Identitas

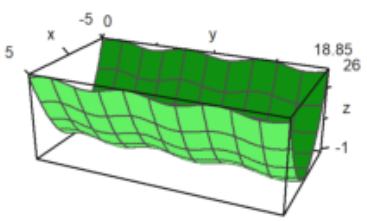
Nama : Salma Nadia Zulfa
NIM : 24030130005
Prodi/Kelas : Pendidikan Matematika A
Semester : 3

Menggambar Plot 3D dengan EMT

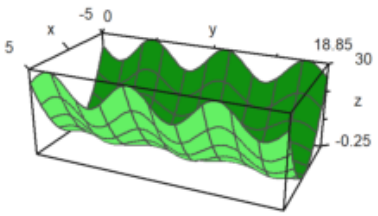
Ini adalah pengantar tentang plot 3D di Euler. Kita memerlukan plot 3D untuk memvisualisasikan sebuah fungsi dari dua variabel.
Euler menggambar fungsi semacam itu dengan menggunakan algoritma penyortiran untuk menyembunyikan bagian yang berada di latar belakang. Secara umum, Euler menggunakan proyeksi sentral. Pengaturan default berasal dari kuadran x-y positif menuju titik asal x=y=z=0, tetapi angle=0° berarti tampilan diarahkan ke sumbu y. Sudut pandang dan ketinggian dapat diubah.
Euler dapat memplot:
- permukaan dengan bayangan dan garis kontur atau rentang kontur,
- 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 (range) plot di sekitar titik (0,0).

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




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



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

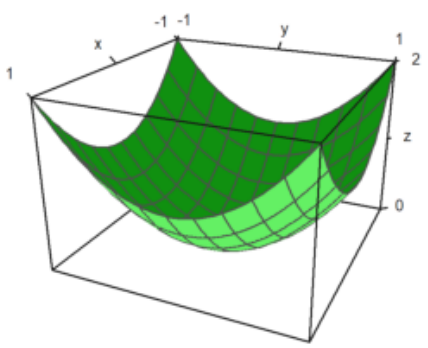
Functions of two Variables

Untuk grafik suatu fungsi, gunakan:
- ekspresi sederhana dalam x dan y,
- nama sebuah fungsi dari dua variabel,
- atau matriks data.

Pengaturan default adalah filled wire grid dengan warna berbeda pada kedua sisinya. Perlu diperhatikan bahwa jumlah interval grid bawaan adalah 10, tetapi plot menggunakan jumlah bawaan 40x40 persegi panjang untuk membentuk permukaan. Hal ini dapat diubah.
- n=40, n=[40,40] : jumlah garis grid pada setiap arah
- grid=10, grid=[10,10] : jumlah garis grid pada setiap arah

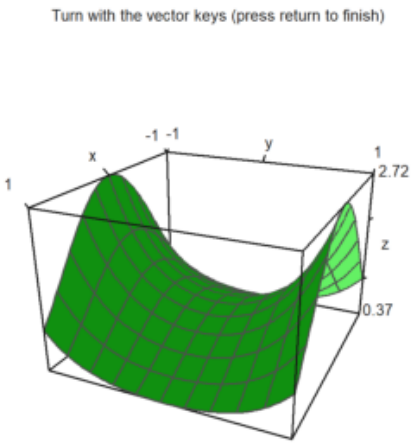
Dalam contoh ini, kita menggunakan nilai default n=40 dan grid=10.

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



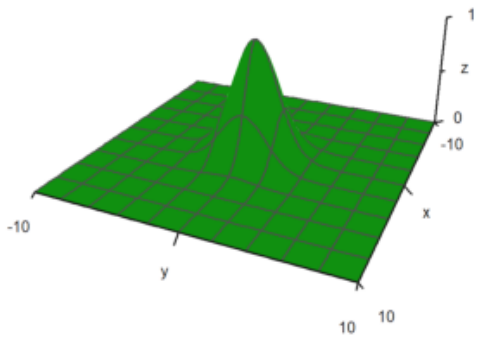
Interaksi pengguna dimungkinkan dengan parameter >user. Pengguna dapat menekan tombol berikut:
left, right, up, down : mengubah sudut pandang
+ , - : memperbesar atau memperkecil (zoom in / zoom out)
a : menghasilkan anaglyph (lihat penjelasan di bawah)
l : mengaktifkan/menonaktifkan perputaran sumber cahaya (lihat penjelasan di bawah)
space : mengatur ulang ke pengaturan default
return (Enter) : mengakhiri interaksi

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



Rentang plot untuk fungsi dapat ditentukan dengan:
a,b : rentang sumbu-x
c,d : rentang sumbu-y
r : kotak simetris di sekitar titik (0,0)
n : jumlah sub-interval untuk plot
Terdapat beberapa parameter untuk melakukan penskalaan 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 (view) dapat diubah dengan berbagai cara:
- distance : jarak pandang terhadap plot
- zoom : nilai perbesaran (zoom)
- angle : sudut terhadap sumbu-y negatif dalam radian
- height : ketinggian tampilan dalam radian

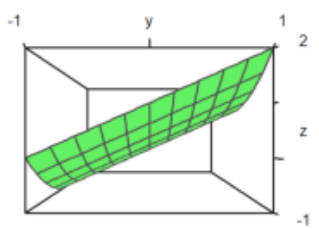
Nilai default dapat dilihat atau diubah dengan fungsi view(). Fungsi ini mengembalikan parameter sesuai urutan di atas.

```
>view
```

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

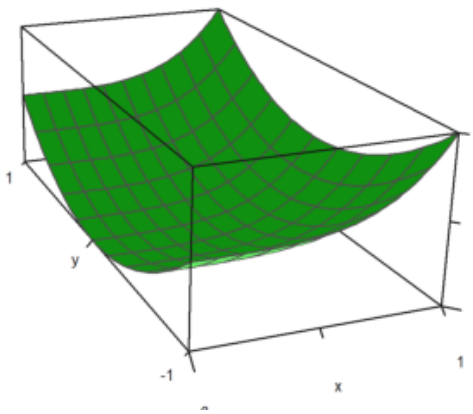
Jarak yang lebih dekat membutuhkan zoom yang lebih kecil. Efeknya mirip dengan penggunaan lensa sudut lebar (wide angle lens).
Pada contoh berikut, angle=0 dan height=0 berarti tampilan diarahkan dari sumbu-y negatif. Pada kasus ini, label sumbu-y akan disembunyikan.

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



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

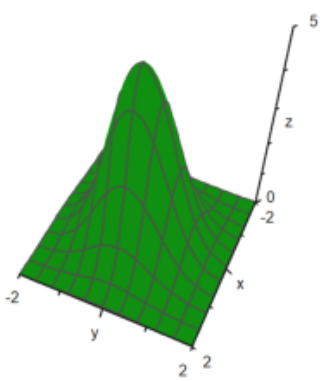
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>  center=[0.4,0,0],zoom=5):
```



Plot diskalakan agar sesuai dengan unit cube untuk tampilan. Jadi, tidak perlu mengubah jarak pandang (distance) atau perbesaran (zoom) tergantung pada ukuran plot. Namun, label tetap merujuk pada ukuran sebenarnya.

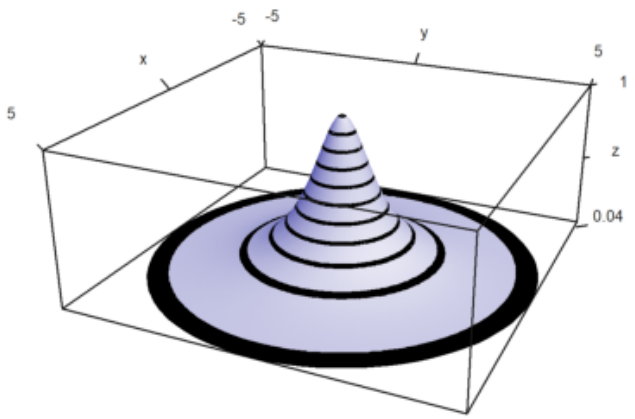
Jika penskalaan ini dimatikan dengan scale=false, maka Anda perlu memastikan plot tetap muat dalam jendela tampilan dengan cara mengubah jarak pandang atau zoom, serta memindahkan pusat (center).

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

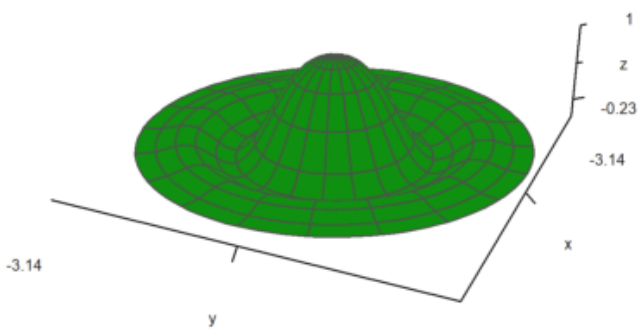


Plot polar juga tersedia. Parameter polar=true digunakan untuk menggambar plot polar. Fungsi yang digunakan tetap harus merupakan fungsi dari x dan y. Parameter fscale akan menskalakan fungsi dengan skala tersendiri. Jika tidak, fungsi akan diskalakan agar sesuai dengan sebuah kubus.

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

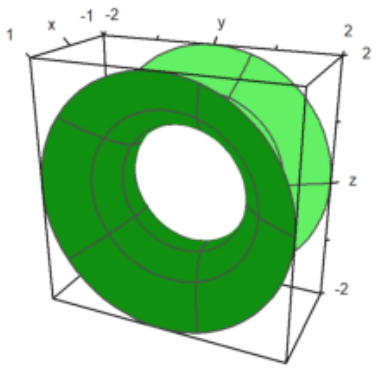



```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```

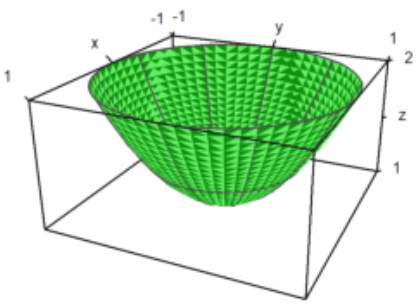


Parameter rotate digunakan untuk memutar suatu fungsi terhadap sumbu-x.
- rotate=1 : menggunakan sumbu-x
- rotate=2 : menggunakan sumbu-z

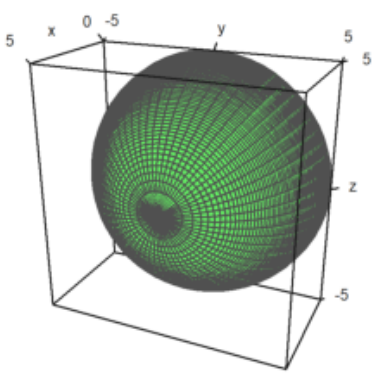
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



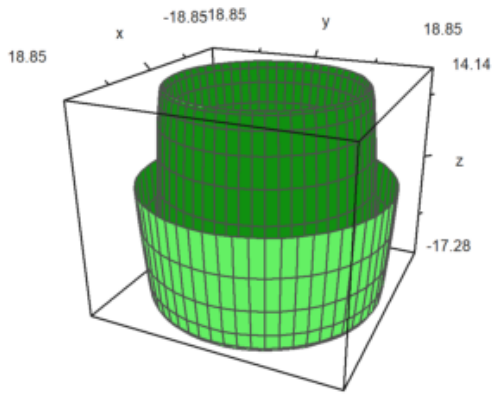

```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




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

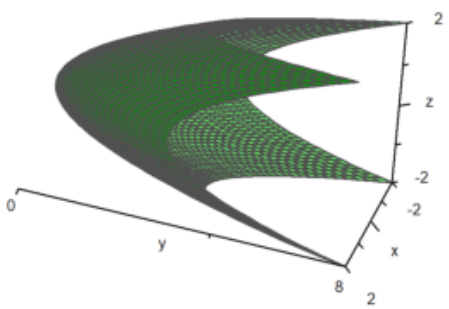


```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Berikut adalah sebuah plot dengan tiga fungsi.

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

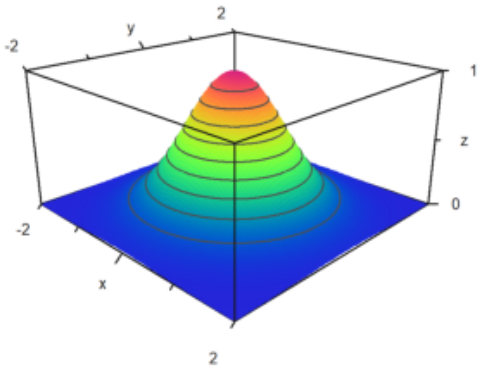


Untuk membuat plot, Euler menambahkan garis grid. Sebagai gantinya, dimungkinkan untuk menggunakan garis kontur (level lines) dengan satu warna (one-color hue) atau dengan spektrum warna (spectral colored hue). Euler juga dapat menggambar ketinggian fungsi pada sebuah plot dengan bayangan (shading). Pada semua plot 3D, Euler dapat menghasilkan anaglyph merah/sian.

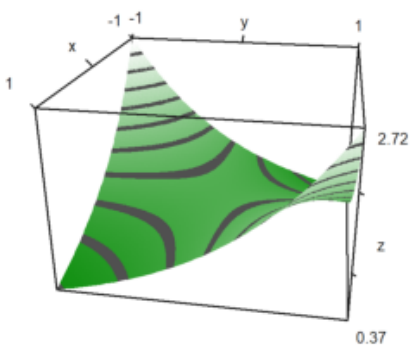
>hue : mengaktifkan bayangan cahaya sebagai pengganti garis grid.
>contour : menggambar garis kontur otomatis pada plot.
level=... (atau levels) : vektor nilai untuk garis kontur.

Pengaturan default adalah level="auto", yang secara otomatis menghitung beberapa garis kontur. Seperti terlihat pada plot, level sebenarnya berupa rentang nilai level.
Gaya default dapat diubah. Untuk plot kontur berikut, digunakan grid yang lebih rapat yaitu 100x100 titik, dilakukan penskalaan fungsi dan plot, serta digunakan sudut pandang (angle of view) 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°):
```



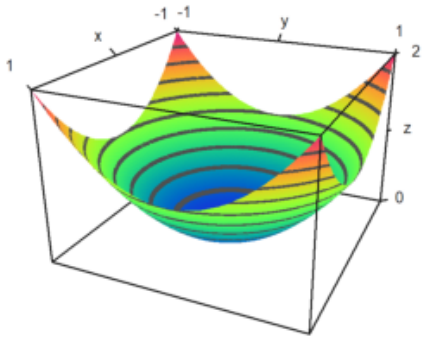

```
>plot3d("exp(x*y)",angle=100°,>contour,color=green):
```



Bayangan (shading) default menggunakan warna abu-abu. Namun, rentang warna spektral juga tersedia.
- >spectral : menggunakan skema spektral bawaan.
- color=... : menggunakan warna khusus atau skema spektral.

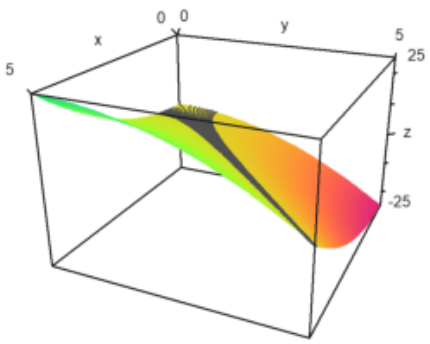
Untuk plot berikut, digunakan skema spektral bawaan dan jumlah titik diperbanyak agar tampilan menjadi sangat halus.

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



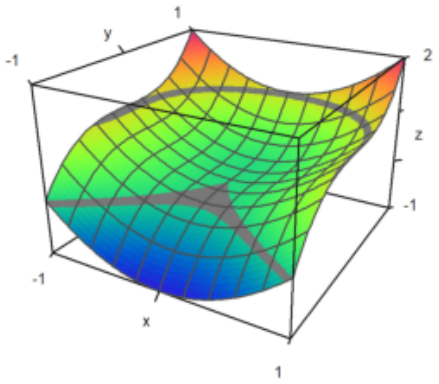
Alih-alih menggunakan garis level otomatis, kita juga dapat menetapkan nilai tertentu untuk garis level. Hal ini akan menghasilkan garis level tipis, bukan rentang level.

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



Pada plot berikut, digunakan dua pita level yang sangat lebar, yaitu dari -0.1 hingga 1 dan dari 0.9 hingga 1. Hal ini dimasukkan sebagai sebuah matriks dengan batas level sebagai kolom.
Selain itu, kita juga 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):
```

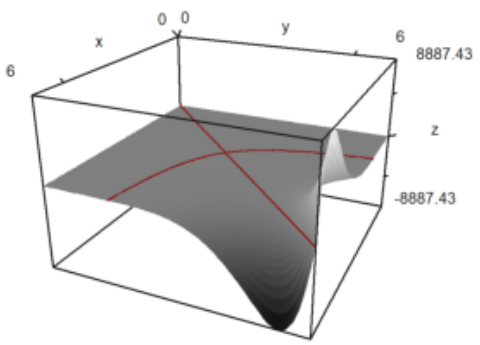


Pada contoh berikut, kita memplot himpunan, di mana

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

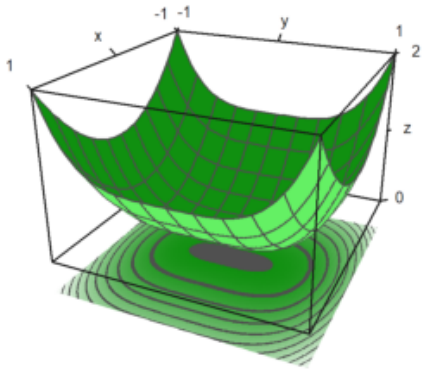
Kita menggunakan satu garis tipis sebagai garis level (level line).

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



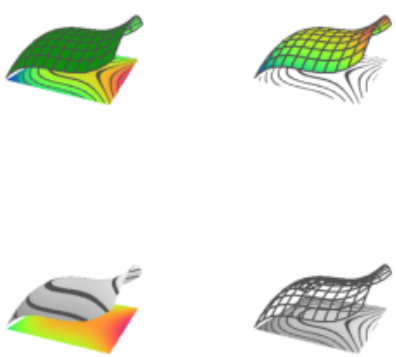
Dimungkinkan untuk menampilkan sebuah bidang kontur (contour plane) di bawah plot. Warna dan jaraknya terhadap plot dapat ditentukan.

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



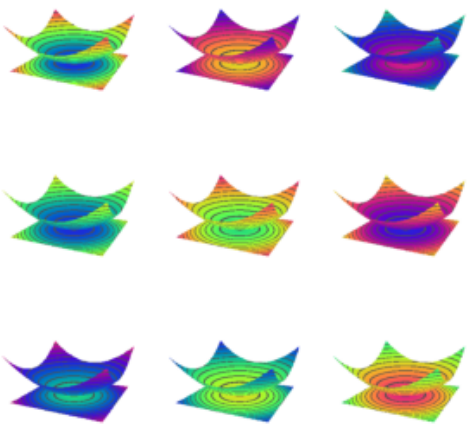
Berikut adalah beberapa gaya tambahan. Kita selalu menonaktifkan bingkai (frame), serta menggunakan berbagai skema warna untuk plot dan grid.

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



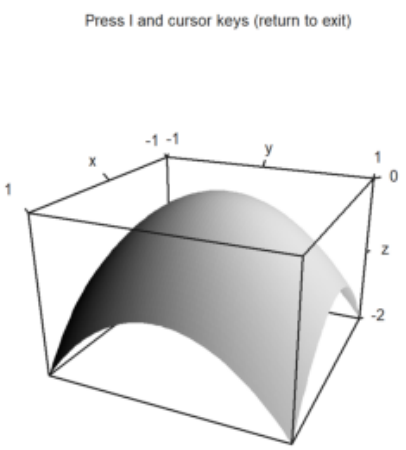
Terdapat beberapa skema spektral lain yang diberi nomor dari 1 hingga 9. Namun, Anda juga dapat menggunakan color=value, dengan nilai sebagai berikut:
- spectral : untuk rentang warna dari biru hingga merah
- white : untuk rentang warna yang lebih lembut
- yellowblue, purplegreen, blueyellow, greenred
- blueyellow, greenpurple, yellowblue, redgreen

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



Sumber cahaya dapat diubah dengan tombol l dan tombol kursor saat interaksi pengguna. Sumber cahaya juga dapat diatur menggunakan parameter:
- light : arah cahaya
- amb : cahaya ambien dengan nilai antara 0 hingga 1
Perlu diperhatikan bahwa program tidak membedakan sisi-sisi plot. Tidak ada bayangan (shadows) yang ditampilkan. Untuk menambahkan bayangan, Anda memerlukan Povray.

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



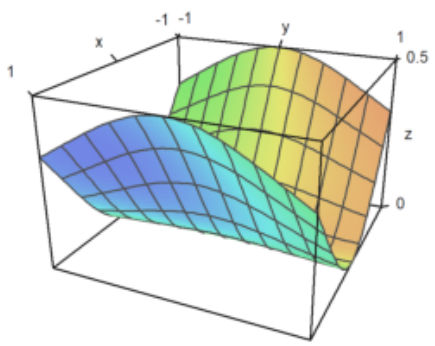
Parameter color digunakan untuk mengubah warna permukaan. Warna garis level (level lines) juga dapat diubah.

```
>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):
```



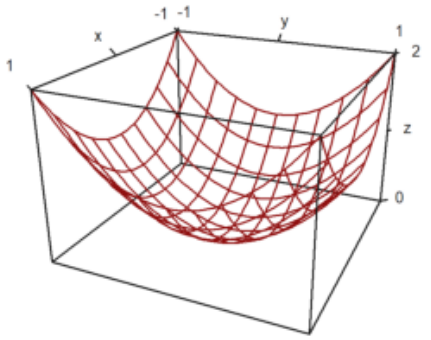
Warna 0 memberikan efek pelangi khusus (rainbow effect).

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



Permukaan juga dapat dibuat transparan.

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



Implicit Plots

Terdapat juga implicit plots dalam tiga dimensi. Euler menghasilkan potongan (cuts) melalui objek. Fitur plot3d mencakup implicit plots. Plot ini menampilkan himpunan nol (zero set) dari suatu fungsi dengan tiga variabel.
Solusi dari persamaan

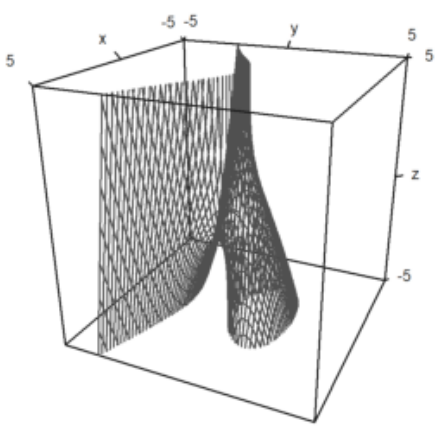
$$f(x, y, z) = 0$$

dapat divisualisasikan dalam potongan yang sejajar dengan bidang x-y, x-z, dan y-z.
- implicit=1 : potongan sejajar dengan bidang y-z
- implicit=2 : potongan sejajar dengan bidang x-z
- implicit=4 : potongan sejajar dengan bidang x-y

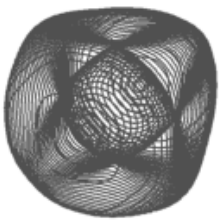
Nilai-nilai ini dapat dijumlahkan jika diperlukan. Pada contoh berikut, kita memplot

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

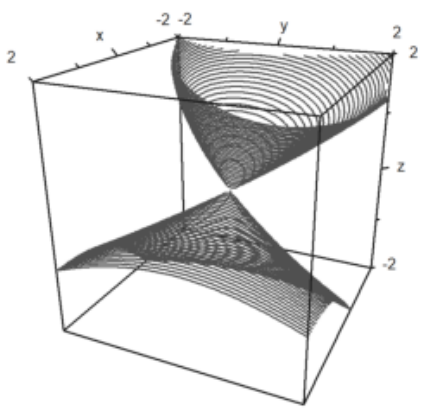
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```



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



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



Plotting 3D Data

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

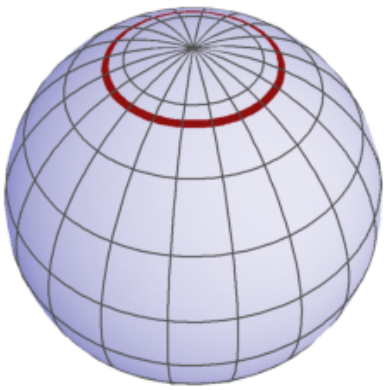
$$\gamma(t,s) = (x(t,s), y(t,s), z(t,s))$$

Karena x,y,z berupa matriks, maka diasumsikan bahwa (t,s) berjalan melalui grid berbentuk persegi. Hasilnya, Anda dapat memplot gambar persegi panjang di ruang 3D.

Anda dapat menggunakan bahasa matriks Euler untuk menghasilkan koordinat dengan lebih efisien.

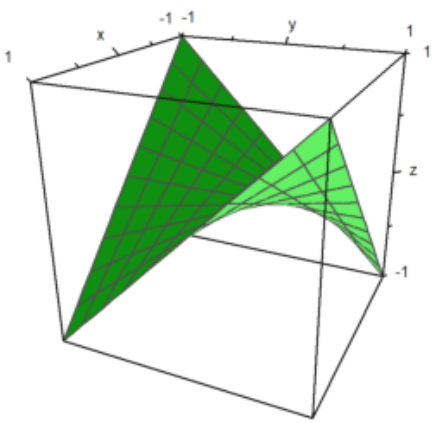
Pada contoh berikut, kita menggunakan sebuah vektor nilai t dan sebuah vektor kolom nilai s untuk memparametrisasi permukaan bola. Pada gambar, kita juga dapat menandai wilayah tertentu — dalam kasus ini, wilayah kutub (polar region).

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



Berikut adalah sebuah contoh, yaitu grafik dari sebuah fungsi.

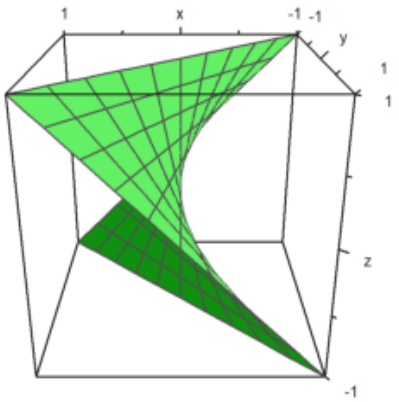
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



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

$$x = y\,z$$

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



Dengan sedikit usaha lebih, kita dapat menghasilkan banyak permukaan.
ada contoh berikut, kita membuat tampilan berbayang (shaded view) dari sebuah bola yang terdistorsi.
Koordinat biasa untuk bola adalah:

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

dengan

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

Kemudian kita mendistorsi bola tersebut dengan sebuah faktor.

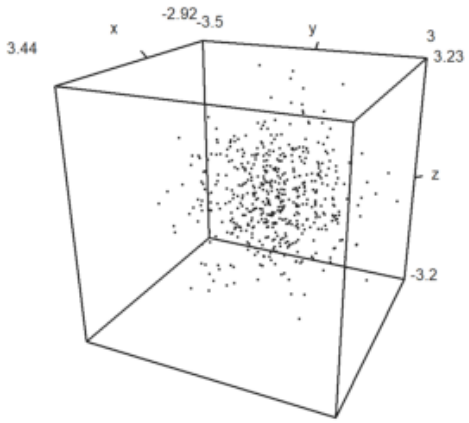
$$d(t,s) = \frac{\cos(4t) + \cos(8s)}{4}.$$

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



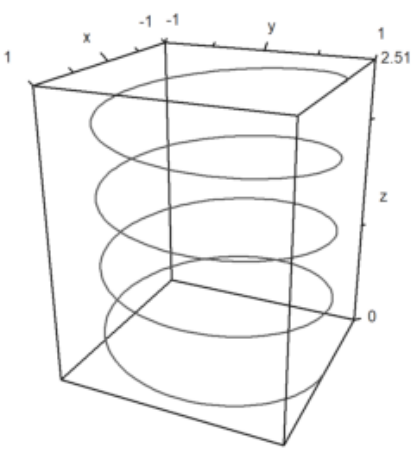
Tentu saja, point cloud juga dimungkinkan. Untuk memplot data berupa titik di dalam ruang, kita memerlukan tiga vektor sebagai koordinat titik-titik tersebut.
Gaya (styles) yang digunakan sama seperti pada plot2d dengan opsi points=true.

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

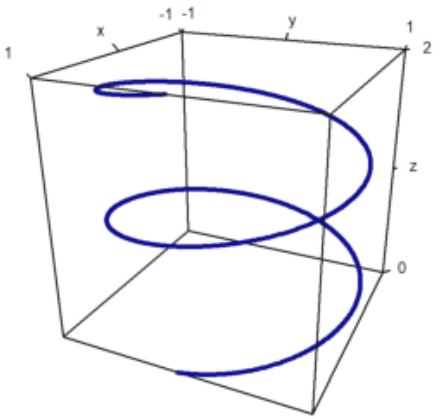


Kita juga dapat memplot sebuah kurva dalam 3D. Dalam kasus ini, lebih mudah jika titik-titik kurva dihitung terlebih dahulu. Untuk kurva pada bidang, kita menggunakan sebuah urutan koordinat serta parameter wire=true.

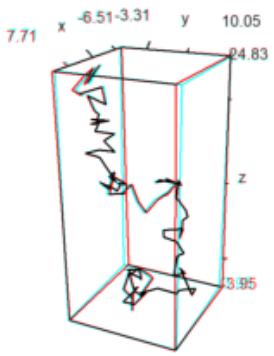
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

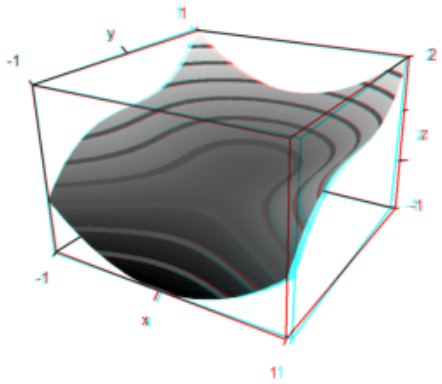



```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



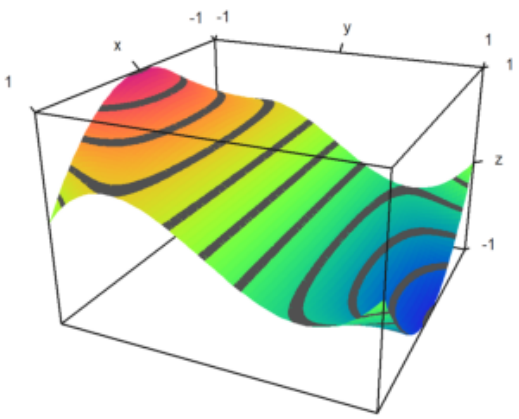
EMT juga dapat membuat plot dalam mode anaglyph. Untuk melihat plot semacam ini, Anda memerlukan kacamata merah/sian (red/cyan glasses).

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



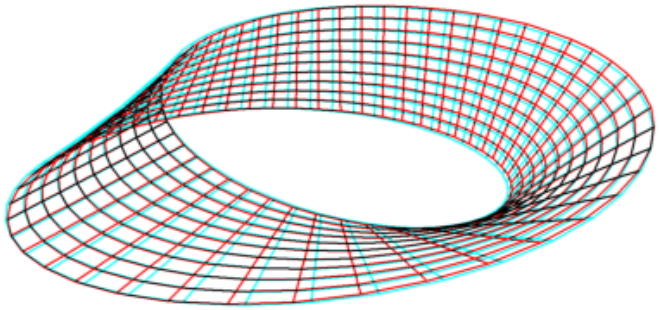
Sering kali, skema warna spektral digunakan untuk plot. Hal ini menekankan ketinggian dari fungsi yang ditampilkan.

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



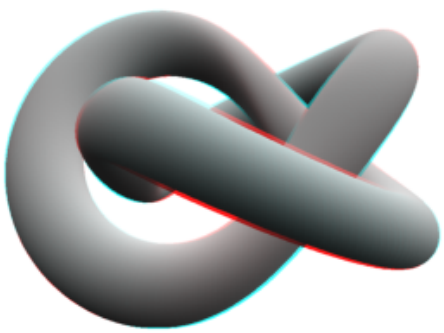
Euler juga dapat memplot permukaan terparametrisasi, ketika parameternya adalah nilai x-, y-, dan z- dari sebuah citra grid persegi panjang di dalam ruang.
Untuk demo berikut, kita menyiapkan parameter u- dan v-, lalu menghasilkan koordinat ruang dari parameter tersebut.

```
>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):
```



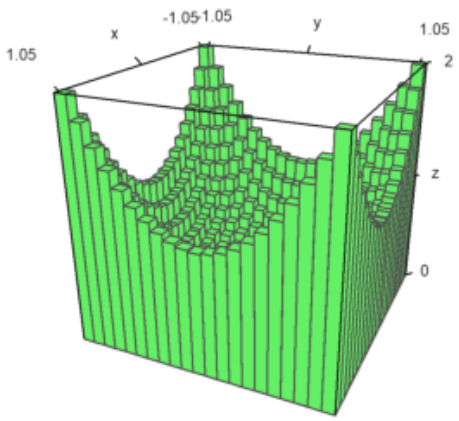
Berikut adalah contoh yang lebih rumit, yang terlihat megah jika dilihat dengan kacamata merah/sian (red/cyan glasses).

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



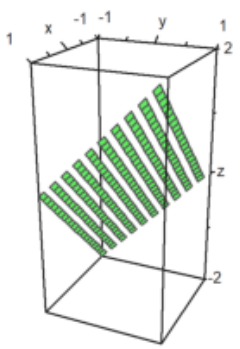
Plot batang (bar plots) juga dimungkinkan. Untuk ini, kita perlu menyediakan:
x : vektor baris dengan n+1 elemen
y : vektor kolom dengan n+1 elemen
z : matriks n×n berisi nilai-nilai
Matriks z dapat lebih besar, tetapi hanya nilai sebanyak
n×n yang akan digunakan.
Pada contoh, pertama-tama kita menghitung nilai-nilainya. Kemudian 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):
```



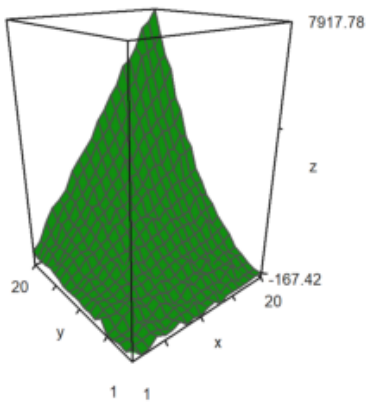
Dimungkinkan untuk membagi plot sebuah permukaan menjadi dua bagian atau lebih.

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

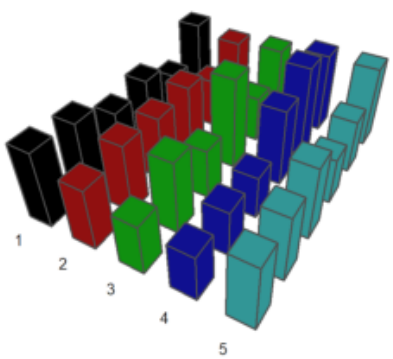


Jika Anda memuat atau menghasilkan sebuah matriks data M dari file dan ingin memplotnya dalam 3D, Anda dapat menskalakan matriks tersebut ke rentang [-1,1] dengan scale(M), atau menskalakannya dengan >zscale. Hal ini dapat dikombinasikan dengan faktor skala individual yang diterapkan secara tambahan.

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

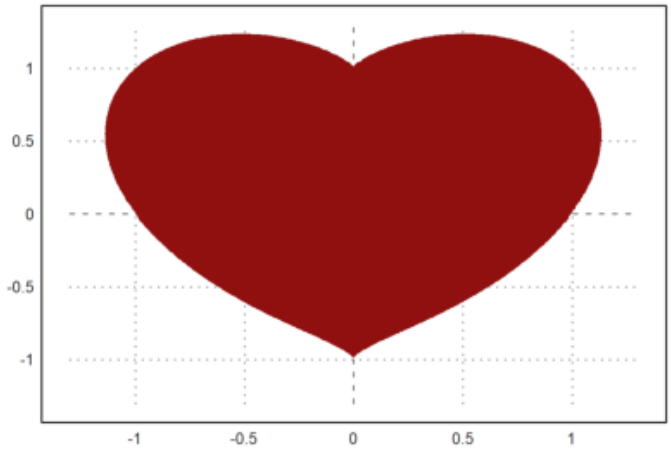



```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



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
```

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

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

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

Selanjutnya kita set

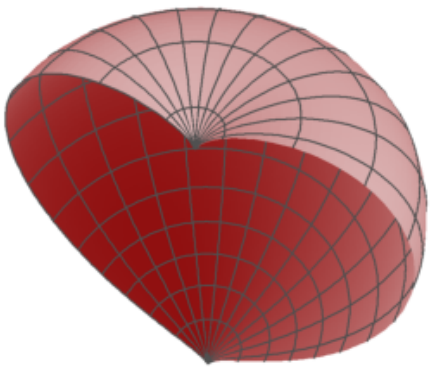
$$x=r.cos(a),\quad y=r.sin(a).$$

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

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

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

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

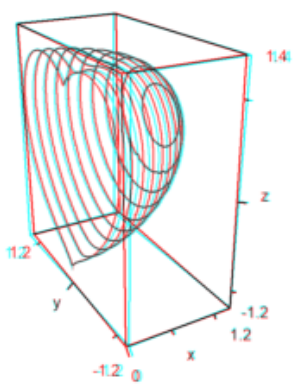


Berikut adalah plot 3D dari gambar di atas yang diputar mengelilingi sumbu-z. Kita mendefinisikan fungsi yang menggambarkan objek tersebut.

```
>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

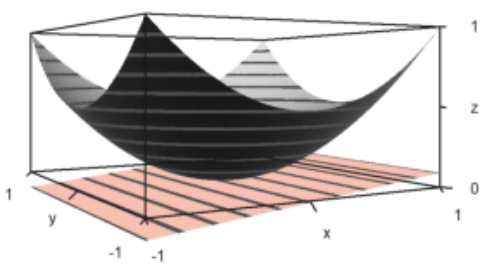
Fungsi plot3d memang sangat berguna, tetapi tidak selalu dapat memenuhi semua kebutuhan. Selain rutinitas dasar lainnya, dimungkinkan juga untuk membuat plot berbingkai (framed plot) dari objek apa pun yang diinginkan.
Meskipun Euler bukanlah program 3D penuh, ia dapat menggabungkan beberapa objek dasar. Dalam contoh ini, 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() digunakan untuk menyediakan bingkai (frames), serta mengatur tampilan (views).

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

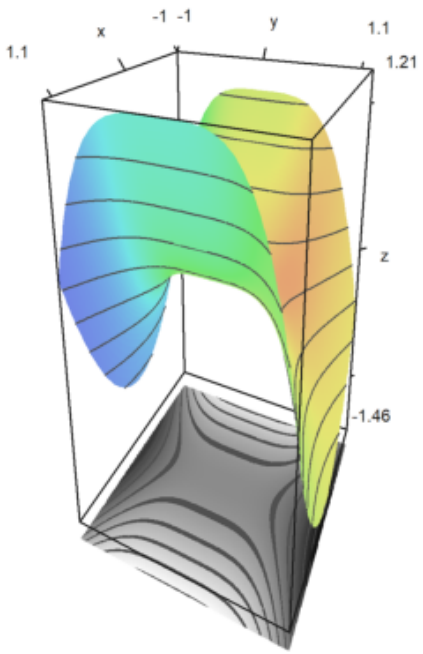


Dengan cara yang sama, Anda dapat memplot bidang kontur (contour plane) secara manual. Perlu diperhatikan bahwa plot3d() secara bawaan (default) mengatur jendela ke fullwindow(), sedangkan plotcontourplane() mengasumsikan hal tersebut.

```
>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):
```

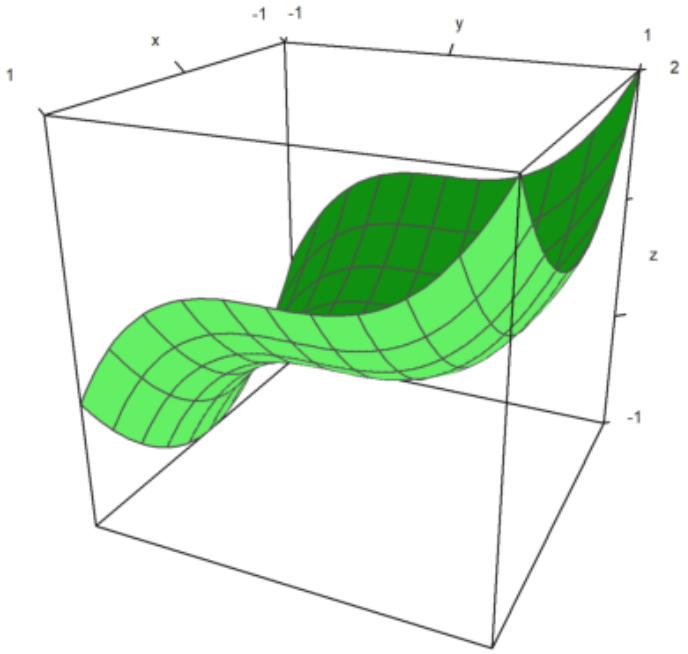


Animasi

Euler dapat menggunakan frame untuk melakukan pra-perhitungan (pre-compute) animasi.
Salah satu fungsi yang menggunakan teknik ini adalah rotate. Fungsi ini dapat mengubah sudut pandang dan menggambar ulang plot 3D. Fungsi tersebut memanggil addpage() untuk setiap plot baru. Terakhir, fungsi ini menganimasikan plot-plot tersebut.

Silakan pelajari source code dari rotate untuk melihat lebih banyak detail.

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



Menggambar Povray

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

Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/ lalu menambahkan sub-direktori "bin" dari Povray ke dalam environment path, atau menetapkan variabel "defaultpovray" dengan path penuh yang menunjuk ke "pvengine.exe".

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

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

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

Fungsi objek memiliki parameter bernama "look", yang membutuhkan sebuah string dengan kode Povray untuk texture dan finish dari objek. Fungsi povlook() dapat digunakan untuk menghasilkan string ini. Fungsi tersebut memiliki parameter untuk warna, transparansi, Phong Shading, dan sebagainya.

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

Anda perlu memuat file povray terlebih dahulu.

```
>load povray;
```

Pastikan direktori bin dari Povray sudah ada di dalam path.
ika belum, ubahlah variabel berikut sehingga berisi path menuju file executable Povray.

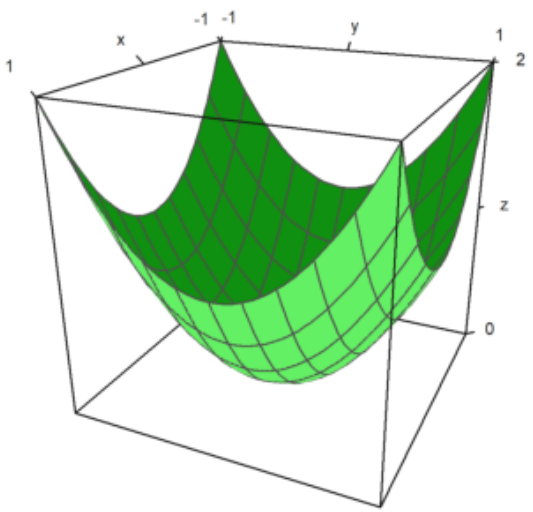
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

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

Jika Anda menjalankan perintah berikut, GUI Povray akan terbuka, menjalankan file tersebut, lalu menutup secara otomatis.
Karena alasan keamanan, Anda akan ditanya apakah ingin mengizinkan file .exe dijalankan. Anda dapat menekan Cancel untuk menghentikan pertanyaan lebih lanjut. Anda mungkin juga perlu menekan OK di jendela Povray untuk mengonfirmasi dialog awal Povray.

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




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

```
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

Terkadang perlu untuk mencegah penskalaan otomatis pada fungsi, dan melakukan penskalaan fungsi secara manual.
Kita memplot himpunan titik pada bidang kompleks, di mana hasil kali jarak terhadap 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);
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

Plotting dengan Koordinat

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

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

Pada contoh berikut, kita memplot sebuah gelombang redam (damped wave). Gelombang ini dibuat menggunakan bahasa matriks Euler.

Kita juga menunjukkan bagaimana sebuah objek tambahan dapat ditambahkan ke scene pov3d. Untuk pembuatan objek, lihat contoh-contoh berikutnya. Perlu diperhatikan bahwa plot3d akan menskalakan plot agar sesuai dengan unit cube.

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

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

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

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

$$x^2 y^3$$

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

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

Kita mendefinisikan normal sebagai hasil cross product dari turunan-turunan tersebut, dan mendefinisikan fungsi koordinatnya.

```
>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);
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

Berikut adalah Trefoil knot yang dibuat oleh A. Busser di Povray. Terdapat versi yang lebih baik dari ini dalam contoh-contoh.

See: Examples\Trefoil Knot | Trefoil Knot

Untuk tampilan yang bagus tanpa terlalu banyak titik, kita menambahkan normal vectors di sini. Kita menggunakan Maxima untuk menghitung normals tersebut. Pertama, dibuat 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, dibuat dua vektor turunan terhadap x dan y.

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

Sekarang dibuat normal, yang merupakan hasil cross product dari kedua turunan tersebut.

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

Sekarang kita evaluasi semua ini secara numerik.

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

Normal vectors adalah hasil evaluasi dari ekspresi simbolik dn[i] untuk 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));
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

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();
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Dengan povgrid(), pembuatan 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();
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

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 seterusnya, hanya mengembalikan vektor [1,0,0], yang bisa digunakan sebagai alternatif.

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

String tersebut berisi kode Povray, yang pada saat ini tidak perlu kita pahami.

```
>c2
```

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

Seperti yang Anda lihat, kita menambahkan texture pada objek dengan tiga warna berbeda.
Hal ini dilakukan menggunakan povlook(), yang mengembalikan string berisi kode Povray yang relevan. Kita dapat menggunakan warna bawaan Euler, atau mendefinisikan warna sendiri. Kita juga bisa menambahkan transparansi, atau mengubah cahaya ambien (ambient light).

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

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

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

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

Irisan (intersection) dari tiga silinder sulit divisualisasikan, jika Anda belum pernah melihatnya sebelumnya.

```
>povend;
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Fungsi-fungsi berikut menghasilkan sebuah fractal secara rekursif.
ungsi pertama menunjukkan bagaimana Euler menangani objek Povray sederhana. Fungsi povbox() mengembalikan sebuah string yang berisi koordinat kotak (box), texture, dan finish-nya.

```
>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();
```

```
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

Differences memungkinkan pemotongan satu objek dari objek lain. Seperti halnya intersections, ini merupakan bagian dari objek CSG (Constructive Solid Geometry) 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 langsung ditulis ke dalam file.
Koordinat kotak -1 hanya berarti [-1, -1, -1].

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

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

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

Kita membuat sebuah kubus kedua, kemudian memutarnya dan sedikit menskalanya.

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

Kemudian kita mengambil difference dari kedua objek tersebut.

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

Sekarang tambahkan tiga sumbu (axes).

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Fungsi Eksplisit

Povray dapat memplot himpunan titik di mana f(x,y,z)=0, sama seperti parameter implicit pada plot3d. Namun, hasilnya terlihat jauh lebih baik.
Sintaks untuk fungsi ini sedikit berbeda. Anda tidak dapat menggunakan output dari Maxima atau ekspresi Euler secara langsung.

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

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  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();
```

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

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

Buat permukaan implisit (implicit surface). Perhatikan sintaks yang berbeda pada ekspresinya.

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Objek Mesh

Pada contoh ini, kita menunjukkan bagaimana cara membuat sebuah mesh object, dan menggambarnya dengan informasi tambahan.
Kita ingin memaksimalkan xy dengan syarat x+y=1 dan menunjukkan sentuhan tangensial pada garis-garis level (level lines).

```
>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 tersebut di dalam file Povray menggunakan declare. Fungsi povtriangle() melakukan hal ini secara otomatis. Fungsi ini dapat menerima normal vectors sama seperti pov3d().
Berikut ini mendefinisikan objek mesh, dan langsung menulisnya 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 (discs), yang akan dipotong (intersected) dengan permukaan.

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

Tulis permukaan dikurangi kedua cakram tersebut.

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

Tulis kedua hasil intersection tersebut.

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

Tulis sebuah titik pada posisi maksimum.

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

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

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Anaglyphs in Povray

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

Tentu saja, Anda memerlukan kacamata merah/sian untuk melihat contoh berikut dengan benar.
ungsi pov3d() memiliki saklar sederhana untuk menghasilkan anaglyph.

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  pov3d:
      if povray then povray(currentfile,w,h,w/h); endif;
```

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

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

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

Fungsi povanaglyph() melakukan semua langkah ini. Parameternya mirip dengan gabungan dari povstart() dan povend().

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povanaglyph:
      povray(currentfile,w,h,aspect,exit);
```

Mendefinisikan Objek Sendiri

Antarmuka Povray di Euler sudah menyediakan banyak objek. Namun, Anda tidak terbatas pada objek-objek tersebut. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau sepenuhnya merupakan objek 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 (origin).

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

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

Berikut adalah torus pertama kita.

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

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

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

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

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

Sekarang kita menempatkan objek-objek ini ke dalam sebuah scene. Untuk look-nya, 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, program ini tidak menampilkan pesan error. Oleh karena itu, Anda sebaiknya menggunakan

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

Jika ada yang tidak berhasil. Cara ini akan membuat jendela Povray tetap terbuka.

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Berikut adalah contoh yang lebih lengkap. Kita menyelesaikan:

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

dan menampilkan titik-titik layak (feasible points) serta titik optimum dalam sebuah 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, solusi ada.
Selanjutnya, kita mendefinisikan dua objek. Objek pertama adalah bidang (plane).

$$a \cdot x \le b$$

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

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

Kemudian kita mendefinisikan irisan (intersection) dari semua half-spaces 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 scene tersebut.

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

Berikut adalah sebuah lingkaran di sekitar titik optimum.

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

Dan sebuah error 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 menempatkan dan memutarnya sesuai dengan sudut pandang kita.

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

```
  exec:
      return _exec(program,param,dir,print,hidden,wait);
  povray:
      exec(program,params,defaulthome);
  Try "trace errors" to inspect local variables after errors.
  povend:
      povray(file,w,h,aspect,exit);
```

Contoh Lainnya

Anda dapat menemukan beberapa contoh tambahan untuk Povray di Euler dalam file-file berikut.

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

Tambahan Catatan

Menggambar Kurva dan Grafik Tiga Dimensi (3D)dengan Euler Math

Toolbox (EMT)

Menggambar grafik tiga dimensi (3D) dalam Euler Math Toolbox (EMT) memiliki berbagai pendekatan sesuai dengan bentuk fungsi atau data yang digunakan.

Pertama, grafik fungsi dua variabel dapat digambarkan dalam bentuk ekspresi langsung, yaitu dengan menuliskan persamaan $z=f(x,y)$ secara eksplisit. Cara ini adalah bentuk paling sederhana karena hanya membutuhkan rumus tanpa definisi tambahan. Jika fungsi yang sama ingin digunakan berulang kali, ekspresi tersebut bisa disimpan dalam variabel sehingga mempermudah pemanggilan kembali tanpa harus menulis ulang.

Selain ekspresi, fungsi juga dapat didefinisikan dalam bentuk numerik maupun simbolik. Fungsi numerik berarti setiap nilai z dihitung langsung berdasarkan perhitungan aritmetika, sedangkan fungsi simbolik memungkinkan manipulasi matematika seperti turunan atau integral sebelum divisualisasikan. Perbedaan ini penting: fungsi numerik lebih sesuai untuk komputasi langsung, sementara fungsi simbolik lebih kuat untuk analisis teoretis.

EMT tidak hanya menggambar fungsi, tetapi juga dapat menampilkan data berupa titik-titik (x,y,z) dalam ruang tiga dimensi. Representasi ini digunakan untuk scatter plot 3D, berguna untuk melihat pola distribusi data atau hasil eksperimen. Grafik juga dapat dibuat interaktif atau dalam bentuk animasi, sehingga sudut pandang, perbesaran, maupun rotasi dapat diamati secara dinamis untuk mendapatkan pemahaman spasial yang lebih baik.

Fungsi parametrik digunakan untuk menggambarkan kurva atau permukaan yang didefinisikan oleh parameter tertentu, misalnya heliks atau lintasan dalam ruang. Berbeda dengan itu, fungsi implisit menggambarkan permukaan yang ditentukan oleh persamaan $F(x,y,z)=0$, contohnya bola atau silinder. Kedua pendekatan ini memperluas jenis objek geometri yang dapat divisualisasikan.

Selain kurva dan permukaan, EMT juga dapat menampilkan titik-titik khusus dalam ruang tiga dimensi untuk menandai posisi atau koordinat penting. Tampilan grafik dapat dikustomisasi dengan mengatur warna, sudut pandang, serta tingkat perbesaran, sehingga hasil visual lebih informatif. Untuk melihat variasi ketinggian atau nilai fungsi dalam bentuk proyeksi dua dimensi, kontur dan bidang kontur dapat digunakan, menyerupai representasi peta topografi.

Mode visualisasi lanjutan meliputi grafik anaglif, yaitu grafik 3D yang dilihat dengan kacamata merah–biru sehingga menampilkan efek kedalaman. Diagram batang 3D memungkinkan representasi data kategorikal dalam tiga dimensi, menyerupai histogram tetapi dengan sudut pandang ruang. EMT juga mendukung pembuatan permukaan benda putar, yaitu permukaan yang terbentuk dari rotasi kurva terhadap sumbu tertentu, misalnya menghasilkan bentuk bola atau vas.

Sebagai tambahan, EMT dapat bekerja sama dengan Povray untuk menghasilkan grafik 3D dengan kualitas tinggi. Povray menambahkan efek pencahayaan, bayangan, dan tekstur sehingga visualisasi menjadi lebih realistis dan layak ditampilkan dalam publikasi atau presentasi.

Keseluruhan materi ini mencakup berbagai teknik representasi fungsi maupun data dalam ruang tiga dimensi, mulai dari bentuk sederhana hingga visualisasi kompleks dengan efek grafis yang mendalam.

1. Ekspresi Langsung

Menggambar Grafik Fungsi Dua Variabel
- Kita langsung menuliskan persamaan $z = f(x,y)$ dalam bentuk string.
- Fungsi ini cocok untuk kasus sederhana.
- Hasilnya berupa permukaan paraboloid terbuka ke atas.

```
>plot3d("x^2+y^2",-2,2,-2,2)
```

2. Grafik dari Variabel Ekspresi

- Persamaan disimpan ke dalam sebuah variabel ekspresi.
- Keuntungannya:ekspresi bisa dipanggil berkali-kali tanpa menulis ulang.
- Cocok untukeksplorasi fungsi yang sama dengan domain berbeda.

```
>f = "sin(x)*cos(y)"
>plot3d(f, -pi, pi, -pi, pi)
```

3. Fungsi Dua Variabel sebagai Fungsi Numerik

- Fungsi didefinisikan secara numerik dengan tanda :=.
- EMT menghitung nilai numerik langsung saat menggambar.
- Digunakan untuk fungsi kompleks dengan perhitungan langsung, misalnya melibatkan eksponensial, trigonometri, dll.

```
>function f(x,y) := x*exp(-x^2-y^2)
>plot3d("f(x,y)", -2, 2, -2, 2)
```

4. Fungsi Dua Variabel sebagai Fungsi Simbolik

- Menggunakan &= agar EMT menganggap fungsi dalam bentuk simbolik.
- Kelebihan: EMT bisa melakukan turunan, integral, atau manipulasi simbolik sebelum digambar.
- Cocok untuk analisis teoretis.

```
>function f(x,y) &= x^3 - 3*x*y^2
>plot3d("f(x,y)", -2,2,-2,2)
```

5. Menggambar Data (x, y, z) dalam Ruang 3D

- Data acak digambar sebagai titik.
- Cocok untuk visualisasi data eksperimen atau hasil simulasi.
- Hasil berupa scatter plot 3D.

```
>x=random(100); y=random(100); z=random(100)
>plot3d(x,y,z, style="points")
```

6. Grafik Interaktif dan Animasi

- Menggunakan perintah animate untuk memutar sudut pandang.
- Animasi membantu memahami bentuk permukaan dari berbagai sisi.
- Bisa disimpan sebagai file animasi (GIF/AVI).

```
>plot3d("sin(x)*cos(y)", -pi, pi, -pi, pi, angle=animate(0,360,5))
```

7. Fungsi Parametrik 3D

- Fungsi didefinisikan berdasarkan parameter t.
- Cocok untuk kurva ruang seperti spiral, lintasan partikel, atau geometri 3D.
- Hasil: kurva heliks berputar di sekitar sumbu z.

```
>t=linspace(0,4*pi,400)
>x=cos(t); y=sin(t); z=t
>plot3d(x,y,z,style="line")
```

8. Fungsi Implisit 3D

- Fungsi implisit: $F(x,y,z)=0$ (bukan $z=f(x,y)$).
- Cocok untuk menggambar bola, silinder, atau bidang level surface.
- Hasil: permukaan bola dengan jari-jari 1.

```
>implicit3d("x^2+y^2+z^2-1",-1.5,1.5,-1.5,1.5,-1.5,1.5)
```

9. Menggambar Titik di Ruang 3D

- Menggambarkan titik-titik tertentu pada ruang 3D.
- Cocok untuk menampilkan hasil pengukuran, koordinat, atau titik penting pada grafik.

```
>plot3d([1,2,3],[2,3,1],[1,0,2],style="points")
```

10. Mengatur Tampilan, Warna, dan Sudut Pandang

- Parameter color, angle, dan zoom digunakan untuk mengatur visualisasi.
- Warna bisa berupa gradasi, kontur, atau solid.
- Tampilan dapat disesuaikan sesuai kebutuhan presentasi.

```
>plot3d("x^2+y^2",-2,2,-2,2,color=rainbow,angle=45,zoom=1.2)
```

11. Menampilkan Kontur dan Bidang Kontur

- Kontur adalah proyeksi ketinggian z ke bidang xy.
- Mirip peta topografi.
- Membantu melihat pola ketinggian tanpa permukaan 3D penuh.

```
>contour("sin(x)*cos(y)", -pi,pi,-pi,pi)
```

12. Grafik 3D dalam Modus Anaglif

- Menampilkan grafik untuk dilihat dengan kacamata 3D merah-biru.
- Memberikan efek kedalaman pada grafik.
- Cocok untuk presentasi interaktif.

```
>plot3d("sin(x)*cos(y)", -pi,pi,-pi,pi, anaglyph=1)
```

13. Diagram Batang 3D

- Menampilkan data matriks dalam bentuk batang 3D.
- Cocok untuk statistik atau perbandingan antar kategori dalam 3 dimensi.
- Mirip histogram tetapi dalam ruang 3D.

```
>M=random(5,5)
>bar3d(M)
```

14. Permukaan Benda Putar

- Menghasilkan permukaan dari rotasi kurva terhadap sumbu tertentu.
- Cocok untuk memodelkan benda simetris (vas, botol, bola, dll).
- Bentuk akhir adalah permukaan rotasi.

```
>plot3d("cos(x)", -pi,pi,-1,1, rotate=true)
```

15. Grafik 3D dengan Povray

- Povray adalah software rendering yang bisa dipanggil dari EMT.
- Memberikan hasil visual lebih realistis (efek cahaya, bayangan, tekstur).
- Cocok untuk membuat gambar publikasi atau presentasi profesional.

```
>plot3d("sin(x)*cos(y)", -pi,pi,-pi,pi, povray=true)
```

Kesimpulan

Setiap metode memiliki fungsi dan tujuan yang berbeda:
- Ekspresi langsung : cepat & sederhana.
- Ekspresi/variabel : fleksibel & bisa digunakan berulang.
- Numerik/simbolik : mendukung perhitungan lebih lanjut.
- Data, titik, batang : visualisasi hasil eksperimen.
- Parametrik & implisit : geometri yang lebih kompleks.
- Interaktif, anaglif, Povray : menambah daya tarik visual & pemahaman spasial.