

Menggambar Plot 3D dengan EMT

Modul ini adalah pengenalan plot 3D di Euler. Kita memerlukan plot 3D untuk memvisualisasikan fungsi dua variabel.

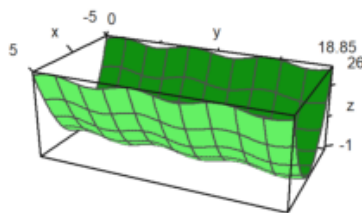
Euler menggambar fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan bagian di latar belakang. Secara umum Euler menggunakan proyeksi sentral. Defaultnya adalah dari kuadran x-y positif menuju titik asal $x=y=z=0$, tetapi sudut= 0° dilihat dari arah sumbu y. Sudut pandang dan ketinggian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

Euler bisa memplot

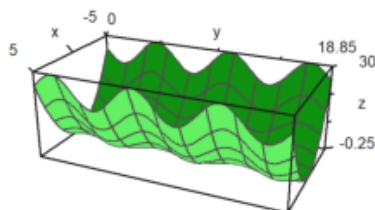
- permukaan dengan garis bayangan dan level atau rentang level,
- titik-titik langit,
- kurva parametrik,
- permukaan implisit.

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

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

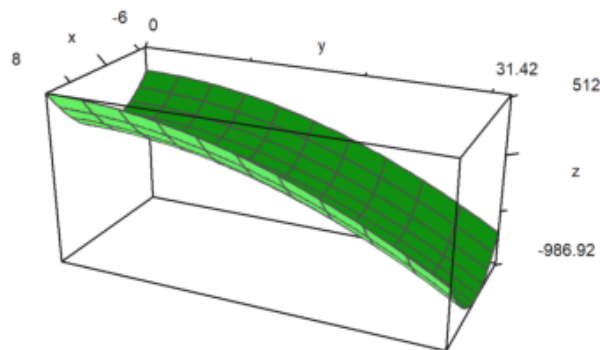


```
>plot3d("x^2+x*sin(y)",-5,5,0,6*pi):
```



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

```
>aspect(1); plot3d("8x^2-y^2",-6,8,0,10*pi):
```



```
>reset();
```

Fungsi Dua Variabel

Untuk grafik suatu fungsi, gunakan

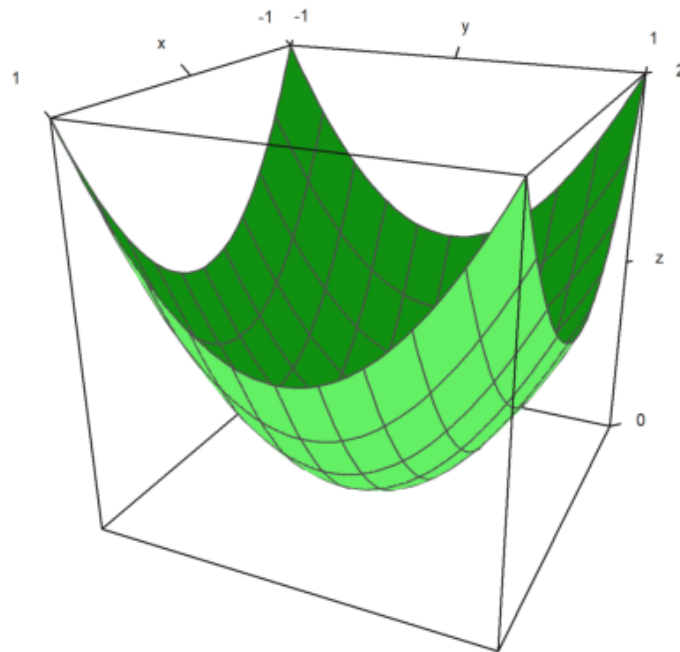
- ekspresi sederhana dalam x dan y ,
- nama fungsi dari dua variabel,
- atau matriks data.

Standarnya adalah kisi-kisi kawat berisi dengan warna berbeda di kedua sisi. Perhatikan bahwa jumlah interval kisi default adalah 10, tetapi plot menggunakan jumlah default persegi panjang 40×40 untuk membuat permukaannya. Ini dapat diubah.

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

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

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

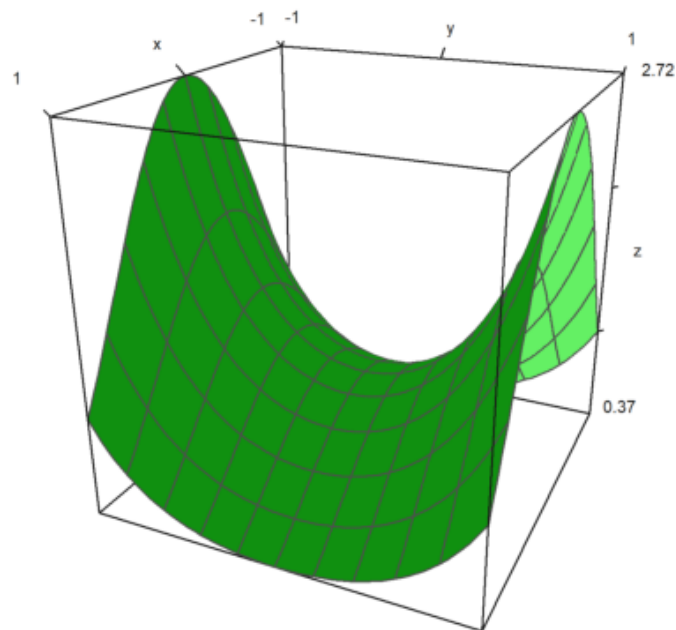


Interaksi pengguna dimungkinkan dengan parameter `> pengguna`. Pengguna dapat menekan tombol berikut.

- kiri, kanan, atas, bawah: memutar sudut pandang
 - +, -: memperbesar atau memperkecil
 - a: menghasilkan anaglyph (lihat di bawah)
 - l: tombol nyalakan sumber cahaya (lihat dibawah)
 - spasi: reset ke default
 - kembali: akhiri interaksi
- `x40` untuk membuat permukaannya. Ini dapat diubah.
- `n=40, n=[40,40]`: jumlah garis kisi di setiap arah
 - `grid=10, grid=[10,10]`: jumlah garis grid di setiap arah
- Kami menggunakan default `n=40` dan `grid=10`.

```
>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 subinterval untuk plot

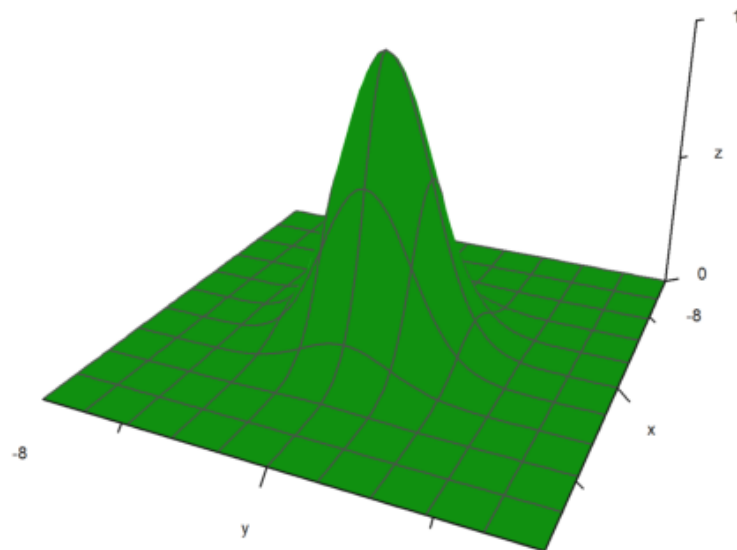
Ada beberapa parameter untuk membuat skala fungsi atau mengubah tampilan grafik.

fscale: membuat skala ke nilai fungsi (defaultnya adalah <fscale>).

scale: angka atau vektor 1x2 untuk membuat skala ke arah x dan y.

frame: jenis bingkai (default 1).

```
>plot3d("exp(-(x^2+y^2)/5)",r=8,n=60,fscale=5,scale=1.2,frame=3,>user):
```



Tampilan dapat diubah dengan berbagai cara yang berbeda.

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

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

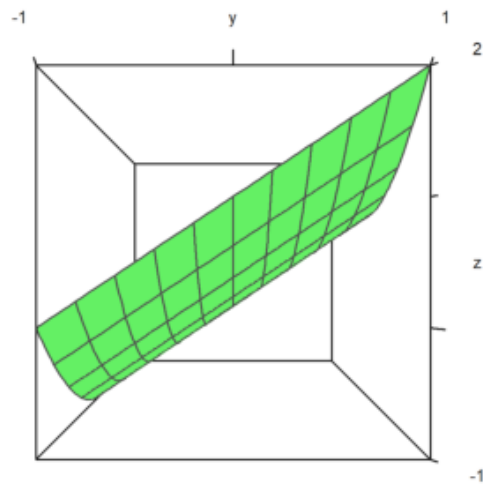
```
>view
```

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

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

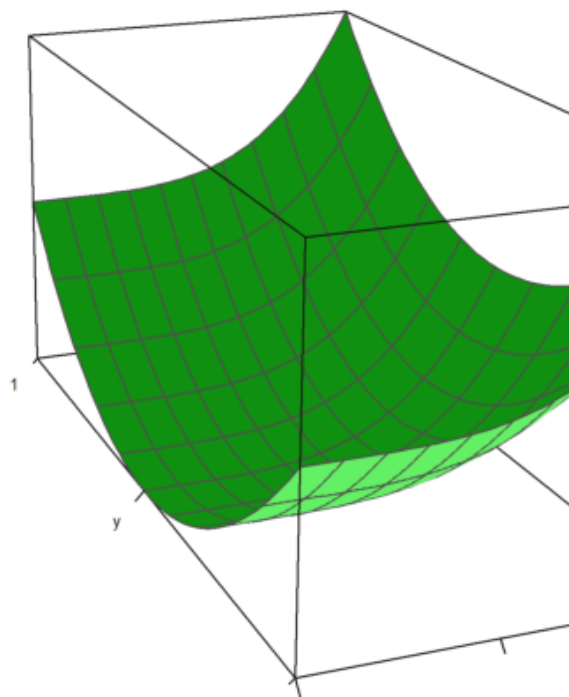
Pada contoh berikut, `angle=0` dan `height=0` dilihat dari sumbu y negatif. Label sumbu untuk y disembunyikan dalam kasus ini.

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



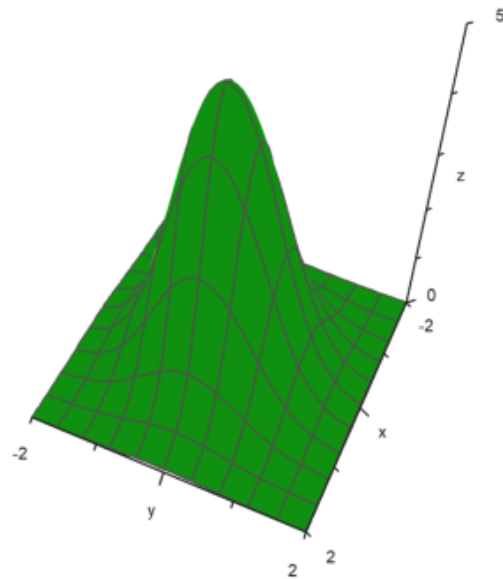
Plot selalu terlihat ke tengah kubus plot. Anda dapat memindahkan pusat dengan parameter pusat.

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



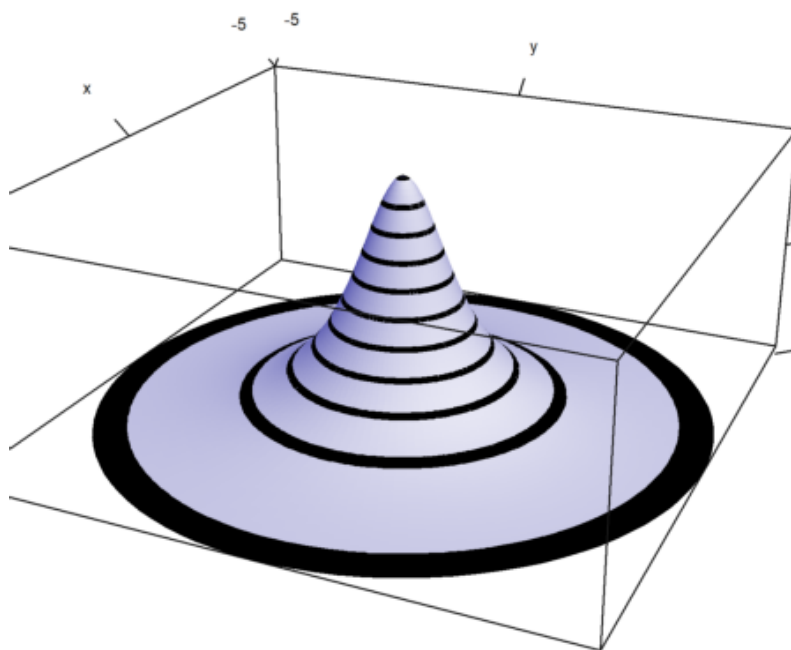
Plot tersebut diubah skala untuk muat ke dalam kubus satuan saat ditampilkan. Jadi, tidak perlu mengubah jarak atau zoom tergantung pada ukuran plot. Label-label merujuk pada ukuran yang sebenarnya, namun. Jika Anda mematikan ini dengan `scale=false`, Anda perlu memastikan bahwa plot masih muat ke dalam jendela plot dengan mengubah jarak pandang atau zoom, dan memindahkan pusatnya.

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

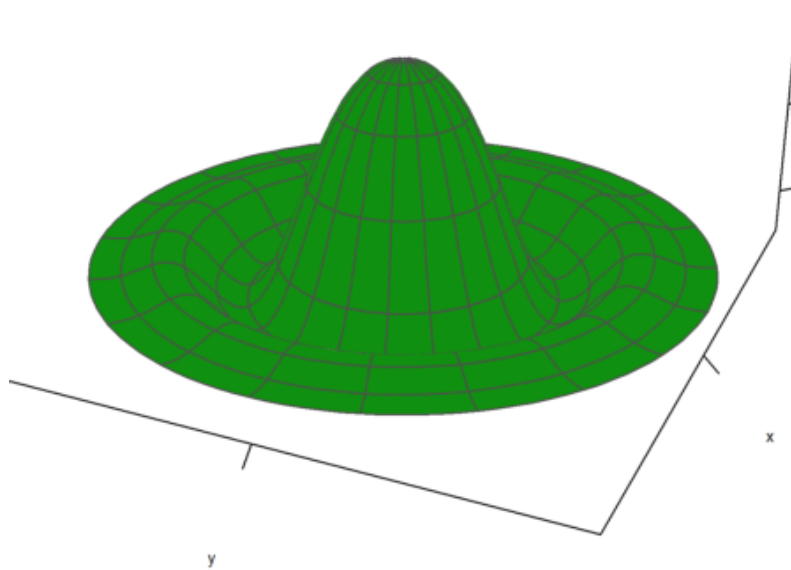


Tersedia juga grafik polar. Parameter `polar=true` menggambar grafik polar. Fungsi tetap harus menjadi fungsi dari x dan y . Parameter `"fscale"` mengubah skala fungsi dengan skala sendiri. Selain itu, fungsi akan disesuaikan dengan ukuran kubus.

```
>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):
```

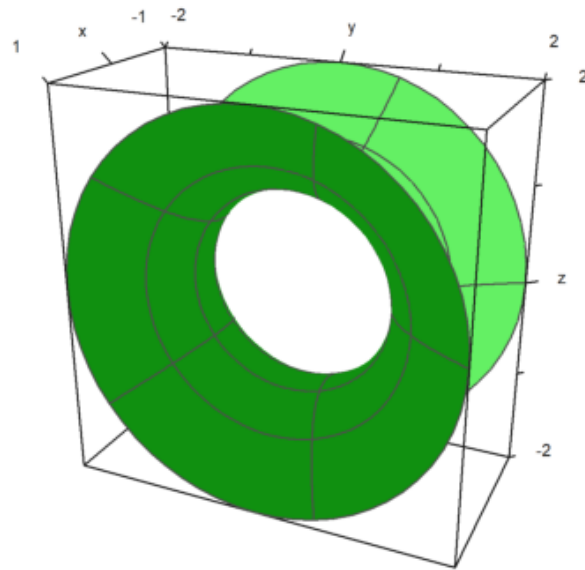


Parameter rotate memutar fungsi dalam sumbu x sekitar sumbu x.

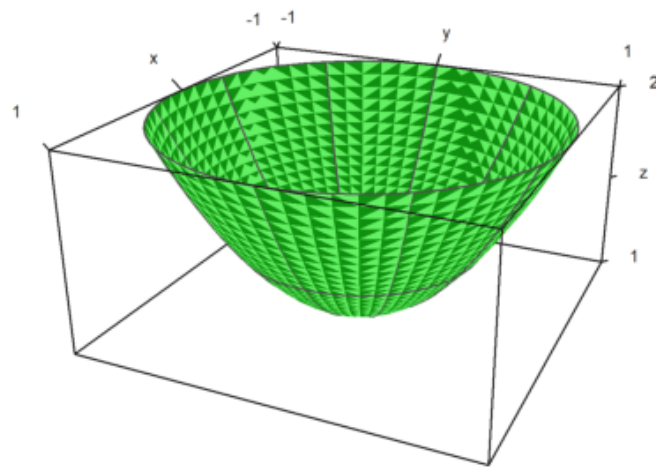
- rotate=1: Menggunakan sumbu x

- rotate=2: Menggunakan sumbu z

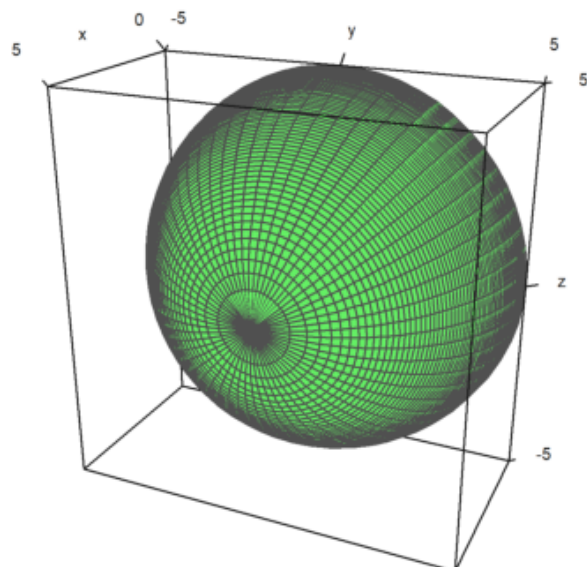
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



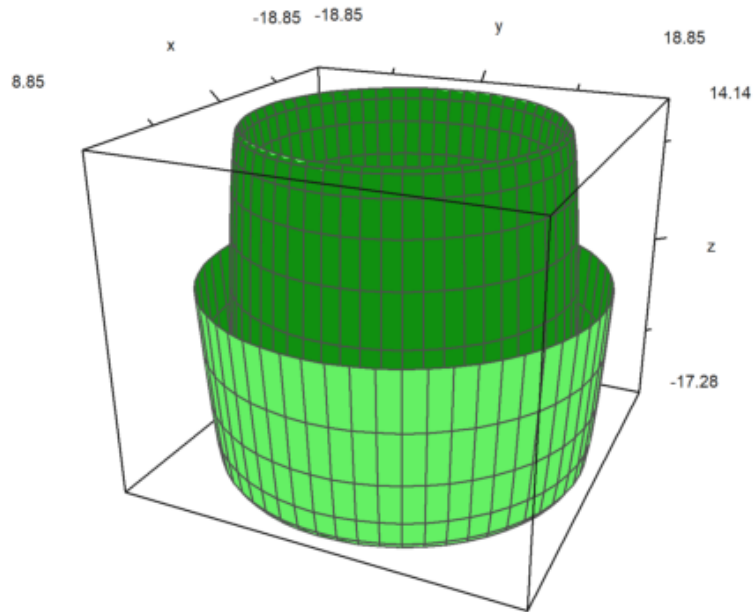
```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```



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

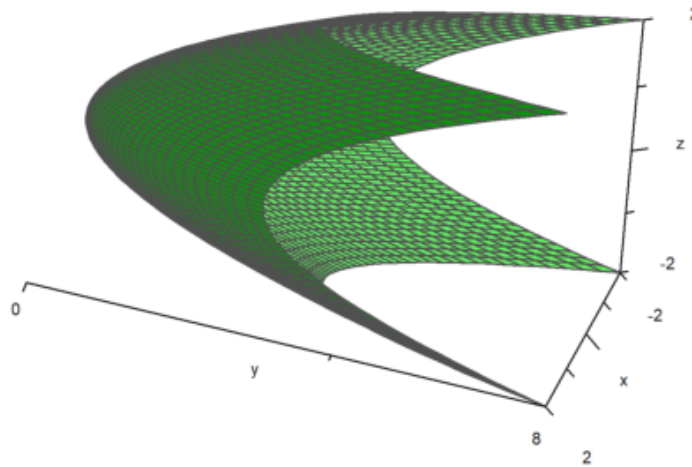


```
>plot3d("x*sin(x)",a=0,b=6pi,rotate=2):
```



Ini adalah sebuah plot dengan tiga fungsi.

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



Plot Kontur

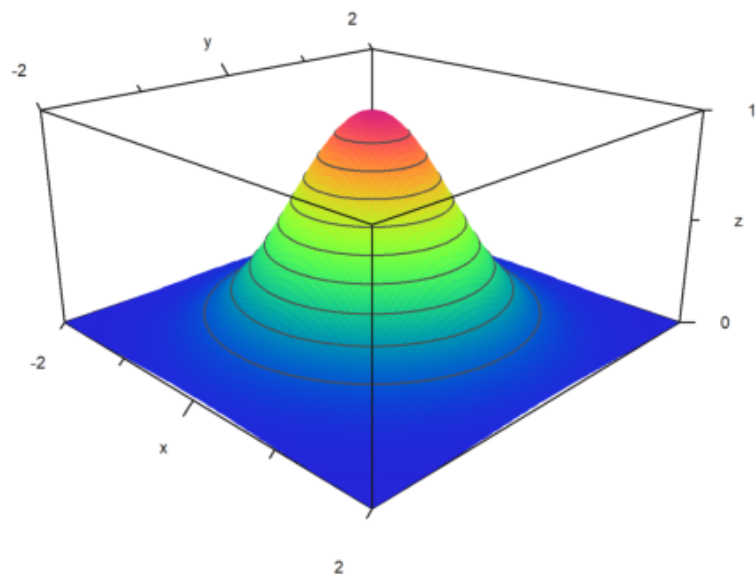
Untuk plot ini, Euler menambahkan garis-garis kisi. Sebagai gantinya, kita bisa menggunakan garis-garis tingkat dan satu warna atau spektrum warna. Euler dapat menggambar tinggi fungsi pada plot dengan shading. Dalam semua plot 3D, Euler dapat menghasilkan anaglif merah/cyan.

- >hue: Mengaktifkan shading ringan alih-alih kawat.
- >contour: Menampilkan garis kontur otomatis pada plot.
- level=... (or levels): Sebuah vektor nilai untuk garis kontur.

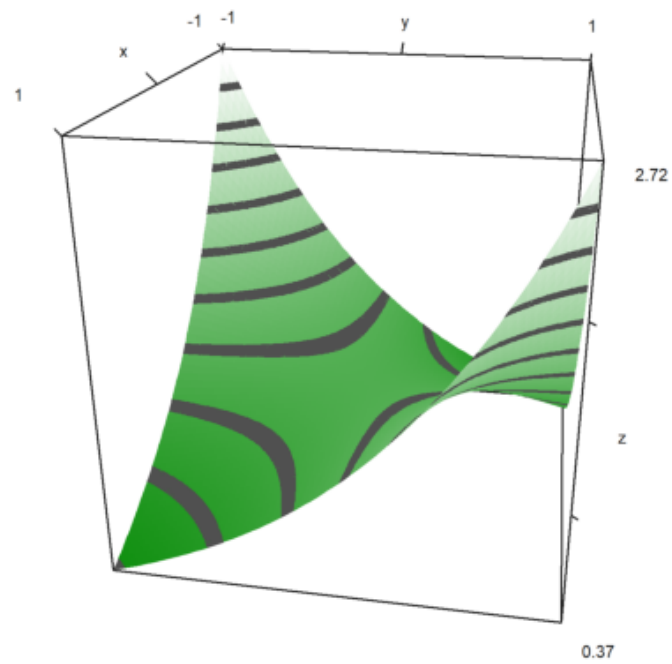
Nilai defaultnya adalah level="auto", yang menghitung beberapa garis kontur secara otomatis. Seperti yang Anda lihat pada plot, tingkat-tingkat tersebut sebenarnya adalah rentang tingkat.

Gaya default dapat diubah. Untuk plot kontur berikutnya, kita menggunakan kisi yang lebih halus dengan 100x100 titik, memperbesar fungsi dan plot, dan mengubah 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):
```



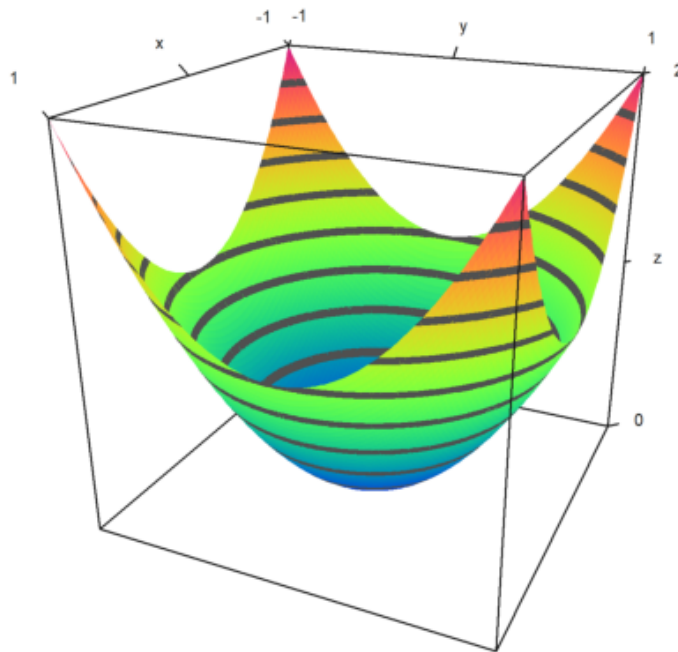
Pengaturan awal menggunakan warna abu-abu. Namun, berbagai pilihan warna spektrum juga tersedia.

- >spectral: Menggunakan skema spektral bawaan

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

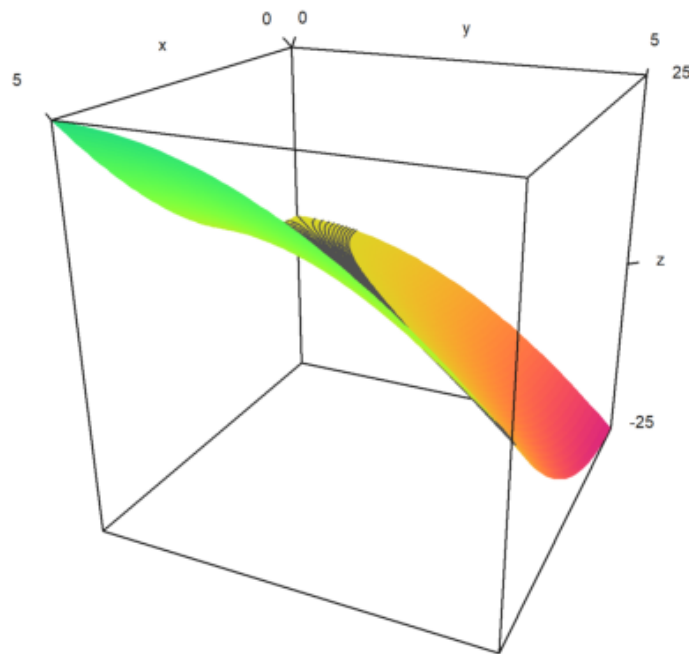
Pada plot berikut, kita menggunakan skema spektral bawaan dan meningkatkan jumlah titik untuk mendapatkan tampilan yang sangat halus.

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



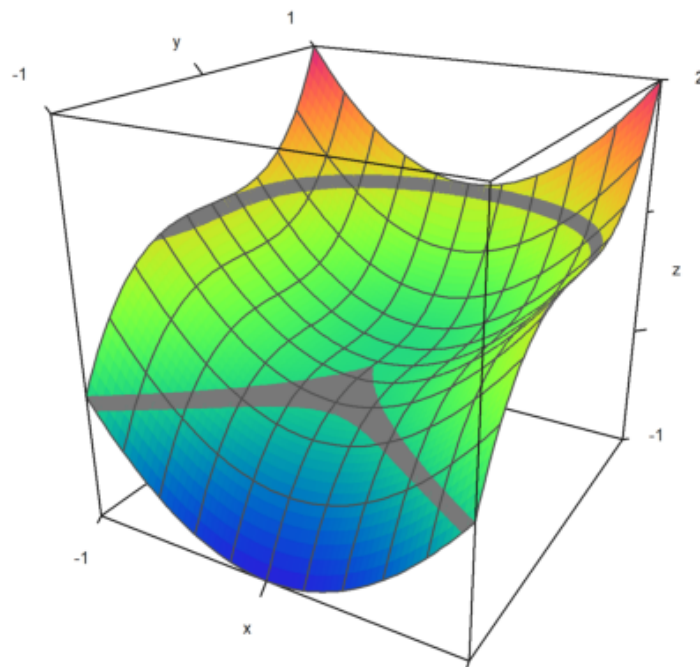
Daripada garis level otomatis, kita juga dapat mengatur nilai-nilai garis level. Ini akan menghasilkan garis level yang tipis daripada rentang level.

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



Dalam plot berikut, kami menggunakan dua rentang level yang sangat luas mulai dari -0,1 hingga 1, dan dari 0,9 hingga 1. Ini dimasukkan sebagai matriks dengan batas level sebagai kolom. Selain itu, kami melapisi grid dengan 10 interval di setiap arah.

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

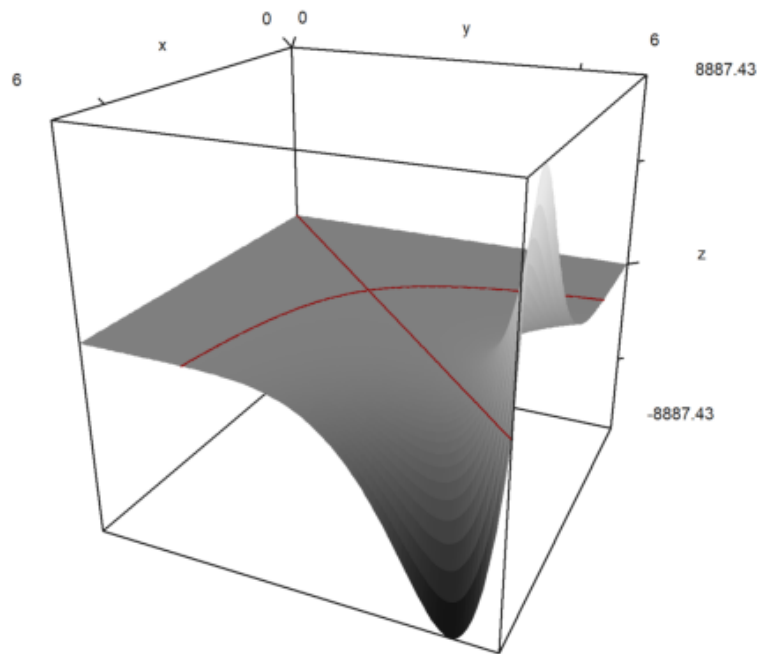


Dalam contoh berikut, kami menggambar himpunan, di mana

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

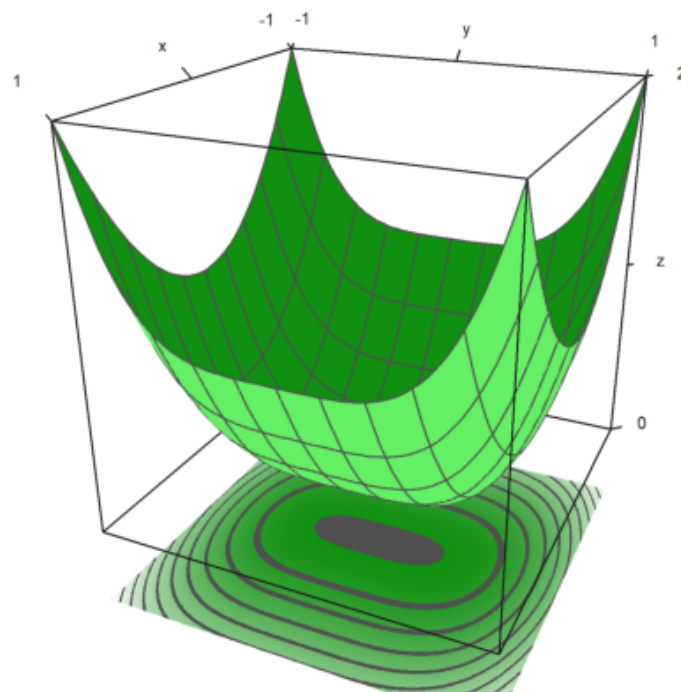
Kami menggunakan satu garis tipis untuk garis tingkat.

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

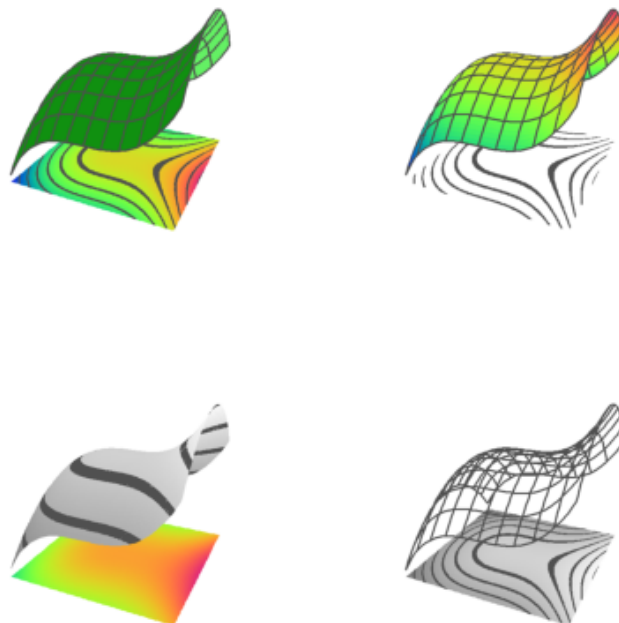
Ini adalah mungkin untuk menampilkan sebuah bidang kontur di bawah plot. Warna dan jarak dari plot dapat ditentukan.

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



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

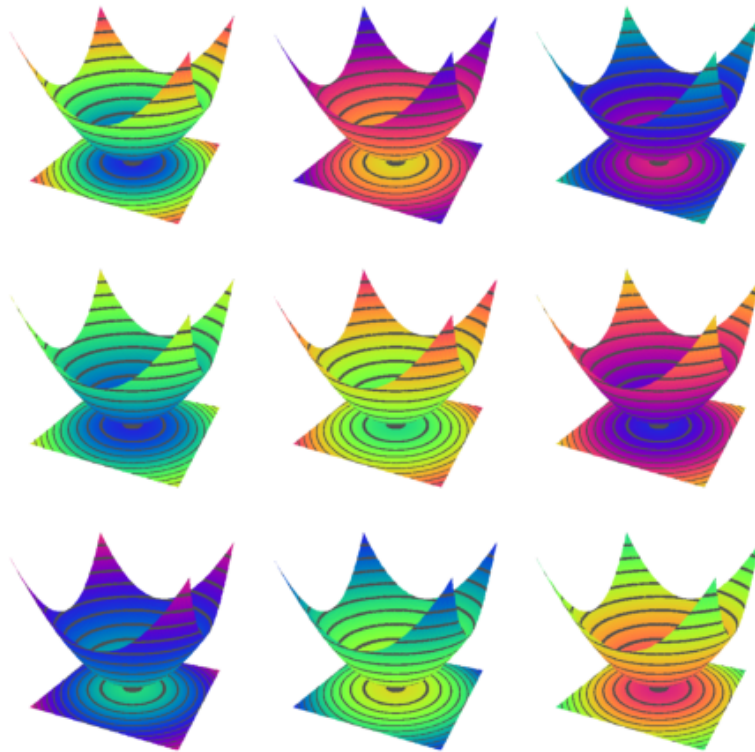
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1); ...
> plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2); ...
> plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3); ...
> plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4); ...
> plot3d(expr,<frame,>hue,grid=10,>transparent,>cp,cpcolor=gray);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



Ada beberapa skema spektral lainnya, diberi nomor dari 1 hingga 9. Tetapi Anda juga dapat menggunakan warna=nilai, di mana nilai

- spectral: untuk rentang dari biru hingga merah
- white: untuk rentang yang lebih lemah
- kuningbiru, unguhijau, birukuning, hijaumerah
- birukuning, hijaupurple, kuningbiru, merahhijau

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



Sumber cahaya dapat diubah dengan tombol "I" dan kunci kuror selama interaksi pengguna. Ini juga dapat diatur dengan parameter.

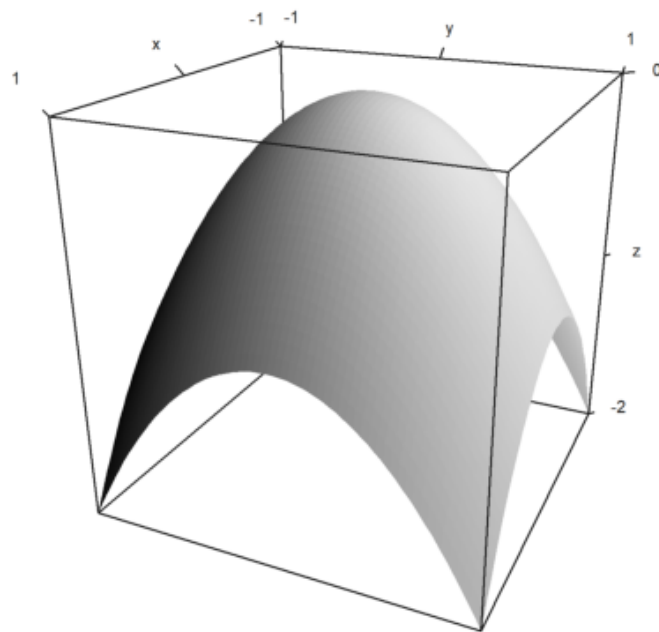
light: arah cahaya

amb: cahaya ambien antara 0 dan 1

Perlu diperhatikan bahwa program tidak membedakan antara sisi plot. Tidak ada bayangan. Untuk ini, 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)":
```

Press I and cursor keys (return to exit)



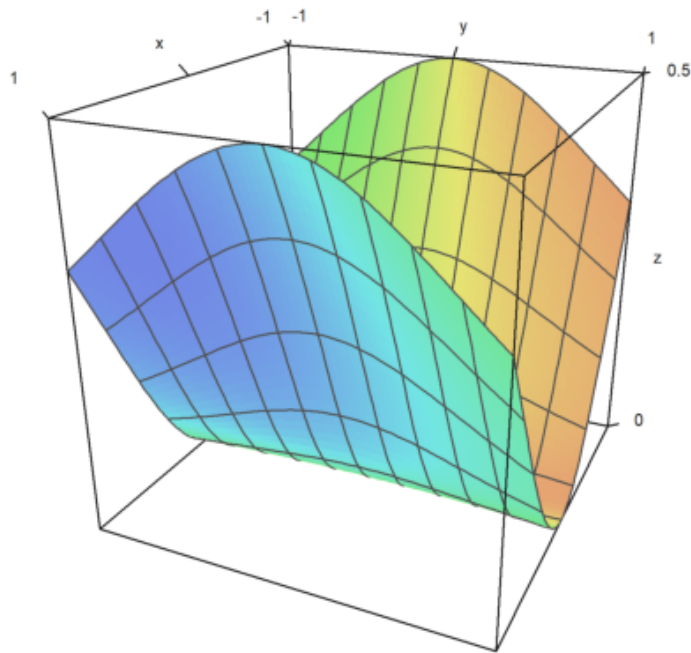
Parameter warna mengubah warna permukaan. Warna garis level juga dapat diubah.

```
>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):
```



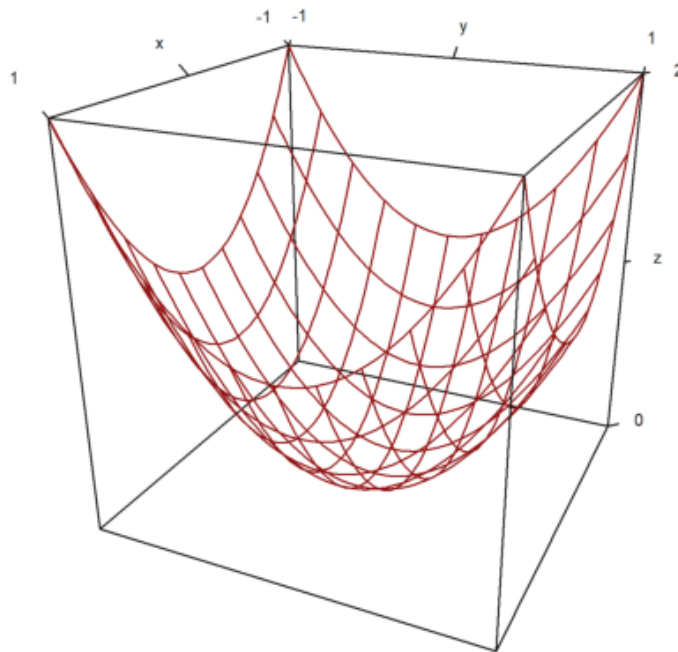
Warna 0 memberikan efek pelangi yang istimewa.

```
>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

Terdapat juga plot implisit dalam tiga dimensi. Euler menghasilkan potongan melalui objek-objek tersebut. Fitur-fitur dari plot3d mencakup plot implisit. Plot ini menampilkan himpunan nol dari suatu fungsi dalam tiga variabel.

Solusi dari

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

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

implicit=1: potongan sejajar dengan bidang y-z

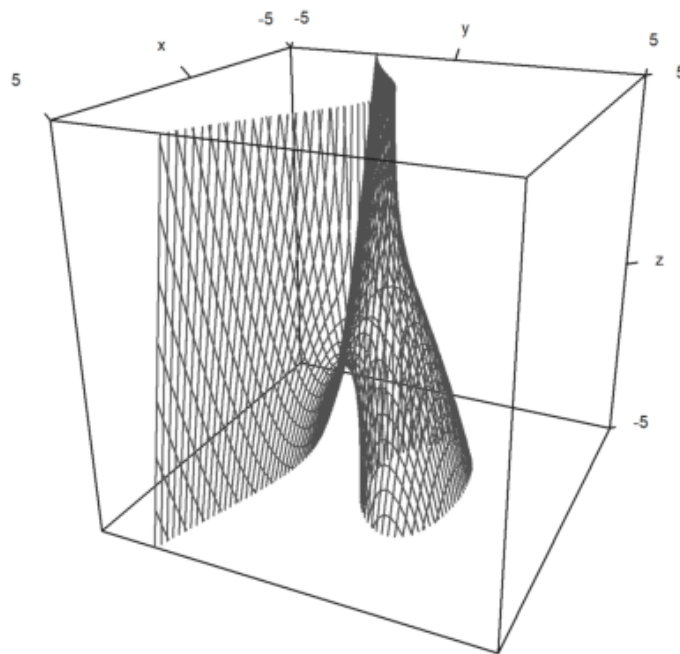
implicit=2: potongan sejajar dengan bidang x-z

implicit=4: potongan sejajar dengan bidang x-y

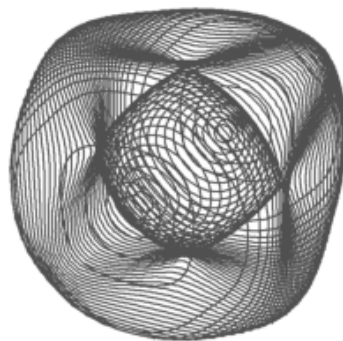
Tambahkan nilai-nilai ini, jika Anda ingin. Dalam contoh ini, kami memplot

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

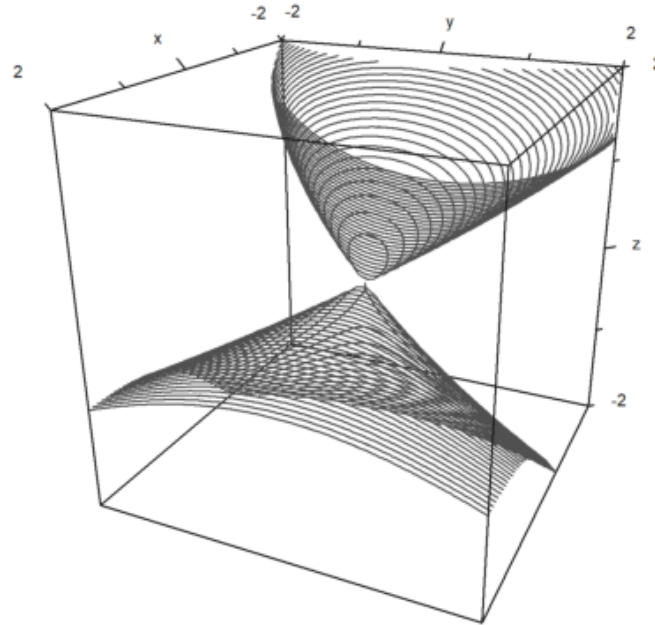
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```



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



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



Plotting 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 $f_x(x, y)$, $f_y(x, y)$, $f_z(x, y)$.

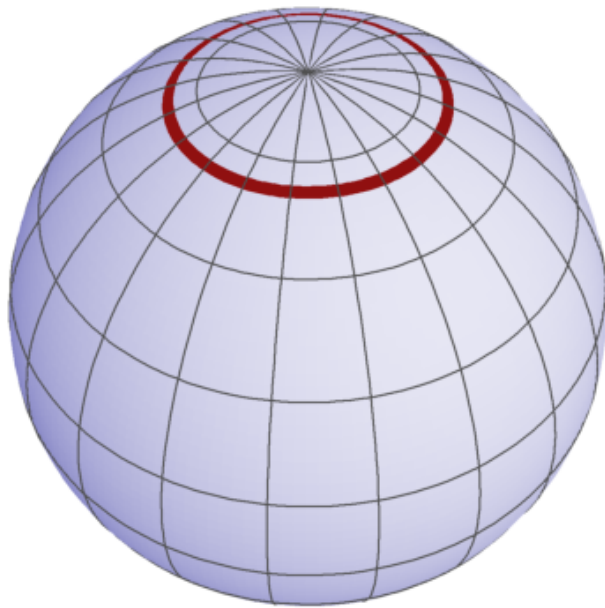
$$\gamma(t, s) = (x(t, s), y(t, s), z(t, s))$$

Karena x, y, z adalah matriks, kita mengasumsikan bahwa (t, s) berjalan melalui grid persegi. Sebagai hasilnya, Anda dapat membuat gambar-gambar persegi panjang dalam ruang.

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

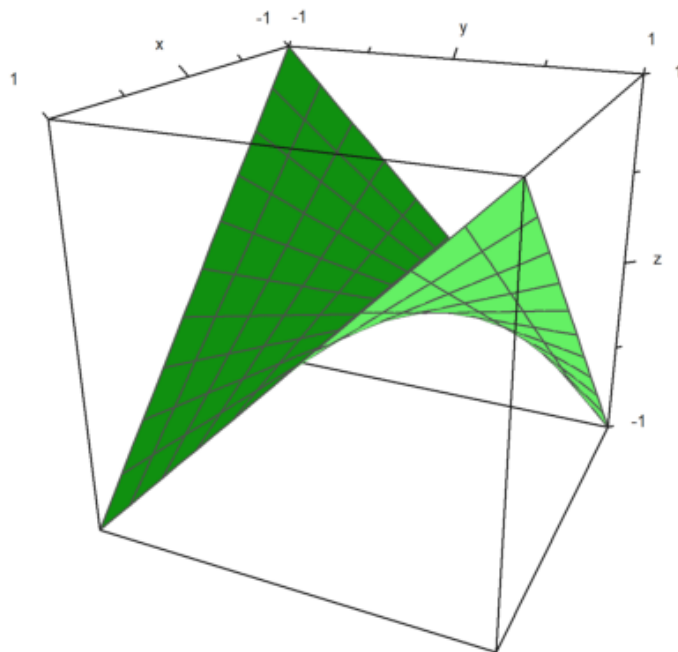
Dalam contoh berikut, kita menggunakan vektor nilai t dan vektor kolom nilai s untuk memparametrikkan permukaan bola. Dalam gambaran, kita dapat menandai wilayah-wilayah, dalam kasus kita, wilayah polar.

```
>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 merupakan grafik dari sebuah fungsi.

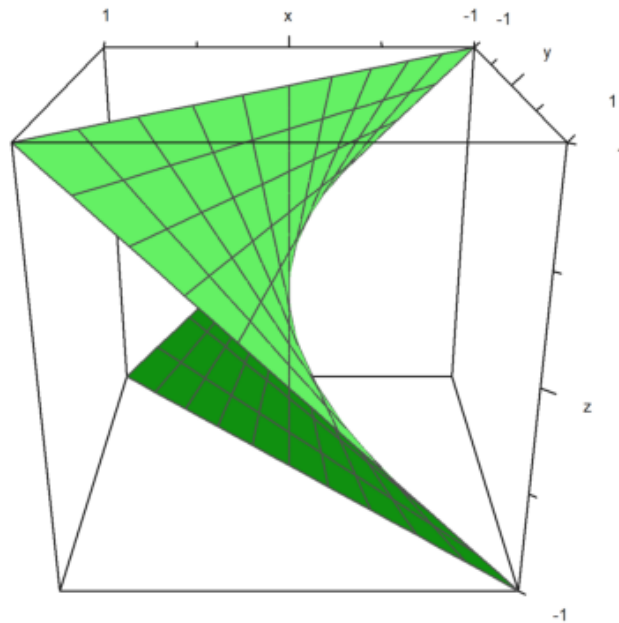
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



Namun, kita dapat membuat berbagai jenis permukaan. Berikut ini adalah permukaan yang sama sebagai fungsi

$$x = yz$$

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



Dengan lebih banyak usaha, kita dapat menghasilkan banyak permukaan.

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

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

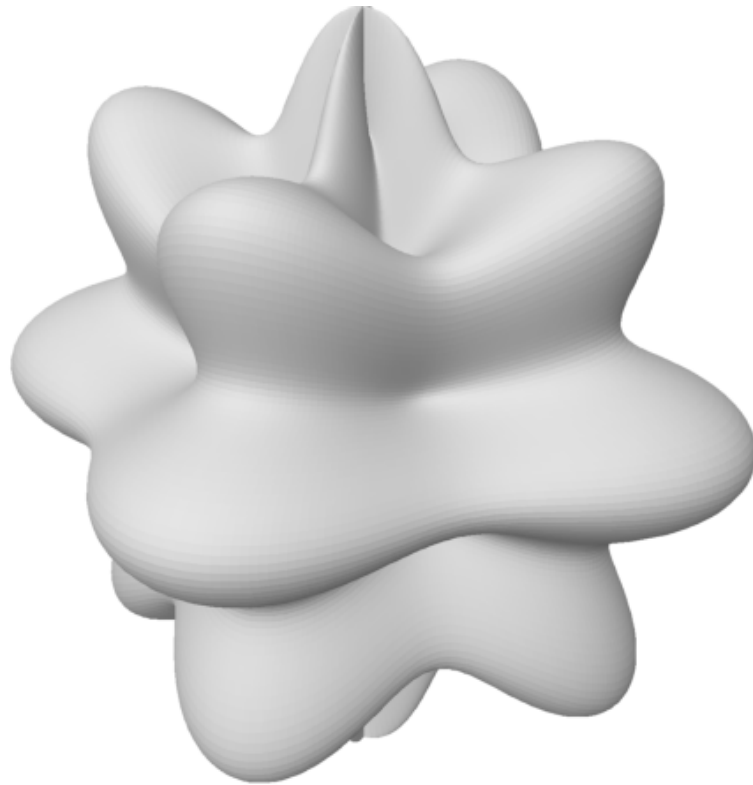
dengan

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

Kita mengdistorsi ini dengan faktor

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

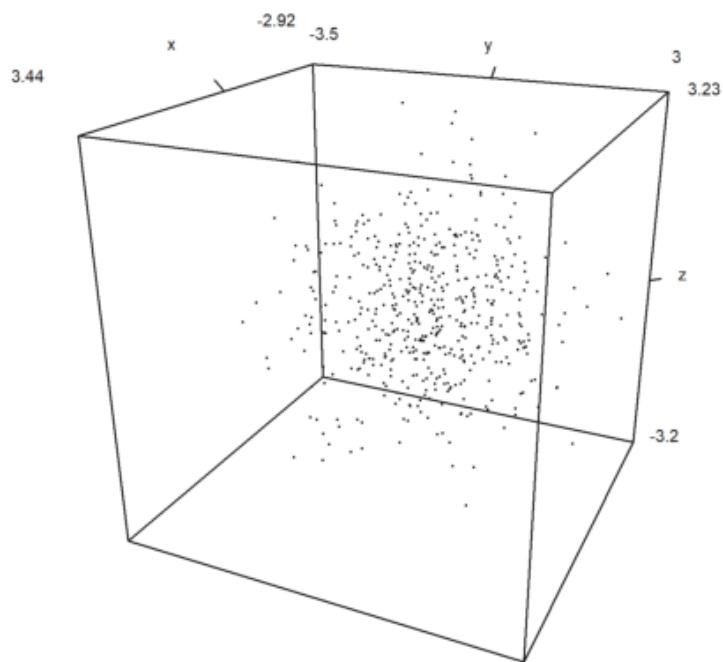
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, awan titik juga mungkin. Untuk menggambarkan data titik dalam ruang, kita memerlukan tiga vektor untuk koordinat titik-titik tersebut.

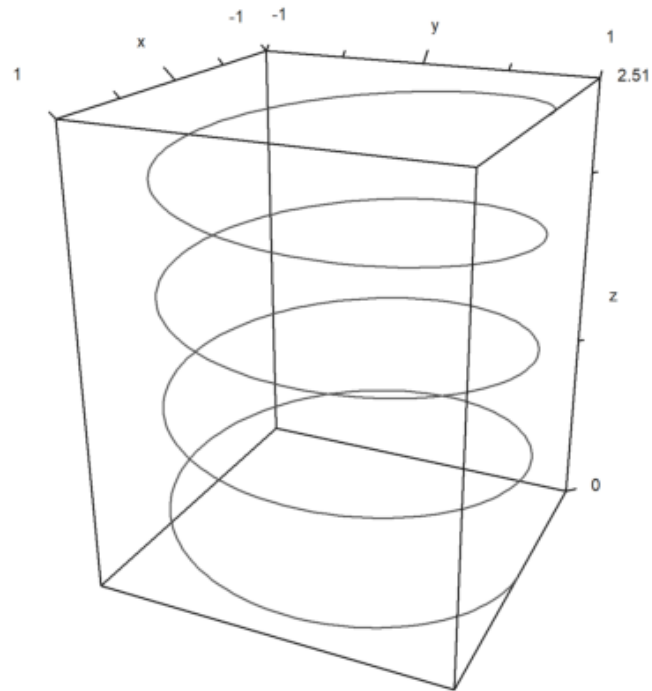
Gaya-gaya tersebut sama seperti dalam plot2d dengan `points=true`;

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

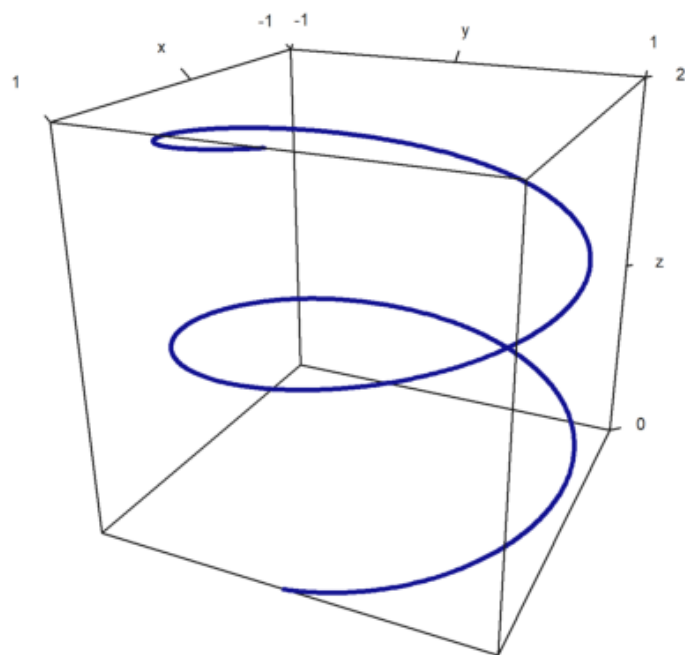


Ini juga memungkinkan untuk menggambar kurva dalam tiga dimensi (3D). Dalam hal ini, lebih mudah untuk menghitung sebelumnya titik-titik dari kurva tersebut. Untuk kurva-kurva dalam bidang, kita menggunakan urutan koordinat dan parameter `wire=true`.

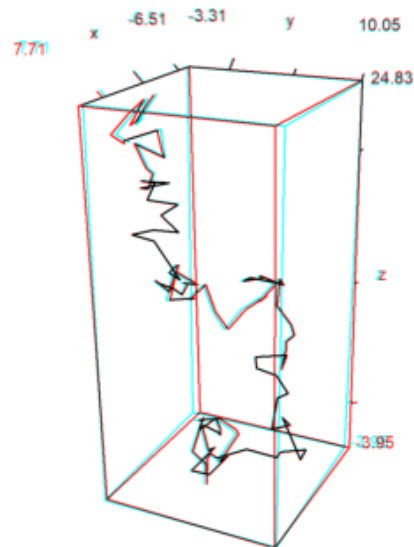
```
>t=linspace(0,8pi,500); ...  
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```



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

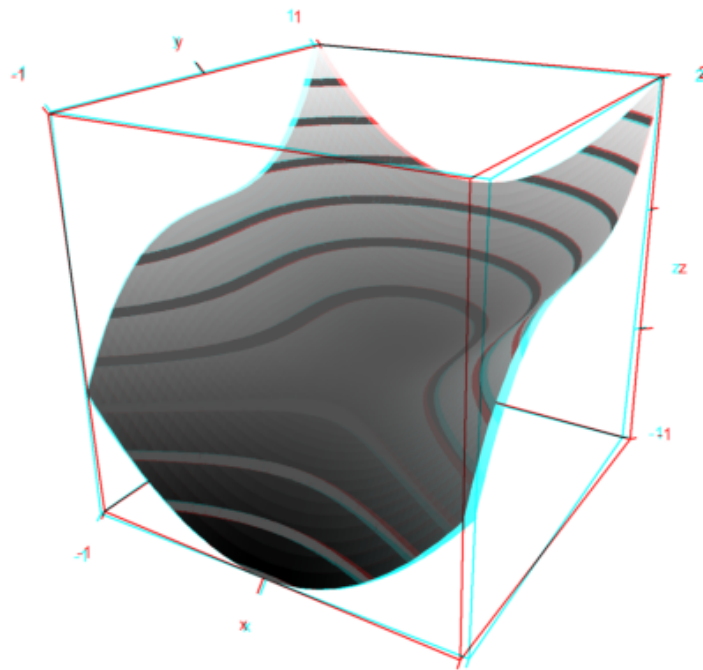


```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



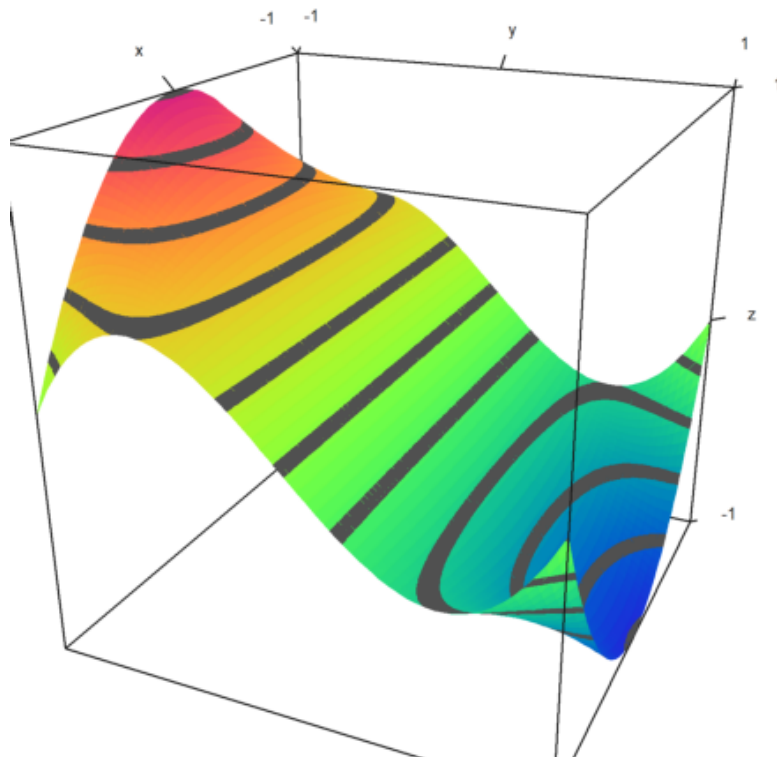
EMT juga dapat membuat plot dalam mode anaglyph. Untuk melihat plot tersebut, Anda memerlukan kaca-mata merah/biru.

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



Seringkali, skema warna spektral digunakan untuk grafik ini. Ini menekankan tinggi fungsi tersebut.

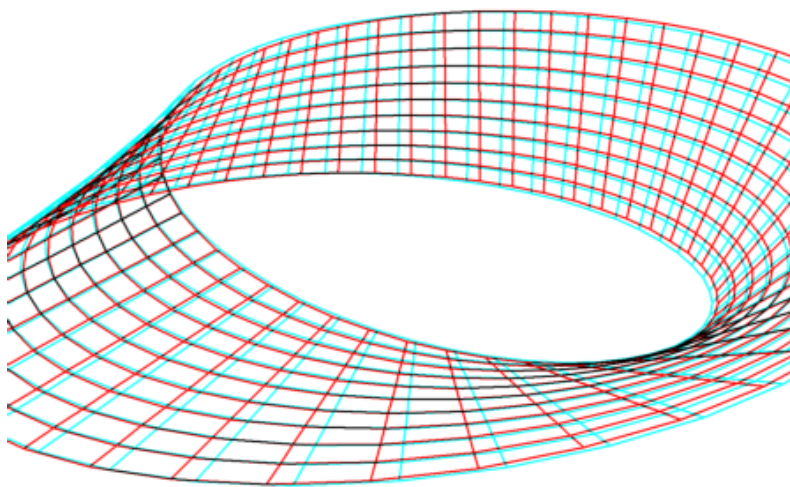
```
>plot3d("x^2*y^3-y",>spectral,>contour, zoom=3.2) :
```



Euler juga dapat menggambar permukaan-parameterkan ketika parameter-parameter tersebut adalah nilai-nilai x -, y -, dan z - dari gambar grid berbentuk persegi panjang di dalam ruang.

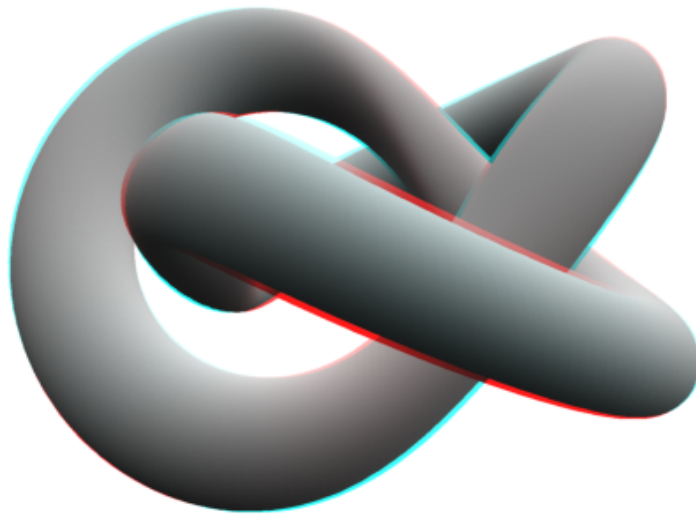
Untuk demonstrasi berikutnya, kami menyiapkan parameter u dan v , dan menghasilkan koordinat ruang dari kedua parameter tersebut.

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

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

Plot Statistik

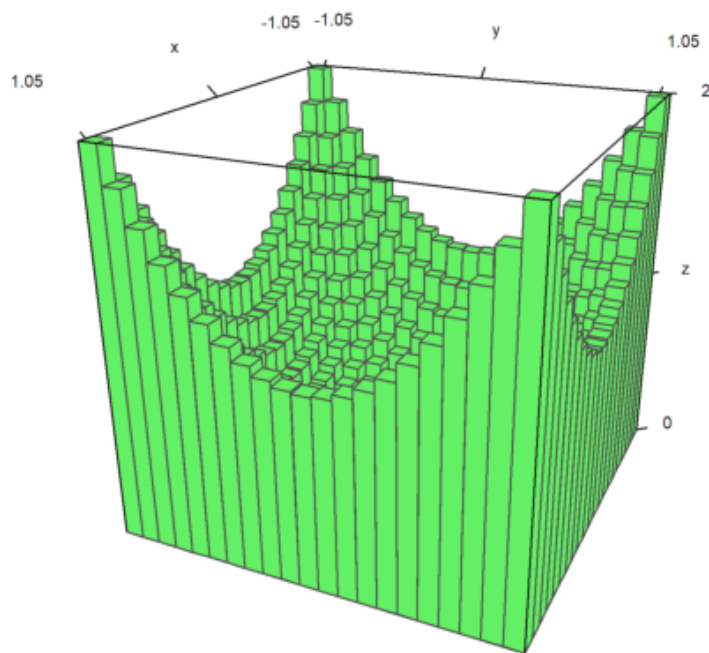
Grafik batang juga mungkin. Untuk ini, kita harus memberikan:

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

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

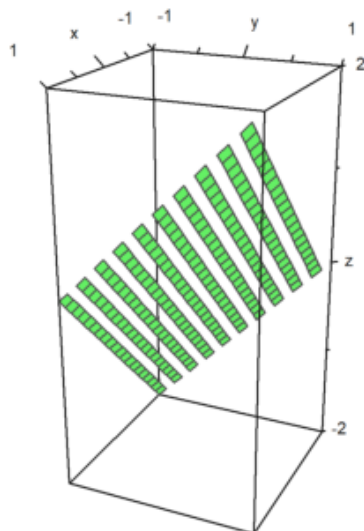
Dalam contoh ini, kita pertama-tama menghitung nilai-nilai. Kemudian kita menyesuaikan x dan y, sehingga vektor-vektor tersebut berpusat pada nilai yang digunakan.

```
>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):
```



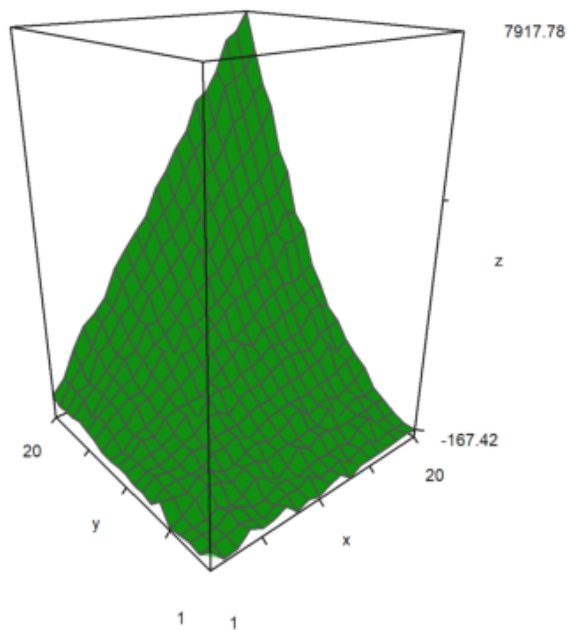
Mungkin untuk membagi plot permukaan menjadi dua atau lebih bagian.

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

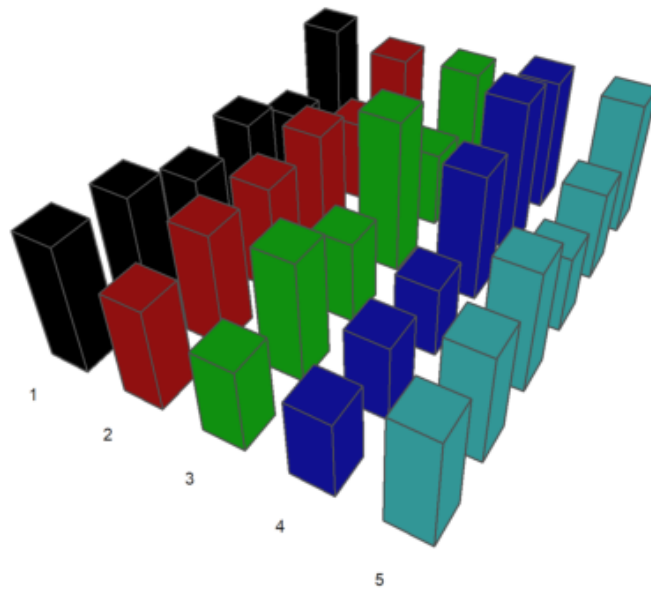


Jika Anda memuat atau menghasilkan matriks data M dari sebuah file dan perlu membuat plotnya dalam 3D, Anda dapat melakukan penskalaan pada matriks tersebut menjadi rentang [-1,1] dengan menggunakan perintah "scale(M)", atau melakukan penskalaan dengan menggunakan "zscale". Ini dapat dikombinasikan dengan faktor-faktor penskalaan individu yang diterapkan secara tambahan.

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

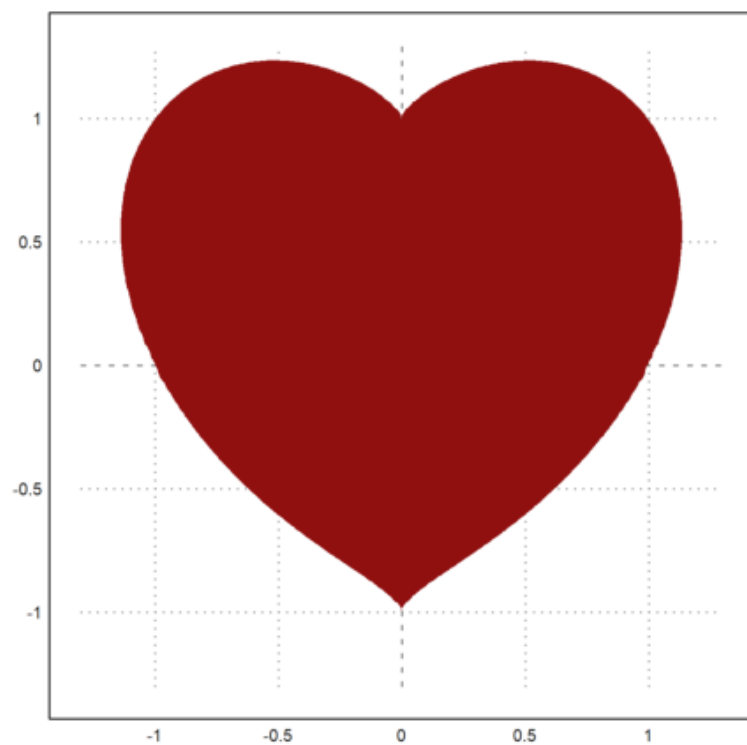


```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



Permukaan Benda Putar

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



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

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

Kami ingin memutar kurva hati sekitar sumbu y. Berikut adalah ekspresi yang mendefinisikan bentuk hati:

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

Selanjutnya kita menetapkan

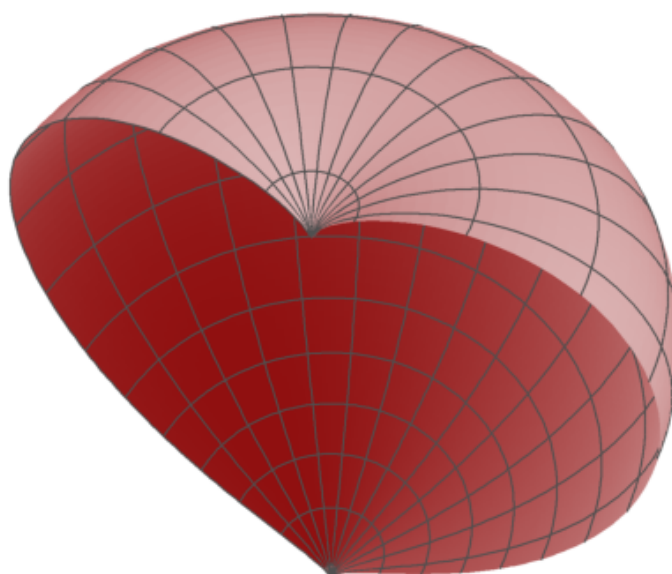
$$x = r \cdot \cos(a), \quad y = r \cdot \sin(a).$$

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

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

Ini memungkinkan untuk mendefinisikan fungsi numerik, yang memecahkan untuk r, jika a diberikan. Dengan fungsi itu, kita dapat menggambar hati yang berputar 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°):
```

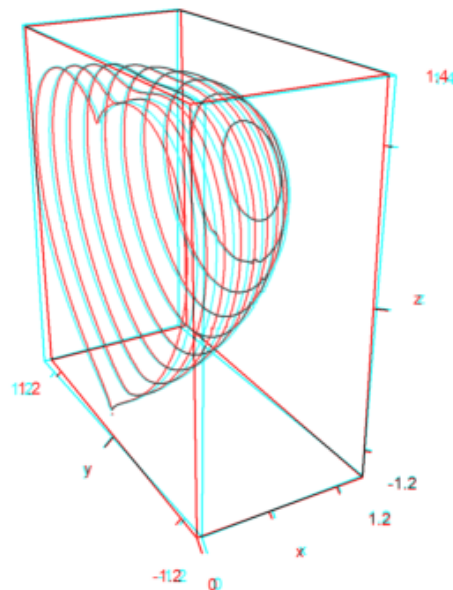


Berikut adalah plot 3D dari gambar di atas yang diputar sekitar sumbu z. Kami mendefinisikan fungsi yang menggambarkan objek tersebut.

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

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

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



Plot 3D Khusus

Fungsi plot3d bagus untuk dimiliki, tetapi tidak memenuhi semua kebutuhan. Selain rutinitas yang lebih dasar, mungkin Anda bisa mendapatkan plot bingkai dari objek apa pun yang Anda sukai.

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

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

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

```

h=holding(1);
plot3d(x,y, (x^2+y^2)/2,<scale,<frame,>contour,>hue);
holding(h);
endfunction

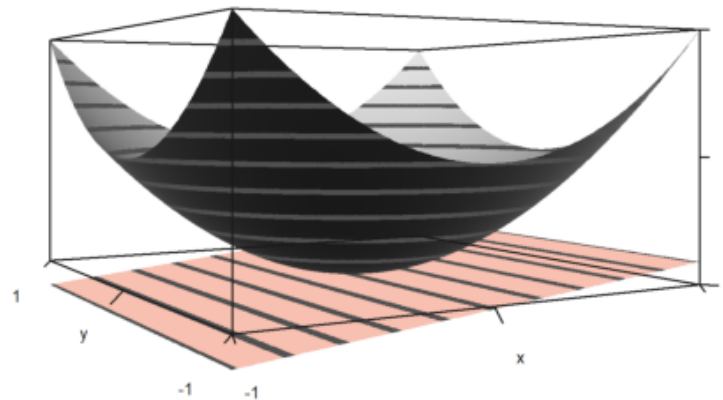
```

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

```

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

```



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

```

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

```

```

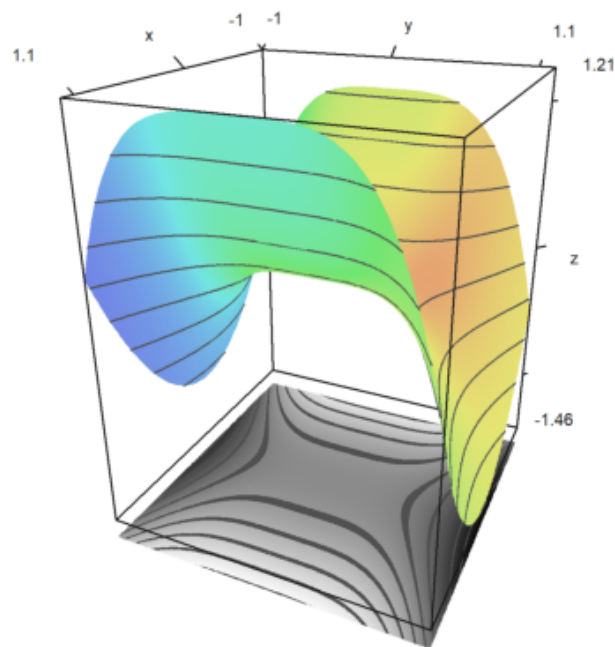
    zoom(2);
    wi=fullwindow();
    plotcontourplane(x,y,z,level="auto",<scale);
    plot3d(x,y,z,>hue,<scale,>add,color=white,level="thin");
    window(wi);
    reset();
endfunction

```

```

>myplot(x,y,z):

```



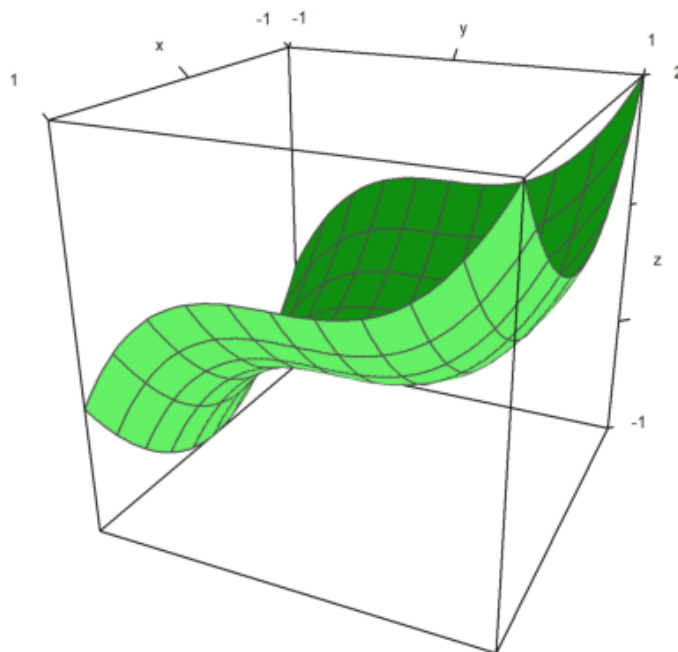
Animasi

Euler dapat menggunakan bingkai (frames) untuk pra-menghitung animasi.

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

Silakan pelajari sumber kode fungsi `rotate` untuk melihat lebih banyak detailnya.

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

Menggambar Povray

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

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

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

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

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

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

Perlu diingat bahwa alam semesta Povray memiliki sistem koordinat yang berbeda. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray. Jadi Anda dapat terus berpikir dalam sistem koordinat Euler dengan z mengarah secara vertikal ke atas, dan sumbu x,y,z sesuai dengan tangan kanan.

Anda perlu memuat berkas povray.

```
>load povray;
```

Pastikan direktori bin Povray ada dalam path. Jika tidak, edit variabel berikut agar berisi path ke eksekusi povray.

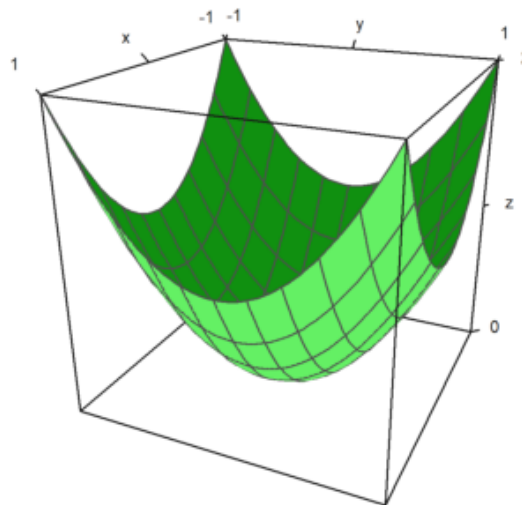
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

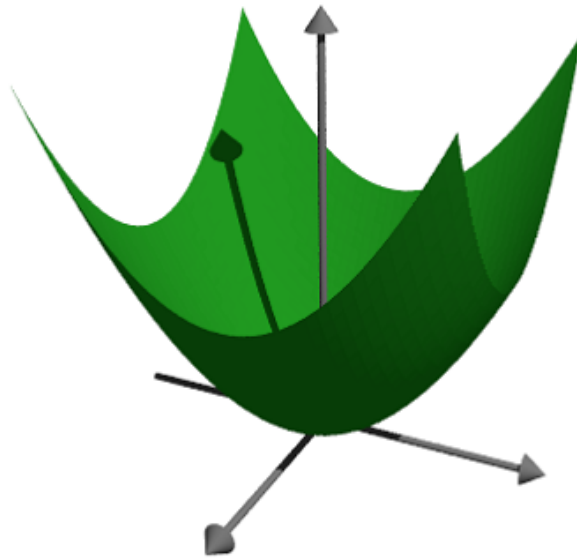
Untuk kesan pertama, kami menggambar fungsi sederhana. Perintah berikut menghasilkan file povray di direktori pengguna Anda, dan menjalankan Povray untuk pelacakan sinar file ini.

Jika Anda memulai perintah berikut, GUI Povray seharusnya terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan ditanyai apakah Anda ingin mengizinkan file exe ini untuk berjalan. 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) :
```

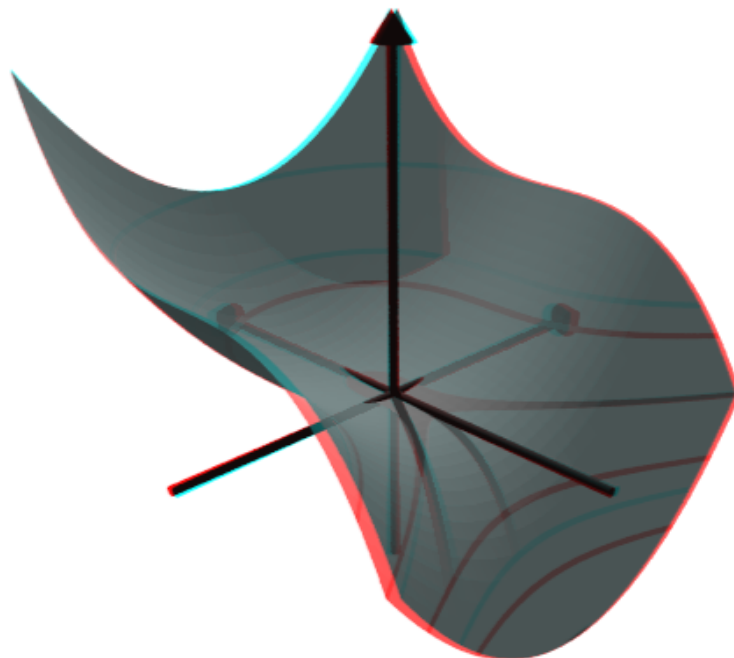


```
>pov3d("x^2+y^2",zoom=3) ;
```



Kita dapat membuat fungsi tersebut transparan dan menambahkan yang lainnya. Kita juga dapat menambahkan garis level pada plot fungsi.

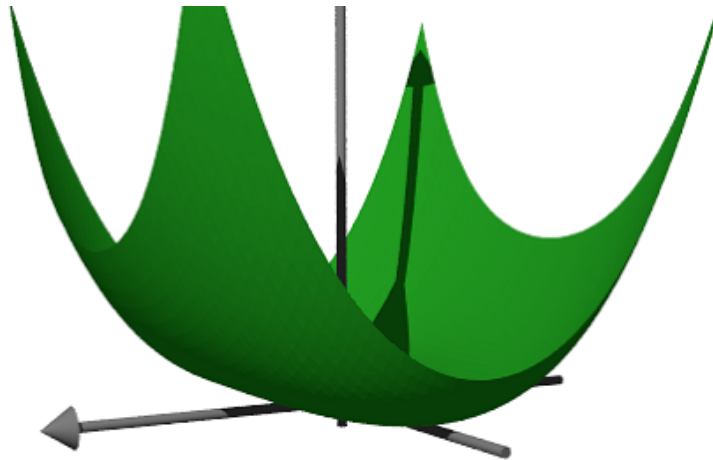
```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
> look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```



Kadang-kadang perlu untuk mencegah penskalaan fungsi, dan penskalaan fungsi secara manual.

Kita menggambar himpunan titik-titik dalam bidang kompleks, di mana hasil kali jarak ke 1 dan -1 sama dengan 1.

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



Plotting dengan Koordinat

Daripada menggunakan fungsi, kita dapat melakukan plotting dengan koordinat. Seperti pada plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek tersebut.

Pada contoh ini, kita memutar sebuah fungsi sekitar sumbu z.

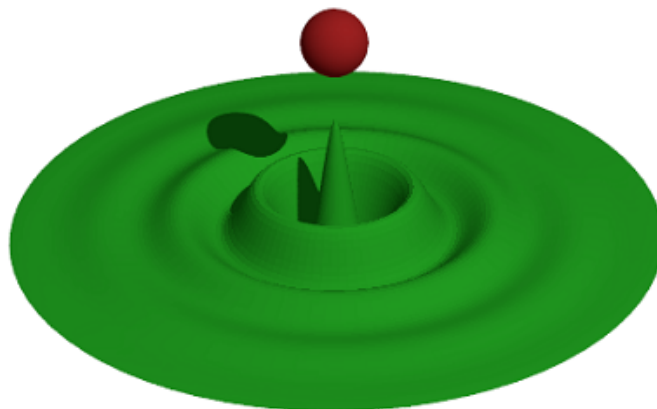
```
>function f(x) := x^3-x+1; ...  
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...  
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...  
>pov3d(X,Y,Z,angle=40°,look=povlook(green,0.1),height=20°,axis=0,zoom=4,light=[10,-5,5]);
```



Dalam contoh berikut, kita menggambar gelombang teredam. Kami menghasilkan gelombang tersebut dengan bahasa matriks Euler.

Kami juga menunjukkan bagaimana objek tambahan dapat ditambahkan ke dalam adegan pov3d. Untuk pembuatan objek, lihat contoh-contoh berikut. Perhatikan bahwa plot3d akan menyesuaikan skala plot sehingga cocok dalam kubus satuan.

```
>r=linspace(0,1,80); phi=linspace(0,2pi,80)'; ...
>x=r*cos(phi); y=r*sin(phi); z=exp(-5*r)*cos(8*pi*r)/3; ...
>pov3d(x,y,z,zoom=5,axis=0,height=30°,add=povsphere([0,0,0.5],0.1,povlook(red)), ...
> w=500,h=300);
```



Dengan metode shading canggih dari Povray, sangat sedikit titik dapat menghasilkan permukaan yang sangat halus. Hanya di batas-batas dan dalam bayangan trik ini 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 dari permukaannya adalah $[x, y, Z]$. Kami menghitung dua turunan terhadap x dan y dari ini dan mengambil hasil perkalian silang sebagai normalnya.

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

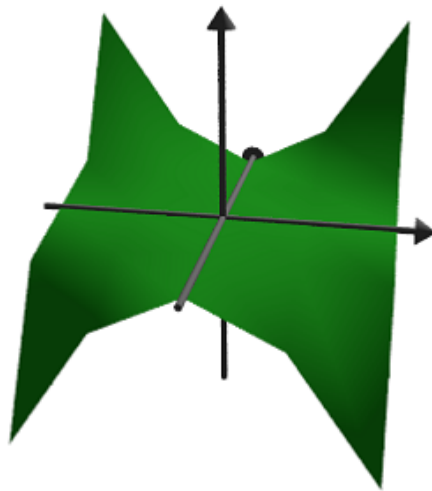
Kami mendefinisikan normal sebagai hasil perkalian silang dari turunan-turunan ini, dan mendefinisikan fungsi koordinat.

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

$$[-2xy^3, -3x^2y^2, 1]$$

Kami hanya menggunakan 25 poin.

```
>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 simpul Trefoil yang dibuat oleh A. Busser dalam Povray. Terdapat versi yang diperbaiki dari ini dalam contoh-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 vektor normal untuk kami. Pertama, tiga fungsi untuk koordinat sebagai ekspresi simbolik.

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

Kemudian turunan dua vektor terhadap x dan y.

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

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

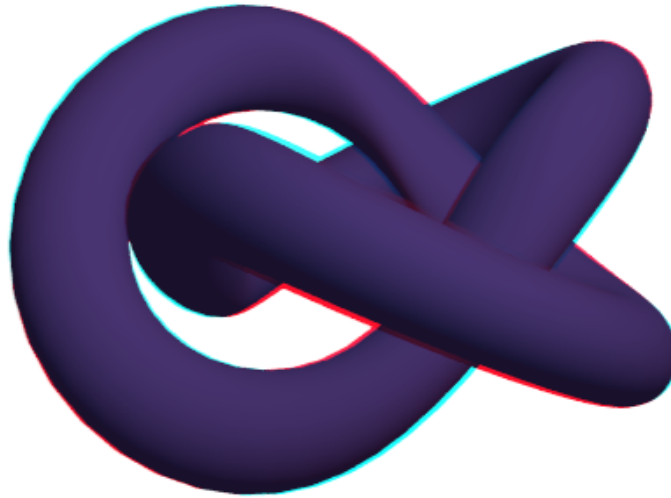
```
>dn &= crossproduct(dx,dy);
```

Sekarang kita mengevaluasi semua ini secara numerik.

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

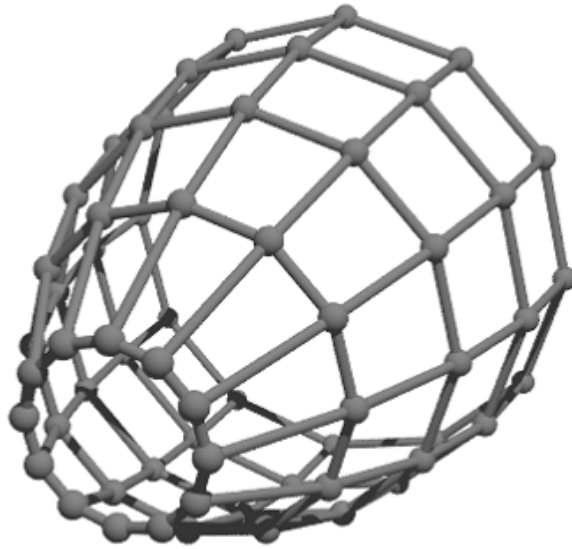
Vektor normal adalah hasil evaluasi dari ekspresi simbolis $dn[i]$ untuk $i=1,2,3$. Syntax untuk ini adalah `&"expression"(parameter)`. Ini merupakan alternatif dari metode pada contoh sebelumnya, di mana kita mendefinisikan ekspresi simbolis NX, NY, NZ terlebih dahulu.

```
>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));
```



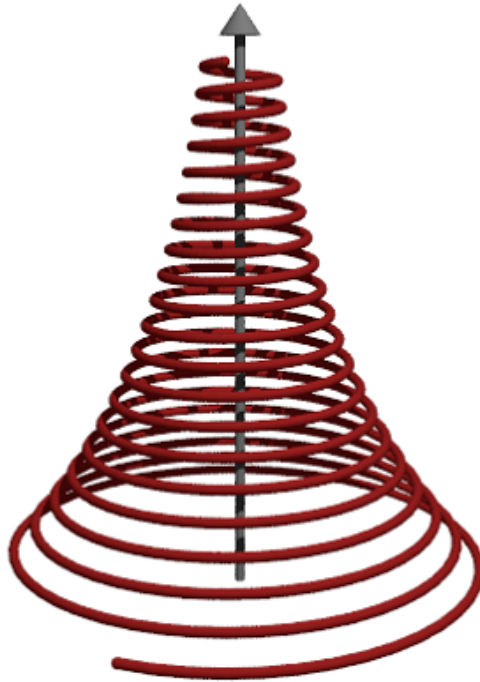
Kita juga dapat membuat grid dalam 3D.

```
>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-kurva menjadi mungkin.

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

Kami memulai output dengan povstart().

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

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

Fungsi-fungsi seperti povx() hanya mengembalikan vektor [1,0,0], yang dapat digunakan sebagai penggantinya.

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

Kalimat-kalimat tersebut berisi kode Povray, yang pada saat itu tidak perlu kita pahami.

```
>c2
```

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

Seperti yang Anda lihat, kami menambahkan tekstur pada objek dalam tiga warna yang berbeda.

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

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

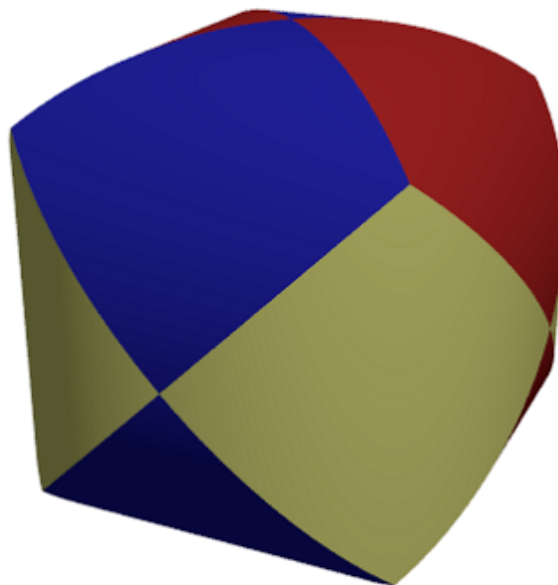
```
texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> } }  
finish { ambient 0.5 }
```

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

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

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

```
>povend;
```



Berikut ini adalah fungsi-fungsi yang menghasilkan fraktal secara rekursif.

Fungsi pertama menunjukkan bagaimana Euler mengatasi objek Povray sederhana. Fungsi `povbox()` mengembalikan sebuah string yang berisi koordinat kotak, tekstur, dan penyelesaian.

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

```

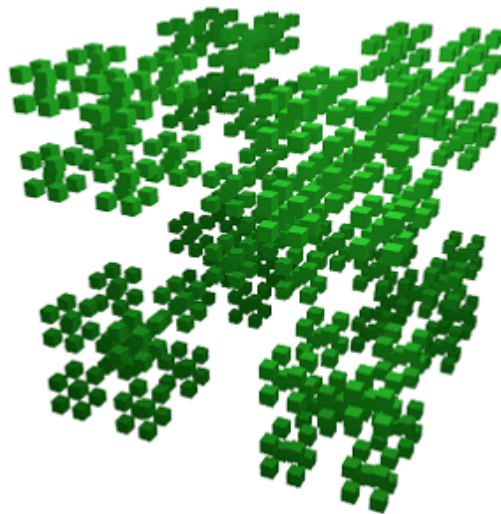
if n==1 then writeln(onebox(x,y,z,h));
else
  h=h/3;
  fractal(x,y,z,h,n-1);
  fractal(x+2*h,y,z,h,n-1);
  fractal(x,y+2*h,z,h,n-1);
  fractal(x,y,z+2*h,h,n-1);
  fractal(x+2*h,y+2*h,z,h,n-1);
  fractal(x+2*h,y,z+2*h,h,n-1);
  fractal(x,y+2*h,z+2*h,h,n-1);
  fractal(x+2*h,y+2*h,z+2*h,h,n-1);
  fractal(x+h,y+h,z+h,h,n-1);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

```

```

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

```



Perbedaan memungkinkan pemisahan satu objek dari yang lain. Seperti perpotongan, itu merupakan bagian dari objek CSG dalam Povray.

```

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

```

Untuk demonstrasi ini, kita mendefinisikan sebuah objek dalam Povray, daripada menggunakan sebuah string dalam Euler. Definisi tersebut langsung ditulis ke dalam file.

Koordinat sebuah kotak dengan nilai -1 hanya berarti [-1,-1,-1].

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

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

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

Kami menghasilkan sebuah kubus kedua, lalu memutarnya dan membesarkannya sedikit.

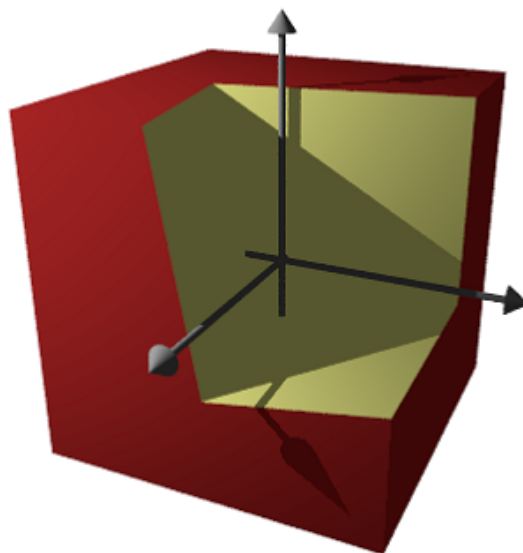
```
>c2=povobject ("mycube",povlook (yellow),translate=[1,1,1], ...  
> rotate=xrotate (10°)+yrotate (10°), scale=1.2);
```

Kemudian kita mengambil perbedaan dari kedua objek tersebut.

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

Sekarang tambahkan tiga sumbu.

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



Povray dapat menggambar himpunan di mana $f(x, y, z) = 0$, seperti parameter implisit dalam plot3d. Hasilnya terlihat jauh lebih baik, namun sintaks untuk fungsi ini agak berbeda. Anda tidak dapat menggunakan keluaran dari Maxima atau ekspresi 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);
>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)-1,2)) (pow(pow(x,2)+pow(z,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);
```

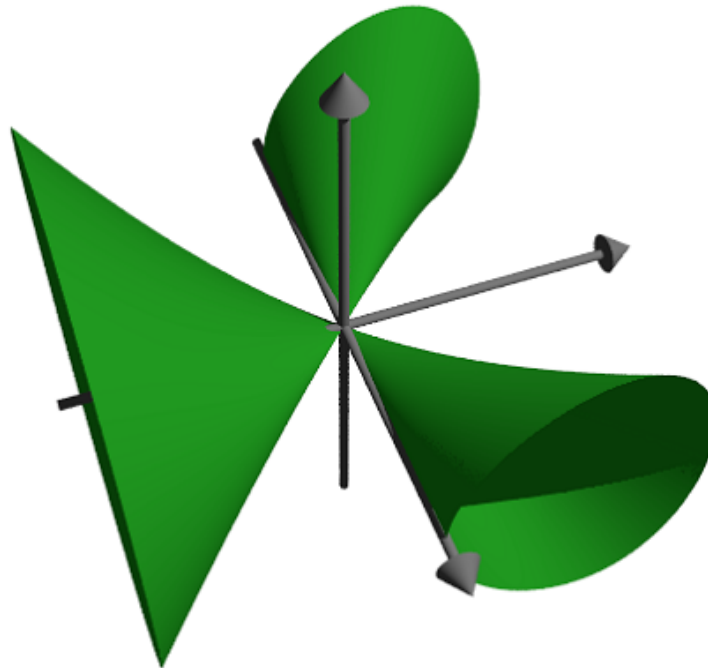
```
>povstart (angle=25°,height=10°);
>writeln(povsurface("pow(x,2)+pow(y,2)*pow(z,2)-1",povlook(blue),povbox(-2,2,"")));
>povend();
```



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

Buatlah permukaan implisit. Perhatikan sintaks yang berbeda dalam ekspresi ini.

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



Objek Jaringan

Dalam contoh ini, kami akan menunjukkan bagaimana cara membuat objek jala, dan menggambarinya dengan informasi tambahan.

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

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

Kita tidak dapat menyimpan objek dalam bentuk string seperti sebelumnya, karena terlalu besar. Jadi, kita mendefinisikan objek dalam sebuah file Povray menggunakan declare. Fungsi povtriangle() melakukan ini secara otomatis. Ini 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 akan mendefinisikan dua cakram, yang akan dipotong dengan permukaan.

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

Ketik permukaannya dikurangi dua cakram.

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

Tulis dua perpotongan.

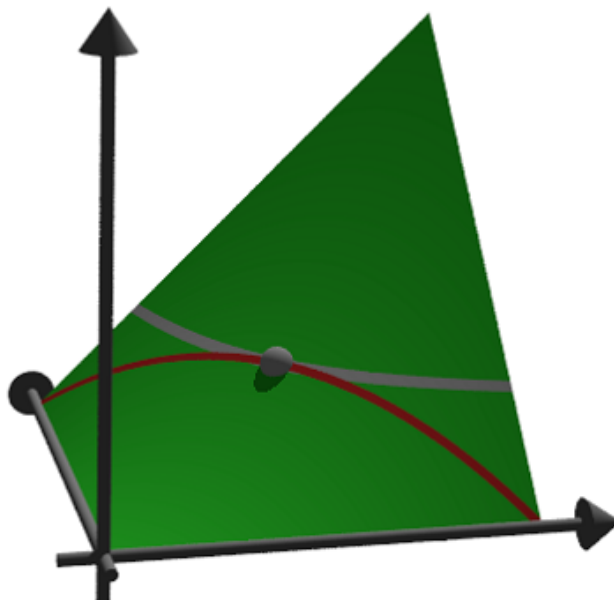
```
>writeln(povintersection([mesh,cl],povlook(red))); ...  
>writeln(povintersection([mesh,ll],povlook(gray)));
```

Tulis sebuah 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();
```



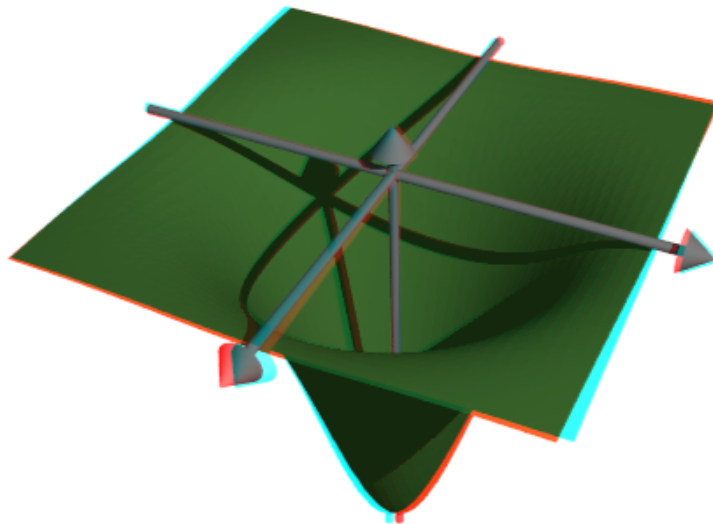
Anaglif dalam Povray

Untuk menghasilkan anaglif untuk kacamata merah/cyan, Povray harus dijalankan dua kali dari posisi kamera yang berbeda. Ini menghasilkan dua file Povray dan dua file PNG, yang dimuat dengan fungsi load-anaglyph().

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

Fungsi pov3d() memiliki sakelar sederhana untuk menghasilkan anaglif.

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



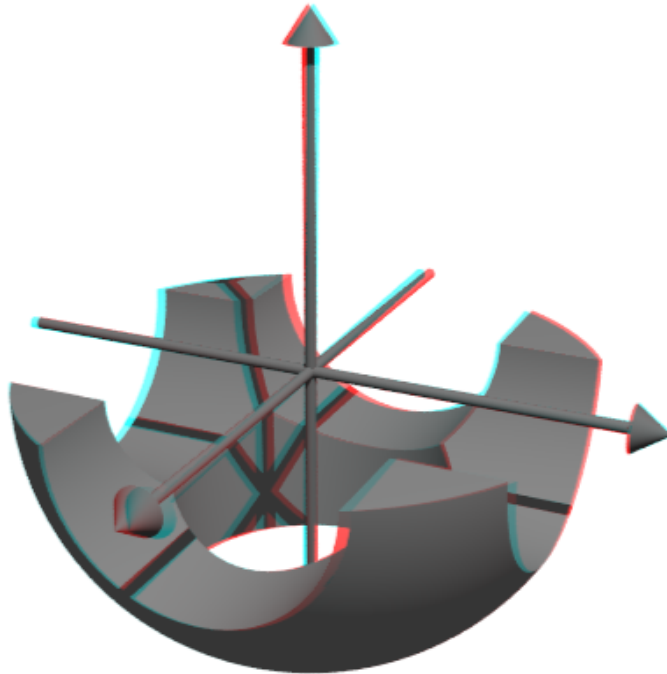
Jika Anda membuat sebuah adegan dengan objek-objek, Anda perlu menempatkan pembuatan adegan tersebut ke dalam sebuah fungsi, dan menjalankannya dua kali dengan nilai yang berbeda untuk parameter anaglyph.

```
>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. Parameter-parameternya mirip dengan yang ada dalam `povstart()` dan `povend()` yang digabungkan.

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



* Mendefinisikan Objek Sendiri

Antarmuka `povray` pada Euler berisi banyak objek. Namun, Anda tidak terbatas pada objek-objek tersebut. Anda dapat membuat objek sendiri, yang menggabungkan objek-objek lain atau benar-benar objek baru.

Kami akan mendemonstrasikan sebuah torus. Perintah `Povray` untuk ini adalah `"torus"`. Jadi, kami akan mengembalikan sebuah string dengan perintah ini beserta parameter-parameternya. Perhatikan bahwa torus selalu berada di pusat asal.

```
>function povdonat (r1,r2,look="") ...  
  
    return "torus {" + r1 + ", " + r2 + look + "}";  
endfunction
```

Ini adalah torus pertama kami.

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

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

Mari kita gunakan objek ini untuk membuat torus kedua, yang telah 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 tampilannya, kita menggunakan Phong Shading.

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

```
>povend();
```

memanggil program Povray. Namun, dalam kasus kesalahan, program ini tidak menampilkan pesan kesalahan. Oleh karena itu, Anda sebaiknya menggunakan

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

jika ada yang tidak berfungsi. Ini akan membuat jendela Povray tetap terbuka.

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



Berikut contoh yang lebih rinci. Kami menyelesaikan

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

dan menunjukkan titik-titik yang layak dan optimum dalam plot 3D.

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

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

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

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

Ya, itu sudah ada.

Selanjutnya, kita akan mendefinisikan dua objek. Yang pertama adalah bidang

$$a \cdot x \leq b$$

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

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

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

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

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

Sekarang kita dapat menggambar adegan tersebut.

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

Berikut adalah lingkaran di sekitar titik optimum.

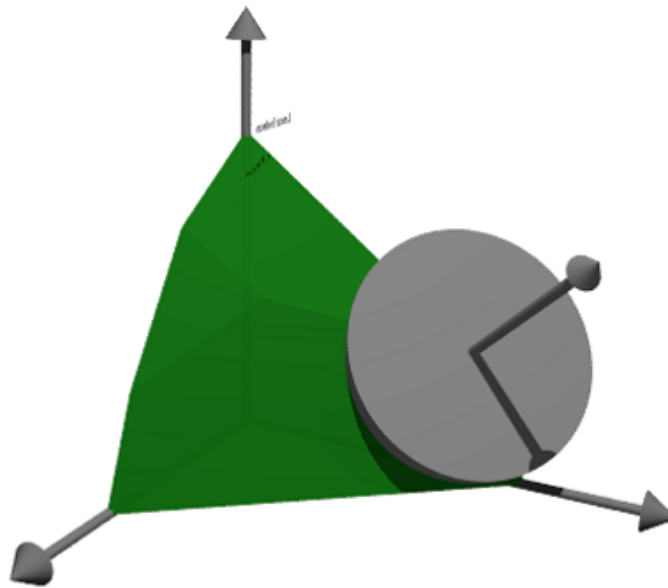
```
>writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')], ...
> povlook(red,0.9)));
```

Dan kesalahan ke arah optimal.

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

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

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



Lebih Banyak Contoh

Anda dapat menemukan beberapa contoh Povray di euler di file berikut.

See: Examples/Dandelin Spheres

See: Examples/Donat Math

See: Examples/Trefoil Knot

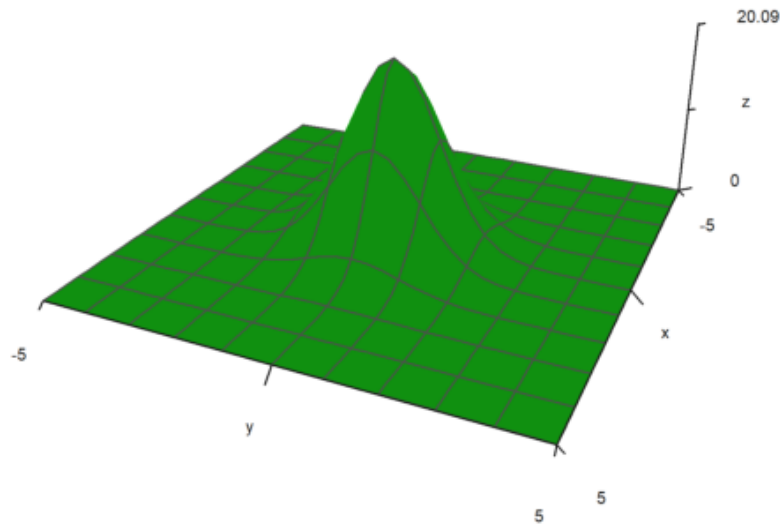
See: Examples/Optimization by Affine Scaling

Contoh Soal

1. Sketsakan plot 3D untuk fungsi berikut

$$z = \sqrt{6 - x^2 - y^2}$$

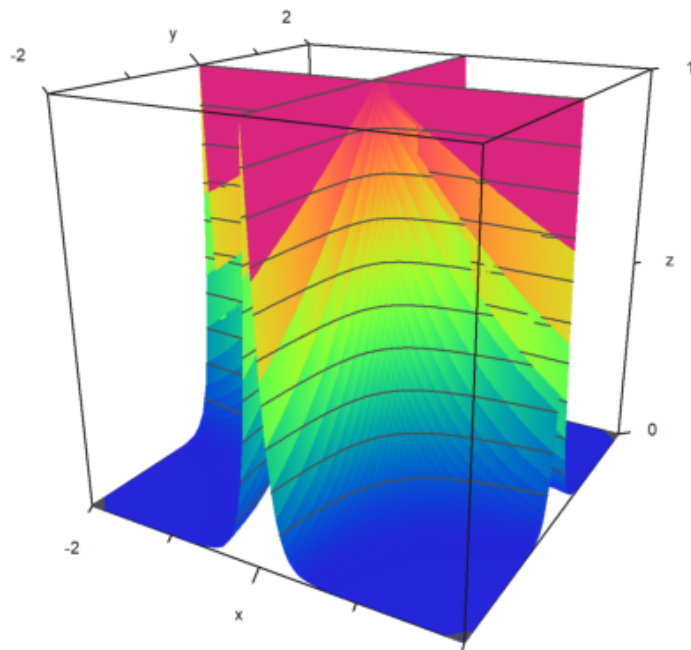
```
>plot3d("exp((6-x^2-y^2)^1/2)",r=5,n=30,fscale=2,scale=1.2,frame=3):
```



2. Sketsakan plot kontur untuk fungsi berikut

$$z = -10\sqrt{|xy|}$$

```
>plot3d("exp(-10*abs(x*y)^1/2)",r=2,n=100,level="thin", ...
> >contour,>spectral,angle=30°,height=20°):
```



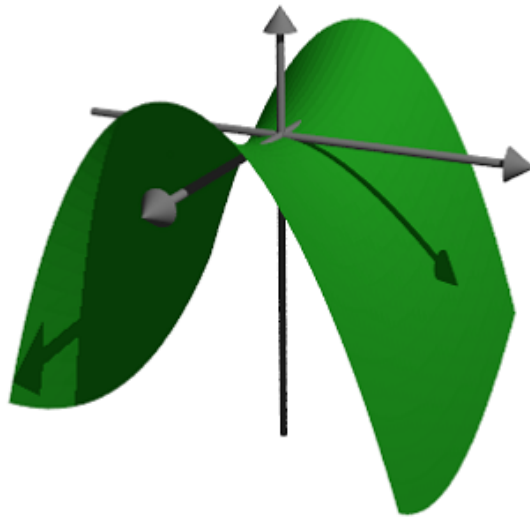
3. Sketsakan plot 3D berikut

$$\frac{9x^2 - 36y^2}{10}$$

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

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

```
>pov3d("(9*x^2-36*y^2)/10",zoom=3);
```



4. Sketsakan peta kontur untuk fungsi berikut

$$z = \frac{y}{1 + x^2 + y^2}$$

```
>plot3d("exp(y/(1+x^2+y^2))",r=4,n=100,level="thin", ...
> >contour,>spectral,angle=45°,height=30°):
```