Menggambar Plot 3D dengan EMT

NAMA : ACHMADdZUL fAUZAN bAHRI
NIM : 24030130061
Ini adalah pengenalan tentang plot 3D di Euler. Kita memerlukan plot 3D untuk memvisualisasikan fungsi dari dua variabel.

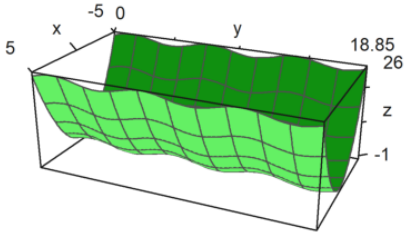
Euler menggambar fungsi semacam itu menggunakan algoritma penyortiran untuk menyembunyikan bagian di latar belakang. Secara umum, Euler menggunakan proyeksi sentral. Secara bawaan, sudut pandang berasal dari kuadran x-y positif menuju titik asal x=y=z=0, tetapi angle=0° melihat ke arah sumbu-y. Sudut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

Euler dapat memplot

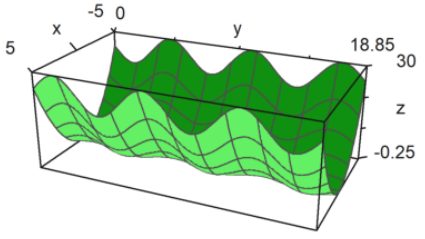
- permukaan dengan bayangan dan garis level atau rentang level,
- kumpulan titik,
- kurva parametrik,
- permukaan implisit.

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

Untuk grafik sebuah fungsi, gunakan

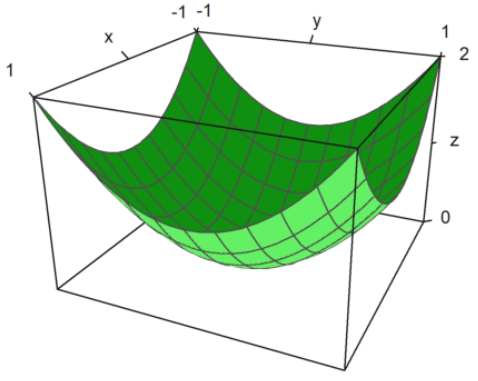
- ekspresi sederhana dalam x dan y,
- nama fungsi dari dua variabel,
- atau matriks data.

Bawaan (default) adalah grid kawat (wire grid) terisi dengan warna berbeda pada kedua sisinya. Perlu dicatat 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 tiap arah
- grid=10, grid=\[10,10]: jumlah garis grid pada tiap arah.

Kita gunakan nilai bawaan **n=40** dan **grid=10**.

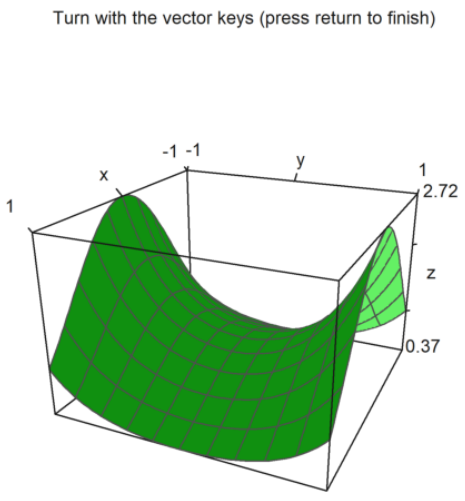
```
>plot3d("x^2+y^2"):
```



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

- **left, right, up, down**: mengubah sudut pandang
- **+ , -**: memperbesar atau memperkecil (zoom in/out)
- **a**: menghasilkan anaglif (lihat penjelasan di bawah)
- **l**: mengaktifkan/nonaktifkan pergerakan sumber cahaya (lihat penjelasan di bawah)
- **space**: mengembalikan ke tampilan bawaan
- **return**: mengakhiri interaksi

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



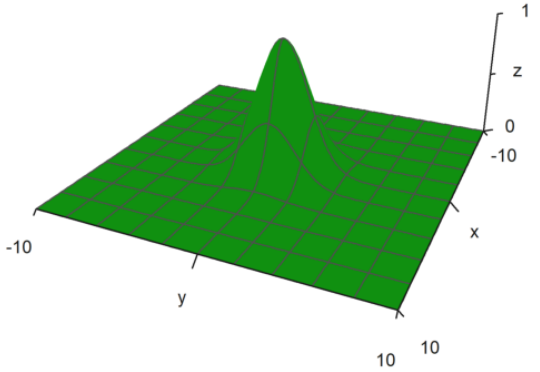
Rentang plot untuk fungsi dapat ditentukan dengan:

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

Ada beberapa parameter untuk melakukan penskalaan fungsi atau mengubah tampilan grafik:

- **fscale**: skala terhadap nilai fungsi (bawaan adalah \<fscale>)
- **scale**: angka atau vektor 1x2 untuk skala ke arah x dan y
- **frame**: tipe bingkai (bawaan = 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 ke plot
- **zoom**: nilai perbesaran (zoom)
- **angle**: sudut terhadap sumbu y negatif dalam radian
- **height**: tinggi pandangan dalam radian

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

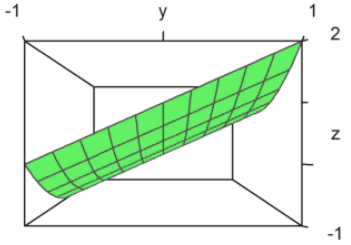
```
>view
```

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

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

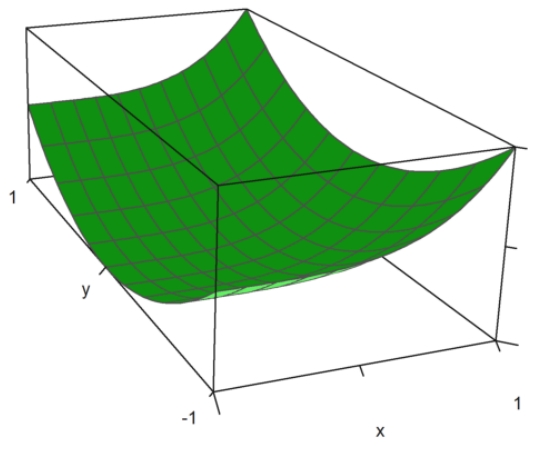
Dalam contoh berikut, **angle=0** dan **height=0** melihat dari arah sumbu y negatif. Pada kasus ini, label sumbu y akan tersembunyi.

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



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

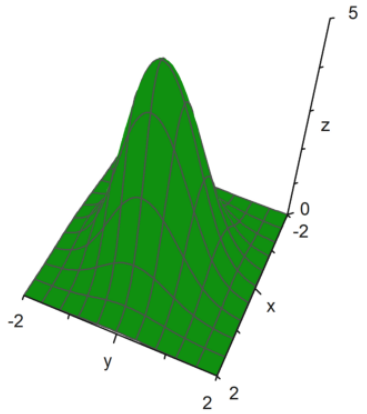
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>  center=[0.4,0,0],zoom=5):
```



Plot diskalakan agar pas ke dalam sebuah kubus satuan untuk tampilan.
Karena itu, tidak perlu mengubah **distance** atau **zoom** tergantung pada ukuran plot.
Namun, label tetap merujuk pada ukuran sebenarnya.

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

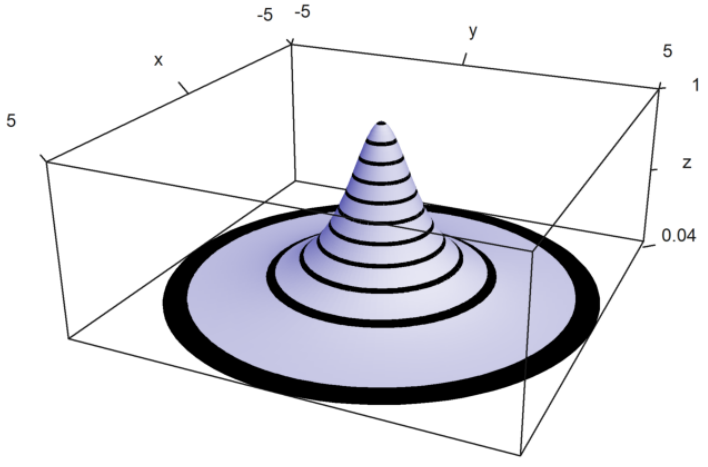
```
>plot3d("5*exp(-x^2-y^2)",r=2,<fscale,<scale,distance=13,height=50°, ...
>  center=[0,0,-2],frame=3):
```



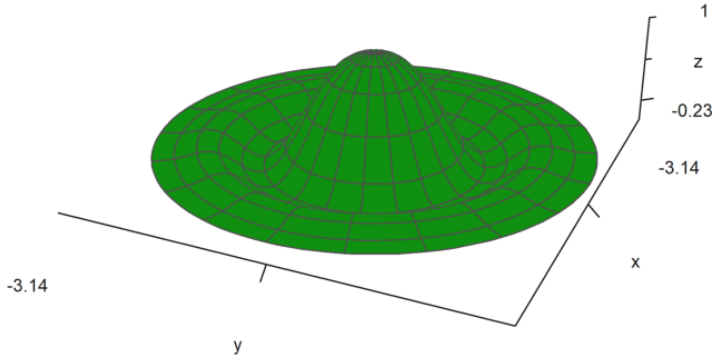
Plot polar juga tersedia.
Parameter **polar=true** akan menggambar plot polar. Fungsi yang digunakan tetap harus berupa fungsi dari **x** dan **y**.

Parameter **fscale** akan menskalakan fungsi dengan skala tersendiri.
Jika tidak, fungsi akan diskalakan agar pas ke dalam sebuah kubus.

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



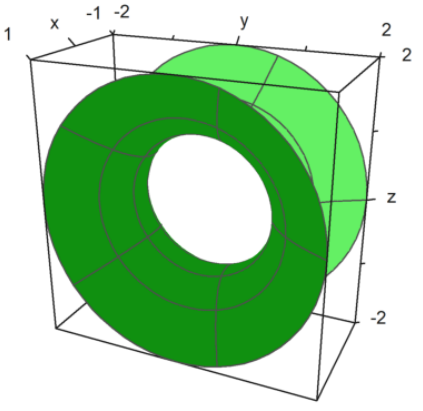

```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



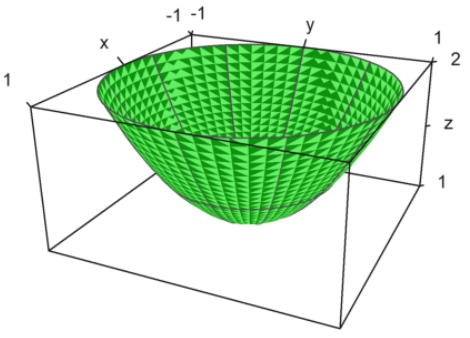
Parameter **rotate** memutar sebuah fungsi dalam **x** di sekitar sumbu tertentu.

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

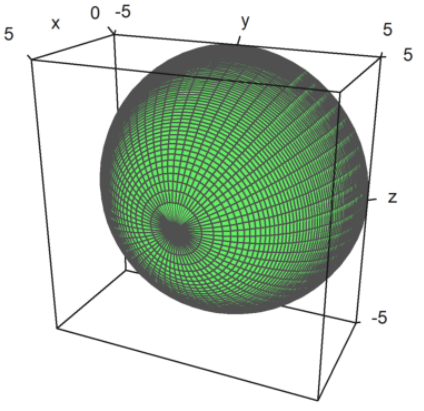
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



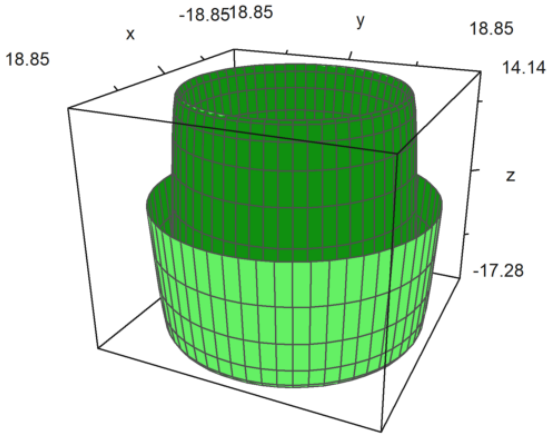

```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```




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

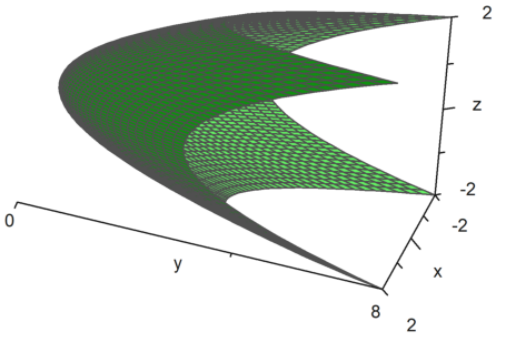



```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Berikut adalah sebuah plot dengan tiga fungsi.

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


2
z
-2
-2
x
2
8
y
0

Plot Kontur (Contour Plots)

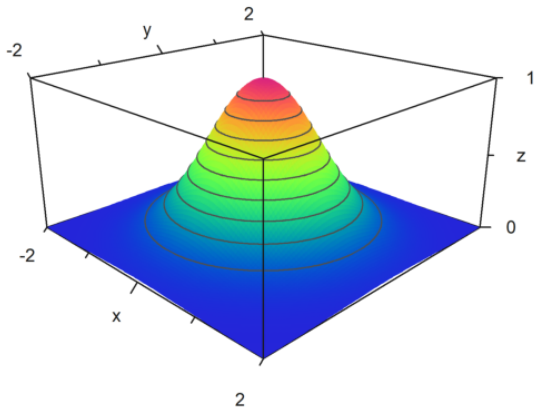
Untuk plot, Euler menambahkan garis grid. Sebagai gantinya, dimungkinkan untuk menggunakan garis level serta pewarnaan satu-warna atau pewarnaan spektral.
Euler juga dapat menampilkan ketinggian fungsi pada plot dengan shading.
Dalam semua plot 3D, Euler dapat menghasilkan **anaglif merah/sian**.

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

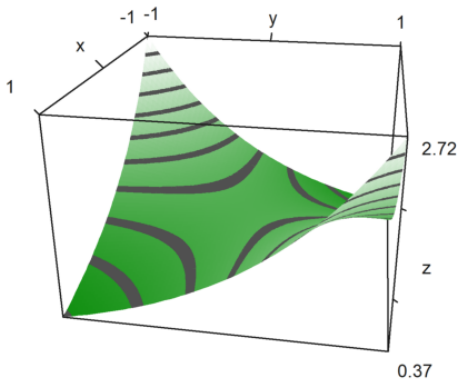
Bawaan (default) adalah **level="auto"**, yang secara otomatis menghitung beberapa garis level.
Seperti terlihat pada plot, level tersebut sebenarnya berupa **rentang level**.

Gaya bawaan dapat diubah.
Untuk plot kontur berikut, kita menggunakan grid yang lebih halus (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):
```

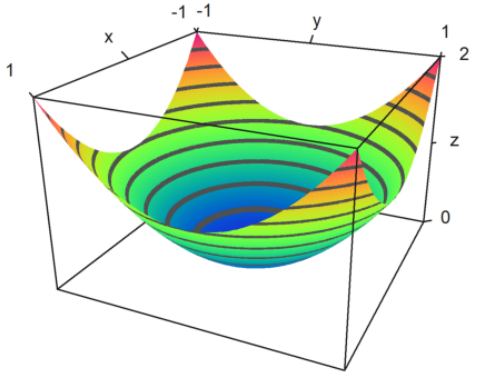


Shading bawaan menggunakan warna abu-abu.
Namun, rentang warna spektral juga tersedia.

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

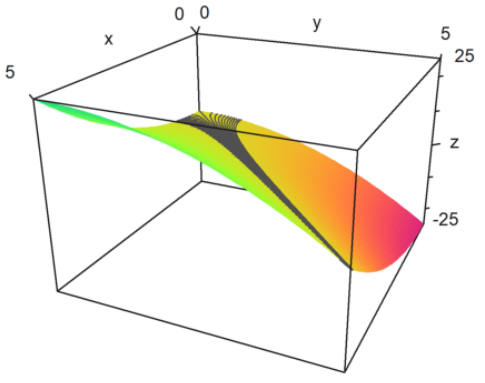
Untuk plot berikut, kita menggunakan skema spektral bawaan dan menambah jumlah titik agar tampilan menjadi sangat halus.

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



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

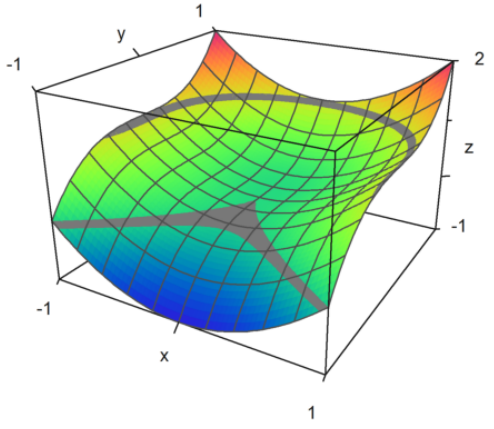
```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



Pada plot berikut, kita menggunakan 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 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):
```

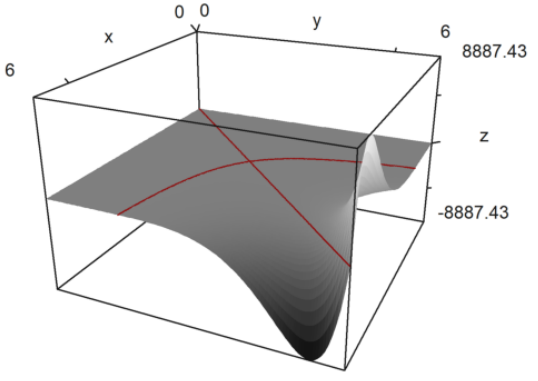


Pada contoh berikut, kita memplot himpunan di mana

f(x,y) = xˆy - yˆx = 0

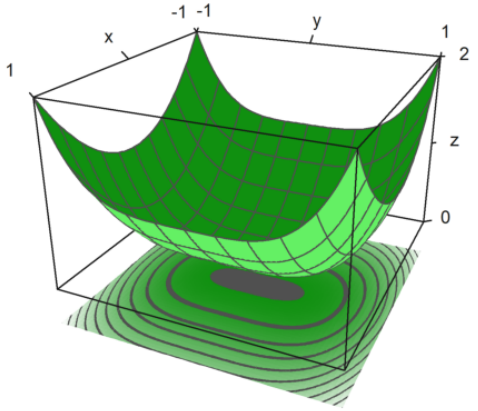
Kita menggunakan satu garis tipis sebagai garis level.

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



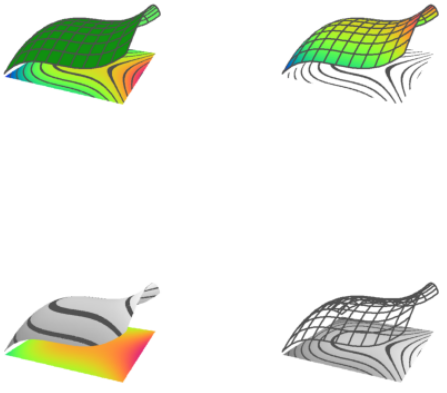
Dimungkinkan untuk menampilkan sebuah **bidang kontur** di bawah plot.
Warna serta jarak bidang tersebut terhadap plot dapat ditentukan.

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



Berikut beberapa gaya tambahan.
Kita selalu mematikan **frame**, dan menggunakan berbagai skema warna untuk plot serta grid.

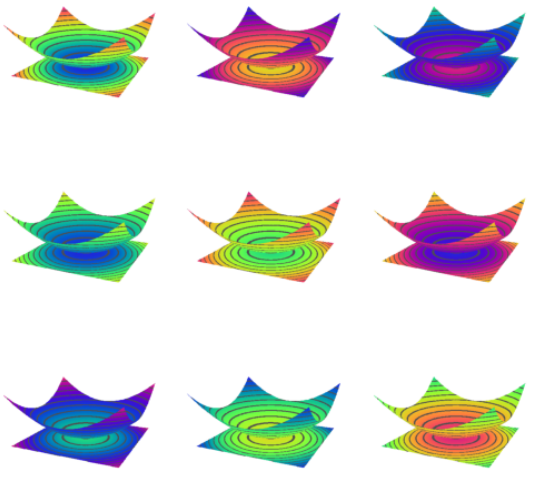
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1);  ...
>  plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2);  ...
>  plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3);  ...
>  plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4);  ...
>  plot3d(expr,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



Ada beberapa skema spektral lain yang diberi nomor dari **1 hingga 9**.
Namun, Anda juga bisa menggunakan **color=value**, dengan nilai berikut:

- **spectral**: rentang dari biru ke merah
- **white**: rentang 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):
```

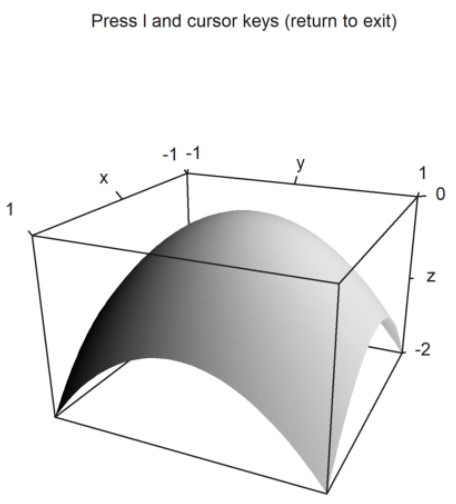


Sumber cahaya dapat diubah dengan menekan **l** dan tombol panah saat interaksi pengguna. Selain itu, cahaya juga dapat diatur dengan parameter.

- **light**: arah datangnya cahaya
- **amb**: cahaya sekitar (ambient) antara 0 dan 1

Perlu dicatat bahwa program tidak membedakan sisi-sisi plot. Tidak ada bayangan yang dihasilkan. 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)"):
```

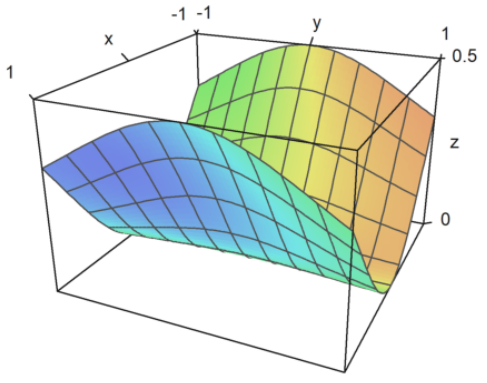


Parameter **color** mengubah warna permukaan.
Warna garis level juga dapat diubah.

```
>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):
```

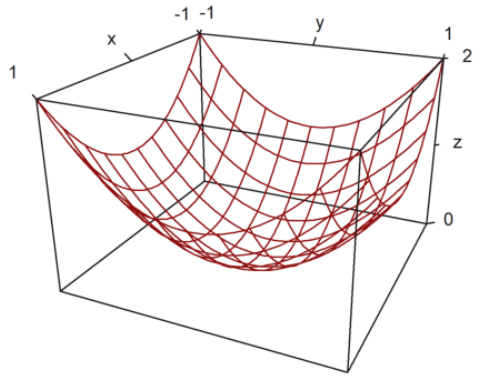
Warna **0** memberikan efek pelangi khusus.

```
>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):
```


-1
-1
x
y
1
1
2
z
0

Plot Implisit (Implicit Plots)

Euler juga mendukung **plot implisit** dalam tiga dimensi.
Euler menghasilkan potongan (cuts) melalui objek tersebut.
Fitur **plot3d** mencakup plot implisit. Plot ini menampilkan himpunan nol dari sebuah fungsi dalam tiga variabel.

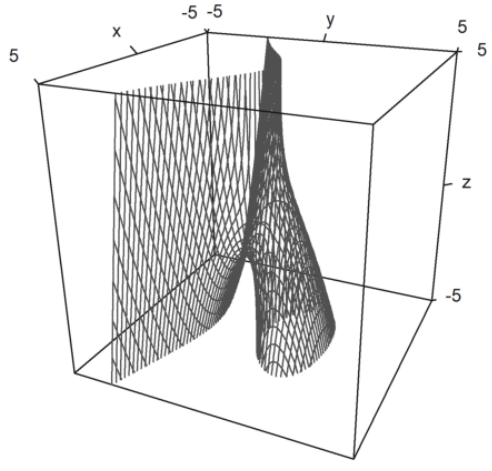
f(x,y,z) = 0

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

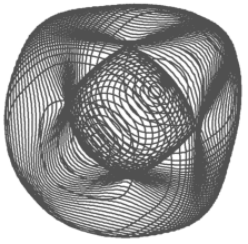
- **implicit=1**: potongan sejajar dengan bidang y-z
- **implicit=2**: potongan sejajar dengan bidang x-z
- **implicit=4**: potongan sejajar dengan bidang x-y

Nilai-nilai ini bisa dijumlahkan sesuai kebutuhan.
Sebagai contoh, kita memplot:
M = \{ (x,y,z) : xˆ2 + yˆ3 + zy = 1 \}

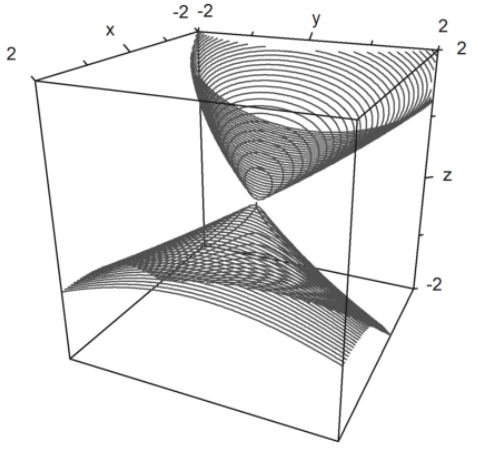
```
>plot3d("xˆ2+yˆ3+z*y-1",r=5,implicit=3):
```




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



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


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

Plot Data 3D (Plotting 3D Data)

Sama seperti **plot2d**, fungsi **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)**.

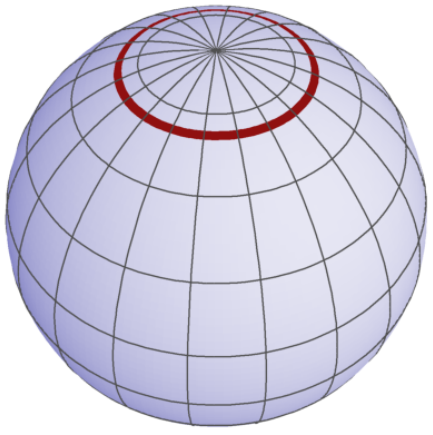
\gamma(t,s) = (x(t,s), y(t,s), z(t,s))

Karena **x, y, z** berbentuk matriks, kita mengasumsikan bahwa (t,s) berjalan melalui grid berbentuk persegi.
Hasilnya, Anda dapat memplot bayangan persegi panjang di ruang 3D.

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

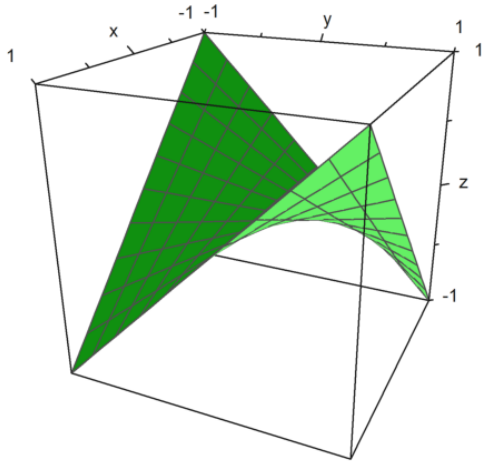
Dalam contoh berikut, kita menggunakan sebuah vektor nilai **t** dan sebuah vektor kolom nilai **s** untuk memparametrisasi permukaan bola.
Pada gambar, kita dapat menandai area tertentu, dalam hal ini **daerah kutub (polar region)**.

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



Berikut adalah sebuah contoh, yaitu grafik dari suatu fungsi.

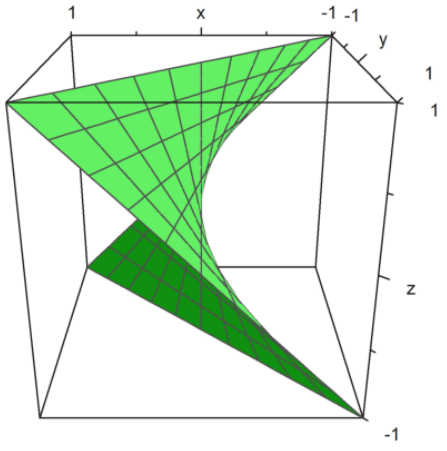
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



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

x = y \, z

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



Dengan usaha lebih, kita dapat menghasilkan banyak permukaan.

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

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

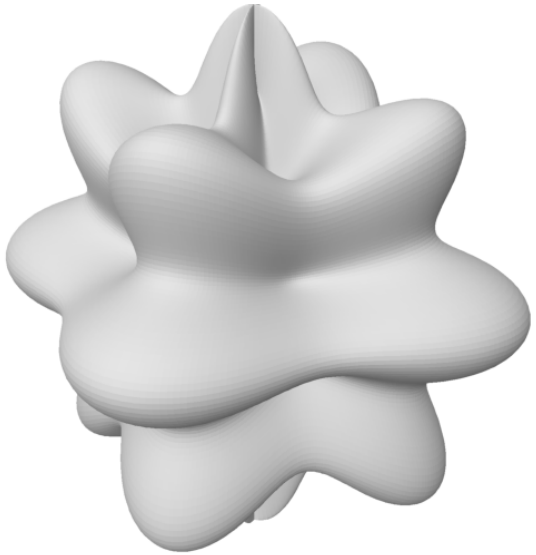
dengan

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

Kita mendistorsi bola tersebut dengan faktor

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

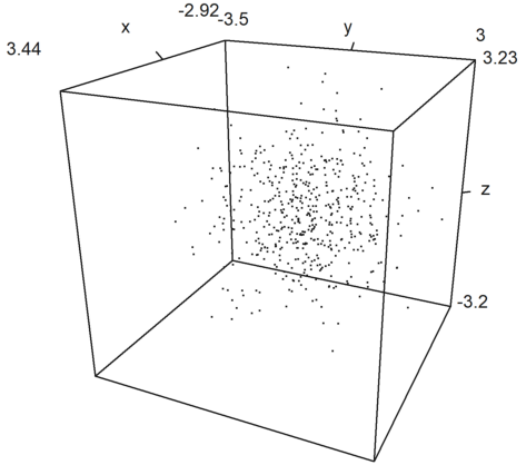
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
>  light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, point cloud juga memungkinkan. Untuk memplot data titik di ruang, kita memerlukan tiga vektor yang berisi koordinat titik-titik tersebut.

Gaya (style) sama seperti pada ‘plot2d‘ dengan opsi ‘points=true‘.

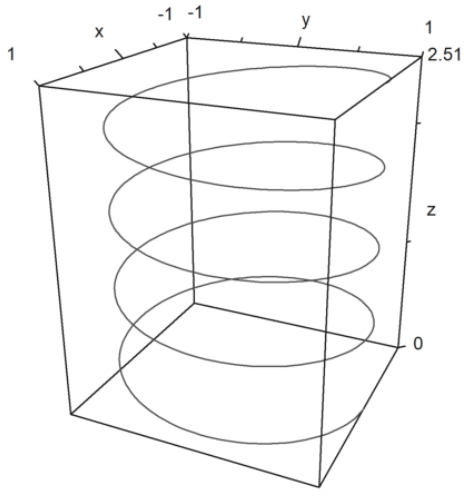
```
>n=500;  ...
>  plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style="."):
```



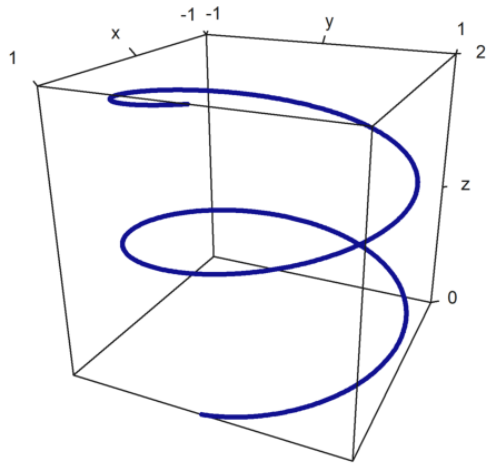
Kita juga dapat memplot sebuah kurva dalam 3D.
Dalam hal ini, lebih mudah jika titik-titik kurva sudah dihitung terlebih dahulu.

Untuk kurva di bidang, kita menggunakan sebuah urutan koordinat dan parameter **wire=true**.

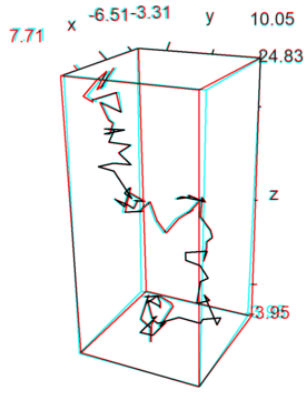
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

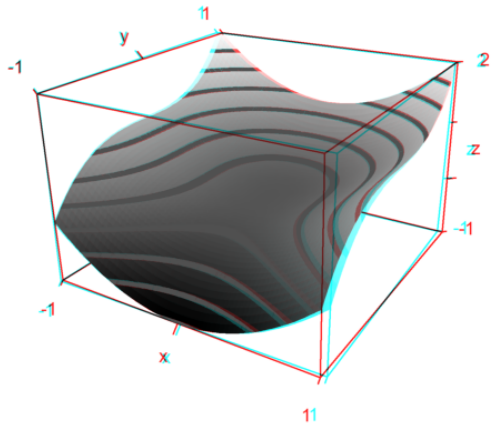



```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



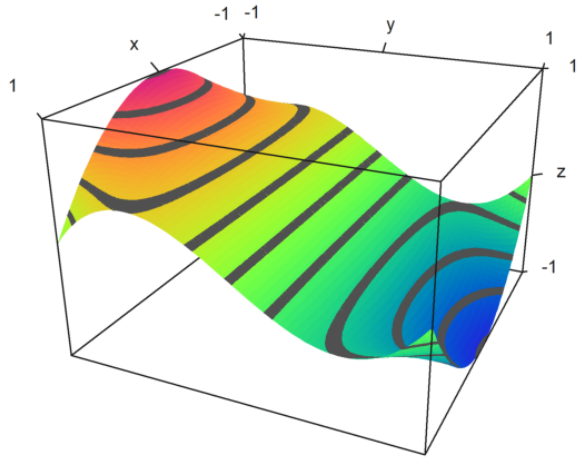
EMT juga dapat membuat plot dalam mode **anaglif**.
Untuk melihat plot tersebut, Anda memerlukan **kacamata merah/sian (red/cyan glasses)**.

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



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

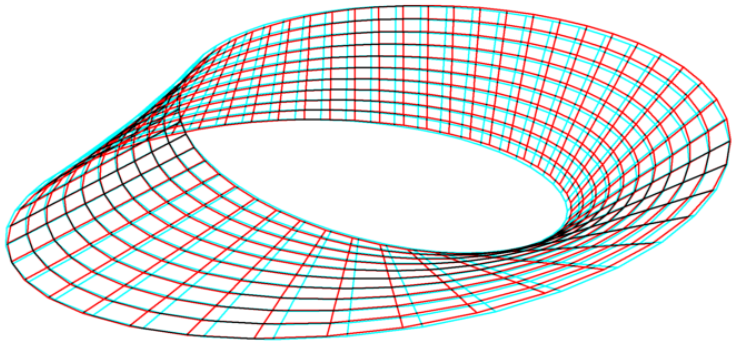
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



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

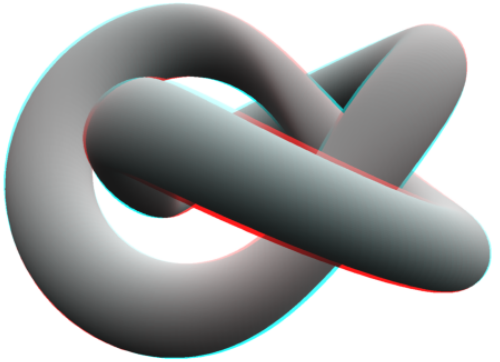
Untuk demo berikut, kita menyiapkan parameter **u** dan **v**, lalu menghasilkan koordinat ruang dari parameter tersebut.

```
>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 bila dilihat dengan **kacamata merah/sian**.

```
>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):
```



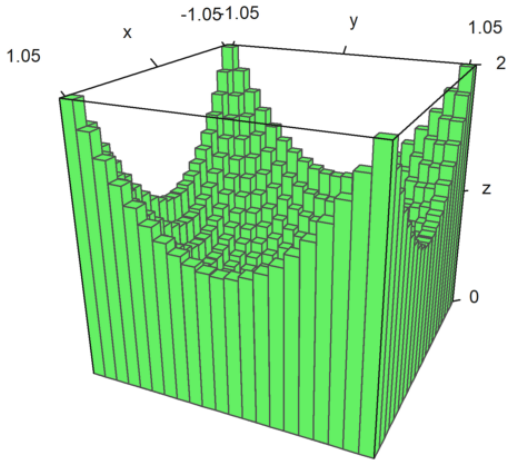
Plot batang (**bar plots**) juga dapat dibuat. Untuk itu, kita harus menyediakan:

- **x**: vektor baris dengan $n+1$ elemen
- **y**: vektor kolom dengan $n+1$ elemen
- **z**: matriks $n \times n$ berisi nilai

Matriks **z** bisa lebih besar, tetapi hanya nilai $n \times n$ yang akan digunakan.

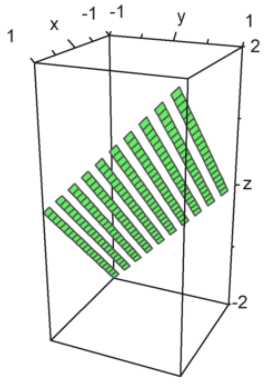
Dalam contoh, pertama-tama kita menghitung nilai-nilai fungsi.
Kemudian kita menyesuaikan **x** dan **y**, sehingga vektor-vektor tersebut terpusat pada nilai yang digunakan.

```
>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 suatu 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):
```

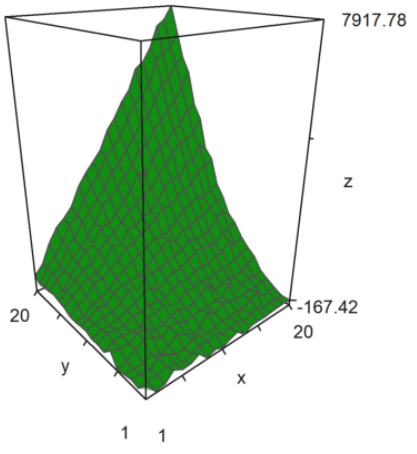


Jika Anda memuat atau menghasilkan sebuah matriks data **M** dari file dan ingin memplotnya dalam 3D, Anda bisa:

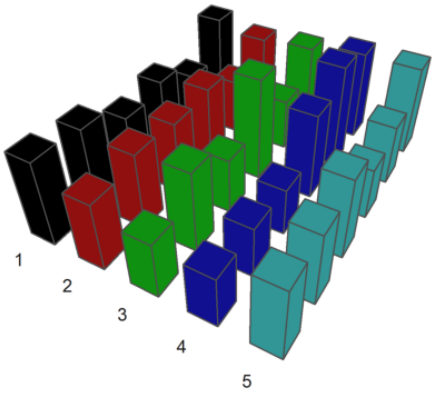
- menskalakan matriks ke rentang $[-1,1]$ dengan **scale(M)**, atau
- menskalakan matriks dengan opsi **>zscale**.

Keduanya dapat dikombinasikan dengan faktor skala individual yang diterapkan secara tambahan.

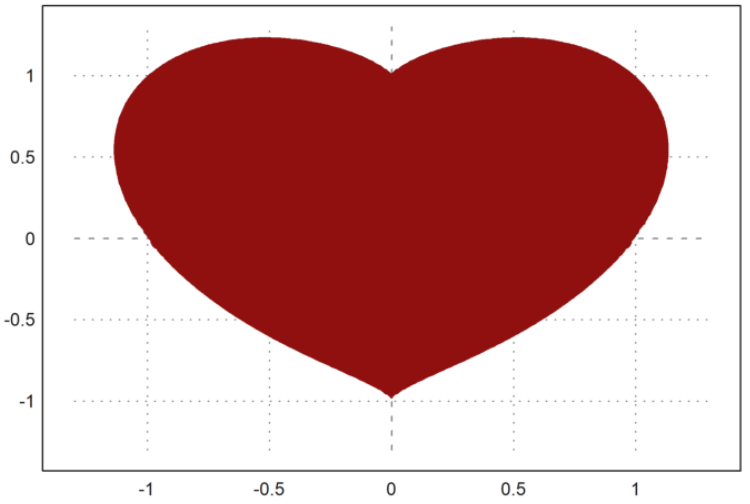
```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8):
```




```
>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]):
```


1
2
3
4
5

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




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

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

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

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

Selanjutnya kita tetapkan

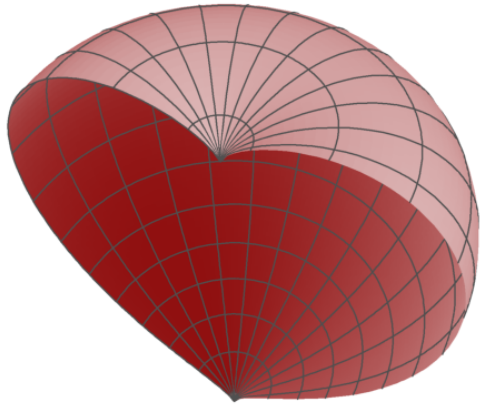
x = r \cos(a), \quad y = r \sin(a).

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

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

Hal ini memungkinkan kita mendefinisikan sebuah fungsi numerik, yang menyelesaikan 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°):
```

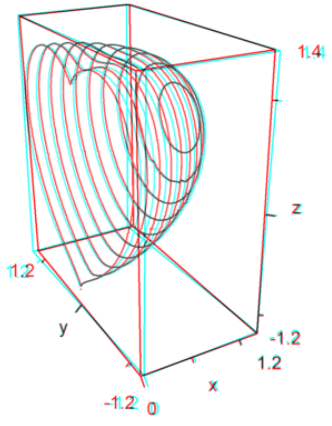


Berikut adalah plot 3D dari gambar di atas yang diputar mengelilingi sumbu-z.
Kita mendefinisikan fungsi yang menggambarkan objek tersebut.

```
>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 (Special 3D Plots)

Fungsi **plot3d** memang sangat berguna, tetapi tidak selalu memenuhi semua kebutuhan.
Selain rutin dasar, dimungkinkan untuk membuat plot berbingkai (framed plot) dari objek apa pun yang diinginkan.

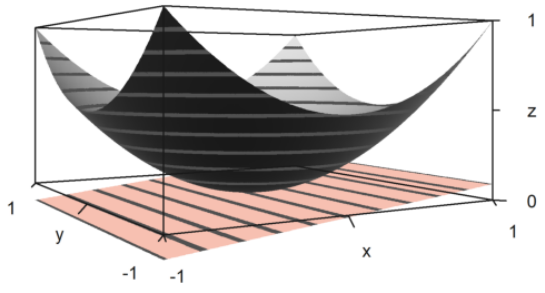
Meskipun Euler bukanlah program 3D penuh, ia dapat mengombinasikan beberapa objek dasar.
Sebagai contoh, kita mencoba memvisualisasikan sebuah **paraboloid** dan bidang **tangent**-nya.

```
>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 (frames) dan mengatur tampilan (views).

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

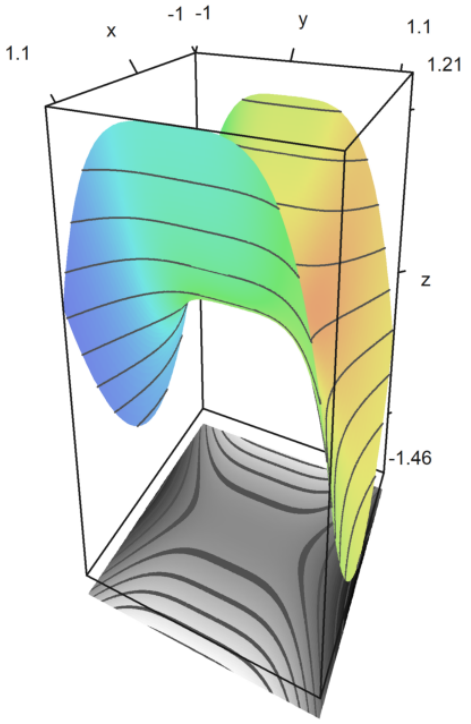


Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual.
Perlu dicatat bahwa **plot3d()** secara bawaan mengatur jendela ke **fullwindow()**, sedangkan **plotcontourplane()** berasumsi demikian.

```
>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
```

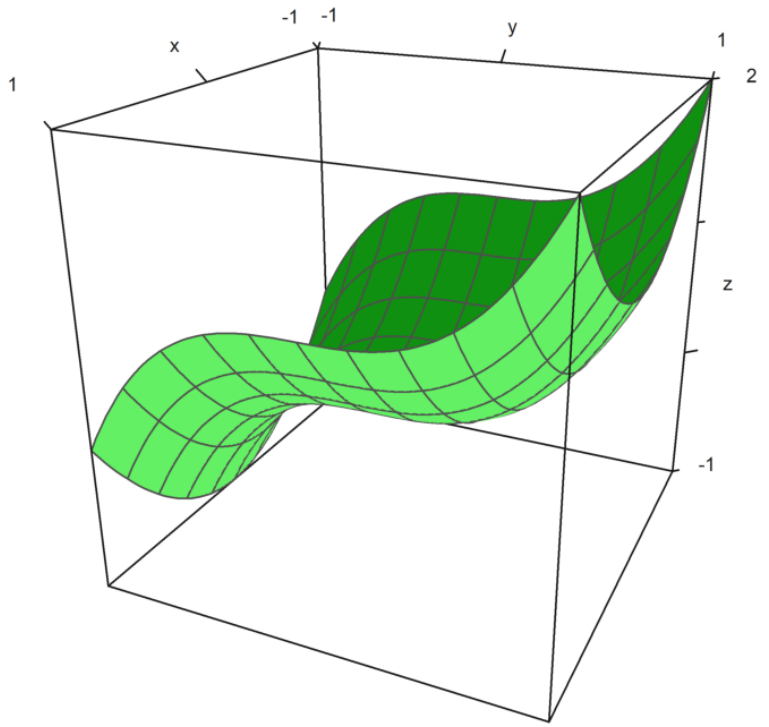
```
>myplot(x,y,z):
```



Animasi (Animation)

Euler dapat menggunakan **frame** untuk melakukan pra-perhitungan animasi.
Salah satu fungsi yang memanfaatkan teknik ini adalah **rotate**.
Fungsi ini dapat mengubah sudut pandang dan menggambar ulang plot 3D.
Fungsi tersebut memanggil **addpage()** untuk setiap plot baru.
Akhirnya, fungsi ini menganimasikan plot-plot tersebut.
Silakan pelajari kode sumber dari **rotate** untuk melihat lebih detail.

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


-1
-1
x
y
1
1
2
z
-1

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

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

Fungsi **pov3d** bekerja serupa dengan **plot3d**. Fungsi ini dapat menghasilkan grafik fungsi $f(x,y)$, atau permukaan dengan koordinat **X, Y, Z** dalam bentuk matriks, termasuk garis level opsional. Fungsi ini akan secara otomatis menjalankan *

```
>load povray;
```

Pastikan direktori ‘bin‘ Povray sudah ada di **path**. Jika belum, ubah variabel berikut sehingga berisi jalur menuju **eksekusi Povray**.

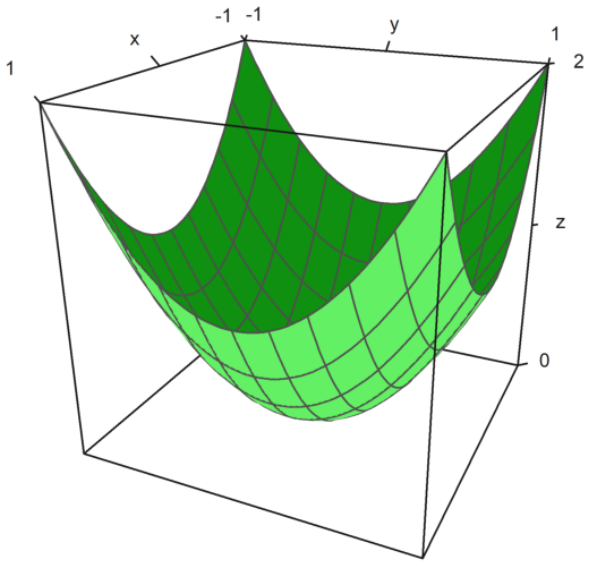
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

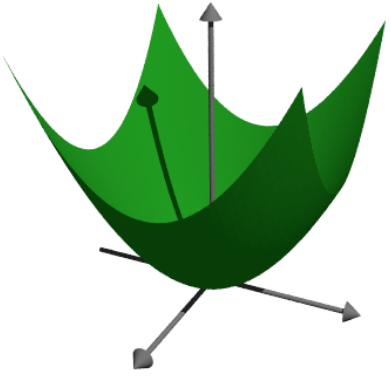
Untuk kesan pertama, kita akan memplot fungsi sederhana. Perintah berikut akan membuat file Povray di direktori pengguna Anda, dan menjalankan Povray untuk melakukan **ray tracing** pada file tersebut.

Jika Anda menjalankan perintah berikut, **GUI Povray** akan terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan diminta konfirmasi apakah ingin mengizinkan file '.exe' dijalankan. Anda bisa 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):
```

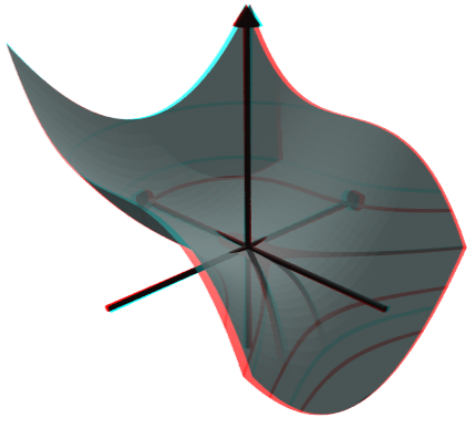



```
>pov3d("x^2+y^2",zoom=3);
```



Kita dapat membuat fungsi menjadi **transparan** dan menambahkan **finish** lain. Kita juga bisa menambahkan **garis kontur (level lines)** pada plot fungsi tersebut.

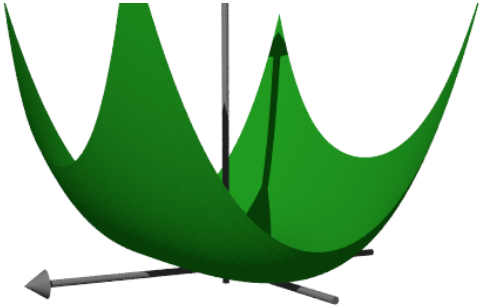
```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
>  look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



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

Kita memplot **himpunan titik di bidang kompleks**, di mana **hasil perkalian jarak ke 1 dan -1 sama dengan 1**.

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

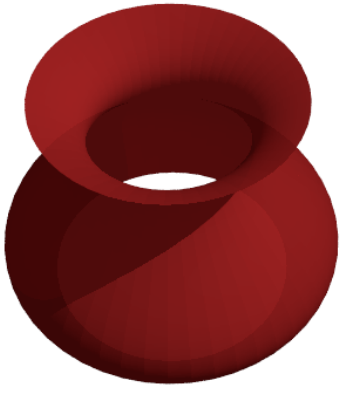


Plotting dengan Koordinat

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

Dalam contoh ini, kita **memutar sebuah fungsi mengelilingi sumbu z**.

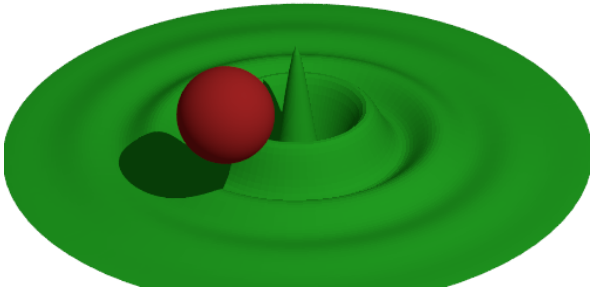
```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,axis=0,zoom=4,light=[10,5,15]);
```



Dalam contoh berikut, kita memplot **gelombang yang teredam**. Gelombang ini dibuat menggunakan **bahasa matriks Euler**.

Kita juga menunjukkan bagaimana **objek tambahan** dapat ditambahkan ke **scene pov3d**. Untuk pembuatan objek, lihat contoh-contoh berikutnya. Perlu diperhatikan bahwa ‘plot3d‘ **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);
```



Dengan metode **advanced shading** Povray, **hanya sedikit titik** sudah dapat menghasilkan **permukaan yang sangat halus**. Hanya di **tepi** dan **bayangan** trik ini mungkin terlihat jelas.

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

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

```
  2  3
x  y
```

Persamaan permukaannya adalah **\[x, y, Z]**. Kita menghitung **dua turunan terhadap x dan y**, lalu mengambil **cross product** dari hasilnya sebagai **vektor normal**.

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

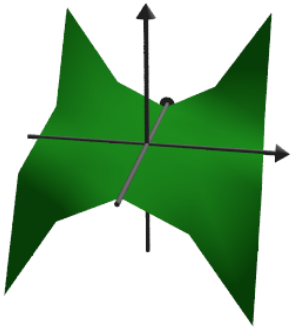
Kita mendefinisikan **vektor normal** sebagai **cross product** dari turunan-turunan tersebut, dan mendefinisikan **fungsi koordinat**.

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

```
          3       2  2
[- 2 x y , - 3 x  y , 1]
```

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. Terdapat versi yang lebih baik dalam contoh-contoh.

Lihat: **Examples\Trefoil Knot | Trefoil Knot**

Agar tampilan lebih baik dengan **tidak terlalu banyak titik**, kita menambahkan **vektor normal** di sini. Kita menggunakan **Maxima** untuk menghitung normal. Pertama, tiga fungsi untuk **koordinat** dibuat 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);
```

Selanjutnya, kita menghitung **dua vektor turunan terhadap x dan y**.

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

Sekarang, kita menentukan **vektor normal**, yang merupakan **cross product dari dua turunan** tersebut.

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

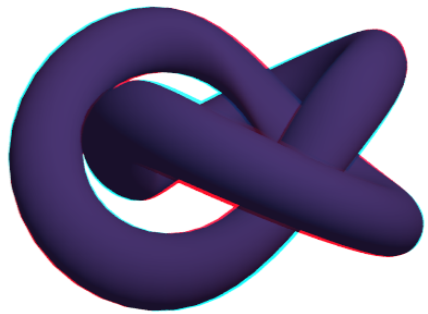
Sekarang, kita **menghitung semuanya secara numerik**.

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

Vektor normal adalah hasil **evaluasi ekspresi simbolik** ‘dn[i]‘ untuk ‘i = 1, 2, 3‘. Sintaksnya adalah **&”expression”(parameters)**.

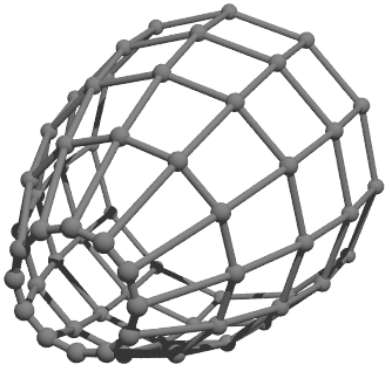
Ini merupakan **alternatif** dari metode pada contoh sebelumnya, di mana kita terlebih dahulu mendefinisikan ekspresi simbolik **NX, NY, NZ**.

```
>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));
```



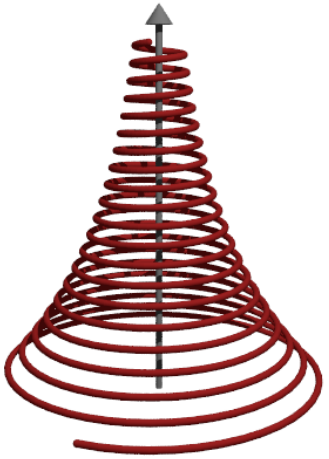
Kita juga bisa **menghasilkan grid dalam 3D**.

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



Dengan **povgrid()**, **kurva** juga dimungkinkan.

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



Objek Povray

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

Kita memulai output dengan **povstart()**.

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

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

Fungsi **povx()** dan sejenisnya hanya mengembalikan **vektor \[1,0,0]**, yang bisa digunakan sebagai alternatif.

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

String-string tersebut berisi **kode Povray**, yang **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 bisa menggunakan **warna default Euler**, atau mendefinisikan warna sendiri. Kita juga bisa menambahkan **transparansi**, atau mengubah **cahaya ambient**.

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

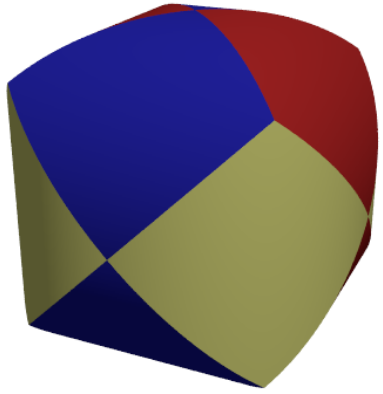
```
  texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> }  }
  finish { ambient 0.5 }
```

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

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

Irisan dari **tiga silinder** sulit untuk divisualisasikan, terutama jika **belum pernah melihatnya sebelumnya**.

```
>povend;
```



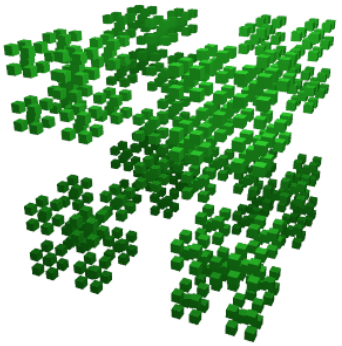
Fungsi-fungsi berikut **menghasilkan fractal secara rekursif**.

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

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

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

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



Perbedaan (**differences**) memungkinkan kita **memotong satu objek dari objek lain**. Seperti **intersections**, ini merupakan bagian dari **CSG objects** di Povray.

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

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

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

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

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

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

Kita membuat **kubus kedua**, lalu **memutar** dan **menskalakan** sedikit.

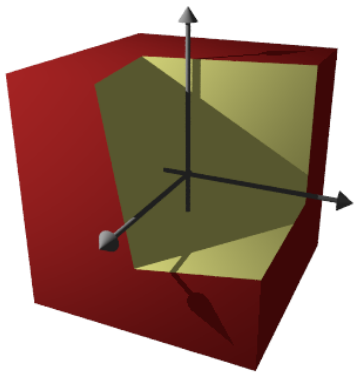
```
>c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...
>  rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Kemudian, kita mengambil **perbedaan (difference)** dari kedua objek tersebut.

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

Sekarang, tambahkan **tiga sumbu (axes)**.

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



Fungsi Implisit

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

Sintaks untuk fungsi sedikit berbeda. **Output dari Maxima atau ekspresi Euler tidak bisa langsung digunakan**.

Contoh dalam LaTeX:

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

```
>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)+pow(z,2)-po
>povend();
```

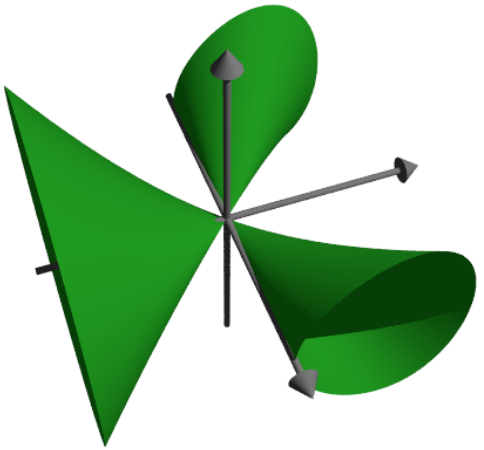
```
Error : Povray error!

Error generated by error() command

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

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

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

Buat **permukaan implisit**. Perhatikan **perbedaan sintaks** pada ekspresinya.

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



Objek Mesh

Dalam contoh ini, kita menunjukkan cara **membuat objek mesh** dan **menggambar**nya dengan informasi tambahan.

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

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

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

Berikut ini mendefinisikan **objek mesh** dan langsung menulisnya ke **file**.

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

Sekarang, kita mendefinisikan **dua cakram (discs)**, yang akan **diiris (intersect) 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**.

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

Tulis **dua irisan (intersections)** tersebut.

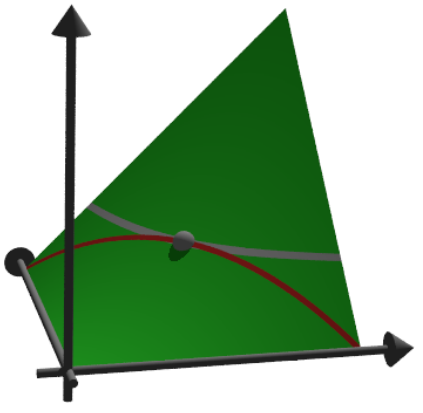
```
>writeln(povintersection([mesh,cl],povlook(red)); ...
>writeln(povintersection([mesh,ll],povlook(gray)));
```

Tulis **titik pada nilai maksimum**.

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

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

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



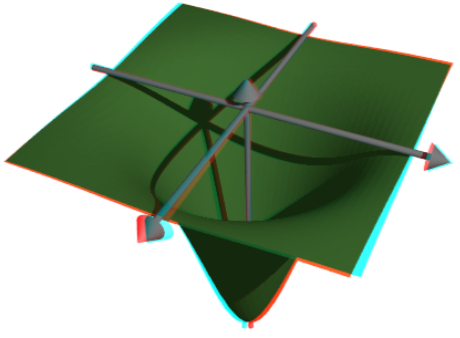
Anaglyphs di Povray

Untuk menghasilkan **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 berikut dengan benar.

Fungsi **pov3d()** memiliki **switch sederhana** untuk menghasilkan anaglyph.

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



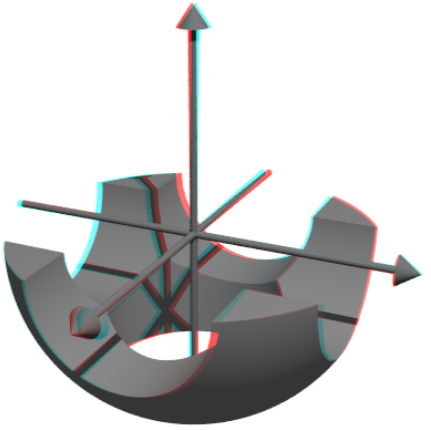
Jika Anda membuat **scene dengan objek**, Anda perlu **menempatkan pembuatan scene dalam sebuah fungsi**, lalu **menjalankannya dua kali** dengan **nilai parameter anaglyph yang berbeda**.

```
>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 ini. Parameternya **mirip dengan gabungan povstart() dan povend()**.

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



Mendefinisikan Objek Sendiri

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

Contohnya, kita mendemonstrasikan **torus**. Perintah Povray untuk ini adalah **"torus"**, sehingga kita mengembalikan **string dengan perintah dan parameternya**. Perlu diperhatikan bahwa **torus selalu berada di pusat (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 **diterjemahkan dan diputar**.

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

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

Sekarang, kita **menempatkan objek-objek ini ke dalam scene**. 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 **error**, **tidak menampilkan pesan kesalahan**.

Oleh karena itu, sebaiknya gunakan:

‘>povend(<exit);‘

jika ada yang **tidak berjalan dengan benar**. Ini akan **meninggalkan jendela Povray tetap terbuka**.

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

Berikut adalah contoh yang lebih **lengkap**. Kita menyelesaikan:
Ax \le b, \quad x \ge 0, \quad c \cdot x \to \text{Maksimum}
dan menampilkan **titik-titik layak (feasible points)** serta **nilai optimum** dalam **plot 3D**.

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

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

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

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

Ya, **solusi ada**.

Selanjutnya, kita mendefinisikan **dua objek**. Yang pertama adalah **bidang (plane)**:

$$a \cdot x \leq b$$

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

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

Kemudian, kita mendefinisikan **irisan dari semua ruang setengah (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 bisa **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 ini adalah **lingkaran di sekitar titik optimum**.

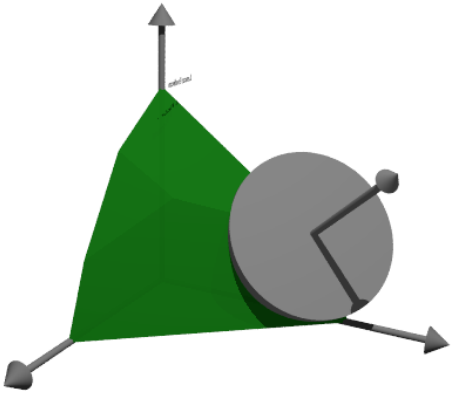
```
>writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')], ...
>  povlook(red,0.9)));
```

Berikut ini adalah **lingkaran di sekitar titik optimum**.

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

Dan sebuah **error** dalam arah menuju **titik optimum**. Kita menambahkan **teks** ke layar. Teks hanyalah sebuah **objek 3D**, yang perlu **ditempatkan dan diputar** sesuai **pandangan kita**.

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



Contoh Lainnya

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

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

```
>
```

Teori singkat (kalkulus / aljabar) grafik 3D Fungsi dua variabel:

fungsi

R,z=f(x,y) grafiknya adalah sebuah permukaan di ruang tiga dimensi:

={(x,y,z):z=f(x,y)}. Lokasi puncak, cekungan (minimum lokal), garis kontur (level set) dan gradien/arah tercepat kenaikan adalah konsep penting.
Permukaan parametrik: permukaan juga dapat diberikan oleh peta parametrik

(u,v)=(x(u,v),y(u,v),z(u,v)).

Banyak permukaan klasik (bola, torus, silinder melengkung) lebih mudah dinyatakan parametrik.
Permukaan implisit: ditentukan oleh persamaan

(x,y,z)=0. Ini berguna untuk benda yang tidak mudah ditulis sebagai fungsi

=f(x,y) (contoh: knotted surfaces, beberapa objek yg berputar).
Garis level / contour: pada ketinggian tetap

=c atau pada nilai fungsi tertentu, garis potong permukaan dengan bidang horizontal memberi gambaran topografi permukaan.

Derajat kehalusan & diskritisasi: komputer menampilkan permukaan dengan grid diskrit (mesh). Parameter seperti jumlah titik grid (n) mempengaruhi ketajaman/kehalusan hasil. **Opsi penting**

di EMT/Euler untuk plot3d (ringkas)

r, a,b,c,d — rentang plotting (symmetric atau pasangan x- & y-range).
n atau n=[nx,ny] — jumlah subdivisi/grid pada sumbu (lebih besar = lebih halus).

grid — jumlah garis grid yang akan digambar di atas surface (visual).
fscale, scale — penskalaan nilai fungsi dan skala pada sumbu x/y.
angle, height, distance, zoom — parameter kamera/view.
>hue, >contour, >spectral — gaya pewarnaan / garis level.
transparent, wire, wirecolor, linewidth — gaya permukaan/kurva.
implicit — untuk plotting himpunan nol

(x,y,z)=0 (menggunakan potongan).
>polar — menggambar dalam koordinat polar (fungsi tetap diberi sebagai fungsi x,y).
points=true, >anaglyph — point clouds dan anaglyph 3D.

Contoh masalah & penyelesaian (soal dalam format LaTeX di komentar + kode EMT + penjelasan)
Contoh 1 — Permukaan kuadratik sederhana

```
>plot3d("x^2+y^2",a=-2,b=2,c=-2,d=2,n=80,>spectral,>contour,zoom=3,grid=10)
```

- Hasil adalah paraboloid, pusat minimum di (0,0,0).
- spectral menonjolkan ketinggian z sebagai warna (biru ke merah).
- contour menggambar garis-garis ketinggian yang membantu melihat topografi.

Contoh 2 — Gaussian (gunung)

```
>plot3d("exp(-x^2-y^2)",r=3,n=120,level="thin",>contour,>hue,scale=1.1,zoom=4)
```

- Permukaan menyerupai puncak gaussian di (0,0).
- Grid n=120 memberi permukaan halus.
- hue memberikan shading lembut terutama bila ingin tampilan tanpa warna spektral.

Contoh 3 — Permukaan parametrik: bola (sphere)

```
>t=linspace(0,2*pi,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,grid=[10,20],frame=1)
```

- Plot menghasilkan permukaan bola dengan shading.
- grid=[10,20] mengatur jumlah garis grid untuk visualisasi meridian/parallel.

Contoh 4 — Kurva parametrik 3D: heliks (trigonometri / aljabar)

```
>t=linspace(0,8*pi,500);
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3,linewidth=2,wirecolor=blue)
```

- Kurva naik searah sumbu z membentuk screw/helix.
- wire memastikan kurva digambar sebagai polyline tebal.

Contoh 5 — Permukaan implisit

```
>plot3d("x^2+y^3+z*y-1",r=3,implicit=3,n=80,angle=-30°,zoom=2.5)
```

EMT menampilkan potongan (cuts) dari solusi di berbagai bidang, membentuk visualisasi lapangan solusi 3D.
implicit=3 memilih potongan tertentu; kombinasi bitmask memungkinkan potongan berbeda **Contoh**

lanjutan: Povray export, normals, dan animasi

Povray export: gunakan pov3d(...) atau rangkaian povstart() / povend() untuk menghasilkan render raytracing lebih realistis (bisa mengatur look=povlook(color,transparency,phong) dsb.). Contoh pada file EMT4Plot3D.pdf. ?filecite?turn0file0?
Normals & shading: untuk shading halus (Povray), sertakan normal matrix (turunan) agar hasil raytracer menghasilkan pencahayaan yang natural.
Animasi: gunakan rotate() wrapper atau buat loop yang memanggil addpage() pada setiap frame, lalu gabungkan menjadi animasi rotasi.

Tips praktis checklist

- Selalu tentukan rentang (r atau a,b,c,d) supaya plot fokus pada daerah yang diinginkan.
Jika permukaan ”kecil” relatif terhadap koordinat, matikan autoscale (scale=false) dan atur fscale.
Untuk publikasi, naikkan n (contoh 100–300) untuk halus; untuk eksplorasi cepat, gunakan n kecil (20–60).
Gunakan >contour untuk analisis topografi; gunakan >spectral untuk menonjolkan ketinggian.
Untuk permukaan kompleks, pertimbangkan plotting parametrik atau export ke Povray untuk render berkualitas.