TI_EMT_Plot3D_Prima Rosalina_24030130035

Nama : Prima Rosalina
NIM : 24030130035
Kelas : Pendidikan Matematika C 2024

Menggambar Plot 3D dengan EMT

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

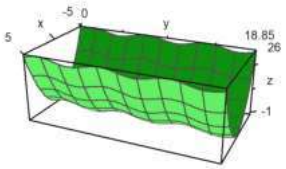
Euler menggambar fungsi semacam itu dengan menggunakan algoritma pengurutan untuk menyembunyikan bagian di latar belakang. Secara umum, Euler menggunakan proyeksi sentral. Default-nya adalah dari kuadran x-y positif menuju titik asal x=y=z=0, tetapi angle=0° melihat ke arah sumbu y. Sudut pandang dan ketinggian dapat diubah.

Euler dapat memplot

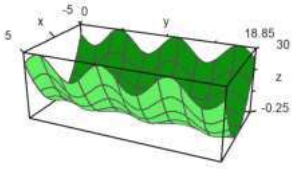
- permukaan dengan shading dan garis level atau rentang level,
- kumpulan titik,
- kurva parametrik,
- permukaan implisit.

Plot 3D dari sebuah fungsi menggunakan plot3d. Cara termudah adalah dengan memplot sebuah ekspresi dalam x dan y. Parameter r mengatur jangkauan plot di 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 Dua Variabel

Untuk grafik sebuah fungsi, gunakan

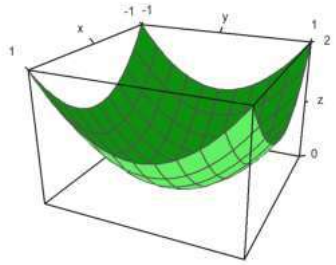
- ekspresi sederhana dalam x dan y,
- nama dari sebuah fungsi dua variabel,
- atau matriks data.

Default-nya adalah grid kawat terisi dengan warna berbeda di kedua sisinya. Perlu dicatat bahwa jumlah interval grid default adalah 10, tetapi plot menggunakan jumlah default 40x40 persegi panjang untuk membentuk permukaan. Hal ini dapat diubah.

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

Kita menggunakan default n=40 dan grid=10.

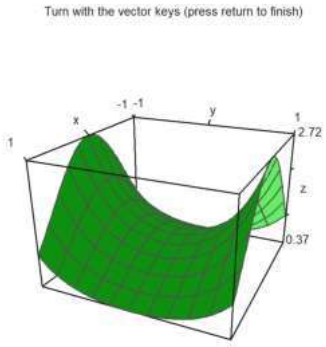
```
>plot3d("x^2+y^2"):
```



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

- left, right, up, down: memutar sudut pandang
- +, -: memperbesar atau memperkecil
- a: menghasilkan anaglyph (lihat di bawah)
- l: mengaktifkan/menonaktifkan perputaran sumber cahaya (lihat di bawah)
- space: kembali ke default
- return: mengakhiri interaksi

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



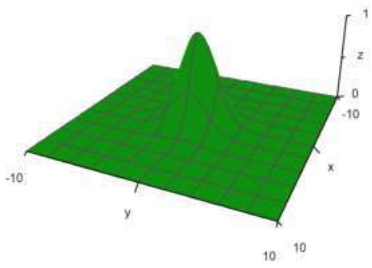
Rentang plot untuk fungsi dapat ditentukan dengan

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

Terdapat beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik.

fscale: skala terhadap nilai fungsi (default adalah <fscale).
scale: 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 dapat diubah dengan berbagai cara.

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

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

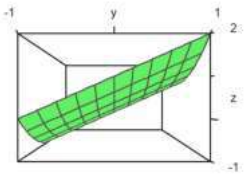
```
>view
```

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

Jarak yang lebih dekat memerlukan zoom yang lebih kecil. Efeknya lebih mirip dengan lensa sudut lebar.

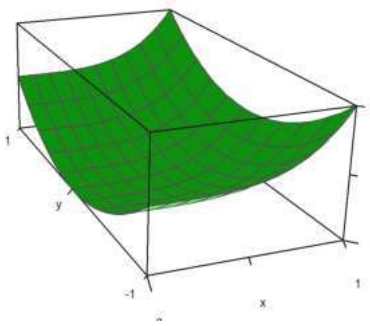
Dalam contoh berikut, angle=0 dan height=0 melihat dari arah sumbu y-negatif. Label sumbu untuk y tersembunyi dalam kasus ini.

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



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

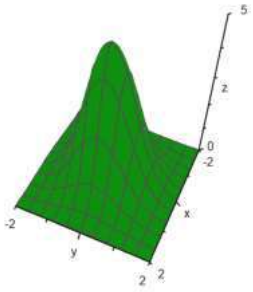
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
   center=[0.4,0,0],zoom=5):
```



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

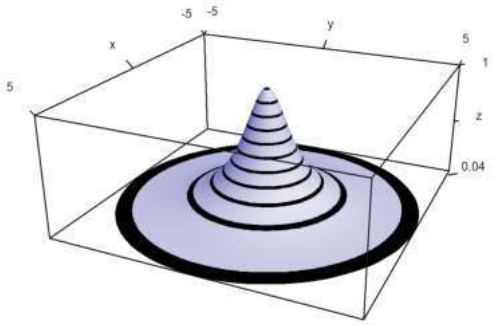
Jika Anda mematikan ini dengan **scale=false**, Anda perlu memastikan bahwa plot tetap muat di dalam jendela plot, dengan cara mengubah jarak pandang atau zoom, serta memindahkan pusat.

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

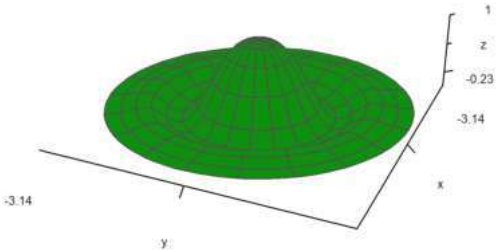


Plot polar juga tersedia. Parameter polar=true akan menggambar plot polar. Fungsi tersebut tetap harus berupa fungsi dari x dan y. Parameter fscale menskalakan fungsi dengan skala sendiri. Jika tidak, fungsi akan diskalakan agar pas ke dalam sebuah kubus.

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



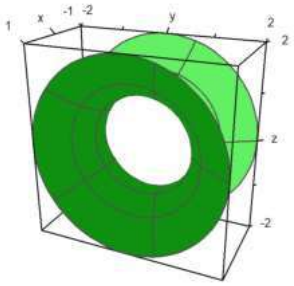

```
>function f(r) := exp(-r/2)*cos(r); ...
 plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



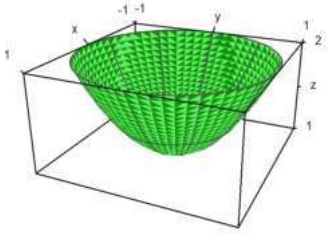
Parameter rotate memutar sebuah fungsi dalam x terhadap sumbu-x.

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

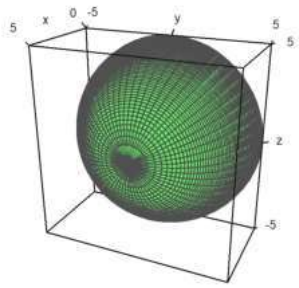
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



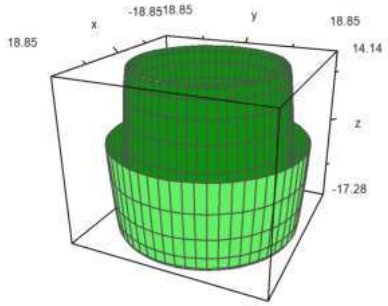

```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




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

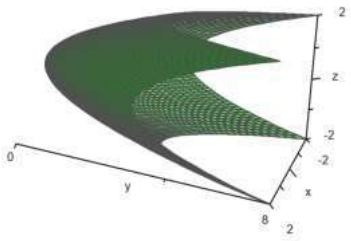



```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Berikut adalah sebuah plot dengan tiga fungsi.

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



Plot Kontur

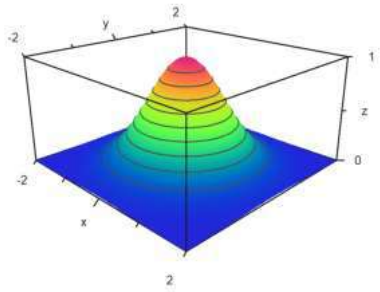
Untuk plot, Euler menambahkan garis grid. Sebagai gantinya, dimungkinkan untuk menggunakan garis level dan rona satu warna atau rona berwarna spektral. Euler dapat menggambar ketinggian fungsi pada sebuah plot dengan shading. Dalam semua plot 3D, Euler dapat menghasilkan anaglyph merah/cyan.

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

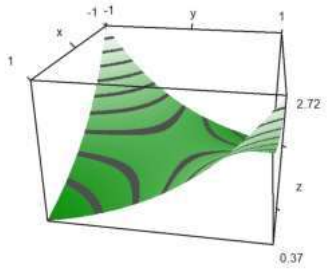
Default-nya adalah level="auto", yang secara otomatis menghitung beberapa garis level. Seperti yang terlihat pada plot, level sebenarnya berupa rentang level.

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

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




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

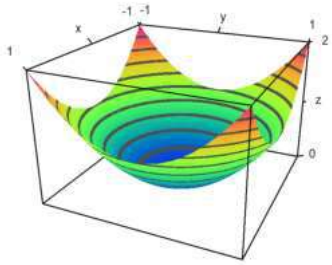


Shading default menggunakan warna abu-abu. Namun, rentang warna spektral juga tersedia.

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

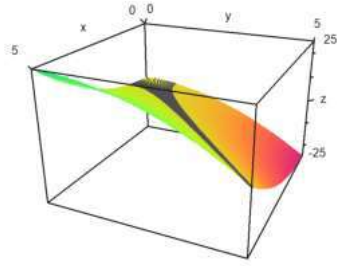
Untuk plot berikut, kita menggunakan skema spektral default dan menambah jumlah titik untuk mendapatkan tampilan yang sangat halus.

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



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

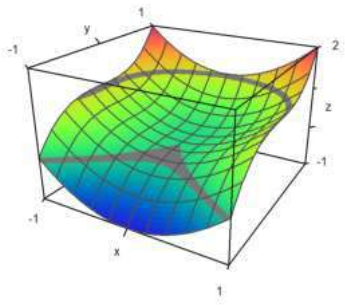
```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



Dalam plot berikut, kita menggunakan dua pita level yang sangat lebar 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.

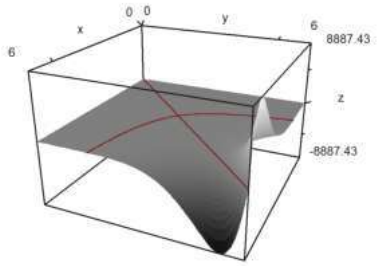
```
>plot3d("x^2+y^3",level=[-0.1,0.9;0,1], ...
   >spectral,angle=30°,grid=10,contourcolor=gray):
```



Dalam contoh berikut, kita memplot himpunan, di mana

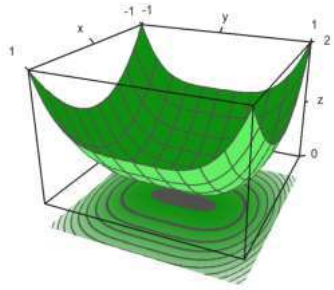
Kita menggunakan satu garis tipis untuk garis level.

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



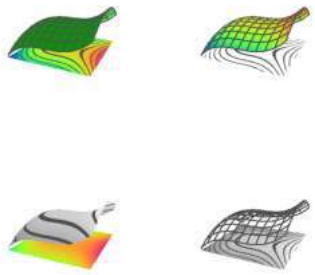
Dimungkinkan untuk menampilkan bidang kontur di bawah plot. Sebuah warna dan jarak ke plot dapat ditentukan.

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



Berikut adalah beberapa gaya lainnya. Kita selalu mematikan bingkai, dan menggunakan berbagai skema warna untuk plot dan grid.

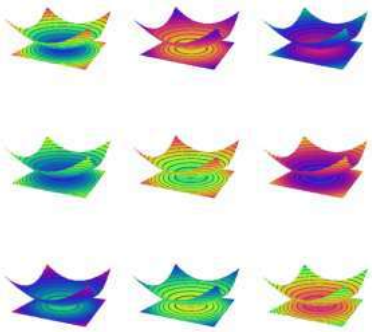
```
>figure(2,2); ...
 expr="y^3-x^2"; ...
 figure(1);  ...
   plot3d(expr,<frame,>cp,cpcolor=spectral); ...
 figure(2);  ...
   plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
 figure(3);  ...
   plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
 figure(4);  ...
   plot3d(expr,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



Ada beberapa skema spektral lain, diberi nomor dari 1 hingga 9. Namun, Anda juga dapat menggunakan color=value, di mana value

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

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

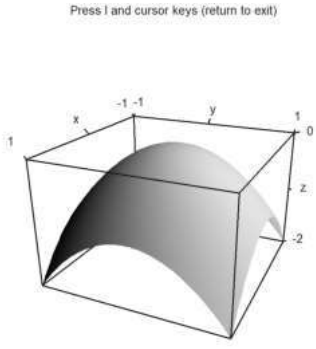


Sumber cahaya dapat diubah dengan tombol l dan tombol kursor selama interaksi pengguna. Hal ini juga dapat diatur dengan parameter.

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

Perlu dicatat bahwa program tidak membedakan antara sisi-sisi plot. Tidak ada bayangan. Untuk hal tersebut Anda memerlukan Povray.

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

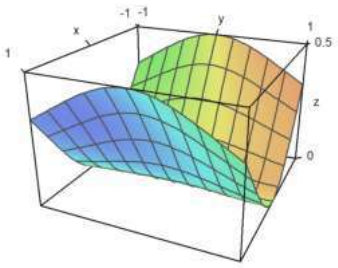


Parameter color mengubah warna permukaan. Warna garis level juga dapat diubah.

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

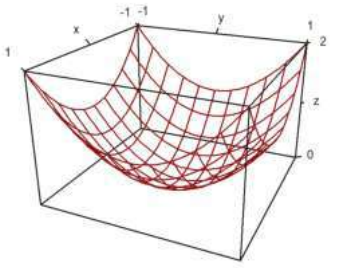
Warna 0 memberikan efek pelangi khusus.

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



Permukaan juga dapat dibuat transparan.

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



Plot Implisit

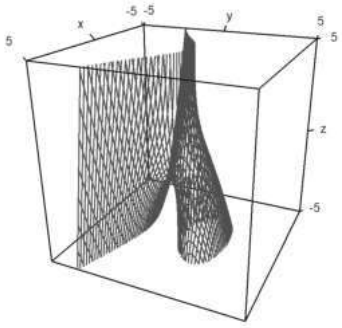
Terdapat juga plot implisit dalam tiga dimensi. Euler menghasilkan irisan melalui objek. Fitur dari plot3d mencakup plot implisit. Plot ini menampilkan himpunan nol dari sebuah fungsi dalam tiga variabel.
Solusi dari

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

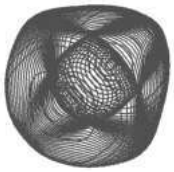
- implicit=1: irisan sejajar dengan bidang y-z
- implicit=2: irisan sejajar dengan bidang x-z
- implicit=4: irisan sejajar dengan bidang x-y

Nilai-nilai ini dapat dijumlahkan jika diinginkan. Dalam contoh, kita memplot

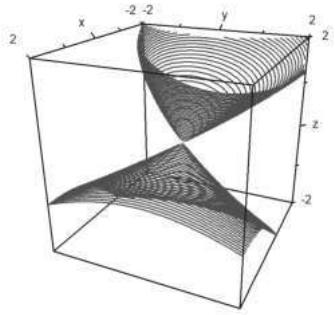
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```




```
>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,<frame,>im
```



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



Memplot Data 3D

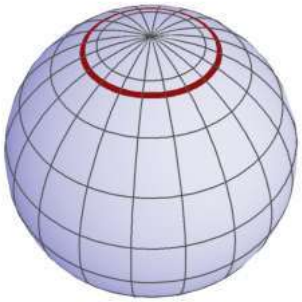
Sama seperti plot2d, plot3d menerima data. Untuk objek 3D, Anda perlu menyediakan matriks nilai x, y, dan z, atau tiga fungsi atau ekspresi fx(x,y), fy(x,y), fz(x,y).

Karena x, y, z berupa matriks, kita mengasumsikan bahwa (t,s) berjalan melalui grid persegi. Sebagai hasilnya, Anda dapat memplot citra dari persegi panjang di ruang.

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

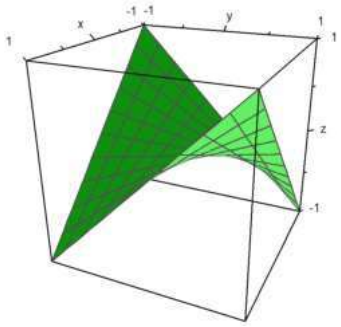
Dalam contoh berikut, kita menggunakan sebuah vektor nilai t dan sebuah vektor kolom nilai s untuk memparametrisasi permukaan bola. Dalam gambar, kita dapat menandai daerah, dalam kasus ini daerah kutub.

```
>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°):
```



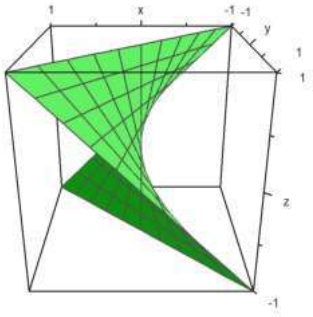
Berikut adalah sebuah contoh, yaitu grafik dari sebuah fungsi.

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



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

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



Dengan usaha lebih, kita dapat menghasilkan banyak permukaan.

Dalam contoh berikut, kita membuat tampilan berarsir dari sebuah bola terdistorsi. Koordinat biasa untuk bola adalah

dengan

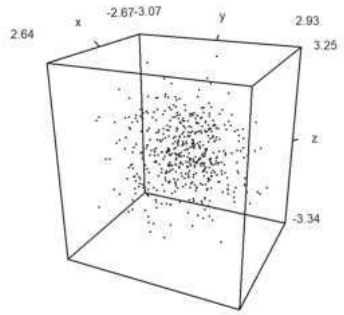
Kita mendistorsi ini dengan sebuah faktor

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

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

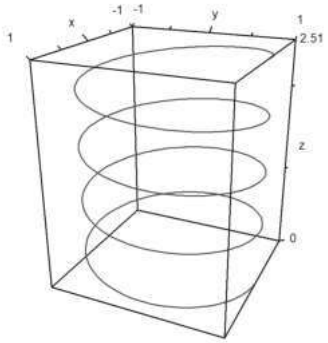
Gaya yang digunakan sama seperti pada plot2d dengan points=true;

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

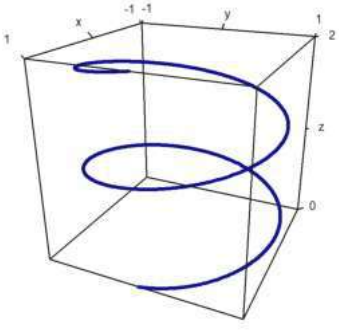


Plot sebuah kurva dalam 3D juga dimungkinkan. Dalam hal ini, lebih mudah untuk menghitung terlebih dahulu titik-titik kurva tersebut. Untuk kurva di bidang, kita menggunakan sebuah urutan koordinat dan parameter wire=true.

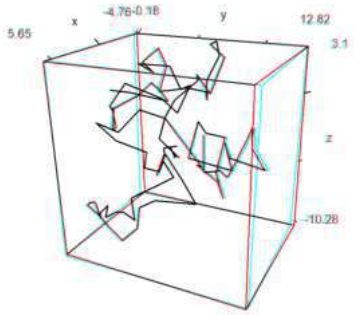
```
>t=linspace(0,8pi,500); ...
 plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

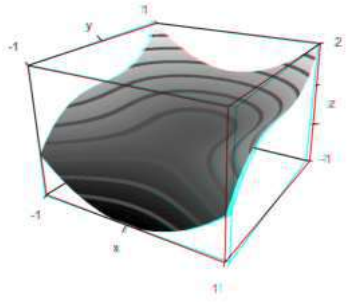


```
>X=cumsum(normal(3,100)); ...
  plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



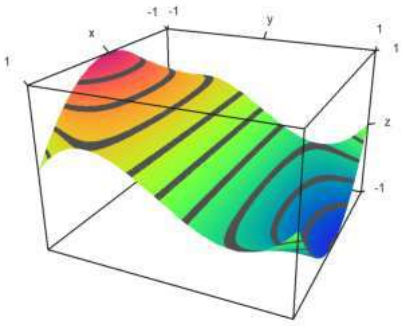
EMT juga dapat memplot dalam mode anaglyph. Untuk melihat plot semacam itu, Anda memerlukan kacamata merah/cyan.

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



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

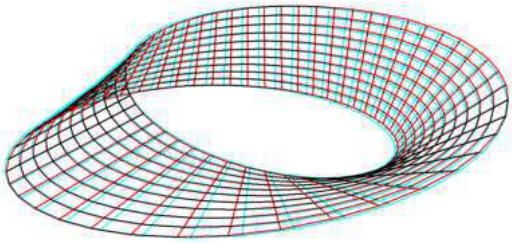
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler juga dapat memplot permukaan terparametrisasi, ketika parameter adalah nilai x, y, dan z dari citra sebuah grid persegi panjang di ruang.

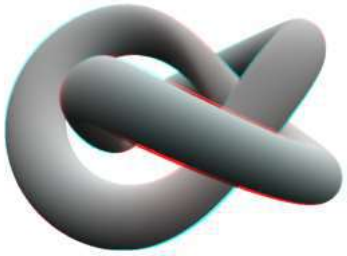
Untuk demo berikut, kita menyiapkan parameter u dan v, lalu menghasilkan koordinat ruang dari parameter tersebut.

```
>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 adalah contoh yang lebih rumit, yang tampak megah dengan kacamata merah/cyan.

```
>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

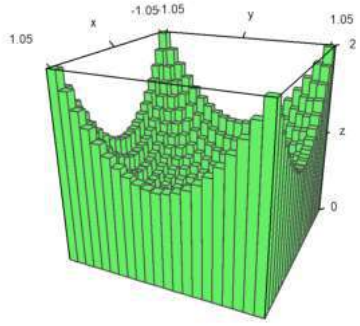
Plot batang juga dimungkinkan. Untuk ini, kita harus menyediakan

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

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

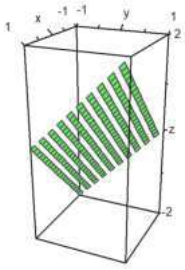
Dalam contoh, kita terlebih dahulu menghitung nilainya. Lalu kita menyesuaikan x dan y, sehingga vektor-vektor tersebut terpusat pada nilai yang digunakan.

```
>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):
```



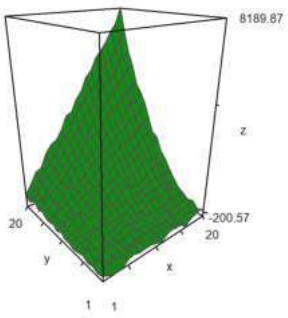
Dimungkinkan untuk membagi plot sebuah permukaan menjadi dua atau lebih bagian.

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

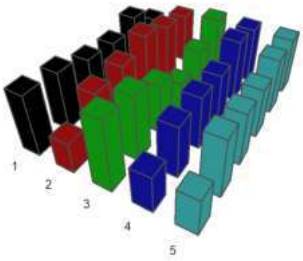


Jika memuat atau menghasilkan matriks data M dari sebuah file dan perlu memplotnya dalam 3D, Anda dapat menskalakan matriks tersebut ke [-1,1] dengan scale(M), atau menskalakan matriks dengan >zscale. Hal ini dapat dikombinasikan dengan faktor skala individual yang diterapkan secara tambahan.

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

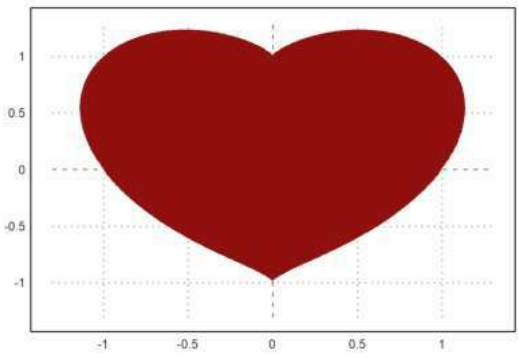



```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
 loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
 columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



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
```

$$(y^2+x^2-1)^3-x^2y^3$$

Kita ingin memutar kurva hati di sekitar sumbu y. Berikut ekspresi yang mendefinisikan hati:

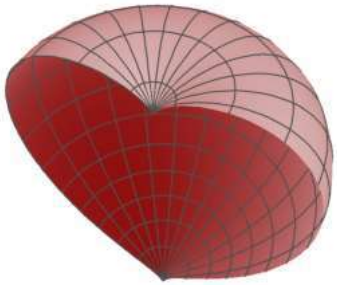
Selanjutnya kita atur

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

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

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

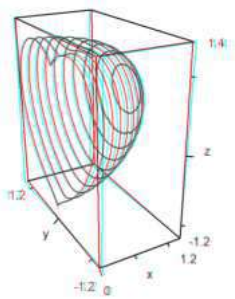
```
>function map f(a) := bisect("fr",0,2;a); ...
 t=linspace(-pi/2,pi/2,100); r=f(t);  ...
 s=linspace(pi,2pi,100)'; ...
 plot3d(r*cos(t)*sin(s),r*cos(t)*cos(s),r*sin(t), ...
 >hue,<frame,color=red,zoom=4,amb=0,max=0.7,grid=12,height=50°):
```



Berikut adalah plot 3D dari gambar di atas yang diputar di sekitar sumbu z. Kita mendefinisikan fungsi yang menggambarkan objek tersebut.

```
>function f(x,y,z) ...
 r=x^2+y^2;
 return (r+z^2-1)^3-r*z^3;
  endfunction
```

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



Plot 3D Khusus

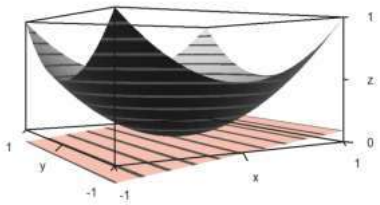
Fungsi plot3d memang berguna, tetapi tidak memenuhi semua kebutuhan. Selain rutinitas dasar, dimungkinkan untuk mendapatkan plot berbingkai dari objek apa pun yang diinginkan.

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

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

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=3):
```

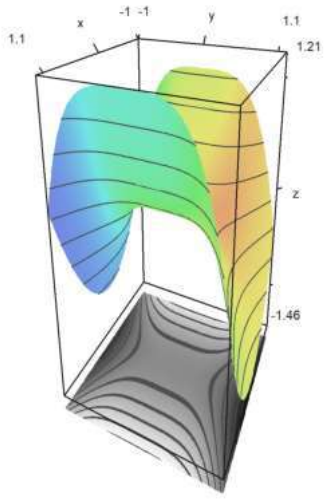


Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual. Perlu dicatat bahwa plot3d() secara default mengatur jendela ke fullwindow(), tetapi plotcontourplane() mengasumsikan hal tersebut.

```
>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):
```



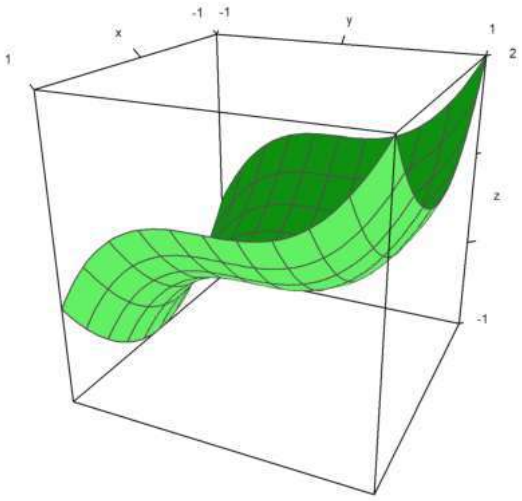
Animasi

Euler dapat menggunakan frame untuk menghitung animasi sebelumnya.

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

Pelajari kode sumber dari rotate untuk melihat lebih banyak detail.

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



Menggambar Povray

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

Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/, dan menempatkan sub-direktori "bin" Povray ke dalam environment path, atau mengatur variabel defaultpovray dengan path lengkap yang mengarah 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 menghasilkan file PNG, yang dapat dimuat oleh Euler ke dalam notebook. Untuk membersihkan file-file ini, gunakan povclear().

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

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

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

Perlu dicatat bahwa universe Povray menggunakan sistem koordinat berbeda. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray, sehingga Anda tetap dapat berpikir dalam sistem koordinat Euler dengan z mengarah ke atas, dan sumbu x, y, z menggunakan aturan tangan kanan.

Anda perlu memuat file povray tersebut.

```
>load povray;
```

Pastikan direktori bin Povray ada dalam path. Jika tidak, ubah variabel berikut agar berisi path ke executable Povray.

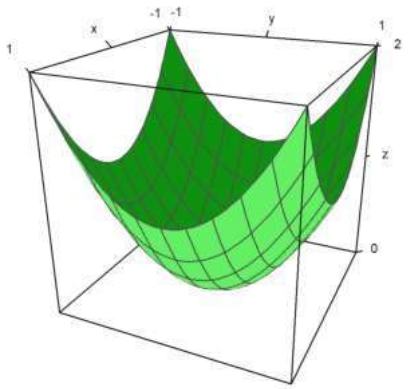
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

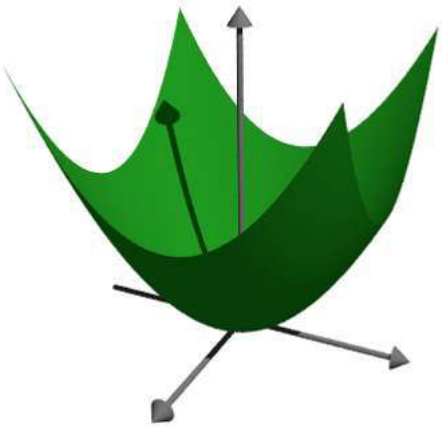
Untuk kesan pertama, kita memplot sebuah fungsi sederhana. Perintah berikut menghasilkan file Povray di direktori pengguna Anda, dan menjalankan Povray untuk melakukan ray tracing pada file tersebut.

Jika Anda menjalankan perintah berikut, GUI Povray seharusnya terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan diminta apakah ingin mengizinkan file exe dijalankan. Anda dapat menekan cancel untuk menghentikan pertanyaan lebih lanjut. Anda mungkin perlu menekan OK di jendela Povray untuk mengonfirmasi dialog start-up Povray.

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

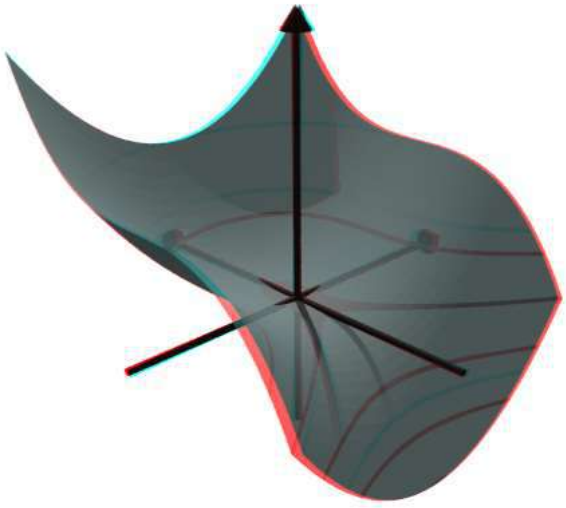



```
>pov3d("x^2+y^2",zoom=3);
```



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

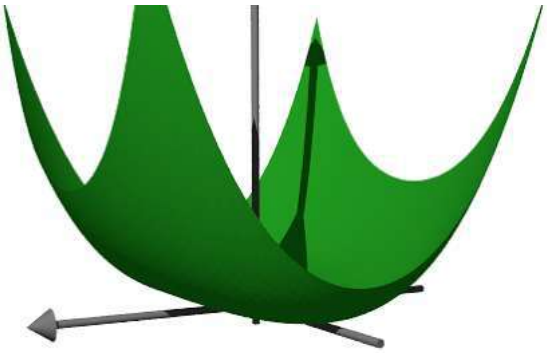
```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
   look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



Terkadang perlu untuk mencegah skala otomatis pada fungsi, dan menskalakan fungsi secara manual.

Kita memplot himpunan 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);
```



Memplot dengan Koordinat

Alih-alih menggunakan fungsi, kita dapat memplot dengan koordinat. Seperti pada plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek.

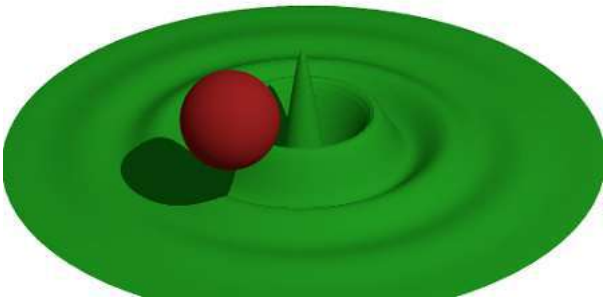
Dalam contoh, kita memutar sebuah fungsi di sekitar sumbu z.

```
>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, kita memplot sebuah gelombang teredam. Gelombang ini dibuat menggunakan bahasa matriks Euler.

Kita juga menunjukkan bagaimana objek tambahan dapat ditambahkan ke dalam sebuah scene pov3d. Untuk pembuatan objek, lihat contoh-contoh berikut. Perlu dicatat bahwa plot3d menskalakan plot agar muat ke dalam kubus satuan.

```
>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, sangat sedikit titik sudah dapat menghasilkan permukaan yang sangat halus. Hanya di tepi dan 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 permukaan adalah [x, y, Z]. Kita menghitung dua turunan terhadap x dan y dari persamaan ini, lalu mengambil hasil perkalian silang sebagai vektor normal.

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

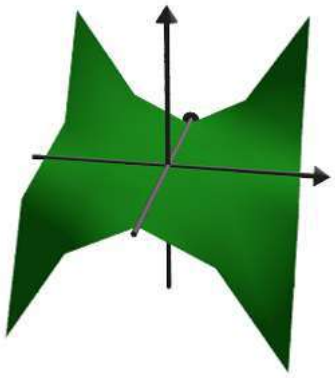
Kita mendefinisikan normal sebagai hasil perkalian silang dari turunan-turunan ini, dan mendefinisikan fungsi-fungsi koordinat.

```
>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.

```
>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 lebih baik dari ini di contoh-contoh.

Trefoil Knot

Untuk tampilan yang baik tanpa terlalu banyak titik, kita menambahkan vektor normal di sini. Kita 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 terhadap x dan y.

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

Sekarang normal, yang merupakan hasil perkalian silang dari dua turunan tersebut.

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

Sekarang kita mengevaluasi semua ini secara numerik.

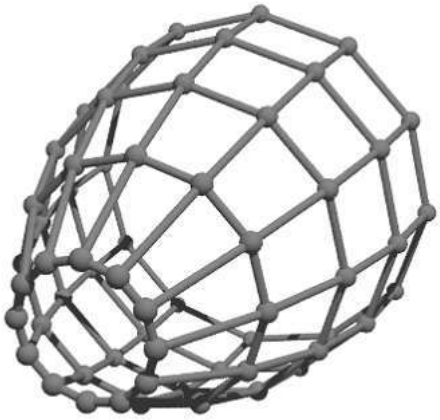
```
>x:=linspace(-%pi,%pi,40); y:=linspace(-%pi,%pi,100)';
```

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));
```

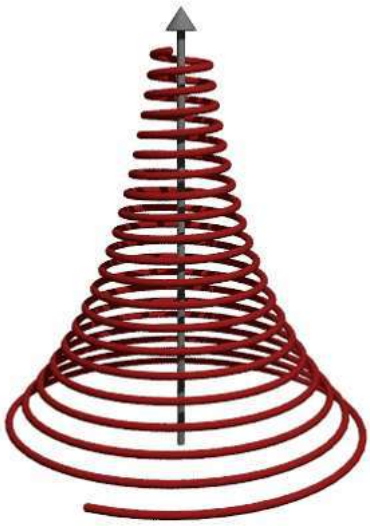
Kita juga dapat menghasilkan sebuah grid dalam 3D.

```
>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 gantinya.

```
>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 tidak perlu kita pahami 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 terlihat, kita menambahkan tekstur pada objek dengan tiga warna berbeda.

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

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

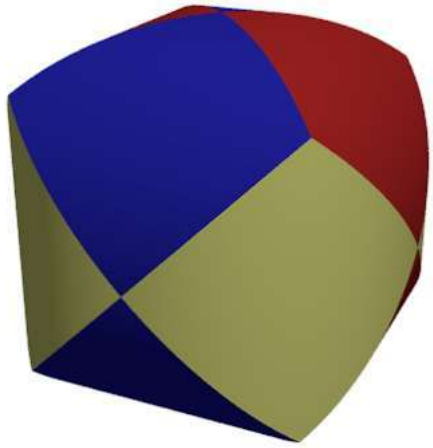
```
  texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> }  }
  finish { ambient 0.5 }
```

Sekarang kita mendefinisikan sebuah objek irisan, dan menulis hasilnya ke file.

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

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

```
>povend;
```



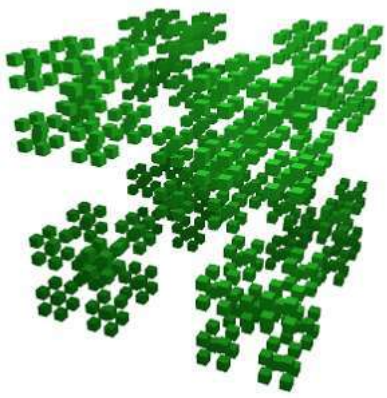
Fungsi-fungsi berikut menghasilkan sebuah fraktal secara rekursif.

Fungsi pertama menunjukkan bagaimana Euler menangani objek Povray sederhana. Fungsi povbox() mengembalikan sebuah 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();
```



Perbedaan memungkinkan pemotongan satu objek dari objek lain. Seperti halnya irisan, ini merupakan bagian dari objek CSG di Povray.

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

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

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

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

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

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

Kita menghasilkan kubus kedua, lalu memutar dan menskalanya sedikit.

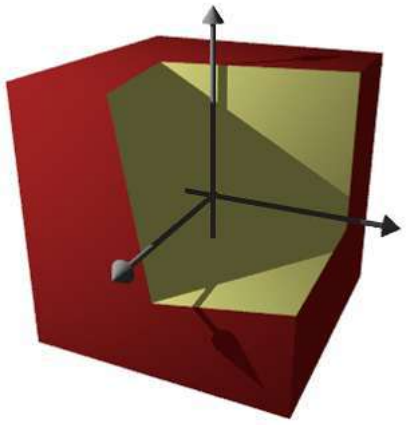
```
>c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...
   rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Kemudian kita mengambil selisih dari kedua objek tersebut.

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

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

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

Buat permukaan implisit. Perhatikan perbedaan sintaks dalam ekspresi.

```
>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 sebuah objek mesh, dan menggambarnya dengan informasi tambahan.

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

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

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

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

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

Sekarang kita mendefinisikan dua cakram, yang akan dipotong 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 tersebut.

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

Tulis kedua irisan tersebut.

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

Tulis sebuah titik pada nilai maksimum.

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

Tambahkan sumbu dan akhiri.

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

Anaglyph dalam Povray

Untuk menghasilkan sebuah anaglyph untuk kacamata merah/sian, Povray harus dijalankan dua kali dari posisi kamera yang berbeda. Hal ini menghasilkan dua file Povray dan dua file PNG, yang kemudian dimuat dengan fungsi loadanaglyph().

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

Fungsi pov3d() memiliki sebuah pengaturan sederhana untuk menghasilkan anaglyph.

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

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

```
>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 mirip dengan gabungan povstart() dan povend().

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

Mendefinisikan Objek Sendiri

Antarmuka povray dari Euler berisi banyak objek. Namun, Anda tidak terbatas pada itu saja. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau merupakan objek yang benar-benar baru.

Kita akan mendemonstrasikan sebuah torus. Perintah Povray untuk ini adalah "torus". Jadi kita mengembalikan sebuah string dengan perintah ini beserta parameternya. Perlu dicatat bahwa torus selalu terpusat di titik asal.

```
>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 adegan. Untuk tampilannya, 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();

memanggil program Povray. Namun, jika terjadi kesalahan, ia tidak menampilkan pesan error. Karena itu, Anda sebaiknya menggunakan

> povend(<exit);

jika ada sesuatu yang tidak berhasil. Perintah ini akan membuat jendela Povray tetap terbuka.

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

Berikut contoh yang lebih rumit. Kita menyelesaikan

dan menampilkan titik-titik layak serta titik optimum dalam plot 3D.

```
>A=[10,8,4;5,6,8;6,3,2;9,5,6];
>b=[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, memang ada.

Selanjutnya kita mendefinisikan dua objek. Yang pertama adalah bidang

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

Lalu kita mendefinisikan irisan 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 dapat memplot adegan 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 ini adalah sebuah lingkaran di sekitar titik optimum.

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

Dan sebuah galat pada arah menuju titik optimum.

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

Kita menambahkan teks ke layar. Teks hanyalah sebuah objek 3D. Kita perlu menempatkannya dan memutarnya sesuai dengan pandangan kita.

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

Contoh Lainnya

Anda dapat menemukan beberapa contoh lain untuk Povray dalam Euler pada file berikut.

Examples/Dandelin Spheres
Examples/Donat Math