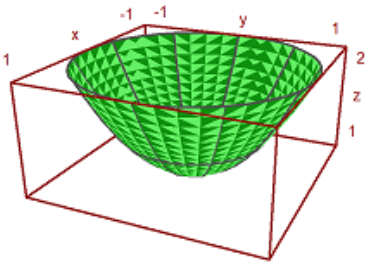
Rotasi parameter memutar fungsi dalam x mengelilingi sumbu x.

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

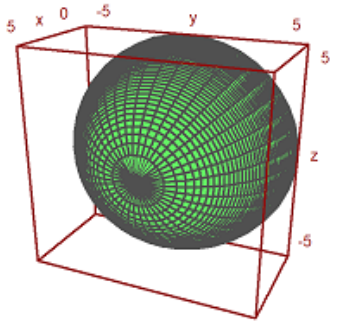
```
> plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



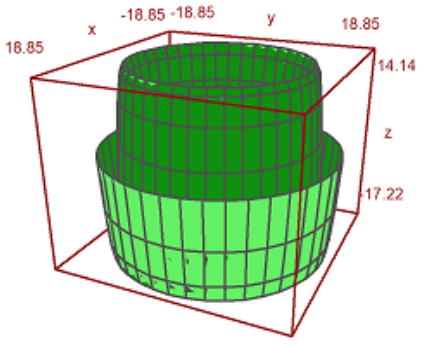

```
> plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




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

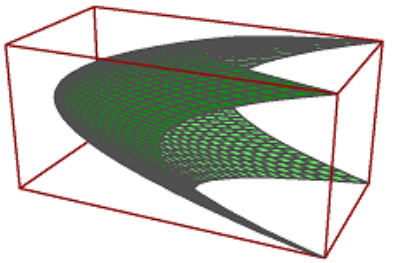



```
> plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Berikut adalah plot dengan tiga fungsi.

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



Plot Kontur

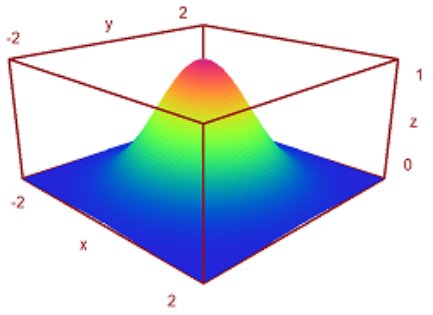
Untuk plot, Euler menambahkan garis kisi. Sebaliknya adalah mungkin untuk menggunakan garis tingkat dan warna satu warna atau warna spektral berwarna. Euler dapat menggambar ketinggian fungsi pada plot dengan bayangan. Dalam semua plot 3D Euler dapat menghasilkan anaglif red/cyan.

- >hue: Menyalakan shading cahaya bukan wires.
- >contour: Plot garis kontur otomatis pada plot.
- level=... (or levels): Sebuah vektor nilai untuk garis kontur.

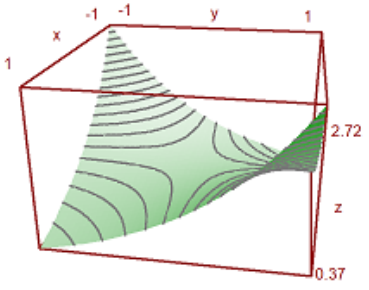
Standarnya adalah level="auto", yang menghitung beberapa baris level secara otomatis. Seperti yang Anda lihat dalam plot, tingkat sebenarnya adalah kisaran tingkat.

Gaya bawaan dapat diubah. Untuk plot kontur berikut, kami menggunakan grid yang lebih halus untuk 100x100 poin, skala fungsi dan plot, dan menggunakan sudut pandang yang berbeda.

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




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

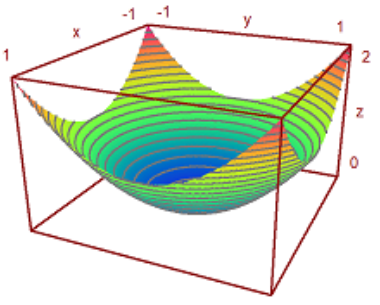


Bayangan default menggunakan warna abu-abu. Tapi rentang spektral warna juga tersedia.

- >spectral: Menggunakan skema spectral default
- color=...: Menggunakan warna khusus atau skema spectral

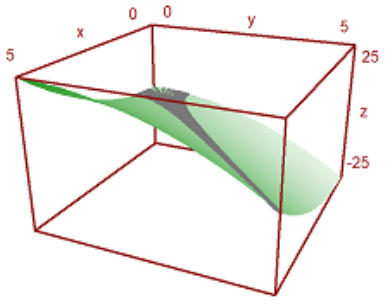
Untuk plot berikut, kami menggunakan skema spectral default dan meningkatkan jumlah poin untuk mendapatkan tampilan yang sangat halus.

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



Alih-alih baris level otomatis, kita juga dapat mengatur nilai dari baris level. Ini akan menghasilkan garis level tipis bukan rentang levels.

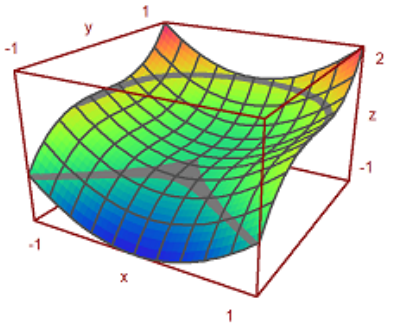
```
> plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=green):
```



Dalam plot berikut, kami menggunakan dua bands tingkat yang sangat luas dari -0.1 ke 1, dan dari 0.9 ke 1. Ini dimasukkan sebagai matriks dengan batas level sebagai kolom.

Selain itu, kita melapisi 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):
```

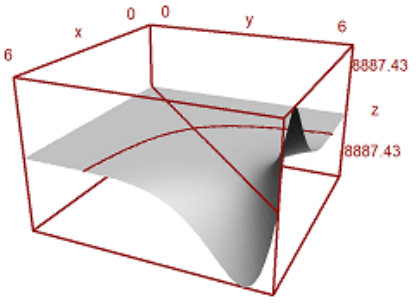


Dalam contoh berikut, kita plot set, di mana

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

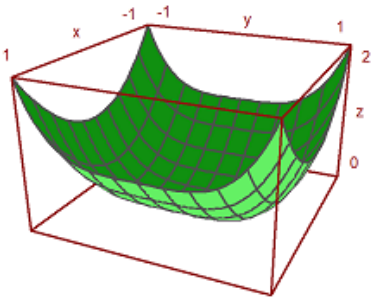
Kami menggunakan satu garis tipis untuk garis tingkat.

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



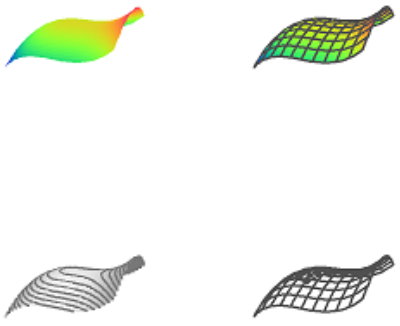
Adalah mungkin untuk menunjukkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

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



Berikut adalah beberapa gaya lagi. Kami selalu mematikan frame, dan menggunakan berbagai skema warna untuk plot dan grid.

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



Ada beberapa skema spectral lainnya, bernomor dari 1 sampai 9. Tapi Anda juga dapat menggunakan color=value, di mana nilai

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

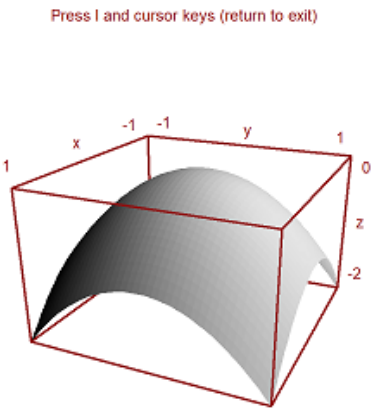
```
> figure(3,3); ...
> for i=1:9;  ...
>    figure(i); plot3d("x^2+y^2",spectral=i,>contour,>cp,<frame,zoom=4);  ...
> end; ...
> figure(0):
```

Sumber cahaya dapat diubah dengan kunci tanah dan kursor selama interaksi pengguna. Hal ini juga dapat diatur dengan parameter.

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

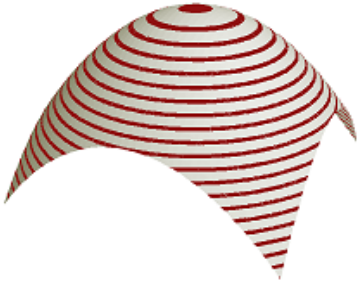
Perhatikan bahwa program tidak membuat perbedaan antara sisi plot. Tidak ada bayangan. Untuk ini Anda akan membutuhkan Povray.

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



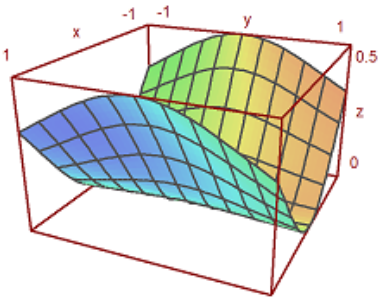
Parameter warna mengubah warna permukaan. Warna garis tingkat juga dapat diubah.

```
> 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):
```



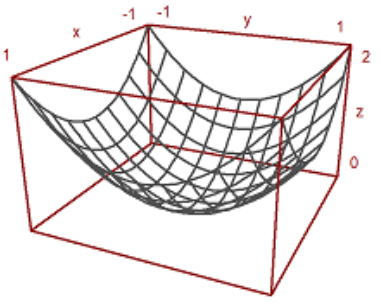
Warna 0 memberikan efek pelangi khusus.

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



Permukaannya juga bisa transparan.

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


-1
-1
x
y
1
1
2
z
0

Plot Implisit

Ada juga plot implisit dalam tiga dimensi. Euler menghasilkan pemotongan melalui objek. Fitur plot3d termasuk plot implisit. Plot ini menunjukkan himpunan nol fungsi dalam tiga variabel.
Solusi dari

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

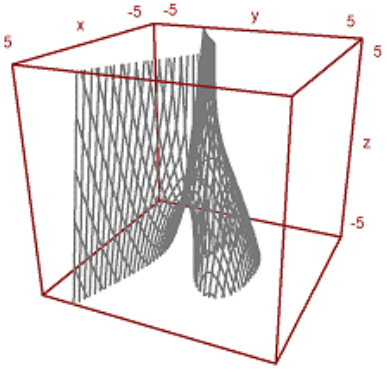
dapat divisualisasikan dalam potongan sejajar dengan x-y-, the x-z- dan the y-z-plane.

- implisit=1: potong paralel ke y-z-plane
- implisit=2: potong paralel ke x-z-plane
- implisit=4: potong paralel ke x-y-plane

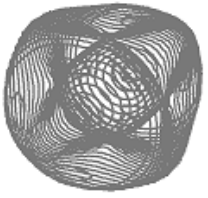
Tambahkan nilai-nilai ini, jika Anda suka. Dalam contoh kita plot

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

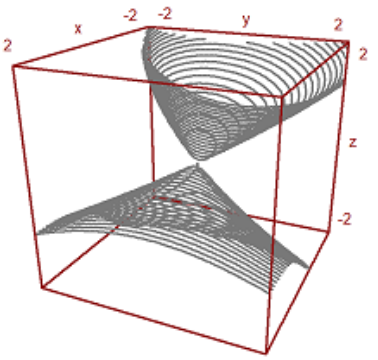
```
> plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```




```
> 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):
```


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

Merencanakan Data 3D

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

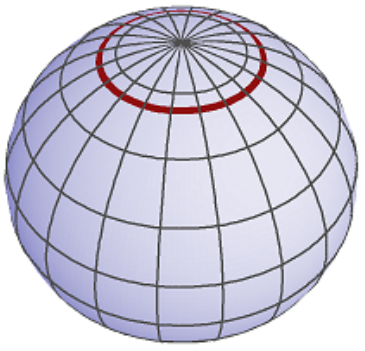
$$\gamma(t,s) = (x(t,s), y(t,s), z(t,s)))$$

Karena x, y, z adalah matriks, kita mengasumsikan bahwa (t, s) berjalan melalui grid persegi. Hasilnya, Anda dapat merencanakan gambar persegi panjang di ruang angkasa.

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

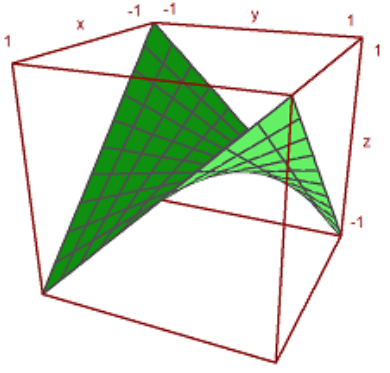
Dalam contoh berikut, kita menggunakan vektor nilai t dan vektor kolom nilai s untuk parameter permukaan bola. Dalam gambar kita dapat menandai daerah, dalam kasus kita daerah kutub.

```
> 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 contoh, yang merupakan grafik fungsi.

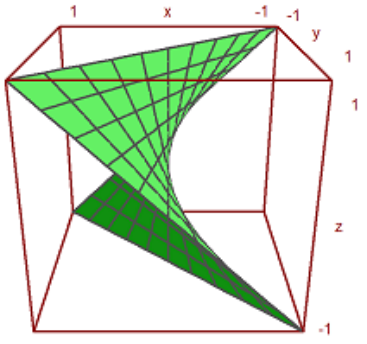
```
> t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



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

$$x = y\,z$$

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



Dengan lebih banyak upaya, kita dapat menghasilkan banyak permukaan.

Dalam contoh berikut kita membuat tampilan berbayang dari bola terdistorsi. Koordinat yang biasa untuk bola adalah

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

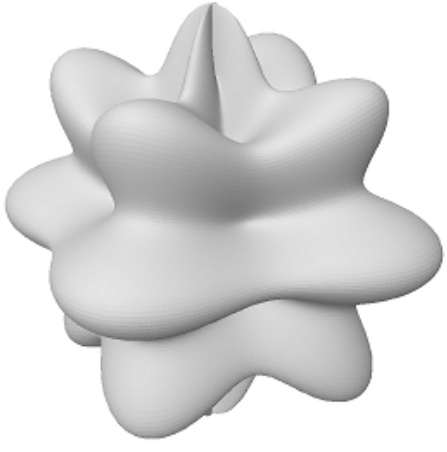
dengan

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

Kami distorsi ini dengan faktor

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

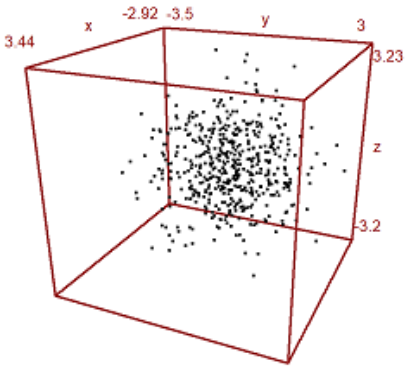
```
> t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
> d=1+0.2*(cos(4*t)+cos(8*s)); ...
> plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
>   light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, titik awan juga mungkin. Untuk plot data titik di ruang, kita perlu tiga vektor untuk koordinat titik.

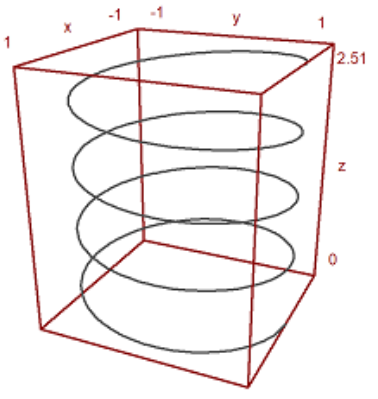
Gaya-gaya seperti dalam plot 2d dengan points=true;

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

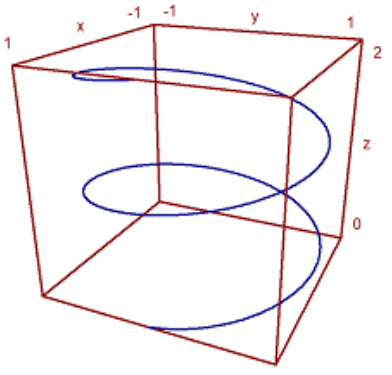


Hal ini juga dimungkinkan untuk plot kurva dalam 3D. Dalam hal ini, lebih mudah untuk menghitung titik kurva. Untuk kurva di bidang kami menggunakan urutan koordinat dan parameter wire=true.

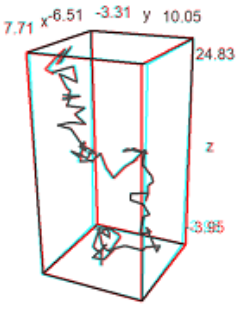
```
> t=linspace(0,8pi,500); ...
> plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

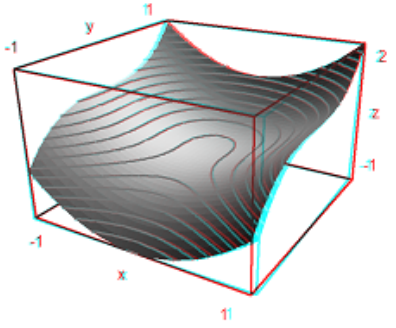



```
> X=cumsum(normal(3,100)); ...
>  plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



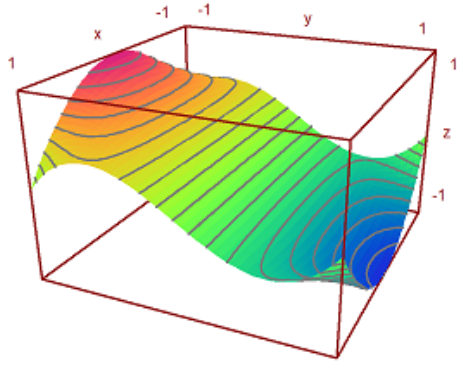
EMT juga dapat merencanakan dalam mode aglyph. Untuk melihat plot seperti itu, Anda perlu kacamata red/cyan glasses.

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



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

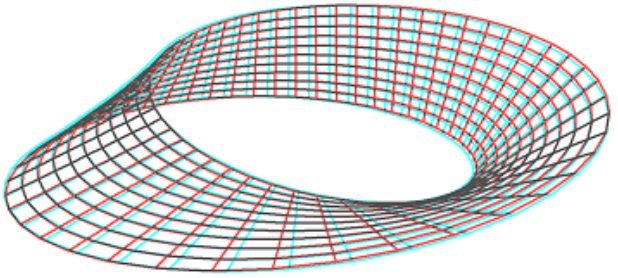
```
> plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler dapat merencanakan permukaan parameter juga, ketika parameter adalah x-, y-, dan z-values dari gambar grid persegi panjang di ruang.

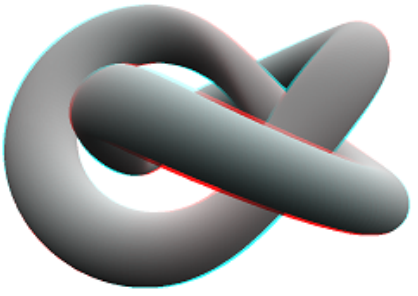
Untuk demo berikut, kami menyiapkan parameter u- dan v-, dan menghasilkan koordinat ruang dari ini.

```
> 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 adalah contoh yang lebih rumit, yang megah dengan 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):
```



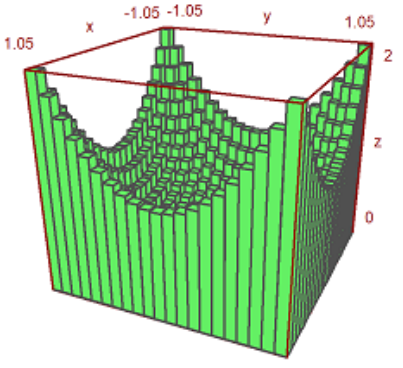
Plot bar mungkin juga. Untuk ini, kita harus menyediakan

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

z dapat lebih besar, tetapi hanya nilai nxn yang akan digunakan.

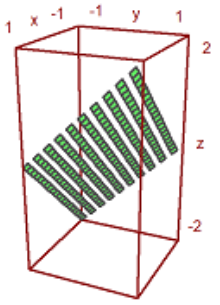
Dalam contoh, pertama-tama kita menghitung nilai. Kemudian kita sesuaikan x dan y, sehingga vektor berpusat pada nilai yang digunakan.

```
> 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 permukaan menjadi dua bagian atau lebih.

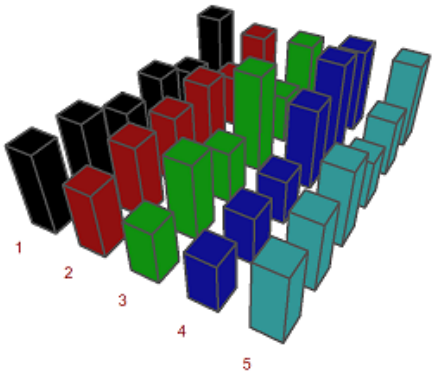
```
> x=-1:0.1:1; y=x'; z=x+y; d=zeros(size(x)); ...
> plot3d(x,y,z,disconnect=2:2:20):
```



Jika memuat atau menghasilkan matriks data M dari berkas dan perlu plot dalam 3D Anda dapat meningkatkan matriks ke [-1,1] dengan skala (M), atau meningkatkan matriks dengan >zscale. Hal ini dapat dikombinasikan dengan faktor skala individu yang diterapkan secara tambahan.

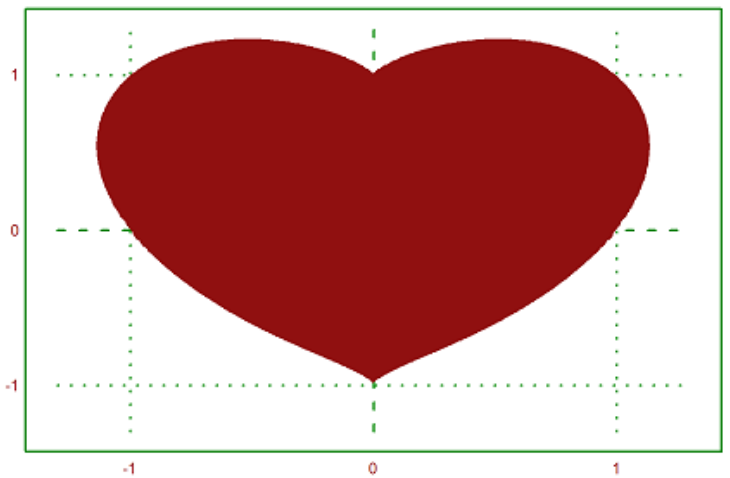
```
> i=1:20; j=i'; ...
> plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8):
```


```
> 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]):
```


1
2
3
4
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
```

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

Kami ingin mengubah kurva jantung di sekitar sumbu y. Berikut adalah ungkapan, yang mendefinisikan hati:

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

Selanjutnya kita atur

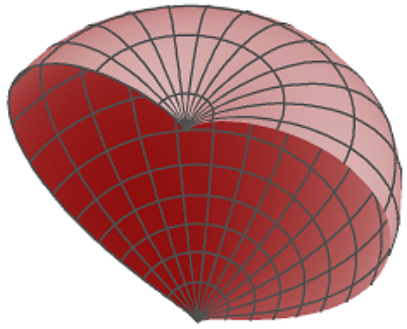
$$x = r.cos(a), \quad y = r.sin(a).$$

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

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

Ini memungkinkan untuk mendefinisikan fungsi numerik, yang menyelesaikan untuk r, jika a diberikan. Dengan fungsi itu kita dapat plot jantung berbalik sebagai permukaan parameter.

```
> 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°):
```

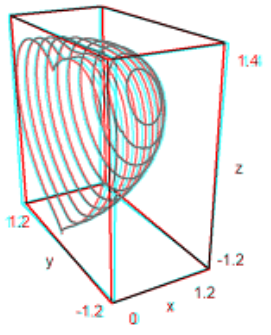


Berikut ini adalah plot 3D dari gambar di atas diputar mengelilingi sumbu-z. Kami mendefinisikan fungsi, yang menggambarkan objek.

```
> 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):
```


1.4
z
-1.2
1.2
x
0
-1.2
y
1.2

Plot 3D Khusus

Fungsi plot3d bagus untuk dimiliki, tetapi tidak memenuhi semua kebutuhan. Selain rutinitas yang lebih mendasar, ada kemungkinan untuk mendapatkan plot berbingkai dari objek apa pun yang Anda suka.

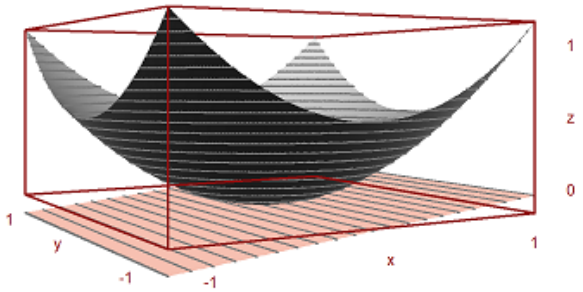
Meskipun Euler bukan program 3D, Euler dapat menggabungkan beberapa objek dasar. Kami mencoba untuk memvisualisasikan paraboloid dan tangennya.

```
> 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 framedplot() menyediakan bingkai, dan mengatur tampilan.

```
> framedplot("myplot",[-1,1,-1,1,0,1],height=0,angle=-30°, ...
>   center=[0,0,-0.7],zoom=4):
```

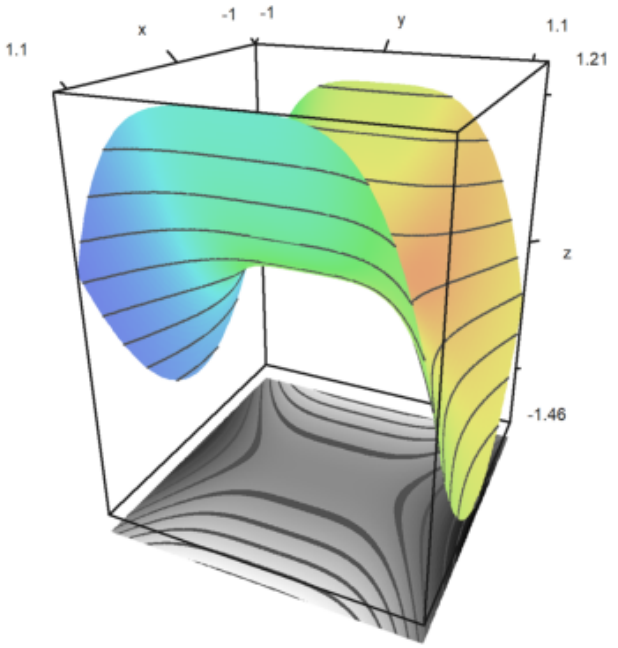


Dengan cara yang sama, Anda dapat merencanakan bidang kontur secara manual. Perhatikan bahwa plot3d() mengatur jendela ke fullwindow() secara default, tetapi plotcontourplane() mengasumsikan bahwa.

```
> 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):
```



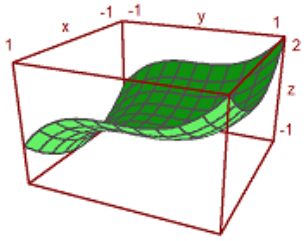
Animasi

Euler dapat menggunakan bingkai untuk pra-komputasi animasi.

Salah satu fungsi, yang memanfaatkan teknik ini adalah rotasi. Hal ini dapat mengubah sudut pandang dan menggambar ulang plot 3D. Fungsi memanggil addpage() untuk setiap plot baru. Akhirnya animasi plot.

Silakan pelajari sumber rotasi untuk melihat rincian lebih lanjut.

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



Menggambar Povray

Dengan bantuan povray.e file Euler, Euler dapat menghasilkan file Povray. Hasilnya sangat bagus untuk dilihat.

Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/, dan memasukkan subdirektori "bin" Povray ke jalur lingkungan, atau mengatur variabel "defaultpovray" dengan jalur penuh menunjuk ke "pvengine.exe".

Antarmuka Povray Euler menghasilkan file Povray di direktori rumah pengguna, dan memanggil Povray untuk mengurai file-file ini. Nama berkas baku adalah current.pov, dan direktori baku adalah eulerhome(), biasanya c:\Users\Username\Euler. Povray menghasilkan file PNG, yang dapat dimuat oleh Euler ke dalam buku catatan. Untuk membersihkan file-file ini, gunakan povclear().

Fungsi pov3d berada dalam semangat yang sama dengan plot3d. Ini dapat menghasilkan grafik fungsi f(x,y), atau permukaan dengan koordinat X, Y, Z dalam matriks, termasuk garis tingkat opsional. Fungsi ini memulai pelacak sinar secara otomatis, dan memuat adegan ke dalam notebook Euler.

Selain pov3d(), ada banyak fungsi, yang menghasilkan objek Povray. Fungsi ini mengembalikan string, berisi kode Povray untuk objek. Untuk menggunakan fungsi ini, mulai file Povray dengan povstart(). Kemudian gunakan writeln(...) untuk menulis objek ke berkas adegan. Terakhir, akhiri file dengan povend(). Secara default, pelacak sinar akan dimulai, dan PNG akan dimasukkan ke dalam notebook Euler.

Fungsi objek memiliki parameter yang disebut "look", yang membutuhkan string dengan kode Povray untuk tekstur dan akhir objek. Fungsi povlook() dapat digunakan untuk menghasilkan string ini. Ini memiliki parameter untuk warna, transparansi, Phong Shading dll.

Perhatikan bahwa alam semesta Povray memiliki sistem koordinat lain. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray. Jadi Anda dapat terus berpikir dalam sistem koordinat Euler dengan z menunjuk vertikal ke atas, dan sumbu x, y, z dalam arti tangan kanan.
Anda perlu memuat file povray.

```
> load povray;
```

Pastikan, direktori Povray bin berada di path. Jika tidak mengedit variabel berikut sehingga berisi path ke povray executable.

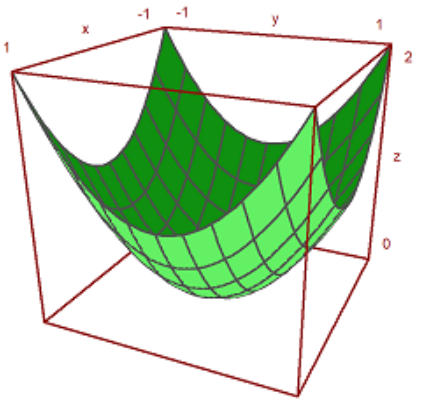
```
> defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

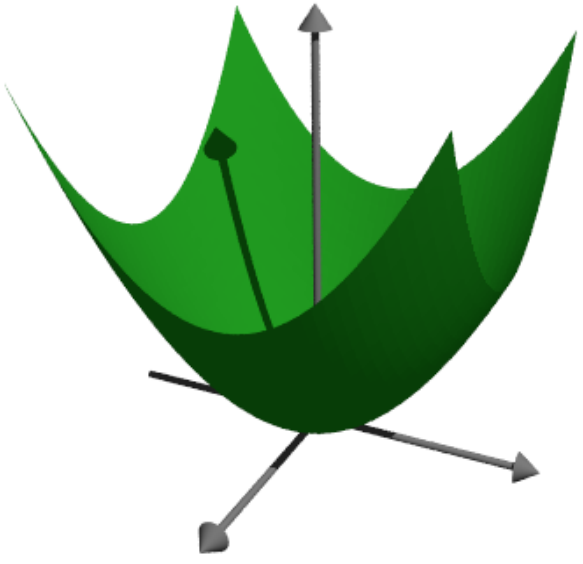
Untuk kesan pertama, kita merencanakan fungsi sederhana. Perintah berikut menghasilkan file povray di direktori pengguna Anda, dan menjalankan Povray untuk melacak file ini.

Jika Anda memulai perintah berikut, Povray GUI harus membuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan diminta, jika Anda ingin membiarkan file exe berjalan. Anda dapat menekan cancel untuk menghentikan pertanyaan lebih lanjut. Anda mungkin harus menekan OK di jendela Povray untuk mengenali dialog start-up Povray.

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

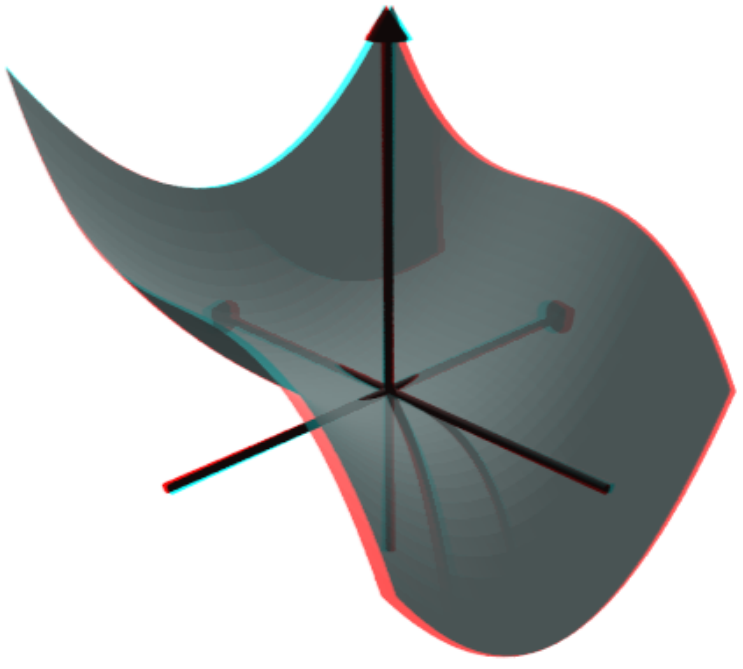



```
> pov3d("x^2+y^2",zoom=3);
```



Kita dapat membuat fungsi transparan dan menambahkan penyelesaian lain. Kita juga dapat menambahkan garis tingkat ke plot fungsi.

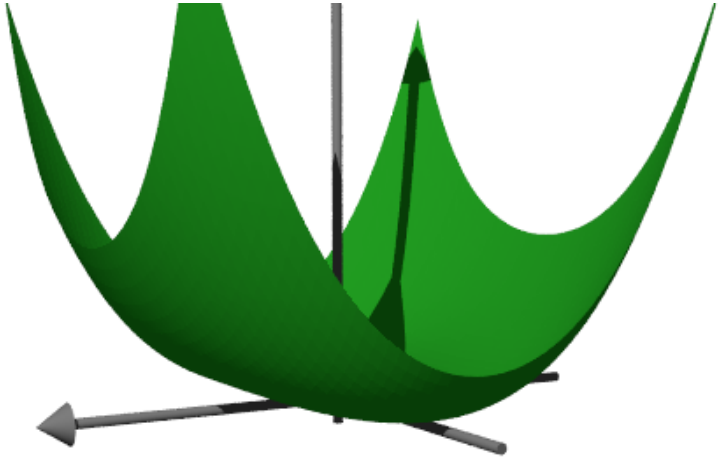
```
> pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
>   look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



Kadang-kadang diperlukan untuk mencegah skala fungsi, dan skala fungsi dengan tangan.

Kami plot set titik dalam bidang kompleks, di mana produk 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);
```

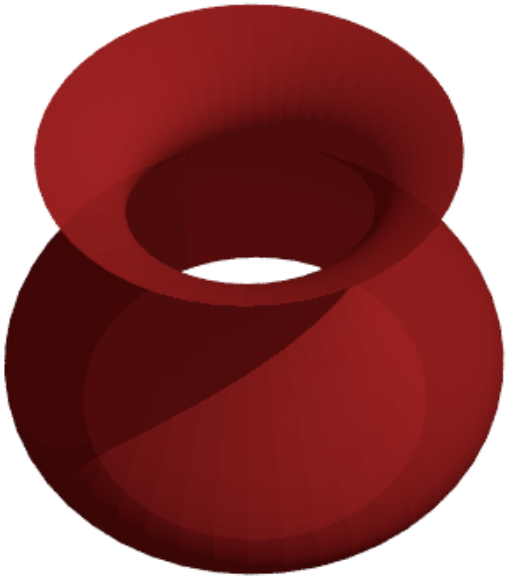


Plotting dengan Koordinat

Alih-alih fungsi, kita dapat plot dengan koordinat. Seperti dalam plot3d, kita perlu tiga matriks untuk mendefinisikan objek.

Dalam contoh kita mengubah fungsi sekitar sumbu-z.

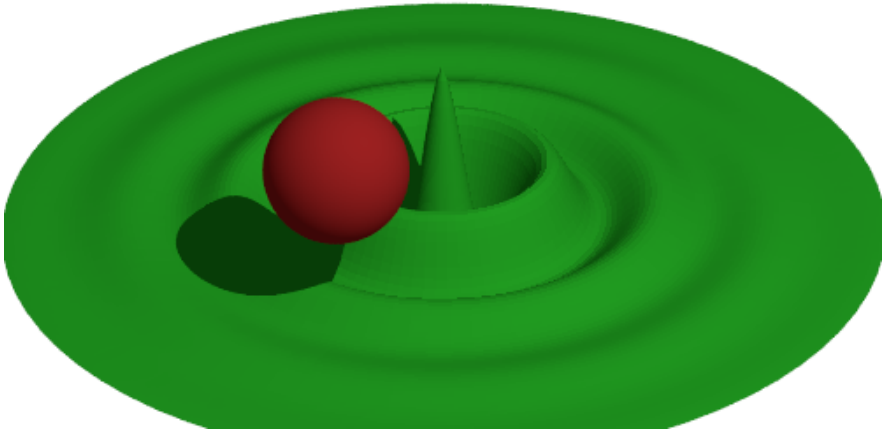
```
> function f(x) := x^3-x+1; ...
> x=-1:0.01:1; t=linspace(0,2pi,50)'; ...
> Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
> pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,axis=0,zoom=4,light=[10,5,15]);
```



Dalam contoh berikut, kami merencanakan gelombang yang lembap. Kita menghasilkan gelombang dengan bahasa matriks Euler.

Kami juga menunjukkan, bagaimana objek tambahan dapat ditambahkan ke adegan pov3d. Untuk pembuatan objek, lihat contoh berikut. Perhatikan bahwa plot 3d skala plot, sehingga cocok ke kubus unit.

```
> 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 bayangan canggih Povray, sangat sedikit titik yang dapat menghasilkan permukaan yang sangat halus. Hanya pada batas-batas dan dalam bayangan trik mungkin menjadi jelas.

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

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

$$x^2\,y^3$$

Persamaan permukaannya adalah [x,y,Z]. Kami menghitung dua turunan ke x dan y dari ini dan mengambil produk silang sebagai normal.

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

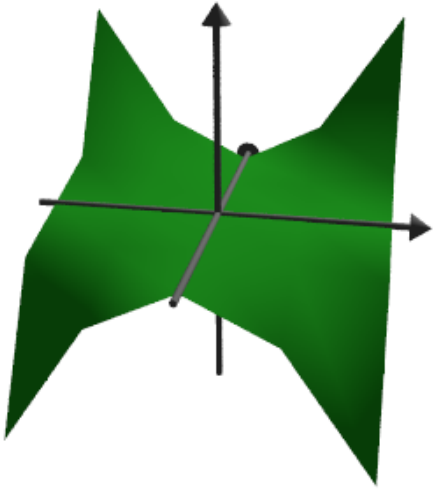
Kita mendefinisikan normal sebagai produk silang dari turunan ini, dan mendefinisikan fungsi koordinat.

```
> 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 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);
```



Berikut ini adalah simpul Trefoil yang dilakukan oleh A. Busser di Povray. Ada versi yang lebih baik dari ini dalam contoh.

See: Examples\Trefoil Knot | Trefoil Knot

Untuk tampilan yang baik dengan tidak terlalu banyak poin, kami menambahkan vektor normal di sini. Kami menggunakan Maxima untuk menghitung normal bagi kita. Pertama, tiga fungsi untuk 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 dua vektor turunan ke x dan y.

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

Sekarang normal, yang merupakan produk silang dari dua turunan.

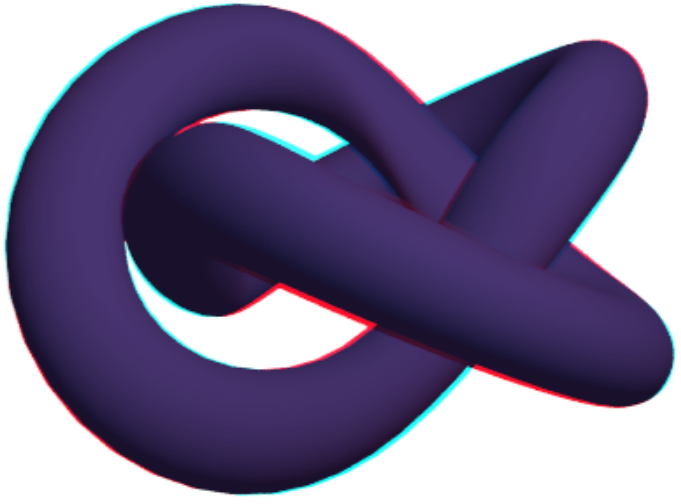
```
> dn &= crossproduct(dx,dy);
```

Sekarang kita mengevaluasi semua ini secara numerik.

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

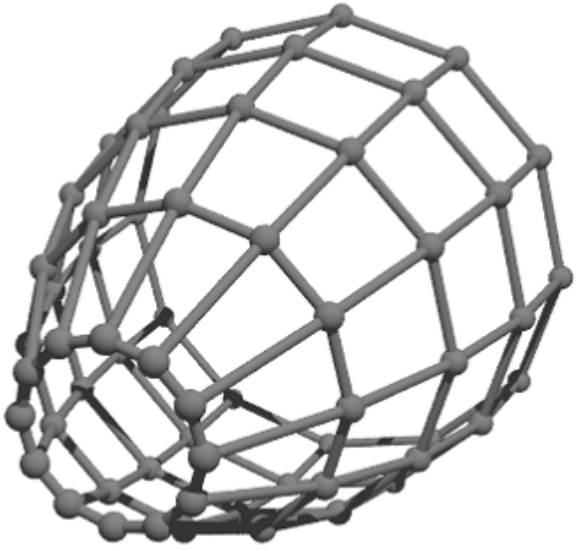
Vektor normal adalah evaluasi ekspresi simbolik dn[i] untuk i=1,2,3. Sintaksis untuk ini adalah &"expression"(parameters). Ini adalah alternatif untuk metode dalam contoh sebelumnya, di mana kita mendefinisikan ekspresi simbolik NX, NY, NZ pertama.

```
> 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));
```



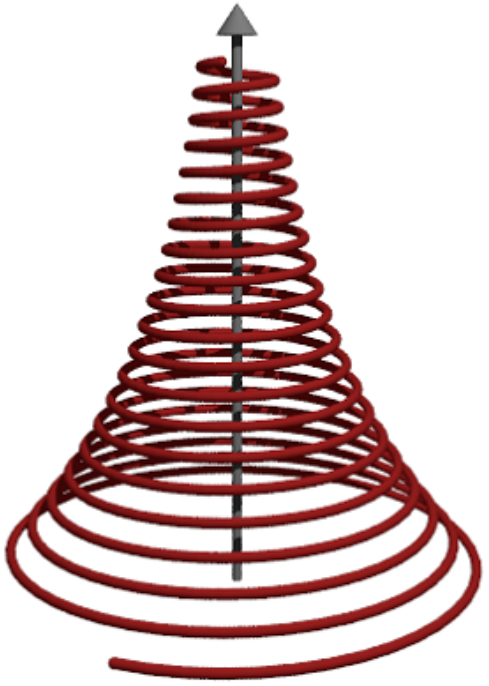
Kita juga dapat menghasilkan grid dalam 3D.

```
> 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 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, kami menggunakan pov3d untuk plot permukaan. Antarmuka povray di Euler juga dapat menghasilkan objek Povray. Objek-objek ini disimpan sebagai string dalam Euler, dan perlu ditulis ke file Povray.

Kita mulai output dengan povstart().

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

Pertama kita mendefinisikan tiga silinder, dan menyimpannya dalam string di Euler.

Fungsi povx() dll. cukup mengembalikan vektor [1,0,0], yang dapat digunakan sebagai gantinya.

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

String mengandung kode Povray, yang kita tidak perlu mengerti pada saat itu.

```
> 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, kami menambahkan tekstur ke objek dalam tiga warna yang berbeda.

Itu dilakukan dengan povlook(), yang mengembalikan string dengan kode Povray yang relevan. Kita dapat menggunakan warna standar Euler, atau menentukan warna kita sendiri. Kita juga dapat menambahkan transparansi, atau mengubah cahaya lingkungan.

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

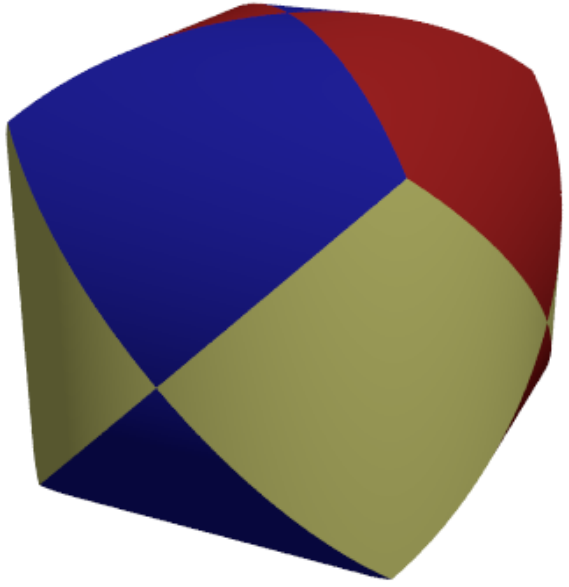
```
 texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> }  }
 finish { ambient 0.5 }
```

Sekarang kita mendefinisikan objek persimpangan, dan menulis hasilnya ke file.

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

Persimpangan tiga silinder sulit untuk memvisualisasikan, jika Anda tidak pernah melihatnya sebelumnya.

```
> povend;
```



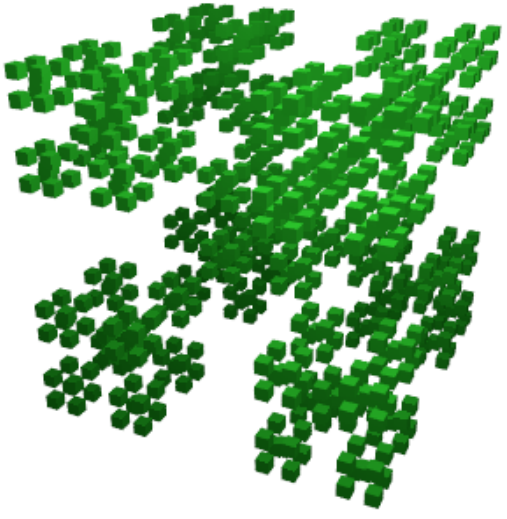
Fungsi berikut menghasilkan fraktal secara rekursif.

Fungsi pertama menunjukkan, bagaimana Euler menangani objek Povray sederhana. Fungsi povbox() mengembalikan string, berisi koordinat kotak, tekstur dan akhir.

```
> 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();
```



Perbedaan memungkinkan pemotongan satu objek dari objek lain. Seperti persimpangan, ada bagian dari objek CSG Povray.

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

Untuk demonstrasi ini, kita mendefinisikan objek dalam Povray, bukan menggunakan string dalam Euler. Definisi ditulis ke file segera.

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

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

Kita dapat menggunakan objek ini dalam povobject(), yang mengembalikan string seperti biasa.

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

Kami menghasilkan kubus kedua, dan memutar dan skala sedikit.

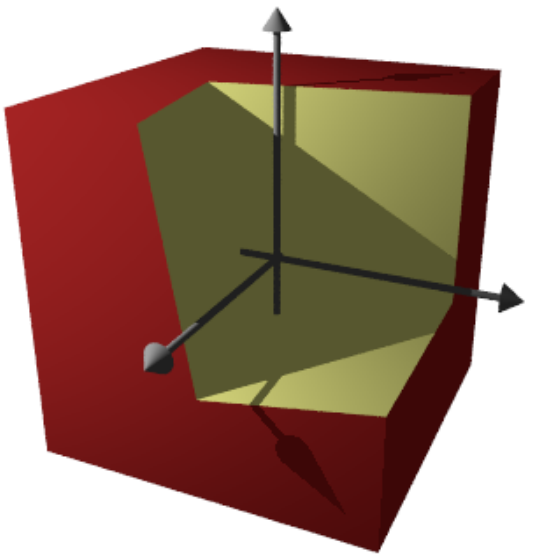
```
> c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...
>  rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Kemudian kita mengambil perbedaan dari dua objek.

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

Sekarang tambahkan tiga sumbu.

```
> 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 merencanakan himpunan dimana f(x,y,z)=0, sama seperti parameter implisit dalam plot3d. Namun, hasilnya terlihat jauh lebih baik.

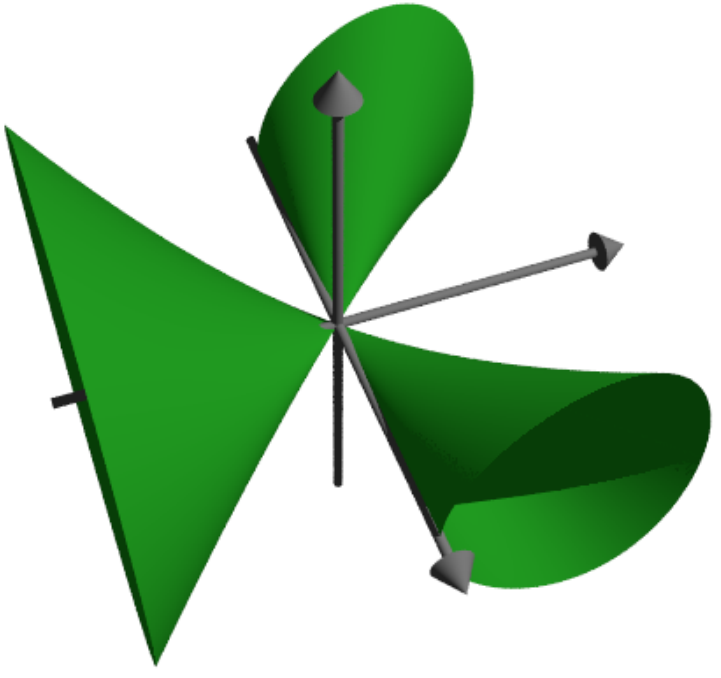
Sintaksis untuk fungsi sedikit berbeda. Anda tidak dapat menggunakan keluaran ekspresi Maxima atau Euler.

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

```
> povstart(angle=70°,height=50°,zoom=4);
```

Buat permukaan implisit. Perhatikan sintaks yang berbeda dalam ekspresi.

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



Objek Mesh

Dalam contoh ini, kami menunjukkan bagaimana membuat objek jaring, dan menggambarnya dengan informasi tambahan.

Kami ingin memaksimalkan xy dalam kondisi x+y=1 dan menunjukkan sentuhan tangensial dari garis tingkat.

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

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

Berikut ini mendefinisikan objek mesh, dan menuliskannya segera 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);
```

Sekarang kita mendefinisikan dua cakram, yang akan berpotongan 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 dikurangi dua cakram.

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

Tulis dua persimpangan.

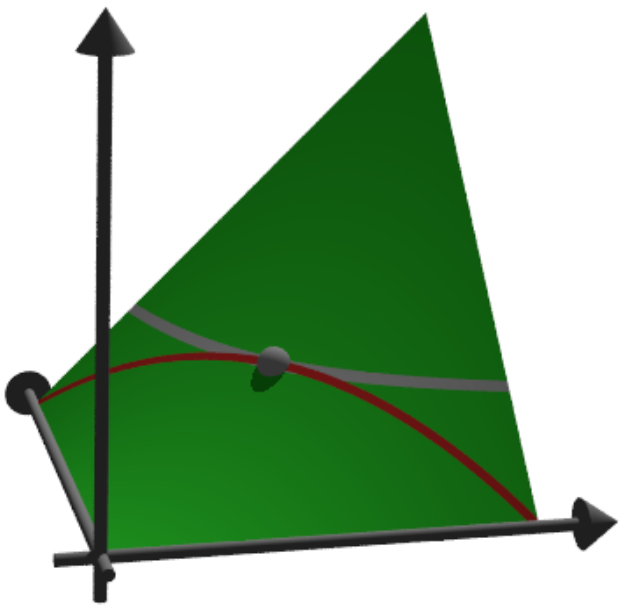
```
> writeln(povintersection([mesh,cl],povlook(red)); ...
> writeln(povintersection([mesh,ll],povlook(gray)));
```

Tulis titik maksimal.

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

Tambahkan sumbu dan selesaikan.

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



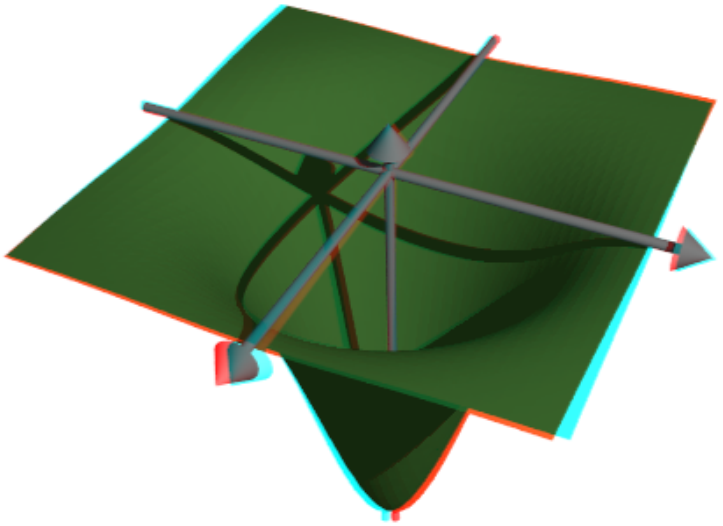
Anaglyphs in Povray

Untuk menghasilkan anaglif untuk kacamata merah/sian, Povray harus berlari dua kali dari posisi kamera yang berbeda. Ini menghasilkan dua file Povray dan dua file PNG, yang dimuat dengan fungsi loadanaglyph().

Tentu saja, Anda perlu kacamata merah/sian untuk melihat contoh-contoh berikut dengan benar.

Fungsi pov3d() memiliki sakelar sederhana untuk menghasilkan sebuah aglif.

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



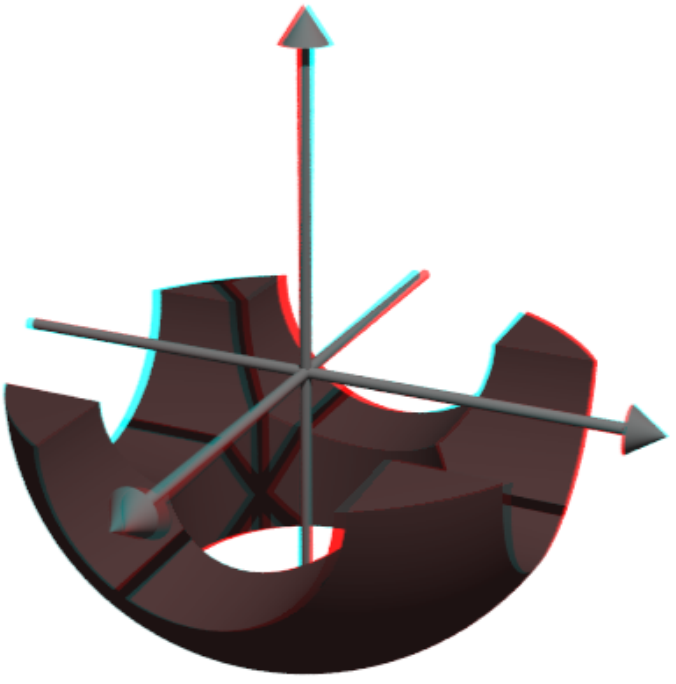
Jika Anda membuat adegan dengan objek, Anda perlu menempatkan pembuatan adegan ke dalam fungsi, dan menjalankannya dua kali dengan nilai yang berbeda untuk parameter aglif.

```
> 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 ini. Parameternya seperti di povstart() dan povend() digabungkan.

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



Menentukan Objek sendiri

Antarmuka povray Euler berisi banyak objek. Tapi kau tidak dibatasi untuk ini. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau objek yang sama sekali baru.

Kami menunjukkan torus. Perintah Povray untuk ini adalah "torus". Jadi kita kembali string dengan perintah ini dan parameternya. Perhatikan bahwa torus selalu berpusat pada asal usulnya.

```
> 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, diterjemahkan 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 tempatkan benda-benda ini ke dalam sebuah adegan. Untuk tampilan, kami 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();
```

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);
```

Berikut adalah contoh yang lebih rumit. Kami memecahkan

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

dan menunjukkan poin layak dan 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, benar.

Selanjutnya kita tentukan dua objek. Yang pertama adalah pesawat

$$a \cdot x \leq b$$

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

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

Kemudian kita tentukan persimpangan dari semua setengah ruang dan 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 bisa memplot 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); ...
```

Berikut ini adalah lingkaran di sekitar optimum.

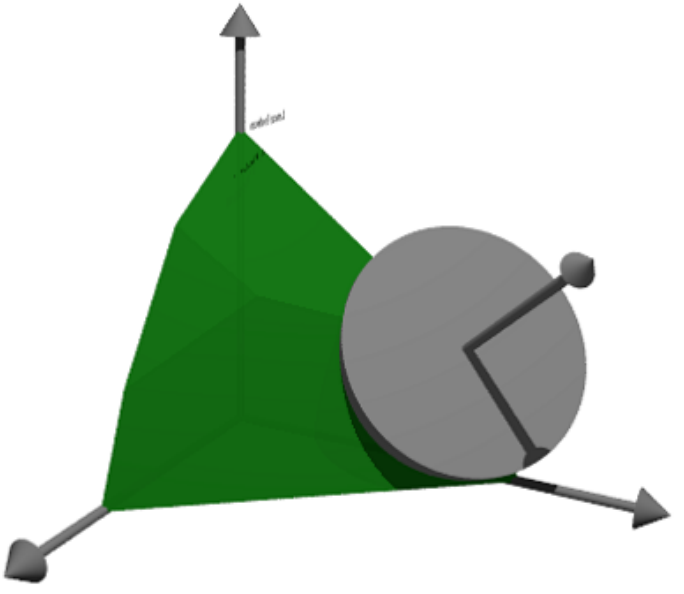
```
> writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')], ...
>   povlook(red,0.9)));
```

Dan kesalahan dalam arah optimum.

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

Kami menambahkan teks ke layar. Teks hanya objek 3D. Kita perlu menempatkan dan mengubahnya sesuai dengan pandangan kita.

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



Contoh lain

Anda dapat menemukan beberapa contoh lagi untuk Povray di Euler di file berikut.

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