

Nama : Marcelline Calya Padmarini
NIM : 23030630036
Kelas : Matematika E

Menggambar Plot 3D dengan EMT

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

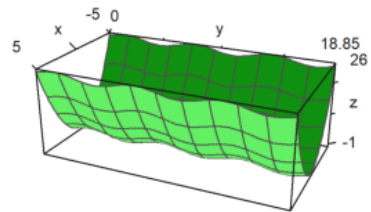
Euler menggambar fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan latar belakang. Secara umum, Euler menggunakan proyeksi pusat. Standarnya adalah dari kuadran x-y positif menuju titik asal $x=y=z=0$, tetapi sudut $=0^\circ$ terlihat dari arah sumbu y. Sudut pandang dan tinggi dapat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

Euler dapat merencanakan

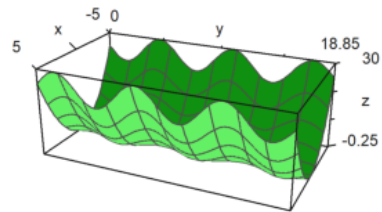
- permukaan dengan bayangan dan garis level atau rentang level,
- awan poin,
- kurva parametrik,
- permukaan implisit.

Plot 3D dari suatu fungsi menggunakan plot3d. Cara termudah adalah dengan memplot ekspresi dalam x dan y. Parameter r mengatur kisaran plot di sekitar (0,0).

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



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



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

Fungsi dua variabel

Untuk grafik fungsi, gunakan

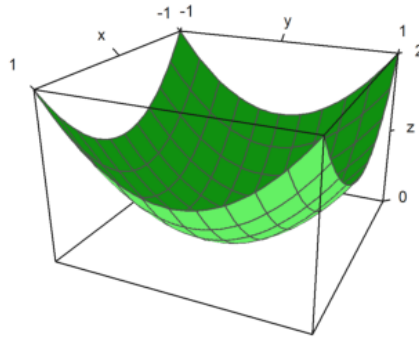
- ekspresi sederhana di x dan y,
- nama dari fungsi dua variabel
- atau matriks data.

Defaultnya adalah grid kawat yang terisi dengan warna berbeda di kedua sisi. Perhatikan bahwa jumlah interval kisi default adalah 10, tetapi plot menggunakan jumlah persegi panjang default 40x40 untuk membuat permukaan. Ini dapat diubah.

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

Kami menggunakan default $n=40$ dan $grid=10$.

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

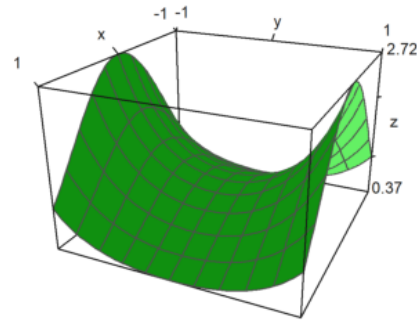


Interaksi pengguna memungkinkan dengan menggunakan parameter `>user`. Pengguna dapat menekan tombol berikut.

- left,right,up,down: memutar sudut pandang
- +,-: memperbesar atau memperkecil
- a: menghasilkan anaglyph (lihat di bawah)
- l: beralih memutar sumber cahaya (lihat dibawah)
- space: menyetel ulang ke default
- return: mengakhiri interaksi

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

Turn with the vector keys (press return to finish)



Rentang plot untuk fungsi dapat ditentukan dengan

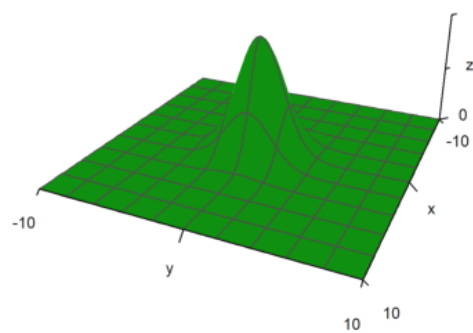
- a,b: rentang x
- c,d: rentang y
- r: persegi simetris di sekitar (0,0).
- n: jumlah subintegral plot

Ada beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik.

fscale: skala ke nilai fungsi (defaultnya <fscale)

frame: jenis bingkai (default 1)

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



Tampilan dapat diubah dengan berbagai cara.

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

Nilai default dapat diperiksa atau diubah dengan fungsi `view()`. Ini mengembalikan parameter dalam urutan di atas.

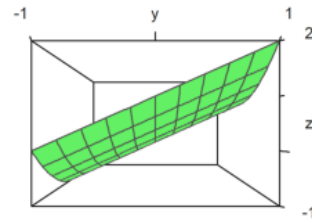
```
>view
```

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


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

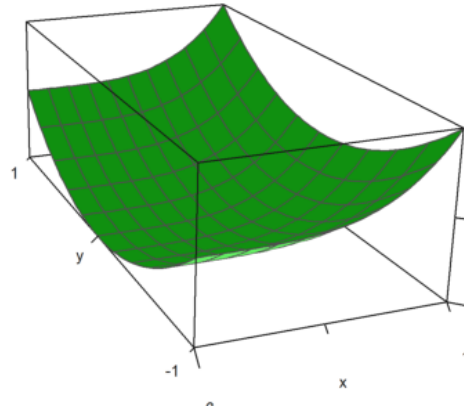
Dalam contoh berikut, sudut=0 dan tinggi=0 terlihat dari sumbu y negatif. Label sumbu untuk y disembunyikan dalam kasus ini.

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



Plot selalu terlihat ke pusat kubus plot. Anda dapat memindahkan pusat dengan parameter pusat.

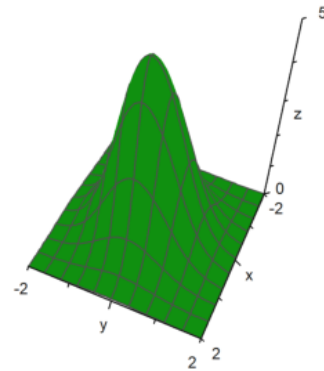
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...  
> center=[0.4,0,0],zoom=5):
```



Plot diskalakan agar sesuai dengan kubus satuan untuk dilihat. Jadi tidak perlu mengubah jarak atau zoom tergantung pada ukuran plot. Namun, label mengacu pada ukuran sebenarnya.

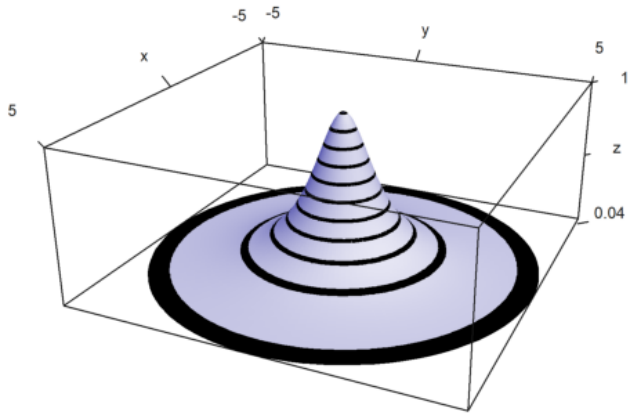
Jika Anda mematakannya dengan `scale=false`, Anda perlu berhati-hati, bahwa plot masih cocok dengan jendela plot, dengan mengubah jarak pandang atau zoom, dan memindahkan pusat.

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

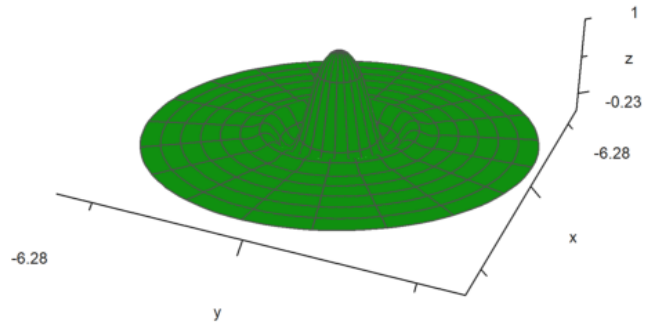


Sebuah plot kutub juga tersedia. Parameter `polar=true` menggambar plot polar. Fungsi tersebut harus tetap merupakan fungsi dari x dan y . Parameter `fscale` menskalakan fungsi dengan skala sendiri. Jika tidak, fungsi diskalakan agar sesuai dengan kubus.

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



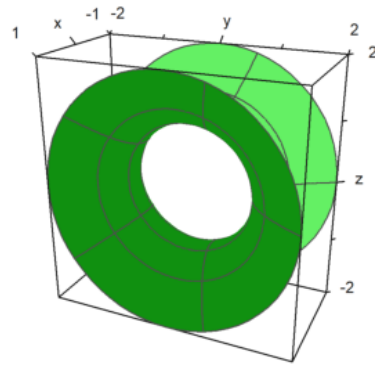
```
>function f(r) := exp(-r/2)*cos(r); ...  
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=2pi,frame=3,zoom=4):
```



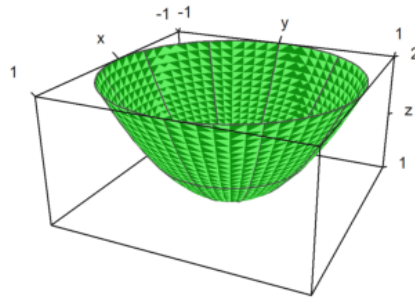
Rotasi parameter memutar fungsi dalam x di sekitar sumbu x.

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

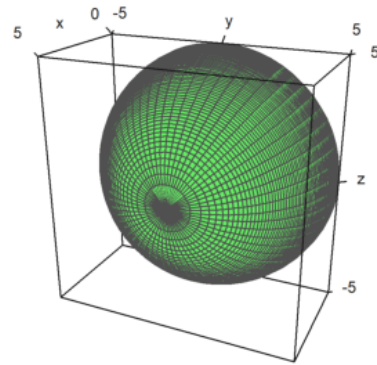
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



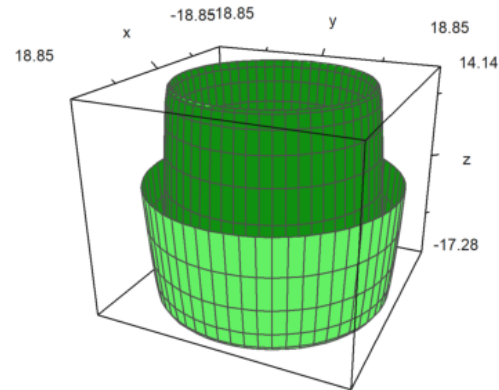
```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```



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

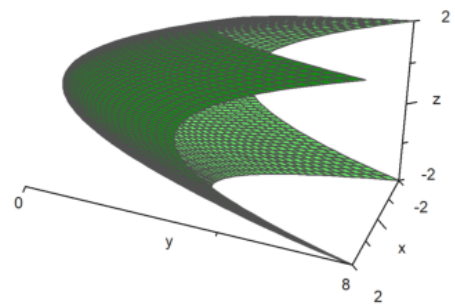


```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```

Ini adalah plot dengan 3 fungsi.

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



Plot Kontur

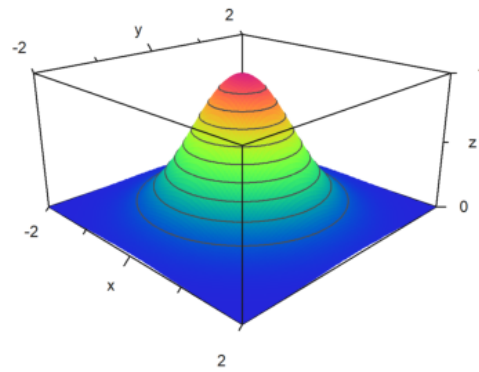
Untuk plot, Euler menambahkan garis grid. Sebagai gantinya dimungkinkan untuk menggunakan garis level dan rona satu warna atau rona berwarna spektral. Euler dapat menggambar tinggi fungsi pada plot dengan bayangan. Di semua plot 3D, Euler dapat menghasilkan anaglyph merah/sian.

- > hue: Menyalakan bayangan cahaya alih-alih kabel.
- > kontur: Memplot garis kontur otomatis pada plot.
- level=... (atau level): Sebuah vektor nilai untuk garis kontur.

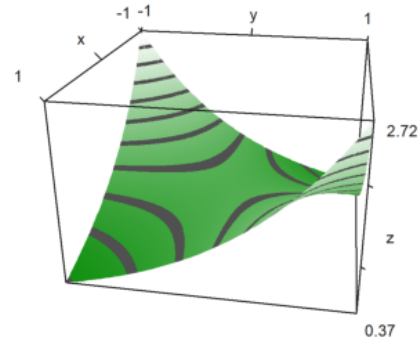
Standarnya adalah level="auto", yang menghitung beberapa garis level secara otomatis. Seperti yang Anda lihat di plot, level sebenarnya adalah rentang level.

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

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



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

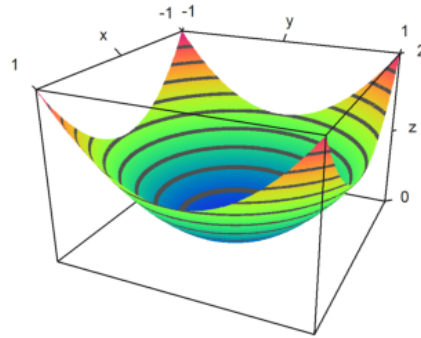


Bayangan default menggunakan warna abu-abu. Tetapi rentang warna spektral juga tersedia.

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

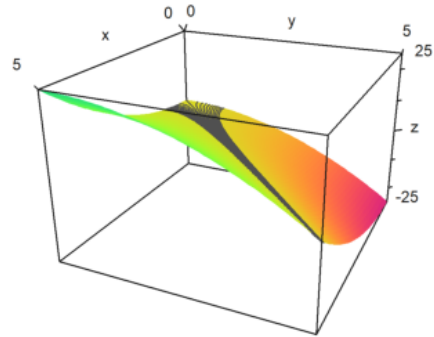
Untuk plot berikut, kami menggunakan skema spektral default dan menambah jumlah titik untuk mendapatkan tampilan yang sangat halus.

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



Alih-alih garis level otomatis, kita bisa mengatur nilai garis level. Ini akan menghasilkan garis level tipis alih-alih rentang level.

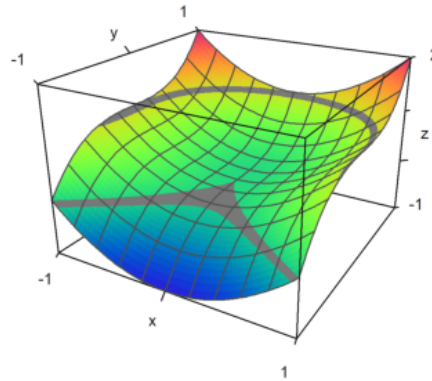
```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



Dalam plot berikut, kami menggunakan dua pita level yang sangat luas dari -0,1 hingga 1, dan dari 0,9 hingga 1. Ini dimasukkan sebagai matriks dengan batas level sebagai kolom.

Selain itu, kami melapisi kisi dengan 10 interval di setiap arah.

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

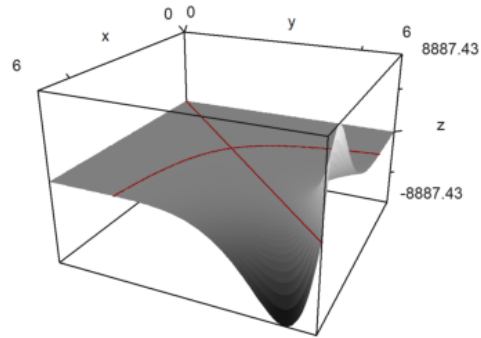


Pada contoh berikut, kita memplot himpunan, dimana

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

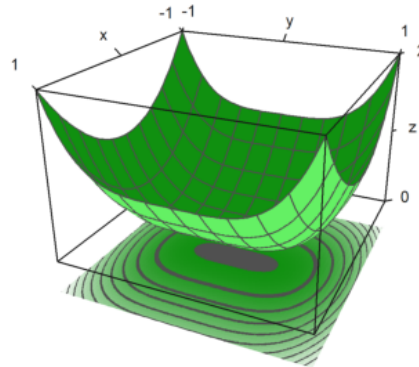
Kami menggunakan satu garis tipis untuk garis level.

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

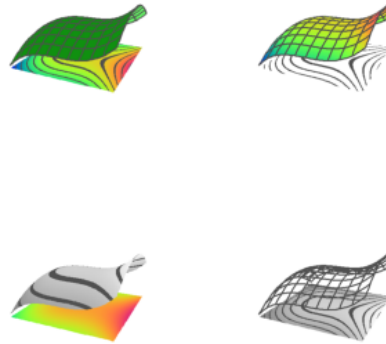
Dimungkinkan untuk menunjukkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

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



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

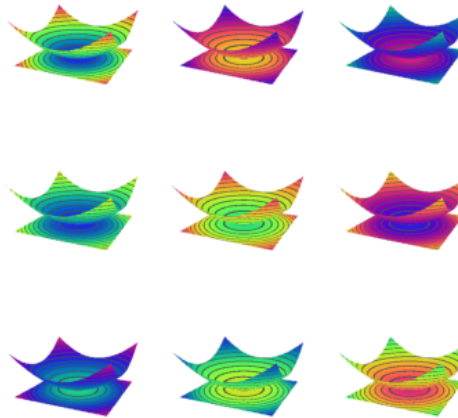
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1); ...
> plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2); ...
> plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3); ...
> plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4); ...
> plot3d(expr,<frame,>hue,grid=10,>transparent,>cp,cpcolor=gray);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



Ada beberapa skema spektral lain, bernomor 1 sampai 9. Tetapi Anda juga dapat menggunakan `color=value`, dimana nilai

- `spectral`: untuk rentang dari biru sampai merah
- `white`: untuk rentang yang lebih redup
- `yellowblue`, `purplegreen`, `blueyellow`, `greenred`
- `blueyellow`, `greenpurple`, `yellowblue`, `redgreen`

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



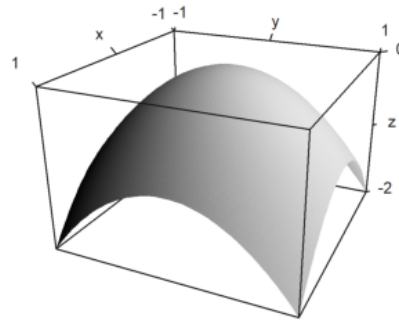
Sumber cahaya dapat diubah dengan `l` dan tombol kursor selama interaksi pengguna. Itu juga dapat diatur dengan parameter.

- `light`: arah untuk cahaya
- `amb`: cahaya sekitar antara 0 and 1

Perhatikan bahwa program tidak membuat perbedaan antara sisi plot. Tidak ada bayangan. Untuk ini, Anda perlu 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)":
```

Press I and cursor keys (return to exit)



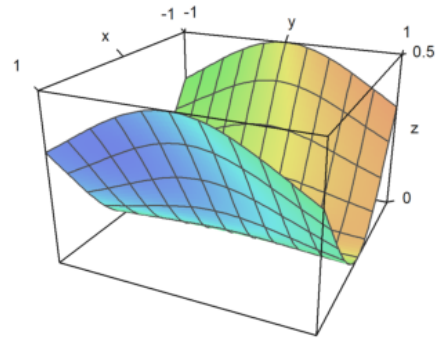
Parameter warna dapat mengubah warna permukaan. Warna garis level juga dapat diubah.

```
>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):
```



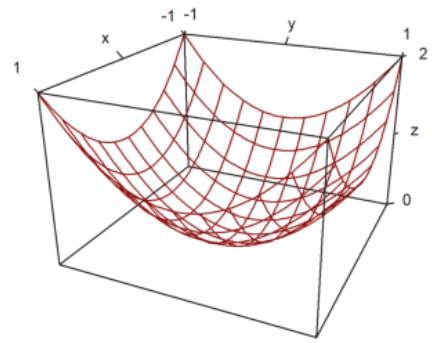
Warna 0 memberikan spesial efek pelangi.

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



Permukaannya juga bisa transparan.

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



Plot Implisit

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

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

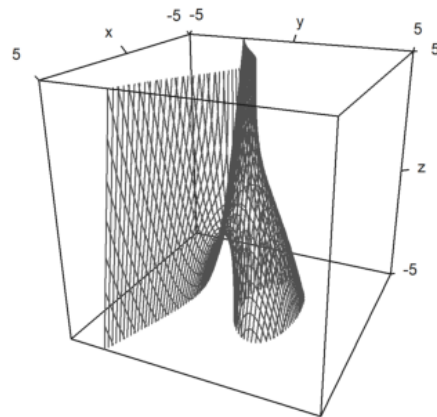
dapat divisualisasikan dalam potongan sejajar dengan bidang x-y-, x-z- dan y-z.

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

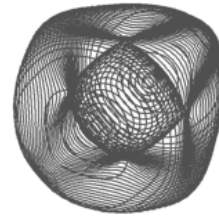
Tambahkan nilai-nilai ini, jika Anda suka. Dalam contoh kita plot

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

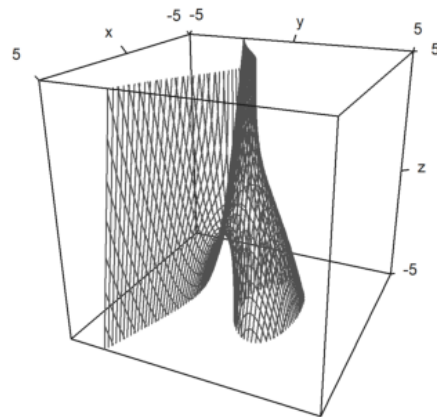
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```



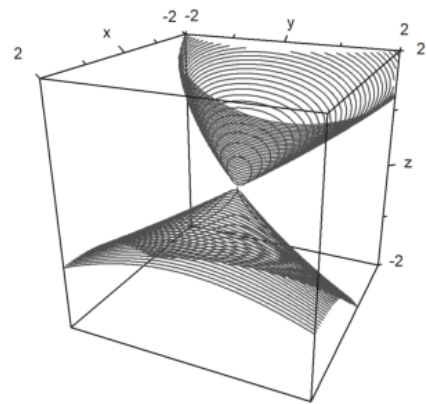
```
>c=1; d=1;
>plot3d("((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^2)-d",r=2,
```



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



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



Memplot Data 3D

Sama seperti `plot2d`, `plot3d` menerima data. Untuk objek 3D, Anda perlu menyediakan matriks nilai x -, y - dan z , atau tiga fungsi atau ekspresi $f_x(x,y)$, $f_y(x,y)$, $f_z(x,y)$.

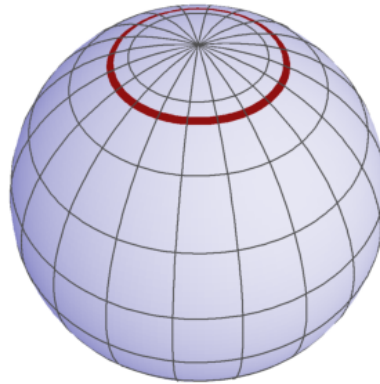
$$\gamma(t, s) = (x(t, s), y(t, s), z(t, s))$$

Karena x,y,z adalah matriks, kita asumsikan bahwa (t,s) melalui sebuah kotak persegi. Hasilnya, Anda dapat memplot gambar persegi panjang di ruang angkasa.

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

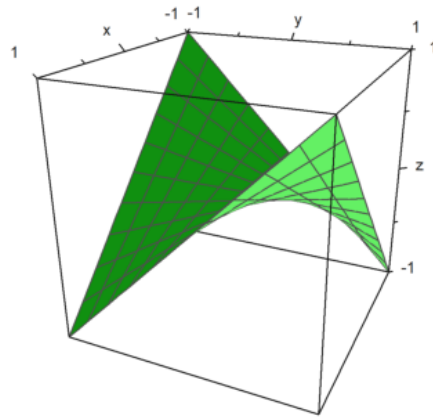
Dalam contoh berikut, kami menggunakan vektor nilai t dan vektor kolom nilai s untuk membuat parameter permukaan bola. Dalam gambar kita dapat menandai daerah, dalam kasus kita daerah polar.

```
>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°):
```



Ini adalah contoh yang termasuk sebuah fungsi grafik.

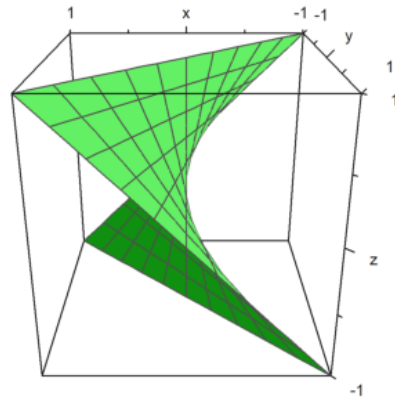
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



Namun, kita dapat membuat semua macam permukaan. Dengan menggunakan fungsi

$$x = y z$$

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

Dengan usaha lebih, kita dapat menghasilkan banyak permukaan.

Pada contoh berikut, kami membuat tampilan bayangan bola yang terdistorsi. Koordinat biasa dari bola adalah

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

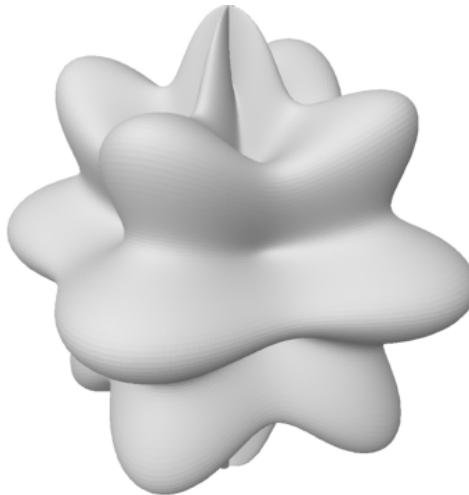
dengan

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

Kami mendistorsi dengan sebuah faktor

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

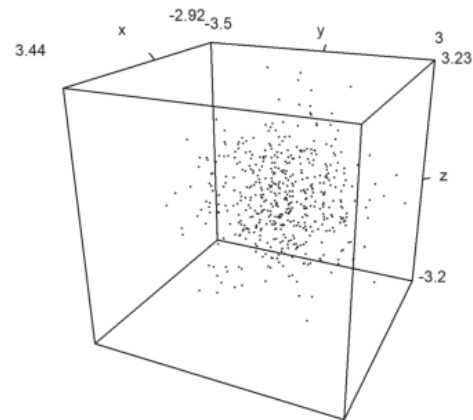
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...  
>d=1+0.2*(cos(4*t)+cos(8*s)); ...  
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...  
> light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, titik cloud juga dimungkinkan. Untuk memplot data titik dalam ruang, kita membutuhkan tiga vektor untuk koordinat titik-titik tersebut.

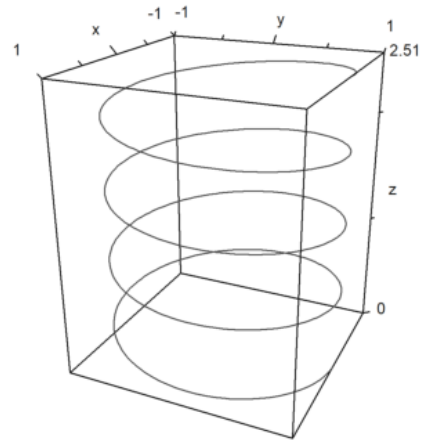
Gayanya sama seperti di plot2d dengan `points=true`;

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

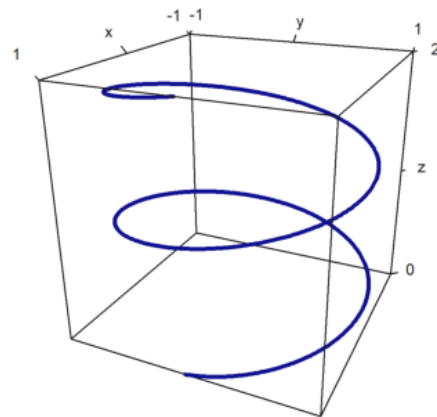


Dimungkinkan juga untuk memplot kurva dalam 3D. Dalam hal ini, lebih mudah untuk menghitung titik-titik kurva. Untuk kurva di pesawat kami menggunakan urutan koordinat dan parameter `wire=true`.

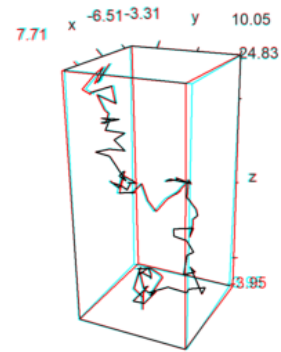
```
>t=linspace(0,8pi,500); ...  
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```



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

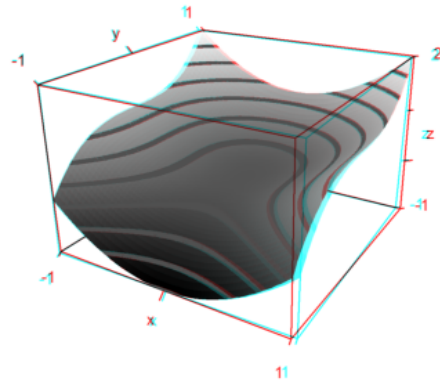


```
>X=cumsum(normal(3,100)); ...  
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



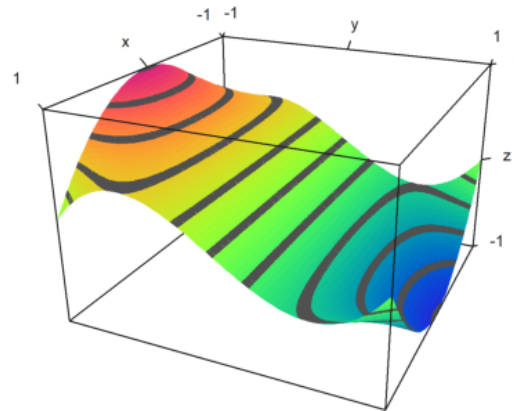
EMT juga dapat memplot mode anaglyph. Untuk melihat plot, Anda membutuhkan kacamata merah/cyan.

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



Seringkali, skema warna spektral digunakan untuk plot. Ini menekankan ketinggian fungsi

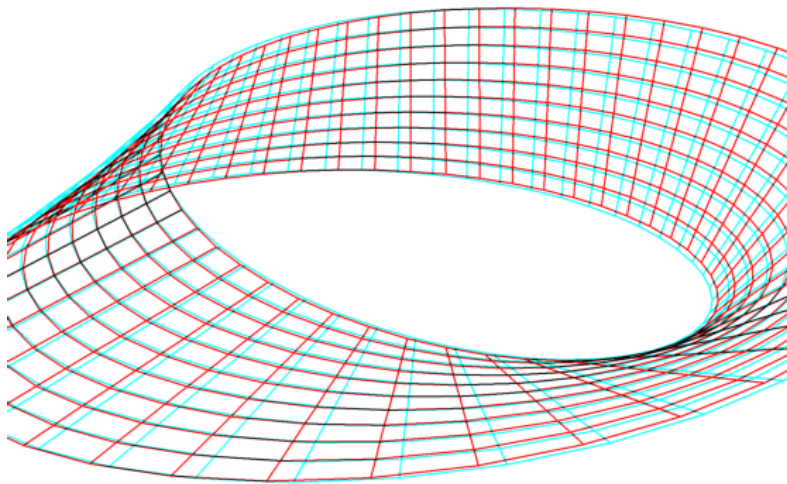
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler juga dapat memplot permukaan berparameter, ketika parameternya adalah nilai x -, y -, dan z dari gambar kotak persegi panjang dalam ruang.

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

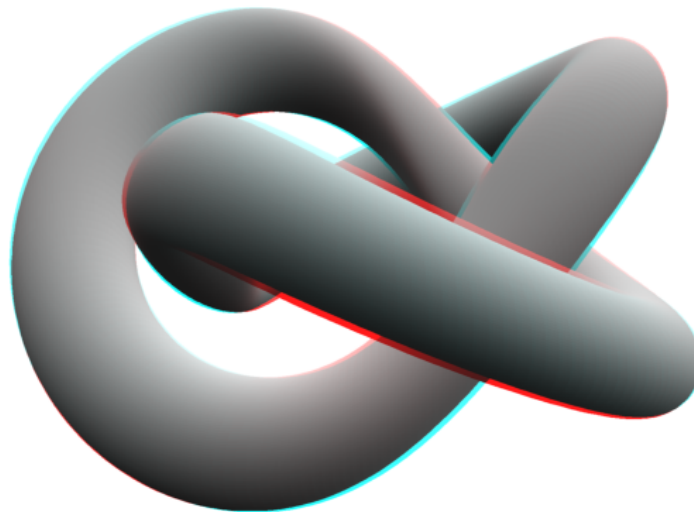
```
>u=linspace(-1,1,10); v=linspace(0,2*pi,50)'; ...
>X=(3+u*cos(v/2))*cos(v); Y=(3+u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3):
```

Ini adalah contoh lebih yang lebih komplikasi, dimana hanya dapat dilihat dengan kacamata merah/cyan.

```
>u:=linspace(-pi,pi,160); v:=linspace(-pi,pi,400)'; ...  
>x:=(4*(1+.25*sin(3*v))+cos(u))*cos(2*v); ...  
>y:=(4*(1+.25*sin(3*v))+cos(u))*sin(2*v); ...
```

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



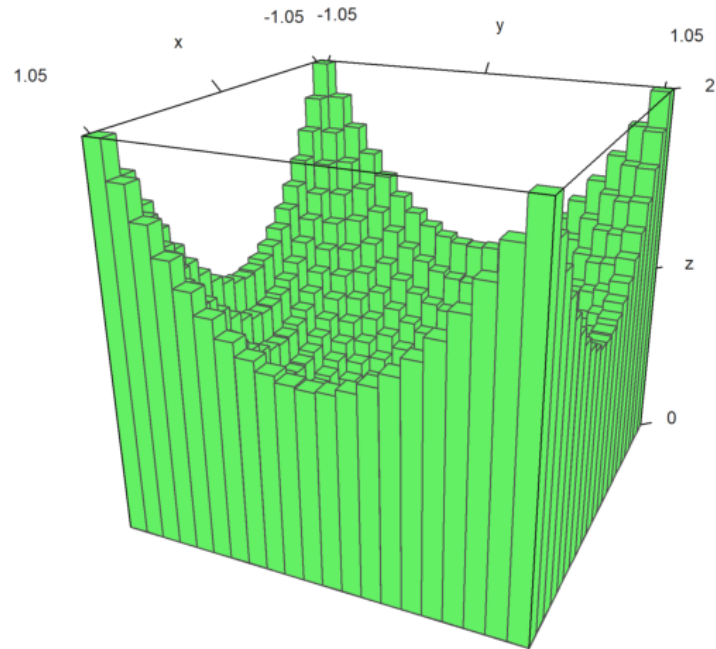
Plot bar juga dimungkinkan. Untuk ini, kita harus menyediakan

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

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

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

```
>x=-1:0.1:1; y=x'; z=x^2+y^2; ...  
>xa=(x|1.1)-0.05; ya=(y_1.1)-0.05; ...  
>plot3d(xa,ya,z,bar=true):
```



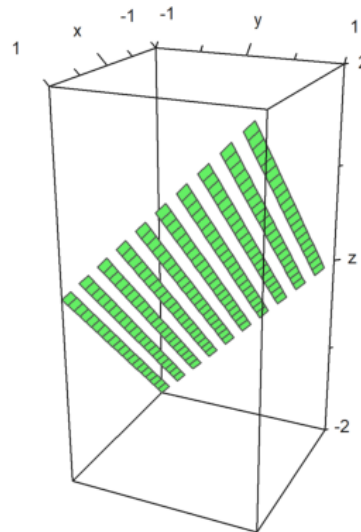
Dimungkinkan untuk membagi plot permukaan menjadi dua atau lebih bagian.
ediakan

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

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

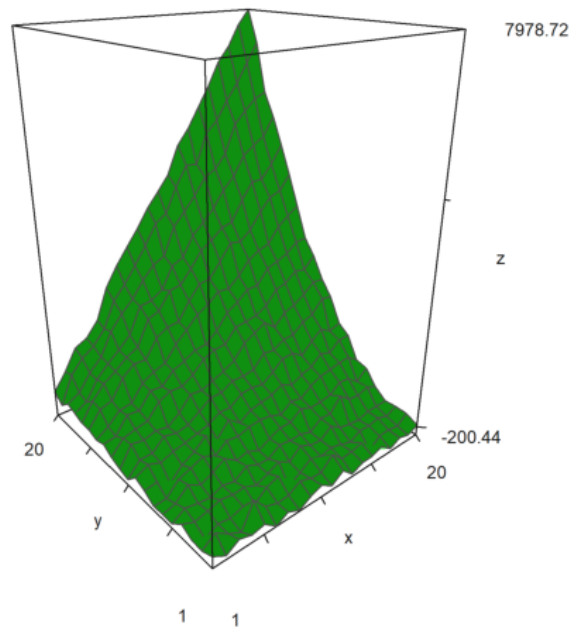
Dalam contoh, pertama-tama kita menghitung nilainya. Kemudian kita sesuaikan x dan y, sehingga vektor berpusat pada nilai yang digunakan.

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

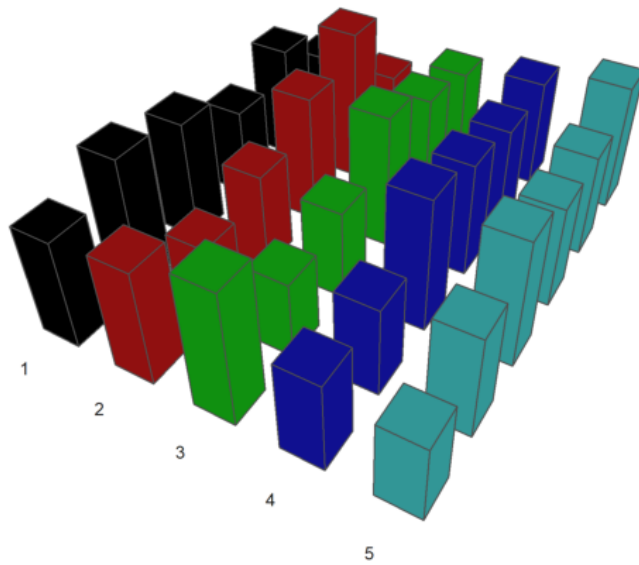


Jika memuat atau menghasilkan matriks data M dari file dan perlu memplotnya dalam 3D, Anda dapat menskalakan matriks ke [-1,1] dengan `scale(M)`, atau menskalakan matriks dengan `>zscale`. Ini dapat dikombinasikan dengan faktor penskalaan individu yang diterapkan sebagai tambahan.

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

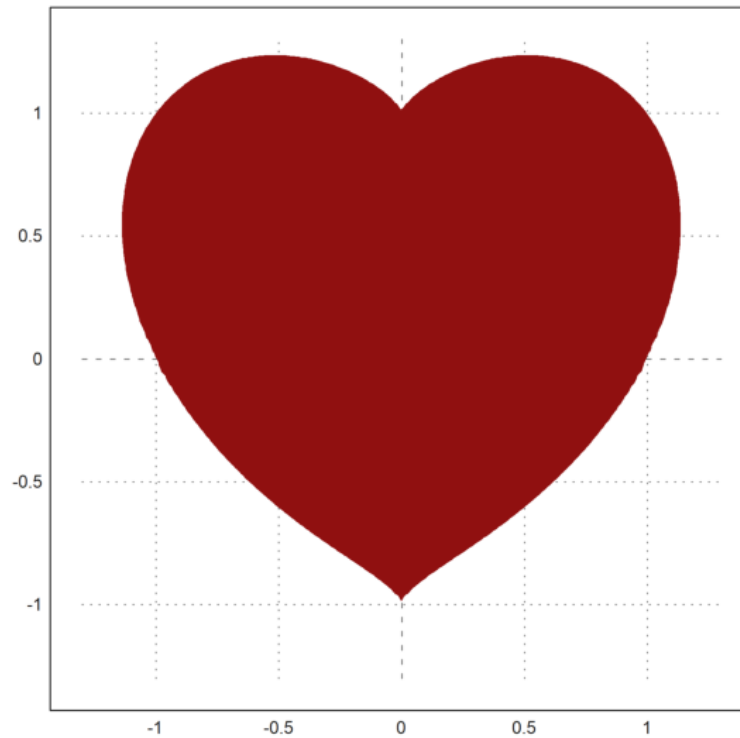


```
>Z=intrandom(5,100,6); v=zeros(5,6); ...  
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...  
>columnplot3d(v',scols=1:5,ccols=[1:5]):
```

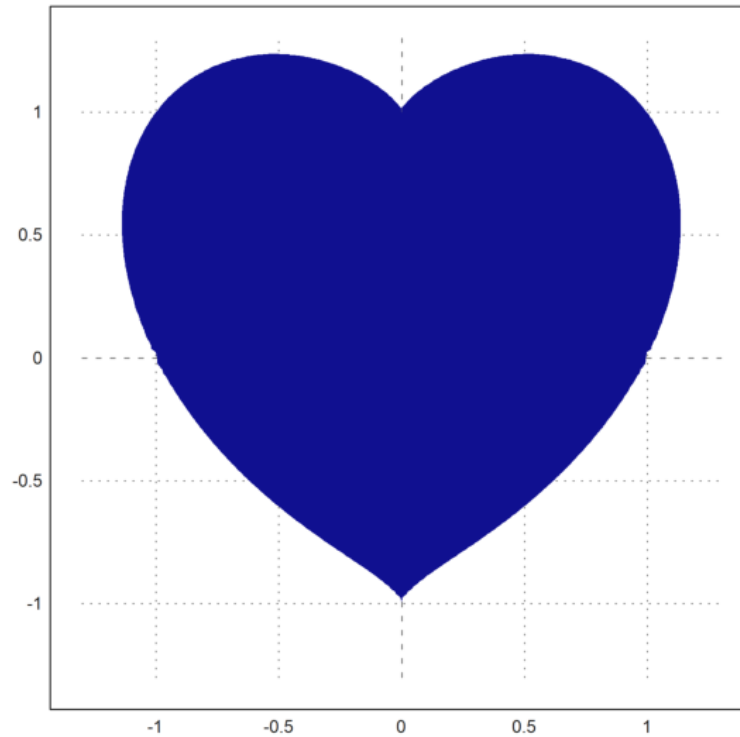


Permukaan Benda Putar

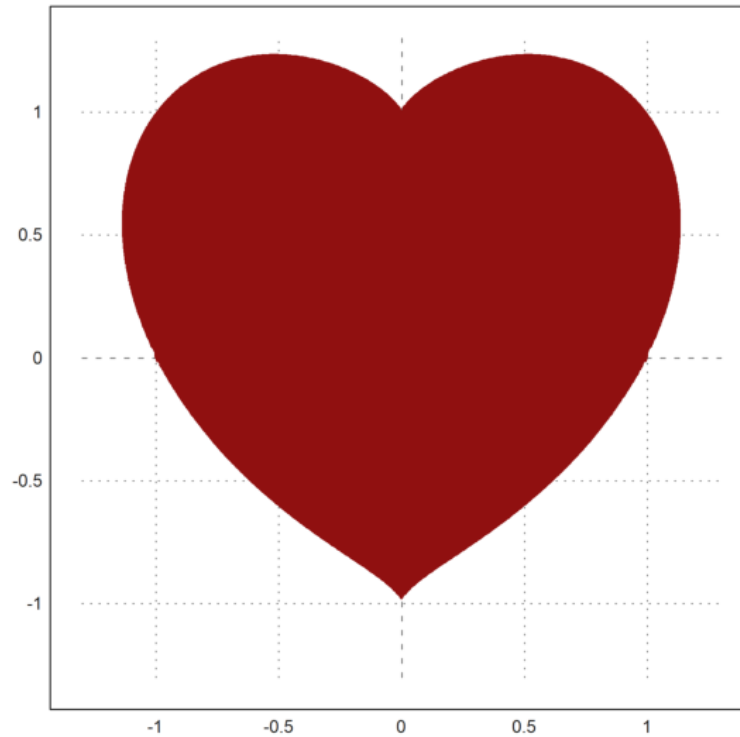
```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...  
>style="#",color=red,<outline, ...  
>level=[-2;0],n=100):
```




```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...  
>style="f",color=blue,<outline, ...  
>level=[-4;0],n=120):
```



```
>plot2d("(x^2+y^2-1)^3-x^2y^3",r=1.3, ...  
>style="#",color=red,<outline,...  
>level=[-2;0],n=150):
```



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

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

Kami ingin memmutar kurva hati disekitar sumbu-y. Berikut adalah ekspresi yang mendefinisikan hati:

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

kemudian himpunan

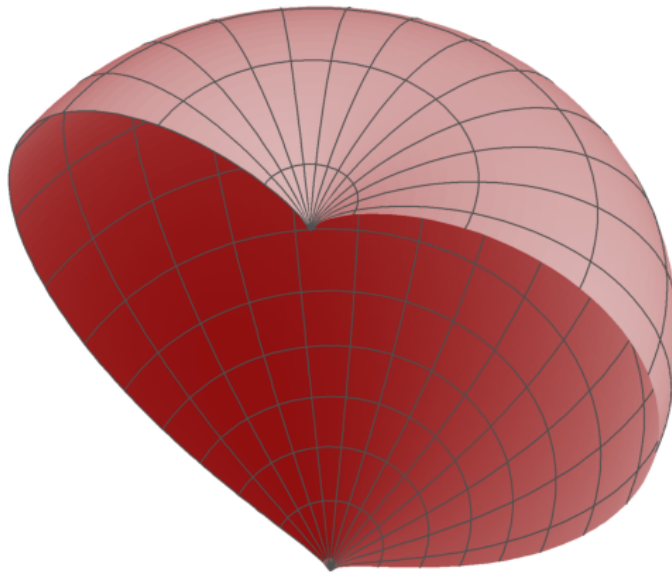
$$x = r.\cos(a), \quad y = r.\sin(a).$$

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

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

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

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

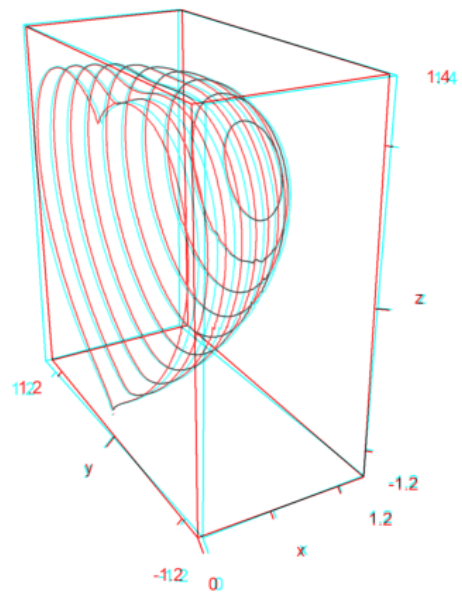


Berikut ini adalah plot 3D dari gambar di atas yang diputar di sekitar sumbu z. Kami mendefinisikan fungsi, yang menggambarkan objek.

```
>function f(x,y,z) ...
```

```
r=x^2+y^2;  
return (r+z^2-1)^3-r*z^3;  
endfunction
```

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



Plot 3D Khusus

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

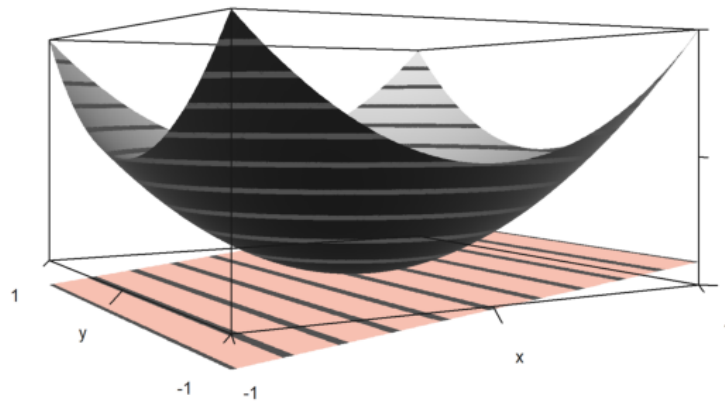
Meskipun Euler bukan program 3D, ia dapat menggabungkan beberapa objek dasar. Kami mencoba memvisualisasikan paraboloid dan garis singgungnya.

```
>function myplot ...
```

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

Sekarang `framedplot()` menyediakan frame dan mengatur tampilan.

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

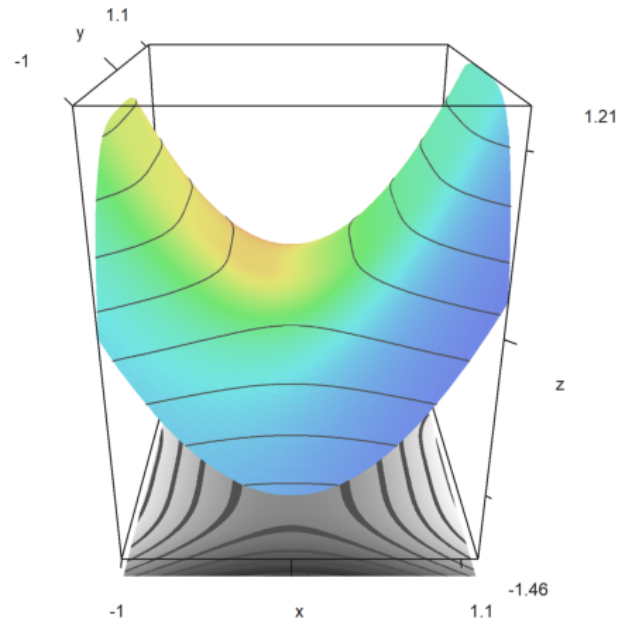
Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual. Perhatikan bahwa `plot3d()` menyetel jendela ke `fullwindow()` secara default, tetapi `plotcontourplane()` mengasumsikan itu. pa pun yang Anda suka.

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

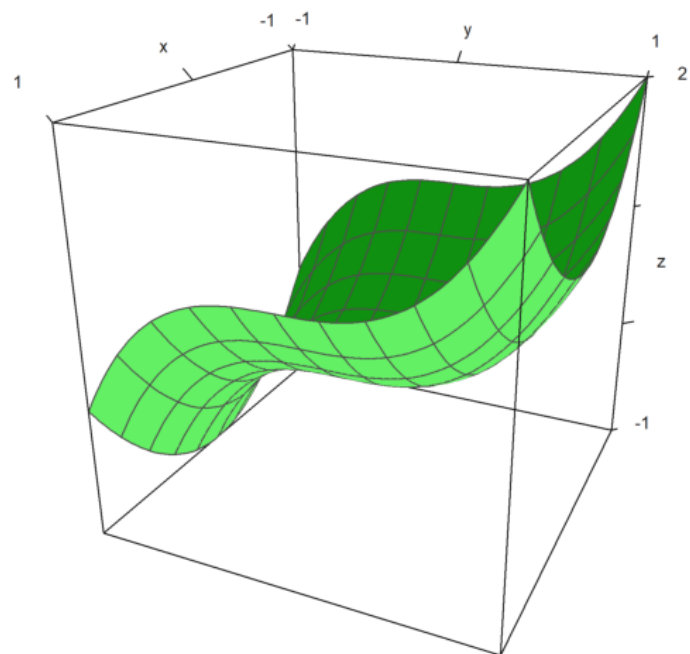
```
>x=-1:0.02:1.1; y=x'; z=x^2-y^4;  
>function myplot (x,y,z) ...
```

```
    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):
```



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



Euler dapat menggunakan frame untuk menghitung animasi terlebih dahulu.

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

Silakan pelajari sumber rotasi untuk melihat lebih detail.

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

Press space to stop, return to end

Press space to stop, return to end

Menggambar Povray

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

Anda perlu menginstal Povray (32bit atau 64bit) dari <http://www.povray.org/>, dan meletakkan sub-direktori "bin" dari Povray ke jalur lingkungan, atau mengatur variabel "defaultpovray" dengan path lengkap yang menunjuk ke "pvengine.exe".

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

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

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

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

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

Anda perlu memuat file povray

```
>load povray;
```

Pastikan, direktori bin Povray berada di jalur. Jika tidak, edit variabel berikut sehingga berisi path ke povray yang dapat dieksekusi.

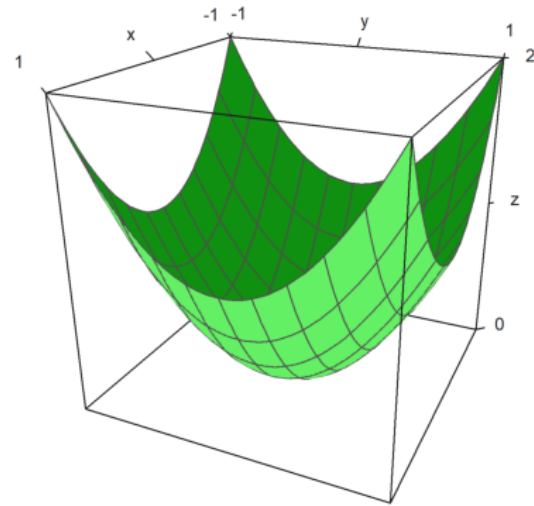
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

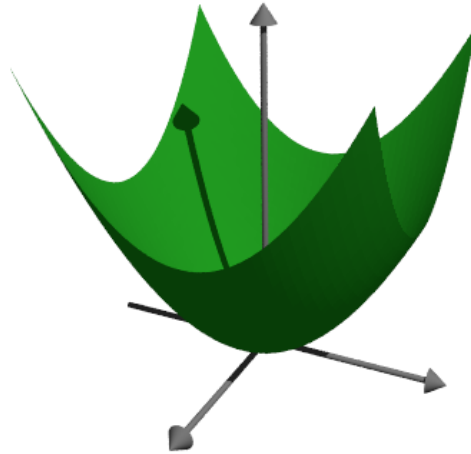
Untuk kesan pertama, kami memplot fungsi sederhana. Perintah berikut menghasilkan file povray di direktori pengguna Anda, dan menjalankan Povray untuk ray tracing file ini.

Jika Anda memulai perintah berikut, GUI Povray akan terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan ditanya, apakah Anda ingin mengizinkan file exe untuk dijalankan. Anda dapat menekan batal untuk menghentikan pertanyaan lebih lanjut. Anda mungkin harus menekan OK di jendela Povray untuk mengakui dialog awal Povray

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

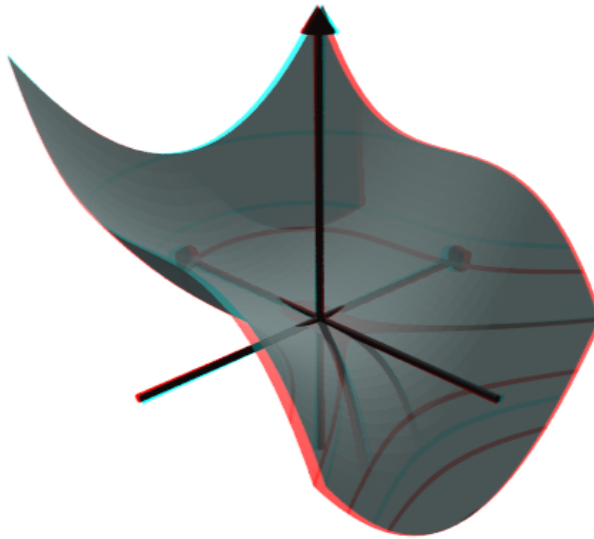


```
>pov3d("x^2+y^2",zoom=3);
```

Kita dapat membuat fungsi transparan dan menambah penyelesaian lain. Kita juga dapat menambah tingkat garis pada fungsi plot.

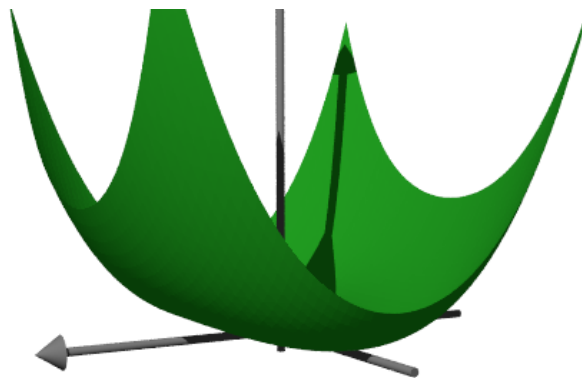
```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...  
> look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



Terkadang perlu untuk mencegah penskalaan fungsi, dan menskalakan fungsi dengan tangan.

Kami memplot himpunan titik di bidang kompleks, di mana produk dari 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);
```

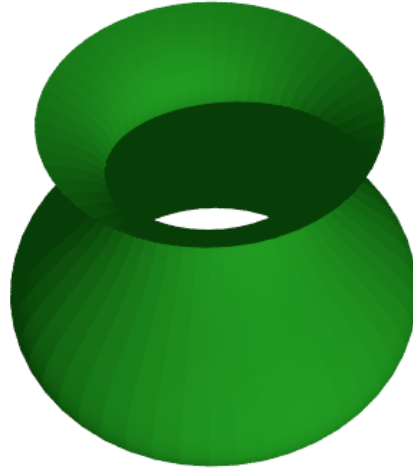


Memploting dengan Koordinat

Alih-alih menggunakan fungsi, kita dapat memplot dengan suatu koordinat. Sama seperti `plot3d`, kita membutuhkan 3 matriks untuk mendefinisikan objek.

Sebagai contoh, kita memutar fungsi di sekitar sumbu-z.

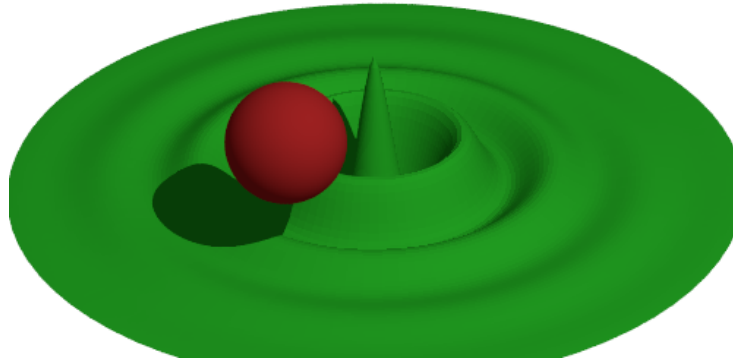
```
>function f(x) := x^3-x+1; ...  
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...  
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...  
>pov3d(X,Y,Z, angle=40°, height=50°, axis=0, zoom=4, light=[10,-5,5]);
```



Dalam contoh berikut, kami memplot gelombang teredam. Kami menghasilkan gelombang dengan bahasa matriks Euler.

Kami juga menunjukkan, bagaimana objek tambahan dapat ditambahkan ke adegan pov3d. Untuk pembuatan objek, lihat contoh berikut. Perhatikan bahwa plot3d menskalakan plot, sehingga cocok dengan kubus satuan.

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



Dengan metode bayangan canggih dari Povray, sangat sedikit titik yang dapat menghasilkan permukaan yang sangat halus. Hanya di perbatasan dan dalam bayang-bayang triknya mungkin menjadi jelas.

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

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

$$z = x^2 y^3$$

Persamaan permukannya adalah $[x,y,Z]$. Kami menghitung dua turunan ke x dan y , serta mengambil produk silang sebagai normal.

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

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

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

$[0, 0, 1]$

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);
```

Unexpected "(". Index () not allowed in strict mode!

In Euler files, use relax to avoid this.

Error in:

```
pov3d(x,y,Z(x,y),angle=10°;    xv=NX(x,y),yv=NY(x,y),zv=NZ(x,y) ...
```

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

See: [Examples\Trefoil Knot](#) | [Trefoil Knot](#)

Untuk tampilan yang bagus dengan tidak terlalu banyak titik, kami menambahkan vektor normal di sini. Kami menggunakan Maxima untuk menghitung normal bagi kami. Pertama, ketiga fungsi koordinat sebagai ekspresi simbolik.

```
>X &= ((4+sin(3*y))+cos(x))*cos(2*y); ...  
>Y &= ((4+sin(3*y))+cos(x))*sin(2*y); ...  
>Z &= sin(x)+2*cos(3*y);
```

Kemudian kedua vektor diturunkan ke x dan y.

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

Sekarang normal, yang merupakan produk silang dari dua turunan.

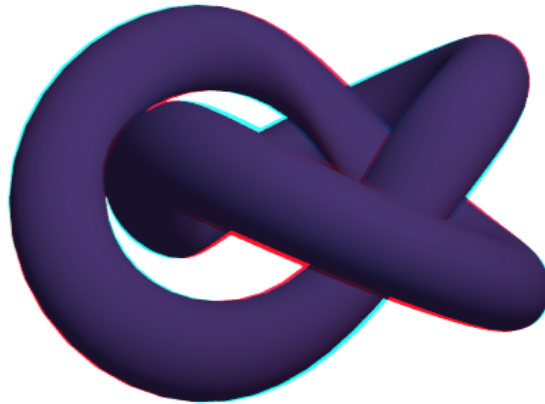
```
>dn &= crossproduct(dx,dy);
```


Kami sekarang mengevaluasi semua ini secara numerik.

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

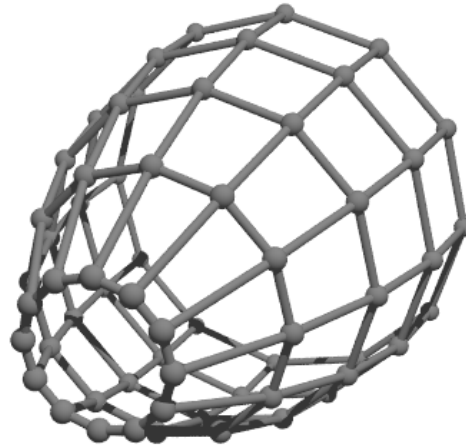
Vektor normal adalah evaluasi dari ekspresi simbolik $\text{dn}[i]$ untuk $i=1,2,3$. Sintaks untuk ini adalah `&"expression"(parameters)`. Ini adalah alternatif dari metode pada contoh sebelumnya, di mana kita mendefinisikan ekspresi simbolik NX, NY, NZ terlebih dahulu.
normal di sini. Kami menggunakan Maxima untuk menghitung normal bagi kami. Pertama, ketiga fungsi koordinat sebagai ekspresi simbolik.

```
>pov3d(X(x,y),Y(x,y),Z(x,y),>anaglyph,axis=0,zoom=5,w=450,h=350, ...  
> <shadow,look=povlook(blue), ...  
> xv=&"dn[1]"(x,y), yv=&"dn[2]"(x,y), zv=&"dn[3]"(x,y));
```



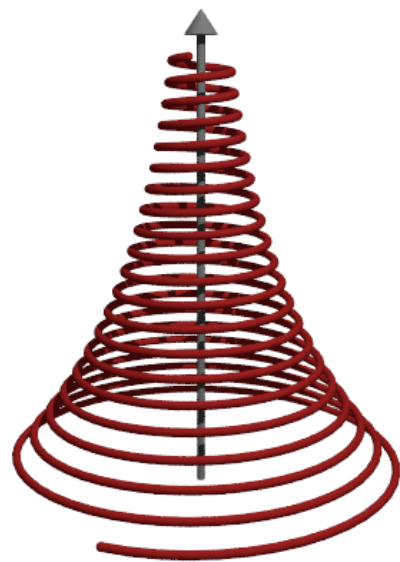
Kita juga dapat membuat sebuah grid di 3D.

```
>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 menggunakan `povgrid()`, kurva dimungkinkan.

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



Objek Povray

Di atas, kami menggunakan pov3d untuk memplot suatu permukaan. Antarmuka povray di Euler juga dapat menghasilkan objek Povray. Objek-objek ini disimpan sebagai string di Euler, dan perlu ditulis ke file Povray.

Kami mulai keluaran dengan povstart().

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

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

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

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

String 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 Anda lihat, kami menambahkan tekstur ke objek dalam tiga warna berbeda.

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

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

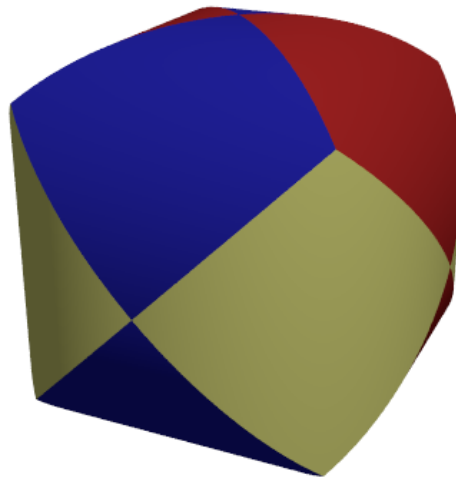
```
texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> } }
finish { ambient 0.5 }
```

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

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

Persimpangan tiga silinder sulit untuk divisualisasikan, jika Anda belum pernah melihatnya sebelumnya.

```
>povend;
```



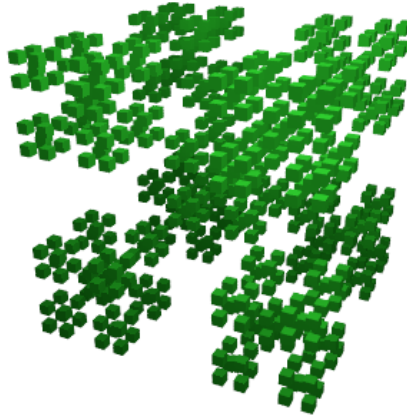
Fungsi berikut menghasilkan fraktal secara rekursif.

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

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

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

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

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

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

Untuk demonstrasi ini, kami mendefinisikan objek di Povray, alih-alih menggunakan string di Euler. Definisi ditulis ke file segera.

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

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

Kita dapat menggunakan objek ini pada povobject(), yang mengembalikan string seperti biasanya.

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

Kami menghasilkan kubus kedua dan memutar serta menskalakannya sedikit.

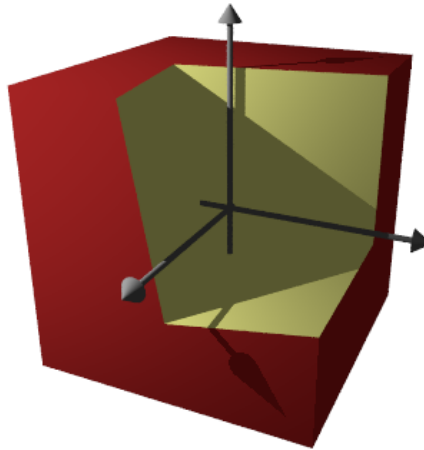
```
>c2=povobject("mycube",povlook(yellow),translate=[1,1,1], ...  
> rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Kemudian kami mengambil perbedaan dari kedua objek.

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

Sekarang tambahkan tiga sumbu.

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



Fungsi Implisit

Povray dapat memplot himpunan di mana $f(x,y,z)=0$, seperti parameter implisit di plot3d. Namun, hasilnya terlihat jauh lebih baik.

Sintaks untuk fungsinya sedikit berbeda. Anda tidak dapat menggunakan output dari ekspresi Maxima atau Euler.

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

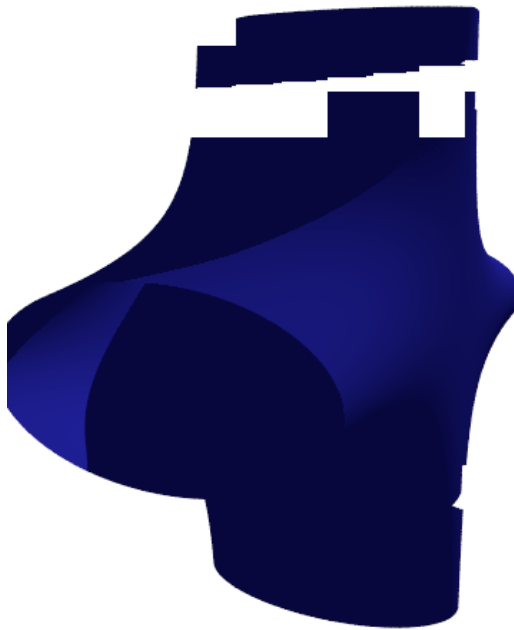
```
>povstart(angle=70°,height=50°,zoom=4);  
>writeAxes(); ...  
>writeln(povsurface("(pow(pow(x,2)+pow(y,2)-pow(c,2),2)+pow(pow(z,2)-1,2))*(pow(pow(y,2)+pow(z,2)-pow(c,2),2)+pow(pow(x,2)+pow(y,2)-1,2))"));  
>povend();
```

Error : Povray error!

Error generated by error() command

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

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



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

Buat permukaan implisit. Perhatikan perbedaan sintaks pada ekspresi.

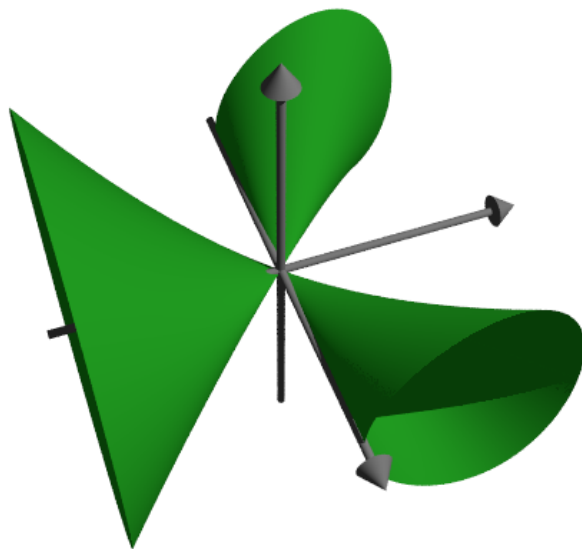
```
>writeln(povsurface("pow(x,2)*y-pow(y,3)-pow(z,2)",povlook(green))); ...  
>writeAxes(); ...  
>povend();
```

```

object {
  isosurface {
    function { pow(x,2)*y-pow(y,3)-pow(z,2) }
    max_gradient 5
    open
    contained_by { box { <-1,-1,-1>, <1,1,1>
      } }
    texture { pigment { color rgb <0.0627451,0.564706,0.0627451> } }
    finish { ambient 0.2 }
  }
}
union {
  cylinder { <-1.1,0,0>,<1.1,0,0>,0.02 }
  cone {
    <1.22,0,0>,0
    <1.1,0,0>,0.08
  }
  texture { pigment { color rgb <0.470588,0.470588,0.470588> } }
}
union {
  cylinder { <-1.1,0,0>,<1.1,0,0>,0.02 }
  cone {
    <1.22,0,0>,0
    <1.1,0,0>,0.08
  }
  rotate 90*z
  texture { pigment { color rgb <0.470588,0.470588,0.470588> } }
}
union {
  cylinder { <-1.1,0,0>,<1.1,0,0>,0.02 }
  cone {
    <1.22,0,0>,0
    <1.1,0,0>,0.08
  }
  rotate -90*y
  texture { pigment { color rgb <0.470588,0.470588,0.470588> } }
}

```

}



Objek Jala

Pada contoh ini, kami menunjukkan bagaimana membuat objek jala, dan menggambarinya dengan informasi tambahan.

Kami ingin memaksimalkan xy dibawah kondisi $x+y=1$ dan menunjukkan sentuhan tangen dari garis level.

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

Kami tidak dapat menyimpan objek dalam string seperti sebelumnya, karena terlalu besar. Jadi kita mendefinisikan objek dalam file Povray menggunakan declare. Fungsi povtriangle() melakukan ini secara otomatis. Itu dapat menerima vektor normal 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, yang akan berpotongan dengan permukaan.

```
>c1=povdisc([0.5,0.5,0],[1,1,0],2); ...  
>l1=povdisc([0,0,1/4],[0,0,1],2);
```


Tulis permukaan dikurangi dua cakram.

```
>writeln(povdifference(mesh,povunion([c1,l1]),povlook(green)));
```

Tulis dua persimpangan

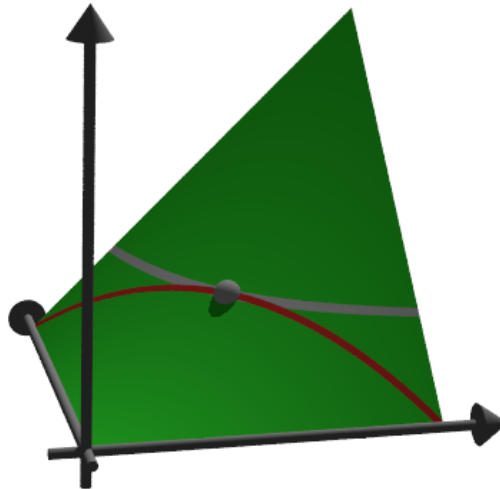
```
>writeln(povintersection([mesh,c1],povlook(red))); ...  
>writeln(povintersection([mesh,l1],povlook(gray)));
```

Tulis titik maksimum

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

Tambahkan sumbu dan selesaikan

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



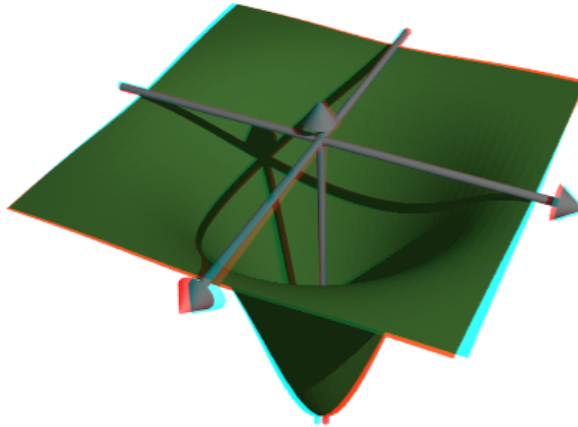
Anaglyphs di Povray

Untuk menghasilkan sebuah anaglyph untuk kacamata merah/cyan, Povray harus berjalan dua kali dari posisi yang berbeda. Ini menghasilkan dua file Povray dan dua file PNG yang dimuat dalam suatu fungsi `loadanaglyph()`.

Tentu saja, anda membutuhkan kacamata merah/cyan untuk melihat contoh berikut dengan benar.

Fungsi `pov3d()` memiliki sakelar sederhana untuk menghasilkan anaglyphs.

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



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

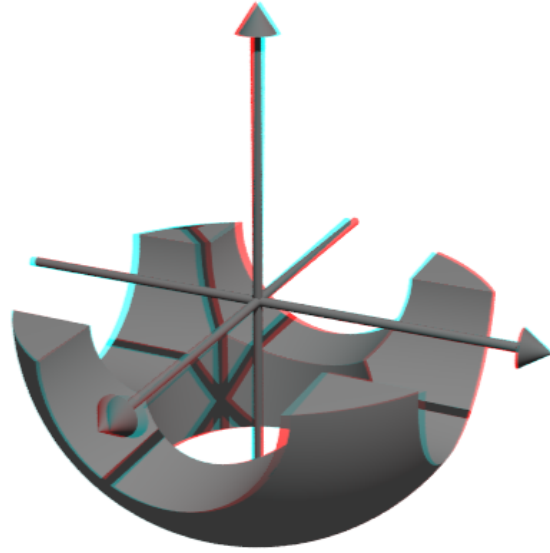
```
>function myscene ...
```

```
s=povsphere(povc,1);  
cl=povcylinder(-povz,povz,0.5);  
clx=povobject(cl,rotate=xrotate(90°));  
cly=povobject(cl,rotate=yrotate(90°));  
c=povbox([-1,-1,0],1);
```

```
un=povunion([cl,clx,cly,c]);  
obj=povdifference(s,un,povlook(red));  
writeln(obj);  
writeAxes();  
endfunction
```

Fungsi `povanaglyph()` melakukan semua ini. Parameternya seperti di `povstart()` dan `povend()` digabungkan.
lankannya dua kali dengan nilai yang berbeda untuk parameter `anaglyph`

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



Mendefinisikan Objek Sendiri

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

Kami mendemonstrasikan sebuah torus. Perintah Povray untuk ini adalah "torus". Jadi kami mengembalikan string dengan perintah ini dan parameternya. Perhatikan bahwa torus selalu berpusat di titik asal.

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

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

iniilah torus pertama kami

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

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

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

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

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

Sekarang kita menempatkan objek-objek ini ke dalam sebuah adegan. Untuk tampilan, kami menggunakan Phong Shading.

```
>povstart(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); ...  
>writeln(povobject(t1,povlook(green,phong=1))); ...  
>writeln(povobject(t2,povlook(green,phong=1))); ...
```

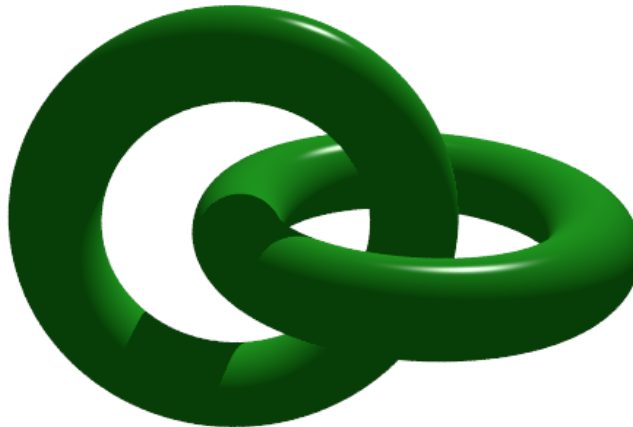
```
>povend();
```

memanggil program Povray. Namun, dalam kasus error, ini tidak menunjukan letak kesalahan. Karena itu Anda harus menggunakan

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


jika ada yang tidak berhasil. Ini akan membiarkan jendela Povray terbuka.

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



Ini adalah contoh yang lebih rumit. Kita selesaikan

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

dan tunjukkan titik layak serta optimal 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 cek, jika contoh ini memiliki solusi.

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

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

Ya, sudah.

Kemudian kita definisikan dua objek. Yang pertama adalah

$$a \cdot x \leq b$$

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

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

Kemudian kita mendefinisikan persimpangan dari semua setengah ruang dan sebuah kubus.

```
>function adm (A, b, r, look="") ...
```

```
    ol=[];  
    loop 1 to rows(A); ol=ol|oneplane(A[#],b[#]); end;  
    ol=ol|povbox([0,0,0],[r,r,r]);  
    return povintersection(ol,look);  
endfunction
```

Kita sekarang dapat merencanakan adegannya.

```
>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 optimal.

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

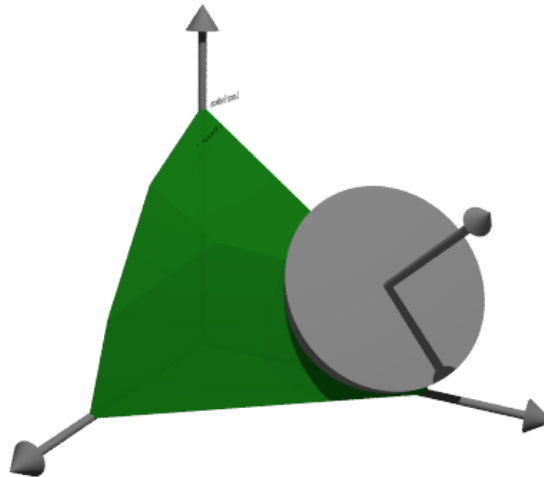
Dan sebuah kesalahan di direktori optimum.

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

Kami menambahkan teks ke layar. Teks hanyalah objek 3D. Kita perlu menempatkan dan memutarinya menurut pandangan kita.

tunjukan titik layak serta optimal dalam plot 3D.

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



Contoh Lain

Anda dapat mencari beberapa contoh lain dari Povray di Euler pada file berikut.

See: Examples/Dandelin Spheres

See: Examples/Donat Math

See: Examples/Trefoil Knot

See: Examples/Optimization by Affine Scaling

```
>load affinescaling;  
>type affinescaling
```

```
function affinescaling (A: real, b: column real, c: vector real, ..  
gamma: number positive, x: none column real, history, infinite ..  
)
```

```
## Default for gamma : 0.9  
## Default for x : none  
## Default for history : 0  
## Default for infinite : 450359962737
```

```
    n=cols(A);  
    if x==none then  
        u=b-sum(A);  
        A=A|u;  
        x=ones(cols(A),1);  
        c=c|infinite;  
    endif;  
    z=c.x;  
    if history then X=x; endif;  
    repeat
```

```

        d=ones(1,cols(A));
        d=x';
        tA=A*d; tc=c*d;
        y=fit(tA',tc');
        v=tc'-tA'.y;
        while !all(v~=0);
        if all(v<=0) then
            error("Problem unbounded");
        endif;
        i=nonzeros(v>0);
        lambda=gamma/max(v[i]');
        tx=x/d'-lambda*v;
        x=d'*tx;
        znew=c.x;
        if history then X=X|x; endif;
        while znew<z;
            z=znew;
        end
        if history then return {x[1:n],X[1:n]};
        else return x[1:n];
        endif;
    endfunction

```

```
>{xopt,X}=affinescaling(A,b,c,gamma=0.5,x=[1,1,1]',>history); xopt,
```

```

1
1
1

```

```
>A=[1,1;4,5]|id(2)
```

1	1	1	0
4	5	0	1

```
>b=[1000;4500]
```

1000
4500

```
>c=-[5,6,0,0]
```

-5,	-6,	0,	0]
-----	-----	----	----

```
>simplex(A,b,c,eq=0,>min,>check)
```

500
500
0
0

```
>affinescaling(A,b,c)
```

499.989

500.01

4.55637e-08

6.91608e-08