Identitas

Nama: Aura Zahwa Allindya Astysa
NIM: 24030130093
Prodi/Kelas: Pendidikan Matematika/A24

Menggambar Plot 3D dengan EMT

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

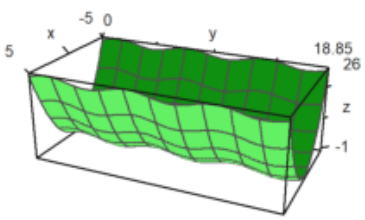
Euler menggambar fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan bagian di latar belakang. Secara umum, Euler menggunakan proyeksi pusat. Default-nya adalah dari kuadran x-y positif menuju titik asal x=y=z=0, tetapi sudut=0° menghadap ke arah sumbu y. Sudut pandang
dan tinggi dapat diubah.

Euler dapat memplot

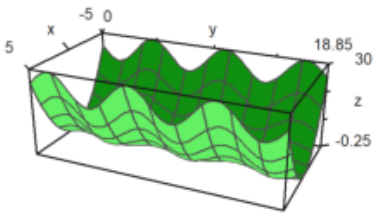
- permukaan dengan bayangan dan garis datar atau rentang datar,
- awan titik,
- kurva parametrik,
- permukaan implisit.

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

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




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



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

Fungsi Dua Variabel

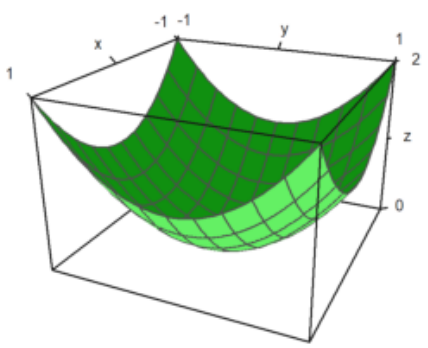
Untuk grafik fungsi, gunakan
- ekspresi sederhana dalam x dan y,
- nama fungsi dari dua variabel
- atau matriks data.

Standarnya adalah kotak kawat yang diisi dengan warna berbeda di kedua sisi. Perhatikan bahwa jumlah default interval grid adalah 10, tetapi plot menggunakan jumlah default 40x40 persegi panjang untuk membangun permukaan. Ini bisa diubah.

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

Kita gunakan default n=40 and grid=10.

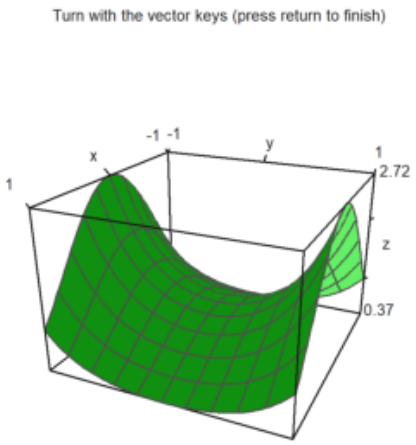
```
>plot3d("x^2+y^2"):
```



reaksi pengguna dimungkinkan dengan >parameter pengguna. Pengguna dapat menekan tombol berikut:

- kiri, kanan, atas, bawah: putar sudut pandang
- +,-: memperbesar atau memperkecil
- a: menghasilkan anaglyph (lihat di bawah)
- l: beralih memutar sumber cahaya (lihat di bawah)
- spasi: reset ke default- kembali: akhiri interaksi

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



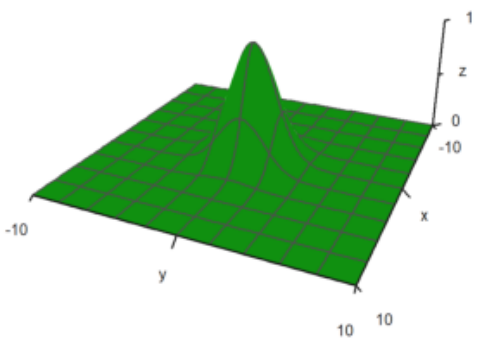
rentang plot untuk fungsi dapat ditentukan dengan

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

Ada beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik.

fscale: skala ke nilai fungsi (defaultnya adalah <fscale).
skala: angka atau vektor 1x2 untuk skala ke arah x dan y.
bingkai: 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.

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

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

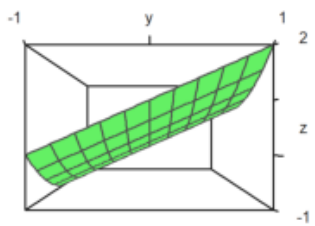
```
>view
```

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

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

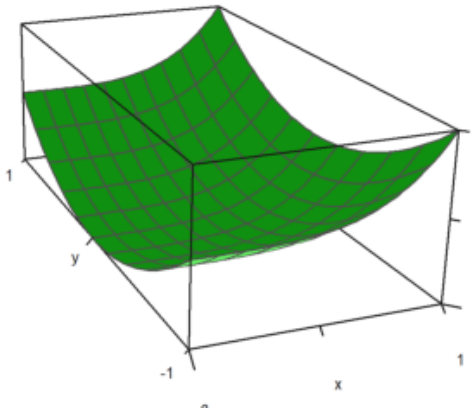
Dalam contoh berikut, sudut=0 dan tinggi=0 terlihat dari sumbu y negatif. Label sumbu untuk y disembu nyikan dalam kasus ini.

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



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

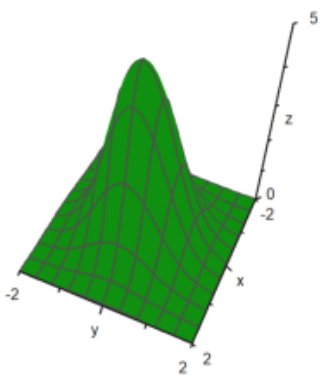
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
>  center=[0.4,0,0],zoom=5):
```



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.

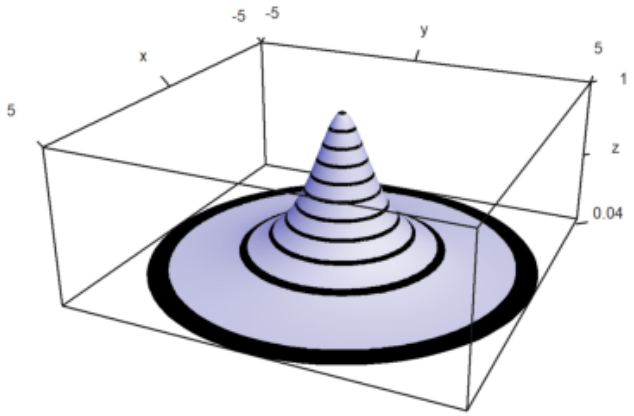
Jika kita mematikannya dengan scale=false, kita perlu berhati-hati, bahwa plot masih cocok dengan jendela plot, dengan mengubah jarak pandang atau zoom, dan memindahkan pusat.

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

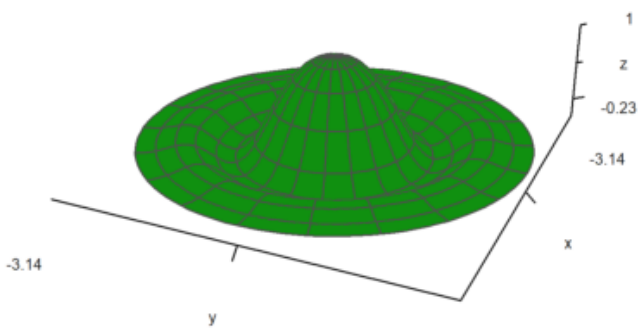


Sebuah plot kutub juga tersedia. Parameter polar=true menggambar plot polar. Fungsi tersebut harus tetap merupakan fungsi dari x dan y. Parameter "fscale" menskalakan fungsi dengan skala sendiri. Jika tidak, fungsi diskalakan agar sesuai dengan kubus.

```
>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=pi,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):
```

Here is a plot with three functions.

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

PLOT KONTUR

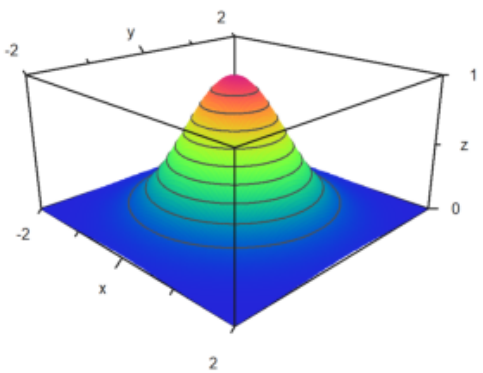
Untuk plot, Euler menambahkan garis grid. Sebagai gantinya dimungkinkan untuk menggunakan garis level dan rona satu warna atau rona berwarna spektral. Euler dapat menggambar tinggi fungsi pada plot dengan bayangan. Di semua plot 3D, Euler dapat menghasilkan anaglyph merah/sian.

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

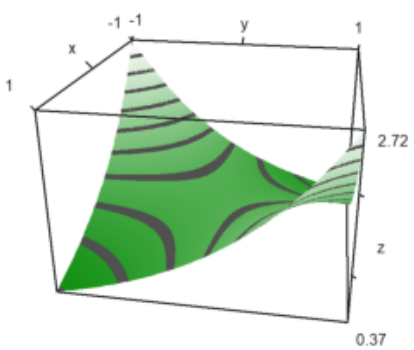
```
 Standarnya adalah level="auto", yang menghitung beberapa garis level
```

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°):
```



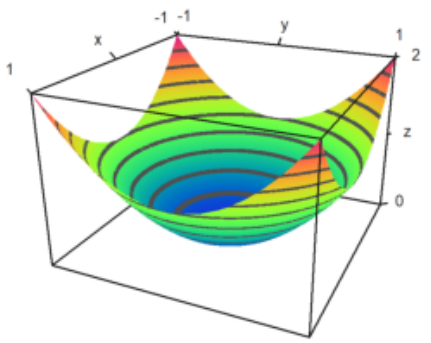

```
>plot3d("exp(x*y)",angle=100°,>contour,color=green):
```



Bayangan default menggunakan warna abu-abu. Tetapi rentang warna spektral juga tersedia.
- >spektral: Menggunakan skema spektral default
- color=...: Menggunakan warna khusus atau skema spektral

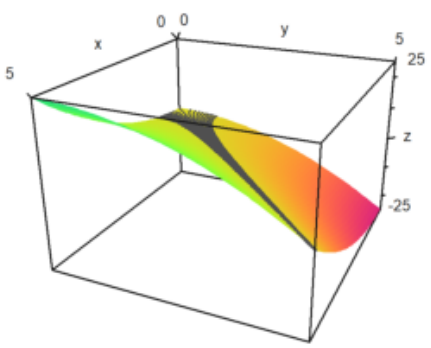
Untuk plot berikut, kami menggunakan skema spektral default dan menambah jumlah titik untuk menda patkan tampilan yang sangat halus.

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



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

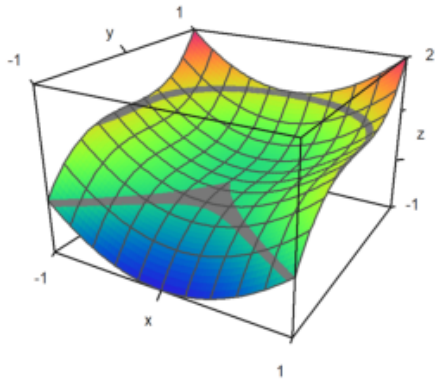
```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



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):
```

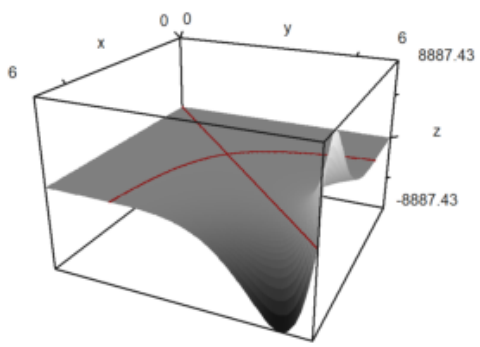


Dalam contoh berikut, kami memplot himpunan, di mana

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

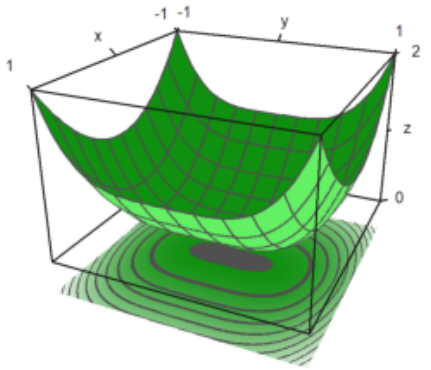
Kami menggunakan satu garis tipis untuk garis level.

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



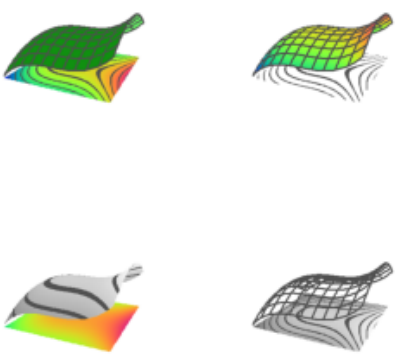
Dimungkinkan untuk menunjukkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

```
>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.

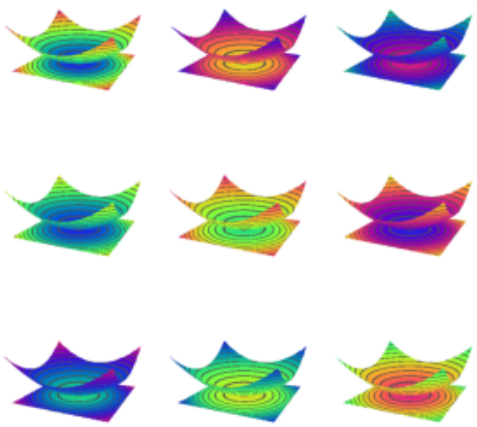
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1);  ...
>  plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2);  ...
>  plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3);  ...
>  plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4);  ...
>  plot3d(expr,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



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

- spektral: untuk rentang dari biru ke merah
- putih: untuk rentang yang lebih redup
- kuningbiru,ungu hijau,birukuning,hijaumerah
- biru kuning, hijau ungu, kuning biru, merah hijau

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

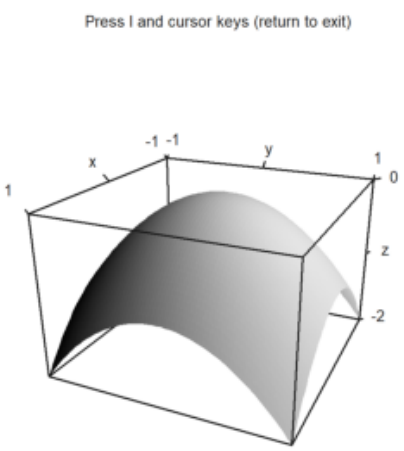


Sumber cahaya dapat diubah dengan l dan tombol kursor selama interaksi pengguna. Itu juga dapat diatur dengan parameter.

- cahaya: arah untuk cahaya
- amb: cahaya sekitar antara 0 dan 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)"):
```

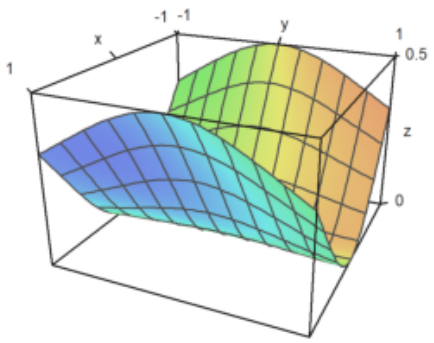


Parameter warna mengubah warna permukaan. Warna garis level juga dapat diubah.

```
>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):
```

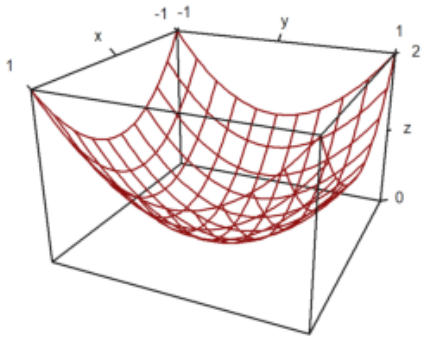
Warna 0 memberikan efek pelangi khusus.

```
>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 potongan melalui objek. Fitur dari plot3d mencakup plot implisit. Plot ini menunjukkan himpunan nol dari suatu fungsi dalam tiga variabel.
olusi dari

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

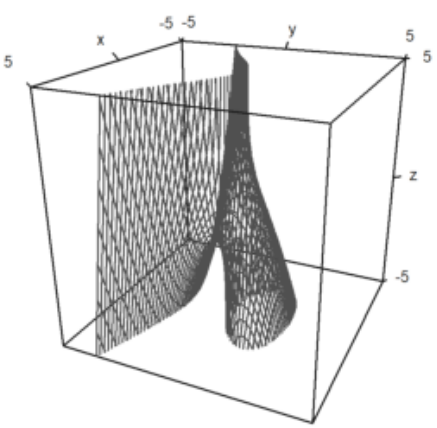
Dapat divisualisasikan dalam potongan yang sejajar dengan bidang x-y, x-z, dan y-z.

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

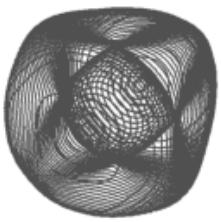
Tambahkan nilai-nilai ini jika Anda mau. Dalam contoh kita plotkan

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

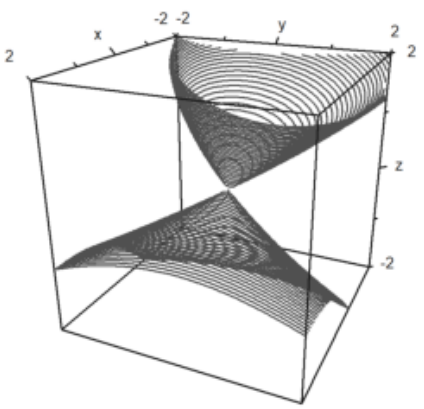
```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
```



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



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



Memplot Data 3D

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

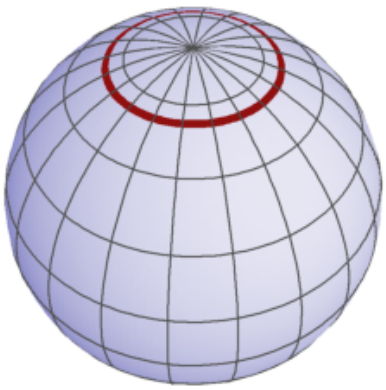
$$\gamma(t,s) = (x(t,s), y(t,s), z(t,s))$$

Karena x,y,z beruupa matriks, kita mengasumsikan bahwa (t,s) berjalan melalui sebuah grid persegi. Hasilnya, Annda dapat memplot gambar persegi panjang di ruang 3D.

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

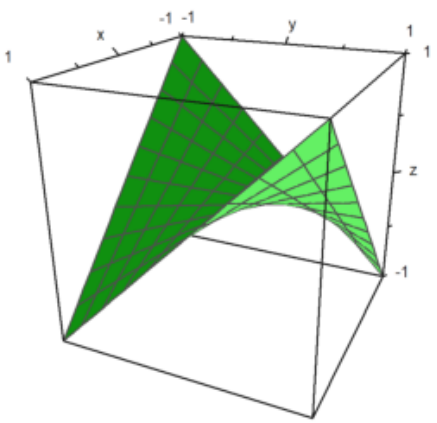
Pada contoh berikut, kita menggunakan sebuah vektor nilai t dan sebuah vektor kolom nilai s untuk memparametrisasi permukaan bola. Dalam gambar, kita bisa menandai daerah tertentu, dalam kasus kita yaitu daerah kutub.

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



Berikut adalah sebuah contoh, yaitu grafik dari suatu fungsi.

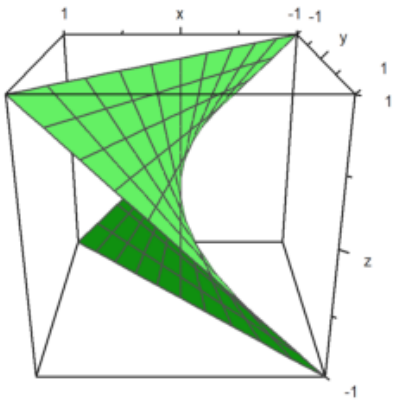
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



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

$$x = y\,z$$

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



Dengan sedikit usaha lebih, kita dapat menghasilkan banyak permukaan.

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

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

dengan

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

Kita mendistorsi ini dengan suatu faktor.

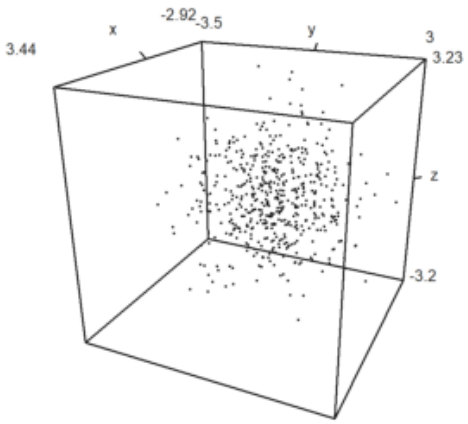
$$d(t,s) = \frac{\cos(4t)+\cos(8s)}{4}.$$

```
>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, plot berupa kumpulan titik (point cloud) juga memungkinkan. Untuk memplot data titik di ruang, kita memerlukan tiga variabel.

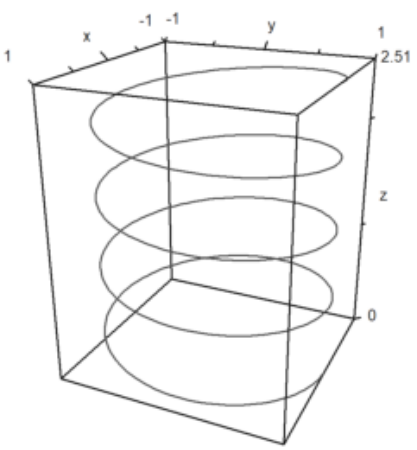
Gaya tampilannya sama seperti pada plot2d dengan points=true;

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

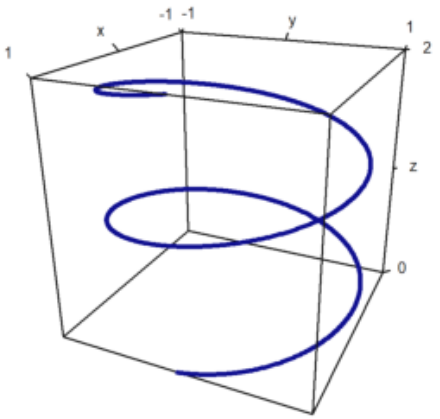


Kita juga dapat memplot kurva dalam 3D. Dalam hal ini, akan lebih mudah jika titik-titik kurva dihitung terlebih dahulu. Untuk kurva pada bidang, kita menggunakan urutan koordinat dan parameter wire=true.

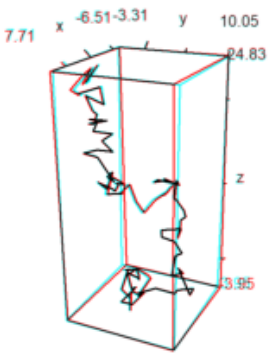
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```




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

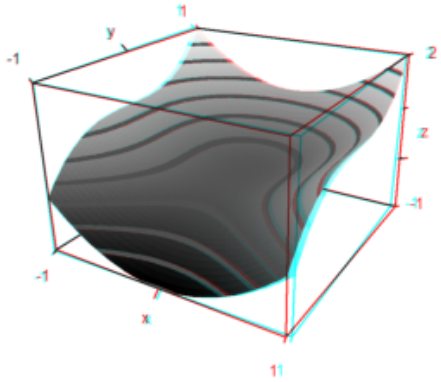



```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



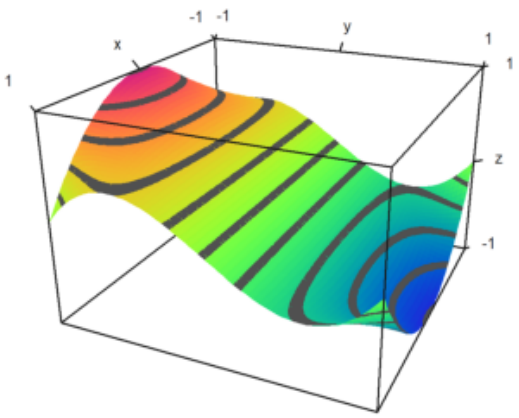
EMT juga dapat memplot dalam mode anaglyph. Untuk melihat plot seperti itu, Anda memerlukan kacamata merah/sian.

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



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

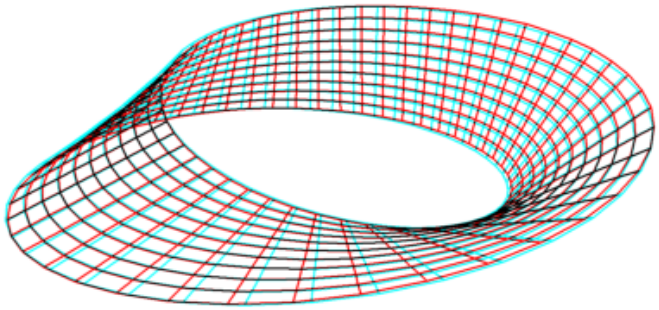
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



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

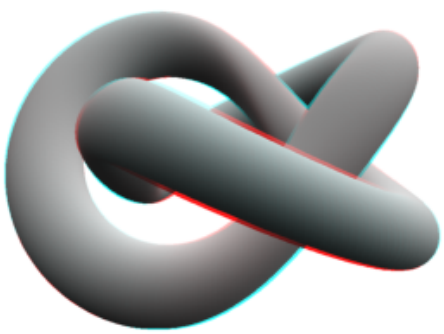
Untuk demo berikut, kita menetapkan parameter u dan v, lalu menghasilkan koordinat ruang dari parameter tersebut.

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



Berikut adalah contoh yang lebih rumit, yang tampak megah bila dilihat dengan kacamata merah/sian.

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



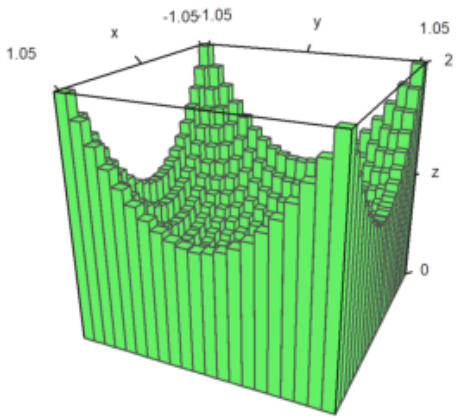
Plot Statistik

Plot batang (bar plots) juga bisa dibuat di EMT. Untuk itu, kita harus menyediakan:
- x: vektor baris dengan n+1 elemen,
- y: vektor kolom dengan n+1 elemen,

```
 z: matriks berukuran n×n berisi nilai-nilai
```

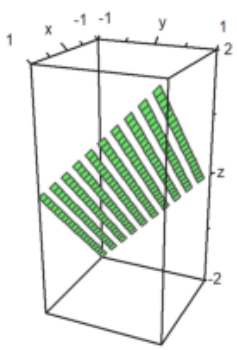
Matriks z sebenarnya bisa lebih besar, tetapi hanya bagian n×n yang akan digunakan.
Dalam contoh, pertama kita menghitung nilai-nilainya. Setelah itu, kita menyesuaikan vektor x dan y supaya posisinya terpusat pada nilai yang dipakai.

```
>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):
```



Kita bisa membagi plot suatu permukaan menjadi dua bagian atau lebih.

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

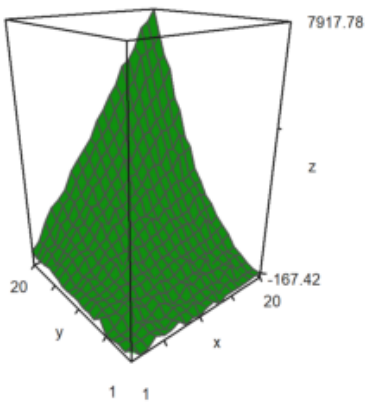


Grafik ini menunjukkan bidang datar z=x+y, tetapi dengan potongan (split)
Kalau kita punya sebuah matriks data

```
(misalnya hasil perhitungan atau data dari file) dan ingin
```

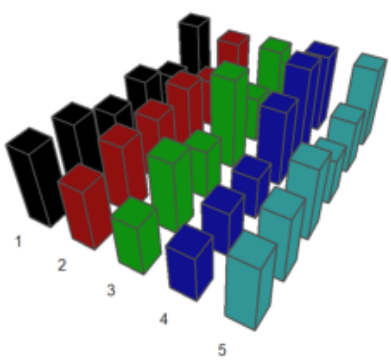
menampilkannya dalam bentuk grafik 3D, kita bisa melakukan scaling (penyesuaian skala) supaya grafiknya lebih rapi.

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



Contoh grafik distribusi hasil lemparan dadu dalam bentuk diagram batang 3D:
Ada 5 set percobaan,
masing-masing percobaan terdiri dari 100 lemparan

```
>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]):
```



Berdasarkan grafik tersebut disimpulkan bahwa:
Distribusi angka pada dadu relatif acak, tetapi cenderung mendekati seimbang. Setiap angka 1–6 muncul dengan jumlah yang kurang lebih sama dalam 100 kali lemparan. Namun, karena sifat acak, ada variasi kecil antara percobaan yang satu dengan lainnya.

Permukaan Benda Putar

Permukaan benda berputar adalah permukaan 3D yang terbentuk ketika suatu kurva diputar mengelilingi salah satu sumbu.

Misalnya, kalau sebuah garis lengkung di bidang (xy) kita putar penuh (360°) terhadap sumbu x atau y, maka terbentuklah permukaan benda berputar.

Rumus Umum

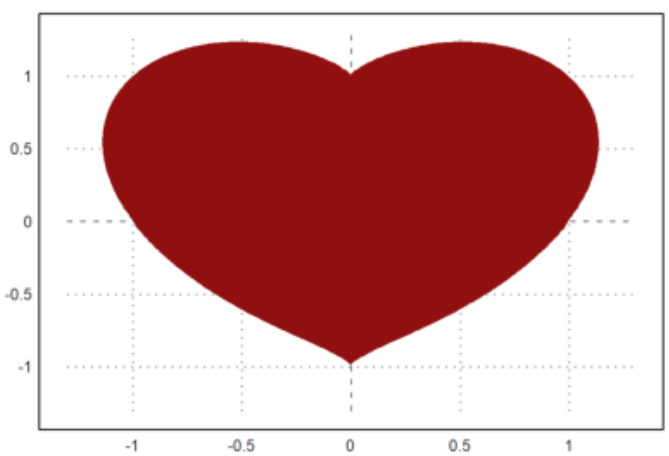
Jika kita punya kurva f(x), a<= x <= b, lalu diputar terhadap sumbu-x, maka luas permukaan yang terbentuk:

$$L = 2\pi \int_a^b y \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

Contoh Benda Berputar dalam EMT

Dalam EMT, contoh benda berputar ada kurva berbentuk hati (heart curve)

```
>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
```

Kita ingin memutar kurva hati tersebut mengelilingi sumbu-y.
Berikut adalah ekspresi (persamaan) yang mendefinisikan kurva hati itu.

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

Selanjutnya kita tetapkan

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

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

$$ekspresi$$

Dengan cara ini kita bisa mendefinisikan sebuah fungsi numerik yang menghitung nilai r, jika sudut a sudah ditentukan. Dengan fungsi tersebut, kita bisa memplot bentuk hati yang sudah 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°):
```

```
  Variable ekspresi not found!
  Use global or local variables defined in function fr.
  fr:
      useglobal; return ekspresi
  bisect:
      if f$(b,args())<y then
  Try "trace errors" to inspect local variables after errors.
  f:
      useglobal; return bisect("fr",0,2;a)
  Error in map.
```

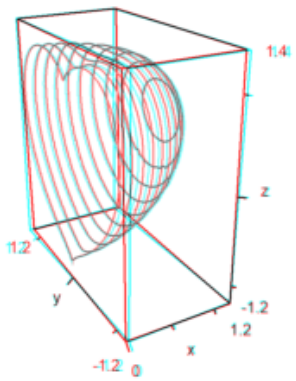
Berikut ini adalah plot 3D dari gambar di atas yang telah diputar mengelilingi sumbu-z. Kita mendefinisikan sebuah fungsi yang menggambarkan objek tersebut.

kemudian kita akan membuat plot 3D permukaan implisit berbentuk hati (heart surface).

```
>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):
```



ini menggambar permukaan hati 3D yang didefinisikan oleh fungsi implisit

Plot 3D Khusus

Fungsi plot3d memang digunakan untuk membuat grafik 3D, tetapi tidak selalu bisa memenuhi semua kebutuhan. Nah dengan Euler, kita bisa menggabungkan objek dasar untuk membuat visualisasi lain, misalnya paraboloid dan bidang singgungnya.

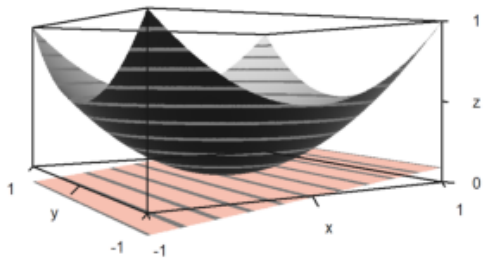
Meskipun Euler bukanlah program 3D murni, program ini bisa menggabungkan beberapa objek dasar.

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

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

Sekarang framedplot() menyediakan bingkai dan mengatur tampilan.

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

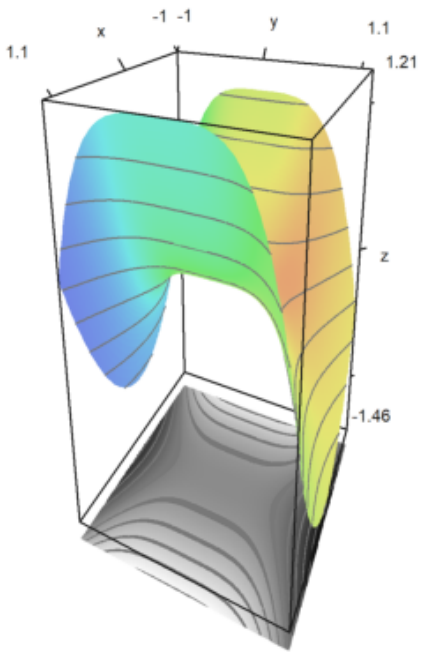


Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual. Perhatikan bahwa plot3d() menetapkan jendela ke fullwindow() secara default, tetapi plotcontourplane()dengan mengasumsikannya.

```
>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):
```



Grafik ini adalah grafik 3D permukaan dari fungsi z=x^2-y^4 dengan tambahan bidang kontur (contour plane) di bagian bawah.

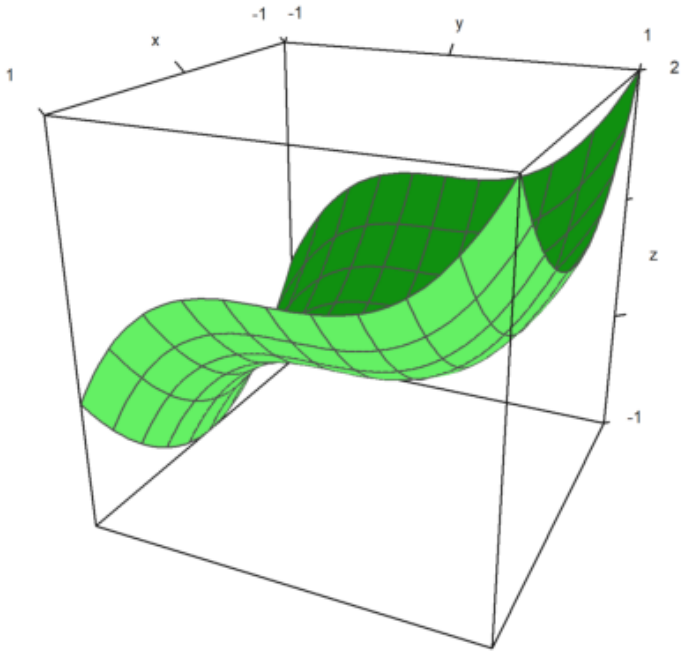
Animasi

Euler dapat menggunakan frames untuk melakukan pra-perhitungan animasi.

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

Silakan pelajari kode sumber dari fungsi rotate untuk melihat lebih detail.

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



Menggambar Povray

Dengan bantuan berkas Euler bernama povray.e, Euler dapat menghasilkan berkas Povray. Hasil yang diperoleh terlihat sangat menarik.

Anda perlu menginstal Povray (32bit atau 64bit) dari http://www.povray.org/
kemudian menambahkan sub-direktori "bin" dari Povray ke dalam environment path, atau mengatur variabel "defaultpovray" dengan alamat lengkap menuju "pvengine.exe".

Antarmuka Povray pada Euler menghasilkan berkas Povray di direktori utama pengguna, lalu memanggil Povray untuk memproses berkas tersebut. Nama berkas default adalah current.pov, dan direktori default adalah eulerhome(), biasanya di c:\Users\Username\Euler. Povray akan menghasilkan berkas PNG, yang dapat dimuat oleh Euler ke dalam notebook. Untuk membersihkan berkas-berkas ini, gunakan perintah povclear().

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

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

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

Perlu dicatat bahwa alam semesta Povray menggunakan sistem koordinat yang berbeda. Antarmuka ini menerjemahkan semua koordinat ke dalam sistem Povray. Jadi, Anda tetap bisa berpikir dalam sistem koordinat Euler dengan sumbu z menunjuk ke atas secara vertikal, serta sumbu x, y, z mengikuti aturan tangan kanan.

Anda perlu memuat berkas Povray tersebut.

```
>load povray;
```

Make sure, the Povray bin directory is in the path. If it is not edit the following variable so that it contains the path to the povray executable.

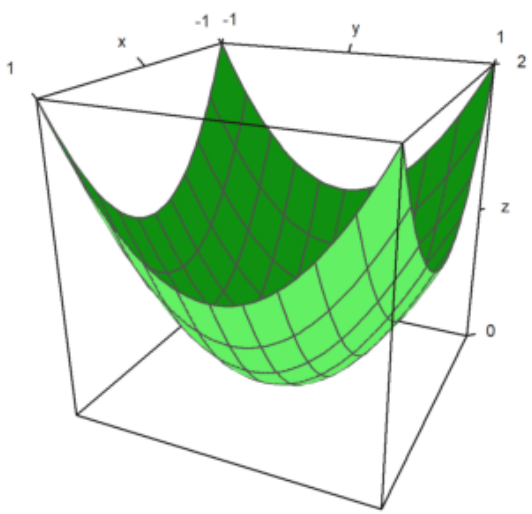
```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

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

For a first impression, we plot a simple function. The following command generates a povray file in your user directory, and runs Povray for ray tracing this file.

If you start the following command, the Povray GUI should open, run the file, and close automatically. Due to security reasons, you will be asked, if you want to allow the exe file to run. You can press cancel to stop further questions. You may have to press OK in the Povray window to acknowledge the start-up dialog of Povray.

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




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

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

We can make the function transparent and add another finish. We can also add level lines to the function plot.

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

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

Sometimes it is necessary to prevent the scaling of the function, and scale the function by hand.
We plot the set of points in the complex plane, where the product of the distances to 1 and -1 is equal to 1.

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

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

Plotting dengan Koordinat

Alih-alih menggunakan fungsi, kita dapat membuat plot dengan koordinat. Sama seperti pada plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek.
Dalam contoh ini, sebuah fungsi diputar mengelilingi sumbu-z.

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

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

Dalam contoh berikut, kita memplot sebuah gelombang redam (damped wave). Gelombang tersebut dibuat menggunakan bahasa matriks dari Euler.

Kita juga menunjukkan bagaimana sebuah objek tambahan dapat ditambahkan ke dalam sebuah adegan pov3d. Untuk pembuatan objek, lihat contoh-contoh berikutnya. Perlu dicatat bahwa plot3d akan melakukan penskalaan plot sehingga dapat sesuai dengan unit cube.

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

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

Dengan metode advanced shading dari Povray, hanya dengan sedikit titik saja sudah dapat dihasilkan permukaan yang sangat halus. Hanya pada bagian tepi dan bayangan trik ini mungkin terlihat jelas.

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

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

$$x^2 y^3$$

Persamaan permukaan adalah [x,y,Z]. Kita menghitung dua turunan terhadap x dan y dari persamaan tersebut, kemudian mengambil perkalian silang (cross product) sebagai vektor normalnya.

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

Kita mendefinisikan vektor normal sebagai hasil perkalian silang (cross product) dari turunan-turunan tersebut, kemudian mendefinisikan fungsi koordinatnya.

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

$$\left[-2 x y^3, -3 x^2 y^2, 1\right]$$

Kita hanya menggunakan 25 titik.

```
>x=-1:0.5:1; y=x';
>pov3d(x,y,Z(x,y),angle=10°, ...
>  xv=NX(x,y),yv=NY(x,y),zv=NZ(x,y),<shadow);
```

Berikut ini adalah simpul Trefoil (Trefoil knot) yang dibuat oleh A. Busser di Povray. Terdapat versi yang lebih baik dari contoh ini pada bagian examples.

Lihat: Examples\Trefoil Knot | Trefoil Knot

Untuk mendapatkan tampilan yang baik tanpa terlalu banyak titik, kita menambahkan vektor normal di sini. Kita menggunakan Maxima untuk menghitung vektor normal tersebut. Pertama, tiga fungsi koordinat didefinisikan 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 dihitung dua vektor turunan terhadap x dan y.

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

Sekarang didapatkan vektor normal, yaitu hasil perkalian silang (cross product) 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 merupakan hasil evaluasi dari ekspresi simbolik dn[i] untuk i=1,2,3. Sintaks yang digunakan adalah &"expression"(parameters). Ini merupakan alternatif dari metode pada contoh sebelumnya, di mana kita terlebih dahulu mendefinisikan ekspresi simbolik NX, NY, NZ.

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

Kita juga dapat menghasilkan sebuah grid dalam 3D.

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

Dengan povgrid(), kurva juga dapat dibuat.

```
>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 di Euler juga dapat menghasilkan objek Povray. Objek-objek ini disimpan sebagai string di Euler, dan perlu ditulis ke berkas Povray.

Kami memulai keluaran dengan povstart().

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

First we define the three cylinders, and store them in strings in Euler.

The functions povx() etc. simply returns the vector [1,0,0], which could be used instead.

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

The strings contain Povray code, which we need not understand at that point.

```
>c2
```

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

As you see, we added texture to the objects in three different colors.

That is done by povlook(), which returns a string with the relevant Povray code. We can use the default Euler colors, or define our own color. We can also add transparency, or change the ambient light.

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

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

Now we define an intersection object, and write the result to the file.

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

The intersection of three cylinders is hard to visualize, if you never saw it before.

```
>povend;
```

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

The following functions generate a fractal recursively.

The first function shows, how Euler handles simple Povray objects. The function povbox() returns a string, containing the box coordinates, the texture and the finish.

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

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

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

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

Perbedaan (differences) memungkinkan pemotongan satu objek dari objek lainnya. Sama seperti intersections, hal ini merupakan bagian dari objek CSG (Constructive Solid Geometry) pada Povray.

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

Untuk demonstrasi ini, kita mendefinisikan sebuah objek di Povray, bukan menggunakan string di Euler. Definisi objek tersebut langsung dituliskan ke dalam berkas.

Koordinat kotak (box coordinate) dengan nilai -1 berarti [-1,-1,-1].

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

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

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

Kita membuat sebuah kubus kedua, kemudian memutarnya dan memperkecil atau memperbesarnya sedikit.

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

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

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

Sekarang tambahkan tiga sumbu koordinat.

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

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

Fungsi Implisit

Povray dapat memplot himpunan di mana f(x,y,z)=0, sama seperti parameter implisit di plot3d. Namun, hasilnya terlihat jauh lebih baik.
Sintaks untuk fungsi-fungsi ini sedikit berbeda. Anda tidak dapat menggunakan keluaran 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);
>c=0.1; d=0.1; ...
>writeln(povsurface("(pow(pow(x,2)+pow(y,2)-pow(c,2),2)+pow(pow(z,2)-1,2))*(pow(pow(y,2)+p
>povend();
```

```
  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();
```

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

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

Buat permukaan implisit. Perhatikan perbedaan sintaksis dalam ekspresi tersebut

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

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

Objek Mesh

Dalam contoh ini, kami menunjukkan cara membuat objek mesh, dan menggambarnya dengan informasi tambahan.

Kami ingin memaksimalkan xy pada kondisi x+y=1 dan mendemonstrasikan perpotongan tangensial garis-garis level.

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

Kita tidak dapat menyimpan objek dalam string seperti sebelumnya, karena ukurannya terlalu besar. Jadi, kita mendefinisikan objek dalam berkas Povray menggunakan declare. Fungsi povtriangle() melakukan ini secara otomatis. Fungsi ini dapat menerima vektor normal, seperti halnya pov3d().
Berikut ini mendefinisikan objek mesh, dan langsung menuliskannya ke dalam berkas.

```
>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 definisikan dua cakram, yang akan berpotongan dengan permukaan.

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

Tuliskan permukaannya dikurangi kedua cakramnya.
rpotongan dengan permukaan.

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

Tuliskan dua titik potongnya.

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

Tuliskan titik maksimumnya.

```
>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();
```

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

Anaglif dalam Povray

Untuk menghasilkan anaglif untuk kacamata merah/sian, Povray harus dijalankan dua kali dari posisi kamera yang berbeda. Fungsi ini menghasilkan dua berkas Povray dan dua berkas PNG, yang dimuat dengan fungsi loadanaglyph().

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

Fungsi pov3d() memiliki sakelar sederhana untuk menghasilkan anaglif.
gai ekspresi simbolik.

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

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

Jika Anda membuat adegan dengan objek, Anda perlu memasukkan pembuatan adegan tersebut ke dalam suatu 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 gabungan povstart() dan povend().

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

Mendefinsikan Objek Sendiri

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

Inilah torus pertama kita.

```
>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])
```

```
  Variable or function t1 not found.
  Error in:
  t2=povobject(t1,rotate=xrotate(90°),translate=[0.8,0,0]) ...
                 ^
```

Sekarang kita tempatkan objek-objek ini ke dalam sebuah scene. Untuk tampilannya, kita gunakan Phong Shading.

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

>povend();
memanggil program Povray. Namun, jika terjadi kesalahan, program tersebut tidak akan menampilkan kesalahan tersebut. Oleh karena itu, Anda harus menggunakan
>povend(<exit);
jika ada yang tidak berhasil. Ini akan membiarkan jendela Povray terbuka

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

```
  Variable or function t1 not found.
  Error in:
  ... gle=0°,zoom=3.8,aspect=1.5); writeln(povobject(t1,povlook(gree ...
                                                    ^
```

Berikut contoh yang lebih rinci. Kita selesaikan

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

dan tunjukkan titik-titik yang layak dan titik-titik 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 periksa, apakah contoh ini mempunyai solusi.

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

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

Ya, sudah.
Selanjutnya, kita mendefinisikan dua objek. Yang pertama adalah bidang

$$a \cdot x \leq b$$

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

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

Lalu kita definisikan irisan semua setengah ruang dan 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 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 titik optimum.

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

```
  Function adm not found.
  Try list ... to find functions!
  Error in:
  ... ,zoom=3.5); writeln(adm(A,b,2,povlook(green,0.4))); writeAxes( ...
                                                          ^
```

Dan kesalahan dalam arah yang optimal.

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

```
  Cannot combine a 1x51 and a 1x3 matrix for +!
  Try "trace errors" to inspect local variables after errors.
  povarrow:
      o1=povcylinder(x,x+v,d);
```

Kita menambahkan teks ke layar. Teks hanyalah objek 3D. Kita perlu menempatkan dan memutarnya sesuai tampilan kita.
n tunjukkan titik-titik yang layak dan titik-titik optimal dalam plot 3D.

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

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

Contoh Lainnya

Anda dapat menemukan beberapa contoh lain untuk Povray di Euler pada berkas-berkas berikut.

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