Menggambar Plot 3D dengan EMT

Ini adalah pengenalan tentang grafik 3D di Euler. Kita membutuhkan grafik 3D untuk memvisualisasikan fungsi dua variabel.

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

Euler dapat menggambar

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

Grafik 3D dari suatu fungsi menggunakan plot3d. Cara termudah adalah menggambar ekspresi dalam x dan y. Parameter r menentukan rentang grafik 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.

```
>u=linspace(-1,1,20); v=linspace(0,2*pi,100)'; R=5; X=(R+u^2)*cos(v); Y=(R+u^2)*sin(v); Z=u^2 + u*sin(10*v); plot3d(X, Y,
```

Fungsi Dua Variabel

Untuk grafik suatu fungsi, gunakan

```
  ekspresi sederhana dalam x dan y,
```

- nama fungsi dua variabel,
- atau matriks data.

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

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

Kita menggunakan default n=40 dan grid=10.

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

User interaction dapat dilakukan dengan parameter >user. Pengguna dapat menekan tombol-tombol berikut.

- kiri, kanan, atas, bawah: mengubah sudut pandang
- +,-: memperbesar atau memperkecil
- a: menghasilkan anaglyph (lihat di bawah)
- l: mengaktifkan/menonaktifkan sumber cahaya (lihat di bawah)
- space: mengembalikan ke pengaturan default
- enter: mengakhiri interaksi

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

Rentang plot untuk fungsi dapat ditentukan dengan

- a,b: rentang 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: menskalakan ke nilai fungsi (defaultnya adalah <fscale).
scale: angka atau vektor 1x2 untuk menskalakan ke arah x dan y.
frame: jenis bingkai (default 1).

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

Tampilan dapat diubah dengan berbagai cara.

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

ilai default dapat diperiksa atau diubah menggunakan fungsi view(). Fungsi ini mengembalikan parameter dalam urutan di atas.

```
>view
```

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

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

Dalam contoh berikut, sudut=0 dan ketinggian=0 dilihat dari sumbu y negatif. Label sumbu y disembunyikan dalam kasus ini.

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

Plot selalu mengarah ke pusat kubus plot. Anda dapat memindahkan pusat tersebut menggunakan 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):
```

Grafik diskalakan agar sesuai dengan kubus unit untuk tampilan. Jadi, tidak perlu mengubah jarak atau zoom tergantung pada ukuran grafik. Label-label tersebut merujuk pada ukuran sebenarnya, bagaimanapun.

Jika Anda mematikan opsi ini dengan scale=false, Anda perlu memastikan bahwa grafik masih sesuai dengan jendela tampilan dengan mengubah jarak tampilan 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):
```

Grafik polar juga tersedia. Parameter polar=true akan menggambar grafik polar. Fungsi tersebut tetap harus merupakan fungsi dari x dan y. Parameter “fscale” akan menskalakan fungsi dengan skala tersendiri. Jika tidak, fungsi akan 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):
```

Parameter rotate memutar fungsi dalam sumbu x.

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

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


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

Berikut adalah grafik dengan tiga fungsi.

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

Grafik Kontur

Euler menambahkan garis kisi. Namun, ini memungkinkan untuk menggunakan garis kontur dan warna tunggal atau warna spektral. Euler dapat menggambar ketinggian fungsi pada grafik dengan bayangan. Pada semua grafik 3D, Euler dapat menghasilkan anaglyph merah/cyan.

- >hue: Mengaktifkan bayangan cahaya alih-alih garis kisi.
- >contour: Menggambar garis kontur otomatis pada plot.
- level=... (or levels): Vektor nilai untuk garis kontur.

Defaultnya adalah level="auto", yang menghitung garis level secara otomatis. Seperti yang terlihat pada plot, level-level tersebut sebenarnya adalah rentang level.

Gaya default dapat diubah. Untuk plot kontur berikut, kita menggunakan grid yang lebih halus dengan 100x100 titik, menskalakan fungsi dan plot, serta menggunakan sudut pandang yang berbeda.

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

Pembayangan default menggunakan warna abu-abu. Namun, rentang spektral warna juga tersedia.

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

ntuk grafik berikut, kita menggunakan skema spektral default dan meningkatkan jumlah titik untuk mendapatkan tampilan yang sangat halus.

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

Alih-alih menggunakan garis level otomatis, kita juga dapat mengatur nilai garis level. Hal 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 grafik berikut, kita menggunakan dua rentang level yang sangat luas, yaitu dari -0,1 hingga 1, dan dari 0,9 hingga 1. Rentang ini dimasukkan sebagai matriks dengan batas level sebagai kolom.

Selain itu, kita menambahkan 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, kita menggambar himpunan, di mana

Kita menggunakan garis tipis tunggal untuk garis tingkat.

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

Ini memungkinkan untuk menampilkan bidang kontur di bawah grafik. Warna dan jarak ke grafik dapat ditentukan.

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

Berikut ini beberapa gaya tambahan. Kita selalu mematikan bingkai, dan menggunakan berbagai skema warna untuk grafik dan grid.

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

Ada beberapa skema spektral lain, bernomor dari 1 hingga 9. Namun, Anda juga dapat menggunakan format color=value, di mana value

```
spectral: untuk rentang dari biru hingga merah
```

- white: untuk rentang yang lebih redup
- yellowblue,purplegreen,blueyellow,greenred
- blueyellow, greenpurple,yellowblue,redgreen

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

Sumber cahaya dapat diubah menggunakan tombol l dan tombol panah selama interaksi pengguna. Sumber cahaya juga dapat diatur melalui parameter.

- light: arah cahaya
- amb: cahaya ambient antara 0 dan 1

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

Parameter warna mengubah warna permukaan. Warna garis tingkat 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 spesial.

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

Permukaan tersebut juga dapat transparan.

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

Grafik Implisit

Terdapat juga grafik implisit dalam tiga dimensi. Euler menghasilkan irisan melalui objek-objek tersebut. Fitur-fitur plot3d mencakup grafik implisit. Grafik-grafik ini menampilkan himpunan nol dari suatu fungsi dalam tiga variabel.
Solusi dari

dapat divisualisasikan dalam irisan 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 diinginkan. Dalam contoh ini, kita menggambar

```
>plot3d("x^2+y^3+z*y-1",r=5,implicit=3):
>c=1; d=1;
>plot3d("((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^2)-d",r=2,<frame,>implicit,>use
```

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

Membuat Grafik 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 fx(x,y), fy(x,y), fz(x,y).

Karena x, y, z adalah matriks, kita mengasumsikan bahwa (t,s) berjalan melalui grid persegi. Akibatnya, Anda dapat memplot gambar persegi panjang di ruang.

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

Dalam contoh berikut, kita menggunakan vektor nilai t dan vektor kolom nilai s untuk memparameterkan permukaan bola. Dalam gambar, kita dapat menandai wilayah, dalam hal ini 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°):
```

Berikut ini adalah 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 jenis permukaan. Berikut adalah permukaan yang sama sebagai fungsi

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

Dengan usaha yang lebih, kita dapat menghasilkan banyak permukaan.

Dalam contoh berikut, kita membuat tampilan bergradasi dari bola yang terdistorsi. Koordinat biasa untuk bola tersebut adalah

dengan

Kita mendistorsi ini dengan faktor

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

Tentu saja, titik awan juga mungkin. Untuk menggambar data titik di 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 3D. Dalam hal ini, lebih mudah untuk menghitung terlebih dahulu titik-titik kurva. Untuk kurva di bidang datar, 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 menampilkan grafik dalam mode anaglyph. Untuk melihat grafik tersebut, Anda memerlukan kacamata merah/biru.

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

Seringkali, skema warna spektral digunakan untuk grafik. Hal ini menonjolkan ketinggian fungsi.

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

Euler juga dapat menggambar permukaan yang diparameterkan, ketika parameternya adalah nilai x, y, dan z dari gambar grid persegi panjang di ruang.

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

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

Berikut adalah contoh yang lebih rumit, yang tampak megah saat dilihat dengan kacamata merah/biru.

```
>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):
```

Grafik Statistik

Grafik batang juga dapat dibuat. Untuk ini, kita perlu menyediakan

- x: vektor baris dengan n+1 elemen
- y: vektor kolom dengan n+1 elemen
- z: matriks nxn berisi nilai-nilai.

z dapat memiliki ukuran yang lebih besar, tetapi hanya nilai-nilai nxn yang akan digunakan.

Dalam contoh ini, kita terlebih dahulu menghitung nilai-nilai tersebut. Kemudian kita menyesuaikan x dan y agar vektor-vektor tersebut berpusat pada nilai-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):
```

Ini memungkinkan untuk membagi bidang 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 berkas dan perlu memplotnya dalam 3D, Anda dapat menskalakan matriks tersebut ke rentang [-1,1] menggunakan fungsi scale(M), atau menskalakan matriks dengan >zscale. Hal ini dapat dikombinasikan dengan faktor skalasi individual yang diterapkan secara tambahan.

```
>i=1:20; j=i'; ...
 plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8):
>Z=intrandom(5,100,6); v=zeros(5,6); ...
 loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
 columnsplot3d(v',scols=1:5,ccols=[1:5]):
```

Permukaan Benda Putar

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

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

Selanjutnya, kita tetapkan

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

Hal ini memungkinkan untuk mendefinisikan fungsi numerik, yang menentukan nilai r jika a diberikan. Dengan fungsi tersebut, kita dapat menggambar jantung yang diputar sebagai permukaan parametrik.

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

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

```
>function f(x,y,z) ...
 r=x^2+y^2;
 return (r+z^2-1)^3-r*z^3;
  endfunction


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

Grafik 3D Khusus

Fungsi plot3d memang berguna, tetapi tidak memenuhi semua kebutuhan. Selain rutinitas dasar, ini memungkinkan Anda membuat grafik berbingkai dari objek apa pun yang Anda inginkan.

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

```
>function myplot ...
   y=-1:0.01:1; x=(-1:0.01:1)';
   plot3d(x,y,0.2*(x-0.1)/2,<scale,<frame,>hue, ..
```

```
    hues=0.5,>contour,color=orange);
  h=holding(1);
  plot3d(x,y,(x^2+y^2)/2,<scale,<frame,>contour,>hue);
  holding(h);
endfunction
```

Sekarang fungsi 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 fungsi plot3d() secara default mengatur jendela ke fullwindow(), tetapi fungsi plotcontourplane() mengasumsikan hal itu.

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


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


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

Animasi

Euler dapat menggunakan frame untuk menghitung animasi secara pra-komputasi.

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

Silakan pelajari sumber kode rotate untuk melihat detail lebih lanjut.

```
>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 menarik untuk dilihat.

Anda perlu menginstal Povray (32-bit atau 64-bit) dari **http://www.povray.org/, dan memasukkan subdirektori "bin" dari Povray ke dalam jalur lingkungan, atau mengatur variabel 'defaultpovray' dengan jalur lengkap yang mengarah ke "pvengine.exe".**

Interface Povray Euler menghasilkan berkas Povray di direktori home pengguna, dan memanggil Povray untuk memproses berkas-berkas tersebut. 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 notebook. Untuk membersihkan berkas-berkas ini, gunakan povclear().

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

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

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

Perhatikan bahwa sistem koordinat Povray berbeda. Interface ini menerjemahkan semua koordinat ke sistem koordinat Povray. Jadi Anda dapat terus menggunakan sistem koordinat Euler dengan sumbu z mengarah vertikal ke atas, dan sumbu x, y, z dalam arah tangan kanan.
nda perlu memuat berkas Povray.

```
>load povray;
```

Pastikan direktori bin Povray ada dalam jalur sistem. Jika tidak, edit variabel berikut agar berisi jalur ke file executable Povray.

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


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

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

Jika Anda menjalankan perintah berikut, antarmuka grafis Povray (GUI) seharusnya terbuka, menjalankan berkas, dan menutup secara otomatis. Karena alasan keamanan, Anda akan diminta untuk mengizinkan berkas exe dijalankan. Anda dapat menekan "Cancel" untuk menghentikan pertanyaan selanjutnya. Anda mungkin perlu menekan "OK" di jendela Povray untuk mengonfirmasi dialog startup Povray.

```
>plot3d("x^2+y^2",zoom=2):
>pov3d("x^2+y^2",zoom=3);
```

Kita dapat membuat fungsi tersebut transparan dan menambahkan lapisan akhir lainnya. Kita juga dapat menambahkan garis tingkat pada grafik fungsi.

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

Terkadang perlu mencegah penskalaan fungsi dan melakukan penskalaan fungsi secara manual.

Kami menggambar himpunan titik di 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);
```

Membuat Grafik dengan Koordinat

Alih-alih menggunakan fungsi, kita dapat membuat grafik dengan koordinat. Seperti pada plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek.

Dalam contoh ini, kita memutar 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(red,0.1),height=50°,axis=0,zoom=4,light=[10,5,15]);
```

Dalam contoh berikut, kita menggambar gelombang teredam. Gelombang ini dihasilkan menggunakan bahasa matriks Euler.

ita juga menunjukkan cara menambahkan objek tambahan ke dalam adegan pov3d. Untuk pembuatan objek, lihat contoh-contoh berikut. Perhatikan bahwa plot3d menskalakan grafik agar sesuai dengan kubus satuan.

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

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

ntuk itu, kita perlu menambahkan vektor normal di 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 ini dan mengambil hasil kali silang sebagai vektor normal.

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

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

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

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

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 ini adalah simpul Trefoil yang dibuat oleh A. Busser di Povray. Ada versi yang lebih baik dari ini di contoh-contoh.

Trefoil Knot

ntuk tampilan yang baik tanpa terlalu banyak titik, kita menambahkan vektor normal di sini. Kita menggunakan Maxima untuk menghitung vektor normal tersebut. 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 dua vektor turunan terhadap x dan y.

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

Sekarang yang normal, yaitu hasil kali silang dari dua turunan tersebut.

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

Kami sekarang mengevaluasi semua ini secara numerik.

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

Vektor normal adalah hasil evaluasi dari ekspresi simbolik dn[i] untuk i=1,2,3. Sintaks untuk ini adalah &“ekspresi”(parameter). Ini adalah alternatif dari metode pada contoh sebelumnya, di mana kita mendefinisikan ekspresi simbolik 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 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, kita menggunakan pov3d untuk menggambar permukaan. Antarmuka Povray di Euler juga dapat menghasilkan objek Povray. Objek-objek ini disimpan sebagai string di Euler dan perlu ditulis ke berkas Povray.

Kita memulai output dengan povstart().

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

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

ungsi povx() dan sebagainya hanya mengembalikan vektor [1,0,0], yang dapat digunakan sebagai pengganti.

```
>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 }
       }
```

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

Hal ini dilakukan dengan fungsi povlook(), yang mengembalikan string berisi kode Povray yang relevan. Kita dapat menggunakan warna Euler default, atau mendefinisikan warna kita sendiri. Kita juga dapat menambahkan transparansi, atau mengubah cahaya ambient.

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

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

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

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

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

```
>povend;
```

Fungsi-fungsi berikut ini menghasilkan fraktal secara rekursif.

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

```
>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 objek lainnya. Seperti persimpangan, perbedaan merupakan bagian dari objek CSG (Common Sense Geometry) dalam Povray.

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

Untuk demonstrasi ini, kami mendefinisikan objek di Povray, bukan menggunakan string di Euler. Definisi ditulis ke file secara langsung.

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

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

Kita dapat menggunakan objek ini dalam fungsi povobject(), yang mengembalikan string seperti biasa.

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

Kami membuat kubus kedua, lalu memutar dan mengubah ukurannya sedikit.

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

Kemudian kita ambil selisih antara 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();
```

Fungsi Implisit

Povray dapat menggambar himpunan di mana f(x,y,z)=0, sama seperti parameter implisit dalam plot3d. Hasilnya terlihat jauh lebih baik, bagaimanapun.

Sintaks untuk fungsi-fungsi ini sedikit berbeda. Anda tidak dapat menggunakan output dari Maxima atau ekspresi Euler.

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


      Error : Povray error!

      Error generated by error() command

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


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


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

Buat permukaan implisit. Perhatikan sintaks yang berbeda dalam ekspresi tersebut.

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

Objek Jaring

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

Kami ingin memaksimalkan xy dengan syarat x+y=1 dan menunjukkan sentuhan tangensial pada garis tingkat.

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

Kami tidak dapat menyimpan objek dalam string seperti sebelumnya, karena ukurannya terlalu besar. Oleh karena itu, kami mendefinisikan objek tersebut dalam berkas Povray menggunakan #declare. Fungsi povtriangle() melakukan hal ini secara otomatis. Fungsi ini dapat menerima vektor normal sama seperti pov3d().

erikut 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 mendefinisikan dua lingkaran, 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);
```

Tulis permukaan dikurangi dua cakram.

```
>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 pada nilai 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();
```

Anaglyph dalam Povray

Untuk menghasilkan anaglyph untuk kacamata merah/biru, Povray harus dijalankan dua kali dari posisi kamera yang berbeda. Hal ini menghasilkan dua berkas Povray dan dua berkas PNG, yang dimuat menggunakan fungsi loadanaglyph().

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

Fungsi pov3d() memiliki opsi sederhana untuk menghasilkan anaglyph.

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

Jika Anda membuat skenario dengan objek, Anda perlu memasukkan proses pembuatan skenario ke dalam fungsi, dan menjalankannya dua kali dengan nilai yang berbeda untuk parameter anaglyph.

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

Fungsi povanaglyph() melakukan semua ini. Parameter-parameternya mirip dengan gabungan parameter povstart() dan povend().

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

Menentukan Objek Sendiri

Interface Povray Euler mengandung banyak objek. Namun, Anda tidak terbatas pada objek-objek tersebut. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau objek yang sepenuhnya 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 kami.

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

      torus {0.8,0.2}
```

Mari kita gunakan objek ini untuk membuat torus kedua, yang dipindahkan 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 tempatkan objek-objek ini ke dalam sebuah adegan. Untuk tampilannya, kami menggunakan Phong Shading.

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

 >povend();
```

memanggil program Povray. Namun, jika terjadi kesalahan, ia tidak menampilkan kesalahan tersebut. Oleh karena itu, Anda harus menggunakan

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

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

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

Berikut adalah contoh yang lebih rinci. Kami memecahkan

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, sudah.

Selanjutnya kita mendefinisikan dua objek. Yang pertama adalah pesawat

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

Kemudian kita mendefinisikan irisan dari semua ruang setengah 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
```

Kita sekarang bisa memplot 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 adalah lingkaran di sekitar titik optimum.

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

Dan kesalahan dalam arah yang optimal.

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

Kita menambahkan teks ke layar. Teks hanyalah objek 3D. Kita perlu menempatkan dan memutarnya sesuai tampilan kita.

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

Contoh Lainnya

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

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

Teori Kalkulus Multivariabel

Menggambar plot 3D, atau grafik fungsi dua variabel

adalah visualisasi dari permukaan di ruang tiga dimensi. Fungsi f(x, y) memetakan setiap pasangan koordinat (x,y) di bidang XY ke ketinggian z tertentu.

Konsep Dasar Permukaan Kuadratik

Dalam Kalkulus, banyak permukaan 3D yang penting dikelompokkan sebagai Permukaan Kuadratik, yaitu grafik dari persamaan berderajat dua dalam tiga variabel,

Jenis-jenis Permukaan Kuadratik yang sering diplot:
1. Paraboloid Eliptik: Berbentuk mangkuk/cawan. Contoh:

2. Paraboloid Hiperbolik: Berbentuk pelana kuda (saddle). Contoh:

3. Elipsoid: Berbentuk bola yang terdistorsi. Contoh:

Metode Plotting 3D

EMT menggunakan beberapa metode untuk memvisualisasikan permukaan:
1. Permukaan Ekspisit: Paling mudah, langsung menggunakan ekspresi dalam x dan y.

2. Permukaan Parametrik: Digunakan untuk objek kompleks seperti bola, torus, atau knot.

3. Permukaan Implisit: Digunakan untuk objek yang tidak dapat diwakili oleh fungsi eksplisit (misalnya, Elipsoid atau Donat). EMT menggambar ini dengan membuat irisan/potongan (cuts) pada bidang koordinat.

Contoh Lain

Paraboloid Hiperbolik

Konsep: Permukaan kuadratik

ikenal sebagai "pelana kuda" karena memiliki kelengkungan berlawanan di sumbu x dan y.

Contoh:
Gambarlah Paraboloid Hiperbolik

Tampilkan garis kontur di permukaan dan di bidang XY.

```
>plot3d("x^2-y^2", r=2, >contour, >cp, cpcolor=gray):
```

Plot menunjukkan bentuk pelana. Garis kontur terlihat sebagai hiperbola (untuk z tidak sama dengan
) yang menyeberang di titik kritis (0,0,0). Peta kontur 2D (gray) di bawah plot memudahkan pemahaman bentuk permukaan.

Permukaan Benda Putar

Konsep: Permukaan yang dihasilkan dengan memutar kurva

di sekitar sumbu Z. Di sini, kita memutar parabola terbalik.

Contoh: Gambarlah permukaan yang dihasilkan dari memutar kurva:

di sekitar sumbu z, untuk

```
>plot3d("3-x^2", a=-1.5, b=1.5, rotate=true, >spectral):
```

Hasilnya adalah bentuk seperti cawan atau mangkuk terbalik (bagian bawah wine glass). Pewarnaan spektral membedakan bagian atas (radius kecil) dan bawah (radius besar) cawan.