

TUGAS PROYEK

APLIKASI KOMPUTER

Untuk Memenuhi Tugas Mata Kuliah Aplikasi Komputer
Dosen Pengampu: Drs. Sahid M.Sc



Oleh:

Dessy Chintya Nur Feika
(22305141023)
Matematika E 2022

PROGRAM STUDI MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI YOGYAKARTA
2023

Daftar Isi

1 KB Pekan 2 : Belajar Menggunakan Software EMT	5
2 KB Pekan 3-4 : Menggunakan EMT untuk menyelesaikan masalah-masalah Al-jabar	23
3 KB Pekan 5-6 : Menggunakan EMT untuk mengambar grafik 2 dimensi (2D)	29
4 KB Pekan 7-8 : Menggunakan EMT untuk mengambar grafik 3 dimensi (3D)	129
5 KB Pekan 11-12 : Menggunakan EMT untuk Geometri	231
6 KB Pekan 13-14 : Menggunakan EMT untuk Statistika	277

BAB 1

KB Pekan 2 : Belajar Menggunakan Software EMT

Pendahuluan dan Pengenalan Cara Kerja EMT

Selamat datang! Ini adalah pengantar pertama ke Euler Math Toolbox (disingkat EMT atau Euler). EMT adalah sistem terintegrasi yang merupakan perpaduan kernel numerik Euler dan program komputer aljabar Maxima.

- Bagian numerik, GUI, dan komunikasi dengan Maxima telah dikembangkan oleh R. Grothmann, seorang profesor matematika di Universitas Eichstätt, Jerman. Banyak algoritma numerik dan pustaka software open source yang digunakan di dalamnya.
- Maxima adalah program open source yang matang dan sangat kaya untuk perhitungan simbolik dan aritmatika tak terbatas. Software ini dikelola oleh sekelompok pengembang di internet.
- Beberapa program lain (LaTeX, Povray, Tiny C Compiler, Python) dapat digunakan di Euler untuk memungkinkan perhitungan yang lebih cepat maupun tampilan atau grafik yang lebih baik.

Yang sedang Anda baca (jika dibaca di EMT) ini adalah berkas notebook di EMT. Notebook aslinya bawaan EMT (dalam bahasa Inggris) dapat dibuka melalui menu File, kemudian pilih "Open Tutorias and Example", lalu pilih file "00 First Steps.en". Perhatikan, file notebook EMT memiliki ekstensi ".en". Melalui notebook ini Anda akan belajar menggunakan software Euler untuk menyelesaikan berbagai masalah matematika.

Panduan ini ditulis dengan Euler dalam bentuk notebook Euler, yang berisi teks (deskriptif), baris-baris perintah, tampilan hasil perintah (numerik, ekspresi matematika, atau gambar/plot), dan gambar yang disisipkan dari file gambar.

Untuk menambah jendela EMT, Anda dapat menekan [F11]. EMT akan menampilkan jendela grafik di layar desktop Anda. Tekan [F11] lagi untuk kembali ke tata letak favorit Anda. Tata letak disimpan untuk sesi berikutnya.

Anda juga dapat menggunakan [Ctrl]+[G] untuk menyembunyikan jendela grafik. Selanjutnya Anda dapat beralih antara grafik dan teks dengan tombol [TAB].

Seperti yang Anda baca, notebook ini berisi tulisan (teks) berwarna hijau, yang dapat Anda edit dengan mengklik kanan teks atau tekan menu Edit -> Edit Comment atau tekan [F5],

dan juga baris perintah EMT yang ditandai dengan ">" dan berwarna merah. Anda dapat menyisipkan baris perintah baru dengan cara menekan tiga tombol bersamaan: [Shift]+[Ctrl]+[Enter].

Komentar (Teks Uraian)

Komentar atau teks penjelasan dapat berisi beberapa "markup" dengan sintaks sebagai berikut.

- * Judul
- ** Sub-Judul
- latex: $F(x) = \int_a^x f(t) dt$
- mathjax: $\frac{x^2-1}{x-1} = x + 1$
- maxima: 'integrate(x^3,x) = integrate(x^3,x) + "C"
- http://www.euler-math-toolbox.de
- See: http://www.google.de | Google
- image: hati.png
- ---

Hasil sintaks-sintaks di atas (tanpa diawali tanda strip) adalah sebagai berikut.

Judul

Sub-Judul

$$F(x) = \int_a^x f(t) dt$$

$$\frac{x^2 - 1}{x - 1} = x + 1$$

maxima: 'integrate(x^3,x) = integrate(x^3,x) + "C"

http://www.euler-math-toolbox.de

See: http://www.google.de | Google



Gambar diambil dari folder images di tempat file notebook berada dan tidak dapat dibaca dari Web. Untuk "See:", tautan (URL)web lokal dapat digunakan.

Paragraf terdiri atas satu baris panjang di editor. Pergantian baris akan memulai baris baru. Paragraf harus dipisahkan dengan baris kosong.

```
>// baris perintah diawali dengan >, komentar (keterangan) diawali dengan //
```

Baris Perintah

Mari kita tunjukkan cara menggunakan EMT sebagai kalkulator yang sangat canggih.

EMT berorientasi pada baris perintah. Anda dapat menuliskan satu atau lebih perintah dalam satu baris perintah. Setiap perintah harus diakhiri dengan koma atau titik koma.

- Titik koma menyembunyikan output (hasil) dari perintah.
- Sebuah koma mencetak hasilnya.
- Setelah perintah terakhir, koma diasumsikan secara otomatis (boleh tidak ditulis).

Dalam contoh berikut, kita mendefinisikan variabel r yang diberi nilai 1,25. Output dari definisi ini adalah nilai variabel. Tetapi karena tanda titik koma, nilai ini tidak ditampilkan. Pada kedua perintah di belakangnya, hasil kedua perhitungan tersebut ditampilkan.

```
>r=1.25; pi*r^2, 2*pi*r
```

4.90873852123

7.85398163397

Latihan untuk Anda

- Sisipkan beberapa baris perintah baru
- Tulis perintah-perintah baru untuk melakukan suatu perhitungan yang Anda inginkan, boleh menggunakan variabel, boleh tanpa variabel.

Beberapa catatan yang harus Anda perhatikan tentang penulisan sintaks perintah EMT.

- Pastikan untuk menggunakan titik desimal, bukan koma desimal untuk bilangan!
- Gunakan * untuk perkalian dan ^ untuk eksponen (pangkat).
- Seperti biasa, * dan / bersifat lebih kuat daripada + atau -.
- ^ mengikat lebih kuat dari *, sehingga pi * r ^ 2 merupakan rumus luas lingkaran.
- Jika perlu, Anda harus menambahkan tanda kurung, seperti pada $2 ^ (2 ^ 3)$.

Perintah $r = 1.25$ adalah menyimpan nilai ke variabel di EMT. Anda juga dapat menulis $r := 1.25$ jika mau. Anda dapat menggunakan spasi sesuka Anda.

Anda juga dapat mengakhiri baris perintah dengan komentar yang diawali dengan dua garis miring (//).

```
>r := 1.25 // Komentar: Menggunakan := sebagai ganti =
```

1.25

Argumen atau input untuk fungsi ditulis di dalam tanda kurung.

```
>sin(45°), cos(pi), log(sqrt(E))
```

0.707106781187

-1

0.5

Seperti yang Anda lihat, fungsi trigonometri bekerja dengan radian, dan derajat dapat diubah dengan °. Jika keyboard Anda tidak memiliki karakter derajat tekan [F7], atau gunakan fungsi deg() untuk mengonversi.

EMT menyediakan banyak sekali fungsi dan operator matematika. Hampir semua fungsi matematika sudah tersedia di EMT. Anda dapat melihat daftar lengkap fungsi-fungsi matematika di EMT pada berkas Referensi (klik menu Help -> Reference)

Untuk membuat rangkaian komputasi lebih mudah, Anda dapat merujuk ke hasil sebelumnya dengan "%". Cara ini sebaiknya hanya digunakan untuk merujuk hasil perhitungan dalam baris perintah yang sama.

```
>(sqrt(5)+1)/2, %^2-%+1 // Memeriksa solusi x^2-x+1=0
```

1.61803398875

2

Latihan untuk Anda

- Buka berkas Reference dan baca fungsi-fungsi matematika yang tersedia di EMT.
 - Sisipkan beberapa baris perintah baru.
 - Lakukan contoh-contoh perhitungan menggunakan fungsi-fungsi matematika di EMT.
-

Satuan

EMT dapat mengubah unit satuan menjadi sistem standar internasional (SI). Tambahkan satuan di belakang angka untuk konversi sederhana.

```
>1miles // 1 mil = 1609,344 m
```

1609.344

Beberapa satuan yang sudah dikenal di dalam EMT adalah sebagai berikut. Semua unit diakhiri dengan tanda dolar (\$), namun boleh tidak ditulis dengan mengaktifkan easy-units.

```
kilometer$:=1000;  
km$:=kilometer$;  
cm$:=0.01;  
mm$:=0.001;  
minute$:=60;  
min$:=minute$;  
minutes$:=minute$;  
hour$:=60*minute$;  
h$:=hour$;  
hours$:=hour$;  
day$:=24*hour$;  
days$:=day$;  
d$:=day$;  
year$:=365.2425*day$;  
years$:=year$;  
y$:=year$;  
inch$:=0.0254;  
in$:=inch$;  
feet$:=12*inch$;  
foot$:=feet$;  
ft$:=feet$;  
yard$:=3*feet$;  
yards$:=yard$;  
yd$:=yard$;  
mile$:=1760*yard$;  
miles$:=mile$;
```

```
kg$:=1;  
sec$:=1;  
ha$:=10000;  
Ar$:=100;  
Tagwerk$:=3408;  
Acre$:=4046.8564224;  
pt$:=0.376mm;
```

Untuk konversi ke dan antar unit, EMT menggunakan operator khusus, yakni ->.

```
>4km -> miles, 4inch -> " mm"
```

```
2.48548476895  
101.6 mm
```

Format Tampilan Nilai

Akurasi internal untuk nilai bilangan di EMT adalah standar IEEE, sekitar 16 digit desimal. Aslinya, EMT tidak mencetak semua digit suatu bilangan. Ini untuk menghemat tempat dan agar terlihat lebih baik. Untuk mengatramilan satu bilangan, operator berikut dapat digunakan.

```
>pi
```

```
3.14159265359
```

```
>longest pi
```

```
3.141592653589793
```

```
>long pi
```

```
3.14159265359
```

```
>short pi
```

```
3.1416
```

```
>shortest pi
```

3.1

```
>fraction pi
```

312689/99532

```
>short 1200*1.03^10, long E, longest pi
```

1612.7
2.71828182846
3.141592653589793

Format aslinya untuk menampilkan nilai menggunakan sekitar 10 digit. Format tampilan nilai dapat diatur secara global atau hanya untuk satu nilai.

Anda dapat mengganti format tampilan bilangan untuk semua perintah selanjutnya. Untuk mengembalikan ke format aslinya dapat digunakan perintah "deformat" atau "reset".

```
>longestformat; pi, deformat; pi
```

3.141592653589793
3.14159265359

Kernel numerik EMT bekerja dengan bilangan titik mengambang (floating point) dalam presisi ganda IEEE (berbeda dengan bagian simbolik EMT). Hasil numerik dapat ditampilkan dalam bentuk pecahan.

```
>1/7+1/4, fraction %
```

0.392857142857
11/28

Perintah Multibaris

Perintah multi-baris membentang di beberapa baris yang terhubung dengan "..." di setiap akhir baris, kecuali baris terakhir. Untuk menghasilkan tanda pindah baris tersebut, gunakan tombol [Ctrl]+[Enter]. Ini akan menyambung perintah ke baris berikutnya dan menambahkan "..." di akhir baris sebelumnya. Untuk menggabungkan suatu baris ke baris sebelumnya, gunakan [Ctrl]+[Backspace].

Contoh perintah multi-baris berikut dapat dijalankan setiap kali kursor berada di salah satu barisnya. Ini juga menunjukkan bahwa ... harus berada di akhir suatu baris meskipun baris tersebut memuat komentar.

```
>a=4; b=15; c=2; // menyelesaikan a*x^2+b*x+c=0 secara manual ...
>D=sqrt(b^2/(a^2*4)-c/a); ...
>-b/(2*a) + D, ...
>-b/(2*a) - D
```

```
-0.138444501319
-3.61155549868
```

Menampilkan Daftar Variabel

Untuk menampilkan semua variabel yang sudah pernah Anda definisikan sebelumnya (dan dapat dilihat kembali nilainya), gunakan perintah "listvar".

```
>listvar
```

r	1.25
a	4
b	15
c	2
D	1.73655549868123

Perintah listvar hanya menampilkan variabel buatan pengguna. Dimungkinkan untuk menampilkan variabel lain, dengan menambahkan string termuat di dalam nama variabel yang diinginkan.

Perlu Anda perhatikan, bahwa EMT membedakan huruf besar dan huruf kecil. Jadi variabel "d" berbeda dengan variabel "D".

Contoh berikut ini menampilkan semua unit yang diakhiri dengan "m" dengan mencari semua variabel yang berisi "m\$".

```
>listvar m$
```

km\$	1000
cm\$	0.01
mm\$	0.001

```
nm$          1853.24496
gram$        0.001
m$           1
hquantum$    6.62606957e-34
atm$         101325
```

Untuk menghapus variabel tanpa harus memulai ulang EMT gunakan perintah "remvalue".

```
>remvalue a,b,c,D
>D
```

Variable D not found!

Error in:

D ...
^

Menampilkan Panduan

Untuk mendapatkan panduan tentang penggunaan perintah atau fungsi di EMT, buka jendela panduan dengan menekan [F1] dan cari fungsinya. Anda juga dapat mengklik dua kali pada fungsi yang tertulis di baris perintah atau di teks untuk membuka jendela panduan.

Coba klik dua kali pada perintah "intrandom" berikut ini!

```
>intrandom(10,6)
```

[4, 2, 6, 2, 4, 2, 3, 2, 2, 6]

Di jendela panduan, Anda dapat mengklik kata apa saja untuk menemukan referensi atau fungsi.

Misalnya, coba klik kata "random" di jendela panduan. Kata tersebut boleh ada dalam teks atau di bagian "See:" pada panduan. Anda akan menemukan penjelasan fungsi "random", untuk menghasilkan bilangan acak berdistribusi uniform antara 0,0 dan 1,0. Dari panduan untuk "random" Anda dapat menampilkan panduan untuk fungsi "normal", dll.

```
>random(10)
```

[0.270906, 0.704419, 0.217693, 0.445363, 0.308411, 0.914541, 0.1935

0.463387, 0.095153, 0.595017]

```
>normal(10)
```

```
[ -0.495418,  1.6463, -0.390056, -1.98151,  3.44132,  0.308178, -0.7334  
-0.526167,  1.10018,  0.108453]
```

Matriks dan Vektor

EMT merupakan suatu aplikasi matematika yang mengerti "bahasa matriks". Artinya, EMT menggunakan vektor dan matriks untuk perhitungan-perhitungan tingkat lanjut. Suatu vektor atau matriks dapat didefinisikan dengan tanda kurung siku. Elemen-elemennya dituliskan di dalam tanda kurung siku, antar elemen dalam satu baris dipisahkan oleh koma(,), antar baris dipisahkan oleh titik koma (;).

Vektor dan matriks dapat diberi nama seperti variabel biasa.

```
>v=[4,5,6,3,2,1]
```

```
[4, 5, 6, 3, 2, 1]
```

```
>A=[1,2,3;4,5,6;7,8,9]
```

1	2	3
4	5	6
7	8	9

Karena EMT mengerti bahasa matriks, EMT memiliki kemampuan yang sangat canggih untuk melakukan perhitungan matematis untuk masalah-masalah aljabar linier, statistika, dan optimisasi.

Vektor juga dapat didefinisikan dengan menggunakan rentang nilai dengan interval tertentu menggunakan tanda titik dua (:),seperti contoh berikut ini.

```
>c=1:5
```

```
[1, 2, 3, 4, 5]
```

```
>w=0:0.1:1
```

```
[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
```

```
>mean(w^2)
```

0.35

Bilangan Kompleks

EMT juga dapat menggunakan bilangan kompleks. Tersedia banyak fungsi untuk bilangan kompleks di EMT. Bilangan imaginer

$$i = \sqrt{-1}$$

dituliskan dengan huruf I (huruf besar I), namun akan ditampilkan dengan huruf i (i kecil).

```
re(x) : bagian riil pada bilangan kompleks x.  
im(x) : bagian imaginer pada bilangan kompleks x.  
complex(x) : mengubah bilangan riil x menjadi bilangan kompleks.  
conj(x) : Konjugat untuk bilangan bilangan komplkes x.  
arg(x) : argumen (sudut dalam radian) bilangan kompleks x.  
real(x) : mengubah x menjadi bilangan riil.
```

Apabila bagian imaginer x terlalu besar, hasilnya akan menampilkan pesan kesalahan.

```
>sqrt(-1) // Error!  
>sqrt(complex(-1))
```

```
>z=2+3*I, re(z), im(z), conj(z), arg(z), deg(arg(z)), deg(arctan(3/2))
```

```
2+3i  
2  
3  
2-3i  
0.982793723247  
56.309932474  
56.309932474
```

```
>deg(arg(I)) // 90°
```

```
90
```

```
>sqrt(-1)
```

```
Floating point error!  
Error in sqrt  
Error in:  
sqrt(-1) ...  
^
```

```
>sqrt(complex(-1))
```

0+1i

EMT selalu menganggap semua hasil perhitungan berupa bilangan riil dan tidak akan secara otomatis mengubah ke bilangan kompleks.

Jadi akar kuadrat -1 akan menghasilkan kesalahan, tetapi akar kuadrat kompleks didefinisikan untuk bidang koordinat dengan cara seperti biasa. Untuk mengubah bilangan riil menjadi kompleks, Anda dapat menambahkan 0i atau menggunakan fungsi "complex".

```
>complex(-1), sqrt(%)
```

-1+0i
0+1i

Matematika Simbolik

EMT dapat melakukan perhitungan matematika simbolis (eksak) dengan bantuan software Maxima. Software Maxima otomatis sudah terpasang di komputer Anda ketika Anda memasang EMT. Meskipun demikian, Anda dapat juga memasang software Maxima tersendiri (yang terpisah dengan instalasi Maxima di EMT).

Pengguna Maxima yang sudah mahir harus memperhatikan bahwa terdapat sedikit perbedaan dalam sintaks antara sintaks asli Maxima dan sintaks ekspresi simbolik di EMT.

Untuk melakukan perhitungan matematika simbolis di EMT, awali perintah Maxima dengan tanda "&". Setiap ekspresi yang dimulai dengan "&" adalah ekspresi simbolis dan dikерjakan oleh Maxima.

```
>& (a+b)^2
```

$$(b + a)^2$$

```
>&expand((a+b)^2), &factor(x^2+5*x+6)
```

$$b^2 + 2ab + a^2$$

$$(x + 2)(x + 3)$$

```
>&solve(a*x^2+b*x+c,x) // rumus abc
```

$$[x = \frac{(-\sqrt{b^2 - 4ac}) - b}{2a}, x = \frac{\sqrt{b^2 - 4ac} - b}{2a}]$$

```
>&(a^2-b^2)/(a+b), &ratsimp(%) // ratsimp menyederhanakan bentuk pecahan
```

$$\frac{a^2 - b^2}{b + a}$$
$$a - b$$

```
>10! // nilai faktorial (modus EMT)
```

3628800

```
>&10! //nilai faktorial (simbolik dengan Maxima)
```

3628800

Untuk menggunakan perintah Maxima secara langsung (seperti perintah pada layar Maxima) awali perintahnya dengan tanda ":" pada baris perintah EMT. Sintaks Maxima disesuaikan dengan sintaks EMT (disebut "modus kompatibilitas").

```
>factor(1000) // mencari semua faktor 1000 (EMT)
```

[2, 2, 2, 5, 5, 5]

```
>::: factor(1000) // faktorisasi prima 1000 (dengan Maxima)
```

$$\begin{matrix} 3 & 3 \\ 2 & 5 \end{matrix}$$

```
>::: factor(20!)
```

$$\begin{matrix} 18 & 8 & 4 & 2 \\ 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 \end{matrix}$$

Jika Anda sudah mahir menggunakan Maxima, Anda dapat menggunakan sintaks asli perintah Maxima dengan menggunakan tanda ">:::" untuk mengawali setiap perintah Maxima di EMT. Perhatikan, harus ada spasi antara ">:::" dan perintahnya.

```
>::: binomial(5,2); // nilai C(5,2)
```

$$10$$

```
>::: binomial(m,4); // C(m,4)=m!/(4!(m-4)!)
```

$$\frac{(m - 3)(m - 2)(m - 1)m}{24}$$

```
>::: trigexpand(cos(x+y)); // rumus cos(x+y)=cos(x) cos(y)-sin(x) sin(y)
```

$$\cos(x) \cos(y) - \sin(x) \sin(y)$$

```
>::: trigexpand(sin(x+y));
```

$$\cos(x) \sin(y) + \sin(x) \cos(y)$$

```
>::: trigsimp(((1-sin(x)^2)*cos(x))/cos(x)^2+tan(x)*sec(x)^2) //menyederhanakan
```

$$\frac{\sin^4(x) + \cos^4(x)}{\cos^3(x)}$$

Untuk menyimpan ekspresi simbolik ke dalam suatu variabel digunakan tanda "&=".

```
>p1 &= (x^3+1)/(x+1)
```

$$\frac{x^3 + 1}{x + 1}$$

```
>&ratsimp(p1)
```

$$x^2 - x + 1$$

Untuk mensubstitusikan suatu nilai ke dalam variabel dapat digunakan perintah "with".

```
>&p1 with x=3 // (3^3+1)/(3+1)
```

```
>&p1 with x=a+b, &ratsimp(%) //substitusi dengan variabel baru
```

$$\frac{(b + a)^3 + 1}{b^3 + a^3 + 1}$$

$$\frac{b^2 + (2a - 1)b^2 + a^2 - a + 1}{b^2 + a^2 - a + 1}$$

```
>&diff(p1,x) //turunan p1 terhadap x
```

$$\frac{3x^2(x^3 + 1)}{(x + 1)^2}$$

```
>&integrate(p1,x) // integral p1 terhadap x
```

$$\frac{2x^3 - 3x^2 + 6x}{6}$$

Tampilan Matematika Simbolik dengan LaTeX

Anda dapat menampilkan hasil perhitungan simbolik secara lebih bagus menggunakan LaTeX. Untuk melakukan hal ini, tambahkan tanda dolar (\$) di depan tanda & pada setiap perintah Maxima.

Perhatikan, hal ini hanya dapat menghasilkan tampilan yang diinginkan apabila komputer Anda sudah terpasang software LaTeX.

```
> $& (a+b)^2  
> $&expand( (a+b)^2), $&factor(x^2+5*x+6)
```

Selamat Belajar dan Berlatih!

Baik, itulah sekilas pengantar penggunaan software EMT. Masih banyak kemampuan EMT yang akan Anda pelajari dan praktikkan.

Sebagai latihan untuk memperlancar penggunaan perintah-perintah EMT yang sudah dijelaskan di atas, silakan Anda lakukan hal-hal sebagai berikut.

- Carilah soal-soal matematika dari buku-buku Matematika.
- Tambahkan beberapa baris perintah EMT pada notebook ini.
- Selesaikan soal-soal matematika tersebut dengan menggunakan EMT.

Pilih soal-soal yang sesuai dengan perintah-perintah yang sudah dijelaskan dan dicontohkan di atas.

Penyelesaian 1

```
> a=60, b=5, a/b
```

60
5
12

```
> p=13, q=7, p-q
```

13
7
6

```
>cos(45°)
```

0.707106781187

Penyelesaian 2

```
> cosh(45°)
```

1.32460908925

```
> log10(2)
```

0.301029995664

Penyelesaian 3

```
>
```

$$\begin{array}{r} 65 \\ 72 \end{array} \times$$

```
> 65 * 72
```

4680

$$88 - 16 : 2 + 5 \times 4 =$$

```
> 88 - 16 / 2 + 5 * 4
```

100

$$100 : 4 \times 20 =$$

```
> 100 / 4 * 20
```

500

BAB 2

KB Pekan 3-4 : Menggunakan EMT untuk menyelesaikan masalah-masalah Aljabar

Nama : Dessy Chintya Nur Feika

NIM : 22305141023

Kelas : Matematika E 2023

Pilih minimal 5 soal dari setiap Latihan atau tipe soal (misalnya diantara soal-soal yang sudah saya blok). Jangan lupa tuliskan soalnya di teks komentar (dengan format LaTeX) dan beri penjelasan hasil output-nya. Ubah file notebook pekerjaan Anda menjadi file PDF menggunakan salah satu metode di atas.

R.2 Exercise Set

```
> $& (( -2/5 ) ^0)
> $& ( a^-2 / b^-8 )
> $& ( ( ( 24 * a^10 * b^-8 * c^7 ) / ( 12 * a^6 * b^-3 * c^5 ) ) ^-5 )
> $& ( ( ( 125 * p^12 * q^-14 * r^22 ) / ( 25 * p^8 * q^6 * r^-15 ) ) ^-4 )
> $& ( ( 2^6 * 2^-3 ) + 2^10 + 2^-8 )
> $& ( ( 4 * ( 8 - 6 ) ^2 - ( 4 * 3 ) + ( 2 * 8 ) ) / ( 3^1 + 19^0 ) )
> $& ( ( [ 4 * ( 8 - 6 ) ^2 + 4 ] ) / ( 2^2 * ( 2^3 + 5 ) ) )
```

R.3 Excercise Set

```
>$& ((2*x+3*y+z-7)+(4*x-2*y-z+8)+(-3*x+y-2*z-4))  
>$& ((2*x^2+12*x*y-11)+(6*x^2-2*x+4)+(-x^2-y-2))  
>$& ((3*x^2-2*x-x^3+2)-(5*x^2-8*x-x^3+4))  
>$& ((y-5)^2)  
>$& ((n+6)(n-6))  
>$& ((3*x+5*y)(3*x-5*y))
```

R.4 Exercise Set

```
>$& (t^2+8*t+15)  
>$& (4*y^2-5)  
>$& (m^2-9*n^2)  
>$& (z^2-81), $&solve(z^2-81)  
>$& (x^2+12*x+36), $&solve(x^2+12*x+36)
```

R.5 Exercise Set

```
>$& (7*(3*x+6)=11-(x+2)), $&solve(7*(3*x+6)=11-(x+2))  
>$& (9*(2*x+8)=20-(x+5)), $&solve(9*(2*x+8)=20-(x+5))  
>$& (y^2-4*y-45=0), $&solve(y^2-4*y-45=0)  
>$& (V=(4/3)*pi*r^3)  
>$& (21*n^2-10=n), $&solve(21*n^2-10=n)
```

R.6 Exercise Set

```
>$& ((x^2-4)/(x^2-4*x+4)), $&solve((x^2-4)/(x^2-4*x+4))  
>$& ((x^3-6*x^2+9*x)/(x^3-3*x^2)), $&solve((x^3-6*x^2+9*x)/(x^3-3*x^2))  
>$& ((6*y^2+12*y-48)/(3*y^2-9*y+6)), $&solve((6*y^2+12*y-48)/(3*y^2-9*y+6))  
>$& (((r-s)/(r+s)).((r^2-s^2)/(r-s)^2))  
>$& ((6-x)/(x^2-36))
```

R.7 Exercise Set

```
>${&(sqrt(-21)^2)
>${&(sqrt(9*y^2))
>${&(sqrt(180))
>${&(5*sqrt(2)+3*sqrt(32))
>${&(2*sqrt(32)+3*sqrt(8)-4*sqrt(18))
>${&(125^(1/3))}
```

2.3 Exercise Set

```
>${&(h(x)=1/(x-2)^4), ${&solve(h(x)=1/(x-2)^4)
>${&(h(x)=1/sqrt(3*x+7)), ${&solve(h(x)=1/sqrt(3*x+7))
>${&(h(x)=(sqrt(x)-3)^4), ${&solve(h(x)=(sqrt(x)-3)^4)
>${&(sqrt((x-5)/(x+2))), ${&solve(sqrt((x-5)/(x+2)))}
```

```
>${&(h(x)=(x+2)^3-5*(x+2)^2+3*(x+2)-1), ${&solve(h(x)=(x+2)^3-5*(x+2)^2+3*(x+2)-1))
```

3.1 Exercise Set

```
>${&((-5+3*i)+(7+8*i))
>${&((-5-i)+(6+2*i))
>${&((6-4*i)-(-5+i))
>${&(sqrt(-4).sqrt(-36))
>${&(3/(5-11*i))}
```

3.4 Exercise Set

```

>$& (1/4+1/5=1/t), $&solve(1/4+1/5=1/t)
>$& (1/3-5/6=1/x), $&solve(1/3-5/6=1/x)
>$& ((x+2)/4-(x-1)/5=15), $&solve((x+2)/4-(x-1)/5=15)
>$& ((3*y+5)/(y^2+5*y)+(y+4)/(y+5)=(y+1)/y)
>$& (sqrt(3*x-4)=1), $&solve(sqrt(3*x-4)=1)

```

3.5 Exercise Set

```

>$& (x^2+4*x-5=0), $&solve(x^2+4*x-5=0)
>$& (3*x^2+2*x=8), $&solve(3*x^2+2*x=8)
>$& (5*x^2=15), $&solve(5*x^2=15)
>$& (x^2+10=0), $&solve(x^2+10=0)
>$& ((2*y+5)*(3*y-1)=0), $&solve((2*y+5)*(3*y-1)=0)

```

Chapter 3 Test

```

>$& (x+5*sqrt(x)-36=0), $&solve(x+5*sqrt(x)-36=0)
>$& ((3/(3*x+4))+(2/(x-1))=2), $&solve((3/(3*x+4))+(2/(x-1))=2)
>$& (sqrt(x+4)-2=1), $&solve(sqrt(x+4)-2=1)
>$& (sqrt(x+4)-sqrt(x-4)=2), $&solve(sqrt(x+4)-sqrt(x-4)=2)
>$& (x^2+4*x=1), $&solve(x^2+4*x=1)

```

4.1 Exercise Set

```

>$& (f(x)=x^3-9*x^2+14*x+24), $&solve(f(x)=x^3-9*x^2+14*x+24)
>$& (f(x)=2*x^3-3*x^2+x+6), $&solve(f(x)=2*x^3-3*x^2+x+6)
>$& (g(x)=x^4-6*x^3+8*x^2+6*x-9), $&solve(g(x)=x^4-6*x^3+8*x^2+6*x-9)
>$& (g(x)=x^4-x^3-3*x^2+5*x-2), $&solve(g(x)=x^4-x^3-3*x^2+5*x-2)
>$& (f(x)=(x+3)^2*(x-1)), $&solve(f(x)=(x+3)^2*(x-1))

```

4.3 Exercise Set

```
>${& ( (2*x^4+7*x^3+x-12)+(x+3) )  
>${& ( (x^3-7*x^2+13*x+3)+(x-2) )  
>${& ( (x^3-2*x^2-8)/(x+2) ), ${& solve( (x^3-2*x^2-8)/(x+2) )  
>${& ( (x^3-3*x+10)/(x-2) ), ${& solve( (x^3-3*x+10)/(x-2) )  
>${& ( (x^4-1)/(x-1) ), ${& solve( (x^4-1)/(x-1) )
```


BAB 3

KB Pekan 5-6 : Menggunakan EMT untuk menggambar grafik 2 dimensi (2D)

Nama : Dassy Chintya Nur Feika

NIM : 22305141023

Kelas : Matematika E 2022

Menggambar Grafik Fungsi Satu Variabel dalam Bentuk Ekspresi

Langsung

Menggambar Grafik Fungsi Satu Variabel dalam Bentuk Ekspresi Langsung Ekspresi tunggal

Di dalam program numerik EMT, ekspresi adalah string. Jika ditandai sebagai simbolis, mereka akan mencetak melalui Maxima, jika tidak melalui EMT. Ekspresi dalam string digunakan untuk membuat plot dan banyak fungsi numerik. Untuk ini, variabel dalam ekspresi harus "x".

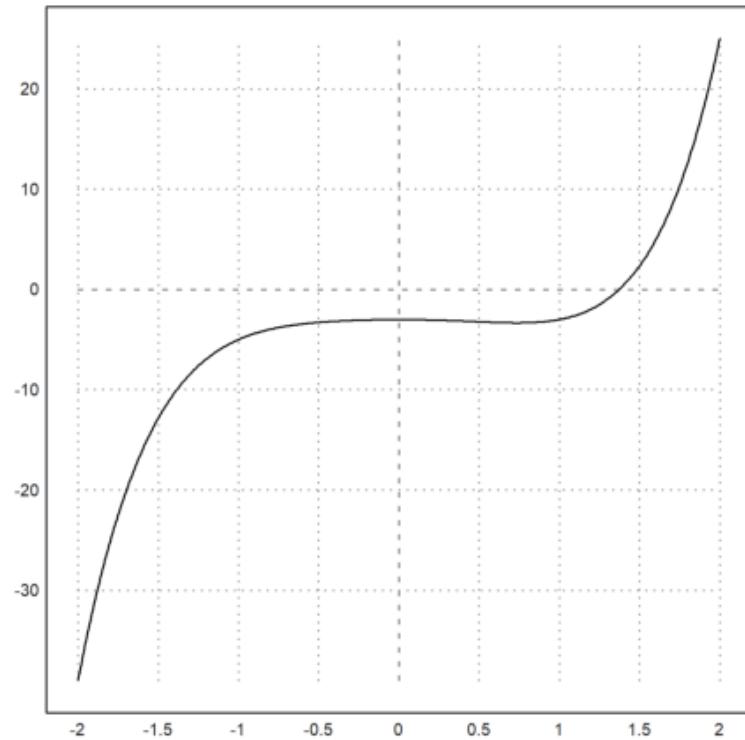
ekspresi dalam string

```
>expr := "x^5-x^2-3"
```

x^5-x^2-3

plot ekspresi

```
>plot2d(expr,-2,2) :
```

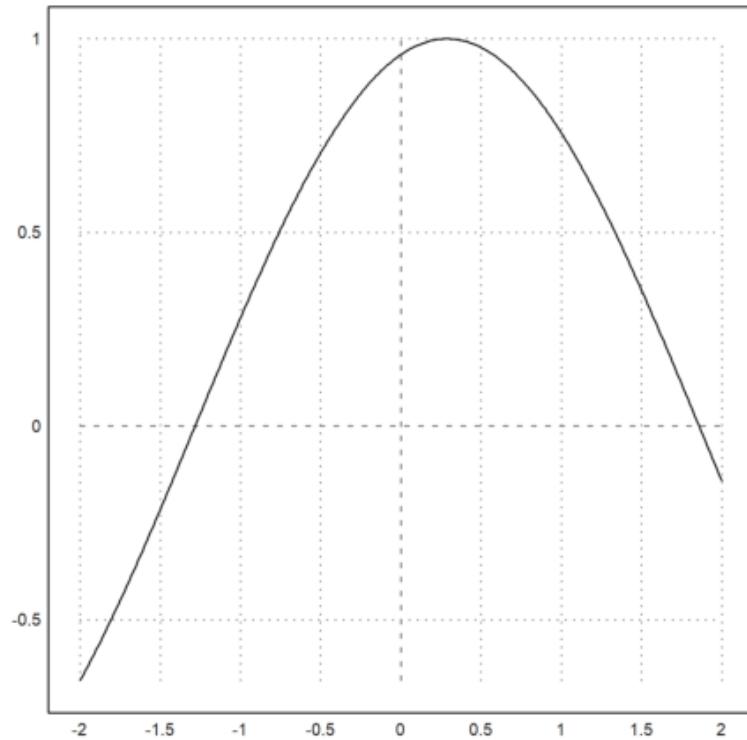


contoh 1

```
>expr := "sin (x-5)"
```

$\sin (x-5)$

```
>aspect (1) ; plot2d(expr,-2,2):
```



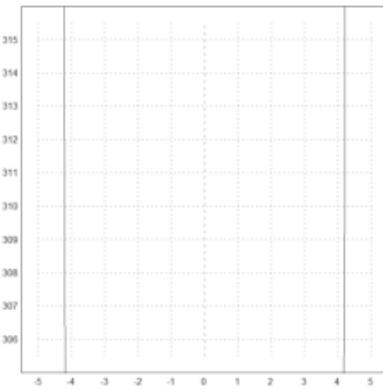
contoh 2 dan penggunaan grid

```
>aspect(1)plot2d("log(x) + 3", -0.1, 2, grid=6):
```

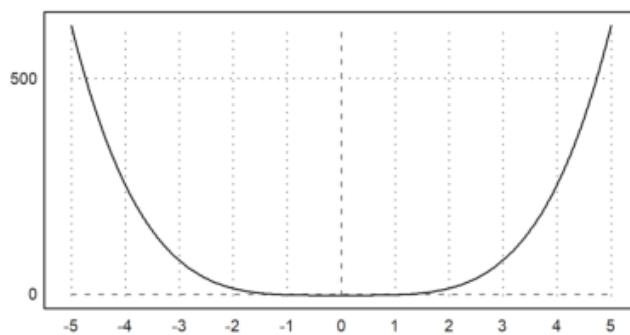
Commands must be separated by semicolon or comma!
 Found: plot2d("log(x) + 3", -0.1, 2, grid=6): (character 112)
 You can disable this in the Options menu.
 Error in:
 aspect(1)plot2d("log(x) + 3", -0.1, 2, grid=6): ...
 ^

contoh 3 dan penggunaan parameter square (atau >square) untuk memilih y-range secara otomatis

```
>aspect(1,1); plot2d("x^4-2", -5, 5, >square); insimg(15)
```

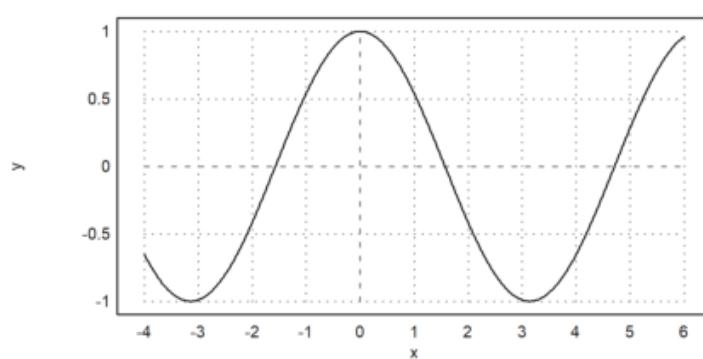


```
>aspect(2) ; plot2d("x^4-2", -5, 5) :
```



contoh 4 dan memberikan nama atau label pada garis sumbu

```
>plot2d("cos(x)", -4, 6, xl="x", yl="y") :
```



Menggambar Grafik Fungsi Satu Variabel yang Rumusnya Disimpan dalam

Variabel Ekspresi

Menggambar Grafik Fungsi Satu Variabel yang rumusnya Disimpan dalam Variabel Ekspresi

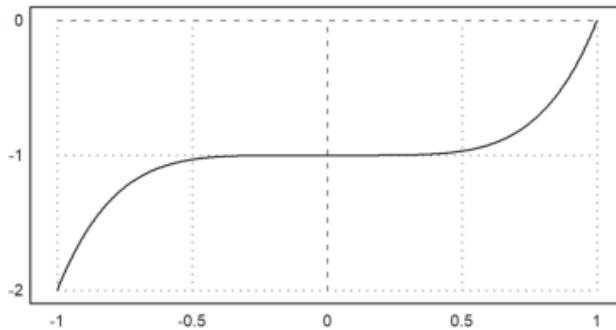
ekspresi

```
>expr &= x^5-1
```

$$x^5 - 1$$

plot dari ekspresi diatas

```
>aspect(2); plot2d(expr,-1,1):
```

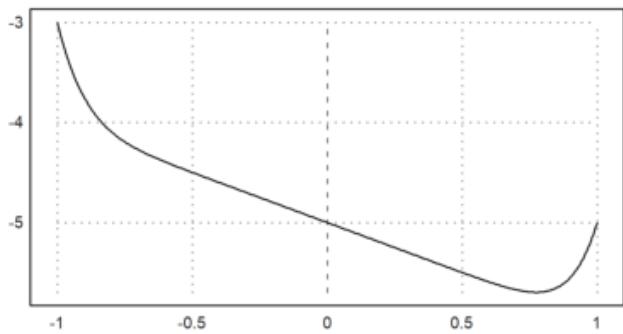


contoh 1

```
>expr := "x^10-x-5"
```

$x^{10}-x-5$

```
>aspect(2) ; plot2d(expr,-1,1):
```



menggunakan variabel lokal

Ekspresi dapat dievaluasi secara numerik. Variabel x,y,z ditetapkan secara otomatis. Variabel lain dapat ditetapkan berdasarkan parameter yang ditetapkan(variabel lokal) atau melalui variabel global. variabel global adalah variabel yang selalu bisa diakses kapan pun dan di mana pun.

```
>expr &= a*x^5
```

$$a \ x^5$$

menggunakan variabel global

```
>a=6; expr(2.5)
```

585.9375

menggunakan variabel lokal

```
>expr(2.5, a=6)
```

585.9375

evaluasi langsung

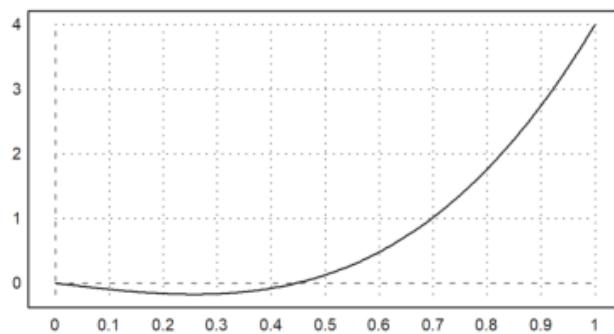
```
>"a*x^5"(3, 4)
```

1458

Oleh karena itu, banyak algoritma EMT yang dapat menggunakan ekspresi dalam x , bukan fungsi. Namun jika parameter tambahan yang tidak bersifat global dilibatkan, fungsi harus diutamakan.

menggunakan variabel global "a"

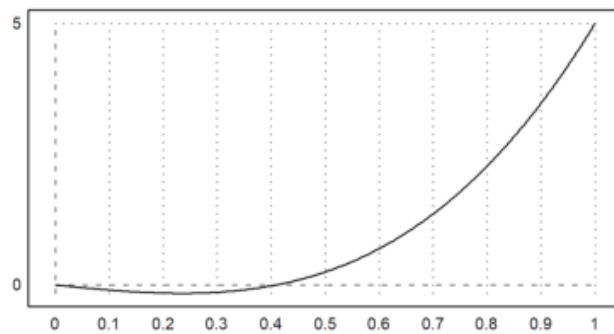
```
>a=5; plot2d("a*x^3-x", 0, 1);
```



```
>function f(x,a) := a*x^3-x
```

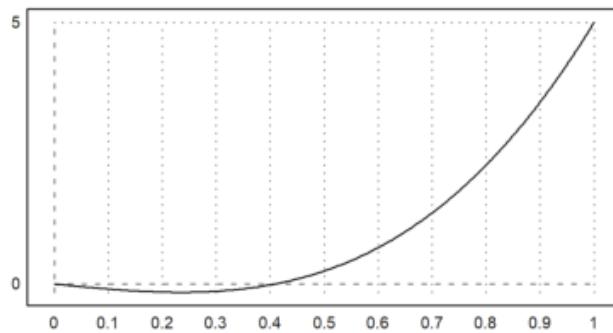
gunakan "a=6" sebagai parameter

```
>plot2d("f", 0, 1; 6);
```



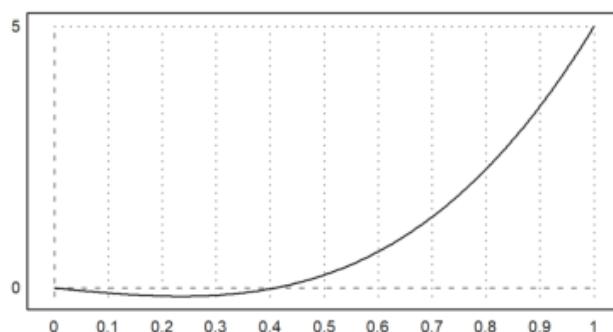
alternatif lain

```
>plot2d({{"f", 6}}, 0, 1) :
```



alternatif lain

```
>plot2d("f(x, 6)", 0, 1) :
```



Menggambar Grafik Fungsi Satu Variabel yang Fungsinya Didefinisikan

sebagai Fungsi Numerik

Menggambar Fungsi Simbolik

Fungsi Plot yang paling penting untuk plot planar adalah plot2d(). Fungsi ini diimplementasikan dalam bahasa Euler dalam file "plot.e", yang dimuat diawal program.

plot2d() menerima ekspresi, fungsi, dan data.

Rentang plot diatur dengan parameter yang ditetapkan sebagai berikut

- a,b: rentang x (default -2,2)

- -c,d: rentang y (default: skala dengan nilai)

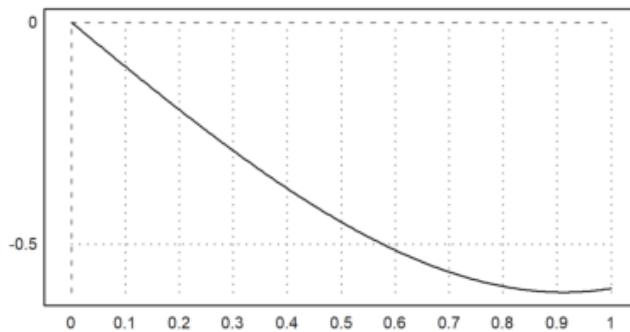
- r: alternatifnya radius di sekitar pusat plot
- cx,cy: koordinat pusat plot (default 0,0)

Keterangan:(menggambar grafik fungsi satu variabel yang fungsinya didefinisikan sebagai fungsi simbolik)

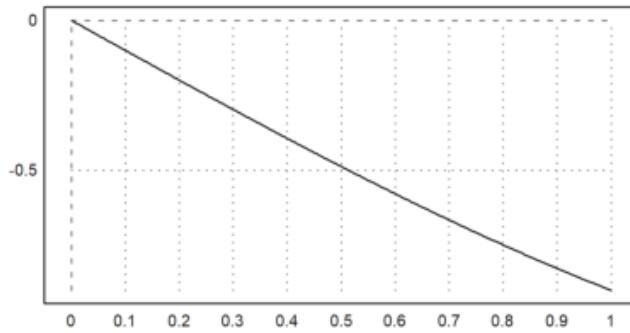
- &: untuk menampilkan variabel pada teks

Berikut adalah beberapa contoh menggunakan fungsi. Seperti biasa di EMT, fungsi yang berfungsi untuk fungsi atau ekspresi lain, jadi kita dapat meneruskan parameter tambahan (selain x) yang bukan variabel global ke fungsi dengan parameter titik koma atau dengan koleksi panggilan.

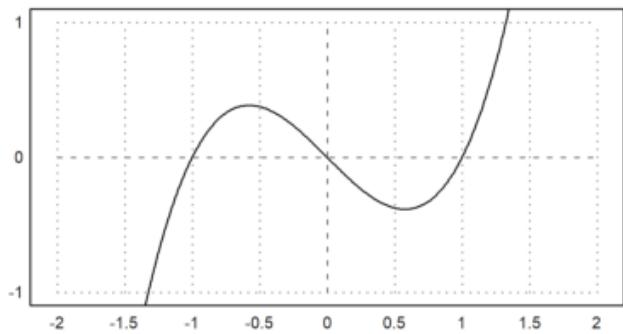
```
>plot2d("f",0,1;0.4): // plot with a=0.4
```



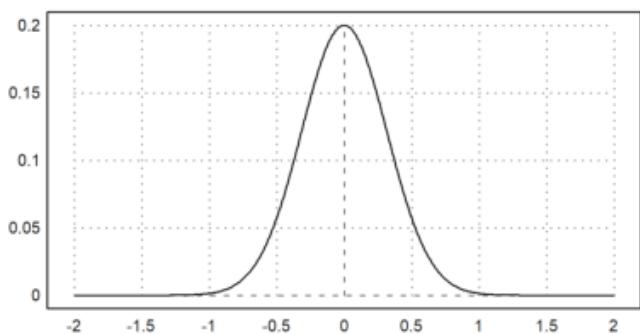
```
>plot2d({{"f",0.2}},0,1);
>plot2d({{"f(x,b)",b=0.1}},0,1):
```



```
>function f(x) := x^3-x;...
>plot2d("f",r=1):
```



```
>plot2d("exp(-a*x^2)/a"):
```



Berikut merupakan ringkasan dari fungsi yang diterima

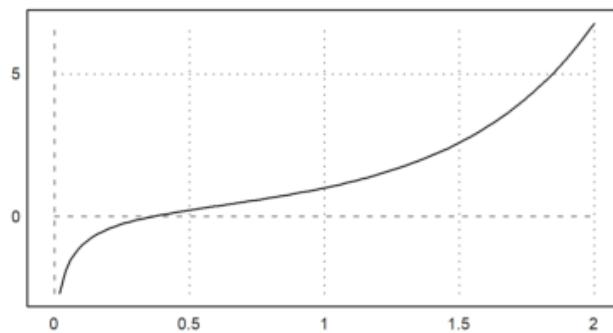
- ekspresi atau ekspresi simbolik dalam x
- fungsi atau fungsi simbolis dengan nama sebagai "f"
- fungsi simbolis hanya dengan nama f

Fungsi plot2d() juga menerima fungsi simbolis. Untuk fungsi simbolis, hanya nama saja yang berfungsi.

```
>function f(x) &= diff(x^x,x)
```

$$\frac{x^x}{x} (\log(x) + 1)$$

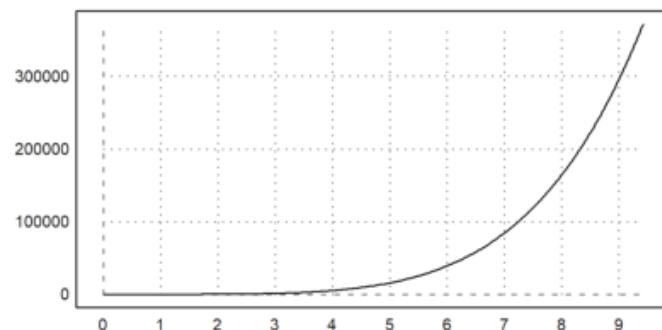
```
>plot2d(f, 0, 2):
```



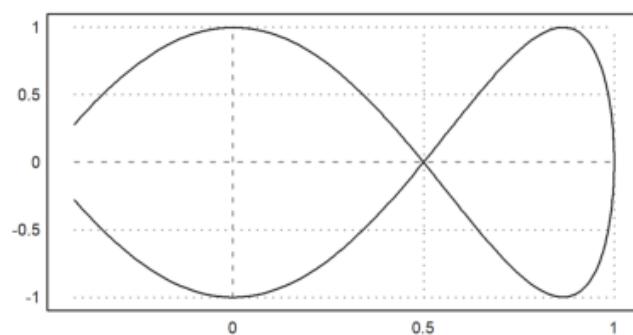
```
>$&expr = sin (x)*exp (-x)
```

$$a x^5 = e^{-x} \sin x$$

```
>plot2d(expr, 0, 3pi):
```



```
>plot2d("cos(x)", "sin(3*x)":
```



Menggambar Fungsi Numerik

Fungsi Numerik adalah sebuah fungsi dengan himpunan bilangan cacah sebagai domain dan himpunan mendasar yang melibatkan hubungan matematis antara bilangan yang menjadi domain dan bilangan sebagai kodomain.

>

Fungsi numerik memiliki 1 atau lebih variabel independen, yang sering dilambangkan sebagai "X". Variabel X adalah nilai atau parameter yang dapat berubah, dan fungsi numerik menggambarkan bagaimana variabel ini memengaruhi variabel dependen. Variabel dependen adalah hasil perhitungan atau keluaran dari fungsi numerik yang bergantung pada nilai atau perubahan dalam variabel independen.

Dalam EMT cara mendefinisikan fungsi menggunakan syntax function. untuk mendefinisikan fungsi numerik menggunakan tanda ":"=

Fungsi numerik menjelaskan bagaimana bilangan dalam domain berhubungan dengan bilangan sebagai kodomain, biasanya diberikan dalam bentuk rumus matematik(persamaan) atau aturan yang memetakan setiap domain kedalam kodomain yang sesuai. contoh:

$$f(x)=2x+3$$

>

(x)(variabel dependen) adalah fungsi yang memetakan setiap nilai x(variabel independen)kedalam nilai $2x+3$. Terdapat berbagai jenis fungsi yang termasuk ke dalam fungsi numerik, diantaranya:

Fungsi linier dengan bentuk umum

$$f(x) = ax + b$$

Fungsi kuadrat dengan bentuk umum

$$f(x) = ax^2 + bx + c$$

Fungsi eksponensial dengan bentuk umum

$$f(x) = ax$$

Fungsi logaritma dengan bentuk umum

$$f(x) = \log a(x)$$

Fungsi trigonometri dengan bentuk umum

$$f(x) = \sin(x), f(x) = \cos(x)$$

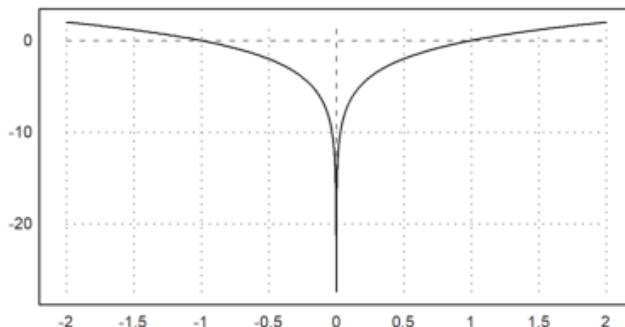
Salah satu cara yang umum digunakan untuk memvisualisasikan fungsi numerik adalah dengan menggambar grafiknya. Grafik ini menggambarkan bagaimana variabel dependen berubah seiring perubahan variabel independen dan membantu dalam memahami sifat-sifat fungsi, seperti titik ekstrim

Contoh soal

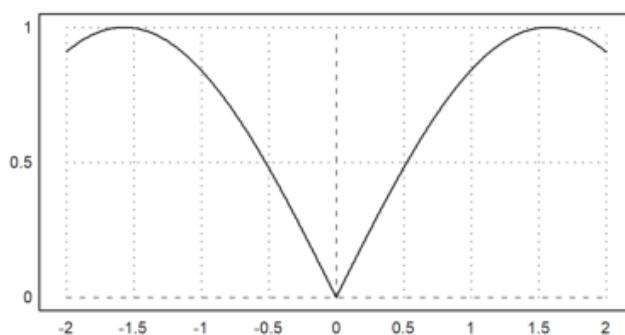
```
>function r(x) := abs(x-10)
>function s(x) := abs(sin(x))
>r(-5)
```

15

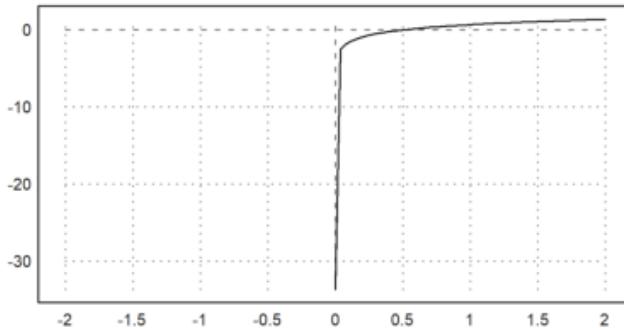
```
>function t(x) := log(x*(2+sin(x/1000)))
>function u(x) := integrate("(sin(x)*exp(-x^2))", 0, x)
>function v(x) := logbase((x^2), 2)
>plot2d("v"):
```



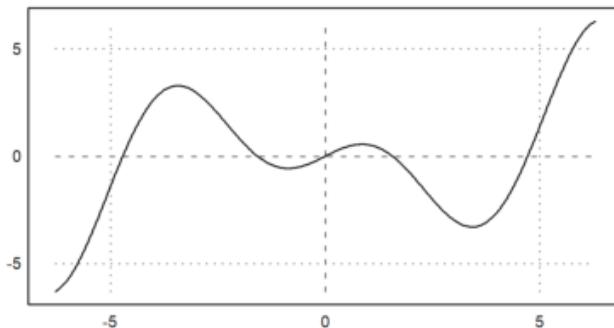
```
>plot2d("s"):
```



```
>plot2d("t", -2, 2):
```



```
>function P(x):=x*cos(x)
>plot2d("P", -2*pi, 2*pi):
```



Fungsi `plot2d()` adalah fungsi serbaguna untuk membuat grafik dalam bidang (grafik 2D). Fungsi ini dapat digunakan untuk membuat grafik fungsi-fungsi satu variabel, grafik data, kurva-kurva dalam bidang, grafik batang (bar plots), grid dari bilangan kompleks, dan grafik implisit dari fungsi dua variabel.

Parameter

`x,y` : persamaan, fungsi, atau vektor data `a,b,c,d` : area plot (default `a=-2, b=2`)

`r` : jika `r` diatur, maka `a=cx-r, b=cx+r, c=cy-r, d=cy+r` `r` bisa berupa vektor `[rx,ry]` atau vektor `[rx1,rx2,ry1,ry2]`. `xmin,xmax` : rentang parameter untuk kurva

`auto` : tentukan rentang `y` secara otomatis (default)

`square` : jika benar, mencoba menjaga rentang `x-y` tetap persegi `n` : jumlah interval (default adalah adaptif)

`grid` : `0` = tanpa grid dan label, `1` = hanya sumbu,

`2` = grid normal (lihat di bawah untuk jumlah garis grid) `3` = di dalam sumbu

`4` = tanpa grid

`5` = grid penuh termasuk margin `6` = tanda di pinggiran

7 = hanya sumbu

8 = hanya sumbu, sub-ticks frame : 0 = tanpa bingkai

framecolor: warna bingkai dan grid

margin : angka antara 0 dan 0,4 untuk margin di sekitar plot color : Warna kurva. Jika ini adalah vektor warna, akan digunakan untuk setiap baris matriks plot. Dalam hal grafik titik, harus berupa vektor kolom. Jika vektor baris atau matriks penuh warna digunakan untuk grafik titik, akan digunakan untuk setiap titik data.

thickness : ketebalan garis untuk kurva

Nilai ini dapat lebih kecil dari 1 untuk garis yang sangat tipis.

style: Gaya plot untuk garis, penanda, dan isian.

Untuk titik gunakan

"[]", "<>", ".", "..", "...", "*", "+", "", "-", "o"

"[]", "<>", "o" (bentuk terisi)

"[]w", "<>w", "ow" (tidak transparan)

Untuk garis gunakan

"-", "--", "-.", ".-", "-.-", "->"

Untuk poligon terisi atau plot batang gunakan

"", "O", "O", "/", "", "/","+", " ", "-","t"

points : plot titik tunggal sebagai gantinya garis segmen addpoints : jika benar, plot segmen garis dan titik

add : tambahkan plot ke plot yang ada

user : aktifkan interaksi pengguna untuk fungsi delta : ukuran langkah untuk interaksi pengguna

bar : plot batang (x adalah batas interval, y adalah nilai interval) histogram : plot frekuensi x dalam n subinterval

distribusi=n : plot distribusi x dengan n subinterval even : gunakan nilai antar untuk histogram otomatis. steps : plot fungsi sebagai fungsi langkah (steps=1,2)

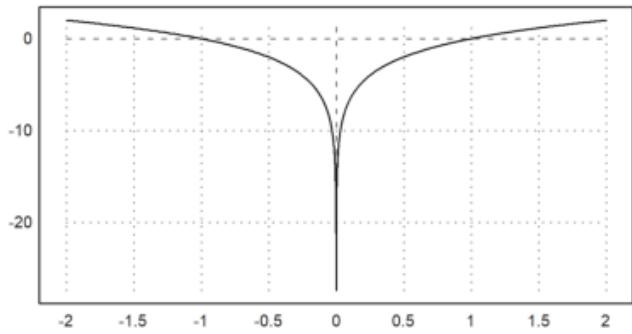
adaptive : gunakan plot adaptif (n adalah jumlah minimal langkah) level : plot garis level dari fungsi implisit dua variabel

outline : menggambar batas rentang level.

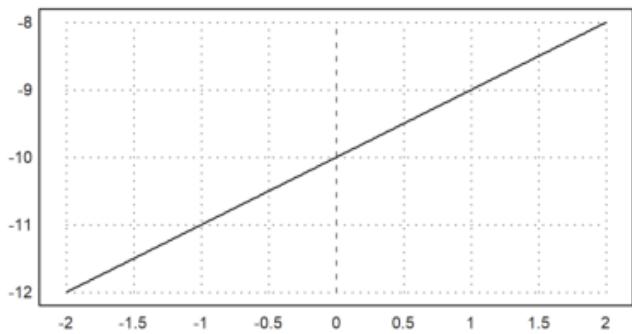
```
>function s(x):=(x-10)
>function r(x):=abs(sin(x))
>s(-5)
```

-15

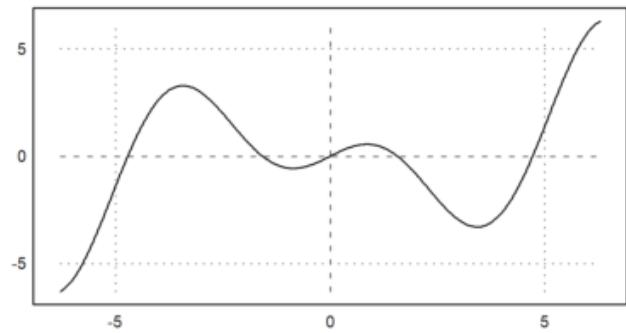
```
>function t(x):=log(x*(2+sin(x/1000)))
>function u(x):=integrate("(sin(x)*exp(-x^2))", 0, x)
>function v(x):=logbase((x^2), 2)
>plot2d("v"):
```



```
>plot2d("s"):
```



```
>function P(x):=x*cos(x)
>plot2d("P", -2*pi, 2*pi):
```



Sub Bab 5

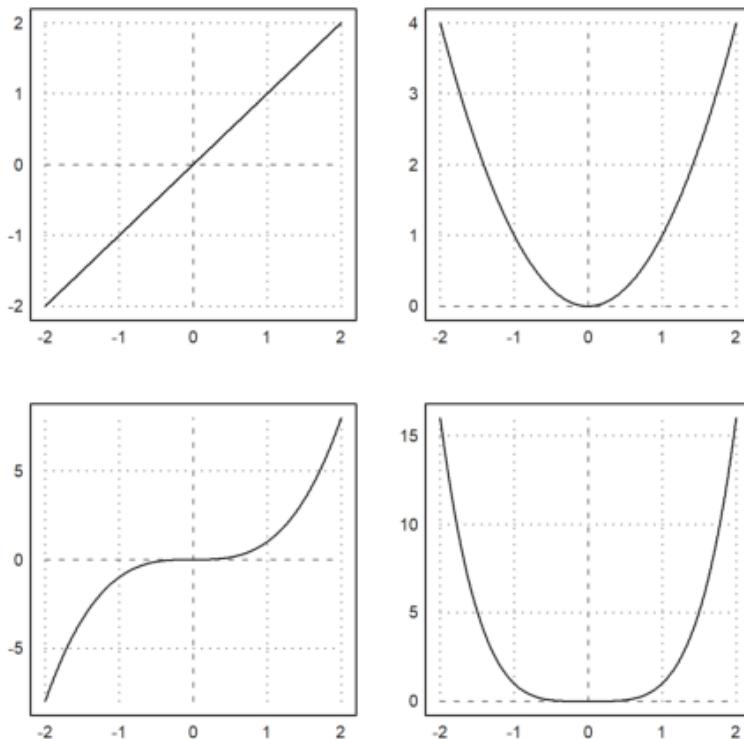
Menggambar Beberapa Kurva Sekaligus

Dalam subtopik ini, kita akan membahas mengenai cara menggambar beberapa kurva sekaligus. Dalam hal ini kita dapat menggambar beberapa kurva dalam jendela grafik yang berbeda secara bersama-sama. Untuk membuat ini kita dapat menggunakan perintah figure(). Berikut contoh dari menggambar beberapa kurva sekaligus

Menggambar plot fungsi

$$x^n, 1 \leq n \leq 4$$

```
>reset;
>figure(2,2);...
>for n=1 to 4; figure(n); plot2d("x^"+n); end;...
>figure(0):
```



Penjelasan sintaks dari plot fungsi

$$x^n, 1 \leq n \leq 4$$

- reset;

Perintah ini berguna untuk menghapus grafik yang telah ada sebelumnya, sehingga kita dapat memulai dari awal untuk menggambar grafik

- figure(2x2);

Perintah figure() digunakan untuk membuat jendela grafik dengan ukuran axb. Dalam kasus ini perintah figure(2,2) memiliki makna bahwa jendela grafik yang dibuat berukuran 2x2. Artinya, akan ada empat jendela grafik yang akan ditampilkan dengan tata letak 2 baris dan 2 kolom.

- for n=1 to 4;

Perintah ini digunakan untuk melakukan pengulangan (looping) perintah sebanyak empat kali, yaitu dari 1 hingga 4.

- figure(n);

Perintah ini digunakan untuk beralih dari jendela grafik satu ke jendela grafik lainnya (jendela grafik ke-n).

- plot2d("xⁿ");

Perintah plot2d() digunakan untuk membuat plot fungsi matematika.

Dalam hal ini fungsi yang diplot adalah x^n , di mana n adalah nilai dari variabel yang sedang diulang. Dengan kata lain, ini akan membuat plot dari x^1, x^2, x^3 , dan x^4 dalam jendela grafik yang sesuai

- end;

Perintah ini menandakan akhir dari looping.

- figure(0);

Perintah ini digunakan untuk beralih kembali ke jendela grafik utama.

Dari sini dapat kita perhatikan untuk membuat kurva fungsi x^n (x pangkat n) perintahnya tidak ditulis dengan (x^n) melainkan ditulis dengan $("x^n")$. Tanda petik dua ("...") digunakan untuk mengidentifikasi bahwa teks tersebut merupakan ekspresi matematika.

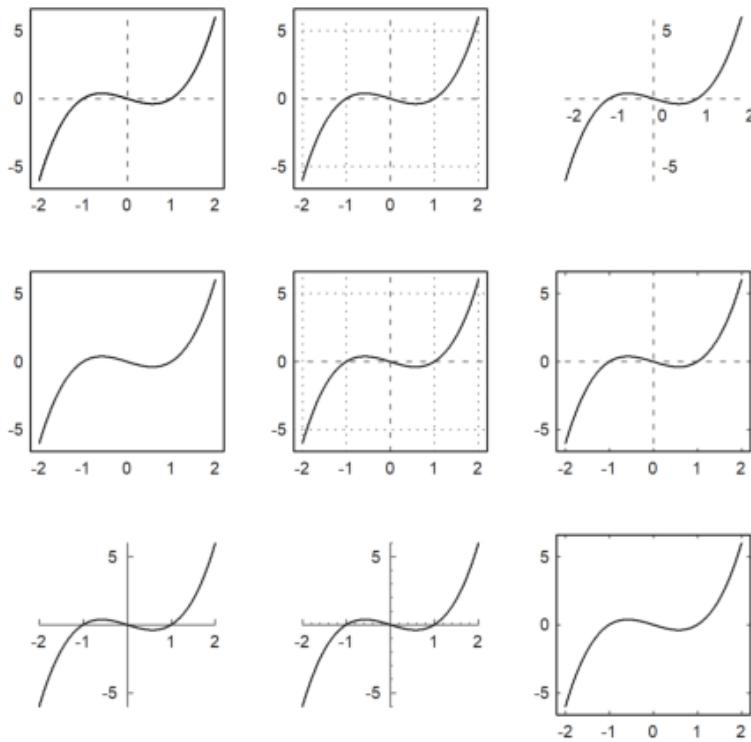
Sedangkan tanda (+) digunakan untuk menggabungkan string dengan nilai yang berubah-ubah atau variabel.

Contoh lain:

Menggambar plot fungsi

$$f(x) = x^3 - x, -2 < x < 2$$

```
>reset;
>figure(3,3);...
>for k=1:9; figure(k); plot2d("x^3-x", -2, 2, grid=k); end;...
>figure(0):
```



Penjelasan sintaks dari plot fungsi

$$f(x) = x^3 - x, -2 < x < 2$$

- reset;

Perintah ini berguna untuk menghapus grafik yang telah ada sebelumnya, sehingga kita dapat memulai dari awal untuk menggambar grafik

- figure (3,3);

Perintah ini digunakan untuk membuat jendela grafik dengan ukuran 3x3. Artinya, akan ada empat jendela grafik yang akan ditampilkan dengan tata letak 3 baris dan 3 kolom.

- for k=1:9;

Perintah ini digunakan untuk melakukan pengulangan (looping) perintah sebanyak sembilan kali.

- figure(n);

Perintah ini digunakan untuk beralih dari jendela grafik satu ke

jendela grafik lainnya (jendela grafik ke-n).

- plot2d("x^3-x",-2,2,grid=k);

Perintah plot2d() digunakan untuk membuat plot fungsi matematika.

Dalam hal ini fungsi yang diplot adalah $x^3 - x$, dengan batas sumbu x dari -2 hingga 2.

Argumen grid=k digunakan untuk mengaktifkan grid pada jendela grafik ke-k.

- end;

Perintah ini menandakan akhir dari looping.

- figure(0);

Perintah ini digunakan untuk beralih kembali ke jendela grafik utama.

Dari contoh diatas dapat kita perhatikan bahwa tampilan plot dari yang ke-1 hingga ke-9 memiliki tampilan yang berbeda-beda. Dalam EMT memiliki berbagai gaya plot 2D yang dapat dijalankan menggunakan perintah grid=n dimana n adalah jumlah langkah minimal. Setiap nilai n memiliki tampilan plot adaptif yang berbeda dalam plot 2D, diantaranya yaitu:

0 : tidak ada grid (kisi), frame, sumbu, dan label, hanya kurva saja

1 : dengan sumbu, label-label sumbu di luar frame jendela grafik

2 : tampilan default

3 : dengan grid pada sumbu x dan y, label-label sumbu berada di dalam jendela grafik

4 : tidak ada grid (kisi), sumbu x dan y, dan label berada di luar frame jendela grafik

5 : tampilan default tanpa margin di sekitar plot

6 : hanya dengan sumbu x y dan label, tanpa grid

7 : hanya dengan sumbu x y dan tanda-tanda pada sumbu.

8 : hanya dengan sumbu dan tanda-tanda pada sumbu, dengan tanda-tanda yang lebih halus pada sumbu.

9 : tampilan default dengan tanda-tanda kecil di dalam jendela

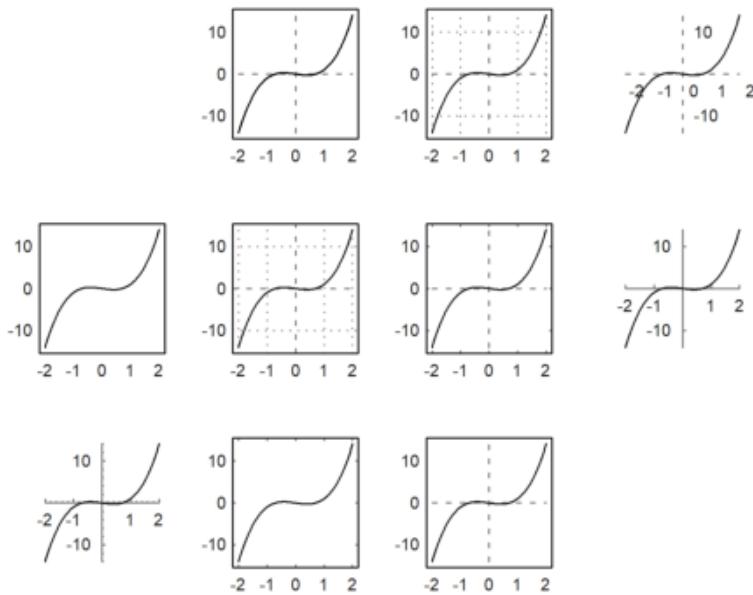
10: hanya dengan sumbu-sumbu, tanpa tanda

Contoh lain:

Menggambar plot fungsi

$$g(x) = 2x^3 - x$$

```
>reset;
>aspect(1.2);
>figure(3,4); ...
> figure(2); plot2d("2x^3-x",grid=1); ... // x-y-axis
> figure(3); plot2d("2x^3-x",grid=2); ... // default ticks
>figure(4); plot2d("2x^3-x",grid=3); ... // x-y- axis with labels inside
> figure(5); plot2d("2x^3-x",grid=4); ... // no ticks, only labels
>figure(6); plot2d("2x^3-x",grid=5); ... // default, but no margin
>figure(7); plot2d("2x^3-x",grid=6); ... // axes only
>figure(8); plot2d("2x^3-x",grid=7); ... // axes only, ticks at axis
>figure(9); plot2d("2x^3-x",grid=8); ... // axes only, finer ticks at axis
>figure(10); plot2d("2x^3-x",grid=9); ... // default, small ticks inside
>figure(11); plot2d("2x^3-x",grid=10); ...// no ticks, axes only
>figure(0):
```



Penjelasan sintaks dari plot fungsi

$$g(x) = 2x^3 - x$$

- aspect(1.2);

Perintah aspect() digunakan untuk mengatur rasio aspek dari jendela grafik. Hal ini berarti perintah aspect(1.2); akan menghasilkan plot dengan perbandingan rasio panjang dan lebar 2:1.

- figure(3,4);

Perintah ini digunakan untuk membuat jendela grafik dengan ukuran 3x4.

Jadi, akan ada total 12 jendela grafik yang akan ditampilkan dalam tata letak 3 baris dan 4 kolom.

- figure(1); plot2d("x^3-x",grid=0); ...

Adalah perintah untuk beralih ke jendela grafik pertama dan menggambar plot dari fungsi $x^3 - x$ tanpa grid, frame, atau sumbu.

- figure(2); plot2d("x^3-x",grid=1); ...

Adalah perintah untuk beralih ke jendela grafik kedua dan menggambar plot dari fungsi $x^3 - x$ dengan grid hanya pada sumbu x dan y.

- figure(3); plot2d("x^3-x",grid=2); ...

Adalah perintah untuk beralih ke jendela grafik ketiga dan menggambar plot dari fungsi $x^3 - x$ dengan tampilan default, termasuk tanda-tanda default pada sumbu.

- figure(4); plot2d("x^3-x",grid=3); ...

Adalah perintah untuk beralih ke jendela grafik keempat dan menggambar plot dari fungsi $x^3 - x$ dengan grid pada sumbu x dan y, serta label-label sumbu yang ada di dalam jendela.

- figure(5); plot2d("x^3-x",grid=4); ...

Adalah perintah untuk beralih ke jendela grafik kelima dan menggambar plot dari fungsi $x^3 - x$ tanpa tanda-tanda sumbu, hanya label-label yang ada.

- figure(6); plot2d("x^3-x",grid=5); ...

Adalah perintah untuk beralih ke jendela grafik keenam dan menggambar plot dari fungsi $x^3 - x$ dengan tampilan default, tetapi tanpa margin di sekitar plot.

- figure(7); plot2d("x^3-x",grid=6); ...

Adalah perintah untuk beralih ke jendela grafik ketujuh dan menggambar plot dari fungsi $x^3 - x$ hanya dengan sumbu-sumbu (tanpa grid atau label).

- figure(8); plot2d("x^3-x",grid=7); ...

Adalah perintah untuk beralih ke jendela grafik kedelapan dan menggambar plot dari fungsi $x^3 - x$ hanya dengan sumbu-sumbu dan tanda-tanda pada sumbu.

- figure(9); plot2d("x^3-x",grid=8); ...

Adalah perintah untuk beralih ke jendela grafik kesembilan dan menggambar plot dari fungsi $x^3 - x$ hanya dengan sumbu-sumbu dan tanda-tanda pada sumbu, dengan tanda-tanda yang lebih halus pada sumbu.

- figure(10); plot2d("x^3-x",grid=9); ...

Adalah perintah untuk beralih ke jendela grafik kesepuluh dan menggambar plot dari fungsi $x^3 - x$ dengan tanda-tanda default kecil di dalam jendela.

- figure(11); plot2d("x^3-x",grid=10); ...

Adalah perintah untuk beralih ke jendela grafik kesebelas dan menggambar plot dari fungsi $x^3 - x$ hanya dengan sumbu-sumbu, tanpa tanda-tanda.

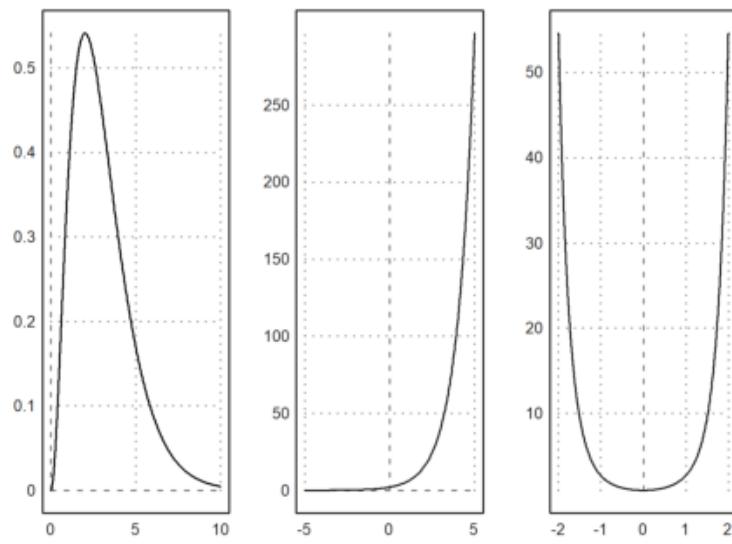
- figure(0);

Adalah perintah untuk beralih kembali ke jendela grafik utama atau jendela grafik dengan nomor 0 setelah semua perintah dalam urutan selesai dieksekusi.

Dari ketiga contoh di atas, dapat kita katakan bahwa untuk menggambar beberapa kurva sekaligus itu dapat dilakukan dengan satu baris perintah ataupun dengan cara mendefinisikannya 1 per 1.

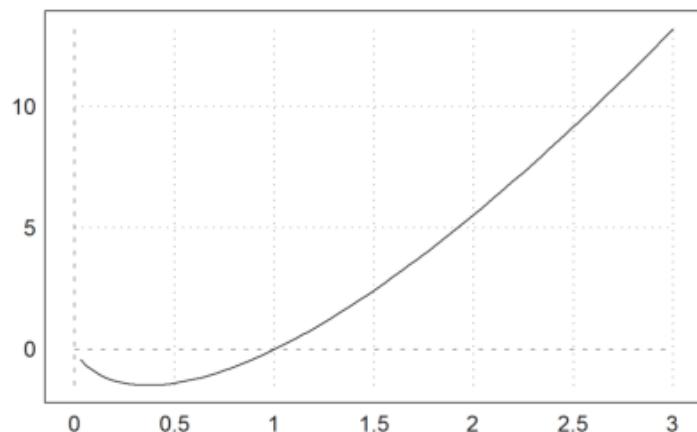
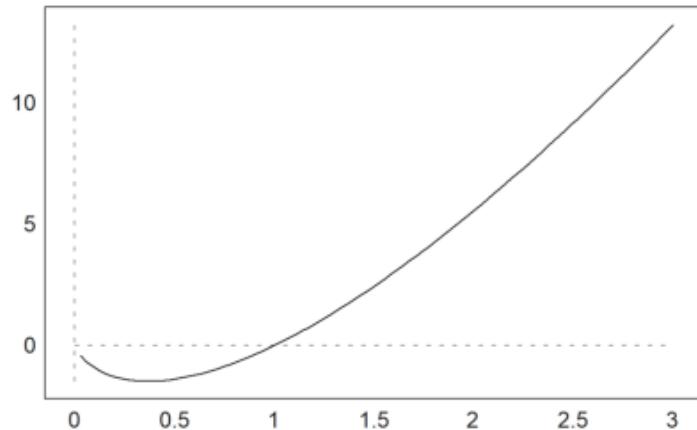
Terlihat beberapa jenis grid memiliki tampilan yang mirip atau sama, seperti 1 dan 2, 2 dan 5, 4 dan 9, 7 dan 8, untuk dapat membedakannya secara lebih jelas, ubah grid dari contoh di bawah ini.

```
>reset;
>aspect(1.3);
>figure(1,3);...
>figure(1); plot2d("x^2*exp(-x)",0,10);...
>figure(2); plot2d("2*exp(x)",-5,5);...
>figure(3); plot2d("exp(x^2)",-2,2);...
>figure(0):
```



Contoh lain:

```
>reset;
>aspect (3/4);
>figure(2,1);...
>for a=1:2; figure(a); plot2d("2*x*log(x^2)",0,3,grid=a); end;...
>figure(0):
```



Sub Bab 6

Menggambar Beberapa Kurva pada bidang koordinat yang sama

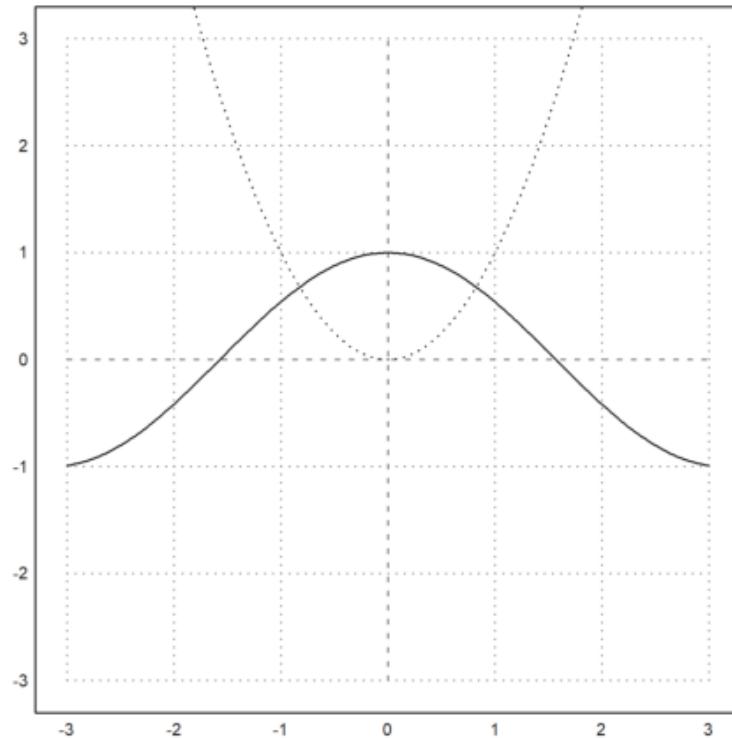
Plot lebih dari satu fungsi (multiple function) ke dalam satu jendela dapat dilakukan dengan berbagai cara. Salah satu caranya adalah menggunakan `>add` untuk beberapa panggilan ke `plot2d` secara keseluruhan, kecuali panggilan pertama.

Berikut contohnya:
menggambar kurva

$$f(x) = \cos(x)$$

$$f(x) = x^2$$

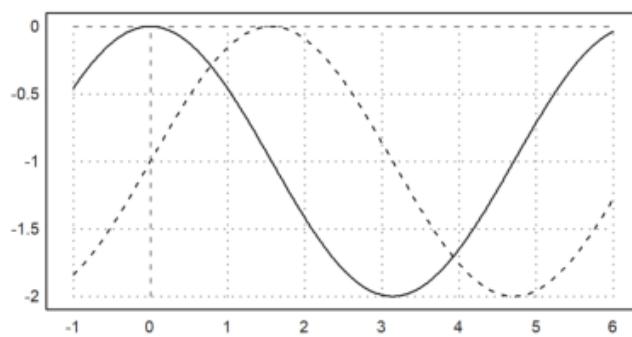
```
>aspect(); plot2d("cos(x)", r=3); plot2d("x^2", style=".",>add):
```



$$f(x) = \cos(x) - 1$$

$$f(x) = \sin(x)$$

```
>aspect(2); plot2d("cos(x)-1",-1,6); plot2d("sin(x)",style="--",>add):
```

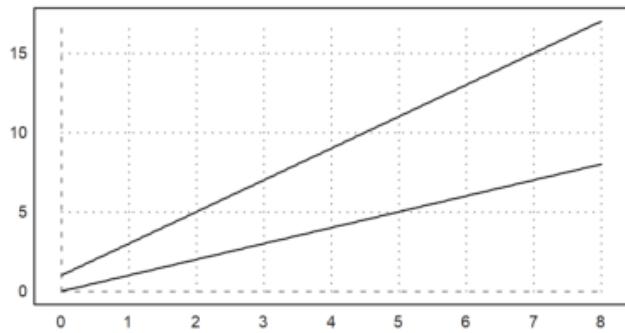


Selain menggunakan >add kita juga bisa menambahkannya secara langsung
Berikut contohnya:
Menggambar kurva

$$f(x) = 2x + 1$$

$$f(x) = -2x + 1$$

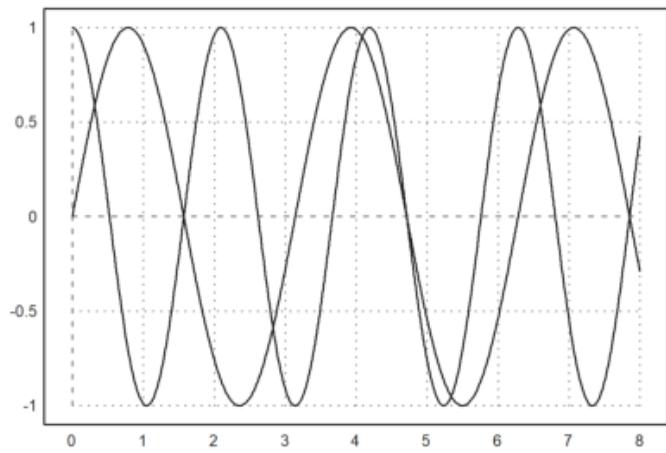
```
>plot2d(["2x+1", "x"], 0, 8):
```



$$f(x) = \sin(2x)$$

$$f(x) = \cos(3x)$$

```
>aspect(1.5); plot2d(["sin(2x)", "cos(3x)"], 0, 8):
```



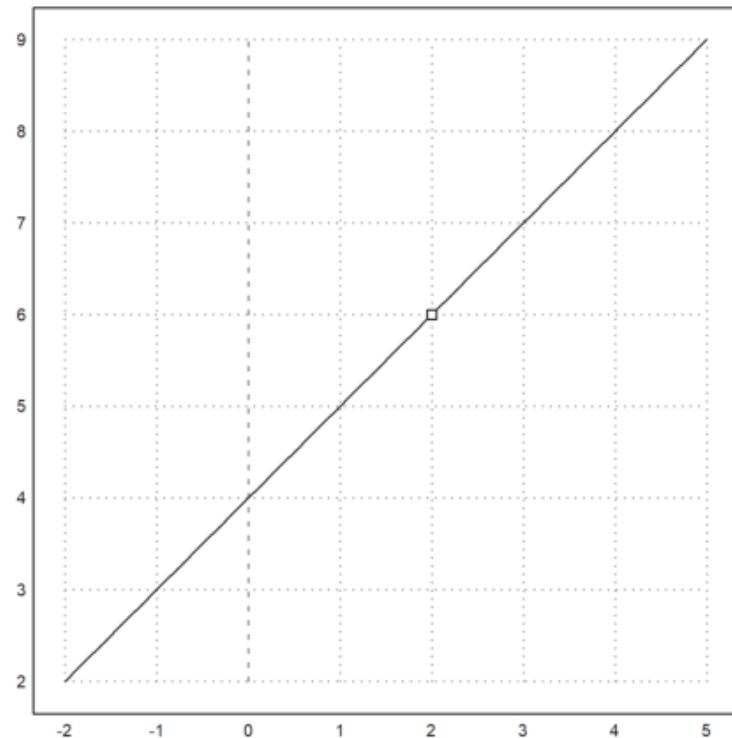
Kegunaan >add yang lain juga bisa untuk menambahkan titik pada kurva.

Berikut contohnya:

Menambahkan sebuah titik di

$$f(x) = x + 4$$

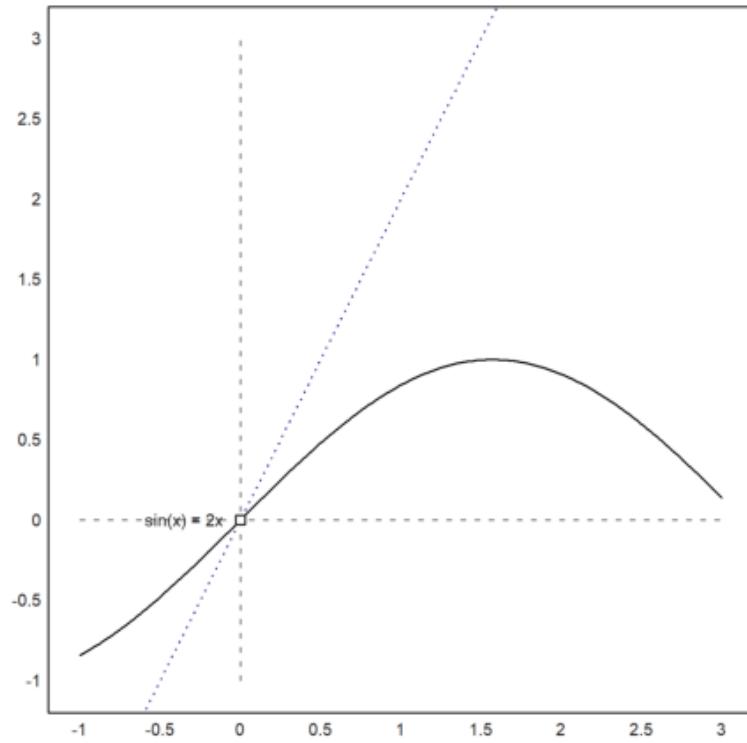
```
>aspect(); plot2d("x+4", -2, 5,); plot2d(2, 6, >points, >add):
```



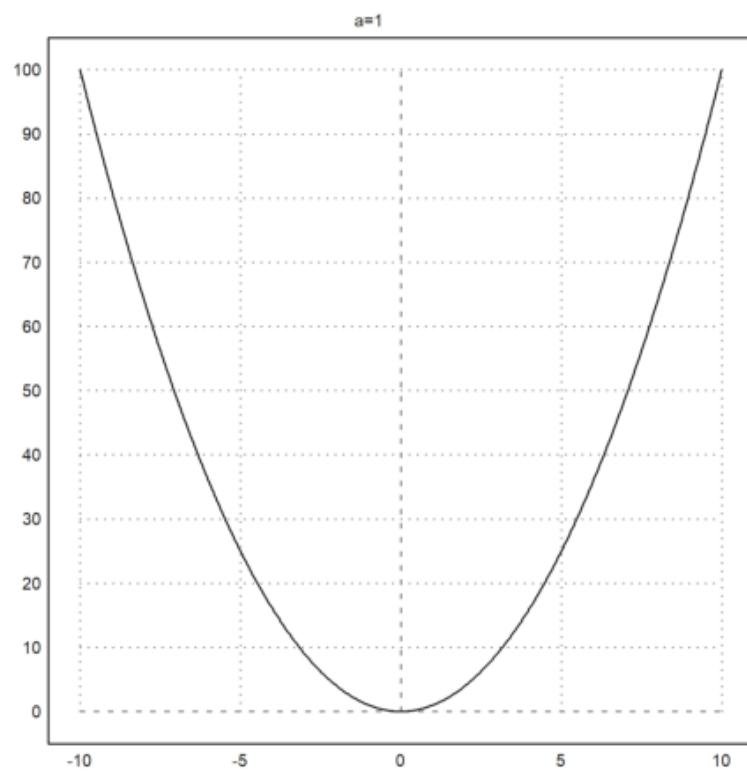
Kita juga bisa mencari titik perpotongan dengan cara berikut:

$$\sin(x) = 2x$$

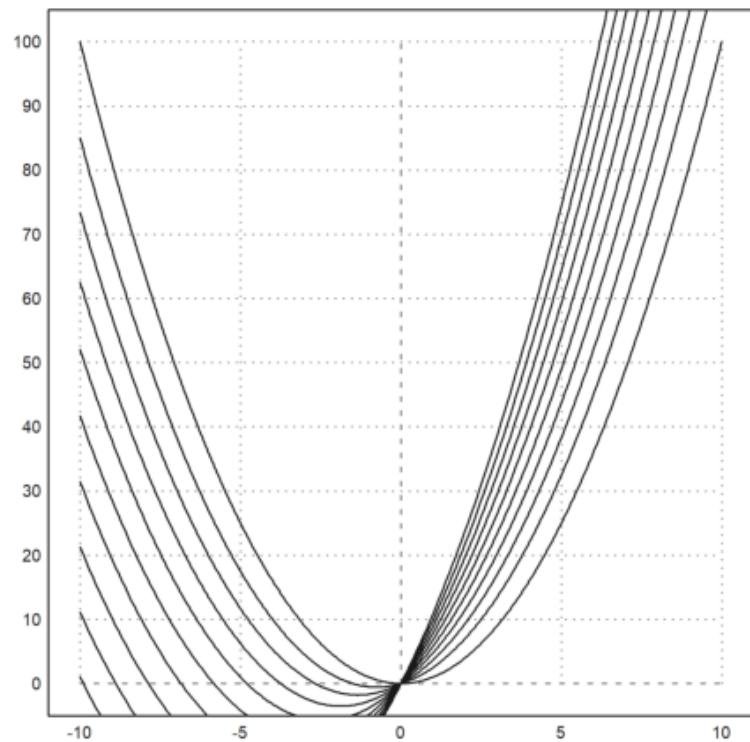
```
>plot2d(["sin(x)", "2x"], r=2, cx=1, cy=1, ...
> color=[black, blue], style=[ "-", ". "], ...
> grid=1);
>x0=solve("sin(x)-2x", 1); ...
> plot2d(x0, x0, >points, >add); ...
> label("sin(x) = 2x", x0, x0, pos="cl", offset=20):
```



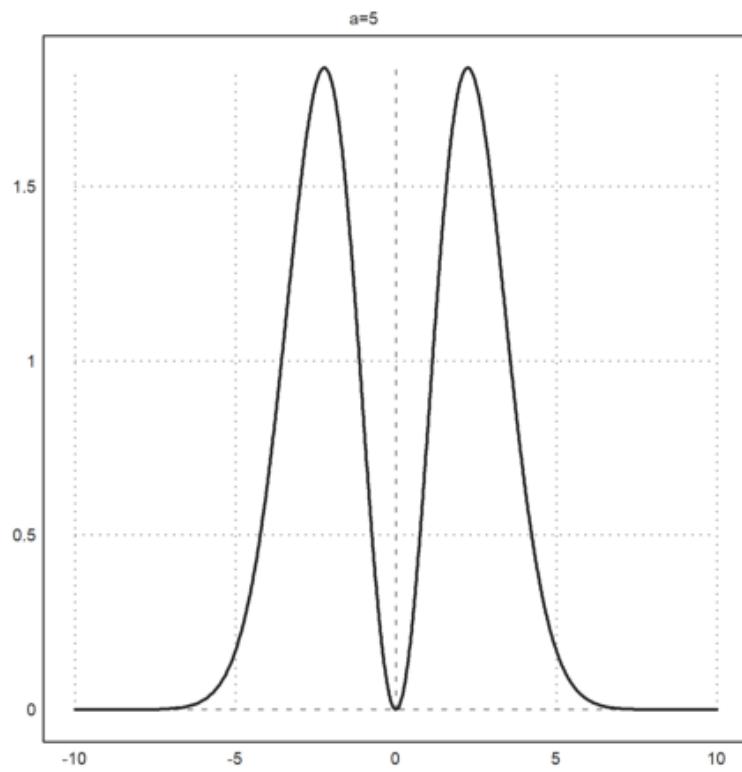
```
>function f(x,a) := x^2+a*x-x/a; ...
>plot2d("f",-10,10;1,title="a=1"):
```



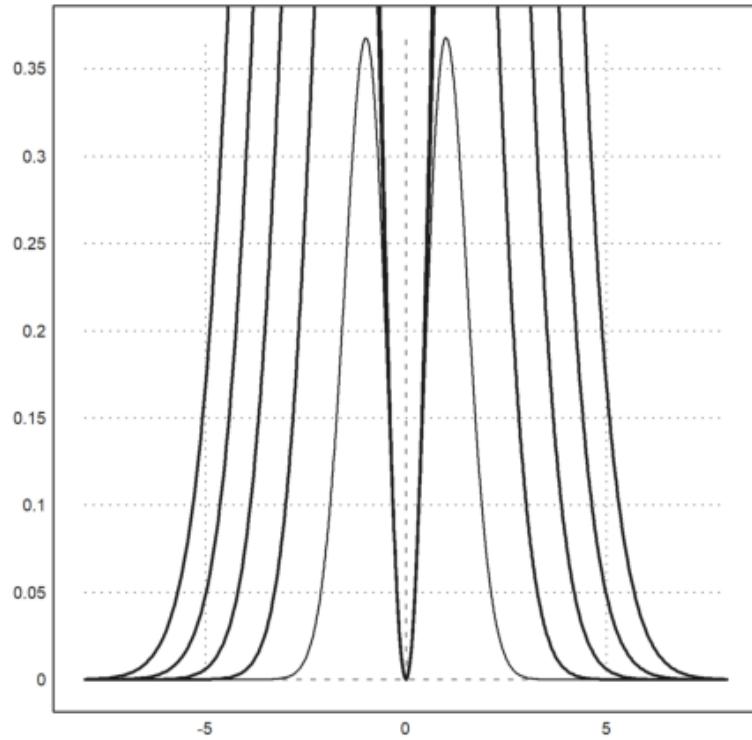
```
> plot2d({{"f",1}},-10,10); ...
>for a=1:10; plot2d({{"f",a}},>add); end;
```



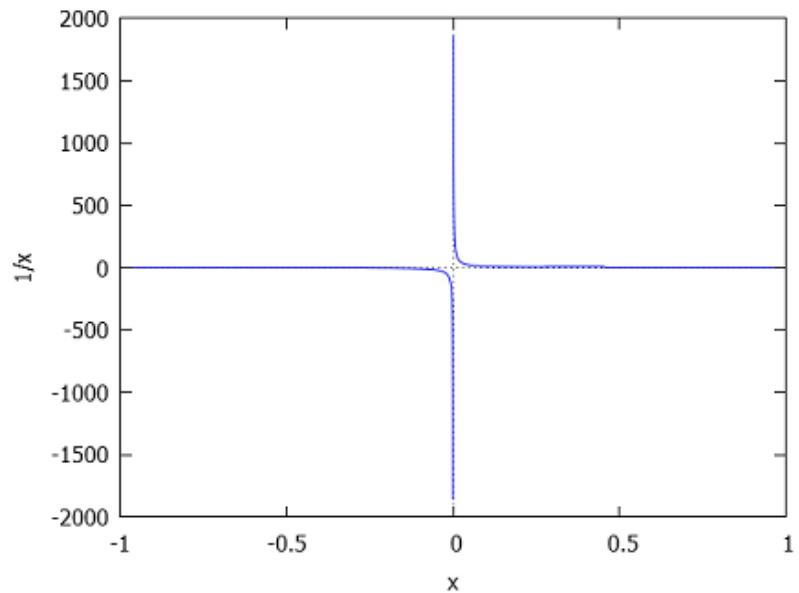
```
>function f(x,a) := x^2*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5");
```



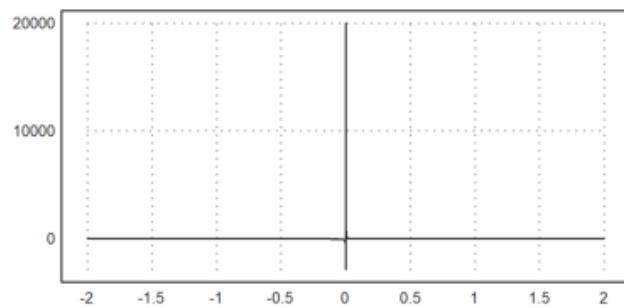
```
>plot2d({{"f",1}},-8,8); ...
>for a=2:5; plot2d({{"f",a}},>add,thickness=2); end:
```



```
>aspect(2.1); &plot2d(1/x, [x,-1,1]):
```



```
>x=linspace(-1,1,50);...
>plot2d("1/x"):
```

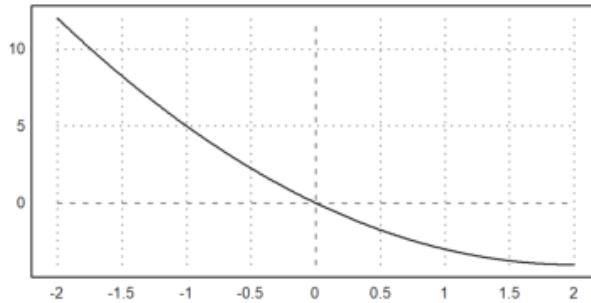


Sub Bab 7

Menuliskan Label koordinat, label kurva, dan keterangan

kurva(legend) Dalam EMT, untuk menambahkan judul dapat dilakukan dengan title="..." untuk menambahkan sumbu x dan sumbu y dapat dilakukan dengan x1="...", y1="..." sebagai contoh:

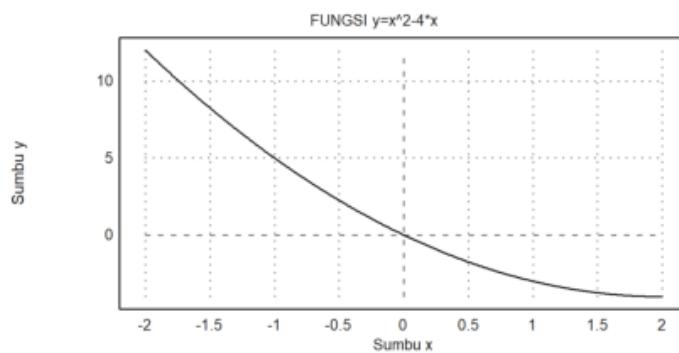
```
>plot2d("x^2-4*x") :
```



untuk menambahkan judul dapat dilakukan dengan title="..."

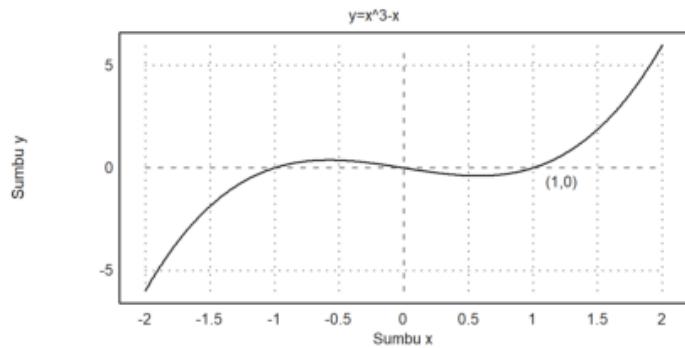
untuk menambahkan sumbu x dan sumbu y dapat dilakukan dengan x1="...", y1="..."

```
>plot2d("x^2-4*x",title="FUNGSI y=x^2-4*x",yl="Sumbu y",xl="Sumbu x") :
```



Selain itu juga dapat dengan cara lain seperti contoh berikut:

```
>expr := "x^3-x"; ...
> plot2d(expr,title="y="+expr,xl="Sumbu x",yl="Sumbu y"); ...
> label("(1,0)",1,0); label("Max",E,expr(E),pos="lc"):
```



Sub Bab 8

Mengatur ukuran gambar,format(style),dan warna kurva

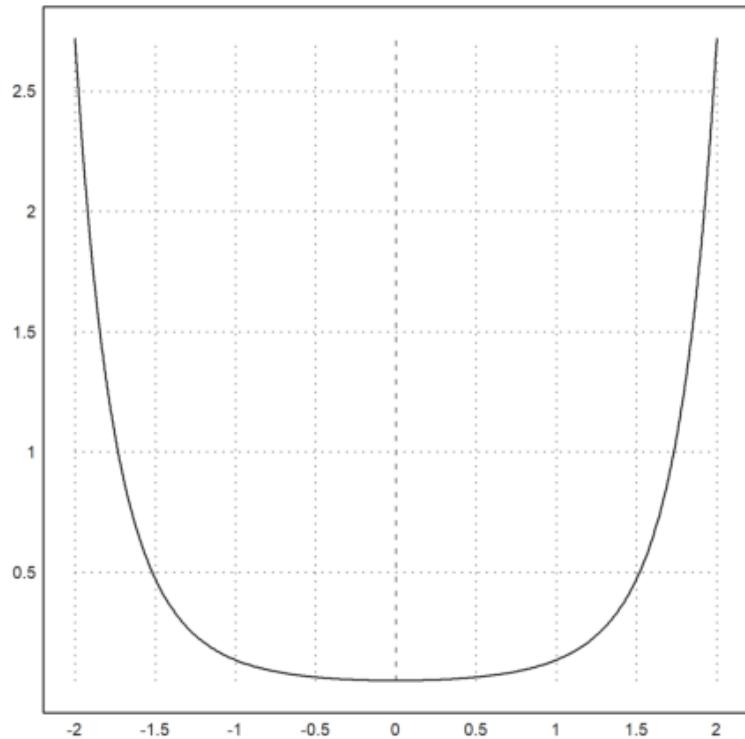
Untuk mengubah ukuran, dapat dilakukan dengan menggunakan aspect="...", semakin besar nilai aspect, maka ukuran kurva akan semakin kecil, begitupun sebaliknya

untuk mengganti style, dapat dipilih dengan berbagai pilihan
style="...", dapat dipilih dari, misal : "-", "_", "-.", "-.", "-.-".

untuk warna dapat dipilih sebagai salah satu warna default
color="...", warna default= red,green,blue,yellow, dll

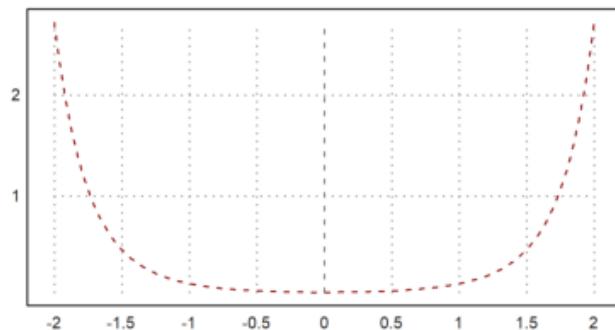
sebagai contoh:

```
>aspect(1); plot2d("exp(x^2-3)":
```



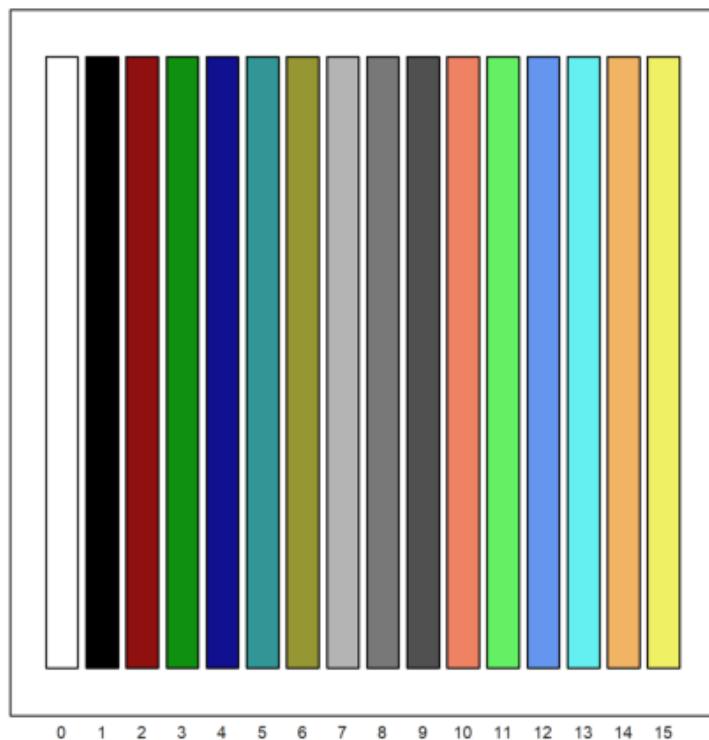
ukuran kurva dapat diganti dengan mengganti nilai aspect="...", semakin besar nilai aspect, maka ukuran kurva akan semakin kecil Untuk mengganti warna dapat ditambahkan dengan color="...", sedangkan untuk mengganti format(style) dapat dilakukan dengan menambahkan style="..."

```
>aspect(2); plot2d("exp(x^2-3)", color=red, style="--") :
```



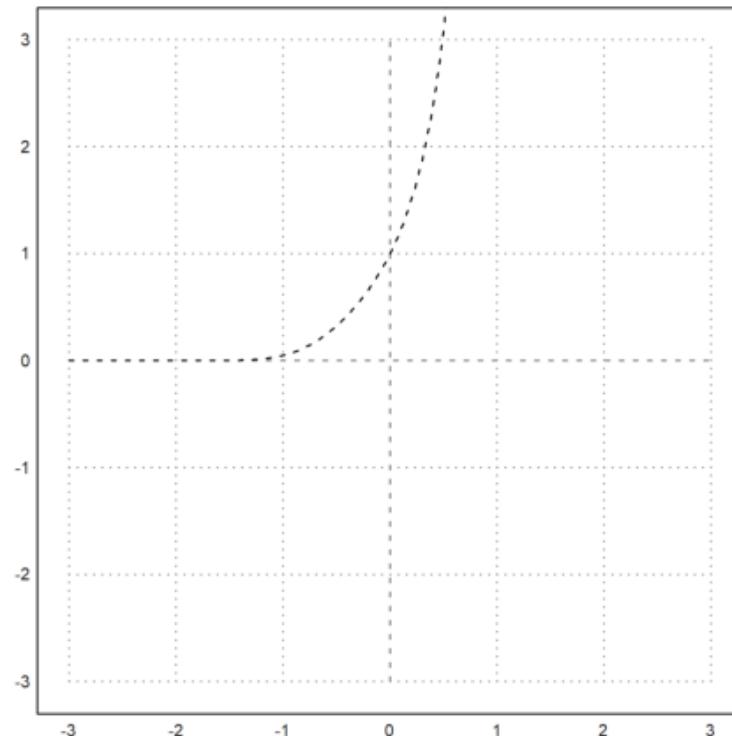
Berikut adalah tampilan warna EMT yang telah ditentukan

```
>aspect(1); columnsplot(ones(1,16), lab=0:15, grid=0, color=0:15) :
```



selain menggunakan warna default, untuk mengubah warna dapat juga dengan menggunakan kode warna di atas sebagai contoh:

```
>aspect(1); plot2d("exp(x^3+2*x)", r=3, color=1, style="--") :
```



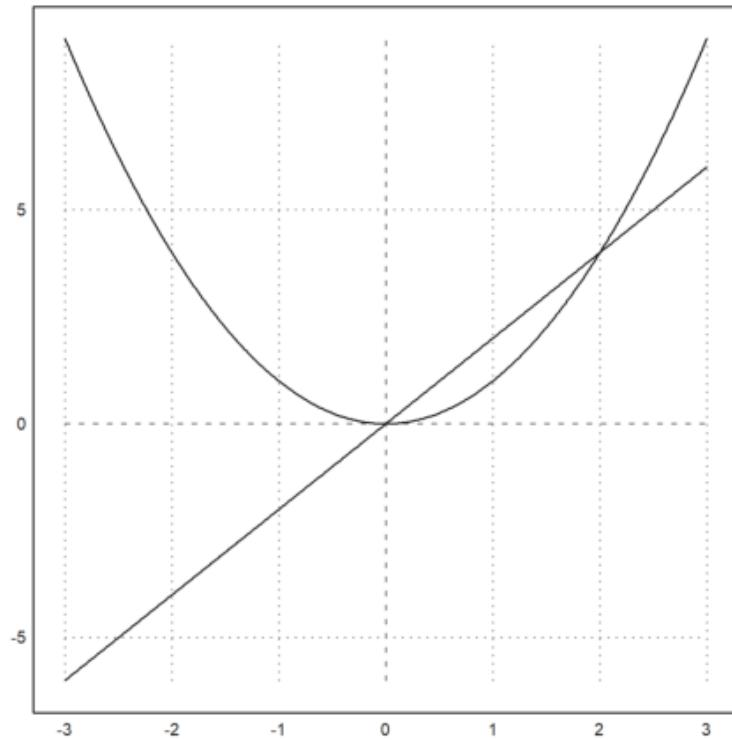
Sub Bab 9

Menggambar Sekumpulan Kurva dalam satu perintah plot2d.

Dalam pembahasan sub-bab 9 kali ini akan membahas mengenai bagaimana menggambar sekumpulan kurva dalam satu perintah plot2d. Menggambar sekumpulan kurva dalam satu perintah plot2d adalah teknik yang digunakan untuk memvisualisasikan beberapa fungsi dalam satu grafik. Ini memudahkan perbandingan antara beberapa kurva.

Contoh

```
>plot2d(["x^2", "2*x"], -3, 3) :
```



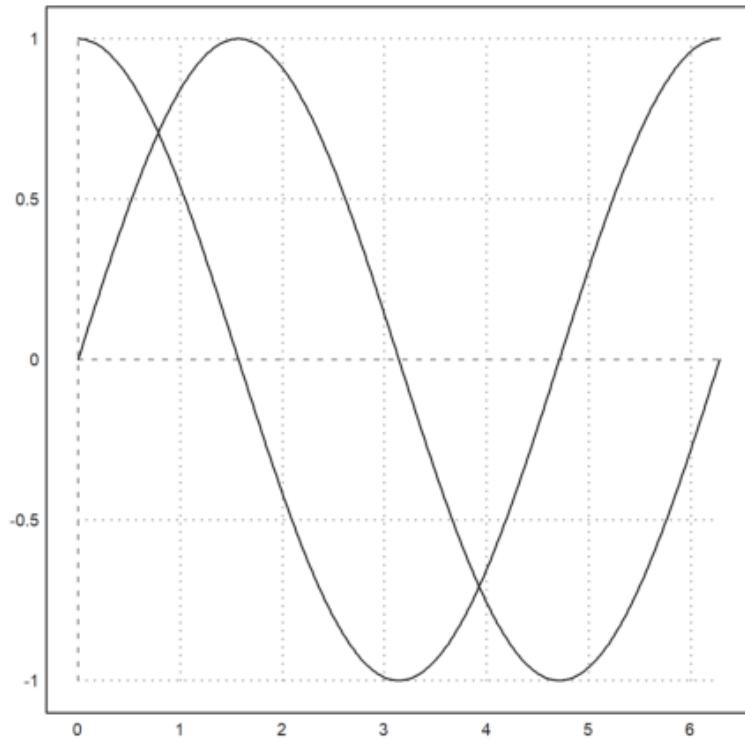
- Dalam contoh ini, merupakan gambar dua kurva sekaligus, yaitu x^2

dan $2x$, pada rentang -3 hingga 3 .

- Hasilnya akan menunjukkan grafik dari kedua fungsi tersebut, dan

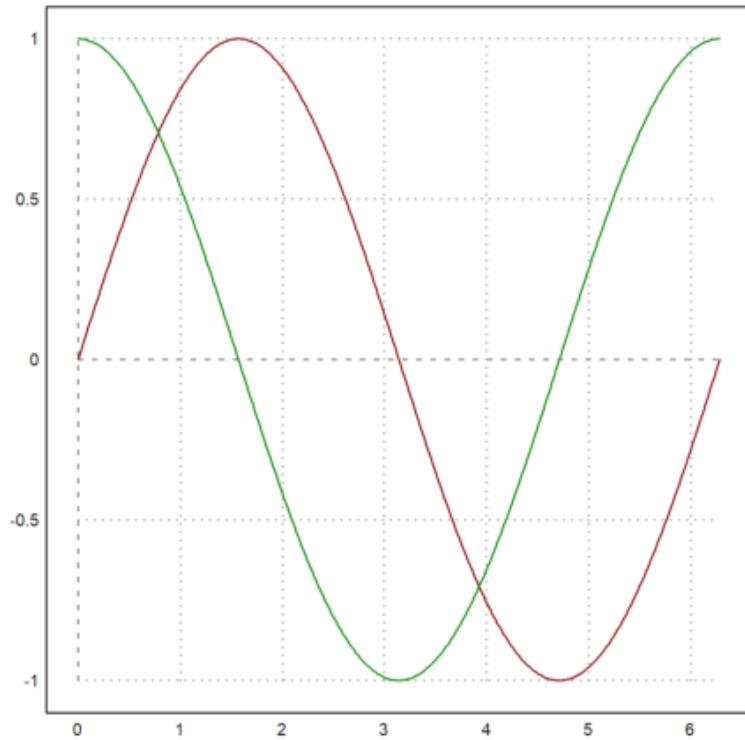
titik-titik potongan antara keduanya adalah solusi dari persamaan kuadrat.

```
>plot2d(["sin(x)", "cos(x)", 0, 2pi):
```



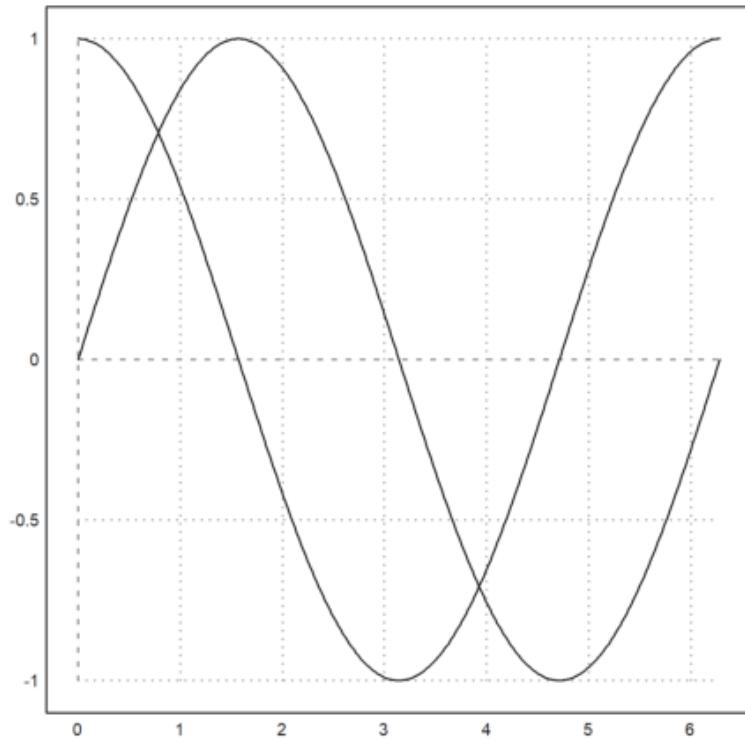
- Pada contoh ini, merupakan gambar dua fungsi trigonometri, $\sin(x)$ dan $\cos(x)$, pada rentang 0 hingga 2π .
- Ini akan menghasilkan dua grafik yang memperlihatkan hubungan antara $\sin(x)$ dan $\cos(x)$ dalam rentang tersebut.

```
>plot2d(["sin(x)", "cos(x")], 0, 2pi, color=red:green):
```



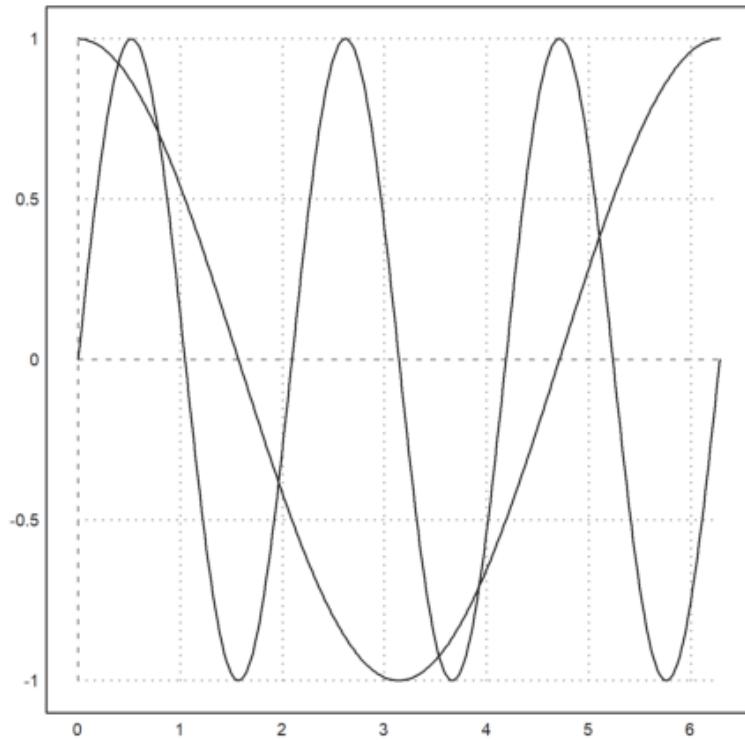
Sama seperti contoh kedua, gambar $\sin(x)$ dan $\cos(x)$ pada rentang 0 hingga 2π , tetapi Anda juga memberikan warna yang berbeda pada kedua grafik ($\sin(x)$ berwarna merah dan $\cos(x)$ berwarna hijau).

```
>plot2d(["sin(x)", "cos(x)", xmin=0, xmax=2pi) :
```



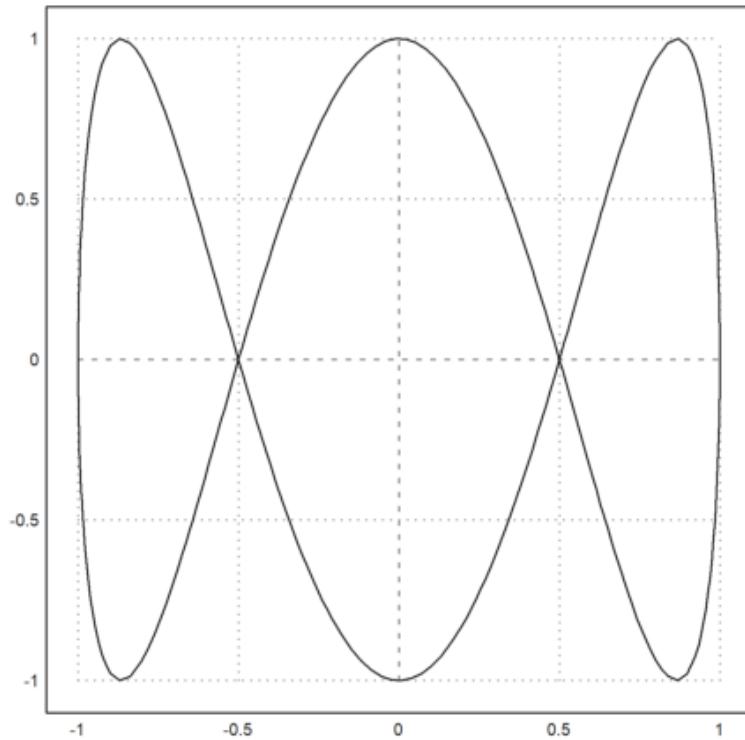
Dalam contoh ini, menggunakan parameter ‘xmin’ dan ‘xmax’ untuk mengatur rentang tampilan grafik pada 0 hingga 2π .

```
>  
>plot2d(["cos (x)", "sin (3*x)"], xmin=0, xmax=2pi):
```



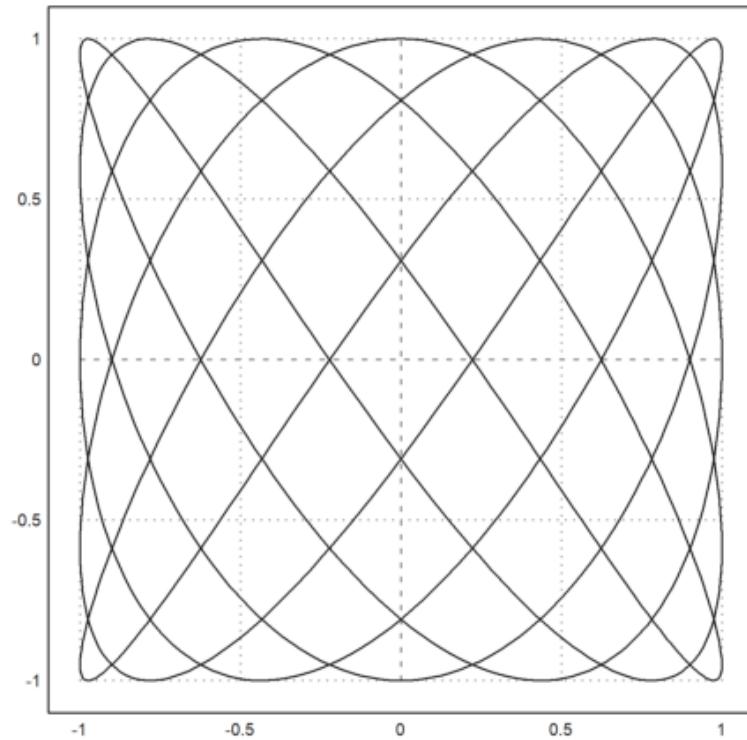
- ini Merupakan gambar dua fungsi, yaitu $\cos(x)$ dan $\sin(3x)$, dalam rentang 0 hingga 2π .
- Penjelasan mencakup konsep bahwa grafik berulang dalam rentang tertentu karena fungsi-fungsi ini memiliki frekuensi, periode, dan amplitudo yang berbeda.

```
>plot2d("cos(x)","sin(3*x)",xmin=0,xmax=2pi):
```



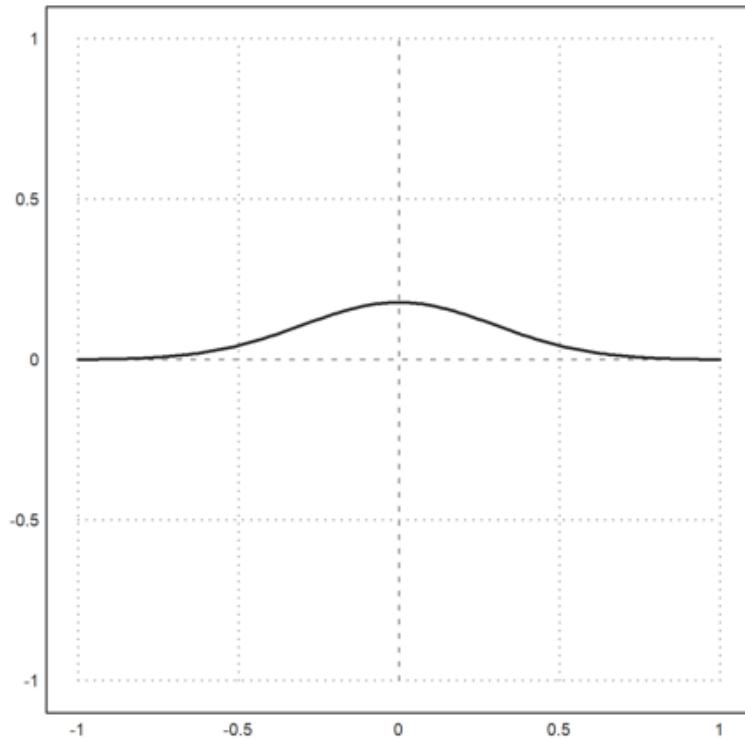
Sintaks diatas lebih menjelaskan bahaimana hubungan periodik grafik fungsi dari 2 fungsi yaitu $\cos x$ dan $\sin 3x$ dari rentang khusus dimana x dari 0 sampai 2π , hal tersebut dapat terjadi karena fungsi $\cos x$ dan fungsi $\sin 3x$ memiliki frekuensi, periode, dan amplitudo yang berbeda. grafik akan berulang pada rentang tertentu dan menghasilkan sebuah pola.

```
>x=linspace(0,2pi,1000); plot2d(sin(5x),cos(7x)):
```



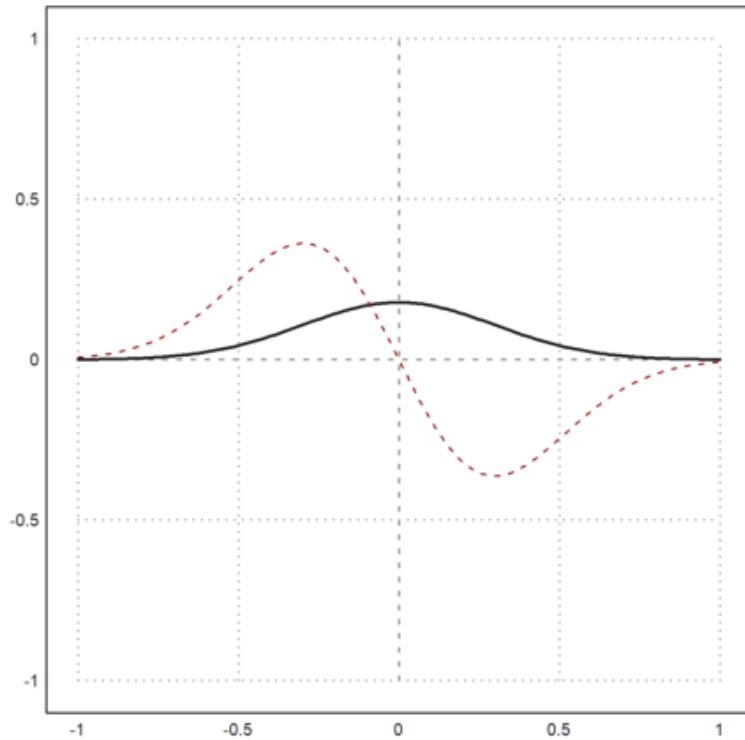
sintaks linspace digunakan untuk menghasilkan vektor x dari rentang yang telah ditentukan yaitu 0 sampai 2pi yang berisi 1000 nilai yang teratur

```
>a:=5.6; f &= exp(-a*x^2)/a;  
>plot2d(f, r=1, thickness=2):
```



- Fungsi $f(x)$ yang merupakan hasil dari ekspresi $\exp(-a^*x^2)/a$. Fungsi ini memiliki parameter a yang bergantung pada nilai yang diterapkan sebelumnya.
- Menggunakan perintah `plot2d` untuk menggambar grafik dari fungsi $f(x)$. Parameter `r` digunakan untuk mengatur rentang plot dan parameter `thickness` digunakan untuk mengatur ketebalan garis grafik.

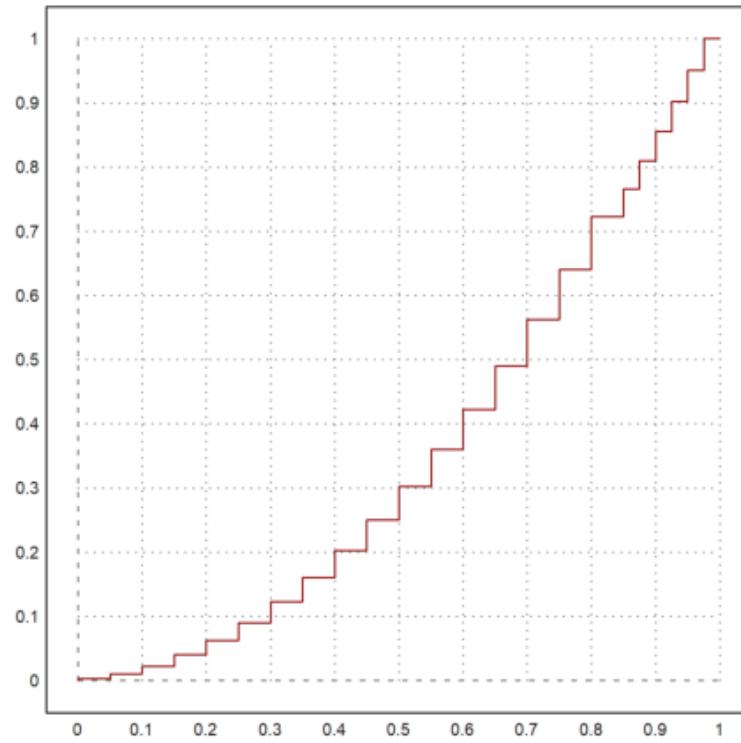
```
>plot2d(&diff(f,x),>add,style="--",color=red) :
```



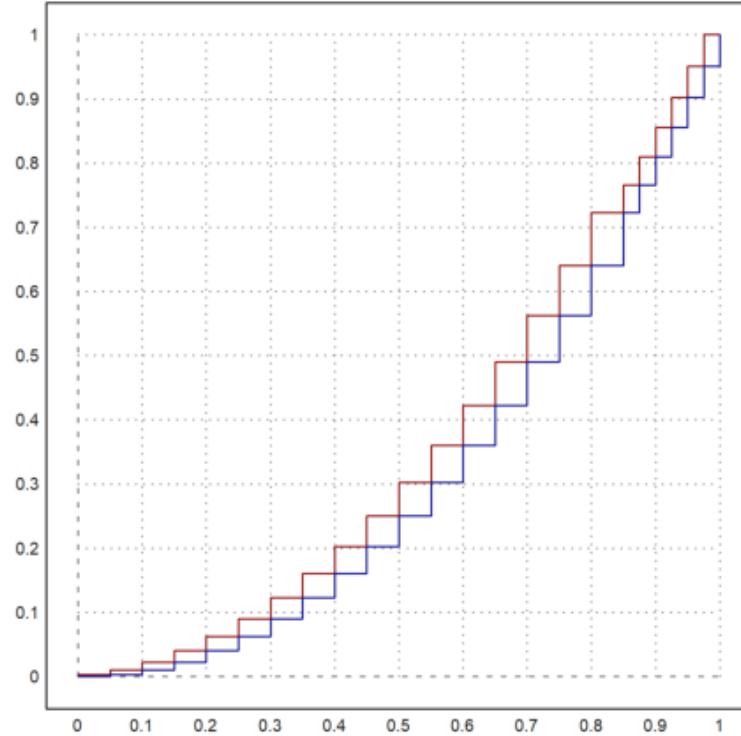
ini adalah grafik fungsi f dan grafik turunan pertama dari fungsi f . sintaks $r=1$ digunakan untuk mengatur rentang yang akan ditampilkan pada plot, $r=1$ berarti rentang dari -1 sampai 1.

sintaks $>\text{add}$ digunakan untuk menambahkan grafik kedalam jendela grafik yang sudah ada sebelumnya.

```
>plot2d("x^2", 0, 1, steps=1, color=red, n=10) :
```



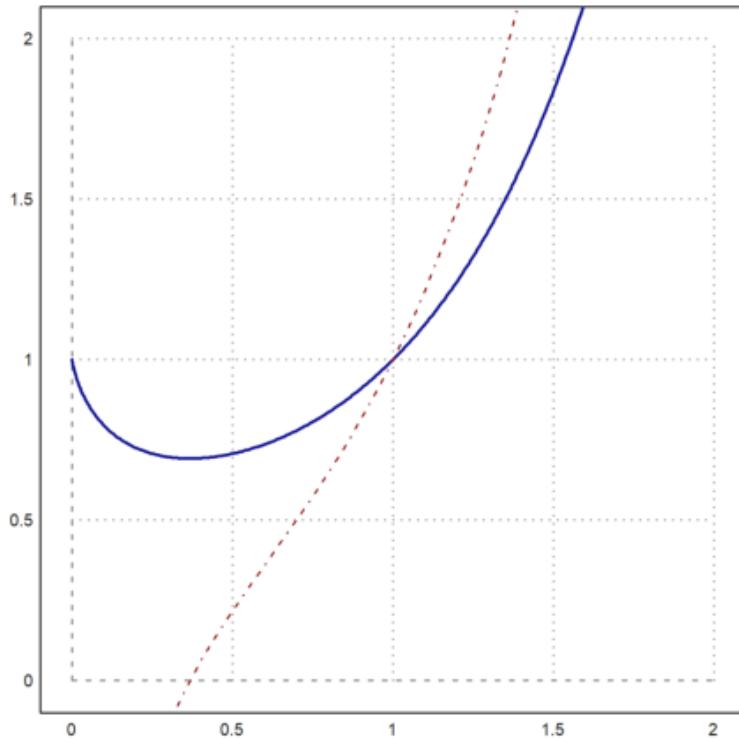
```
>plot2d("x^2",>add,steps=2,color=blue,n=10):
```



sintaks steps digunakan untuk mengatur jumlah langkah atau titik-titik yang digunakan dalam plot.

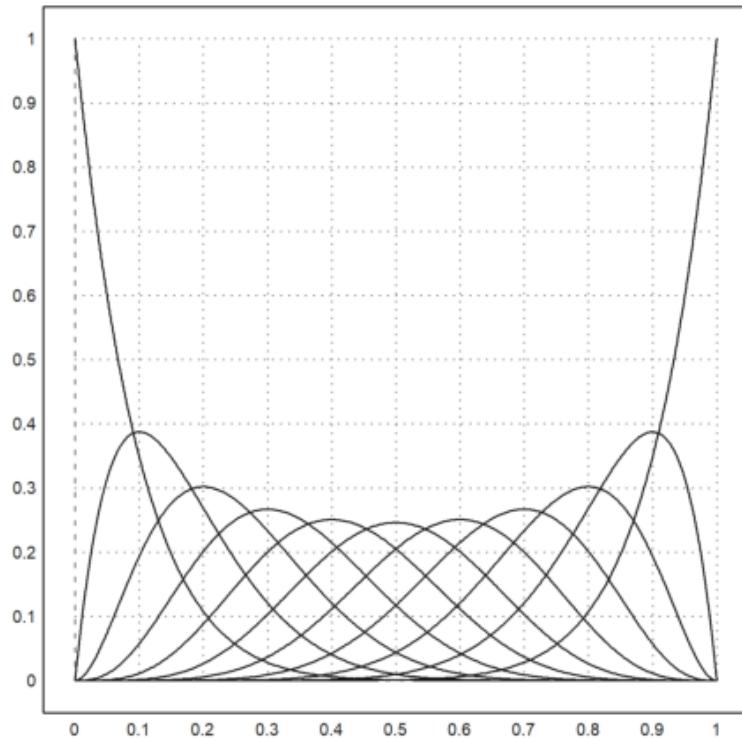
dan sintaks n digunakan untuk mengatur jumlah step yang akan digunakan. semakin banyak n, maka bentuk grafik akan semakin mendekati aslinya.

```
>function f(x) &= x^x;
>plot2d(f,r=1,cx=1,cy=1,color=blue,thickness=2);
>plot2d(&diff(f(x),x),>add,color=red,style="-.-"):
```



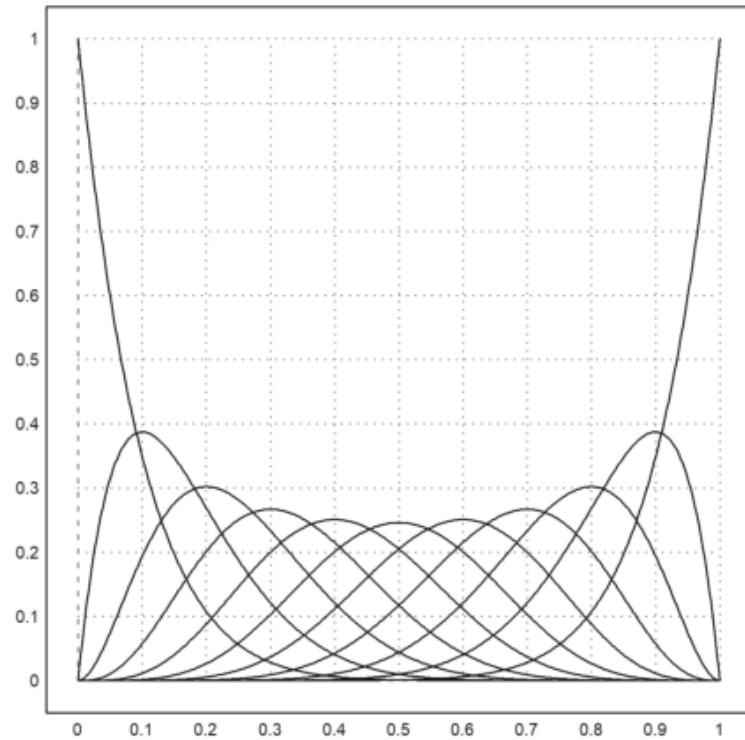
sintaks cx=1, cy=1 digunakan untuk mengatur pusat tampilan grafik, maka plot akan diatur dengan titik pusat (1,1).

```
>plot2d("(1-x)^10",0,1);
>for i=1 to 10; plot2d("bin(10,i)*x^i*(1-x)^(10-i)",>add); end;
>insimg;
```



dalam contoh kita menggambar serangkaian plot yang menggambarkan distribusi binomial dengan berbagai nilai i dari 1 hingga 10. kali ini kita menggunakan sintaks untuk melakukan looping pada fungsi yang berasosiasi dengan koefisien binomial dengan kombinasi 10 item. ini memungkinkan untuk memahami bagaimana distribusiu binomial berubah dengan berbagai parameter.

```
>x=linspace(0,1,500);
>n=10; k=(0:n)';
>y=bin(n,k)*x^k*(1-x)^(n-k);
>plot2d(x,y);
```

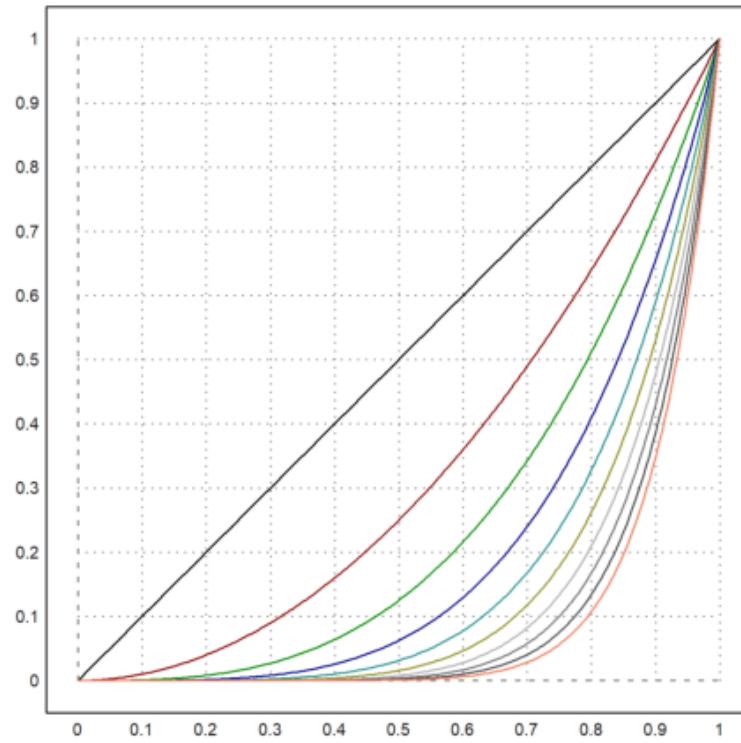


n adalah vektor baris

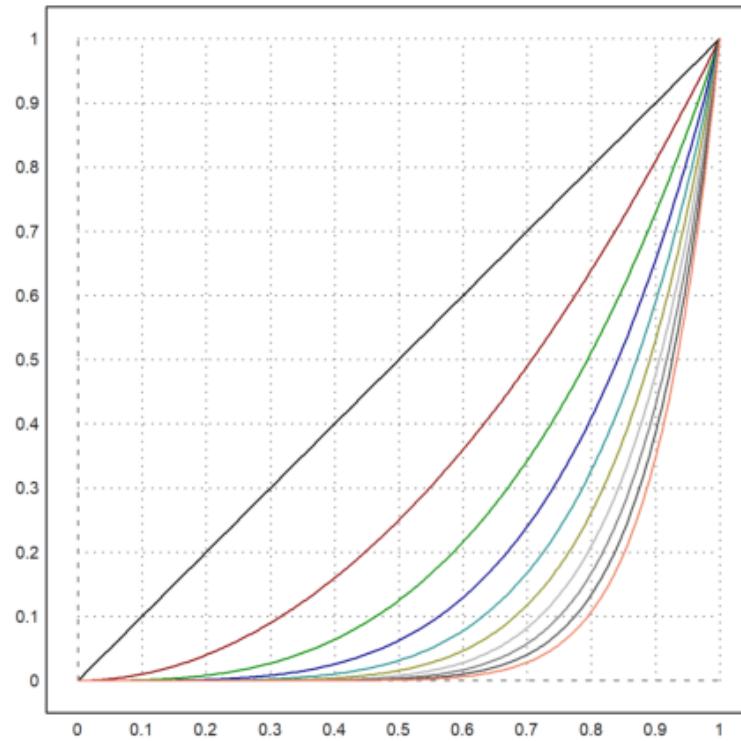
k adalah vektor kolom

y adalah matrik dari vektor baris dan vektor kolom tersebut dengan menggunakan fungsi binomial.

```
>x=linspace(0,1,200); y=x^(1:10)'; plot2d(x,y,color=1:10):
```



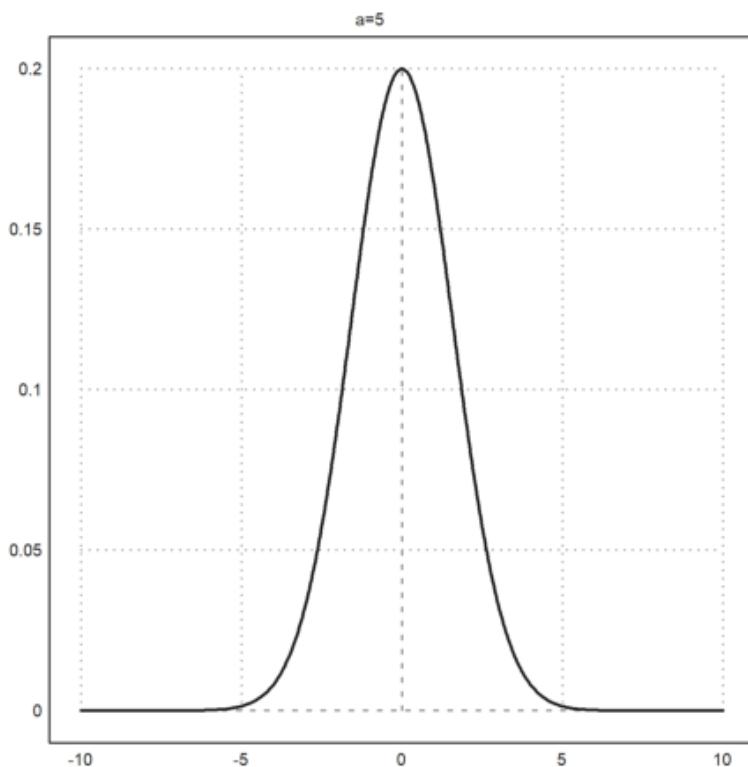
```
>n=(1:10)'; plot2d("x^n",0,1,color=1:10):
```



```

>
>function f(x,a) := 1/a*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5") :

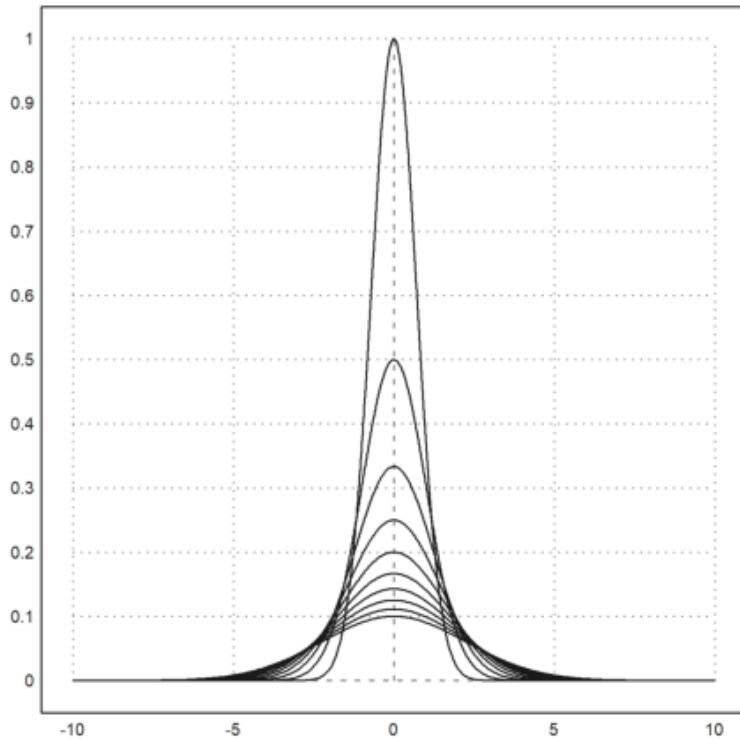
```



```

>plot2d({{"f",1}},-10,10); ...
>for a=2:10; plot2d({{"f",a}}),>add); end:

```



Sub Bab 10

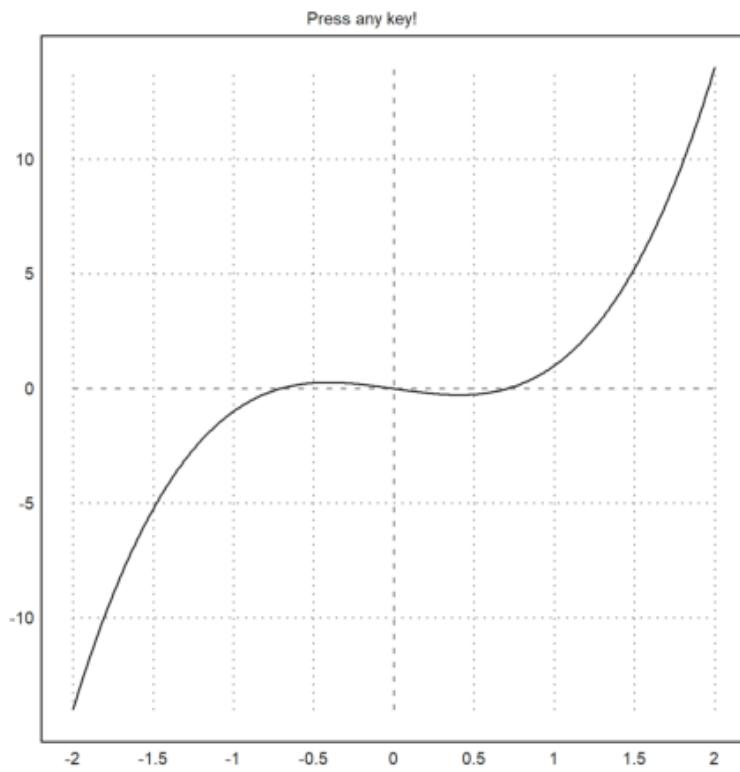
Membuat Gambar Kurva yang Bersifat Interaktif

Kode ini, menggunakan ‘plot2d’ untuk membuat plot dari fungsi matematika ‘ $2*x^3-a*x$ ’ dengan parameter ‘a’. Flag ‘>user’ memungkinkan interaksi pengguna. Setelah plot ditampilkan, pengguna dapat melakukan beberapa tindakan interaktif.

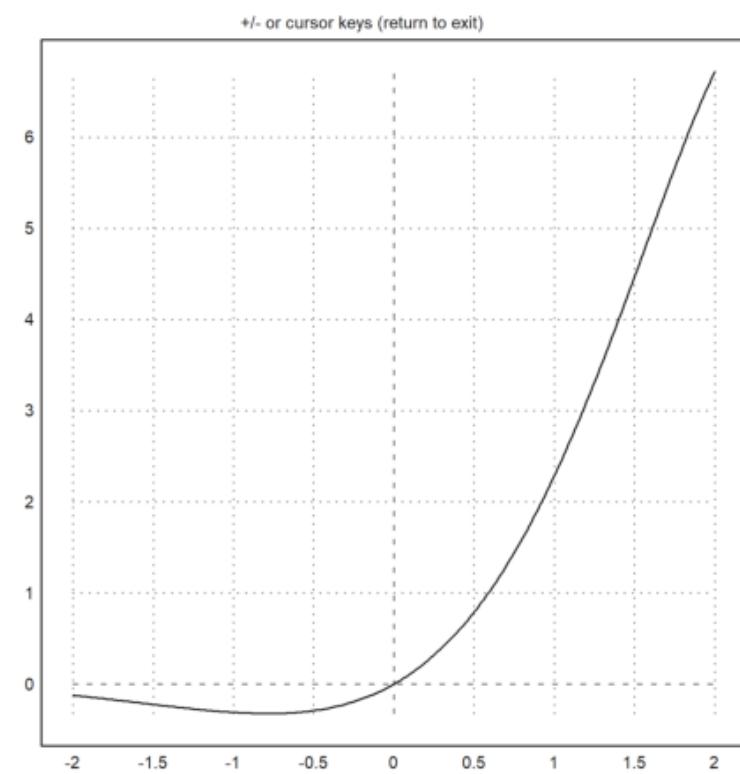
Saat plot ditampilkan dengan flag ‘>user’, pengguna dapat melakukan beberapa tindakan interaktif sebagai berikut:

- Perbesar dengan + atau -: Pengguna dapat memperbesar atau memperkecil plot dengan menggunakan tombol + atau - pada keyboard.
- Pindahkan Plot dengan Tombol Kursor: Pengguna dapat menggeser plot dengan menggunakan tombol kursor (panning).
- Pilih Jendela Plot dengan Mouse: Pengguna dapat memilih area tertentu dalam plot dengan menggunakan mouse.
- Atur Ulang Tampilan dengan Spasi: Jika pengguna menekan tombol spasi, maka tampilan plot akan diatur ulang ke jendela plot.
- Keluar dengan Kembali: Jika pengguna menekan tombol kembali, maka pengguna dapat keluar dari interaksi plot.

```
>plot2d({{ "2*x^3-a*x", a=1 }}, >user, title="Press any key!"); ...
>insimg;
```



```
> plot2d("exp(x)*sin(x)", user=true, ...
> title="+/- or cursor keys (return to exit)":
```



Berikut ini menunjukkan cara interaksi pengguna tingkat lanjut

Ini adalah pemanggilan fungsi plot2d yang digunakan untuk membuat plot dari fungsi matematika $\exp(x) * \sin(x)$. Parameter user=true menunjukkan bahwa ini adalah plot yang interaktif, yang berarti pengguna dapat berinteraksi dengan plot ini.

title="+/- or cursor keys (return to exit)": Ini adalah judul yang akan ditampilkan di atas plot. Pesan ini memberi petunjuk kepada pengguna tentang bagaimana mereka dapat berinteraksi dengan plot ini. Mereka dapat menggunakan tombol + atau - atau tombol kurSOR untuk berinteraksi dengan plot, dan tombol return (Enter) untuk keluar dari interaksi.

Berikut ini menunjukkan cara interaksi pengguna tingkat lanjut:

- mousedrag(): Ini adalah fungsi bawaan yang digunakan untuk menunggu event mouse atau keyboard. Fungsi ini dapat mendeteksi kejadian seperti klik mouse, pergerakan mouse, atau penekanan tombol.
- dragpoints(): Fungsi ini memanfaatkan mousedrag() untuk memungkinkan pengguna menyeret titik-titik pada plot. Ini berarti pengguna dapat mengklik dan menarik titik-titik dalam plot sesuai dengan preferensi mereka.

Kita membutuhkan fungsi plot terlebih dahulu. Sebagai contoh, kita interpolasi dalam 5 titik dengan polinomial. Fungsi harus diplot ke area plot tetap.

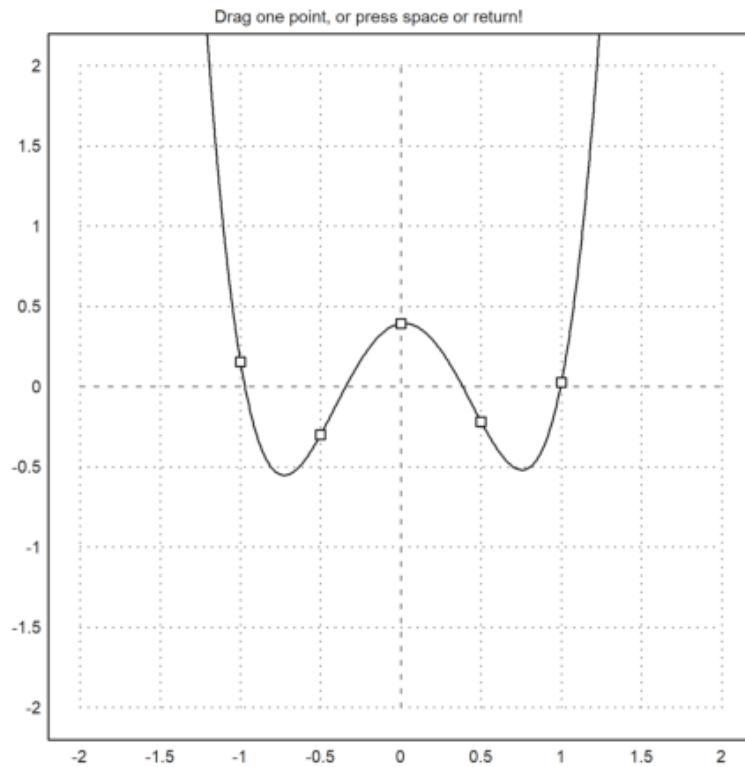
```
>function plotf(xp,yp,select) ...
```

```
d=interp(xp,yp);
plot2d("interval(xp,d,x)";d,xp,r=2);
plot2d(xp,yp,>points,>add);
if select>0 then
    plot2d(xp[select],yp[select],color=red,>points,>add);
endif;
title("Drag one point, or press space or return!");
endfunction
```

Perhatikan parameter titik koma di plot2d (d dan xp), yang diteruskan ke evaluasi fungsi interp(). Tanpa ini, kita harus menulis fungsi plotinterp() terlebih dahulu, mengakses nilai secara global.

Sekarang kita menghasilkan beberapa nilai acak, dan membiarkan pengguna menyeret poin. kode berikut digunakan untuk menghasilkan beberapa nilai acak t dan membiarkan pengguna menyeret titik-titik pada plot dengan menggunakan fungsi dragpoints():

```
>t=-1:0.5:1; dragpoints("plotf",t,random(size(t))-0.5):
```



Ada juga fungsi, yang memplot fungsi lain tergantung pada vektor parameter, dan memungkinkan pengguna menyesuaikan parameter ini.

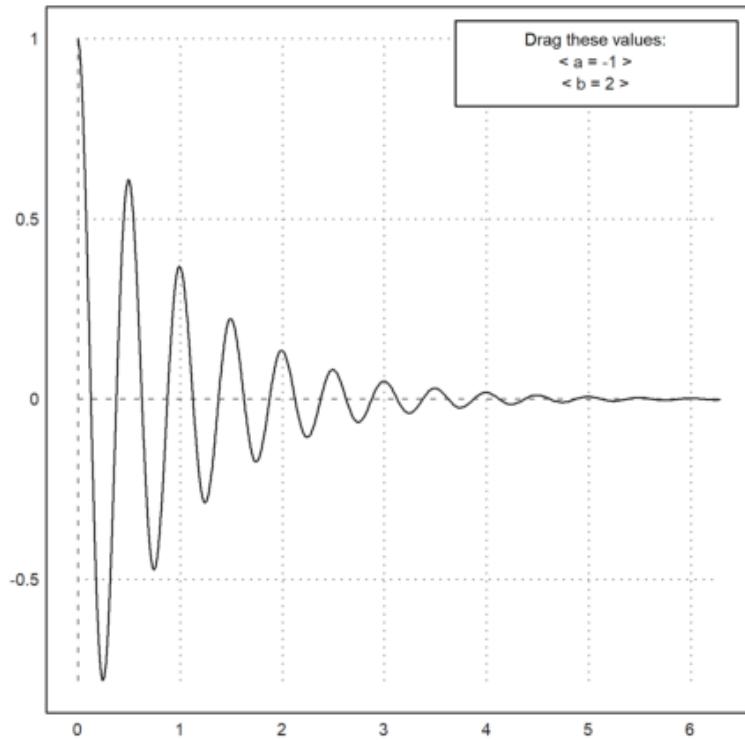
Pertama kita membutuhkan fungsi plot.

```
>function plotf([a,b]) := plot2d("exp(a*x)*cos(2pi*b*x)",0,2pi;a,b);
```

Kemudian kita membutuhkan nama untuk parameter, nilai awal dan matriks rentang nx2, opsional baris judul.

Ada slider interaktif, yang dapat mengatur nilai oleh pengguna. Fungsi dragvalues() menyediakan ini.

```
>dragvalues("plotf", ["a", "b"], [-1, 2], [[-2, 2]; [1, 10]], ...
> heading="Drag these values:", hcolor=black):
```



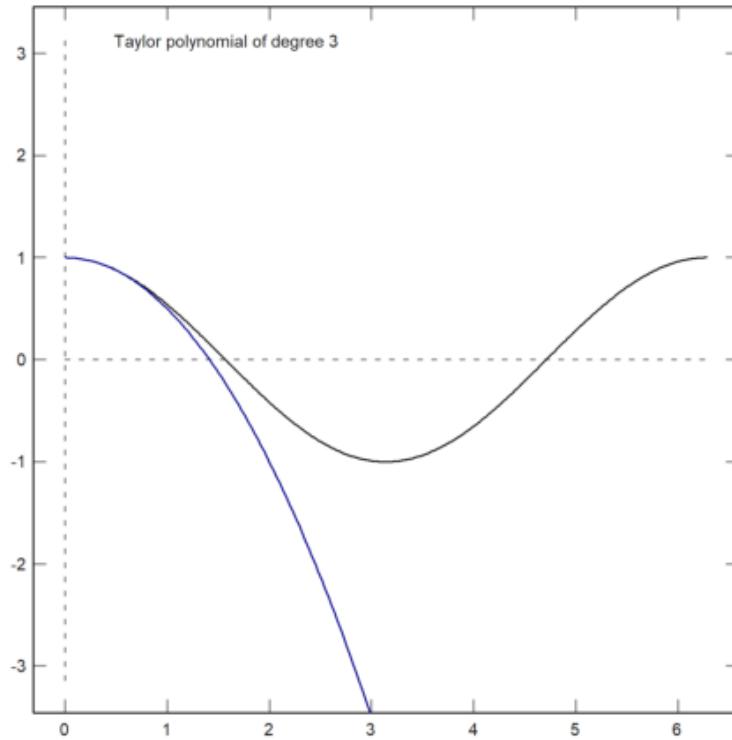
Dimungkinkan untuk membatasi nilai yang diseret ke bilangan bulat. Sebagai contoh, kita menulis fungsi plotf, yang memplot polinomial Taylor derajat n ke fungsi kosinus.

```
>function plotf(n) ...
```

```
plot2d("cos(x)", 0, 2pi, >square, grid=6);
plot2d(&"taylor(cos(x),x,0,@n)", color=blue, >add);
textbox("Taylor polynomial of degree "+n, 0.1, 0.02, style="t", >left);
endfunction
```

Sekarang kami mengizinkan derajat n bervariasi dari 0 hingga 20 dalam 20 pemberhentian. Hasil dragvalues() digunakan untuk memplot sketsa dengan n ini, dan untuk memasukkan plot ke dalam buku catatan.

```
>nd=dragvalues("plotf", "degree", 3, [0,10], 10, y=0.8, ...
> heading="Drag the value:"); ...
>plotf(nd);
```



Berikut ini adalah demonstrasi sederhana dari fungsi tersebut. Pengguna dapat menggambar di atas jendela plot, meninggalkan jejak poin.

```
>function dragtest ...
```

```
plot2d(none,r=1,title="Drag with the mouse, or press any key!");
start=0;
repeat
  {flag,m,time}=mousedrag();
  if flag==0 then return; endif;
  if flag==2 then
    hold on; mark(m[1],m[2]); hold off;
  endif;
end
endfunction
```

Sub Bab 11

Menggambar Kurva Fungsi Parametrik

Kita telah terbiasa dengan kurva yang didefinisikan oleh sebuah persamaan yang menghubungkan koordinat x dan y Contohnya

$$y = x^2$$

Atau

$$x^2 + y^2 = 13$$

dimana persamaan-persamaan ini tidak dikaitkan dengan panjang kurva s , waktu t, dan besaran lainnya. Besaran besaran ini disebut parameter
persamaan parametrik adalah persamaan yang menyatakan hubungan variabel x, y dituliskan dengan

$$x = f(t)$$

$$y = g(t)$$

dengan $a \leq t \leq b$ tiap nilai t menentukan titik(x,y) pada kurva. Jadi , dengan berubahnya nilai t. titik

$$(x, y) = (f(t), g(t))$$

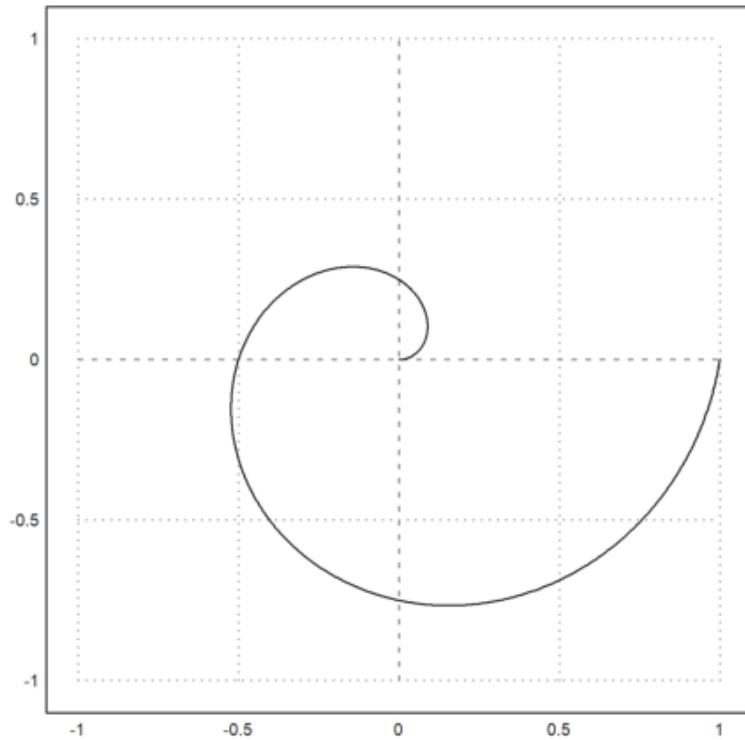
bergerak sepanjang kurva yang disebut kurva parametrik

Dalam contoh berikut, kita memplot spiral

$$\gamma(t) = t \cdot (\cos(2\pi t), \sin(2\pi t))$$

Kita perlu menggunakan banyak titik untuk tampilan yang halus

```
>t=linspace(0,1,1000); ...
>plot2d(t*cos(2*pi*t),t*sin(2*pi*t),r=1):
```

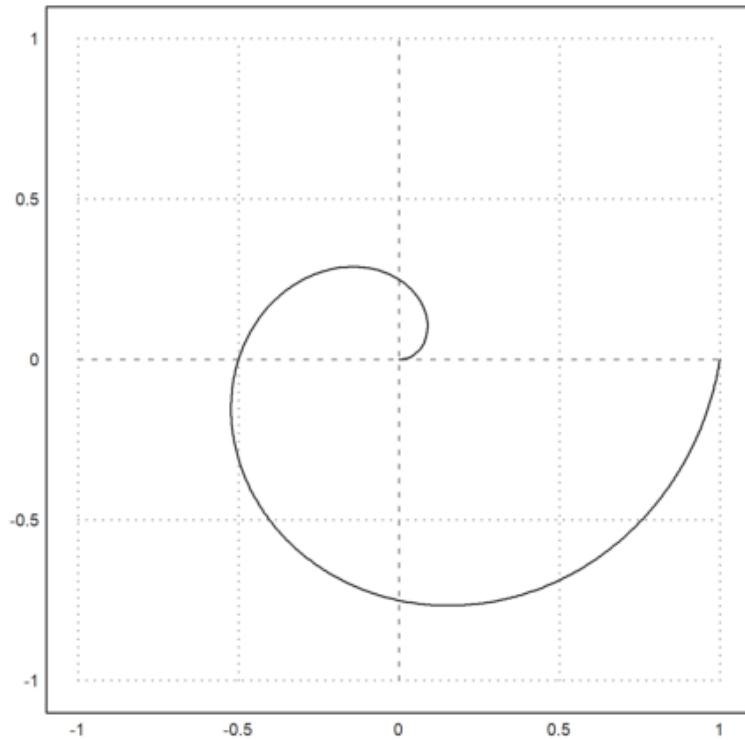


`r` digunakan untuk mengatur radius marker titik-titik yang akan

digunakan dalam plot.

Sebagai alternatif, dimungkinkan untuk menggunakan dua ekspresi untuk kurva. Berikut ini plot kurva yang sama seperti di atas.

```
>plot2d("x*cos(2*pi*x)", "x*sin(2*pi*x)", xmin=0, xmax=1, r=1) :
```

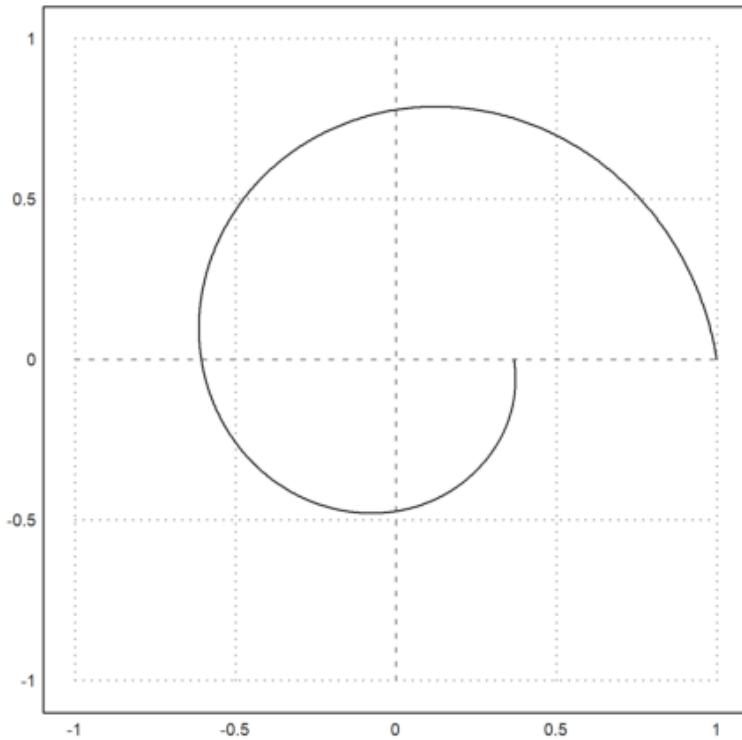


Perintah linspace digunakan untuk membuat array nilai yang terdistribusi secara merata antara dua angka tertentu. Fungsi ini sangat berguna untuk menentukan rentang nilai yang ingin digunakan pada sumbu x atau y ketika membuat plot.

0 : Nilai awal dari rentang.
 1 : Nilai akhir dari rentang.

Perintah linspace akan menghasilkan array dengan n elemen yang terdistribusi merata antara start dan stop.

```
>t=linspace(0,1,1000); r=exp(-t); x=r*cos(2pi*t); y=r*sin(2pi*t);
>plot2d(x,y,r=1):
```



$\exp(-t)$ menghasilkan nilai yang semakin mendekati nol seiring dengan pertambahan nilai t , karena eksponensial dari nilai negatif semakin mendekati nol saat nilai t semakin besar. Jadi, $r = \exp(-t)$ memberikan suatu fungsi yang menurun dengan nilai t . Dalam konteks program ini, r digunakan untuk mengontrol jari-jari dari kurva spiral dalam plot 2D. Jari-jari ini semakin kecil seiring dengan pertambahan nilai t , menciptakan efek spiral yang semakin rapat ke pusat pada bagian ujung kurva.

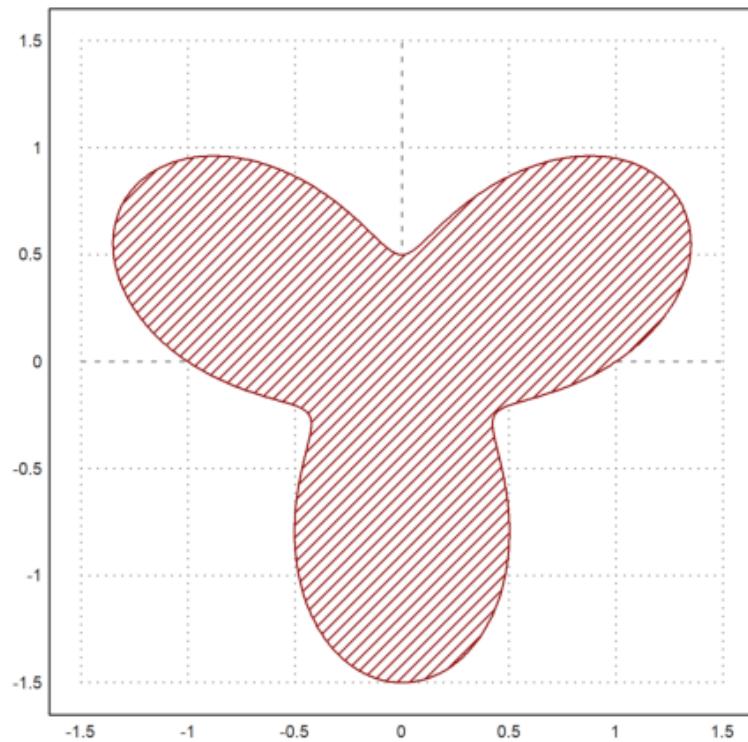
Pada contoh berikutnya, kita memplot kurvanya

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

dengan

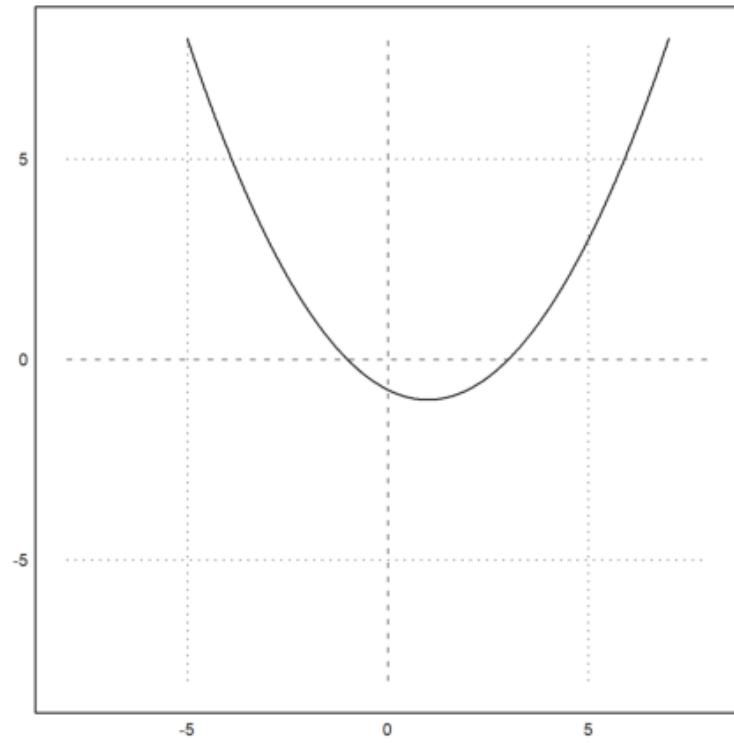
$$r(t) = 1 + \frac{\sin(3t)}{2}.$$

```
>t=linspace(0,2pi,1000); r=1+sin(3*t)/2; x=r*cos(t); y=r*sin(t); ...
>plot2d(x,y,>filled,fillcolor=red,style="/" ,r=1.5):
```

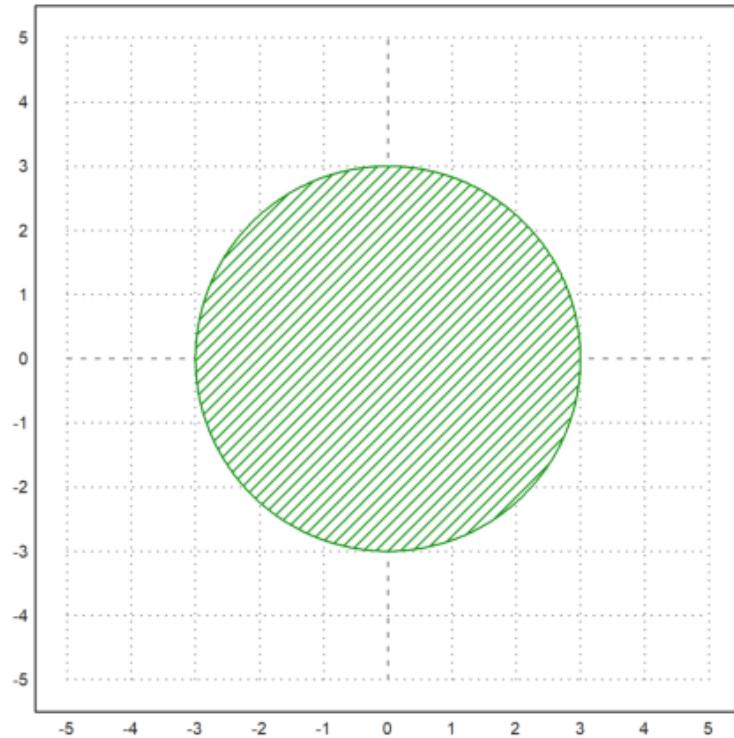


Contoh lain

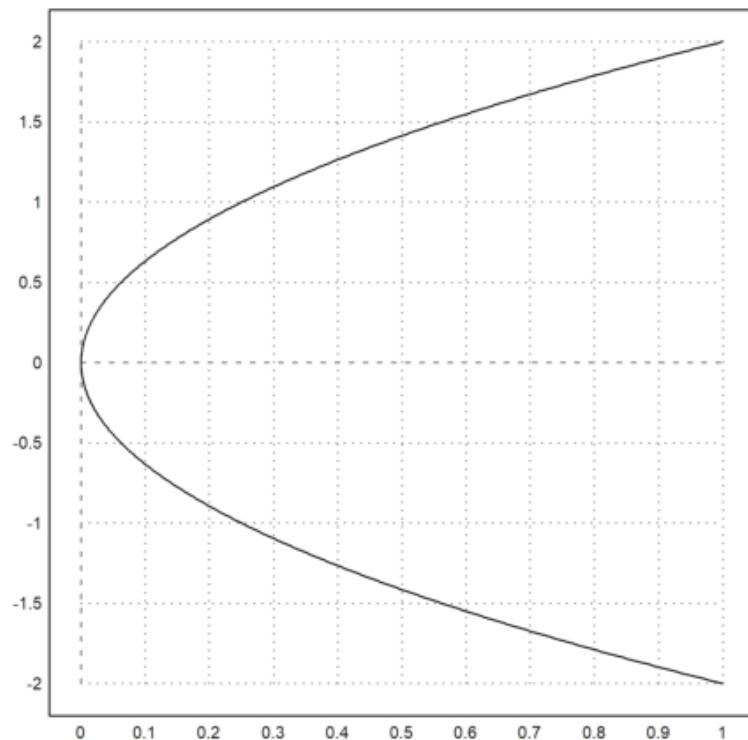
```
>t=linspace(-3,3,1000); x=2*t+1; y=t^2-1;  
>plot2d(x,y,r=8):
```



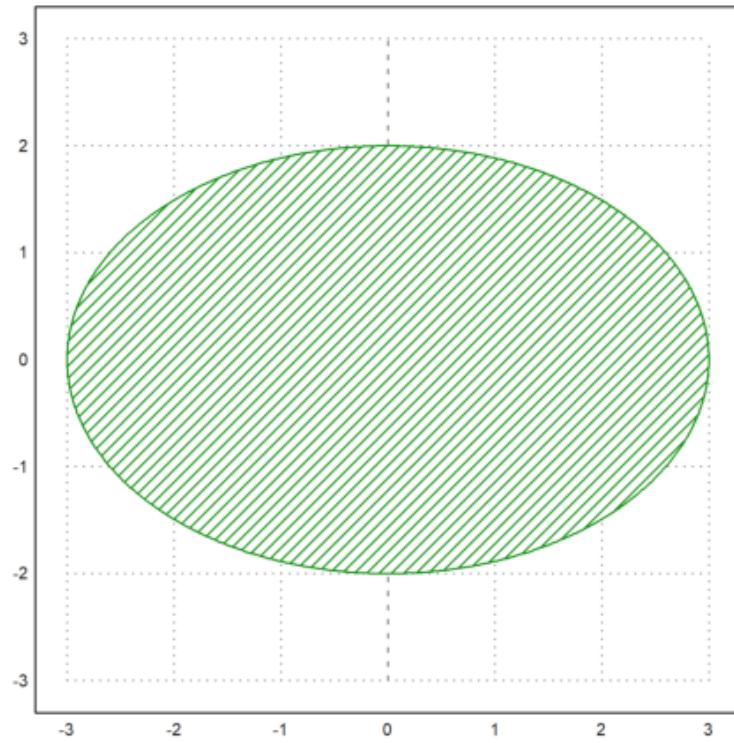
```
>t=linspace(0,2pi,1000); r=3; x=r*cos(t); y=r*sin(t);...
>plot2d(x,y,>filled,fillcolor=green,style="/" ,r=5) :
```



```
>t=linspace(-1,1,1000); x=t^2; y=2*t;...
>plot2d(x,y):
```



```
>t=linspace(0,2pi,1000); x=3*cos(t); y=2*sin(t);...
>plot2d(x,y,>filled,fillcolor=green,style="/",r=3):
```



Sub Bab 12

Menggambar Kurva Fungsi Implisit

Fungsi implisit adalah fungsi yang memuat lebih dari satu variabel, berjenis variabel bebas dan variabel terikat yang berada dalam satu ruas sehingga tidak bisa dipisahkan pada ruas yang berbeda.

Untuk fungsi implisit, harus berupa fungsi atau ekspresi dari parameter x dan y.

$$f(x, y) = c$$

Untuk menggambar himpunan $f(x,y)=c$ untuk satu atau lebih konstanta c , dapat menggunakan
plot2d().

Fungsi implisit juga dapat diisi dengan persamaan tingkat

$$a \leq f(x, y) \leq b$$

Untuk fungsi ini harus berupa matriks $2 \times n$ dimana baris pertama berisi awal dan baris kedua adalah akhir dari setiap interval.

Plot implisit menunjukkan garis level yang menyelesaikan $f(x,y)=\text{level}$, di mana "level" dapat berupa nilai tunggal atau vektor nilai. Jika $\text{level}=\text{"auto"}$, akan ada garis level nc, yang akan menyebar antara fungsi minimum dan maksimum secara merata. Warna yang lebih gelap atau lebih terang dapat ditambahkan dengan >hue untuk menunjukkan nilai fungsi.

Untuk fungsi implisit, xv harus berupa fungsi atau ekspresi dari parameter x dan y, atau, sebagai alternatif, xv dapat berupa matriks nilai.

Euler dapat menandai garis level

$$f(x, y) = c$$

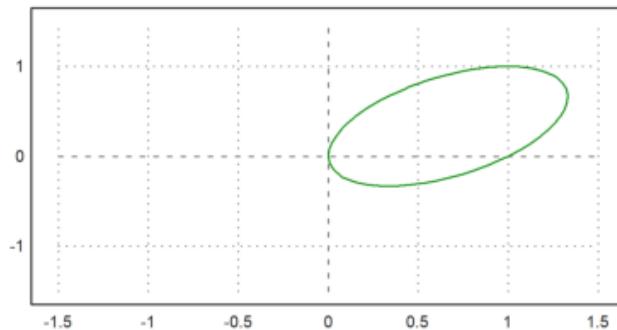
dari fungsi apapun.

Untuk menggambar himpunan $f(x,y)=c$ untuk satu atau lebih konstanta c, Anda dapat menggunakan plot2d() dengan plot implisitnya di dalam bidang. Parameter untuk c adalah level=c, di mana c dapat berupa vektor garis level. Selain itu, skema warna dapat digambar di latar belakang untuk menunjukkan nilai fungsi untuk setiap titik dalam plot. Parameter "n" menentukan kehalusan plot.

Contoh Soal

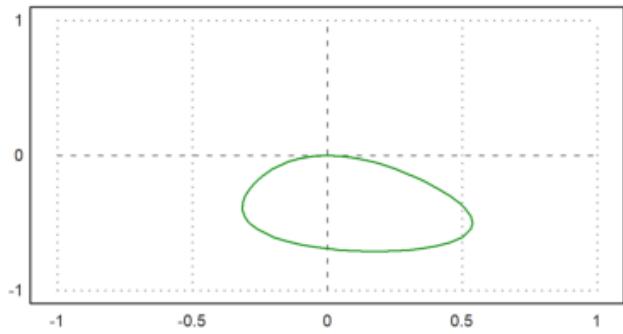
$$x^2 + y^2 - xy - x = 0$$

```
>aspect(2)
>plot2d("x^2+y^2-x*y-x", r=1.5, level=0, contourcolor=green):
```

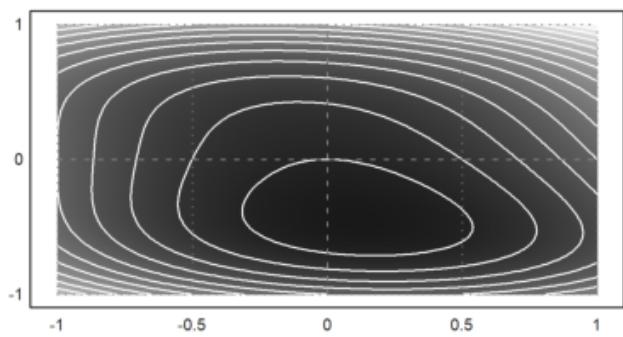


$$2x^2 + xy + 3y^4 + y = 0$$

```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr, level=0, contourcolor=green): // Solutions of f(x,y)=0
```

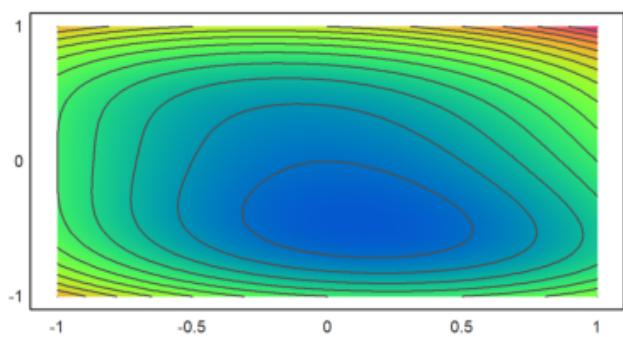


```
>plot2d(expr,level=0:0.5:20,>hue,contourcolor=white,n=200): // nice
```



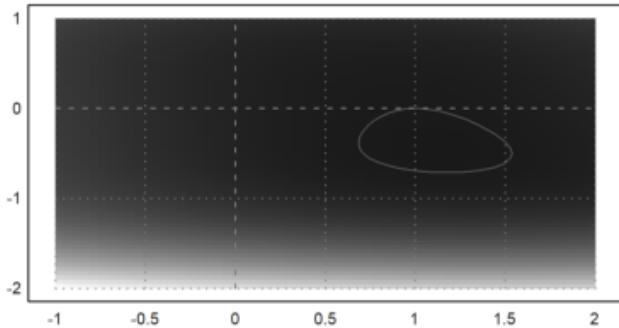
Parameter `>hue` digunakan untuk memberikan warna pada kontur sesuai dengan levelnya. Kontur dengan level yang lebih tinggi akan memiliki warna yang berbeda.

```
>plot2d(expr,level=0:0.5:20,>hue,>spectral,n=200,grid=4): // nicer
```

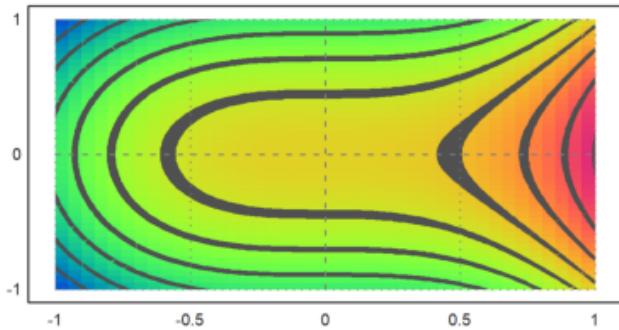


`>spectral` digunakan untuk mengatur palet warna yang akan digunakan pada kontur. Dalam hal ini, digunakan palet warna "spectral".

```
>x=-2:0.05:1; y=x'; z=expr(x,y);
>plot2d(z,level=0,a=-1,b=2,c=-2,d=1,>hue):
```



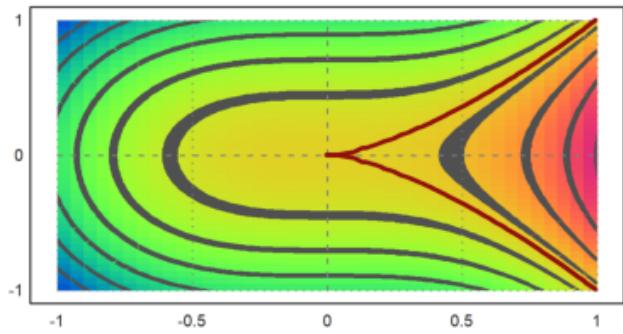
```
>plot2d("x^3-y^2",>contour,>hue,>spectral):
```



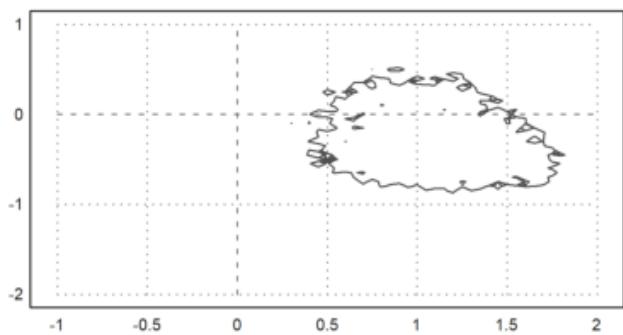
Perintah `>contour` adalah cara untuk menghasilkan plot kontur, yaitu plot yang menunjukkan garis-garis kontur yang mewakili tingkat-tingkat dari suatu fungsi. Jumlah dan posisi garis kontur akan secara otomatis diatur oleh Euler Math Toolbox berdasarkan distribusi nilai-nilai fungsi.

Penggunaan level memungkinkan Anda secara eksplisit menentukan tingkat kontur yang ingin kita tampilkan pada plot. kita dapat mengatur level kontur sesuai dengan preferensi kita, dan plot akan menampilkan garis kontur pada tingkat-tingkat yang kita tentukan.

```
>plot2d("x^3-y^2",level=0,contourwidth=3,>add,contourcolor=red):
```

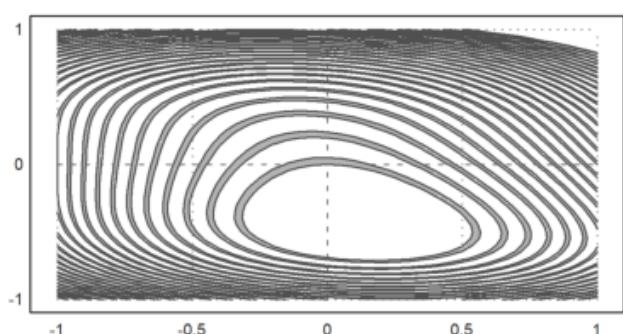


```
>z=z+normal(size(z))*0.2;
>plot2d(z,level=0.5,a=-1,b=2,c=-2,d=1):
```

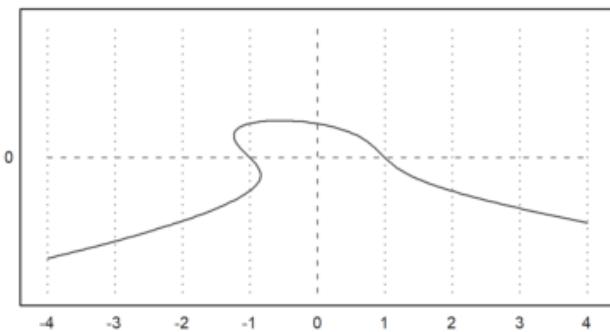


`normal(size(z))` menghasilkan matriks dengan ukuran yang sama dengan matriks `z`, dan setiap elemennya diambil dari distribusi normal standar (mean 0, deviasi standar 1). Kemudian, matriks `z` diubah dengan menambahkan nilai-nilai acak ini, yang telah dikalikan dengan 0.2. Ini menciptakan variasi acak dalam matriks `z`.

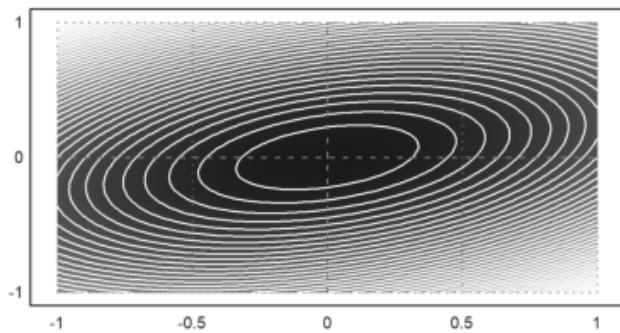
```
>plot2d(expr,level=[0:0.2:5;0.05:0.2:5.05],color=lightgray):
```



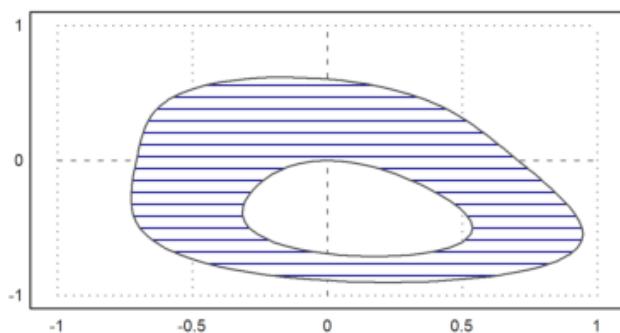
```
>plot2d("x^2+y^3+x*y",level=1,r=4,n=100):
```



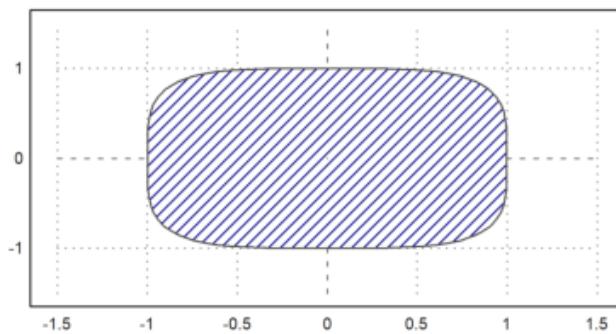
```
>plot2d("x^2+2*y^2-x*y",level=0:0.1:10,n=100,contourcolor=white,>hue):
```



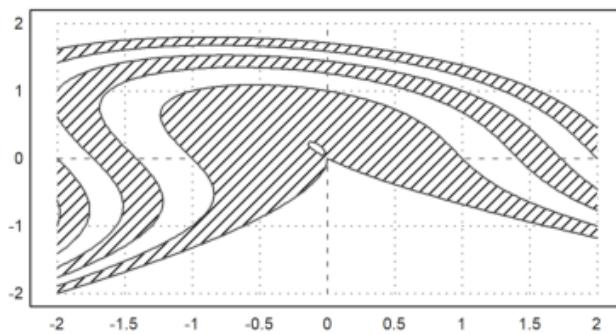
```
>plot2d(expr,level=[0;1],style="-",color=blue): // 0 <= f(x,y) <= 1
```



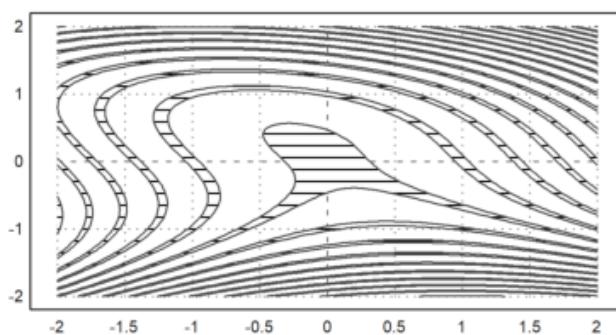
```
>plot2d("x^4+y^4", r=1.5, level=[0;1], color=blue, style="/"):
```



```
>plot2d("x^2+y^3+x*y", level=[0,2,4;1,3,5], style="/", r=2, n=100):
```

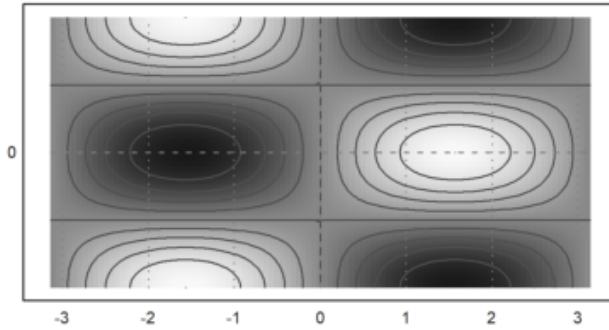


```
>plot2d("x^2+y^3+x*y", level=-10:20, r=2, style="-", dl=0.1, n=100):
```

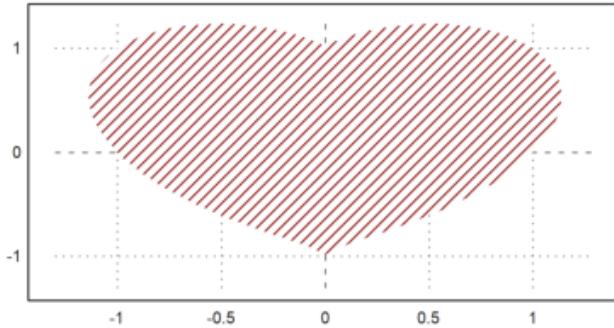


dl Parameter ini mengatur tingkat penghalusan pada plot. Semakin kecil nilai ini, semakin halus plotnya.

```
>plot2d("sin(x)*cos(y)",r=pi,>hue,>levels,n=100) :
```



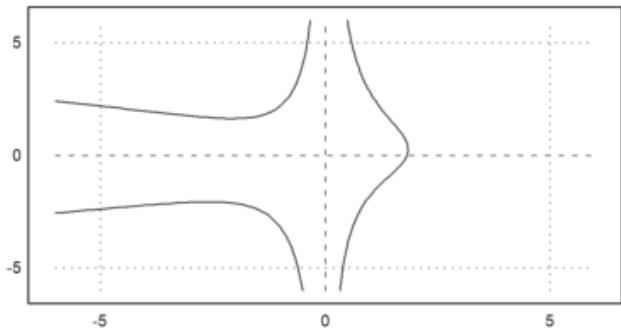
```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...
>style="/",color=red,<outline, ...
>level=[-2;0],n=100) :
```



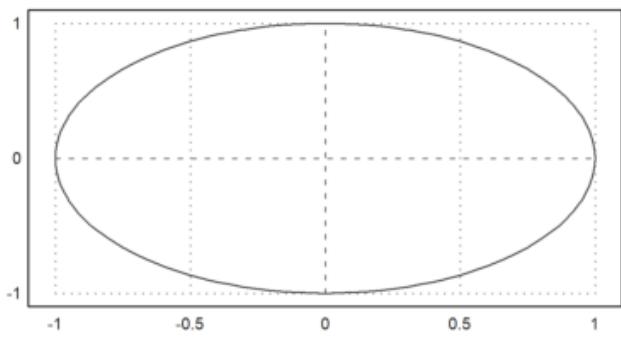
<outline: Parameter ini mengatur plot agar hanya memiliki kontur saja tanpa diisi.
Misal plot solusi dari persamaan

$$x^3 - xy + x^2y^2 = 6$$

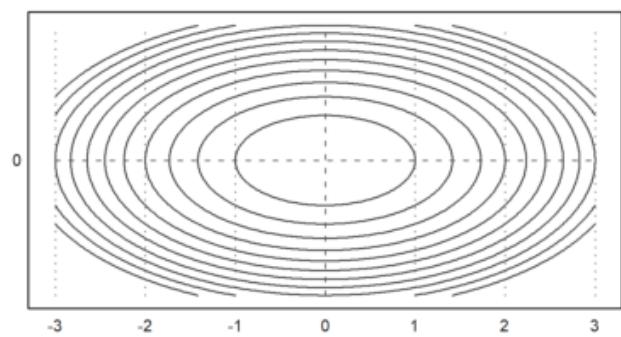
```
>plot2d("x^3-x*y+x^2*y^2",r=6,level=6,n=100) :
```



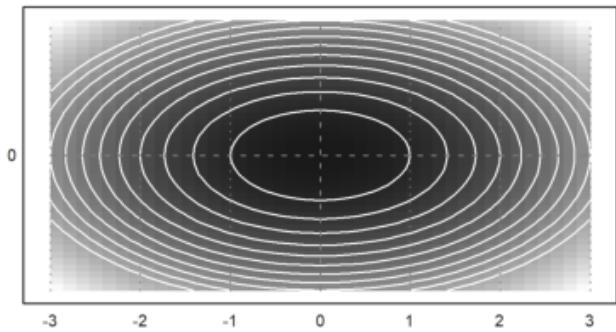
```
>plot2d("x^2+y^2-1", level=0):
```



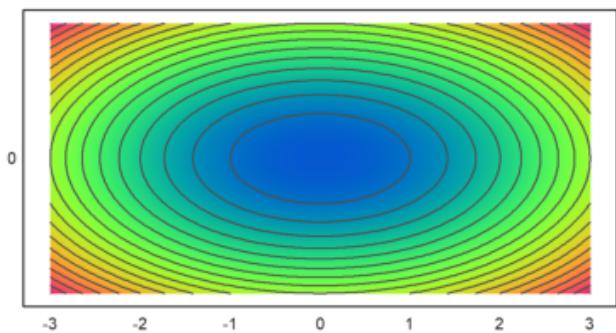
```
>plot2d("x^2+y^2-1", r=3, level=0:1:10, n=200):
```



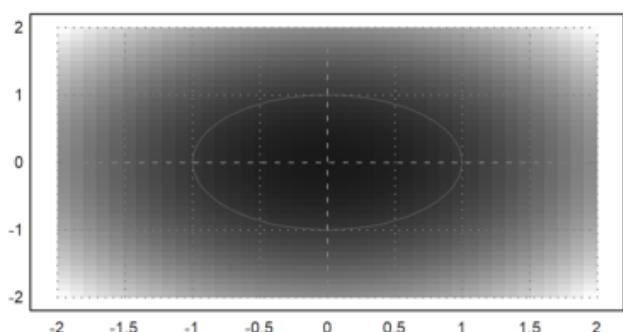
```
>plot2d("x^2+y^2-1", r=3, level=0:1:10, >hue, contourcolor=white):
```



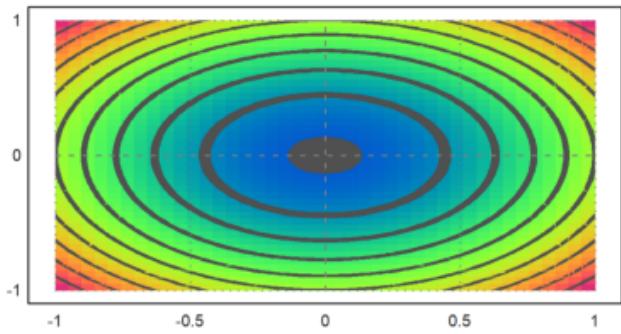
```
>plot2d("x^2+y^2-1", r=3, level=0:1:20, >hue, >spectral, n=200, grid=4) :
```



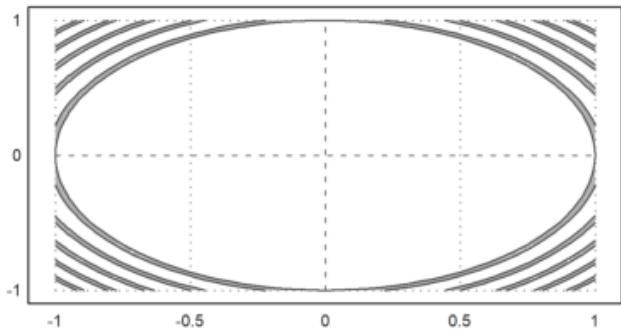
```
>plot2d("x^2+y^2-1", level=0, a=-2, b=2, c=-2, d=2, >hue) :
```



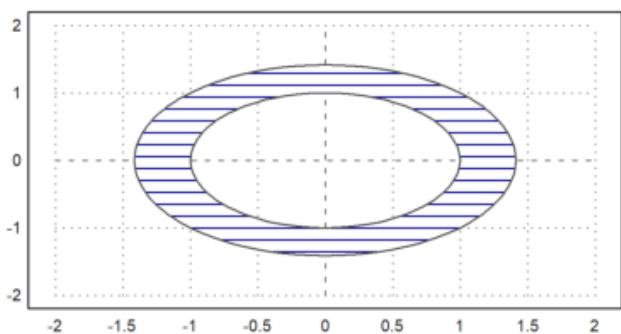
```
>plot2d("x^2+y^2-1", >contour, >hue, >spectral) :
```



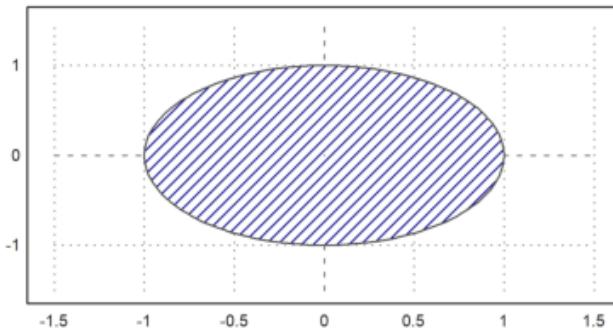
```
>plot2d("x^2+y^2-1", level=[0:0.2:5; 0.05:0.2:5.05], color=lightgray):
```



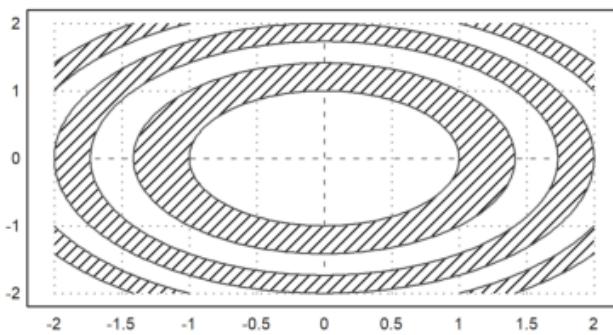
```
>plot2d("x^2+y^2-1", r=2, level=[0;1], style="-", color=blue): // 0 <= f(x,y) <
```



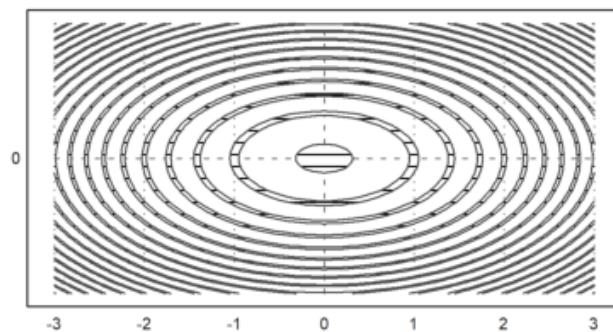
```
>plot2d("x^2+y^2", r=1.5, level=[0;1], color=blue, style="/"):
```



```
>plot2d("x^2+y^2-1",level=[0,2,4;1,3,5],style="/",r=2,n=100):
```



```
>plot2d("x^2+y^2-1",level=-10:20,r=3,style="-",dl=0.1,n=100):
```



Sub bab 13

Menggambar Grafik Bilangan Kompleks

Bilangan kompleks secara visual dapat direpresentasikan sebagai sepasang angka (a, b) membentuk vektor pada diagram yang disebut diagram Argand, mewakili yang bidang kompleks. Sumbu-x adalah sumbu nyata dan sumbu-y adalah sumbu imajiner.

Menggambar kurva fungsi kompleks sendiri adalah proses visualisasi grafis dari fungsi matematika kompleks (yaitu fungsi yang melibatkan bilangan kompleks, yaitu bilangan dengan bagian real dan imajiner) berperilaku dalam koordinat kompleks. Hal tersebut memungkinkan untuk melihat bagaimana pola, bentuk, dan sifat dari fungsi kompleks tersebut.

Array bilangan kompleks juga dapat diplot. Kemudian titik-titik grid akan terhubung. Jika sejumlah garis kisi ditentukan (atau vektor garis kisi 1×2) dalam argumen cgrid, hanya garis kisi tersebut yang terlihat.

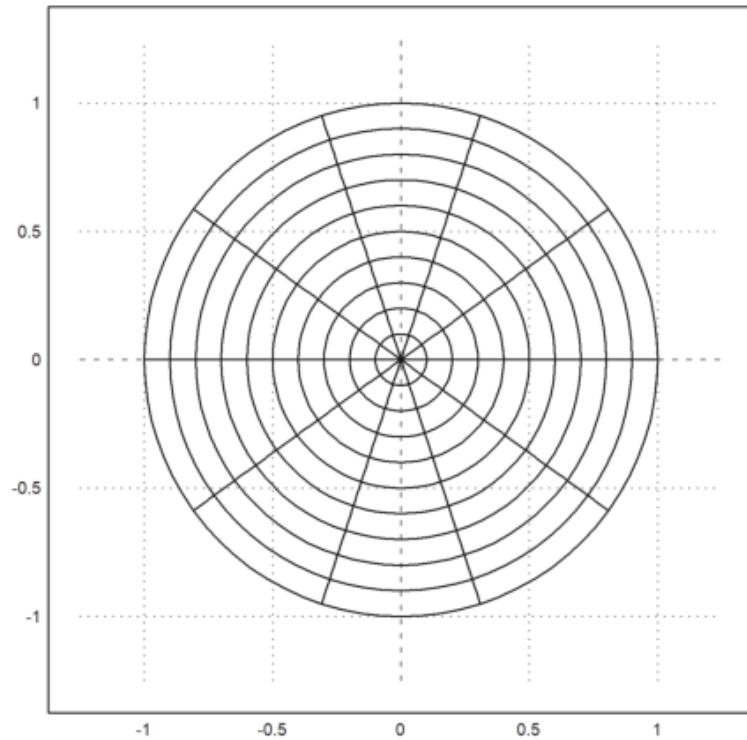
Matriks bilangan kompleks akan secara otomatis diplot sebagai kisi di bidang kompleks.

- > Definisi fungsi kompleks, mendefinisikan fungsi kompleks yang dianalisis atau digambarkan. Fungsi ini memiliki variabel kompleks z , yang melibatkan bagian real dan imajiner.
- > Selanjutnya kita dapat menggunakan fungsi linspace. Fungsi linspace sendiri adalah salah satu fungsi yang umum digunakan dalam pemrograman, terutama dalam konteks pemrograman numerik dan ilmu data. Ini sering digunakan untuk menghasilkan urutan nilai dalam rentang tertentu dengan jumlah titik yang sama di antara dua titik ujungnya. Penggunaannya tidak terbatas pada pemrosesan sinyal atau elektromagnetik, tetapi bisa digunakan dalam berbagai konteks di mana Anda perlu membuat urutan nilai.
- > Penentuan rentang, memilih rentang nilai z yang ingin ditampilkan di dalam plot. Rentang ini mencakup wilayah kompleks tertentu yang ingin diamati.
- > Menggunakan sintaks plot2d.
- > Penyesuaian plot, mengubah plot sesuai yang diinginkan (mengubah warna, format (style), dan sebagainya).

Dalam contoh berikut, kami memplot gambar lingkaran satuan di bawah fungsi eksponensial. Parameter cgrid menyembunyikan beberapa kurva grid.

Contoh

```
>aspect(); r=linspace(0,1,50); a=linspace(0,2pi,80)'; z=r*exp(I*a);...  
>plot2d(z,a=-1.25,b=1.25,c=-1.25,d=1.25,cgrid=10):
```



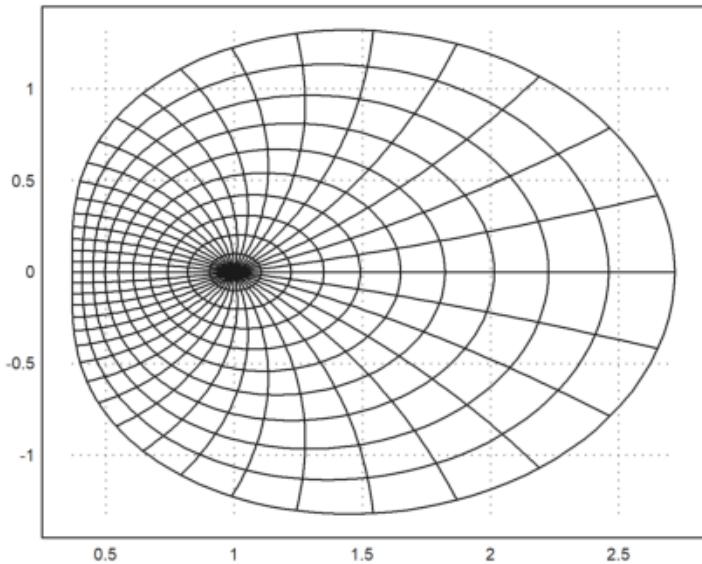
Penjelasan sintaks

`z` : sebuah ekspresi atau fungsi yang akan digambar dalam koordinat kompleks.
`a,b,c,d` : parameter-parameter yang digunakan untuk mengatur jendela tampilan (viewport) dalam koordinat kompleks. Parameter-parameter ini akan menentukan rentang sumbu x dan sumbu y yang akan ditampilkan di dalam plot.
`cgrid` : parameter ini mengontrol tampilan grid pada plot. Jika `cgrid=n`, maka grid akan ditampilkan, jika `cgrid=0`, maka grid akan disembunyikan.

Bentuk lain

```

>aspect(1.25); r=linspace(0,1,50); a=linspace(0,2pi,200)'; z=r*exp(I*a);
>plot2d(exp(z),cgrid=[40,10]):
```



Penjelasan :

Perintah tersebut merupakan perintah untuk menggambar kurva dari fungsi kompleks eksponensial "exp(z)" dalam koordinat kompleks. Dalam perintah tersebut juga menggunakan parameter cgrid dengan nilai [40,10] untuk mengatur grid pada plot.

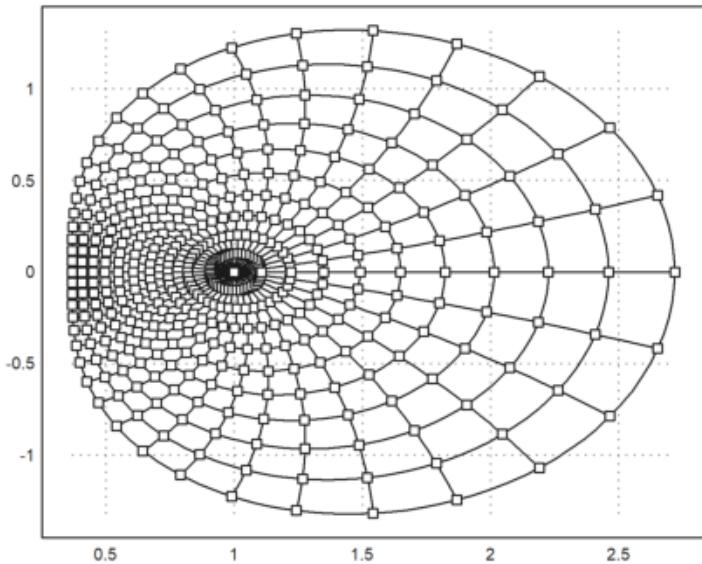
Dalam sintaks ini,

`exp(z)` : fungsi eksponensial kompleks yang akan digambar

`cgrid=[40,10]` : mengatur grid pada plot. cgrid tersebut adalah jumlah garis grid yang akan digunakan pada sumbu x dan sumbu y. Nah di dalam plot ini, akan ada 40 garis grid pada sumbu x dan 10 grid pada sumbu y.

Bentuk lain

```
>r=linspace(0,1,10); a=linspace(0,2pi,40)'; z=r*exp(I*a);
>plot2d(exp(z),>points,>add):
```



Sebuah vektor bilangan kompleks secara otomatis diplot sebagai kurva pada bidang kompleks dengan bagian real dan bagian imajiner.

Penjelasan :

Perintah plot2d di atas adalah perintah untuk menggambar kurva fungsi kompleks dalam koordinat kompleks, namun dengan opsi yang berbeda,

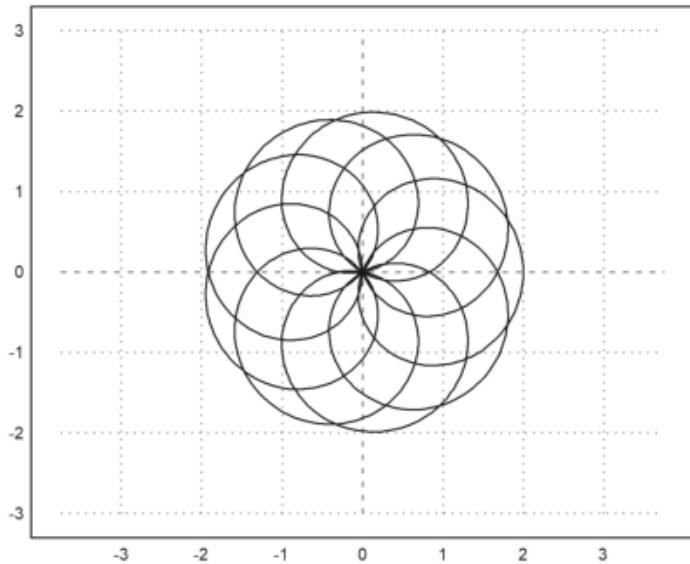
`exp(z)`: fungsi kompleks yang akan digambar

`>points` : opsi ini mengubah cara plot untuk dilakukan. Dengan menggunakan `>points`, plot ini akan menggunakan titik-titik diskrit untuk merepresentasikan fungsi ke dalam bentuk titik,titik

`>add` : sintaks ini menginstrusikan perintah untuk menambahkan plot ini ke plot sebelumnya jika ada.

Contoh

```
>t=linspace(0,2pi,1000); ...
>plot2d(exp(I*t)+exp(10*I*t),r=3):
```



Penjelasan :

Perintah plot2d di atas menggambarkan kurva dari fungsi kompleks yang diberikan dalam koordinat kompleks dengan parameter-parameter tertentu.

Sintaks yang digunakan yaitu,

$\exp(I*t)+\exp(10*I*t)$: fungsi kompleks yang akan digambar. Fungsi ini terdiri dari dua bagian yang masing-masing merupakan fungsi kompleks eksponensial. Dengan 10 adalah berapa kali putaran dalam gambar tersebut.

r : parameter r digunakan untuk menentukan rentang nilai dari variabel t . Dalam contoh ini, $r=3$, yaitu mengatur rentang nilai t dari 3 hingga 3.

Sub Bab 14

Menggambar Daerah Yang Dibatasi Kurva

Plot data benar-benar poligon. Kita juga dapat memplot kurva atau kurva terisi.

Pada subtopik sebelumnya telah kita ketahui dan pelajari bersama bahwa EMT dapat melakukan visualisasi plot mulai dari bentuk ekspresi langsung hingga plot dari fungsi-fungsi. Pada subtopik ini merupakan kelanjutan dari subtopik sebelumnya, bahwa kita dapat membentuk/menggambar daerah dari perpotongan beberapa kurva yang telah didefinisikan. Hal ini dapat bermanfaat untuk membantu dalam menyelesaikan permasalahan dalam matematika, salah satu contohnya seperti optimasi program linear, dimana disajikan beberapa fungsi-fungsi kendala beserta dengan fungsi tujuannya dan perlu divisualisasikan dalam bentuk grafik untuk melihat dimana letak daerah layaknya untuk menentukan nilai optimum.

Dalam EMT ada beberapa perintah yang digunakan untuk menggambar daerah yang dibatasi oleh beberapa kuva, diantaranya yaitu:

- plot2d

Digunakan untuk melakukan plotting.

- filled=true

Digunakan untuk memberikan isian/arsiran pada daerah/area di bawah kurva saat plotting.

- style="..."

Digunakan untuk memilih gaya kurva yang akan digunakan saat

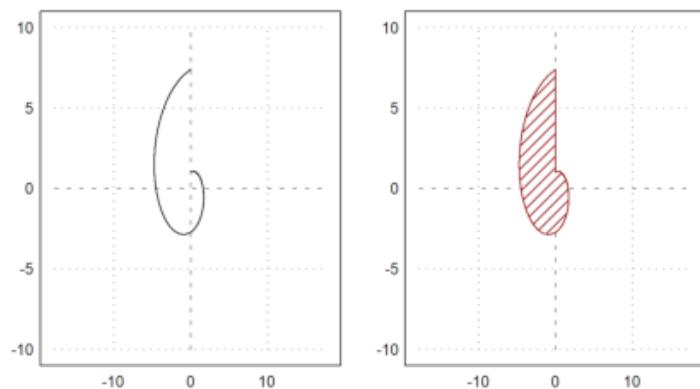
plotting. Anda dapat memilih dari beberapa gaya, seperti "", "/", "\", atau "-". Dan hal ini mempengaruhi tampilan daerah kurva yang terbentuk.

- fillcolor

Digunakan untuk menentukan warna isian yang akan digunakan untuk mengisi area di bawah kurva.

Contoh

```
>t=linspace(0,2pi,1000); // parameter for curve  
>x=sin(t)*exp(t/pi); y=cos(t)*exp(t/pi); // x(t) and y(t)  
>figure(1,2); aspect(16/9)  
>figure(1); plot2d(x,y,r=10); // plot curve  
>figure(2); plot2d(x,y,r=10,>filled,style="/",fillcolor=red); // fill curve  
>figure(0):
```



Penjelasan:

- `t=linspace(0,2pi,1000);`

Pada langkah pertama yaitu mendefinisikan parameter t sebagai serangkaian 1000 titik antara 0 dan 2pi. Parameter t ini akan digunakan sebagai parameter untuk menggambar kurva.

- `x=sin(t)*exp(t/pi); y=cos(t)*exp(t/pi); // x(t) and y(t)`

Kemudian kita definisikan dua vektor x dan y yang merupakan koordinat x dan y dari kurva yang akan digambar. Fungsi

$$\sin(t) * \exp(t/\pi)$$

digunakan untuk menghitung komponen x ($x(t)$), dan

$$\cos(t) * \exp(t/\pi)$$

digunakan untuk menghitung komponen y ($y(t)$) dari kurva.

- `figure(1,2); aspect(16/9)`

Perintah ini digunakan untuk mengatur tampilan gambar. Perintah `figure(1,2)` digunakan membuat dua gambar (1 dan 2) dalam satu jendela gambar. Dan perintah `aspect(16/9)` mengatur rasio aspek gambar menjadi 16:9, yang mempengaruhi bentuk dan ukuran gambar yang akan digambar.

- `figure(1); plot2d(x,y,r=10); // membuat plot kurva`

Perintah ini memilih gambar pertama (1) dan menggunakan perintah `plot2d` untuk menggambar kurva yang dihitung sebelumnya. Parameter `r=10` mengatur lebar garis plot. Ini menghasilkan kurva tanpa adanya isi atau arsiran di dalamnya.

- `figure(2); plot2d(x,y,r=10,>filled,style="/" ,fillcolor=red); // fill curve`

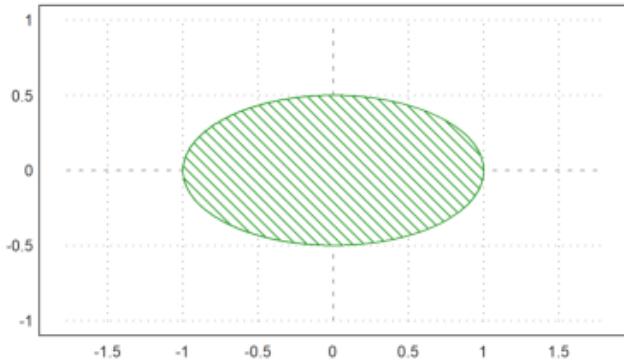
Selanjutnya pada perintah ini beralih ke gambar kedua (2) dan menggunakan perintah `plot2d` lagi untuk menggambar kurva yang sama dengan pengisian area di bawahnya. Perintah `>filled` digunakan untuk mengisi area di bawah kurva, `style="/"` digunakan untuk mengatur gaya garis menjadi garis miring, dan `fillcolor=red` digunakan untuk mengatur warna isian menjadi merah.

-`figure(0):`

Baris perintah ini digunakan untuk mengakhiri gambar dan kembali ke tampilan biasa tanpa gambar. Ini berfungsi untuk menyelesaikan proses penggambaran.

Contoh

```
>x=linspace(0,2pi,1000); plot2d(cos(x),sin(x)*0.5,r=1,>filled,style="\"):
```



Penjelasan:

- `x=linspace(0,2pi,100);`

Mendefinisikan vektor x dengan menggunakan perintah linspace. linspace digunakan untuk membuat vektor dengan 100 titik yang secara merata tersebar antara 0 dan 2phi. Dalam konteks ini, vektor x akan digunakan sebagai parameter saat menggambar kurva.

- `plot2d(cos(x),sin(x)*0.5,r=1,>filled,style="\"):`

Ini merupakan perintah utama yang digunakan untuk menggambar plot. Perintah ini memiliki beberapa parameter sebagai berikut:

> `cos(x)` adalah komponen x dari kurva. Ini adalah hasil dari fungsi kosinus yang dihitung pada vektor x.

> `sin(x)*0.5` adalah komponen y dari kurva. Ini adalah hasil dari fungsi sinus yang dihitung pada vektor x dan kemudian dikalikan dengan 0,5, yang mengubah amplitudonya.

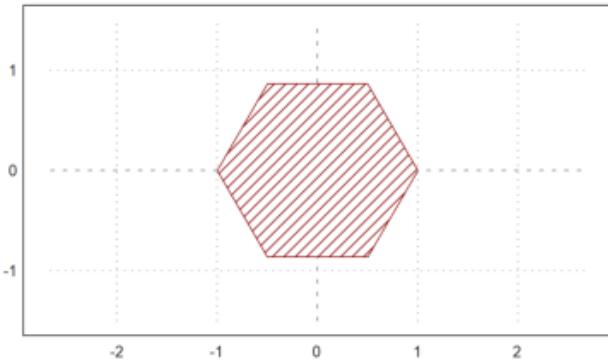
> `r=1` mengatur lebar garis plot menjadi 1.

> `filled` digunakan untuk mengisi area di bawah kurva, sehingga menciptakan daerah yang terisi.

> `style="\\"` mengatur gaya garis kurva untuk membentuk garis miring yang gunanya menutupi semua bagian kurva dengan garis miring.

Contoh

```
>t=linspace(0,2pi,6); ...
>plot2d(cos(t),sin(t),>filled,style="/",fillcolor=red,r=1.5):
```



Penjelasan:

- t=linspace(0,2pi,6); ...

Pada perintah ini, kita definisikan vektor t dengan menggunakan

perintah linspace. linspace digunakan untuk membuat vektor dengan 6 titik yang terletak secara merata antara 0 dan 2pi. Dalam konteks ini, vektor t akan digunakan sebagai parameter saat menggambar kurva.

- plot2d(cos(t),sin(t),>filled,style="/",fillcolor=red,r=1.5);

Ini adalah perintah utama yang digunakan untuk menggambar plot.

Perintah ini memiliki beberapa parameter sebagai berikut:

> cos(t) adalah komponen x dari kurva.

> sin(t) adalah komponen y dari kurva.

> filled digunakan untuk mengisi area di bawah kurva, sehingga menciptakan bentuk yang terisi. Ini berarti daerah di bawah kurva akan diwarnai.

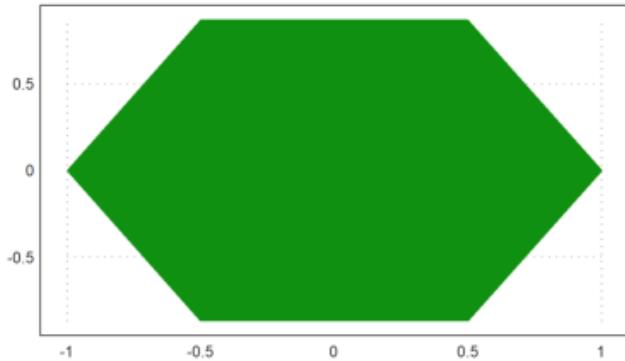
> style="/" mengatur gaya garis kurva menjadi garis miring ("/").

> fillcolor=orange mengatur warna isian daerah di bawah kurva menjadi oranye.

> r=1.5 mengatur lebar garis plot menjadi 1.5.

Contoh

```
>t=linspace(0,2pi,6); plot2d(cos(t),sin(t),>filled,style="#" );
```



Penjelasan:

- `t=linspace(0,2pi,6);`

Pada perintah ini, kita definisikan vektor t dengan menggunakan perintah linspace. linspace digunakan untuk membuat vektor dengan 6 titik yang terletak secara merata antara 0 dan 2phi. Dalam konteks ini, vektor t akan digunakan sebagai parameter saat menggambar kurva.

- `plot2d(cos(t),sin(t),>filled,style="");`

Ini adalah perintah utama yang digunakan untuk menggambar plot. Perintah ini memiliki beberapa parameter sebagai berikut:

> `cos(t)` adalah komponen x dari kurva.

> `sin(t)` adalah komponen y dari kurva.

> `filled` digunakan untuk mengisi area di bawah kurva, sehingga menciptakan bentuk yang terisi. Ini berarti daerah di bawah kurva akan diisi dengan warna atau pola tertentu.

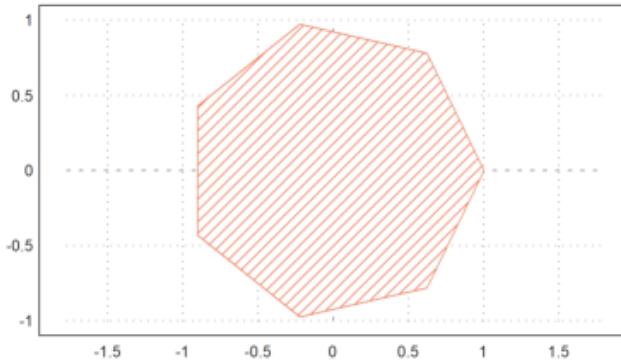
> `style=""` mengatur isian kurva menjadi warna solid dengan menggunakan simbol tanda pagar ("")

Pada contoh ini tidak ada perintah untuk mengatur warna, maka warna yang dihasilkan pada plot ini akan mengikuti pada warna yang disetting pada bagian sebelumnya.

Contoh

Contoh lainnya adalah segi empat, yang kita buat dengan 7 titik pada lingkaran satuan.

```
>t=linspace(0,2pi,7); ...  
>plot2d(cos(t),sin(t),r=1,>filled,style="/",fillcolor=orange):
```



Penjelasan:

- t=linspace(0,2pi,7);

Fungsi linspace digunakan untuk membuat array berisi sejumlah nilai yang merata dalam rentang tertentu. Dalam hal ini, rentangnya adalah dari 0 hingga 2π (dua kali nilai π) dan sebanyak 7 titik akan dihasilkan. Ini akan digunakan sebagai sudut dalam koordinat polar untuk menggambarkan data.

- plot2d(cos(t),sin(t),r=1,>filled,style="/",fillcolor=orange);

Ini adalah perintah untuk melakukan plotting data. Terdapat beberapa argumen di sini:

> cos(t): Ini adalah nilai kosinus dari setiap elemen dalam array t. Ini akan digunakan sebagai komponen sumbu Y dalam koordinat polar.

> sin(t): Ini adalah nilai sinus dari setiap elemen dalam array t. Ini akan digunakan sebagai komponen sumbu X dalam koordinat polar.

> r=1: Ini adalah argumen opsional yang menentukan radius plot. Dalam hal ini, radiusnya diatur menjadi 1.

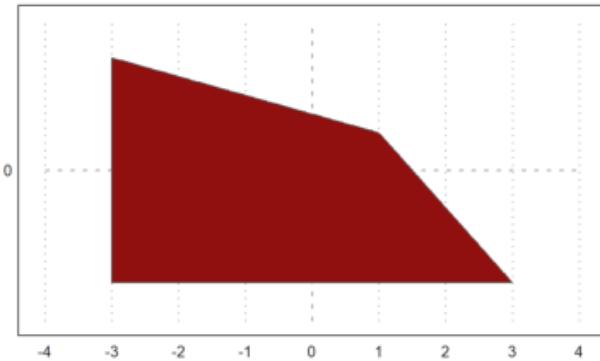
> filled: Ini adalah argumen yang menginstruksikan untuk mengisi area di dalam kurva plot.

> style="/": Ini adalah argumen yang menentukan gaya garis yang digunakan untuk plot. Di sini, garisnya akan berbentuk garis miring ("/").

> fillcolor=orange: Ini adalah argumen yang menentukan warna pengisian untuk area di dalam kurva plot. Dalam hal ini, warnanya diatur menjadi oren.

Contoh

```
>A=[2,1;1,2;-1,0;0,-1];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=4,level=[0;3],color=red,n=111):
```



Penjelasan:

- $A=[2,1;1,2;-1,0;0,-1];$

Ini adalah perintah untuk membuat matriks A. Matriks ini memiliki dimensi 4×2 , yang berarti memiliki 4 baris dan 2 kolom. Isinya adalah:

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

- function $f(x,y) := \max([x,y].A');$

Ini adalah perintah untuk mendefinisikan sebuah fungsi bernama $f(x, y)$. Fungsi ini mengambil dua argumen input, yaitu x dan y. Fungsi ini melakukan operasi berikut:

> $[x, y]$ menghasilkan vektor baris dengan elemen $[x, y]$.

> $[x, y].A'$ adalah perkalian dot (dot product) dari vektor baris $[x, y]$ dengan transpose dari matriks A.

> $\max([x, y].A')$ menghitung nilai maksimum dari hasil perkalian dot tersebut.

Dengan kata lain, fungsi ' $f(x, y)$ ' mengambil vektor ' $[x, y]$ ' sebagai input, mengalikannya dengan matriks ' A' , dan mengembalikan nilai maksimum dari hasil perkalian tersebut.

- `plot2d("f",r=4,level=[0;3],color=red,n=111);`

Ini adalah perintah untuk membuat plot 2D dari fungsi ' $f(x, y)$ ' yang telah didefinisikan. Rincian perintah ini adalah sebagai berikut:

> "f" adalah nama fungsi yang akan diplot.

> $r=4$ menentukan rentang plot, yang dalam hal ini adalah $[-4, 4]$ untuk kedua sumbu x dan y.

> $level=[0;3]$ menentukan tingkat kontur (contour levels) yang akan digunakan dalam plot. Ada dua tingkat kontur: 0 dan 3.

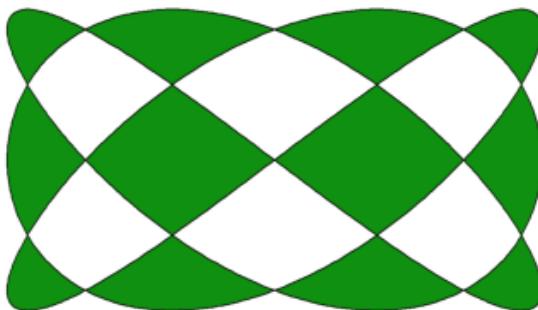
> $color=green$ mengatur warna kontur plot menjadi merah.

> $n=111$ mengendalikan jumlah titik yang digunakan dalam plot.

Hasilnya akan menjadi sebuah grafik kontur 2D dari fungsi ' $f(x, y)$ ' dengan kontur berwarna merah pada tingkat 0 dan 3, yang mencakup rentang -4 hingga 4 pada kedua sumbu x dan y.

Contoh

```
>t=linspace(0,2pi,1000); x=cos(3*t); y=sin(4*t);  
>plot2d(x,y,<grid,<frame,>filled):
```



Penjelasan:

- $t = \text{linspace}(0, 2\pi, 1000)$;

Ini adalah perintah untuk membuat vektor t yang berisi 1000 nilai yang merata terdistribusi antara 0 hingga 2π . Vektor t ini akan digunakan sebagai parameter waktu atau sudut dalam parameterisasi lingkaran.

$\text{linspace}(0, 2\pi, 1000)$ membuat 1000 titik antara 0 hingga 2π , memberikan sudut-sudut yang merata di sepanjang satu putaran lingkaran.

- $x = \cos(3*t)$; $y = \sin(4*t)$;

Ini adalah perintah untuk menghitung vektor x dan y yang menggambarkan lintasan dalam koordinat polar.

$> x = \cos(3*t)$; menghitung nilai x sebagai hasil dari fungsi kosinus dari 3 kali nilai t . Ini akan menghasilkan osilasi yang lebih cepat pada sumbu x .

$> y = \sin(4*t)$; menghitung nilai y sebagai hasil dari fungsi sinus dari 4 kali nilai t . Ini akan menghasilkan osilasi yang lebih cepat pada sumbu y .

- $\text{plot2d}(x, y, <\text{grid}, <\text{frame}, >\text{filled})$;

Ini adalah perintah untuk membuat plot dari vektor x dan y . Berikut adalah rincian perintah ini:

x adalah vektor yang digunakan sebagai data untuk sumbu x .

y adalah vektor yang digunakan sebagai data untuk sumbu y .

$<\text{grid}$ mengaktifkan garis-garis koordinat (grid) di latar belakang plot, membantu dalam visualisasi.

$<\text{frame}$ mengaktifkan bingkai (frame) di sekitar plot.

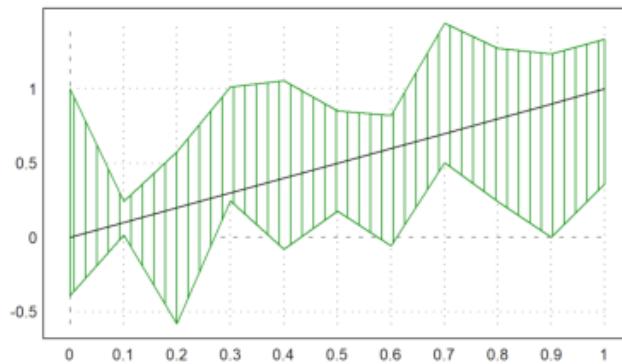
$>\text{filled}$ mengisi area di bawah kurva dengan warna, membuat plot menjadi lebih berwarna.

Contoh

Sebuah vektor interval diplot terhadap nilai x sebagai daerah terisi antara nilai interval bawah dan atas.

Ini dapat berguna untuk memplot kesalahan perhitungan. Tapi itu bisa juga digunakan untuk memplot kesalahan statistik.

```
>t=0:0.1:1; ...
>plot2d(t,interval(t-random(size(t)),t+random(size(t))),style="|");
>plot2d(t,t,add=true): ...
```



Penjelasan:

- $t = 0:0.1:1;$

Ini adalah perintah untuk membuat vektor t yang berisi nilai-nilai dari 0 hingga 1 dengan interval 0.1. Hasilnya adalah vektor $[0, 0.1, 0.2, 0.3, \dots, 0.9, 1]$.

- `plot2d(t, interval(t - random(size(t)), t + random(size(t))), style="|");`

Ini adalah perintah untuk membuat plot pertama. Rincian perintah ini adalah sebagai berikut:

> `interval(t - random(size(t)), t + random(size(t)))` adalah interval yang digunakan untuk menggambar "garis" pada plot. Setiap titik pada sumbu x (t) akan dihubungkan oleh dua garis vertikal yang dibuat secara acak di sekitar titik tersebut menggunakan `random(size(t))`. Hasilnya adalah plot dengan garis-garis vertikal yang mewakili interval acak di sekitar setiap titik pada sumbu x.

> `style="|"` mengatur gaya plot menjadi garis vertikal ("|").

- `plot2d(t, t, add=true);`

Ini adalah perintah untuk membuat plot kedua dan menambahkannya ke dalam plot yang sudah ada dari perintah sebelumnya. Rincian perintah ini adalah sebagai berikut:

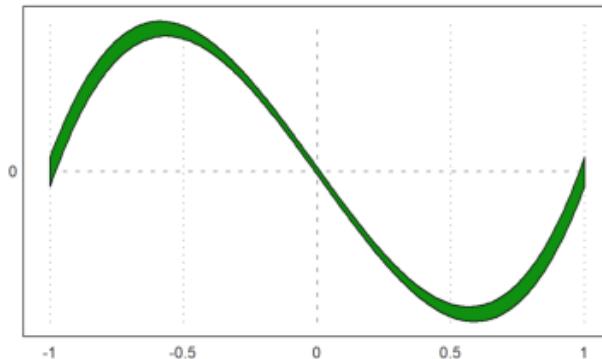
> t adalah sumbu x dan y plot ini, sehingga plot ini akan menjadi plot garis diagonal dengan kemiringan 45 derajat.

> `add=true` digunakan untuk menambahkan plot ini ke dalam plot sebelumnya, sehingga kedua plot akan ditampilkan dalam satu plot yang sama.

Contoh

Jika x adalah vektor yang diurutkan, dan y adalah vektor interval, maka `plot2d` akan memplot rentang interval yang terisi dalam bidang. Gaya isian sama dengan gaya poligon.

```
>t=-1:0.01:1; x=~t-0.01,t+0.01~; y=x^3-x;
>plot2d(t,y):
```



Penjelasan:

- $t = -1:0.01:1;$

Ini adalah perintah untuk membuat vektor t yang berisi nilai-nilai dari -1 hingga 1 dengan interval 0.01 . Hasilnya adalah vektor t yang berisi nilai-nilai seperti $[-1, -0.99, -0.98, \dots, 0.99, 1]$. Vektor t ini akan digunakan sebagai sumbu x pada plot.

- $x = t - 0.01, t + 0.01 ;$

Ini adalah perintah yang menghitung vektor x . Tanda \sim digunakan di sini untuk mendefinisikan dua interval, yaitu $[t - 0.01, t + 0.01]$. Ini menghasilkan vektor x yang memiliki dua interval, satu yang kurang dari $t - 0.01$ dan satu yang lebih dari $t + 0.01$.

- $y = x^3 - x;$

Ini adalah perintah yang menghitung vektor y sebagai fungsi dari x . Fungsi ini menghitung nilai y dengan memasukkan setiap nilai x ke dalam rumus $x^3 - x$.

- `plot2d(t, y);`

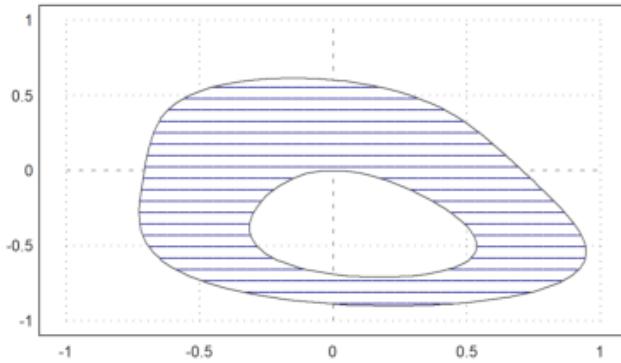
Ini adalah perintah untuk membuat plot dari fungsi y sebagai fungsi dari t . Rincian perintah ini adalah sebagai berikut:

> t adalah sumbu x pada plot, yang berisi vektor t yang telah didefinisikan sebelumnya.

> y adalah sumbu y pada plot, yang berisi vektor y yang dihitung dari rumus $x^3 - x$.

Contoh

```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr,level=[0;1],style="-",color=blue): // 0 <= f(x,y) <= 1
```



Penjelasan:

- expr := "2*x^2+x*y+3*y^4+y";

Ini adalah perintah untuk mendefinisikan ekspresi matematika yang disimpan dalam variabel expr. Ekspresi ini merupakan suatu fungsi $f(x, y)$ yang tergantung pada dua variabel, yaitu x dan y . Ekspresi ini memiliki bentuk matematika yang terdiri dari berbagai suku, seperti kuadrat dari x , perkalian $x*y$, kuadrat dari y , dan lainnya.

- plot2d(expr, level=[0;1], style="-", color=blue);

Ini adalah perintah untuk membuat plot dari fungsi $f(x, y)$ yang telah didefinisikan sebelumnya. Berikut adalah rincian perintah ini:

> expr adalah ekspresi yang akan digunakan sebagai fungsi yang akan diplotkan. Dalam hal ini, ekspresi $2*x^2+x*y+3*y^4+y$ adalah fungsi $f(x, y)$ yang telah didefinisikan sebelumnya.

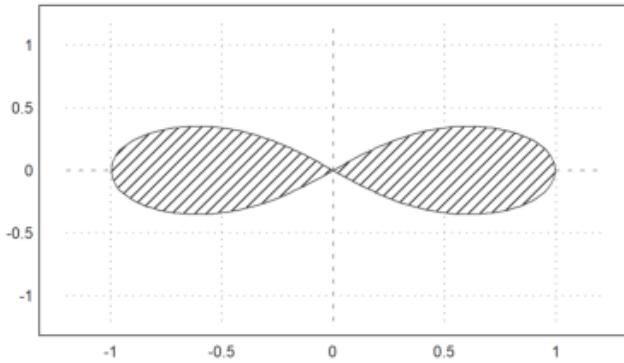
> level=[0;1] mengatur tingkat kontur (contour levels) yang akan digunakan dalam plot. Dalam hal ini, tingkat kontur adalah 0 hingga 1, yang berarti plot akan menunjukkan wilayah di mana $f(x, y)$ memiliki nilai antara 0 hingga 1.

> style="-" mengatur gaya plot menjadi garis berjenis -, yang akan menghasilkan plot kontur.

> color=blue mengatur warna garis plot menjadi biru.

Contoh

```
>plot2d("(x^2+y^2)^2-x^2+y^2", r=1..2, level=[-1;0], style="/") :
```



Penjelasan:

```
plot2d("(x^2+y^2)^2-x^2+y^2", r=1.2, level=[-1;0], style="/");
```

Ini adalah perintah untuk membuat plot dari fungsi matematika yang didefinisikan dalam bentuk string: " $(x^2+y^2)^2-x^2+y^2$ ". Fungsi ini tergantung pada dua variabel, yaitu x dan y .

$(x^2+y^2)^2-x^2+y^2$ adalah rumus dari fungsi matematika yang akan diplotkan.

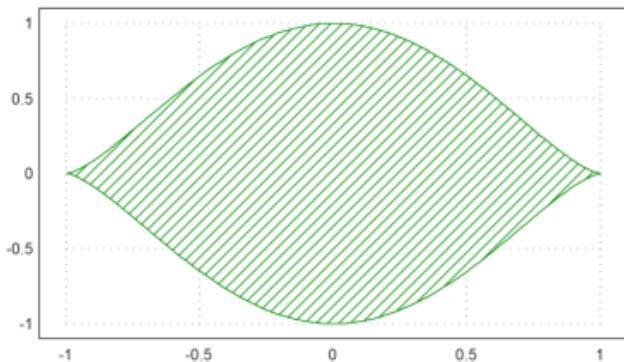
$r=1.2$ mengatur rentang (range) plot untuk kedua sumbu x dan y . Dalam hal ini, rentangnya adalah $[-1.2, 1.2]$, yang berarti plot akan berada dalam wilayah ini.

$level=[-1;0]$ mengatur tingkat kontur (contour levels) yang akan digunakan dalam plot. Dalam hal ini, ada dua tingkat kontur: -1 dan 0. Ini akan menentukan wilayah kontur dalam plot.

$style="/"$ mengatur gaya plot menjadi garis miring ("/"). Ini akan menghasilkan plot dengan garis-garis miring yang menggambarkan kontur fungsi.

Contoh

```
>plot2d("cos(x)", "sin(x)^3", xmin=0, xmax=2pi, >filled, style="/") :
```



Penjelasan:

```
plot2d("sin(x)^3", "cos(x)", xmin=0, xmax=2*pi, >filled, style="/");
```

Ini adalah perintah untuk membuat plot dari dua fungsi matematika, yaitu $\sin(x)^3$ dan $\cos(x)$, dalam satu plot yang sama. Berikut adalah rincian perintah ini:

" $\sin(x)^3$ " adalah ekspresi pertama yang akan diplotkan. Ini adalah fungsi trigonometri $\sin(x)$ yang dipangkatkan tiga. Fungsi ini tergantung pada variabel x .

" $\cos(x)$ " adalah ekspresi kedua yang akan diplotkan. Ini adalah fungsi trigonometri $\cos(x)$. Fungsi ini juga tergantung pada variabel x .

$x_{\min}=0$ dan $x_{\max}=2\pi$ mengatur rentang (range) plot untuk sumbu x dari 0 hingga 2π . Ini adalah rentang yang akan ditampilkan dalam plot.

$>\text{filled}$ mengisi area di bawah kurva fungsi dengan warna, sehingga area di bawah kurva fungsi akan diisi dengan warna.

$\text{style}="/"$ mengatur gaya plot menjadi garis miring (" \backslash "). Ini akan menghasilkan plot dengan garis-garis miring.

Sub Bab 15

Menggambar Segi Banyak

Data plot merupakan poligon atau segi banyak. Kita juga dapat membuat kurva atau mengisi kurva.

Fungsi perintah yang digunakan untuk menggambar segi banyak atau poligon.

Membentuk poligon dengan fungsi:

```
x=linspace(0,2pi,n); plot2d(cos(x),sin(x),r=1,>filled,style="...");
```

atau

```
x=linspace(0,2pi,n); plot2d(sin(x),cos(x),r=1,>filled,style="...",fillcolor=red);
```

Keterangan

- filled=true, mengisi plot.

- style="...": Pilih dari "", "/", "\\", "\\/" dan gaya gaya lainnya.

- fillcolor: untuk memberikan warna.

Warna isian ditentukan oleh argumen "fillcolor", dan pada <outline opsional mencegah menggambar batas untuk semua gaya kecuali yang default.

Poligon dalam EMT dapat digambar dengan fungsi maksimal. Dengan fungsi maksimal ini, poligon yang dihasilkan dapat berupa poligon tak beraturan.

```
A=[2,1;1,2;-1,0;0,-1];
```

```
function f(x,y) := max([x,y].A');
```

```
plot2d("f",r=4,level=[0;3],color=green,n=111);
```

Keterangan:

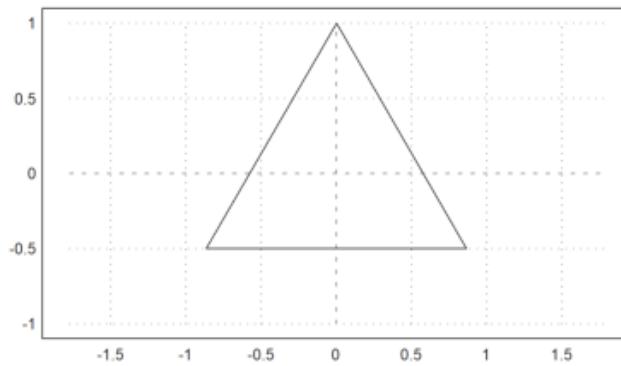
- A adalah titik koordinat dari poligon yang akan dibuat.

- "r" untuk menentukan ukuran bidang koordinat.

Berikut adalah himpunan nilai maksimal dari empat kondisi linear yang kurang dari atau sama dengan 3. Ini merupakan $A[k].v \leq 3$ untuk semua baris A. Untuk mendapatkan sudut yang bagus, kita menggunakan n yang relatif besar.

1. Menggambar Segitiga

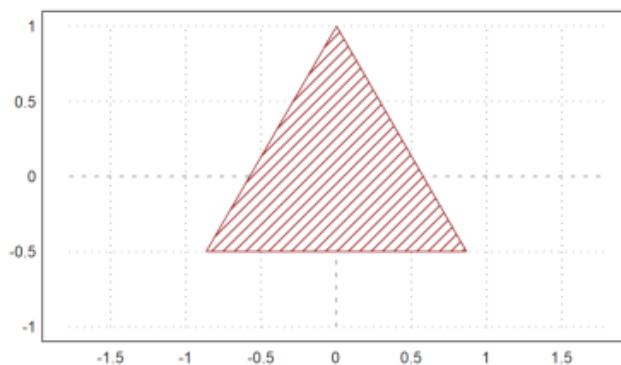
```
>x=linspace(0,2pi,3); ...
>plot2d(sin(x),cos(x),r=1):
```



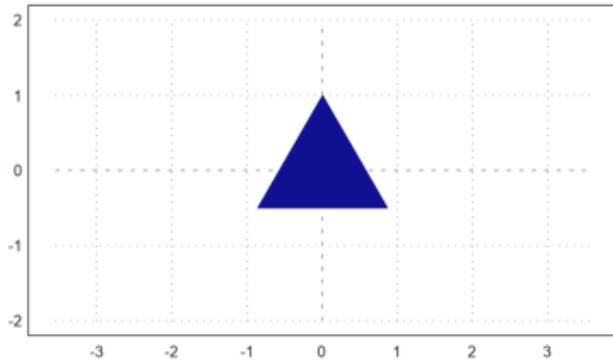
Segitiga diatas digambar dari kurva tertutup dengan 3 titik.

Kita dapat membuat segitiga dengan gaya yang berbeda-beda. Seperti pada contoh berikut ini.

```
>x=linspace(0,2pi,3); ...
>plot2d(sin(x),cos(x),>filled,style="/",fillcolor=red,r=1):
```



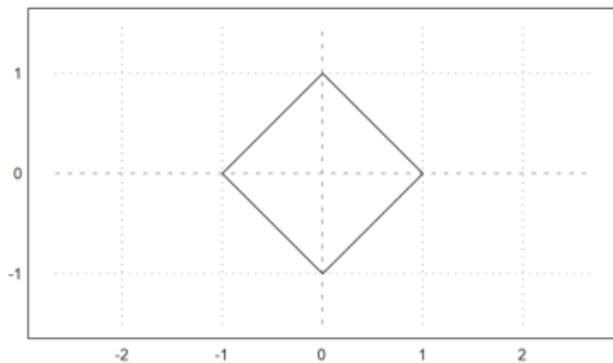
```
>x=linspace(0,2pi,3); ...
>plot2d(sin(x),cos(x),>filled,style="#",fillcolor=blue,r=2):
```



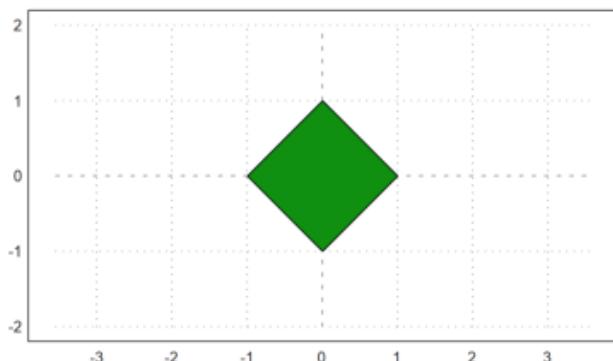
Dua gambar segitiga diatas memiliki gaya yang berbeda, dengan menggunakan fungsi perintah "style=". Gambar segitiga juga dapat dibuat dengan posisi yang berbeda, tergantung pada fungsi yang akan diplot.

2. Menggambar Segiempat

```
>x=linspace(0,2pi,4); ...
>plot2d(cos(x),sin(x),r=1.5):
```

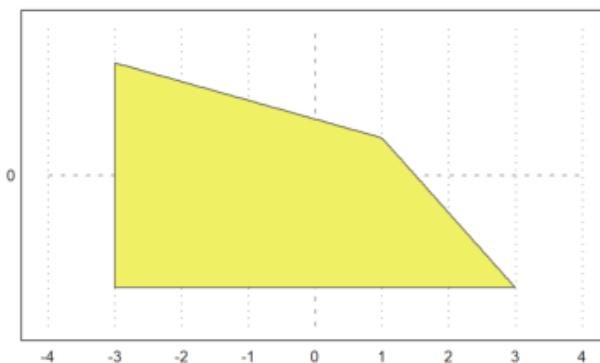


```
>x=linspace(0,2pi,4);
>plot2d(cos(x),sin(x),r=2,>filled,outline=1):
```

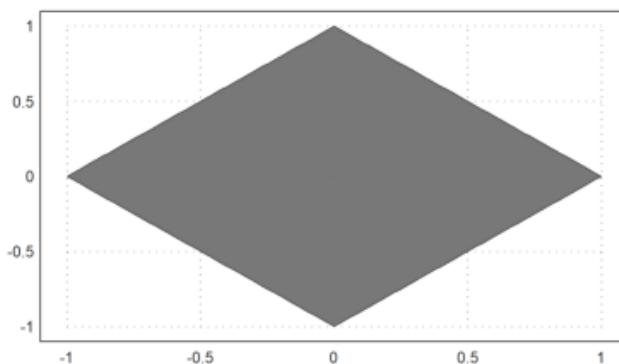


Gambar diatas merupakan salah satu contoh segiempat yang dapat digambar di EMT. Fungsi perintah yang digunakan masih sama seperti fungsi perintah untuk menggambar segitiga. Selain fungsi perintah diatas, untuk menggambar segi banyak, dapat menggunakan fungsi maksimum.

```
>A=[2,1;1,2;-1,0;0,-1];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=4,level=[0;3],color=yellow,n=111):
```



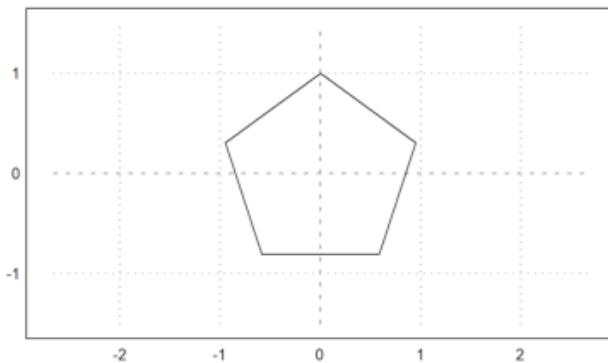
```
>A=[1,1;-1,1;-1,-1;1,-1];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=1,level=[0;1],color=gray,n=90):
```



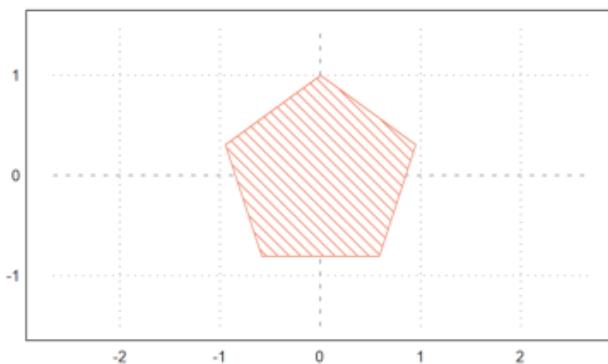
Dengan fungsi maksimal ini, kita dapat menggambar segiempat atau segi banyak sebarang.

3. Menggambar Segilima

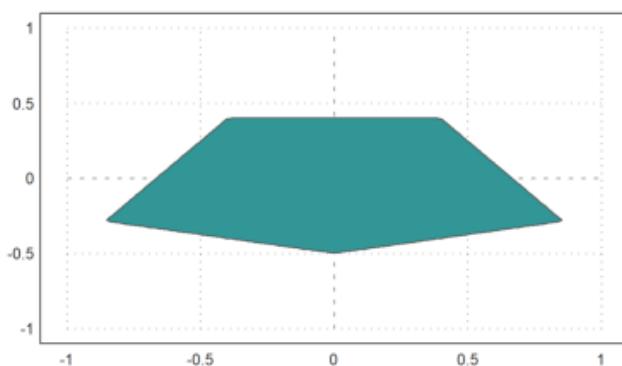
```
>t=linspace(0,2pi,5); plot2d(sin(t),cos(t),r=1.5):
```



```
>t=linspace(0,2pi,5); ...
>plot2d(sin(t),cos(t),r=1.5,>filled,style="\\",fillcolor=orange):
```

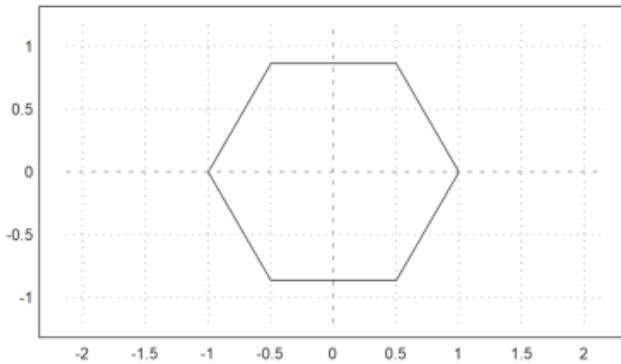


```
>A=[0,5;3,2;1,-4;-1,-4;-3,2];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=1,level=[0;2],color=cyan,n=111):
```

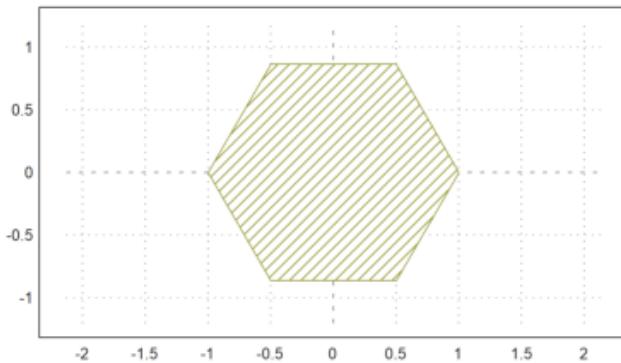


4. Menggambar Segienam

```
>t=linspace(0,2pi,6); ...
>plot2d(cos(t),sin(t),r=1.2):
```

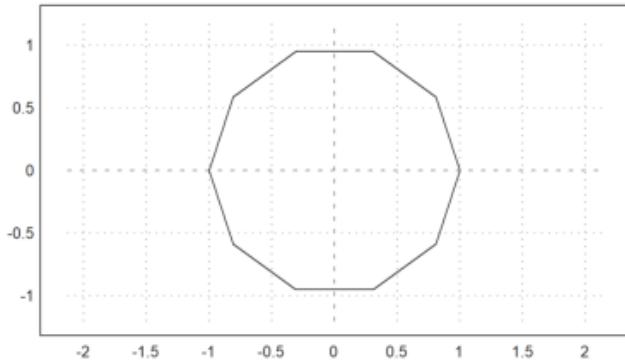


```
>t=linspace(0,2pi,6); ...
>plot2d(cos(t),sin(t),>filled,style="/",fillcolor=olive,r=1.2):
```

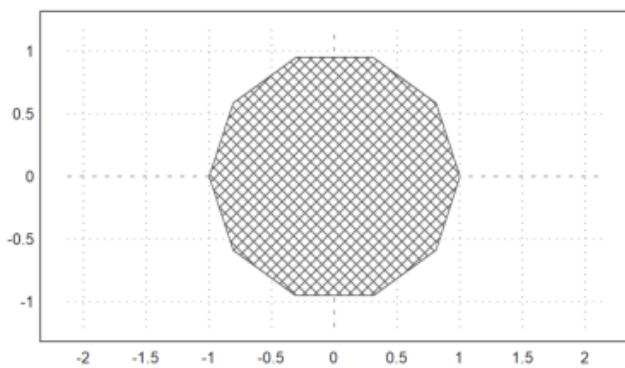


5. Menggambar dekagon

```
>t=linspace(0,2pi,10); ...
>plot2d(cos(t),sin(t),r=1.2):
```



```
>t=linspace(0,2pi,10); ...
>plot2d(cos(t),sin(t),>filled,style="/",fillcolor=darkgray,r=1.2):
```



BAB 4

KB Pekan 7-8 : Menggunakan EMT untuk menggambar grafik 3 dimensi (3D)

[a4paper,10pt]article eumat

Nama : Dassy Chintya N F

NIM : 22305141023

Kelas : Matematika E 2022

Menggambar Plot 3D dengan EMT

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

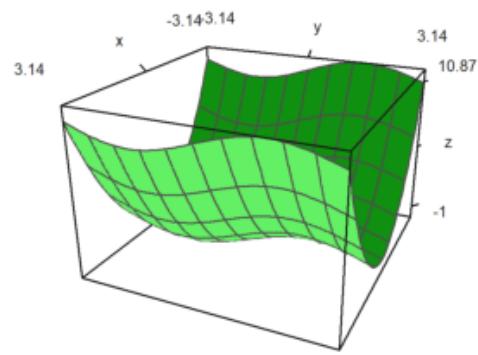
Euler menggambar fungsi tersebut menggunakan algoritma pengurutan untuk menyembunyikan bagian di latar belakang. Secara umum, Euler menggunakan proyeksi pusat. Standarnya adalah dari kuadran x-y positif menuju titik asal $x=y=z=0$, tetapi sudut=0° terlihat dari arah sumbu y. Sudut pandang dan ketinggian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

Euler dapat merencanakan

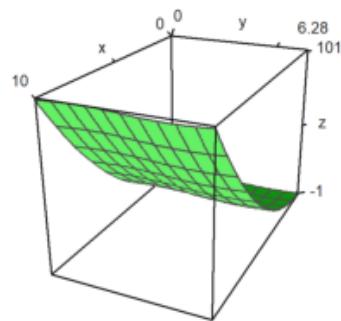
- permukaan dengan bayangan dan garis level atau rentang level,
- awan poin,
- kurva parametrik,
- permukaan implisit.

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

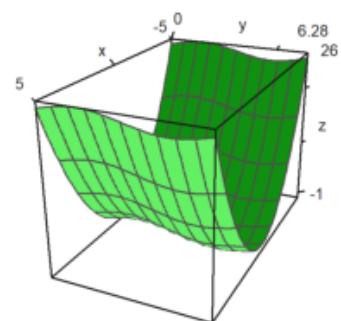
```
> aspect(1.5); plot3d("x^2+sin(y)",r=pi):
```



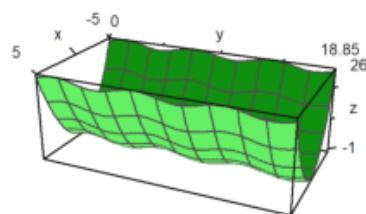
```
>aspect(1.5); plot3d("x^2+sin(y)",0,10,0,2*pi):
```



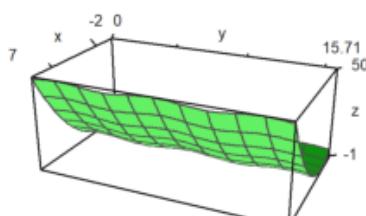
```
>aspect(1.5); plot3d("x^2+sin(y)",-5,5,0,2*pi):
```



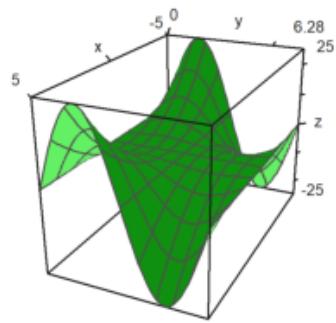
```
>aspect(1.5); plot3d("x^2+sin(y)", -5, 5, 0, 6*pi):
```



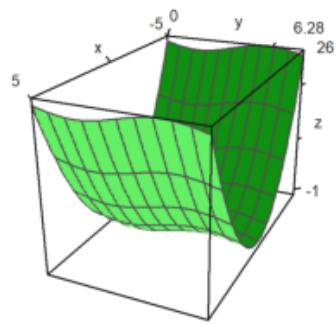
```
>aspect(1.5); plot3d("x^2+sin(y)", -2, 7, 0, 5*pi):
```



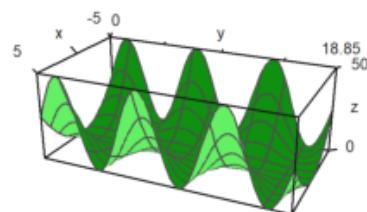
```
>aspect(1.5); plot3d("x^2*sin(y)", -5, 5, 0, 2*pi):
```



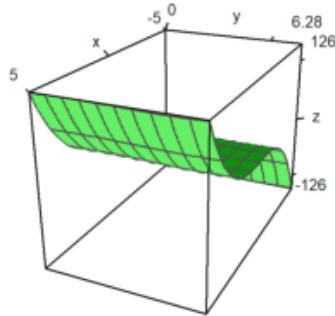
```
>aspect(1.5); plot3d("x^2-sin(y)",-5,5,0,2*pi):
```



```
>aspect(1.5); plot3d("x^2*(1+sin(y))",-5,5,0,6*pi):
```



```
>aspect(1.5); plot3d("x^3+sin(y)", -5, 5, 0, 2*pi):
```



Fungsi dua Variabel

Untuk grafik fungsi, gunakan

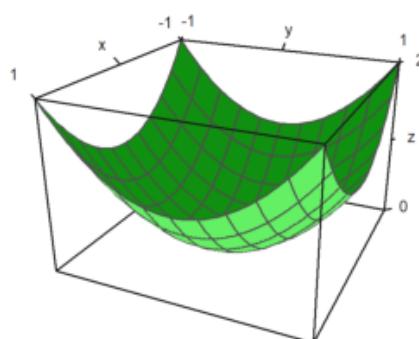
- ekspresi sederhana dalam x dan y,
- nama fungsi dari dua variabell
- atau matriks data.

Standarnya adalah kisi 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.

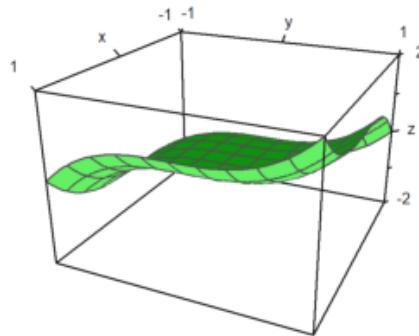
- n=40, n=[40,40]: jumlah garis kisi di setiap arah
- grid=10, grid=[10,10]: jumlah garis grid di setiap arah.

Kami menggunakan default n=40 dan grid=10.

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



```
>plot3d("x^3+y^3"):
```

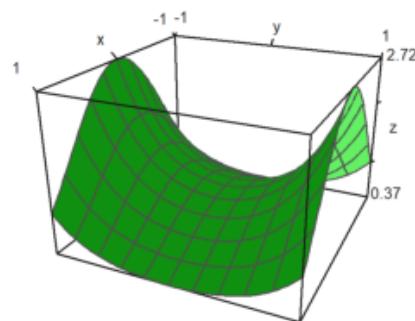


Interaksi pengguna dimungkinkan dengan >parameter pengguna. Pengguna dapat menekan tombol berikut.

- 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

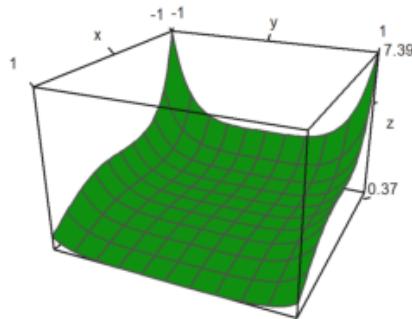
```
>plot3d("exp(-x^2+y^2)",>user, ...
> title="Turn with the vector keys (press return to finish)":
```

Turn with the vector keys (press return to finish)



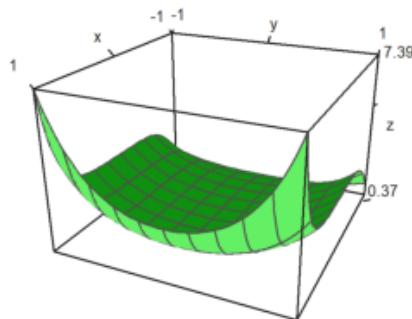
```
>plot3d("exp(-x^3+y^4)",>user, ...
> title="Turn with the vector keys (press return to finish)":
```

Turn with the vector keys (press return to finish)



```
>plot3d("exp(x^5+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.

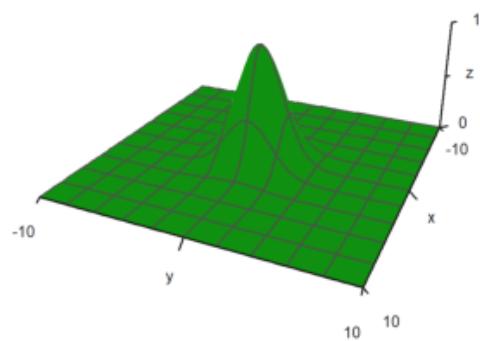
Ada beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik.

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

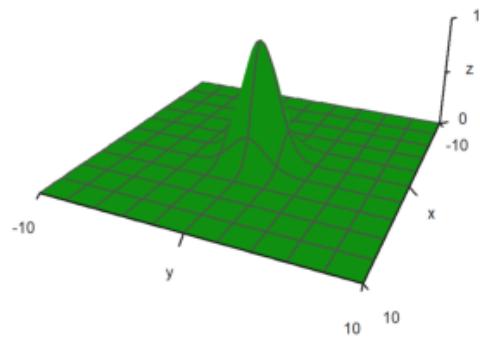
skala: angka atau vektor 1x2 untuk skala ke arah x dan y.

bingkai: jenis bingkai (default 1).

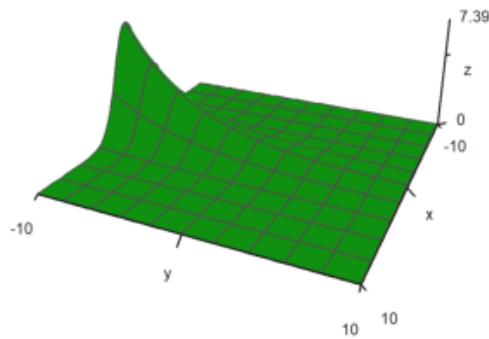
```
>plot3d("exp(-(x^2+y^2)/5)",r=10,n=80,fscale=4,scale=1.2,frame=3):
```



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



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



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.

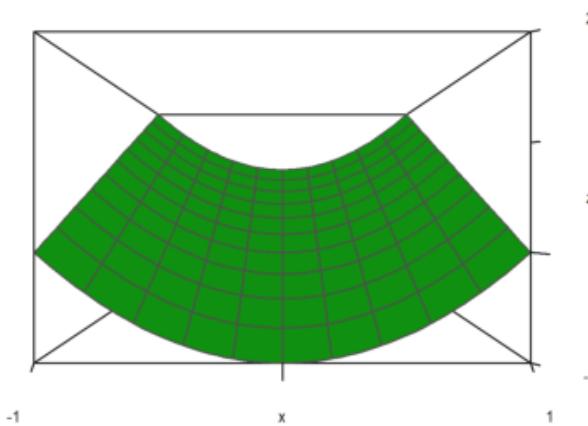
```
>view
```

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

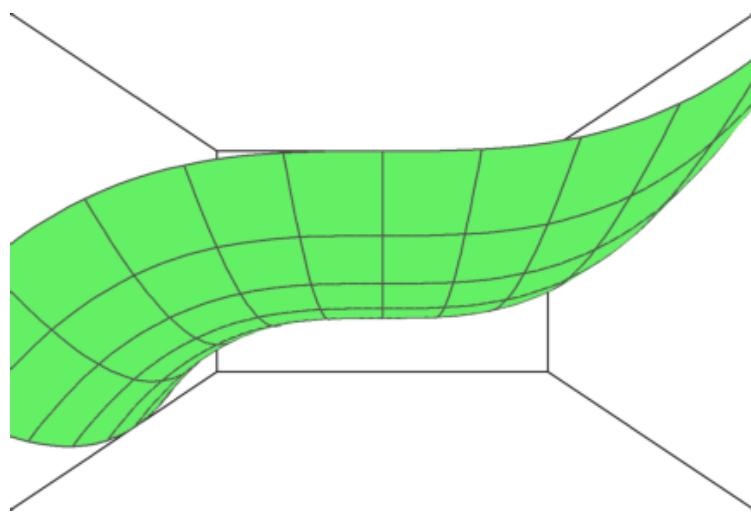
Jarak yang lebih dekat membutuhkan lebih sedikit zoom. Efeknya lebih seperti lensa sudut lebar.

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

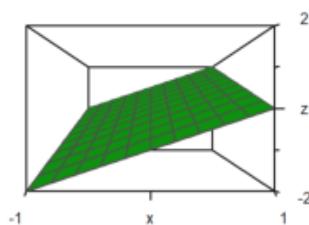
```
>plot3d("x^2+y", distance=3, zoom=2, angle=0, height=0) :
```



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

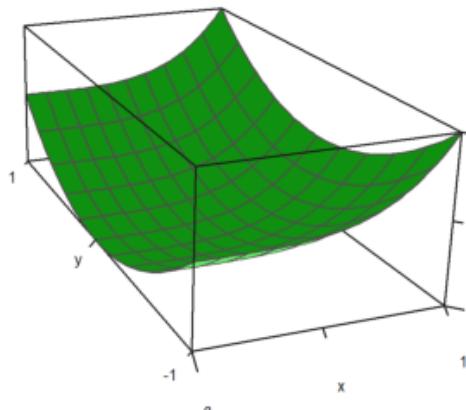


```
>plot3d("x+y",distance=3,zoom=1,angle=0,height=0) :
```

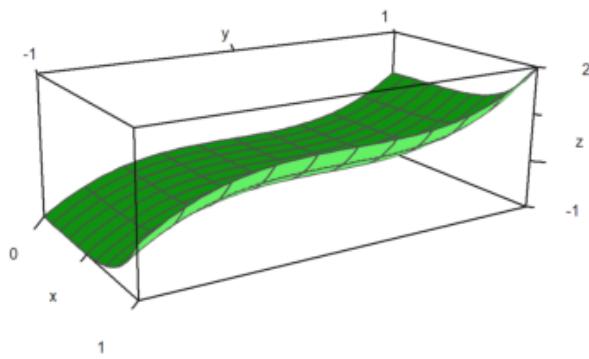


Plot terlihat selalu ke pusat kubus plot. Anda dapat memindahkan pusat dengan parameter tengah.

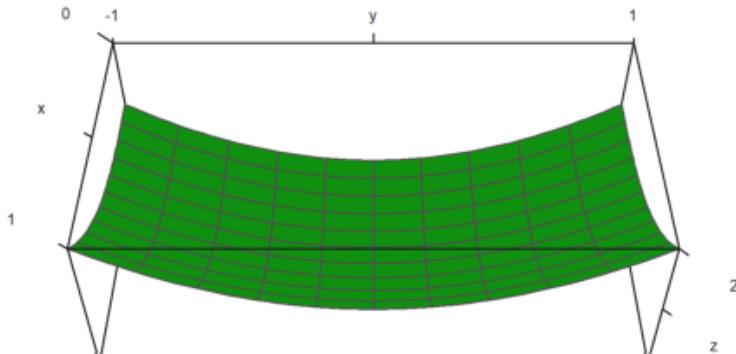
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
> center=[0.4,0,0],zoom=5) :
```



```
>plot3d("x^6+y^3", a=0, b=1, c=-1, d=1, angle=60°, height=15°, ...
>  center=[0.6, 0, 0], zoom=4):
```



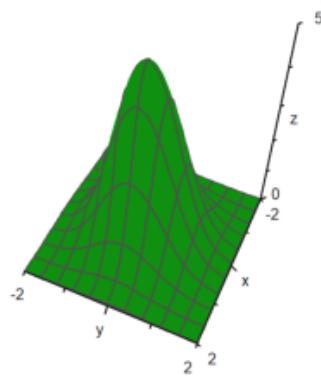
```
>plot3d("x^3+y^2", a=0, b=1, c=-1, d=1, angle=90°, height=45°, ...
>  center=[0.5, 0, 0], zoom=5):
```



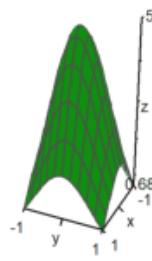
Plot diskalakan agar sesuai dengan kubus satuan untuk dilihat. Jadi tidak perlu mengubah jarak atau zoom tergantung pada ukuran plot. Namun, label mengacu pada ukuran sebenarnya.

Jika Anda mematikannya dengan `scale=false`, Anda harus berhati-hati, agar plot tetap pas dengan jendela plot, dengan mengubah jarak pandang atau zoom, dan memindahkan bagian tengahnya.

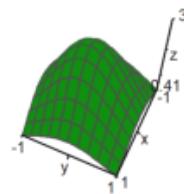
```
>plot3d("5*exp(-x^2-y^2)", r=2, <fscale, <scale, distance=13, height=50°, ...
> center=[0, 0, -2], frame=3):
```



```
>plot3d("5*exp(-x^2-y^2)", r=1, <fscale, <scale, distance=15, height=30°, ...
> center=[0, 0, -2], frame=3):
```

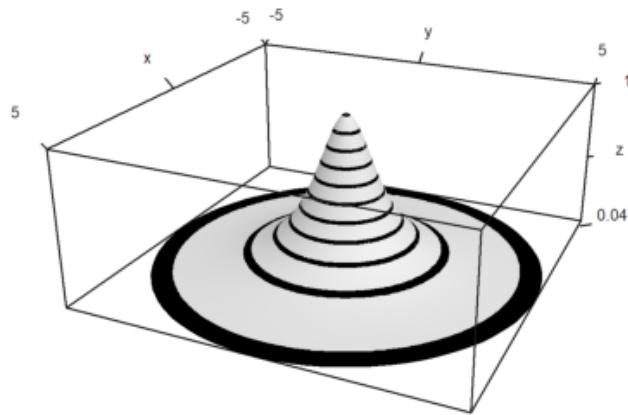


```
>plot3d("3*exp(-x^2-y^2)",r=1,<fscale,<scale,distance=15,height=70°, ...
> center=[0,0,2],frame=3):
```

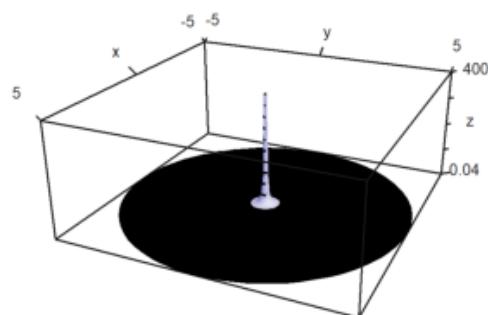


Sebuah plot kutub juga tersedia. Parameter polar=true menggambar plot polar. Fungsi tersebut harus tetap merupakan fungsi dari x dan y. Parameter "fscale" menskalakan fungsi dengan skala sendiri. Jika tidak, fungsi diskalakan agar sesuai dengan kubus.

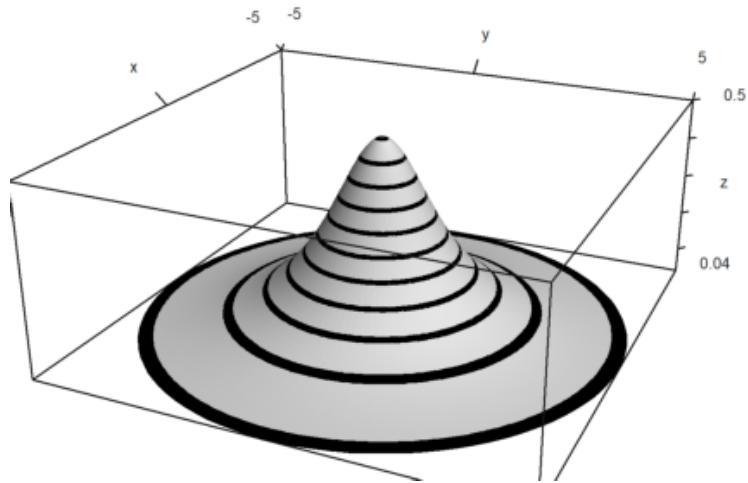
```
>plot3d("1/(x^2+y^2+1)",r=5,>polar, ...
>fscale=2,>hue,n=100,zoom=4,>contour,color=gray):
```



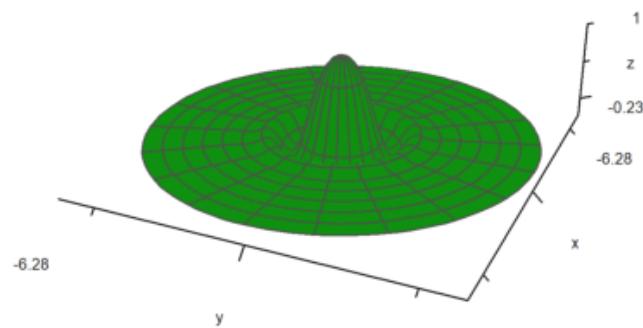
```
>plot3d("1/(x^2+y^2)",r=5,>polar, ...
>fscale=2,>hue,n=100,zoom=3,>contour,color=blue):
```



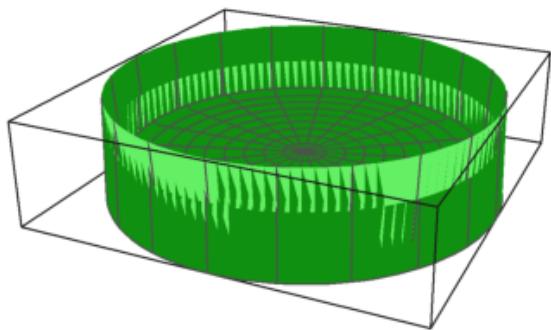
```
>plot3d("1/(x^2+y^2+2)",r=5,>polar, ...
>fscale=2,>hue,n=100,zoom=5,>contour,color=black):
```



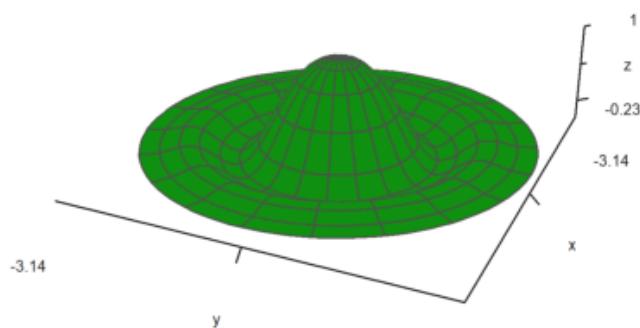
```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=2pi,frame=3,zoom=4):
```



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



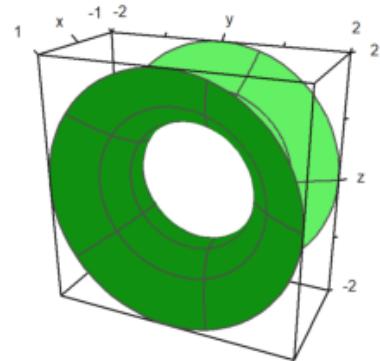
```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



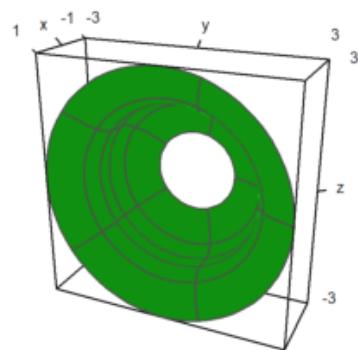
Rotasi parameter memutar fungsi dalam x di sekitar sumbu x.

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

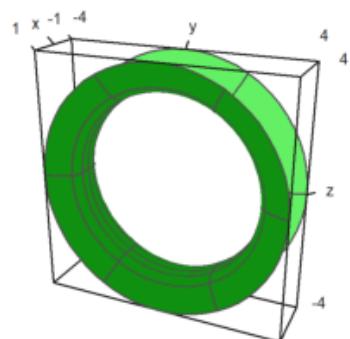
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



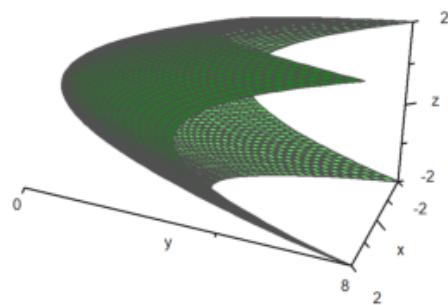
```
>plot3d("x^3+2", a=-1, b=1, rotate=true, grid=5):
```



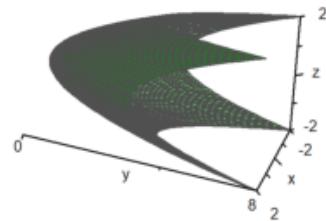
```
>plot3d("x^4+3", a=-1, b=1, rotate=true, grid=6):
```



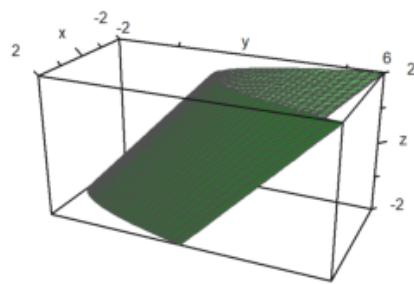
```
>plot3d("x", "x^2+y^2", "y", r=2, zoom=3.5, frame=3) :
```



```
>plot3d("x", "x^2+y^2", "y", r=2, zoom=2.5, frame=3) :
```



```
>plot3d("x", "x^2+y", "y", r=2, zoom=3, 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.

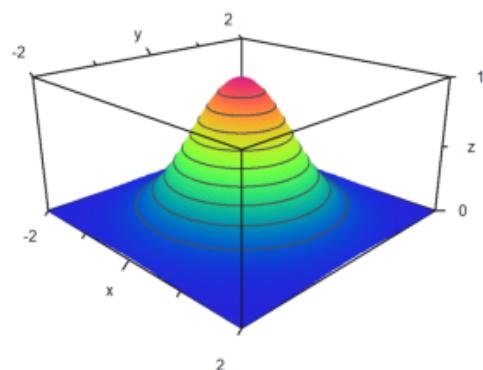
-> kontur: Memplot garis kontur otomatis pada plot.

- level=... (atau level): Sebuah vektor nilai untuk garis kontur.

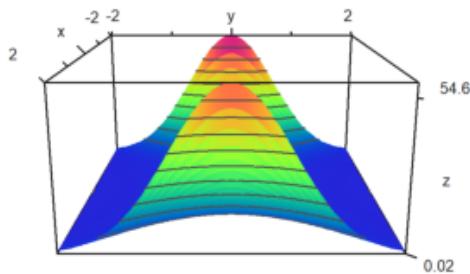
Standarnya adalah level="auto", yang menghitung beberapa garis level secara otomatis. Seperti yang Anda lihat di plot, level sebenarnya adalah rentang level.

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.

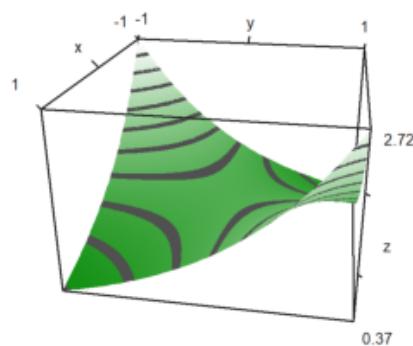
```
>plot3d("exp(-x^2-y^2)",r=2,n=100,level="thin", ...
> >contour,>spectral,fscale=1,scale=1.1,angle=45°,height=20°):
```



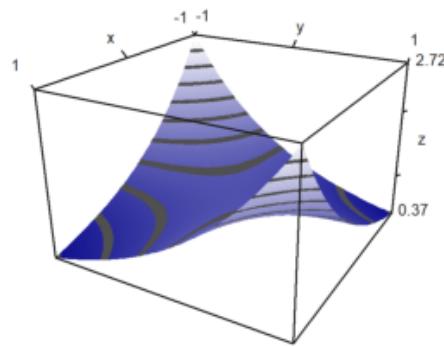
```
>plot3d("exp(x^2-y^2)",r=2,n=100,level="thin", ...
>>contour,>spectral,fscale=1,scale=1.1,angle=90°,height=15°):
```



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



```
>plot3d("exp(x*y)",angle=120°,>contour,color=blue):
```



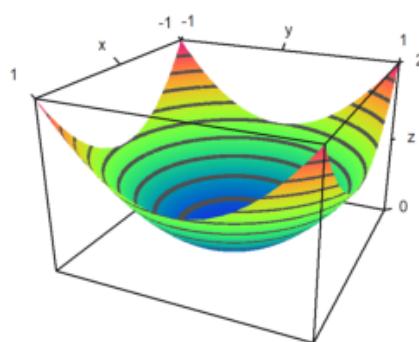
Bayangan default menggunakan warna abu-abu. Tetapi rentang warna spektral juga tersedia.

-> spektral: Menggunakan skema spektral default

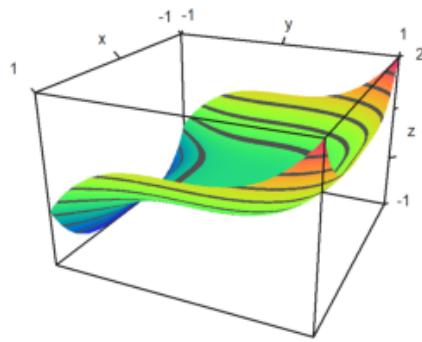
- color=...: Menggunakan warna khusus atau skema spektral

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

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

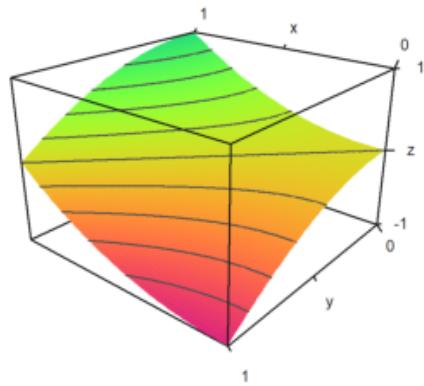


```
>plot3d("x^4+y^3", >spectral, >contour, n=100) :
```

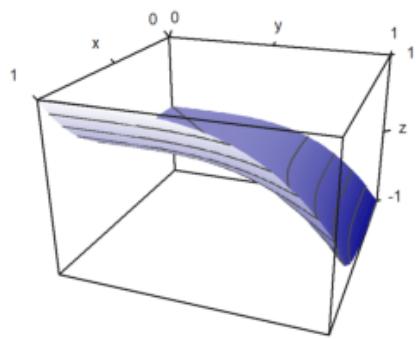


Alih-alih garis level otomatis, kita juga dapat mengatur nilai garis level. Ini akan menghasilkan garis level tipis alih-alih rentang level.

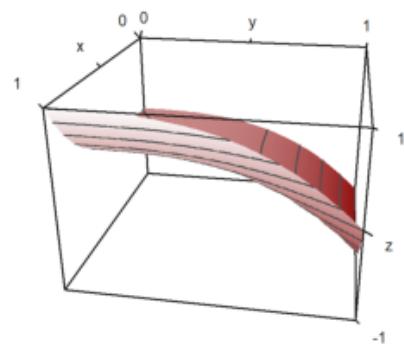
```
>plot3d("x^2-y^2",0,1,0,1,angle=220°,level=-1:0.2:1,color=redgreen):
```



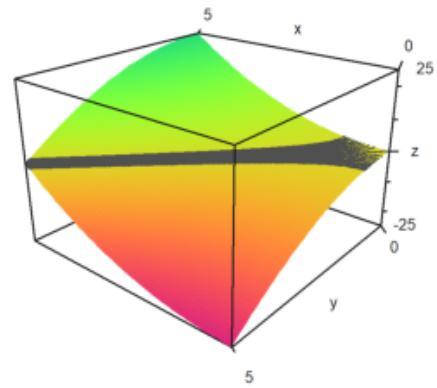
```
>plot3d("x^4-y^4",0,1,0,1,angle=110°,level=-1:0.2:1,color=blue):
```



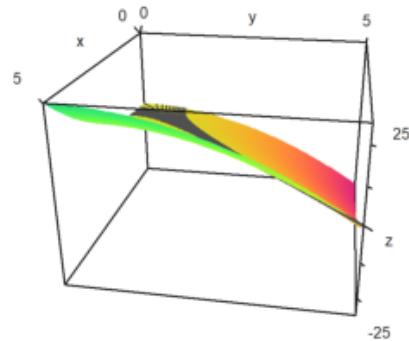
```
>plot3d("x^4-y^3",0,1,0,1,angle=100°,level=-1:0.2:1,color=red):
```



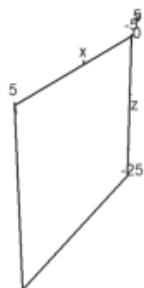
```
>plot3d("x^2-y^2",0,5,0,5,angle=220°,level=-1:0.1:1,color=redgreen):
```



```
>plot3d("x^2-y^2",0,5,0,5,angle=100°,level=-1:0.1:1,color=redgreen) :
```

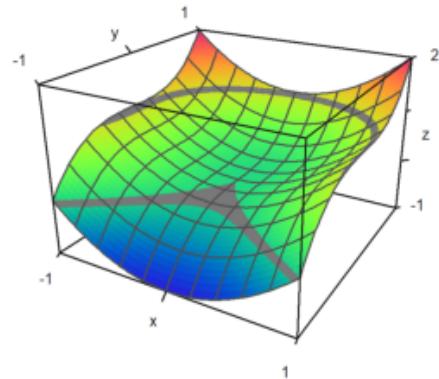


```
>plot3d("x^2-y^2",-5,5,5,5,level=-1:0.2:1,color=redgreen) :
```

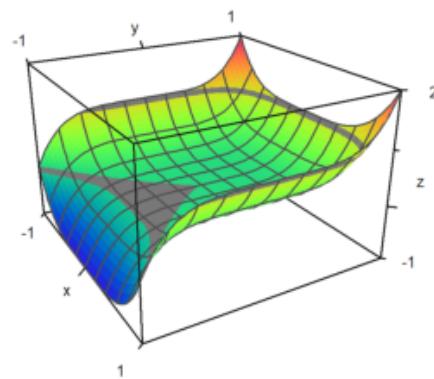


Dalam plot berikut, kami menggunakan dua pita level yang sangat luas dari -0,1 hingga 1, dan dari 0,9 hingga 1. Ini dimasukkan sebagai matriks dengan batas level sebagai kolom. Selain itu, kami melapisi kisi dengan 10 interval di setiap arah.

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



```
>plot3d("x^4+y^5",level=[-0.1,0.9;0,1], ...
> >spectral,angle=60°,grid=10,contourcolor=gray):
```

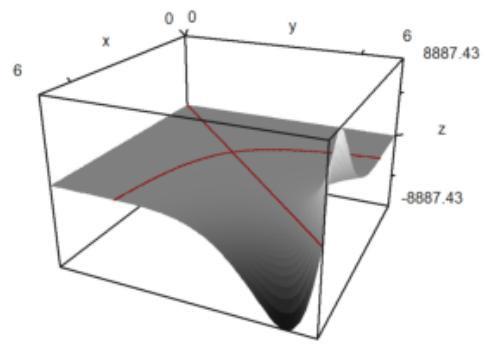


Dalam contoh berikut, kami memplot himpunan, di mana

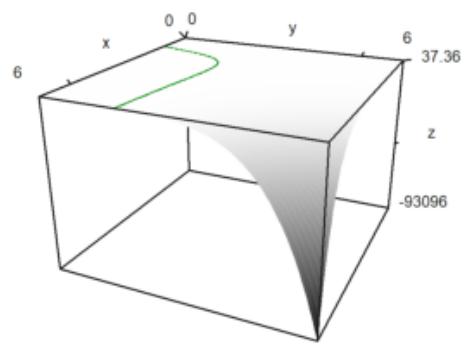
$$f(x, y) = x^y - y^x = 0$$

Kami menggunakan satu garis tipis untuk garis level.

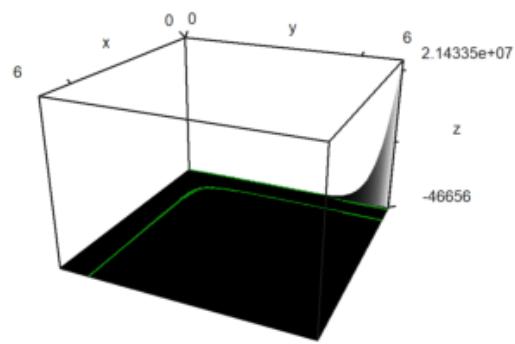
```
>plot3d("x^y-y^x",level=0,a=0,b=6,c=0,d=6,contourcolor=red,n=100):
```



```
>plot3d("x^2y-2y^2x",level=0,a=0,b=6,c=0,d=6,contourcolor=green,n=100):
```

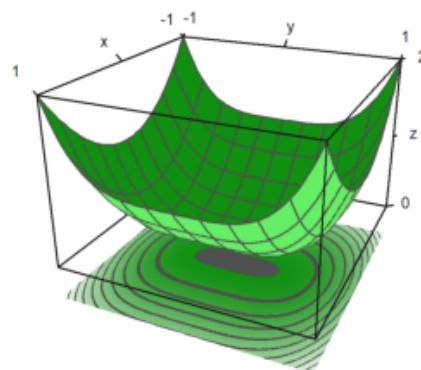


```
>plot3d("x^-y-y^2x",level=0,a=0,b=6,c=0,d=6,contourcolor=green,n=100):
```

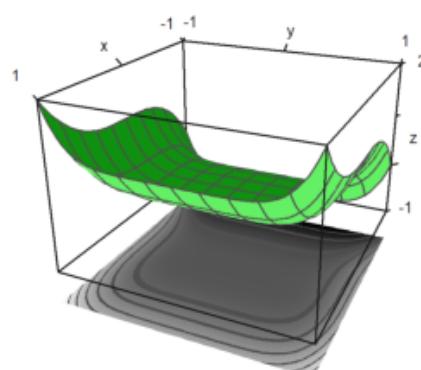


Dimungkinkan untuk menunjukkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

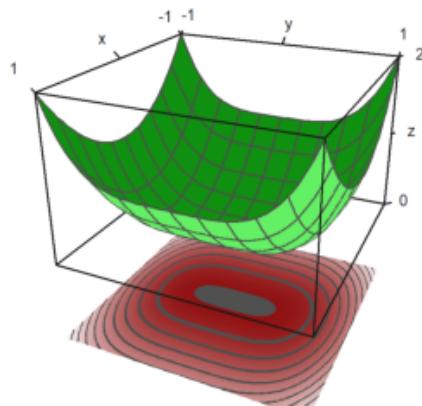
```
>plot3d("x^2+y^4", >cp, cpcolor=green, cpdelta=0.2) :
```



```
>plot3d("x^3+y^6", >cp, cpcolor=black, cpdelta=0.2) :
```



```
>plot3d("x^2+y^4", >cp, cpcolor=red, cpdelta=0.5) :
```



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

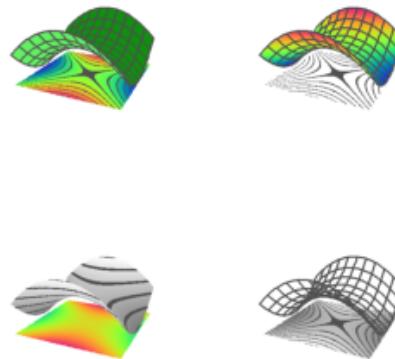
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1); ...
> plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2); ...
> plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3); ...
> plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4); ...
> plot3d(expr,<frame,>hue,grid=10,>transparent,>cp,cpcolor=gray);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



```

>figure(2,2); ...
>expr="x^2 - y^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):

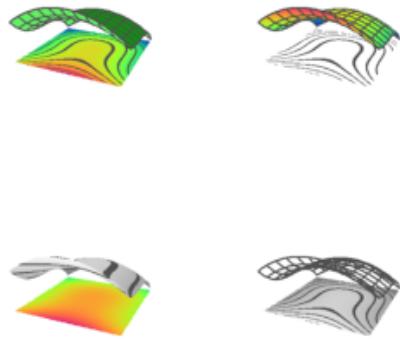
```



```

>figure(2,2); ...
>expr="x^3 - y^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):

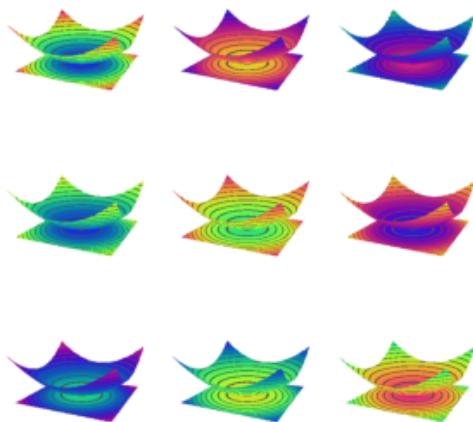
```



Ada beberapa skema spektral lainnya, bermnomor 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
- birukuning, 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):
```



```

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

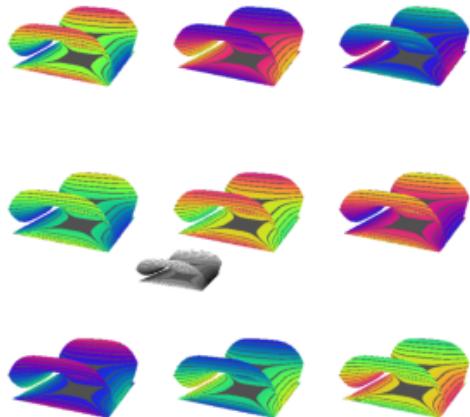
```



```

>figure(3,3); ...
>for i=1:10; ...
> figure(i); plot3d("x^4-y^4",spectral=i,>contour,>cp,<frame,zoom=4); ...
>end; ...
>figure(0):

```

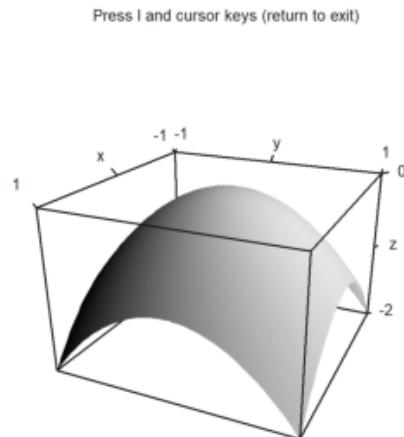


Sumber cahaya dapat diubah dengan 1 dan tombol cursor selama interaksi pengguna. Itu juga dapat diatur dengan parameter.

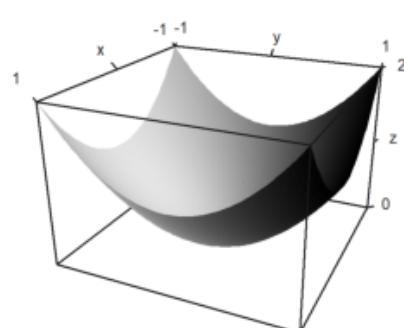
- cahaya: arah untuk cahaya
- amb: cahaya sekitar antara 0 dan 1

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)":
```



```
>plot3d("x^2+y^2", ...
>  hue=true, light=[0,1,1], amb=0, user=true, ...
>  title="Press l and cursor keys (return to exit)":
```

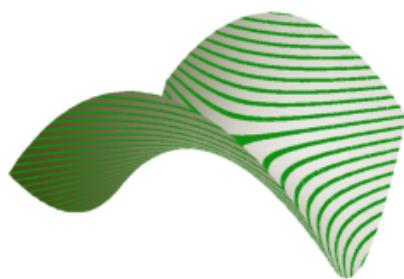


Parameter warna mengubah warna permukaan. Warna garis level juga dapat diubah.

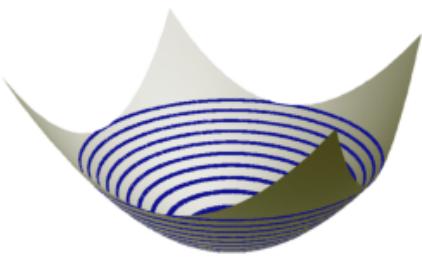
```
>plot3d("-x^2-y^2",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>  zoom=3,contourcolor=red,level=-2:0.1:1,dl=0.01):
```



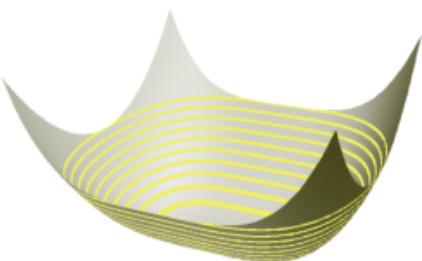
```
>plot3d("x^2-y^2",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>  zoom=3,contourcolor=green,level=-2:0.1:1,dl=0.01):
```



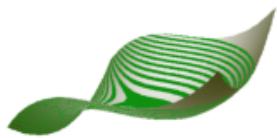
```
>plot3d("x^2+y^2",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>  zoom=3,contourcolor=blue,level=-2:0.1:1,dl=0.01):
```



```
>plot3d("x^2+y^4",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>   zoom=3,contourcolor=yellow,level=-2:0.1:1,d1=0.01):
```

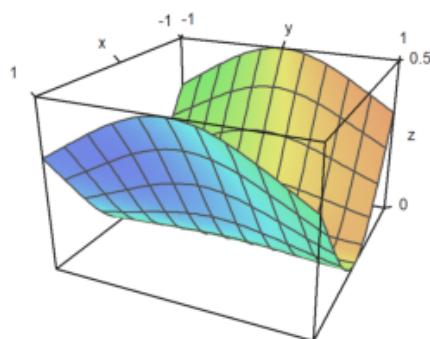


```
>plot3d("x^2+y^3",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>   zoom=2,contourcolor=green,level=-2:0.1:1,d1=0.01):
```

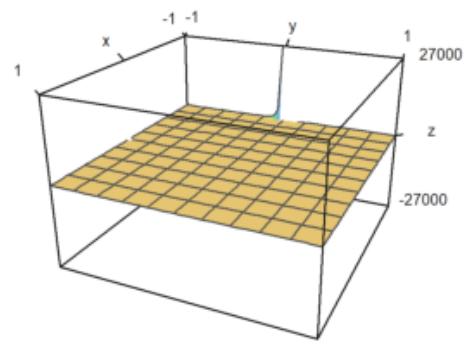


Warna 0 memberikan efek pelangi khusus.

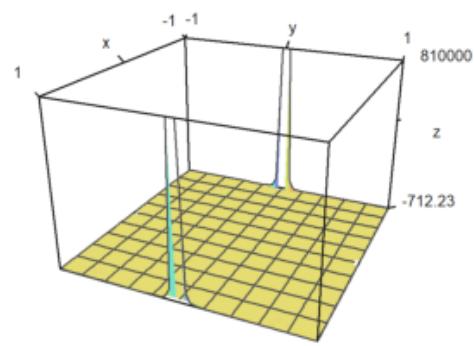
```
>plot3d("x^2/(x^2+y^2+1)",color=0,hue=true,grid=10):
```



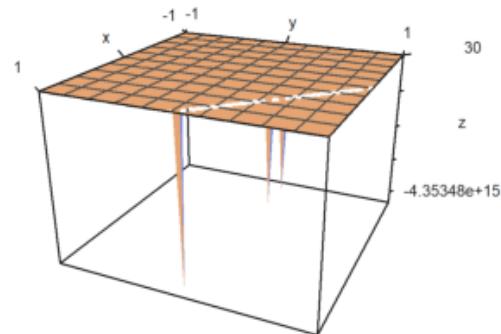
```
>plot3d("x^3/(x^3+y^3+1)",color=0,hue=true,grid=10):
```



```
>plot3d("x^4/(x^4+y^4-1)",color=0,hue=true,grid=10):
```

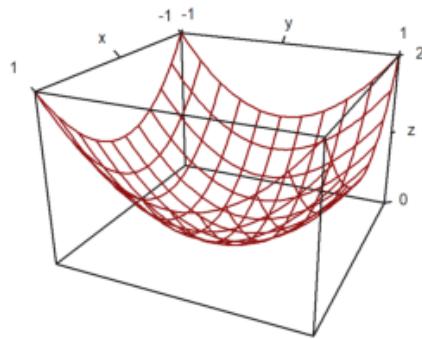


```
>plot3d("x/(x+y-1)",color=0,hue=true,grid=10):
```

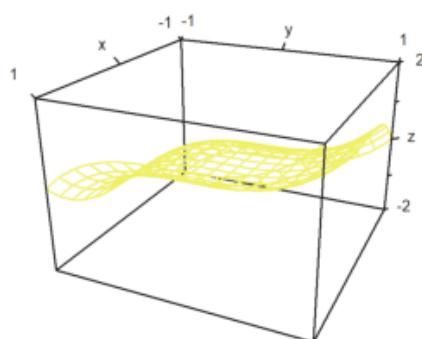


Permukaannya juga bisa transparan.

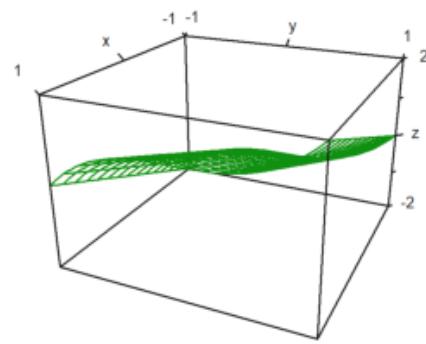
```
>plot3d("x^2+y^2",>transparent,grid=10,wirecolor=red):
```



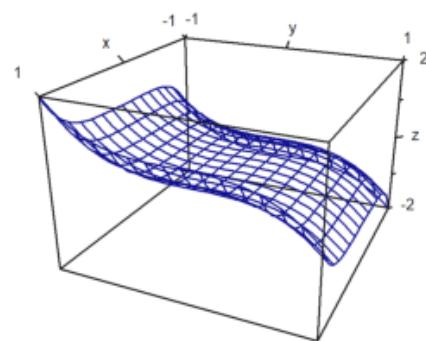
```
>plot3d("x^3+y^3",>transparent,grid=15,wirecolor=yellow):
```



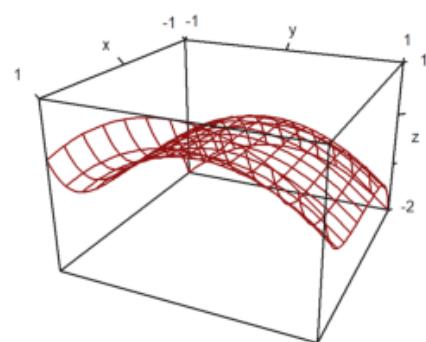
```
>plot3d("x+y^3",>transparent,grid=15,wirecolor=green):
```



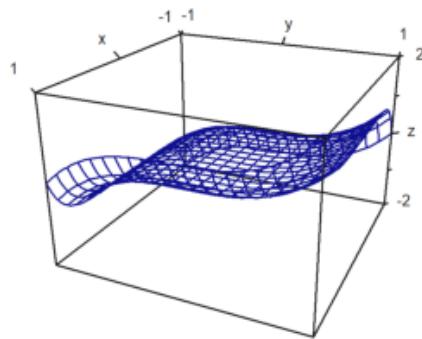
```
>plot3d("x^5-y^3",>transparent,grid=15,wirecolor=blue):
```



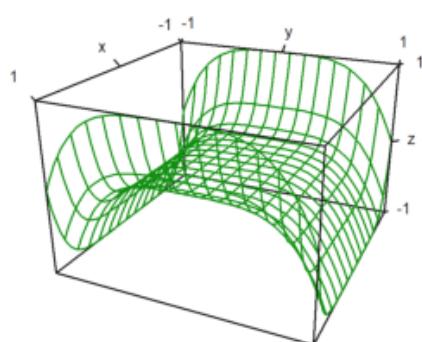
```
>plot3d("x^5-y^2",>transparent,grid=13,wirecolor=red):
```



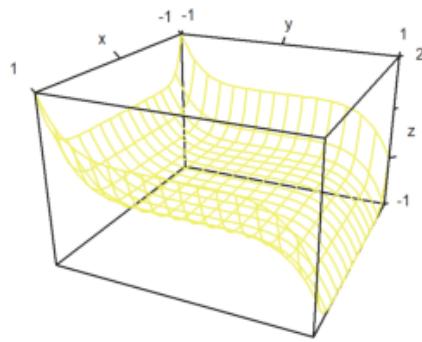
```
>plot3d("x^7+y^3",>transparent,grid=20,wirecolor=blue):
```



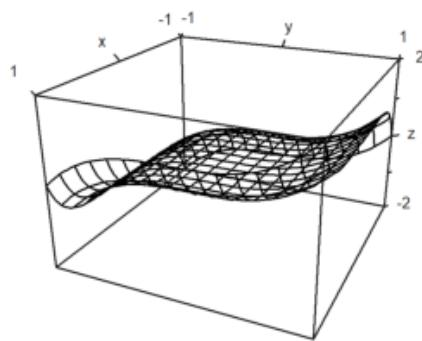
```
>plot3d("x^8-y^4",>transparent,grid=15,wirecolor=green):
```



```
>plot3d("x^10-y^5",>transparent,grid=15,wirecolor=yellow):
```



```
>plot3d("x^7+y^3", >transparent, grid=15, wirecolor=black) :
```



Plot Implisit

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

Solusi dari

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

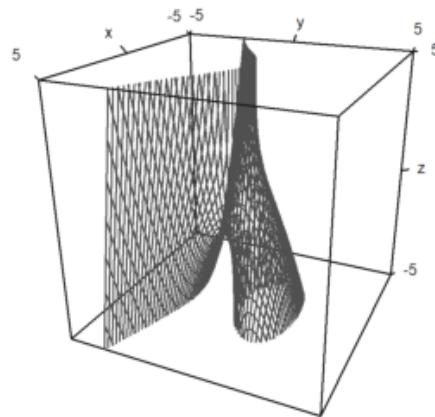
dapat divisualisasikan dalam potongan sejajar dengan bidang x-y-, x-z- dan y-z.

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

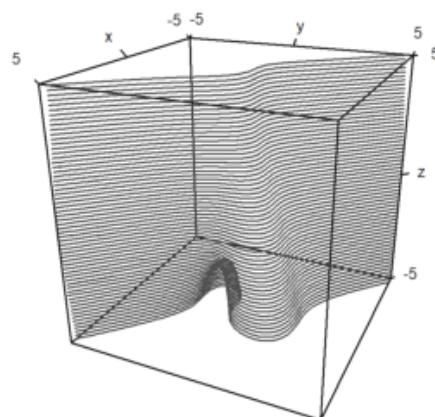
Tambahkan nilai-nilai ini, jika Anda suka. Dalam contoh kita plot

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

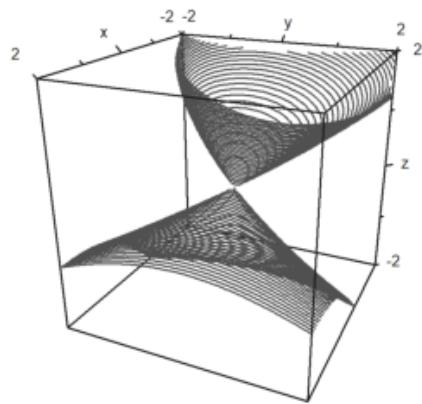
```
>plot3d("x^2+y^3+z*y-1", r=5, implicit=3):
```



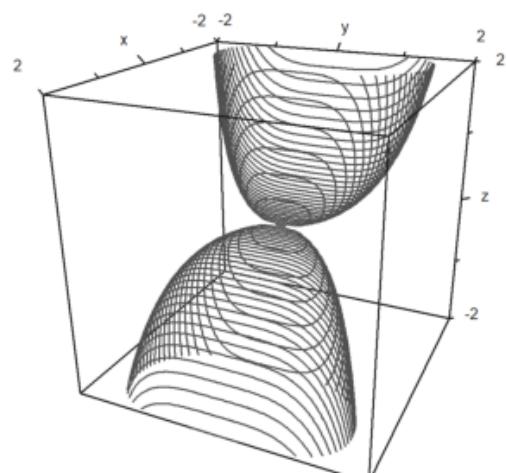
```
>plot3d("x^3+y^3+z*y-2", r=5, implicit=4):
```



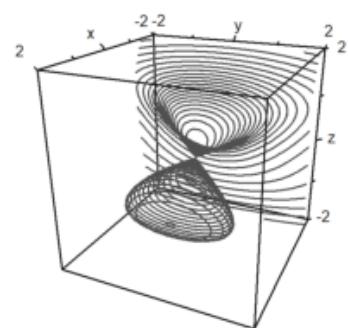
```
>plot3d("x^2+y^2+4*x*z+z^3", >implicit, r=2, zoom=2.5):
```



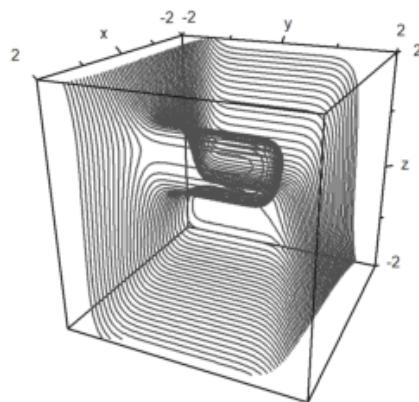
```
>plot3d("x^4+y^4+5*x*z+z^2",>implicit,r=2,zoom=3):
```



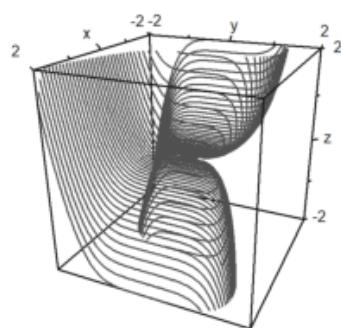
```
>plot3d("x^7+y^2+3*x*z+z^4",>implicit,r=2,zoom=2):
```



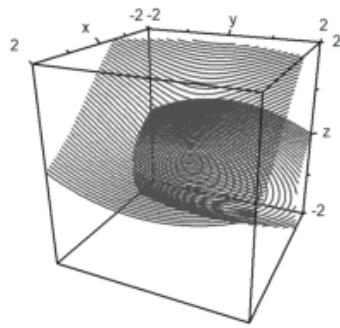
```
>plot3d("x^5+y^9-3*x*z+z^5",>implicit,r=2,zoom=2.5):
```



```
>plot3d("x^4+y^5+5*x*z+z^2",>implicit,r=2,zoom=2):
```



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



Merencanakan 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 $fx(x,y)$, $fy(x,y)$, $fz(x,y)$.

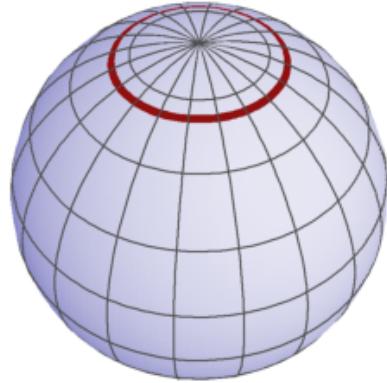
$$\gamma(t, s) = (x(t, s), y(t, s), z(t, s))$$

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

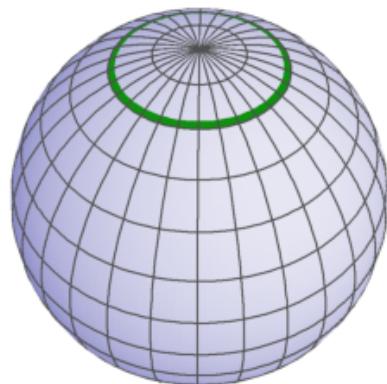
Anda dapat menggunakan bahasa matriks Euler untuk menghasilkan koordinat secara efektif.

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

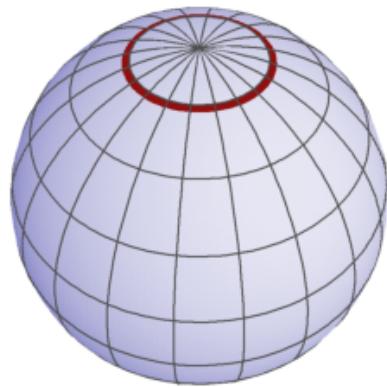
```
>t=linspace(0,2pi,180); s=linspace(-pi/2,pi/2,90)'; ...
>x=cos(s)*cos(t); y=cos(s)*sin(t); z=sin(s); ...
>plot3d(x,y,z,>hue, ...
>color=blue,<frame,grid=[10,20], ...
>values=s,contourcolor=red,level=[90°-24°;90°-22°], ...
>scale=1.4,height=50°):
```



```
>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=[15,30], ...
>values=s,contourcolor=green,level=[90°-24°;90°-22°], ...
>scale=1.4,height=50°):
```

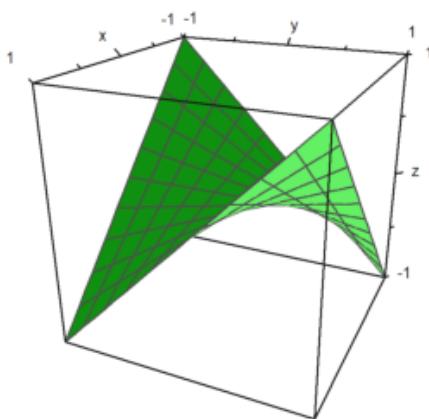


```
>t=linspace(0,2pi,180); s=linspace(-pi/2,pi/2,90)'; ...
>x=cos(s)*cos(t); y=cos(s)*sin(t); z=sin(s); ...
>plot3d(x,y,z,>hue, ...
>color=blue,<frame,grid=[10,20], ...
>values=s,contourcolor=red,level=[90°-20°;90°-18°], ...
>scale=1.4,height=50°):
```

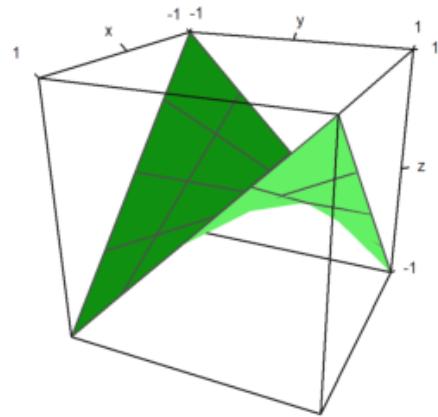


Berikut adalah contoh, yang merupakan grafik fungsi.

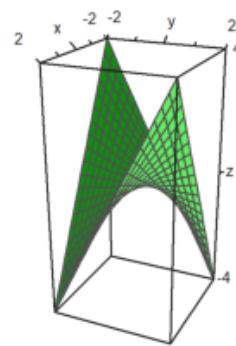
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



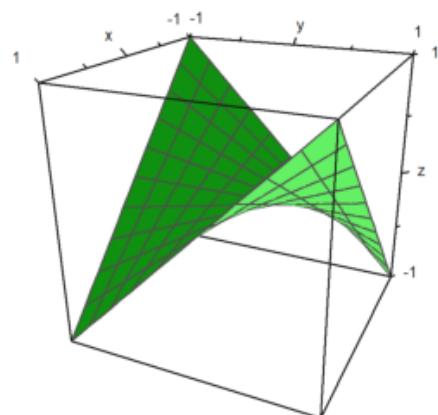
```
>t=-1:0.5:1; s=(-1:0.5:1)'; plot3d(t,s,t*s,grid=10):
```



```
>t=-2:0.2:2; s=(-2:0.2:2)'; plot3d(t,s,t*s,grid=15):
```



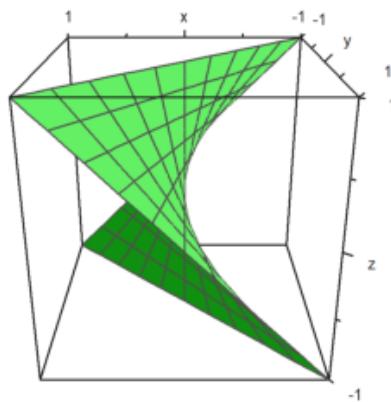
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=11):
```



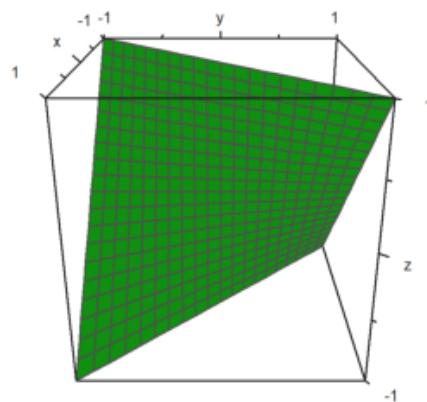
Namun, kita bisa membuat segala macam permukaan. Berikut adalah permukaan yang sama dengan fungsi

$$x = yz$$

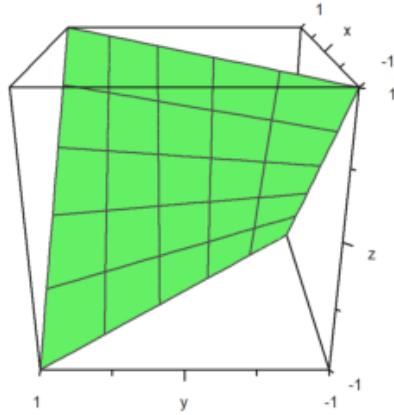
```
>plot3d(t*s,t,s,angle=180°,grid=10) :
```



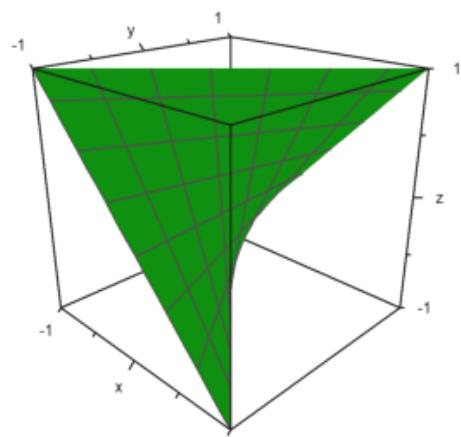
```
>plot3d(t*s,t,s,angle=90°,grid=16) :
```



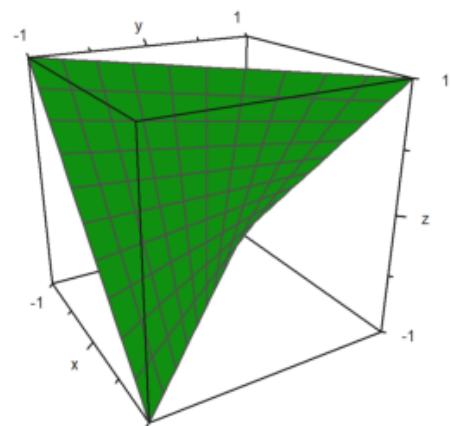
```
>plot3d(t*s,t,s,angle=270°,grid=5) :
```



```
>plot3d(t*s,t,s,angle=45°,grid=7):
```



```
>plot3d(t*s,t,s,angle=60°,grid=10):
```



Dengan lebih banyak usaha, kami dapat menghasilkan banyak permukaan.

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

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

dengan

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

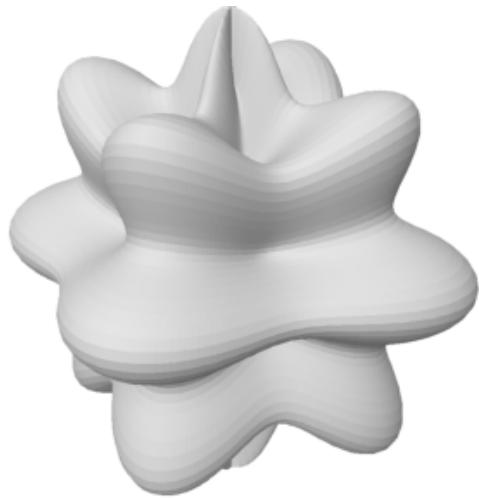
Kami mendistorsi ini dengan sebuah faktor

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

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

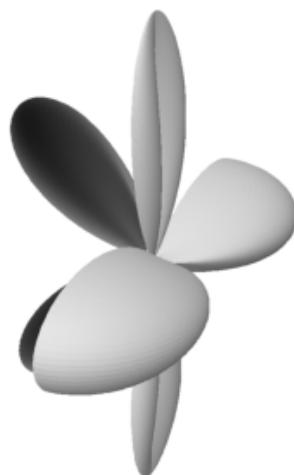


```
>t=linspace(0,2pi,270); s=linspace(-pi/2,pi/2,80)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```

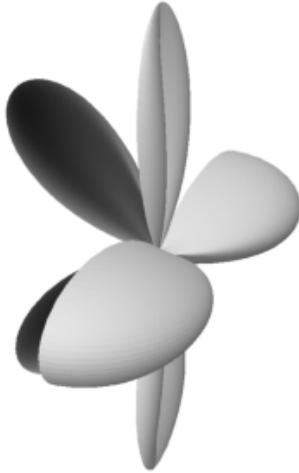


Dalam percobaan saya ini ternyata dengan fungsi tersebut dapat membentuk sebuah bunga. berikut beberapa bentuk contohnya

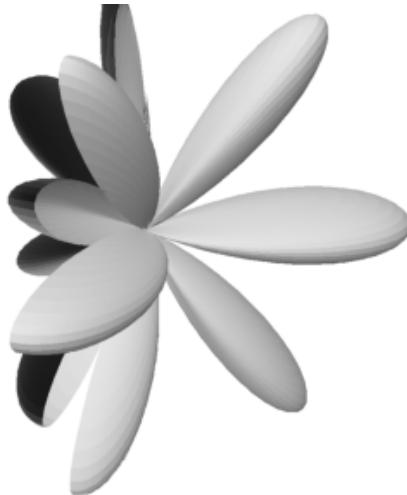
```
>t=linspace(0,pi,320); s=linspace(-pi/2,pi/2,150)'; ...
>d=1+5*(cos(2*t)+cos(4*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



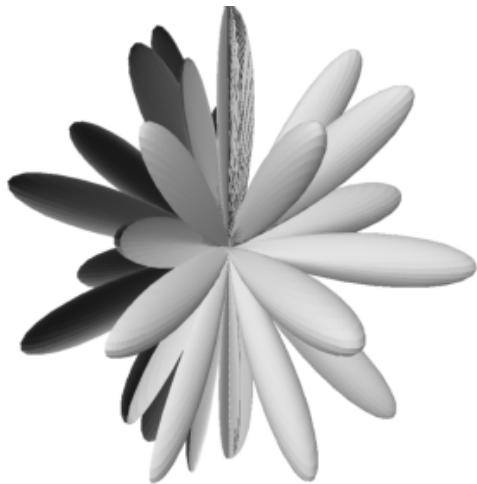
```
>t=linspace(0,pi,320); s=linspace(-pi/2,pi/2,150)'; ...
>d=1+7*(cos(2*t)+cos(4*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



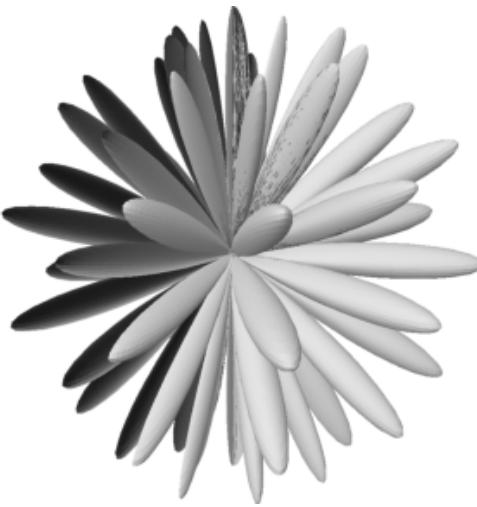
```
>t=linspace(0,pi,260); s=linspace(-pi/2,pi/2,120)'; ...
>d=1+10*(cos(3*t)+cos(7*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



```
>t=linspace(0,pi,260); s=linspace(-pi/2,pi/2,120)'; ...
>d=1+15*(cos(7*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):
```



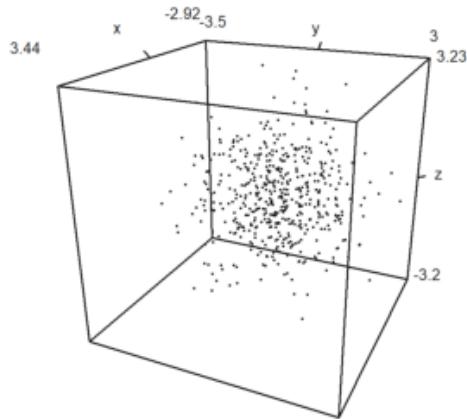
```
>t=linspace(0,pi,270); s=linspace(-pi/2,pi/2,180)'; ...
>d=1+15*(cos(10*t)+cos(12*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



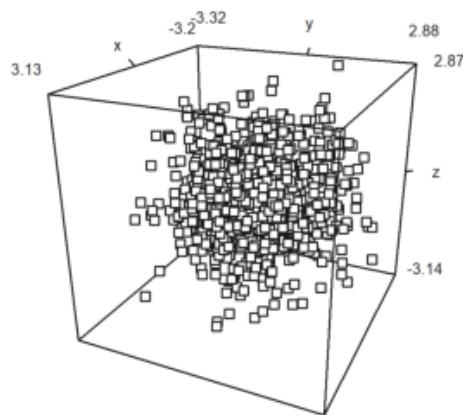
Tentu saja, titik cloud juga dimungkinkan. Untuk memplot data titik dalam ruang, kita membutuhkan tiga vektor untuk koordinat titik-titik tersebut.

Gayanya sama seperti di plot2d dengan points=true;

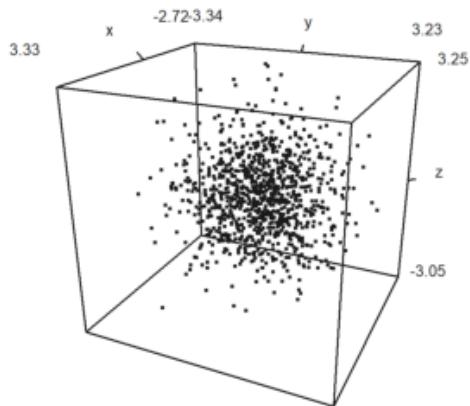
```
>n=500; ...
> plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style="."):
```



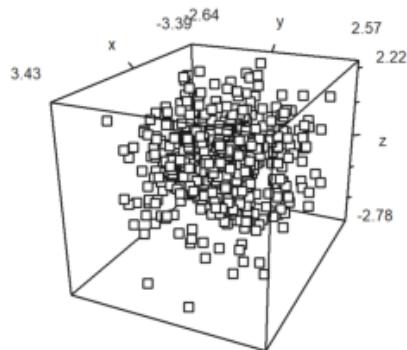
```
>n=1000; ...  
> plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style="--"):
```



```
>n=1000; ...  
> plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style=".."):
```

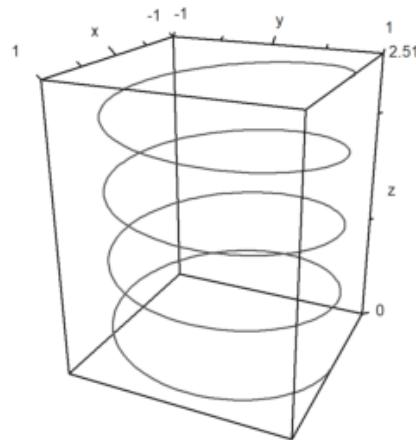


```
>n=500; ...
> plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style="**") :
```

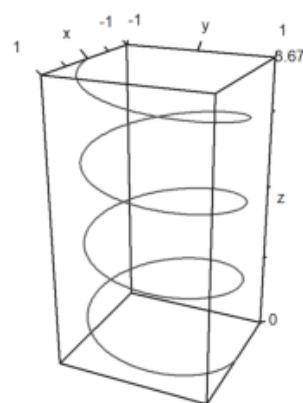


Dimungkinkan juga untuk memplot kurva dalam 3D. Dalam hal ini, lebih mudah untuk menghitung titik-titik kurva. Untuk kurva di pesawat kami menggunakan urutan koordinat dan parameter wire=true.

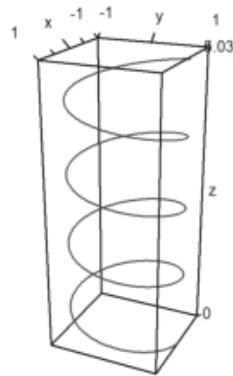
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3) :
```



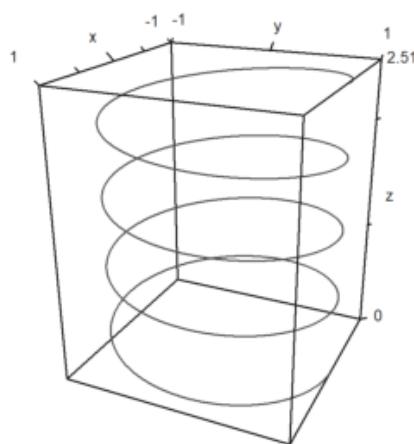
```
>t=linspace(0,7pi,1000); ...
>plot3d(sin(t),cos(t),t/6,>wire,zoom=3):
```



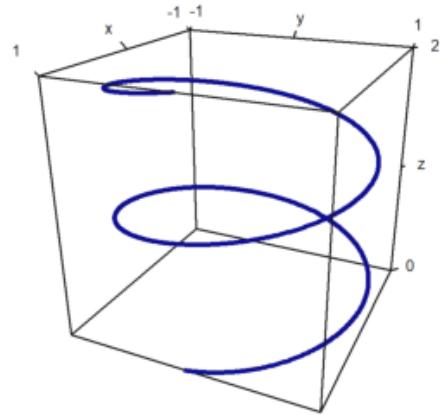
```
>t=linspace(0,8*pi,1000); ...
>plot3d(sin(t),cos(t),t/5,>wire,zoom=3):
```



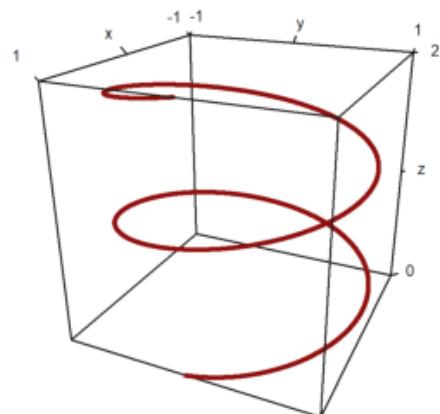
```
>t=linspace(0,8*pi,3000); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```



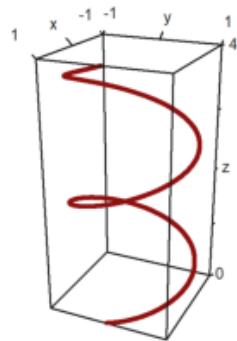
```
>t=linspace(0,4pi,1000); plot3d(cos(t),sin(t),t/2pi,>wire, ...
>lineWidth=3,wirecolor=blue):
```



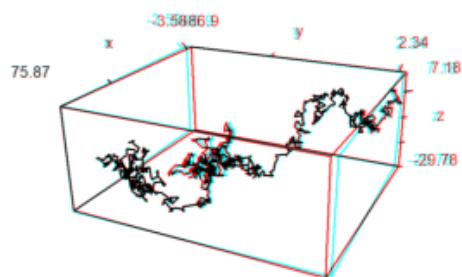
```
>t=linspace(0,4pi,4000); plot3d(cos(t),sin(t),t/2pi,>wire, ...
>lineWidth=3,wirecolor=red):
```



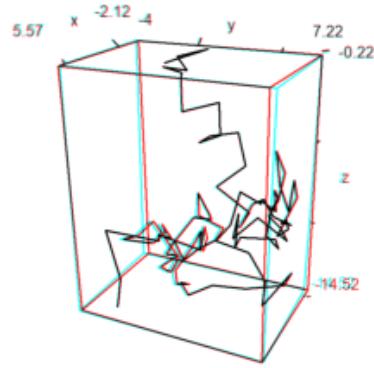
```
>t=linspace(0,4pi,4000); plot3d(cos(t),sin(t),t/pi,>wire, ...
>lineWidth=3,wirecolor=red):
```



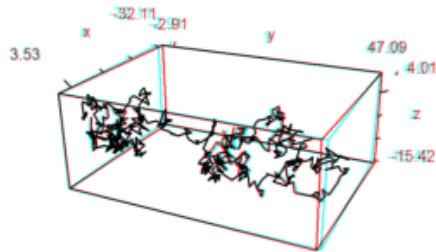
```
>X=cumsum(normal(3,1000)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



```
>X=cumsum(normal(5,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```

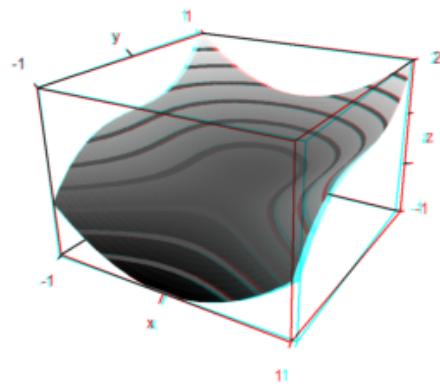


```
>X=cumsum(normal(6,600)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire) :
```

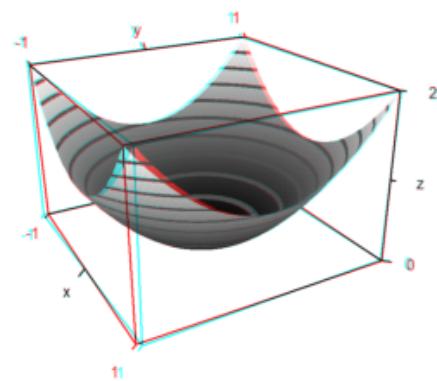


EMT juga dapat memplot dalam mode anaglyph. Untuk melihat plot seperti itu, Anda memerlukan kacamata merah/sian.

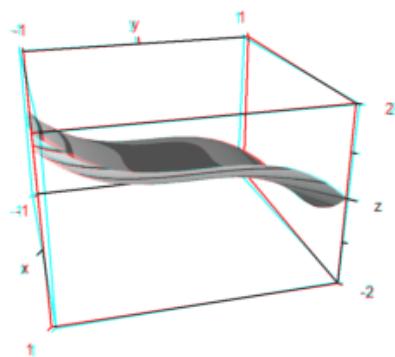
```
> plot3d("x^2+y^3",>anaglyph,>contour,angle=30°) :
```



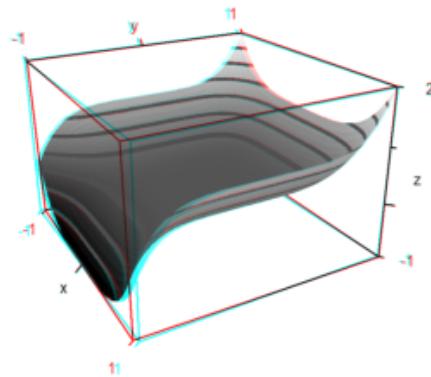
```
>plot3d("x^2+y^2",>anaglyph,>contour,angle=60°):
```



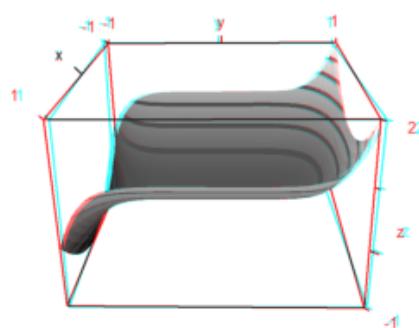
```
> plot3d("x^3-y^3",>anaglyph,>contour,angle=75°):
```



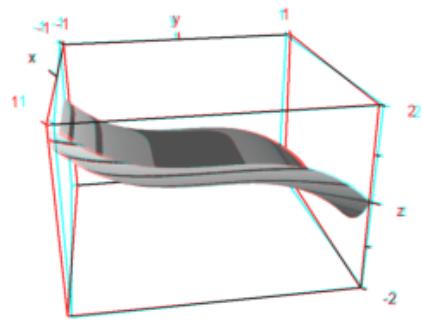
```
> plot3d("x^4+y^5",>anaglyph,>contour,angle=60°) :
```



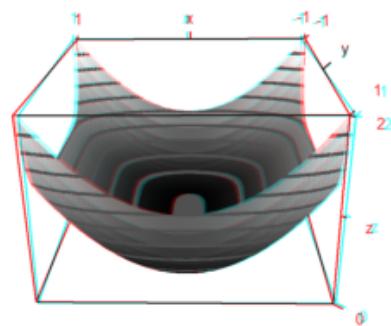
```
> plot3d("x^2+y^7",>anaglyph,>contour,angle=90°) :
```



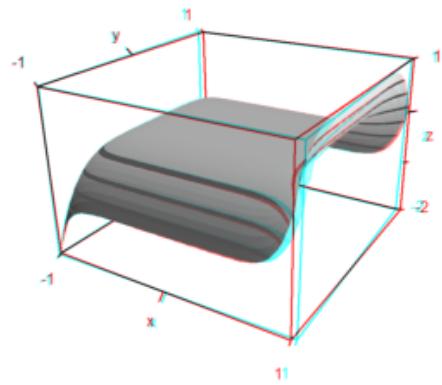
```
> plot3d("x^5-y^3",>anaglyph,>contour,angle=80°) :
```



```
> plot3d("x^2+y^8",>anaglyph,>contour,angle=180°):
```

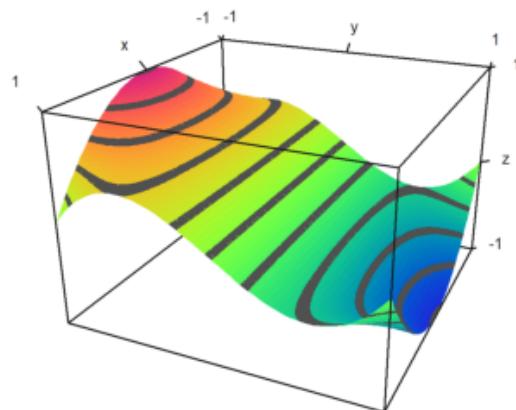


```
> plot3d("x^9-y^6",>anaglyph,>contour,angle=30°):
```

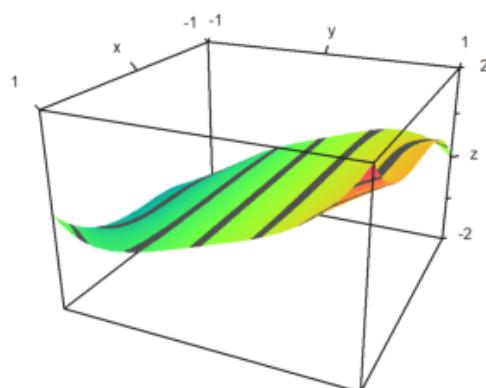


Seringkali, skema warna spektral digunakan untuk plot. Ini menekankan ketinggian fungsi.

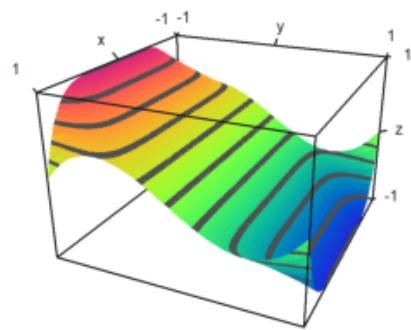
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2) :
```



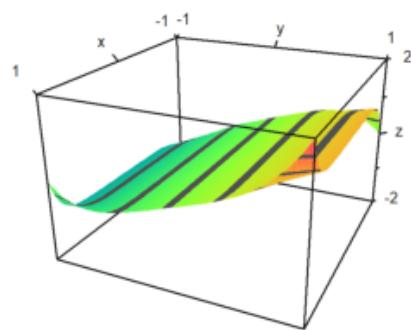
```
>plot3d("x^3*y^4+y",>spectral,>contour,zoom=3) :
```



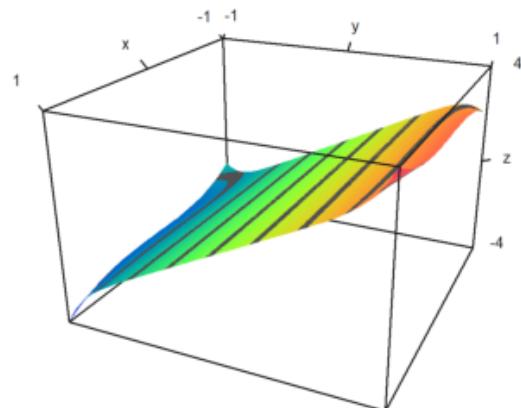
```
>plot3d("x^6*y^3-y",>spectral,>contour,zoom=2.5) :
```



```
>plot3d("x^9*y^4+y",>spectral,>contour,zoom=2.5):
```



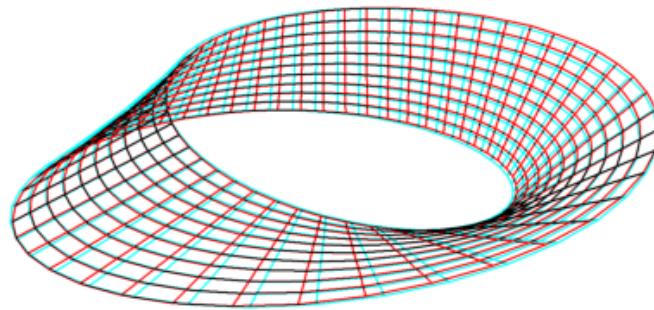
```
>plot3d("x^3*y^9+3y",>spectral,>contour,zoom=3.2):
```



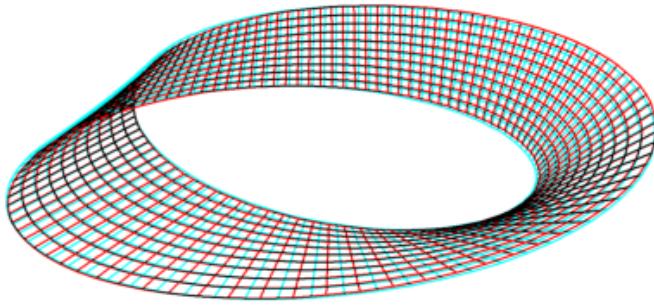
Euler juga dapat memplot permukaan berparameter, ketika parameternya adalah nilai x-, y-, dan z dari gambar kotak persegi panjang dalam ruang.

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

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



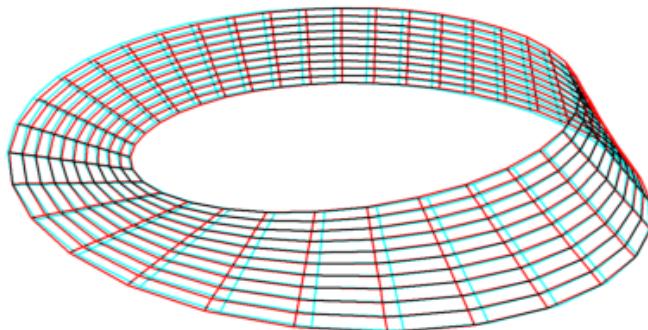
```
>u=linspace(-1,1,10); v=linspace(0,2*pi,100)'; ...
>X=(4+u*cos(v/2))*cos(v); Y=(4+u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3):
```



```

>u=linspace(-1,1,10); v=linspace(0,4*pi,70)'; ...
>X=(5-u*cos(v/2))*cos(v); Y=(5-u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3):

```

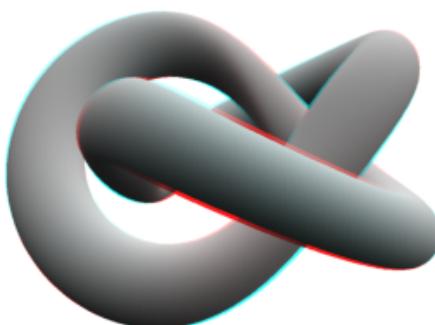


Berikut adalah contoh yang lebih rumit, yang megah dengan kacamata merah/sian.

```

>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):

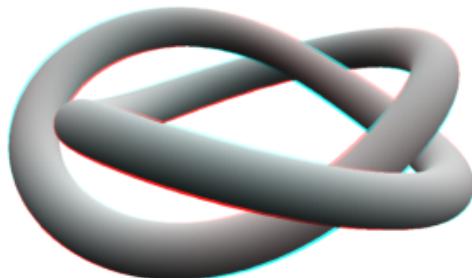
```



```

>u:=linspace(-pi,pi,160); v:=linspace(-pi,pi,500)'; ...
>x:=(4*(2+.25*sin(3*v))+cos(u))*cos(2*v); ...
>y:=(4*(2+.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=3,>anaglyph):

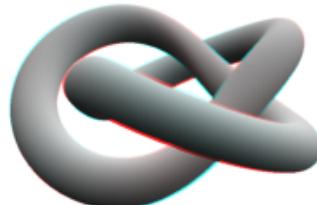
```



```

>u:=linspace(-pi,pi,170); v:=linspace(-pi,pi,700)'; ...
>x:=(5*(1+.25*sin(3*v))+cos(u))*cos(2*v); ...
>y:=(5*(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,>anaglyph):

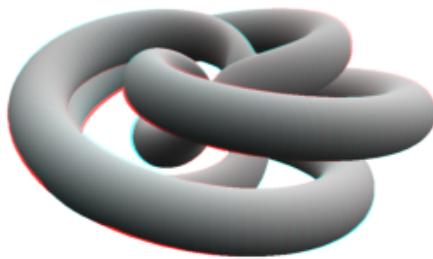
```



```

>u:=linspace(-pi,pi,160); v:=linspace(-pi,pi,400)'; ...
>x:=(4*(1+.75*sin(3*v))+cos(u))*cos(4*v); ...
>y:=(4*(1+.75*sin(3*v))+cos(u))*sin(4*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 bar juga dimungkinkan. Untuk ini, kita harus menyediakan

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

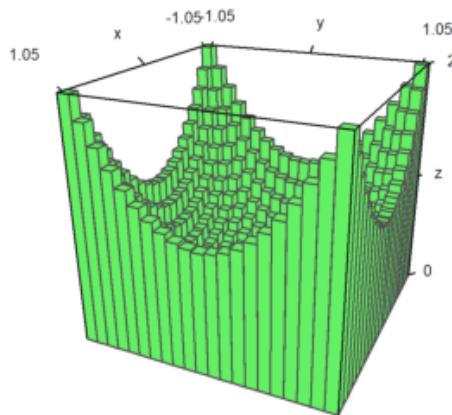
z bisa lebih besar, tetapi hanya nilai nxn yang akan digunakan.

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

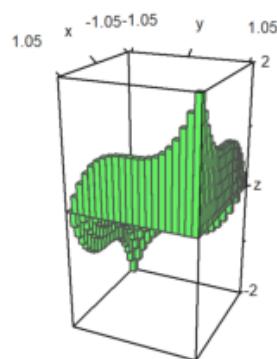
```

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

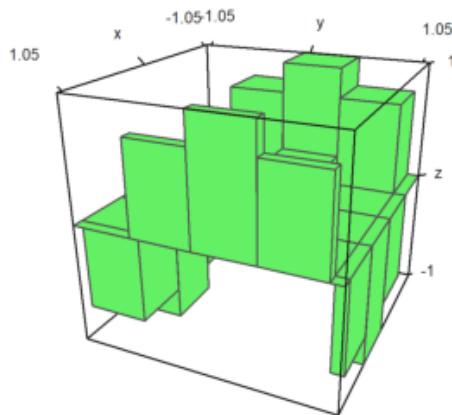
```



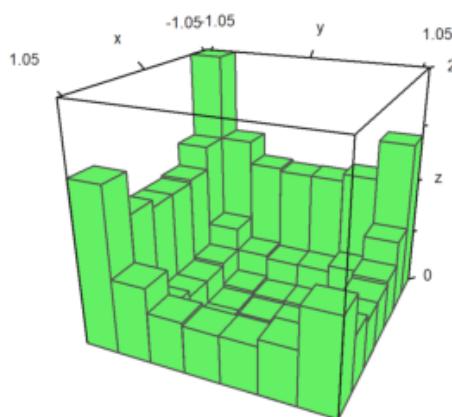
```
>x=-1:0.1:1; y=x'; z=x^3+y^3; ...
>xa=(x|1.1)-0.05; ya=(y_1.1)-0.05; ...
>plot3d(xa,ya,z,bar=true):
```



```
>x=-1:0.5:1; 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):
```

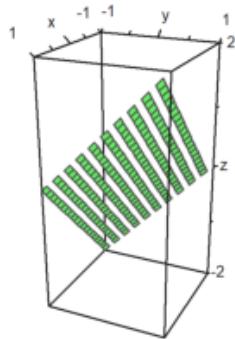


```
>x=-1:0.3:1; y=x'; z=x^4+y^4; ...
>xa=(x\1.1)-0.05; ya=(y\1.1)-0.05; ...
>plot3d(xa,ya,z,bar=true):
```

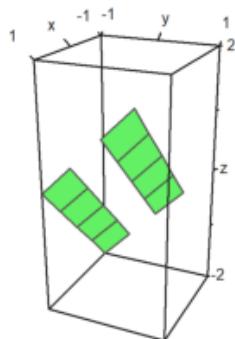


Dimungkinkan untuk membagi plot permukaan menjadi dua atau lebih bagian.

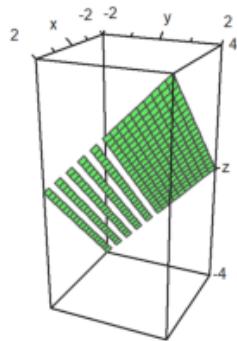
```
>x=-1:0.1:1; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:20):
```



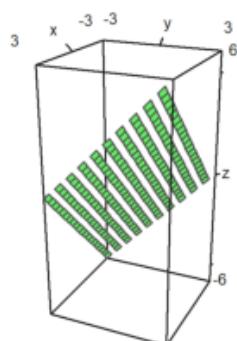
```
>x=-1:0.5:1; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:20);
```



```
>x=-2:0.2:2; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:10);
```

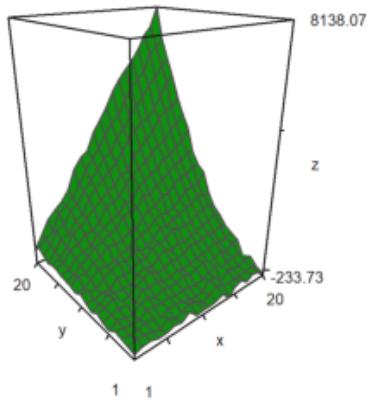


```
>x=-3:0.3:3; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:30);
```

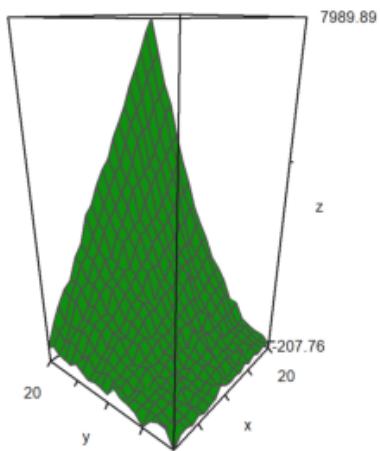


Jika memuat atau menghasilkan matriks data M dari file dan perlu memplotnya dalam 3D, Anda dapat menskalakan matriks ke [-1,1] dengan scale(M), atau menskalakan matriks dengan >zscale. Ini dapat dikombinasikan dengan faktor penskalaan individu yang diterapkan sebagai tambahan.

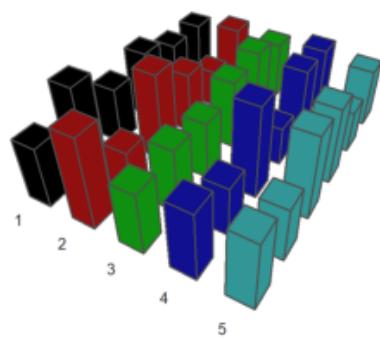
```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8)
```



```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,2],angle=-50°,zoom=1.8):
```

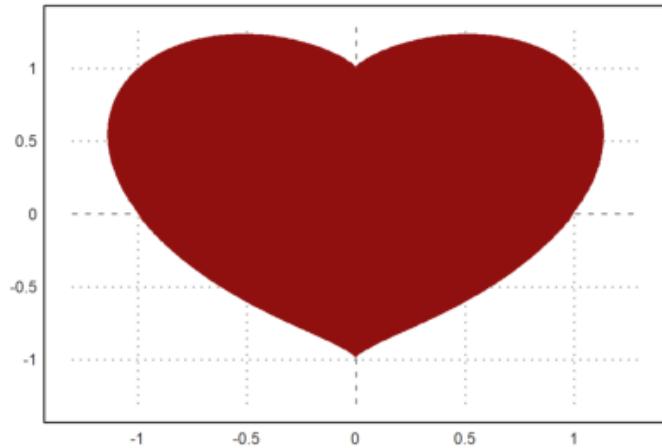


```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



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):
```



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

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

Kami ingin memutar kurva jantung di sekitar sumbu y. Berikut adalah ungkapan, yang mendefinisikan hati:

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

Selanjutnya kita atur

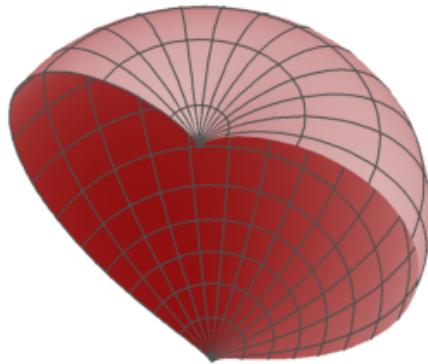
$$x = r.\cos(a), \quad y = r.\sin(a).$$

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

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

Hal ini memungkinkan untuk mendefinisikan fungsi numerik, yang memecahkan r , jika a diberikan. Dengan fungsi itu kita dapat memplot 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°):
```

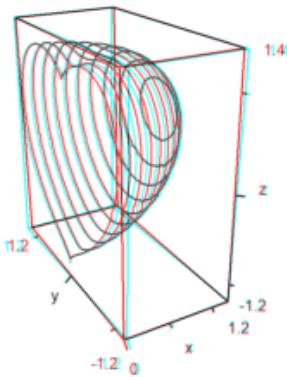


Berikut ini adalah plot 3D dari gambar di atas yang diputar di sekitar sumbu z. Kami mendefinisikan fungsi, yang menggambarkan objek.

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

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

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



Plot 3D Khusus

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

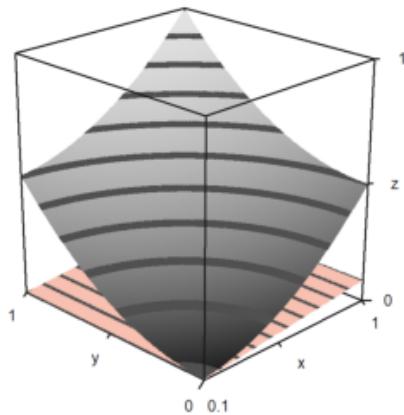
Meskipun Euler bukan program 3D, ia dapat menggabungkan beberapa objek dasar. Kami mencoba memvisualisasikan paraboloid dan garis singgungnya.

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

```
y=0:0.01:1; x=(0.1:0.01:1)';
plot3d(x,y,0.2*(x-0.1)/2,<scale,<frame,>hue, ..
    hues=0.5,>contour,color=orange);
h=holding(1);
plot3d(x,y,(x^2+y^2)/2,<scale,<frame,>contour,>hue);
holding(h);
endfunction
```

Sekarang framedplot() menyediakan frame, dan mengatur tampilan.

```
>framedplot("myplot",[0.1,1,0,1,0,1],angle=-45°, ...
> center=[0,0,-0.7],zoom=6):
```

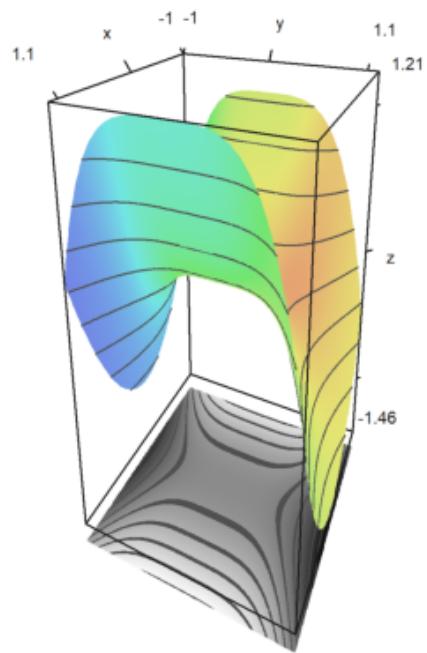


Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual. Perhatikan bahwa `plot3d()` menyetel jendela ke `fullwindow()` secara default, tetapi `plotcontourplane()` mengasumsikan itu.

```
>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);
```



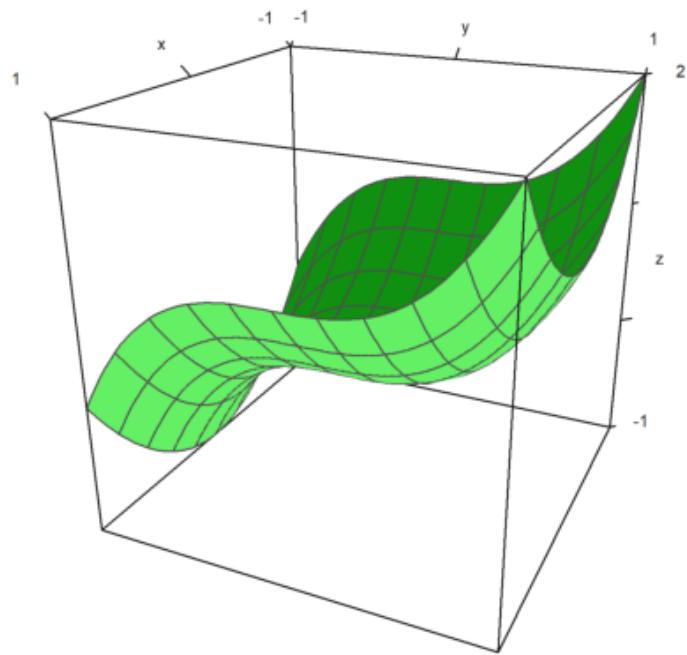
Animasi

Euler dapat menggunakan frame untuk menghitung animasi terlebih dahulu.

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

Silakan pelajari sumber rotasi untuk melihat lebih detail.

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



*Menggambar Povray

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

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

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

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

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

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

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

Anda perlu memuat file povray.

```
>load povray;
```

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

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

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

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

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

```
>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;
```

```
>pov3d ("x^3+y^3", 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;
```

Kita dapat membuat fungsi menjadi transparan dan menambahkan hasil akhir lainnya. Kami juga dapat menambahkan garis level ke plot fungsi.

```
>pov3d("x^2+y^3",axiscolor=red,angle=20°, ...
> look=povlook(blue,0.2),level=-1:0.5:1,zoom=3.8);
```

```
>pov3d("x^2+y^4",axiscolor=red,angle=25°, ...
> look=povlook(blue,0.2),level=-1:0.5:1,zoom=3.8);
```

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

Kami memplot himpunan titik di bidang kompleks, di mana produk dari jarak ke 1 dan -1 sama dengan 1.

```
>pov3d("((x-1)^2+y^2)*((x+1)^2+y^2)/40",r=1.5, ...
> angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=45°,n=50, ...
> <fscale,zoom=3.8);
>pov3d("((x-1)^3+y^3)*((x+1)^2+y^2)/40",r=1.5, ...
> angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=90°,n=50, ...
> <fscale,zoom=3.8);
```

Merencanakan dengan Koordinat

Alih-alih fungsi, kita dapat memplot dengan koordinat. Seperti pada plot3d, kita membutuhkan tiga matriks untuk mendefinisikan objek.

Dalam contoh kita memutar fungsi di sekitar sumbu z.

```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,8)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,height=20°,axis=0,zoom=4,light=[10,-5,5]);
>function f(x) := x^4+x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,8)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,height=20°,axis=0,zoom=4,light=[10,-5,5]);
```

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

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

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

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

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

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

$$\begin{matrix} 2 & 3 \\ x & y \end{matrix}$$

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

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

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

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

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

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

Lihat: Contoh\Trefoil Simpul | Simpul trefoil

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

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

Kemudian kedua vektor turunan ke x dan y.

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

Sekarang normal, yang merupakan produk silang dari dua turunan.

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

Kami sekarang mengevaluasi semua ini secara numerik.

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

Vektor normal adalah evaluasi dari ekspresi simbolik dn[i] untuk i=1,2,3. Sintaks untuk ini adalah &"expression"(parameters). Ini adalah alternatif dari metode pada contoh sebelumnya, di mana kita mendefinisikan ekspresi simbolik NX, NY, NZ terlebih dahulu.

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

```

Function povlook not found.
Try list ... to find functions!
Error in:
... ,zoom=5,w=450,h=350,    <shadow,look=povlook(gray),    xv=&"dn[1 ...
^

```

Kami juga dapat menghasilkan grid dalam 3D.

```

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

```

```

Function povstart not found.
Try list ... to find functions!
Error in:
povstart(zoom=4); x=-1:0.5:1; r=1-(x+1)^2/6; t=(0°:30°:360°)'; ...
^

```

Dengan povgrid(), kurva dimungkinkan.

```

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

```

```

Function povstart not found.
Try list ... to find functions!
Error in:
povstart(center=[0,0,1],zoom=3.6); t=linspace(0,2,1000); r=exp ...
^

```

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 file Povray.

Kami memulai output dengan povstart().

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
povstart(zoom=4); ...  
^
```

Pertama kita mendefinisikan tiga silinder, dan menyimpannya dalam string di Euler. Fungsi povx() dll. hanya mengembalikan vektor [1,0,0], yang dapat digunakan sebagai gantinya.

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

String berisi kode Povray, yang tidak perlu kita pahami pada saat itu.

```
>c1
```

```
Variable povx not found!  
Error in:  
c1=povcylinder(-povx,povx,1,povlook(red)); c2=povcylinder(-pov ...  
^
```

Seperti yang Anda lihat, kami menambahkan tekstur ke objek dalam tiga warna berbeda. Itu dilakukan oleh povlook(), yang mengembalikan string dengan kode Povray yang relevan. Kita dapat menggunakan warna Euler default, atau menentukan warna kita sendiri. Kami juga dapat menambahkan transparansi, atau mengubah cahaya sekitar.

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

```
Function povlook not found.  
Try list ... to find functions!  
Error in:  
povlook(rgb(0.1,0.2,0.3),0.1,0.5) ...  
^
```

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

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

```
Variable c1 not found!  
Error in:  
writeln(povintersection([c1,c2,c3])); ...  
^
```

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

```
>povend;
```

```
Variable povend not found!
Error in:
povend; ...  
^
```

Fungsi berikut menghasilkan fraktal secara rekursif.

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

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

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

```
>povstart(fade=10,<shadow);
```

```
Function povstart not found.
Try list ... to find functions!
Error in:
povstart(fade=10,<shadow); ...  
^
```

```
>fractal(-1,-1,-1,2,4);
```

```
Function povlook not found.  
Try list ... to find functions!  
onebox:  
    useglobal; return povbox([x,y,z],[x+d,y+d,z+d],povlook());  
fractal:  
    if n==1 then writeln(onebox(x,y,z,h));  
fractal:  
    fractal(x,y,z,h,n-1);  
fractal:  
    fractal(x,y,z,h,n-1);  
Try "trace errors" to inspect local variables after errors.  
fractal:  
    fractal(x,y,z,h,n-1);
```

```
>povend();
```

```
Function povend not found.  
Try list ... to find functions!  
Error in:  
povend(); ...  
^
```

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

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
povstart(light=[5,-5,5],fade=10); ...  
^
```

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

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

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

```
Function povbox not found.  
Try list ... to find functions!  
Error in:  
povdefine ("mycube", povbox(-1,1)); ...  
^
```

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

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

```
Function povlook not found.  
Try list ... to find functions!  
Error in:  
c1=povobject ("mycube", povlook(red)); ...  
^
```

Kami menghasilkan kubus kedua, dan memutar dan menskalakannya sedikit.

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

```
Function povlook not found.  
Try list ... to find functions!  
Error in:  
c2=povobject ("mycube", povlook(yellow), translate=[1,1,1], rot ...  
^
```

Kemudian kita ambil selisih kedua benda tersebut.

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

```
Variable or function c1 not found.  
Error in:  
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();
```

```
Function writeAxis not found.  
Try list ... to find functions!  
Error in:  
writeAxis(-1.2,1.2, axis=1); writeAxis(-1.2,1.2, axis=2); writeA ...  
^
```

Fungsi Implisit

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

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

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
povstart(angle=70°, height=50°, zoom=4); ...  
^
```

Buat permukaan implisit. Perhatikan sintaks yang berbeda dalam ekspresi.

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

```
Function povlook not found.  
Try list ... to find functions!  
Error in:  
... ace("pow(x,2)*y-pow(y,3)-pow(z,2)", povlook(green)); writeAxes ...  
^
```

Objek Jala

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

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

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
povstart(angle=-10°,center=[0.5,0.5,0.5],zoom=7); ...  
^
```

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

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

```
Function povtriangles not found.  
Try list ... to find functions!  
Error in:  
mesh=povtriangles(x,y,z,"",vx,vy,vz); ...  
^
```

Sekarang kita mendefinisikan 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);
```

```
Function povdisc not found.  
Try list ... to find functions!  
Error in:  
cl=povdisc([0.5,0.5,0],[1,1,0],2); ll=povdisc([0,0,1/4],[0,0,1 ...  
^
```

Tulis permukaan dikurangi dua cakram.

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

```
Variable or function mesh not found.  
Error in:  
writeln(povdifference(mesh,povunion([cl,ll]),povlook(green))); ...  
^
```

Tulis dua persimpangan.

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

Variable mesh not found!
Error in:
writeln(povintersection([mesh,cl],povlook(red))); writeln(povi ...
^

Tulis titik maksimum.

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

Function povlook not found.
Try list ... to find functions!
Error in:
writeln(povpoint([1/2,1/2,1/4],povlook(gray),size=2*defaultpoi ...
^

Tambahkan sumbu dan selesaikan.

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

Function writeAxes not found.
Try list ... to find functions!
Error in:
writeAxes(0,1,0,1,0,1,d=0.015); povend(); ...
^

Anaglyph di Povray

Untuk menghasilkan anaglyph untuk kacamata merah/sian, Povray harus berjalan dua kali dari posisi kamera yang berbeda. Ini menghasilkan dua file Povray dan dua file PNG, yang dimuat dengan fungsi loadanaglyph().

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

Fungsi pov3d() memiliki sakelar sederhana untuk menghasilkan anaglyphs.

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

```
Function pov3d not found.  
Try list ... to find functions!  
Error in:  
... eight=45°,>anaglyph,    center=[0,0,0.5],zoom=3.5); ...  
^
```

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

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

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

Fungsi povanaglyph() melakukan semua ini. Parameternya seperti di povstart() dan povend() digabungkan.

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

```
Function povanaglyph not found.  
Try list ... to find functions!  
Error in:  
povanaglyph("myscene",zoom=4.5); ...  
^
```

Mendefinisikan Objek sendiri

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

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

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

=Inilah torus pertama kami.

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

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

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

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

```
Function xrotate not found.  
Try list ... to find functions!  
Error in:  
t2=povobject(t1,rotate=xrotate(90°),translate=[0.8,0,0]) ...  
^
```

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

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

```
>povend();
```

memanggil program Povray. Namun, jika terjadi kesalahan, itu tidak menampilkan kesalahan. Karena itu Anda harus menggunakan

```
>povend(<keluar>);
```

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

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
... rt(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); writeln(pov ...  
^
```

Berikut adalah contoh yang lebih rumit. Kami memecahkan lateks: $Ax \leq b$, $x \geq 0$, $c \cdot x \rightarrow \text{Max.}$ dan menunjukkan titik layak dan 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 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 lateks: $a \cdot x \leq b$

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

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

Kemudian kita mendefinisikan persimpangan 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
```

Kita sekarang dapat merencanakan adegannya.

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

Berikut ini adalah lingkaran di sekitar optimal.

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

```
Function povstart not found.  
Try list ... to find functions!  
Error in:  
... ovstart(angle=120°,center=[0.5,0.5,0.5],zoom=3.5); writeln(adm ...  
^
```

Dan kesalahan ke arah yang optimal.

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

```
Function povlook not found.  
Try list ... to find functions!  
Error in:  
writeln(povarrow(x,c*0.5,povlook(red))); ...  
^
```

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

```
>writeln(povtext("Linear Problem",[0,0.2,1.3],size=0.05,rotate=125°)); ...  
>povend();
```

```
Function povtext not found.  
Try list ... to find functions!  
Error in:  
... inear Problem",[0,0.2,1.3],size=0.05,rotate=125°)); povend(); ...  
^
```

LATIHAN - LATIHAN SOAL

Fungsi Dua Variabel atau Lebih

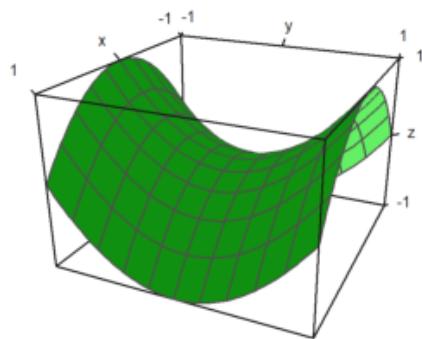
Grafik dari fungsi f dengan dua variabel yang dimaksud adalah grafik

dari persamaan $z = f(x,y)$. Biasanya grafik ini berupa permukaan dan karena setiap (x,y) di daerah asal hanya berpadanan dengan satu nilai z , maka setiap garis tegak lurus bidang-xy memotong permukaan pada paling banyak satu titik.

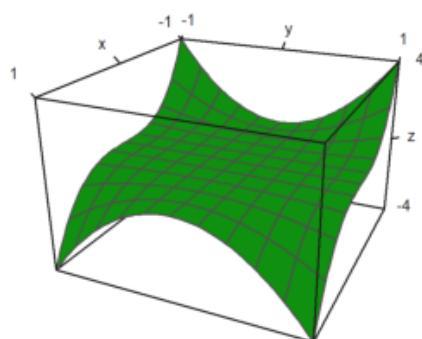
Soal :

1. Grafik merupakan sebuah paraboloida $f(x,y) = y^2 - x^2$
2. $z = -4x^3y^2$
3. $z = xy \exp(-x^2-y^2)$
4. $z = x - 1/8x^3 - 1/3y^2$

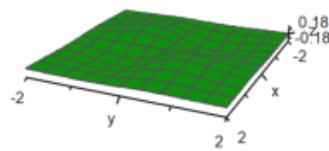
```
>aspect(1.5); plot3d("y^2-x^2"):
```



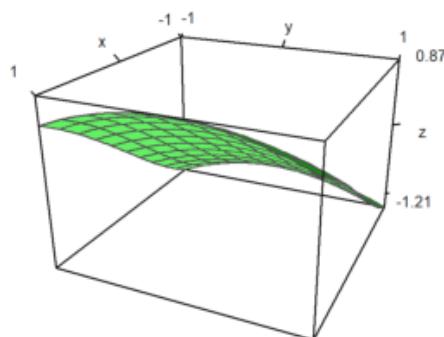
```
>aspect(1.5); plot3d("-4x^3*y^2"):
```



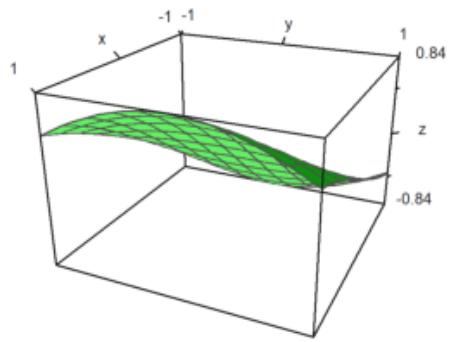
```
>plot3d("x*y*exp(-x^2-y^2)",r=2,<fscale,<scale,distance=13,height=20°, ...
> center=[0,0,-0.2],frame=3):
```



```
>aspect(1.5); plot3d("x-1/8*x^3-1/3*y^2"):
```



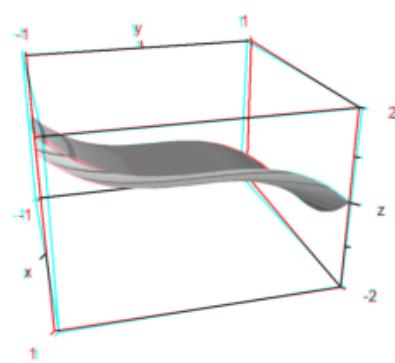
```
>plot3d("sin(x)*cos(y)":
```



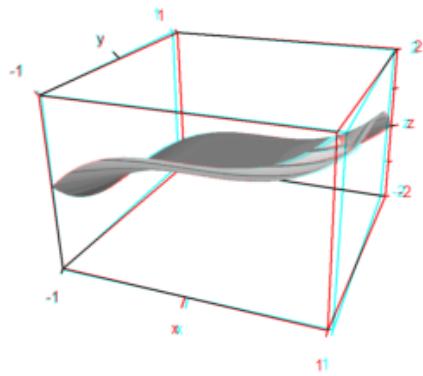
Membuat Talang (Perosotan pada kolam renang)

Dalam membuat perosotan kolam renang ini saya belum bisa membuat secara bagus, karena perosotan yang saya buat masih kurang jadi dan bentuknya pun masih dari arah kiri ke kanan, bahkan terlihat dari sisi belakang dan dari samping, belum terlihat dari depan.

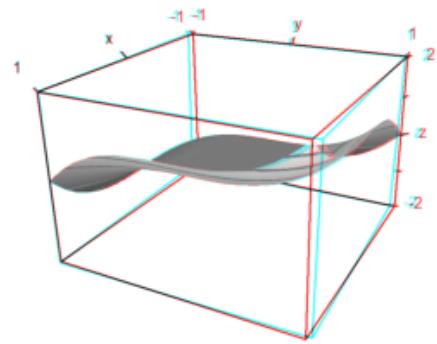
```
>plot3d("x^3-y^3",>anaglyph,>contour,angle=75°):
```



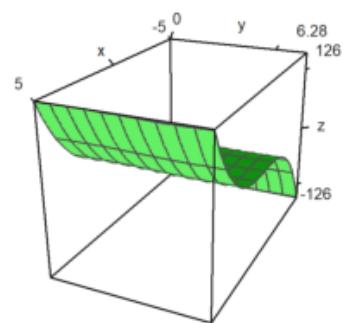
```
>plot3d("x^3-y^3",>anaglyph,>contour,angle=20°):
```



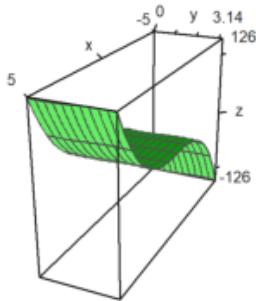
```
>plot3d("x^3+y^3",>anaglyph,>contour,angle=116°):
```



```
>aspect(1.5); plot3d("x^3+sin(y)",-5,5,0,2*pi):
```



```
>aspect(1.5); plot3d("x^3-sin(2y)",-5,5,0,pi):
```



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

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

```
>povstart(center=[0,0,1],zoom=3.6); ...
>t=linspace(0,2,1000); r=exp(-t); ...
>x=cos(2*pi*20*t)*r; y=sin(2*pi*10*t)*r; z=t; ...
>writeln(povgrid(x,y,z,povlook(red))); ...
>writeAxis(0,2,axis=4); ...
>povend();
```

```
>povstart(center=[0,0,1],zoom=3); ...
>t=linspace(0,2,1000); r=exp(-t); ...
>x=cos(pi*10*t)*r; y=sin(pi*10*t)*r; z=t; ...
>writeln(povgrid(x,y,z,povlook(yellow))); ...
>writeAxis(0,2,axis=4); ...
>povend();
>povstart(zoom=3); ...
>x=-1:0.5:1; r=1-(x+1)^3/6; ...
>t=(0°:30°:360°)'; y=r*cos(t); z=r*sin(t); ...
>writeln(povgrid(x,y,z,d=0.02,dballs=0.05)); ...
>povend();
```

Lebih Banyak Contoh

Anda dapat menemukan beberapa contoh lagi untuk Povray di Euler di file berikut.

See: Examples/Dandelin Spheres

See: Examples/Donut Math

See: Examples/Trefoil Knot

See: Examples/Optimization by Affine Scaling

BAB 5

KB Pekan 11-12 : Menggunakan EMT untuk Geometri

Visualisasi dan Perhitungan Geometri dengan EMT

Nama : Dessy Chintya Nur Feika
NIM : 22305141023
Kelas : Matematika E 2022

Euler menyediakan beberapa fungsi untuk melakukan visualisasi dan perhitungan geometri, baik secara numerik maupun analitik (seperti biasanya tentunya, menggunakan Maxima). Fungsi-fungsi untuk visualisasi dan perhitungan geometri tersebut disimpan di dalam file program "geometry.e", sehingga file tersebut harus dipanggil sebelum menggunakan fungsi-fungsi atau perintah-perintah untuk geometri.

```
>load geometry
```

Numerical and symbolic geometry.

Fungsi-fungsi Geometri

Fungsi-fungsi untuk Menggambar Objek Geometri:

```
defaultd:=textheight()*1.5: nilai asli untuk parameter d  
setPlotrange(x1,x2,y1,y2): menentukan rentang x dan y pada bidang
```

koordinat

```
setPlotRange(r): pusat bidang koordinat (0,0) dan batas-batas  
sumbu-x dan y adalah -r sd r
```

```
plotPoint (P, "P"): menggambar titik P dan diberi label "P"  
plotSegment (A,B, "AB", d): menggambar ruas garis AB, diberi label  
"AB" sejauh d
```

```
plotLine (g, "g", d): menggambar garis g diberi label "g" sejauh d  
plotCircle (c,"c",v,d): Menggambar lingkaran c dan diberi label "c"  
plotLabel (label, P, V, d): menuliskan label pada posisi P
```

Fungsi-fungsi Geometri Analitik (numerik maupun simbolik):

```
turn(v, phi): memutar vektor v sejauh phi  
turnLeft(v): memutar vektor v ke kiri  
turnRight(v): memutar vektor v ke kanan  
normalize(v): normal vektor v  
crossProduct(v, w): hasil kali silang vektorv dan w.  
lineThrough(A, B): garis melalui A dan B, hasilnya [a,b,c] sdh.
```

$ax+by=c$.

```
lineWithDirection(A,v): garis melalui A searah vektor v  
getLineDirection(g): vektor arah (gradien) garis g  
getNormal(g): vektor normal (tegak lurus) garis g  
getPointOnLine(g): titik pada garis g  
perpendicular(A, g): garis melalui A tegak lurus garis g  
parallel (A, g): garis melalui A sejajar garis g  
lineIntersection(g, h): titik potong garis g dan h  
projectToLine(A, g): proyeksi titik A pada garis g  
distance(A, B): jarak titik A dan B  
distanceSquared(A, B): kuadrat jarak A dan B  
quadrance(A, B): kuadrat jarak A dan B  
areaTriangle(A, B, C): luas segitiga ABC  
computeAngle(A, B, C): besar sudut  $\angle ABC$   
angleBisector(A, B, C): garis bagi sudut  $\angle ABC$   
circleWithCenter (A, r): lingkaran dengan pusat A dan jari-jari r  
getCircleCenter(c): pusat lingkaran c  
getCircleRadius(c): jari-jari lingkaran c  
circleThrough(A,B,C): lingkaran melalui A, B, C  
middlePerpendicular(A, B): titik tengah AB  
lineCircleIntersections(g, c): titik potong garis g dan lingkaran c  
circleCircleIntersections (c1, c2): titik potong lingkaran c1 dan
```

c2

planeThrough(A, B, C) : bidang melalui titik A, B, C

Fungsi-fungsi Khusus Untuk Geometri Simbolik:

getLineEquation (g,x,y) : persamaan garis g dinyatakan dalam x dan y
getHesseForm (g,x,y,A) : bentuk Hesse garis g dinyatakan dalam x dan

y dengan titik A pada sisi positif (kanan/atas) garis

quad(A,B) : kuadrat jarak AB

spread(a,b,c) : Spread segitiga dengan panjang sisi-sisi a,b,c, yakni
 $\sin(\alpha)^2$ dengan alpha sudut yang menghadap sisi a.

crosslaw(a,b,c,sa) : persamaan 3 quads dan 1 spread pada segitiga
dengan panjang sisi a, b, c.

triplespread(sa,sb,sc) : persamaan 3 spread sa, sb, sc yang memebntuk
suatu segitiga

doublespread(sa) : Spread sudut rangkap Spread 2ϕ , dengan
 $sa=\sin(\phi)^2$ spread a.

Contoh 1: Luas, Lingkaran Luar, Lingkaran Dalam Segitiga

Untuk menggambar objek-objek geometri, langkah pertama adalah menentukan rentang sumbu-sumbu koordinat. Semua objek geometri akan digambar pada satu bidang koordinat, sampai didefinisikan bidang koordinat yang baru.

```
>setPlotRange (-0.5,2.5,-0.5,2.5); // mendefinisikan bidang koordinat baru
```

Sekarang atur tiga poin dan plot.

```
>A=[1,0]; plotPoint(A,"A"); // definisi dan gambar tiga titik  
>B=[0,1]; plotPoint(B,"B");  
>C=[2,2]; plotPoint(C,"C");
```

Lalu tiga segmen.

```
>plotSegment(A,B,"c"); // c=AB  
>plotSegment(B,C,"a"); // a=BC  
>plotSegment(A,C,"b"); // b=AC
```

Fungsi geometri meliputi fungsi untuk membuat garis dan lingkaran. Format untuk garis adalah [a, b, c], yang merepresentasikan garis dengan persamaan $ax + by = c$.

```
>lineThrough(B,C) // garis yang melalui B dan C
```

```
[-1, 2, 2]
```

Hitung garis tegak lurus melalui A pada BC.

```
>h=perpendicular(A,lineThrough(B,C)); // garis h tegak lurus BC melalui A
```

Dan perpotongannya dengan BC.

```
>D=lineIntersection(h,lineThrough(B,C)); // D adalah titik potong h dan BC
```

Plot itu.

```
>plotPoint(D,value=1); // koordinat D ditampilkan  
>aspect(1); plotSegment(A,D); // tampilkan semua gambar hasil plot...()
```

Hitung luas ABC:

$$L_{\triangle ABC} = \frac{1}{2}AD \cdot BC.$$

```
>norm(A-D)*norm(B-C)/2 // AD=norm(A-D), BC=norm(B-C)
```

1.5

Cara lain menghitung rumus determinan.

```
>areaTriangle(A,B,C) // hitung luas segitiga langsung dengan fungsi
```

1.5

Cara lain menghitung luas segitiga ABC:

```
>distance(A,D)*distance(B,C)/2
```

1.5

Sudut di C

```
>degprint(computeAngle(B,C,A))
```

$36^\circ 52' 11.63''$

Sekarang lingkaran sirkuit segitiga.

```
>c=circleThrough(A,B,C); // lingkaran luar segitiga ABC
>R=getCircleRadius(c); // jari2 lingkaran luar
>O=getCircleCenter(c); // titik pusat lingkaran c
>plotPoint(O,"O"); // gambar titik "O"
>plotCircle(c,"Lingkaran luar segitiga ABC"):
```

Tampilkan koordinat titik pusat dan jari-jari lingkaran luar.

```
>O, R
```

```
[1.16667, 1.16667]
1.17851130198
```

Sekarang akan digambar lingkaran dalam segitiga ABC. Titik pusat lingkaran dalam adalah titik potong garis-garis bagi sudut.

```
>l=angleBisector(A,C,B); // garis bagi <ACB  
>g=angleBisector(C,A,B); // garis bagi <CAB  
>P=lineIntersection(l,g) // titik potong kedua garis bagi sudut
```

[0.86038, 0.86038]

Tambahkan semua ke plot.

```
>color(5); plotLine(l); plotLine(g); color(1); // gambar kedua garis bagi s  
>plotPoint(P,"P"); // gambar titik potongnya  
>r=norm(P-projectToLine(P,lineThrough(A,B))) // jari-jari lingkaran dalam
```

0.509653732104

```
>plotCircle(circleWithCenter(P,r),"Lingkaran dalam segitiga ABC"); // gambar
```

Latihan

1. Tentukan ketiga titik singgung lingkaran dalam dengan sisi-sisi segitiga ABC.
2. Gambar segitiga dengan titik-titik sudut ketiga titik singgung tersebut.
3. Tunjukkan bahwa garis bagi sudut yang ketiga juga melalui titik pusat lingkaran dalam.
4. Gambar jari-jari lingkaran dalam.

Jawab:

1. Tentukan ketiga titik singgung lingkaran dalam dengan sisi-sisi segitiga ABC.
Titik singgung garis BC dengan lingkaran dalam.

```
>s=lineThrough(B,C)
```

[-1, 2, 2]

```
>m=circleWithCenter(P,r)
```

[0.86038, 0.86038, 0.509654]

```
>S=lineCircleIntersections(s,m)
```

[0.632456, 1.31623]

Titik singgung garis AC dengan lingkaran dalam.

```
>p=lineThrough(A,C)
```

```
[-2, 1, -2]
```

```
>Q=lineCircleIntersections(p,m)
```

```
[1.31623, 0.632456]
```

Titik singgung garis AB dengan lingkaran dalam.

```
> q=lineThrough(A,B)
```

```
[-1, -1, -1]
```

```
>L=lineCircleIntersections(q,m)
```

```
[0.5, 0.5]
```

Jadi titik singgung lingkaran dalam dengan sisi-sisi segitiga adalah $(0.632456, 1.31623)$, $(1.31632, 0.632456)$, $(0.5, 0.5)$.

2. Gambar segitiga dengan titik-titik sudut ketiga titik singgung tersebut.

```
>plotSegment(S,Q,"a");
>plotSegment(S,L,"b");
>plotSegment(L,Q,"c");
```

3. Tunjukkan bahwa garis bagi sudut yang ke tiga juga melalui titik pusat lingkaran dalam.

```
> P, r
```

```
[0.86038, 0.86038]
```

```
0.509653732104
```

```
>k=angleBisector(A,B,C)
```

```
[-0.264911, -1.63246, -1.63246]
```

```
>color(2); plotLine(k):
```

4. Gambar jari-jari lingkaran dalam.

```
>plotSegment(P,L,"r"):
```

Contoh 2: Geometri Smbolik

Kita dapat menghitung geometri tepat dan simbolis menggunakan Maxima.

Geometri file.e menyediakan fungsi yang sama (dan lebih banyak lagi) di Maxima. Namun, sekarang kita dapat menggunakan perhitungan simbolik.

```
>A &= [1,0]; B &= [0,1]; C &= [2,2]; // menentukan tiga titik A, B, C
```

Fungsi garis dan lingkaran bekerja seperti fungsi Euler, tetapi menyediakan penghitungan simbolik.

```
>c &= lineThrough(B,C) // c=BC
```

```
[- 1, 2, 2]
```

Kita bisa mendapatkan persamaan untuk sebuah garis dengan mudah.

```
>$getLineEquation(c,x,y), $solve(%,y) | expand // persamaan garis c  
>$getLineEquation(lineThrough(A,[x1,y1]),x,y) // persamaan garis melalui A  
>h &= perpendicular(A,lineThrough(B,C)) // h melalui A tegak lurus BC
```

```
[2, 1, 2]
```

```
>Q &= lineIntersection(c,h) // Q titik potong garis c=BC dan h
```

```
2   6  
[-, -]  
5   5
```

```
>$projectToLine(A,lineThrough(B,C)) // proyeksi A pada BC  
>$distance(A,Q) // jarak AQ  
>cc &= circleThrough(A,B,C); $cc // (titik pusat dan jari-jari) lingkaran m  
>r=>getCircleRadius(cc); $r , $float(r) // tampilan nilai jari-jari  
>$computeAngle(A,C,B) // nilai <ACB  
>$solve(getLineEquation(angleBisector(A,C,B),x,y),y)[1] // persamaan garis  
>P &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A)); $P // ti  
>P() // hasilnya sama dengan perhitungan sebelumnya
```

```
[0.86038, 0.86038]
```

Garis dan Lingkaran yang Berpotongan

Tentu saja, kita juga bisa memotong garis dengan lingkaran, dan lingkaran dengan lingkaran.

```
>A &:= [1,0]; c=circleWithCenter(A,4);  
>B &:= [1,2]; C &:= [2,1]; l=lineThrough(B,C);  
>setPlotRange(5); plotCircle(c); plotLine(l);
```

Perpotongan garis dengan lingkaran mengembalikan dua titik dan jumlah titik perpotongan.

```
>{P1,P2,f}=lineCircleIntersections(l,c);  
>P1, P2,
```

```
[4.64575, -1.64575]  
[-0.645751, 3.64575]
```

```
>plotPoint(P1); plotPoint(P2);
```

Hal yang sama di Maxima.

```
>c &= circleWithCenter(A, 4) // lingkaran dengan pusat A jari-jari 4
```

```
[1, 0, 4]
```

```
>l &= lineThrough(B,C) // garis l melalui B dan C
```

```
[1, 1, 3]
```

```
>$lineCircleIntersections(l,c) | radcan, // titik potong lingkaran c dan ga
```

Akan ditunjukkan bahwa sudut-sudut yang menghadap bsuusr yang sama adalah sama besar.

```
>C=A+normalize([-2,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C)
```

$69^\circ 17' 42.68''$

```
>C=A+normalize([-4,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C)
```

$69^\circ 17' 42.68''$

```
>insimg;
```

Garis Sumbu

Berikut adalah langkah-langkah menggambar garis sumbu ruas garis AB:

1. Gambar lingkaran dengan pusat A melalui B.
2. Gambar lingkaran dengan pusat B melalui A.
3. Tarik garis melalui kedua titik potong kedua lingkaran tersebut. Garis ini merupakan garis sumbu (melalui titik tengah dan tegak lurus) AB.

```
>A=[2,2]; B=[-1,-2];
>c1=circleWithCenter(A,distance(A,B));
>c2=circleWithCenter(B,distance(A,B));
>{P1,P2,f}=circleCircleIntersections(c1,c2);
>l=lineThrough(P1,P2);
>setPlotRange(5); plotCircle(c1); plotCircle(c2);
>plotPoint(A); plotPoint(B); plotSegment(A,B); plotLine(l):
```

Selanjutnya, kita melakukan hal yang sama di Maxima dengan koordinat umum.

```
>A &= [a1,a2]; B &= [b1,b2];
>c1 &= circleWithCenter(A,distance(A,B));
>c2 &= circleWithCenter(B,distance(A,B));
>P &= circleCircleIntersections(c1,c2); P1 &= P[1]; P2 &= P[2];
```

Persamaan untuk persimpangan cukup terlibat. Tapi kita bisa menyederhanakan, jika kita menyelesaikan y.

```
>g &= getLineEquation(lineThrough(P1,P2),x,y);
>$solve(g,y)
```

Ini memang sama dengan tengah tegak lurus, yang dihitung dengan cara yang sama sekali berbeda.

```
>$solve(getLineEquation(middlePerpendicular(A,B),x,y),y)
>h &= getLineEquation(lineThrough(A,B),x,y);
>$solve(h,y)
```

Perhatikan hasil kali gradien garis g dan h adalah:

$$\frac{-(b_1 - a_1)}{(b_2 - a_2)} \times \frac{(b_2 - a_2)}{(b_1 - a_1)} = -1.$$

Artinya kedua garis tegak lurus.

Contoh 3: Rumus Heron

Rumus Heron menyatakan bahwa luas segitiga dengan panjang sisi-sisi a , b dan c adalah:

$$L = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{dengan } s = (a+b+c)/2.$$

Untuk membuktikan hal ini kita misalkan $C(0,0)$, $B(a,0)$ dan $A(x,y)$, $b=AC$, $c=AB$. Luas segitiga ABC adalah

$$L_{\triangle ABC} = \frac{1}{2}a \times y.$$

Nilai y didapat dengan menyelesaikan sistem persamaan:

$$x^2 + y^2 = b^2, \quad (x-a)^2 + y^2 = c^2.$$

```
>setPlotRange (-1,10,-1,8); plotPoint([0,0], "C(0,0)"); plotPoint([5.5,0], "
>plotPoint([7.5,6], "A(x,y)");
>plotSegment([0,0],[5.5,0], "a",25); plotSegment([5.5,0],[7.5,6],"c",15);
>plotSegment([0,0],[7.5,6], "b",25);
>plotSegment([7.5,6],[7.5,0], "t=y",25):
>&assume(a>0); sol &= solve([x^2+y^2=b^2, (x-a)^2+y^2=c^2], [x,y])
```

$$\begin{aligned} & -c^2 + b^2 + a^2 \\ [x] &= \frac{\sqrt{-c^4 + 2b^2c^2 + 2a^2c^2 - b^4 + 2a^2b^2 - a^4}}{2a}, \quad y = \\ & -\frac{\sqrt{-c^4 + 2b^2c^2 + 2a^2c^2 - b^4 + 2a^2b^2 - a^4}}{2a}, \\ [x] &= \frac{\sqrt{-c^4 + 2b^2c^2 + 2a^2c^2 - b^4 + 2a^2b^2 - a^4}}{2a}, \quad y = \\ & -\frac{\sqrt{-c^4 + 2b^2c^2 + 2a^2c^2 - b^4 + 2a^2b^2 - a^4}}{2a} \end{aligned}$$

Ekstrak solusi y

```
>ysol &= y with sol[2][2]; $'y=sqrt(factor(ysol^2))
```

Kita mendapatkan formula Heron.

```
>function H(a,b,c) &= sqrt(factor((ysol*a/2)^2)); $'H(a,b,c)=H(a,b,c)
```

Tentu saja, setiap segitiga persegi panjang adalah kasus yang terkenal.

```
>H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5
```

6

Dan jelas juga, bahwa ini adalah segitiga dengan luas maksimal dan kedua sisinya 3 dan 4.

```
>aspect (1.5); plot2d(&H(3,4,x),1,7); // Kurva luas segitiga sengan panjang
```

Kasus umum juga berfungsi.

```
>$solve(diff(H(a,b,c)^2,c)=0,c)
```

Sekarang mari kita cari himpunan semua titik di mana $b + c = d$ untuk beberapa konstanta d . Diketahui bahwa ini adalah elips.

```
>s1 &= subst(d-c,b,sol[2]); $s1
```

Dan membuat persamaan seperti ini

```
>function fx(a,c,d) &= rhs(s1[1]); $fx(a,c,d), function fy(a,c,d) &= rhs(s1
```

Sekarang kita bisa menggambar setnya. Sisi b bervariasi dari 1 hingga 4. Diketahui bahwa kita mendapatkan elips.

```
>aspect(1); plot2d(&fx(3,x,5),&fy(3,x,5),xmin=1,xmax=4,square=1):
```

Kita dapat memeriksa persamaan umum ellips ini, yaitu.

$$\frac{(x - x_m)^2}{u^2} + \frac{(y - y_m)^2}{v^2} = 1,$$

di mana (x_m, y_m) adalah pusat, dan u dan v adalah setengah sumbu.

```
>$ratsimp((fx(a,c,d)-a/2)^2/u^2+fy(a,c,d)^2/v^2 with [u=d/2,v=sqrt(d^2-a^2)]
```

Kita melihat bahwa tinggi dan luas segitiga adalah maksimal untuk $x = 0$. Jadi luas segitiga dengan $a + b + c = d$ adalah maksimal, jika sama sisi. Kami ingin mendapatkan ini secara analitis.

```
>eqns &= [diff(H(a,b,d-(a+b))^2,a)=0,diff(H(a,b,d-(a+b))^2,b)=0]; $eqns
```

Kami mendapatkan beberapa minima, yang termasuk dalam segitiga dengan satu sisi 0, dan solusi $a=b=c=d/3$.

```
>$solve(eqns,[a,b])
```

Ada juga metode Lagrange, memaksimalkan $H(a,b,c)^2$ terhadap $a+b+d=d$.

```
>&solve([diff(H(a,b,c)^2,a)=la,diff(H(a,b,c)^2,b)=la, ...
> diff(H(a,b,c)^2,c)=la,a+b+c=d],[a,b,c,la])
```

$$\begin{aligned} & \left[\left[a = 0, b = \frac{d}{2}, c = \frac{d}{2}, la = 0 \right], \right. \\ & \left[a = \frac{d}{2}, b = 0, c = \frac{d}{2}, la = 0 \right], \left[a = \frac{d}{2}, b = \frac{d}{2}, c = 0, la = 0 \right], \\ & \left. \left[a = \frac{d}{3}, b = \frac{d}{3}, c = \frac{d}{3}, la = \frac{d}{108} \right] \right] \end{aligned}$$

Kita bisa membuat plot situasinya.

Pertama, atur poin di Maxima

```
>A &= at ([x,y],sol[2]); $A  
>B &= [0,0]; $B, C &= [a,0]; $C
```

Kemudian atur rentang plot, dan plot poinnya.

```
>setPlotRange (0,5,-2,3); ...  
>a=4; b=3; c=2; ...  
>plotPoint (mxmeval ("B"), "B"); plotPoint (mxmeval ("C"), "C"); ...  
>plotPoint (mxmeval ("A"), "A");
```

Plot segmennya.

```
>plotSegment (mxmeval ("A"),mxmeval ("C")); ...  
>plotSegment (mxmeval ("B"),mxmeval ("C")); ...  
>plotSegment (mxmeval ("B"),mxmeval ("A"));
```

Hitung tengah tegak lurus di Maxima.

```
>h &= middlePerpendicular(A,B); g &= middlePerpendicular(B,C);
```

Dan bagian tengah dari keliling.

```
>U &= lineIntersection (h,g);
```

Kita mendapatkan rumus untuk jari-jari lingkaran.

```
>&assume (a>0,b>0,c>0); $distance (U,B) | radcan
```

Mari kita tambahkan ini ke plot.

```
>plotPoint (U()); ...  
>plotCircle (circleWithCenter (mxmeval ("U")),mxmeval ("distance (U,C) "));
```

Menggunakan geometri, kita mendapatkan rumus sederhana

$$\frac{a}{\sin(\alpha)} = 2r$$

untuk radius. Kita dapat memeriksa, apakah ini benar dengan Maxima. Maxima akan men faktorkannya hanya jika kita mengkuadratkannya.

```
>$c^2/sin(computeAngle(A,B,C))^2 | factor
```

Contoh 4: Garis Euler dan Parabola

Garis euler adalah garis yang ditentukan dari segitiga yang tidak sama sisi. Ini adalah garis tengah segitiga, dan melewati beberapa titik penting yang ditentukan dari segitiga, termasuk pusat ortosentrum, sirkumenter, pusat massa, titik Exeter, dan pusat lingkaran sembilan titik segitiga.

Untuk demonstrasi, kami menghitung dan memplot garis Euler dalam segitiga.

Pertama, kami menentukan sudut segitiga di Euler. Kami menggunakan definisi, yang terlihat dalam ekspresi simbolik.

```
>A:=[-1,-1]; B:=[2,0]; C:=[1,2];
```

Untuk memplot objek geometris, kami menyiapkan area plot, dan menambahkan poin ke dalamnya. Semua plot objek geometris ditambahkan ke plot saat ini.

```
>setPlotRange(3); plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C");
```

Kita juga bisa menambahkan sisi segitiga.

```
>plotSegment(A,B,""); plotSegment(B,C,""); plotSegment(C,A,"");
```

Berikut adalah luas segitiga menggunakan rumus determinan. Tentu saja kita harus mengambil nilai absolut dari hasil ini.

```
>$areaTriangle(A,B,C)
```

Kita dapat menghitung koefisien dari sisi c.

```
>c &= lineThrough(A,B)
```

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

Dan juga dapatkan rumus untuk baris ini.

```
>$getLineEquation(c,x,y)
```

Untuk bentuk Hesse, kita perlu menentukan titik, sehingga titik tersebut berada di sisi positif dari bentuk Hesse. Memasukkan titik menghasilkan jarak positif ke garis.

```
>$getHesseForm(c,x,y,C), $at(%,[x=C[1],y=C[2]])
```

Sekarang kita menghitung sirkit ABC.

```
>LL &= circleThrough(A,B,C); $getCircleEquation(LL,x,y)
>O &= getCircleCenter(LL); $O
```

Plot lingkaran dan pusatnya. Cu dan U adalah simbolik. Kami mengevaluasi ekspresi ini untuk Euler.

```
>plotCircle(LL()); plotPoint(O(),"O"):
```

Kita dapat menghitung perpotongan ketinggian di ABC (orthocenter) secara numerik dengan perintah berikut.

```
>H &= lineIntersection(perpendicular(A,lineThrough(C,B)),...
> perpendicular(B,lineThrough(A,C))); $H
```

Sekarang kita dapat menghitung garis Euler dari segitiga tersebut.

```
>el &= lineThrough(H,O); $getLineEquation(el,x,y)
```

Tambahkan ke plot kita.

```
>plotPoint(H(),"H"); plotLine(el(),"Garis Euler");
```

Pusat gravitasi harus berada di garis ini.

```
>M &= (A+B+C)/3; $getLineEquation(el,x,y) with [x=M[1],y=M[2]]  
>plotPoint(M(),"M"); // titik berat
```

Teorinya mengatakan bahwa $MH=2*MO$. Kita perlu menyederhanakan dengan radcan untuk mencapai ini.

```
>$distance(M,H)/distance(M,O)|radcan
```

Fungsinya termasuk fungsi untuk sudut juga.

```
>$computeAngle(A,C,B), degprint(%())
```

$60^\circ 15' 18.43''$

Persamaan untuk pusat lingkaran tidak terlalu bagus.

```
>Q &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A))|radcan; $
```

Mari kita hitung juga ekspresi jari-jari lingkaran yang tertulis.

```
>r &= distance(Q,projectToLine(Q,lineThrough(A,B)))|ratsimp; $r  
>LD &= circleWithCenter(Q,r); // Lingkaran dalam
```

Mari kita tambahkan ini ke plot.

```
>color(5); plotCircle(LD());
```

Parabola

Selanjutnya akan dicari persamaan tempat kedudukan titik-titik yang berjarak sama ke titik C dan ke garis AB.

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

Persamaan tersebut dapat digambar menjadi satu dengan gambar sebelumnya.

```
>plot2d(p,level=0,add=1,contourcolor=6):
```

Ini seharusnya menjadi beberapa fungsi, tetapi pemecah default Maxima dapat menemukan solusi hanya, jika persamaan kita kuadratkan. Akibatnya, kami mendapatkan solusi palsu.

```
>akar &= solve(getHesseForm(lineThrough(A,B),x,y,C)^2-distance([x,y],C)^2,y)
```

$$[y = -3x - \sqrt{70} \sqrt{9 - 2x} + 26, \\ y = -3x + \sqrt{70} \sqrt{9 - 2x} + 26]$$

Solusi pertama adalah

maxima: akar[1]

Menambahkan solusi pertama ke pertunjukkan plot, bahwa itu memang jalan yang kita cari. Teori mengatakan kepada kita bahwa itu adalah parabola yang diputar.

```
>plot2d(&rhs(akar[1]),add=1):  
>function g(x) &= rhs(akar[1]); $'g(x)=g(x)// fungsi yang mendefinisikan k  
>T &=[-1, g(-1)]; // ambil sebarang titik pada kurva tersebut  
>dTC &= distance(T,C); $fullratsimp(dTC), $float(%)// jarak T ke C  
>U &= projectToLine(T,lineThrough(A,B)); $U // proyeksi T pada garis AB  
>dU2AB &= distance(T,U); $fullratsimp(dU2AB), $float(%)// jarak T ke AB
```

Ternyata jarak T ke C sama dengan jarak T ke AB. Coba Anda pilih titik T yang lain dan ulangi perhitungan-perhitungan di atas untuk menunjukkan bahwa hasilnya juga sama.

Contoh 5: Trigonometri Rasional

Ini terinspirasi oleh ceramah N.J.Wildberger. Dalam bukunya "Proporsi Agung", Wildberger mengusulkan untuk menggantikan pengertian klasik tentang jarak dan sudut dengan kuadrans dan penyebaran. Dengan menggunakan ini, memang mungkin untuk menghindari fungsi trigonometri dalam banyak contoh, dan tetap "rasional".

Berikut ini, saya memperkenalkan konsep, dan memecahkan beberapa masalah. Saya menggunakan perhitungan simbolik Maxima di sini, yang menyembunyikan keuntungan utama dari trigonometri rasional bahwa perhitungan dapat dilakukan dengan kertas dan pensil saja. Anda diundang untuk memeriksa hasil tanpa komputer.

Intinya adalah bahwa perhitungan rasional simbolis sering kali menghasilkan hasil yang sederhana. Sebaliknya, trigonometri klasik menghasilkan hasil trigonometri yang rumit, yang mengevaluasi ke pendekatan numerik saja.

```
>load geometry;
```

Untuk pendahuluan pertama, kami menggunakan segitiga persegi panjang dengan proporsi Mesir terkenal 3, 4 dan 5. Perintah berikut adalah perintah Euler untuk memplot geometri bidang yang terdapat dalam file Euler "geometry.e".

```
>C&:=[0,0]; A&:=[4,0]; B&:=[0,3]; ...
>setPlotRange (-1,5,-1,5); ...
>plotPoint (A, "A"); plotPoint (B, "B"); plotPoint (C, "C"); ...
>plotSegment (B,A,"c"); plotSegment (A,C,"b"); plotSegment (C,B,"a"); ...
>insimg(30);
```

Tentu saja,

$$\sin(w_a) = \frac{a}{c},$$

di mana w_a adalah sudut di A. Cara biasa untuk menghitung sudut ini, adalah dengan melakukan invers dari fungsi sinus. Hasilnya adalah sudut yang tidak dapat dicerna, yang hanya dapat dicetak secara perkiraan.

```
>wa := arcsin(3/5); degprint(wa)
```

$36^\circ 52' 11.63''$

Trigonometri rasional mencoba menghindari hal ini.

Pengertian pertama dari trigonometri rasional adalah kuadran, yang menggantikan jarak. Faktanya, itu hanyalah kuadrat jarak. Berikut ini, a, b, dan c menunjukkan kuadran sisi-sisinya.

Teorema Pythagoras menjadi $a+b=c$ lalu.

```
>a &= 3^2; b &= 4^2; c &= 5^2; &a+b=c
```

$$25 = 25$$

Gagasan kedua dari trigonometri rasional adalah penyebarannya. Spread mengukur buaan antar baris. Ini adalah 0, jika garis sejajar, dan 1, jika garis persegi panjang. Ini adalah kuadrat dari sinus sudut antara dua garis.

Penyebaran garis AB dan AC pada gambar di atas didefinisikan sebagai

$$s_a = \sin(\alpha)^2 = \frac{a}{c},$$

di mana a dan c adalah kuadrat dari segitiga persegi panjang mana pun dengan satu sudut di A.

```
>sa &= a/c; $sa
```

Ini lebih mudah dihitung daripada sudut, tentu saja. Tetapi Anda kehilangan properti yang sudut dapat ditambahkan dengan mudah.

Tentu saja, kita dapat mengubah nilai perkiraan sudut wa menjadi sprad, dan mencetaknya sebagai pecahan.

```
>fracprint(sin(wa)^2)
```

$$9/25$$

Hukum cosinus dari trigonometri klasik diterjemahkan menjadi "hukum silang" berikut.

$$(c + b - a)^2 = 4bc(1 - s_a)$$

Di sini a, b, dan c adalah kuadran dari sisi-sisi segitiga, dan sa adalah sebaran di sudut A. Sisi a, seperti biasa, berlawanan dengan sudut A.

Hukum ini diimplementasikan dalam file geometry.e yang kami muat ke Euler.

```
>$crosslaw(aa,bb,cc,saa)
```

Dalam kasus kami, kita mendapatkan

```
>$crosslaw(a,b,c,sa)
```

Mari kita gunakan crosslaw ini untuk mencari sebaran di A. Untuk melakukan ini, kita menghasilkan crosslaw untuk kuadran a, b, dan c, dan menyelesaiakannya untuk sebaran yang tidak diketahui sa.

Anda dapat melakukan ini dengan tangan dengan mudah, tetapi saya menggunakan Maxima. Tentu saja, kami mendapatkan hasilnya, kami sudah mendapatkannya.

```
>$crosslaw(a,b,c,x), $solve(%,x)
```

Kami sudah tahu ini. Definisi penyebaran adalah kasus khusus dari hukum lintas hukum. Kita juga bisa menyelesaikan ini untuk umum a, b, c. Hasilnya adalah rumus yang menghitung sebaran sudut segitiga berdasarkan kuadran ketiga sisinya.

```
>$solve(crosslaw(aa,bb,cc,x),x)
```

Kita bisa membuat fungsi dari hasilnya. Fungsi seperti itu sudah ditentukan dalam file geometry.e Euler.

```
>$spread(a,b,c)
```

Sebagai contoh, kita bisa menggunakannya untuk menghitung sudut segitiga bersisi

$$a, \quad a, \quad \frac{4a}{7}$$

Hasilnya rasional, yang tidak mudah didapat jika kita menggunakan trigonometri klasik.

```
>$spread(a,a,4*a/7)
```

Ini adalah sudut dalam derajat.

```
>degprint(arcsin(sqrt(6/7)))
```

67° 47' 32.44''

Contoh lain

Sekarang, mari kita coba contoh yang lebih canggih.

Kami mengatur tiga sudut segitiga sebagai berikut.

```
>A&:=[1,2]; B&:=[4,3]; C&:=[0,4]; ...
>setPlotRange(-1,5,1,7); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg;
```

Menggunakan Pythagoras, mudah untuk menghitung jarak antara dua titik. Saya pertama kali menggunakan jarak fungsi file Euler untuk geometri. Jarak fungsi menggunakan geometri klasik.

```
>$distance(A,B)
```

Euler juga memiliki fungsi kuadrans antara dua titik.

Dalam contoh berikut, karena $c + b$ bukan a , segitiga tidak persegi panjang.

```
>c &= quad(A,B); $c, b &= quad(A,C); $b, a &= quad(B,C); $a,
```

Pertama, mari kita hitung sudut tradisional. Fungsi computeAngle menggunakan metode biasa berdasarkan perkalian titik dari dua vektor. Hasilnya adalah beberapa pendekatan floating point.

```
>wb &= computeAngle(A,B,C); $wb, $(wb/pi*180)()
```

32.4711922908

Menggunakan pensil dan kertas, kita bisa melakukan hal yang sama dengan hukum silang. Kami memasukkan kuadran a , b , dan c ke dalam hukum silang dan menyelesaikan untuk x .

```
>$crosslaw(a,b,c,x), $solve(%,x),
```

Artinya, fungsi penyebaran yang didefinisikan dalam "geometry.e".

```
>sb &= spread(b,a,c); $sb
```

Maxima mendapatkan hasil yang sama dengan menggunakan trigonometri biasa, jika kita memaksakannya. Itu menyelesaikan istilah $\sin(\arccos(...))$ menjadi hasil pecahan. Kebanyakan siswa tidak dapat melakukan ini.

```
>$sin(computeAngle(A,B,C))^2
```

Setelah kita mendapatkan sebaran di B, kita bisa menghitung tinggi ha di sisi a. Ingat bahwa

$$s_b = \frac{h_a}{c}$$

Menurut definisi.

```
>ha &= c*sb; $ha
```

Gambar berikut telah diproduksi dengan program geometri C.a.R., yang dapat menggambarkan kuadran dan menyebar.

gambar : (20) Rational_Geometry_CaR.png

Menurut definisi, panjang ha adalah akar kuadrat dari kuadrannya.

```
>$sqrt(ha)
```

Sekarang kita bisa menghitung luas segitiga. Jangan lupa, bahwa kita berurusan dengan kuadran!

```
>$sqrt(ha)*sqrt(a)/2
```

Rumus determinan yang biasa menghasilkan hasil yang sama.

```
>$areaTriangle(B,A,C)
```

Formula Heron

Sekarang, mari kita selesaikan masalah ini secara umum!

```
>&remvalue(a,b,c,sb,ha);
```

Pertama-tama kita menghitung spread di B untuk segitiga dengan sisi a, b, dan c. Kemudian kami menghitung luas area yang dikuadratkan ("kuadrea"?), Memfaktorkannya dengan Maxima, dan kami mendapatkan rumus Heron yang terkenal.

Memang, ini sulit dilakukan dengan pensil dan kertas.

```
>$spread(b^2,c^2,a^2), $factor(%*c^2*a^2/4)
```

Aturan Triple Spread

Kerugian dari spread adalah bahwa mereka tidak lagi hanya menambahkan sudut serupa. Namun, tiga sebaran segitiga memenuhi aturan "penyebaran rangkap tiga" berikut.

```
>&remvalue(sa,sb,sc); $triplespread(sa,sb,sc)
```

Aturan ini berlaku untuk tiga sudut yang bertambah menjadi 180° .

$$\alpha + \beta + \gamma = \pi$$

Sejak penyebaran

$$\alpha, \pi - \alpha$$

sama, aturan penyebaran tiga kali lipat juga benar, jika

$$\alpha + \beta = \gamma$$

Karena penyebaran sudut negatif adalah sama, aturan penyebaran tiga kali lipat juga berlaku, jika

$$\alpha + \beta + \gamma = 0$$

Misalnya, kita dapat menghitung sebaran sudut 60° . Ini $3/4$. Persamaan memiliki solusi kedua, di mana semua spread adalah 0.

```
>$solve(triplespread(x,x,x),x)
```

Sebaran 90° jelaslah 1. Jika dua sudut dijumlahkan menjadi 90° , penyebarannya menyelesaikan persamaan penyebaran rangkap tiga dengan $a, b, 1$. Dengan perhitungan berikut kita mendapatkan $a+b=1$.

```
>$triplespread(x,y,1), $solve(%,x)
```

Karena penyebaran $180^\circ-t$ sama dengan penyebaran t , rumus penyebaran rangkap tiga juga berlaku, jika satu sudut adalah jumlah atau perbedaan dari dua sudut lainnya.

Jadi kita bisa menemukan sebaran sudut berlipat ganda. Perhatikan bahwa ada dua solusi lagi. Kami menjadikan ini sebuah fungsi.

```
>$solve(triplespread(a,a,x),x), function doublespread(a) &= factor(rhs(%[1]))
```

$$- 4 (a - 1) a$$

Pembagi Sudut

Ini situasinya, kita sudah tahu.

```
>C&:=[0,0]; A&:=[4,0]; B&:=[0,3]; ...
>setPlotRange(-1,5,-1,5); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg;
```

Mari kita hitung panjang bisektor sudut pada A. Tapi kita ingin menyelesaikannya untuk umum a, b, c.

```
>&remvalue(a,b,c);
```

Jadi pertama-tama kita menghitung sebaran sudut terbagi di A, menggunakan rumus sebaran rangkap tiga.

Masalah dengan rumus ini muncul lagi. Ini memiliki dua solusi. Kami harus memilih yang benar. Solusi lainnya mengacu pada sudut terbagi $180^\circ - \omega_a$.

```
>$triplespread(x,x,a/(a+b)), $solve(%,x), sa2 &= rhs(%[1]); $sa2
```

Mari kita periksa persegi panjang Mesir.

```
>$sa2 with [a=3^2,b=4^2]
```

Kami dapat mencetak sudut di Euler, setelah mentransfer penyebaran ke radian.

```
>wa2 := arcsin(sqrt(1/10)); degprint(wa2)
```

$18^\circ 26' 5.82''$

Titik P adalah perpotongan dari garis bagi sudut dengan sumbu y.

```
>P := [0,tan(wa2)*4]
```

[0, 1.33333]

```
>plotPoint(P,"P"); plotSegment(A,P):
```

Mari kita periksa sudut dalam contoh spesifik kita.

```
>computeAngle(C,A,P), computeAngle(P,A,B)
```

0.321750554397
0.321750554397

Sekarang kita menghitung panjang bisektor AP.

Kita menggunakan teorema sinus di segitiga APC. Teorema ini menyatakan bahwa

$$\frac{BC}{\sin(w_a)} = \frac{AC}{\sin(w_b)} = \frac{AB}{\sin(w_c)}$$

memegang di segitiga apa pun. Persegi itu, itu diterjemahkan ke dalam apa yang disebut "hukum penyebaran"

$$\frac{a}{s_a} = \frac{b}{s_b} = \frac{c}{s_b}$$

dimana a, b, c menunjukkan qudrance.

Karena BPA sebaran adalah $1-sa^2$, kita dapatkan darinya bisa / $1 = b / (1-sa^2)$ dan dapat menghitung bisa (kuadran garis-garis).

```
>&factor(ratsimp(b/(1-sa2))); bisa &= %; $bisa
```

Mari kita periksa rumus ini untuk nilai Mesir kita.

```
>sqrt(mxmeval("at(bisa,[a=3^2,b=4^2])")), distance(A,P)
```

```
4.21637021356  
4.21637021356
```

Kami juga dapat menghitung P menggunakan rumus spread.

```
>py&=factor(ratsimp(sa2*bisa)); $py
```

Nilainya sama dengan yang kita dapatkan dengan rumus trigonometri.

```
>sqrt(mxmeval("at(py,[a=3^2,b=4^2])"))
```

```
1.33333333333
```

Sudut Akord

Perhatikan situasi berikut.

```

>setPlotRange(1.2); ...
>color(1); plotCircle(circleWithCenter([0,0],1)); ...
>A:=[cos(1),sin(1)]; B:=[cos(2),sin(2)]; C:=[cos(6),sin(6)]; ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>color(3); plotSegment(A,B,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a");
>color(1); O:=[0,0]; plotPoint(O,"O"); ...
>plotSegment(A,O); plotSegment(B,O); plotSegment(C,O,"r"); ...
>insimg;

```

Kita bisa menggunakan Maxima untuk menyelesaikan rumus sebaran rangkap tiga untuk sudut di pusat O untuk r. Jadi kita mendapatkan rumus untuk jari-jari kuadrat dari keliling dalam hal kuadrat sisi.

Kali ini, Maxima menghasilkan beberapa angka nol yang kompleks, yang kita abaikan.

```

>&remvalue(a,b,c,r); // hapus nilai-nilai sebelumnya untuk perhitungan baru
>rabc &= rhs(solve(triplespread(spread(b,r,r),spread(a,r,r),spread(c,r,r)),

```

Kita bisa menjadikannya sebagai fungsi Euler.

```

>function periradius(a,b,c) &= rabc;

```

Mari kita periksa hasilnya untuk poin A, B, C kita.

```

>a:=quadrance(B,C); b:=quadrance(A,C); c:=quadrance(A,B);

```

Radiusnya memang 1.

```

>periradius(a,b,c)

```

1

Faktanya, penyebaran CBA hanya bergantung pada b dan c. Ini adalah teorema sudut akord.

```

>$spread(b,a,c)*rabc | ratsimp

```

Sebenarnya sebarannya adalah $b/(4r)$, dan kita melihat bahwa sudut akor b adalah setengah dari sudut tengah.

```
>$double$spread(b/(4*r))-spread(b,r,r) | ratsimp
```

Contoh 6: Jarak Minimal pada Bidang

Catatan awal

Fungsi yang, ke titik M di bidang, menetapkan jarak AM antara titik tetap A dan M, memiliki garis level yang agak sederhana: lingkaran berpusat di A.

```
>&remvalue();  
>A=[-1,-1];  
>function d1(x,y):=sqrt((x-A[1])^2+(y-A[2])^2)  
>fcontour("d1",xmin=-2,xmax=0,ymin=-2,ymax=0,hue=1,...  
>title="If you see ellipses, please set your window square":
```

dan grafiknya juga agak sederhana: bagian atas kerucut:

```
>plot3d("d1",xmin=-2,xmax=0,ymin=-2,ymax=0):
```

Tentu saja minimal 0 dicapai di A.

Dua titik

Sekarang kita melihat fungsi MA + MB dimana A dan B adalah dua titik (tetap). Ini adalah "fakta yang terkenal" bahwa kurva level adalah elips, titik fokusnya adalah A dan B; kecuali untuk minimum AB yang konstan pada segmen [AB]:

```
>B=[1,-1];  
>function d2(x,y):=d1(x,y)+sqrt((x-B[1])^2+(y-B[2])^2)  
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
```

Grafiknya lebih menarik:

```
>plot3d("d2", xmin=-2, xmax=2, ymin=-3, ymax=1) :
```

Batasan ke baris (AB) lebih terkenal:

```
>plot2d("abs(x+1)+abs(x-1)", xmin=-3, xmax=3) :
```

Tiga titik

Sekarang hal-hal menjadi kurang sederhana: Sedikit kurang diketahui bahwa $MA+MB+MC$ mencapai minimumnya pada satu titik bidang tetapi untuk menentukannya kurang sederhana:

1) Jika salah satu sudut segitiga ABC lebih dari 120° (katakanlah dalam A), maka minimum tercapai pada titik ini (katakanlah AB+AC).

Contoh:

```
>C=[-4,1];
>function d3(x,y):=d2(x,y)+sqrt((x-C[1])^2+(y-C[2])^2)
>plot3d("d3",xmin=-5,xmax=3,ymin=-4,ymax=4);
>insimg;
>fcontour("d3",xmin=-4,xmax=1,ymin=-2,ymax=2,hue=1,title="The minimum is on
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);
>insimg;
```

2) Tetapi jika semua sudut segitiga ABC kurang dari 120° , minimum berada pada titik F di bagian dalam segitiga, yang merupakan satu-satunya titik yang melihat sisi ABC dengan sudut yang sama (lalu masing-masing 120°):

```
>C=[-0.5,1];
>plot3d("d3",xmin=-2,xmax=2,ymin=-2,ymax=2):
>fcontour("d3",xmin=-2,xmax=2,ymin=-2,ymax=2,hue=1,title="The Fermat point");
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);
>insimg;
```

Merupakan kegiatan yang menarik untuk mewujudkan gambar di atas dengan perangkat lunak geometri; sebagai contoh, saya tahu soft tertulis di Java yang memiliki instruksi "garis kontur" ...

Semua ini di atas telah ditemukan oleh seorang hakim Prancis bernama Pierre de Fermat; dia menulis surat kepada para penggila lainnya seperti pendeta Marin Mersenne dan Blaise Pascal yang bekerja di bagian pajak penghasilan. Jadi titik unik F sehingga $FA + FB + FC$ minimal disebut titik Fermat segitiga. Tetapi tampaknya beberapa tahun sebelumnya, Torricelli Italia telah menemukan titik ini sebelum Fermat melakukannya! Pokoknya tradisinya adalah mencatat poin ini ...

Empat titik

Langkah selanjutnya adalah menambahkan titik D ke-4 dan mencoba meminimalkan $MA + MB + MC + MD$; katakanlah bahwa Anda adalah operator TV kabel dan ingin mencari di bidang mana Anda harus meletakkan antena sehingga Anda dapat memberi makan empat desa dan menggunakan kabel sesedikit mungkin!

```
>D=[1,1];
>function d4(x,y):=d3(x,y)+sqrt((x-D[1])^2+(y-D[2])^2)
>plot3d("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5):
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)';
>plot2d(P[1],P[2],points=1,add=1,color=12);
>insimg;
```

Masih ada minimum dan tidak ada yang dicapai pada simpul A, B, C atau D:

```
>function f(x):=d4(x[1],x[2])
>neldermin("f",[0.2,0.2])
```

[0.142858, 0.142857]

Tampaknya dalam kasus ini, koordinat titik optimal rasional atau mendekati rasional ...

Sekarang ABCD adalah bujur sangkar, kami berharap bahwa titik optimal adalah pusat ABCD:

```
>C=[-1,1];
>plot3d("d4",xmin=-1,xmax=1,ymin=-1,ymax=1):
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)';
>plot2d(P[1],P[2],add=1,color=12,points=1);
>insimg;
```

Contoh 7: Bola Dandelin dengan Povray

Anda dapat menjalankan demonstrasi ini, jika Anda memiliki Povray diinstal, dan pveninge.exe di jalur program.

Pertama kami menghitung jari-jari bola.

Jika Anda melihat gambar di bawah, Anda melihat bahwa kita membutuhkan dua lingkaran yang menyentuh dua garis yang membentuk kerucut, dan satu garis yang membentuk bidang yang memotong kerucut.

Kami menggunakan file geometry.e dari Euler untuk ini.

```
>load geometry;
```

Pertama, dua garis yang membentuk kerucut.

```
>g1 &= lineThrough([0,0],[1,a])
```

```
[- a, 1, 0]
```

```
>g2 &= lineThrough([0,0],[-1,a])
```

```
[- a, - 1, 0]
```

Lalu baris ketiga.

```
>g &= lineThrough([-1,0],[1,1])
```

```
[- 1, 2, 1]
```

Kita merencanakan semuanya sejauh ini.

```
>setPlotRange(-1,1,0,2);  
>color(black); plotLine(g(),"")  
>a:=2; color(blue); plotLine(g1(),""), plotLine(g2(),"");
```

Sekarang kita ambil titik umum pada sumbu y.

```
>P &= [0,u]
```

```
[0, u]
```

Hitung jarak ke g1.

```
>d1 &= distance(P,projectToLine(P,g1)); $d1
```

Hitung jarak ke g.

```
>d &= distance(P,projectToLine(P,g)); $d
```

Dan temukan pusat kedua lingkaran, di mana jaraknya sama.

```
>sol &= solve(d1^2=d^2,u); $sol
```

Ada dua solusi.

Kami mengevaluasi solusi simbolis, dan menemukan kedua pusat, dan kedua jarak.

```
>u := sol()
```

```
[0.333333, 1]
```

```
>dd := d()
```

```
[0.149071, 0.447214]
```

Plot lingkaran ke dalam gambar.

```
>color(red);
>plotCircle(circleWithCenter([0,u[1]],dd[1]), "");
>plotCircle(circleWithCenter([0,u[2]],dd[2]), "");
>insimg;
```

Plot dengan Povray

Selanjutnya kami merencanakan semuanya dengan Povray. Perhatikan bahwa Anda mengubah perintah apa pun dalam urutan perintah Povray berikut, dan menjalankan kembali semua perintah dengan Shift-Return.

Pertama kita memuat fungsi povray.

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

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

Kita mengatur adegan dengan tepat.

```
>povstart(zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```

Selanjutnya kita menulis dua bidang ke file Povray.

```
>writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
>writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
```

Dan kerucutnya, transparan.

```
>writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
```

Kami menghasilkan pesawat terbatas pada kerucut.

```
>gp=g();
>pc=povcone([0,0,0],0,[0,0,a],1,"");
>vp=[gp[1],0,gp[2]]; dp=gp[3];
>writeln(povplane(vp,dp,povlook(blue,0.5),pc));
```

Sekarang kami menghasilkan dua titik pada lingkaran, di mana bola menyentuh kerucut.

```
>function turnz(v) := return [-v[2],v[1],v[3]]
>P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);
>writeln(povpoint(P1,povlook(yellow)));
>P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);
>writeln(povpoint(P2,povlook(yellow))');
```

Kemudian kami menghasilkan dua titik di mana bola menyentuh bidang. Ini adalah fokus ellips.

```
>P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];
>writeln(povpoint(P3,povlook(yellow)));
>P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];
>writeln(povpoint(P4,povlook(yellow)));
```

Selanjutnya kita menghitung perpotongan P1P2 dengan bidang.

```
>t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);
>writeln(povpoint(P5,povlook(yellow)));
```

Kami menghubungkan titik dengan segmen garis.

```
>writeln(povsegment(P1,P2,povlook(yellow)));
>writeln(povsegment(P5,P3,povlook(yellow)));
>writeln(povsegment(P5,P4,povlook(yellow)));
```

Sekarang kami membuat pita abu-abu, di mana bola menyentuh kerucut.

```
>pcw=povcone([0,0,0],0,[0,0,a],1.01);
>pc1=povcylinder([0,0,P1[3]-defaultpointsiz/2],[0,0,P1[3]+defaultpointsiz
>writeln(povintersection([pcw,pc1],povlook(gray)));
>pc2=povcylinder([0,0,P2[3]-defaultpointsiz/2],[0,0,P2[3]+defaultpointsiz
>writeln(povintersection([pcw,pc2],povlook(gray)));
```

Mulai program Povray.

```
>povend();
```

Untuk mendapatkan Anaglyph ini, kita perlu memasukkan semuanya ke dalam fungsi scene. Fungsi ini akan digunakan dua kali nanti.

```
>function scene () ...  
  
    global a,u,dd,g,g1,defaultpointsize;  
    writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));  
    writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));  
    writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));  
    gp=g();  
    pc=povcone([0,0,0],0,[0,0,a],1,"");  
    vp=[gp[1],0,gp[2]]; dp=gp[3];  
    writeln(povplane(vp,dp,povlook(blue,0.5),pc));  
    P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);  
    writeln(povpoint(P1,povlook(yellow)));  
    P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);  
    writeln(povpoint(P2,povlook(yellow)));  
    P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];  
    writeln(povpoint(P3,povlook(yellow)));  
    P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];  
    writeln(povpoint(P4,povlook(yellow)));  
    t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);  
    writeln(povpoint(P5,povlook(yellow)));  
    writeln(povsegment(P1,P2,povlook(yellow)));  
    writeln(povsegment(P5,P3,povlook(yellow)));  
    writeln(povsegment(P5,P4,povlook(yellow)));  
    pcw=povcone([0,0,0],0,[0,0,a],1.01);  
    pc1=povcylinder([0,0,P1[3]-defaultpointsize/2],[0,0,P1[3]+defaultpointsize/2],  
    writeln(povintersection([pcw,pc1],povlook(gray)));  
    pc2=povcylinder([0,0,P2[3]-defaultpointsize/2],[0,0,P2[3]+defaultpointsize/2],  
    writeln(povintersection([pcw,pc2],povlook(gray)));  
endfunction
```

Anda membutuhkan kacamata merah / cyan untuk mengapresiasi efek berikut.

```
>povanaglyph("scene",zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```

Contoh 8: Geometri Bumi

Di notebook ini, kami ingin melakukan beberapa komputasi bola. Fungsi-fungsi tersebut terdapat dalam file "spherical.e" di folder contoh. Kita perlu memuat file itu dulu.

```
>load spherical.e
```

Spherical functions for Euler.

Untuk memasukkan posisi geografis, kami menggunakan vektor dengan dua koordinat dalam radian (utara dan timur, nilai negatif untuk selatan dan barat). Berikut koordinat Kampus FMIPA UNY.

```
>FMIPA=[rad(-7,-46.467),rad(110,23.05)]
```

[-0.13569, 1.92657]

Anda dapat mencetak posisi ini dengan sposprint (cetak posisi bola).

```
>sposprint(FMIPA) // posisi garis lintang dan garis bujur FMIPA UNY
```

S 7°46.467' E 110°23.050'

Mari kita tambahkan dua kota lagi, Solo dan Semarang.

```
>Solo=[rad(-7,-34.333),rad(110,49.683)]; Semarang=[rad(-6,-59.05),rad(110,24.533)];
```

S 7°34.333' E 110°49.683'

S 6°59.050' E 110°24.533'

Pertama kita menghitung vektor dari satu bola ke bola lainnya pada bola ideal. Vektor ini adalah [heading, distance] dalam radian. Untuk menghitung jarak di bumi, kita mengalikan dengan jari-jari bumi pada garis lintang 7°.

```
>br=svector(FMIPA,Solo); degprint(br[1]), br[2]*rearth(7°)->km // perkiraan
```

65°20'26.60''

53.8945384608

Ini adalah perkiraan yang bagus. Rutinitas berikut menggunakan perkiraan yang lebih baik. Pada jarak yang begitu dekat hasilnya hampir sama.

```
>esdist(FMIPA,Semarang)->" km", // perkiraan jarak FMIPA-Semarang
```

88.0114026318 km

Ada fungsi untuk heading, dengan mempertimbangkan bentuk bumi yang elips. Sekali lagi, kami mencetak dengan cara yang canggih.

```
>sdegprint(esdir(FMIPA,Solo))
```

65.34°

Sudut segitiga melebihi 180° pada bola.

```
>asum=sangle(Solo,FMIPA,Semarang)+sangle(FMIPA,Solo,Semarang)+sangle(FMIPA,
```

180°0'10.77''

Ini dapat digunakan untuk menghitung luas segitiga. Catatan: Untuk segitiga kecil, ini tidak akurat karena kesalahan pengurangan dalam asum-pi.

```
>(asum-pi)*rearth(48°)^2->" km^2", //perkiraan luas segitiga FMIPA-Solo-Sem
```

2116.02948749 km²

Ada fungsi untuk ini, yang menggunakan garis lintang rata-rata segitiga untuk menghitung jari-jari bumi, dan menangani kesalahan pembulatan untuk segitiga yang sangat kecil.

```
>esarea(Solo,FMIPA,Semarang)->" km^2", //perkiraan yang sama dengan fungsi
```

2123.64310526 km²

Kami juga dapat menambahkan vektor ke posisi. Vektor berisi heading dan jarak, keduanya dalam radian. Untuk mendapatkan vektor, kami menggunakan svector. Untuk menambahkan vektor ke posisi, kami menggunakan saddvector.

```
>v=svector(FMIPA,Solo); sposprint(saddvector(FMIPA,v)), sposprint(Solo),
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

Fungsi-fungsi ini mengasumsikan bola yang ideal. Hal yang sama di bumi.

```
>sposprint(esadd(FMIPA,esdir(FMIPA,Solo),esdist(FMIPA,Solo))), sposprint(So
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

Mari kita beralih ke contoh yang lebih besar, Tugu Jogja dan Monas Jakarta (menggunakan Google Earth untuk mencari koordinatnya).

```
>Tugu=[-7.7833°,110.3661°]; Monas=[-6.175°,106.811944°];  
>sposprint(Tugu), sposprint(Monas)
```

S 7°46.998' E 110°21.966'
S 6°10.500' E 106°48.717'

Menurut Google Earth, jaraknya 429,66 km. Kami mendapatkan perkiraan yang bagus.

```
>esdist(Tugu,Monas)->" km", // perkiraan jarak Tugu Jogja - Monas Jakarta
```

431.565659488 km

Judulnya sama dengan yang dihitung di Google Earth.

```
>degprint(esdir(Tugu,Monas))
```

294°17'2.85''

Namun, kita tidak lagi mendapatkan posisi target yang tepat, jika kita menambahkan heading dan jarak ke posisi semula. Hal ini terjadi, karena kita tidak menghitung fungsi invers secara tepat, tetapi mengambil perkiraan jari-jari bumi di sepanjang jalan.

```
>sposprint(esadd(Tugu,esdir(Tugu,Monas),esdist(Tugu,Monas)))
```

S 6°10.500' E 106°48.717'

Namun, kesalahannya tidak besar.

```
>sposprint(Monas),
```

S 6°10.500' E 106°48.717'

Tentunya kita tidak bisa berlayar dengan tujuan yang sama dari satu tujuan ke tujuan lainnya, jika kita ingin mengambil jalur terpendek. Bayangkan, Anda terbang NE mulai dari titik mana pun di bumi. Kemudian Anda akan berputar ke kutub utara. Lingkaran besar tidak mengikuti arah yang konstan!

Perhitungan berikut menunjukkan bahwa kami jauh dari tujuan yang benar, jika kami menggunakan tajuk yang sama selama perjalanan kami.

```
>dist=esdist(Tugu,Monas); hd=esdir(Tugu,Monas);
```

Sekarang kita tambahkan 10 kali sepersepuluh jaraknya, menggunakan heading ke Monas, kita sampai di Tugu.

```
>p=Tugu; loop 1 to 10; p=esadd(p,hd,dist/10); end;
```

Hasilnya masih jauh.

```
>sposprint(p), skmprint(esdist(p,Monas))
```

```
S 6°11.250' E 106°48.372'  
1.529km
```

Sebagai contoh lain, mari kita ambil dua titik di bumi pada ketinggian yang sama.

```
> P1=[30°,10°]; P2=[30°,50°];
```

Jalur terpendek dari P1 ke P2 bukanlah lingkaran dengan garis lintang 30° , tetapi jalur yang lebih pendek mulai 10° lebih jauh ke utara di P1.

```
>sdegprint(esdir(P1,P2))
```

```
79.69°
```

Tapi, jika kita mengikuti pembacaan kompas ini, kita akan berputar ke kutub utara! Jadi kita harus menyesuaikan arah tujuan kita di sepanjang jalan. Untuk tujuan kasar, kami menyesuaikannya pada 1/10 dari jarak total.

```
>p=P1; dist=esdist(P1,P2); ...  
> loop 1 to 10; dir=esdir(p,P2); sdegprint(dir), p=esadd(p,dir,dist/10); e
```

```
79.69°  
81.67°  
83.71°  
85.78°  
87.89°  
90.00°  
92.12°  
94.22°  
96.29°  
98.33°
```

Jaraknya tidak tepat, karena kita akan menambahkan sedikit kesalahan, jika kita mengikuti tajuk yang sama terlalu lama.

```
>skmprint(esdist(p,P2))
```

0.203km

Kami mendapatkan perkiraan yang baik, jika kami menyesuaikan heading setelah setiap 1/100 dari total jarak dari Tugu ke Monas.

```
>p=Tugu; dist=esdist(Tugu,Monas); ...
> loop 1 to 100; p=esadd(p,esdir(p,Monas),dist/100); end;
>skmprint(esdist(p,Monas))
```

0.000km

Untuk keperluan navigasi, kita bisa mendapatkan urutan posisi GPS di sepanjang lingkaran besar menuju Monas dengan fungsi navigasi.

```
>load spherical; v=navigate(Tugu,Monas,10); ...
> loop 1 to rows(v); sposprint(v[#]), end;
```

S 7°46.998' E 110°21.966'
S 7°37.422' E 110°0.573'
S 7°27.829' E 109°39.196'
S 7°18.219' E 109°17.834'
S 7°8.592' E 108°56.488'
S 6°58.948' E 108°35.157'
S 6°49.289' E 108°13.841'
S 6°39.614' E 107°52.539'
S 6°29.924' E 107°31.251'
S 6°20.219' E 107°9.977'
S 6°10.500' E 106°48.717'

Kami menulis sebuah fungsi, yang menggambarkan bumi, dua posisi, dan posisi di antaranya.

```
>function testplot ...
```

```
useglobal;
plotearth;
plotpos(Tugu,"Tugu Jogja"); plotpos(Monas,"Tugu Monas");
plotposline(v);
endfunction
```

Sekarang plot semuanya.

```
>plot3d("testplot",angle=25, height=6,>own,>user,zoom=4) :
```

Atau gunakan plot3d untuk mendapatkan tampilan anaglyphnya. Ini terlihat sangat bagus dengan kacamata merah / cyan.

```
>plot3d("testplot",angle=25,height=6,distance=5,own=1,anaglyph=1,zoom=4) :
```

Latihan

1. Gambarlah segi-n beraturan jika diketahui titik pusat O, n, dan jarak titik pusat ke titik-titik sudut segi-n tersebut (jari-jari lingkaran luar segi-n), r.

Petunjuk:

- Besar sudut pusat yang menghadap masing-masing sisi segi-n adalah $(360/n)$.
- Titik-titik sudut segi-n merupakan perpotongan lingkaran luar segi-n dan garis-garis yang melalui pusat dan saling membentuk sudut sebesar kelipatan $(360/n)$.
- Untuk n ganjil, pilih salah satu titik sudut adalah di atas.
- Untuk n genap, pilih 2 titik di kanan dan kiri lurus dengan titik pusat.
- Anda dapat menggambar segi-3, 4, 5, 6, 7, dst beraturan.

2. Gambarlah suatu parabola yang melalui 3 titik yang diketahui.

Petunjuk:

- Misalkan persamaan parabolanya $y = ax^2 + bx + c$.
- Substitusikan koordinat titik-titik yang diketahui ke persamaan tersebut.
- Selesaikan SPL yang terbentuk untuk mendapatkan nilai-nilai a, b, c.

3. Gambarlah suatu segi-4 yang diketahui keempat titik sudutnya, misalnya A, B, C, D.

- Tentukan apakah segi-4 tersebut merupakan segi-4 garis singgung

(sisinya-sisinya merupakan garis singgung lingkaran yang sama yakni lingkaran dalam segi-4 tersebut).

- Suatu segi-4 merupakan segi-4 garis singgung apabila keempat garis bagi sudutnya bertemu di satu titik.

- Jika segi-4 tersebut merupakan segi-4 garis singgung, gambar

lingkaran dalamnya.

- Tunjukkan bahwa syarat suatu segi-4 merupakan segi-4 garis

singgung apabila hasil kali panjang sisi-sisi yang berhadapan sama.

4. Gambarlah suatu ellips jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang jumlah jarak ke P dan ke Q selalu sama (konstan).

5. Gambarlah suatu hiperbola jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang selisih jarak ke P dan ke Q selalu sama (konstan).

Jawab:

1.

```
>o &:= [0,0]; z=circleWithCenter(o,5);
>color(1); setPlotRange(5); plotPoint(o); plotCircle(c);
>A=[-5,0]; plotPoint(A,"A");
>B=[5,0]; plotPoint(B,"B");
>plotSegment(A,B,"");
>z1=circleWithCenter(A,distance(A,o));
>z2=circleWithCenter(B,distance(B,o));
>f=circleCircleIntersections(z1,z);
>g=circleCircleIntersections(z,z2);
>h=circleCircleIntersections(z2,z);
>i=circleCircleIntersections(z,z1);
>r=lineThrough(f,h); s=lineThrough(g,i);
>setPlotRange(8); plotPoint(o); plotCircle(z); plotPoint(A,"A"); plotPoint(B,"B");
>color(4); plotCircle(z1); plotCircle(z2); plotPoint(f); plotPoint(g); plotPoint(h);
>color(5); plotLine(r); plotLine(s);
>color(6); plotSegment(A,f,""); plotSegment(A,i,""); plotSegment(f,g,"");
> plotSegment(g,B,""); plotSegment(B,h,""); plotSegment(h,i,"");
```

2.

```
>setPlotRange(5); D=[1,0]; E=[4,0]; F=[0,-4];
>plotPoint(D,"D"); plotPoint(E,"E"); plotPoint(F,"F");
>sol &= solve([a+b=-c,16*a+4*b=-c,c=-4],[a,b,c])
```

[[a = - 1, b = 5, c = - 4]]

```
>function y&=-x^2+5*x-4
```

$$- x^2 + 5 x - 4$$

```
>plot2d("-x^2+5*x-4", -5, 5, -5, 5) :
```

3.

```
>setPlotRange (-5,5,-5,5);
>A=[-2,2]; plotPoint(A,"A");
>B=[2,2]; plotPoint(B,"B");
>C=[2,-2]; plotPoint(C,"C");
>D=[-2,-2]; plotPoint(D,"D");
>plotSegment(A,B);
>plotSegment(B,C);
>plotSegment(C,D);
>plotSegment(D,A);
>plotSegment(A,C,"k1");
>plotSegment(B,D,"k2");
>k1=lineThrough(A,C);
>k2=lineThrough(B,D);
>p=lineIntersection(k1,k2);
>plotLine(k1); plotLine(k2);
>plotPoint(p, "P");
>r=norm(p-projectToLine(p, lineThrough(A,B)))
```

2

```
>plotCircle(circleWithCenter(p,r),"lingkaran dalam segi-4 ABCD") :
>AB=norm(A-B) // panjang sisi AB
```

4

```
>CD=norm(C-D) // panjang sisi CD
```

4

```
>AD=norm(A-D) // panjang sisi AD
```

4

```
>BC=norm(B-C) // panjang sisi BC
```

4

```
>AB.CD
```

16

```
>AD.BC
```

16

4.

```
>P=[-1,-1]; Q=[1,-1];
>function d1(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)
>Q=[1,-1]; function d2(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)+sqrt((x-Q[1])^2+(y-Q[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
>plot3d("d2",xmin=-2,xmax=2,ymin=-3,ymax=1):
```

5.

```
>P=[-1,-1]; Q=[1,-1];
>function d1(x,y):=sqrt((x-p[1])^2+(y-p[2])^2)
>Q=[1,-1]; function d2(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)+sqrt((x+Q[1])^2+(y+Q[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
>plot3d("d2",xmin=-2,xmax=2,ymin=-3,ymax=1):
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
>
```

BAB 6

KB Pekan 13-14 : Menggunakan EMT untuk Statistika

Nama : Dassy Chintya Nur Feika

NIM : 22305141023

Kelas : Matematika E 2022

EMT untuk Statistika

Dalam buku catatan ini, kami mendemonstrasikan plot statistik utama, pengujian, dan distribusi di Euler.

Mari kita mulai dengan beberapa statistik deskriptif. Ini bukan pengantar statistik. Jadi, Anda mungkin memerlukan beberapa latar belakang untuk memahami detailnya.

Asumsikan pengukuran berikut. Kami ingin menghitung nilai rata-rata dan standar deviasi yang diukur.

```
>M=[1000,1004,998,997,1002,1001,998,1004,998,997]; ...
>mean (M) , dev (M) ,
```

```
999.9
2.72641400622
```

Kita dapat memplot plot kotak-dan-kumis untuk data. Dalam kasus kami tidak ada outlier.

```
>boxplot (M) :
```

Kami menghitung probabilitas bahwa suatu nilai lebih besar dari 1005, dengan asumsi nilai terukur dan distribusi normal.

Semua fungsi untuk distribusi di Euler diakhiri dengan ...dis dan menghitung distribusi probabilitas kumulatif (CPF).

$$\text{normaldis}(x,m,d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

Kami mencetak hasilnya dalam % dengan akurasi 2 digit menggunakan fungsi cetak.

```
>print((1-normaldis(1005,mean(M),dev(M)))*100,2,unit=" %")
```

3.07 %

Untuk contoh berikutnya, kami mengasumsikan jumlah pria berikut dalam rentang ukuran yang diberikan.

```
>r=155.5:4:187.5; v=[22,71,136,169,139,71,32,8];
```

Berikut adalah plot distribusinya.

```
>plot2d(r,v,a=150,b=200,c=0,d=190,bar=1,style="/"):
```

Kita bisa memasukkan data mentah tersebut ke dalam sebuah tabel.

Tabel adalah metode untuk menyimpan data statistik. Tabel kita harus berisi tiga kolom: Awal jangkauan, akhir jangkauan, jumlah orang dalam jangkauan.

Tabel dapat dicetak dengan header. Kami menggunakan vektor string untuk mengatur header.

```
>T:=r[1:8]' | r[2:9]' | v'; writetable(T,labc=["from","to","count"])
```

from	to	count
155.5	159.5	22
159.5	163.5	71
163.5	167.5	136
167.5	171.5	169
171.5	175.5	139
175.5	179.5	71
179.5	183.5	32
183.5	187.5	8

Jika kita membutuhkan nilai rata-rata dan statistik lain dari ukuran, kita perlu menghitung titik tengah rentang. Kita dapat menggunakan dua kolom pertama dari tabel kita untuk ini. Sumbu " | " digunakan untuk memisahkan kolom, fungsi "writetable" digunakan untuk menulis tabel, dengan opsi "labc" adalah untuk menentukan header kolom.

```
> (T[,1]+T[,2])/2 // the midpoint of each interval
```

```
157.5
161.5
165.5
169.5
173.5
177.5
181.5
185.5
```

Tetapi lebih mudah, untuk melipat rentang dengan vektor [1/2.1/2].

```
>M=fold(r, [0.5, 0.5])
```

```
[157.5, 161.5, 165.5, 169.5, 173.5, 177.5, 181.5, 185.5]
```

Sekarang kita dapat menghitung mean dan deviasi sampel dengan frekuensi yang diberikan.

```
>{m, d}=meandev(M, v); m, d,
```

```
169.901234568
5.98912964449
```

Mari kita tambahkan distribusi normal dari nilai-nilai ke plot batang di atas. Rumus untuk distribusi normal dengan mean m dan standar deviasi d adalah:

$$y = \frac{1}{d\sqrt{2\pi}} e^{\frac{-(x-m)^2}{2d^2}}.$$

Karena nilainya antara 0 dan 1, untuk memplotnya pada bar plot harus dikalikan dengan 4 kali jumlah total data.

```
>plot2d("qnormal(x,m,d)*sum(v)*4", ...
> xmin=min(r), xmax=max(r), thickness=3, add=1):
```

Meja

Di direktori notebook ini Anda menemukan file dengan tabel. Data tersebut mewakili hasil survei. Berikut adalah empat baris pertama dari file tersebut. Data berasal dari buku online Jerman "Einführung in die Statistik mit R" oleh A. Handl.

```
>printfile("table.dat", 4);
```

Person	Sex	Age	Titanic	Evaluation	Tip	Problem
1	m	30	n	.	1.80	n
2	f	23	y	g	1.80	n
3	f	26	y	g	1.80	y

Tabel berisi 7 kolom angka atau token (string). Kami ingin membaca tabel dari file. Pertama, kami menggunakan terjemahan kami sendiri untuk token.

Untuk ini, kami mendefinisikan set token. Fungsi strtokens() mendapatkan vektor string token dari string yang diberikan.

```
>mf:=[ "m", "f" ]; yn:=[ "y", "n" ]; ev:=strtokens("g vg m b vb");
```

Sekarang kita membaca tabel dengan terjemahan ini.

Argumen tok2, tok4 dll. adalah terjemahan dari kolom tabel. Argumen ini tidak ada dalam daftar parameter readtable(), jadi Anda harus menyediakannya dengan ":=".

```
>{MT,hd}=readtable("table.dat",tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);  
>load over statistics;
```

Untuk mencetak, kita perlu menentukan set token yang sama. Kami mencetak empat baris pertama saja.

```
>writetable(MT[1:4],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

Person	Sex	Age	Titanic	Evaluation	Tip	Problem
1	m	30	n	.	1.8	n
2	f	23	y	g	1.8	n
3	f	26	y	g	1.8	y
4	m	33	n	.	2.8	n

Titik "." mewakili nilai-nilai, yang tidak tersedia.

Jika kita tidak ingin menentukan token untuk terjemahan terlebih dahulu, kita hanya perlu menentukan, kolom mana yang berisi token dan bukan angka.

```
>ctok=[2,4,5,7]; {MT,hd,tok}=readtable("table.dat",ctok=ctok);
```

Fungsi readtable() sekarang mengembalikan satu set token.

```
>tok
```

```
m  
n  
f  
y  
g  
vg
```

Tabel berisi entri dari file dengan token yang diterjemahkan ke angka.

String khusus NA="." ditafsirkan sebagai "Tidak Tersedia", dan mendapatkan NAN (bukan angka) dalam tabel. Terjemahan ini dapat diubah dengan parameter NA, dan NAval.

```
>MT [1]
```

```
[1, 1, 30, 2, NAN, 1.8, 2]
```

Berikut isi tabel dengan angka yang belum diterjemahkan.

```
>writetable(MT,wc=5)
```

1	1	30	2	.	1.8	2
2	3	23	4	5	1.8	2
3	3	26	4	5	1.8	4
4	1	33	2	.	2.8	2
5	1	37	2	.	1.8	2
6	1	28	4	5	2.8	4
7	3	31	4	6	2.8	2
8	1	23	2	.	0.8	2
9	3	24	4	6	1.8	4
10	1	26	2	.	1.8	2
11	3	23	4	6	1.8	4
12	1	32	4	5	1.8	2
13	1	29	4	6	1.8	4
14	3	25	4	5	1.8	4
15	3	31	4	5	0.8	2
16	1	26	4	5	2.8	2

17	1	37	2	.	3.8	2
18	1	38	4	5	.	2
19	3	29	2	.	3.8	2
20	3	28	4	6	1.8	2
21	3	28	4	1	2.8	4
22	3	28	4	6	1.8	4
23	3	38	4	5	2.8	2
24	3	27	4	1	1.8	4
25	1	27	2	.	2.8	4

Untuk kenyamanan, Anda dapat memasukkan output readtable() ke dalam daftar.

```
>Table={{readtable("table.dat",ctok=ctok)}};
```

Menggunakan kolom token yang sama dan token yang dibaca dari file, kita dapat mencetak tabel. Kita dapat menentukan ctok, tok, dll. Atau menggunakan daftar Tabel.

```
>writetable(Table,ctok=ctok,wc=5);
```

Person	Sex	Age	Titanic	Evaluation	Tip	Problem
1	m	30	n	.	1.8	n
2	f	23	y	g	1.8	n
3	f	26	y	g	1.8	y
4	m	33	n	.	2.8	n
5	m	37	n	.	1.8	n
6	m	28	y	g	2.8	y
7	f	31	y	vg	2.8	n
8	m	23	n	.	0.8	n
9	f	24	y	vg	1.8	y
10	m	26	n	.	1.8	n
11	f	23	y	vg	1.8	y
12	m	32	y	g	1.8	n
13	m	29	y	vg	1.8	y
14	f	25	y	g	1.8	y
15	f	31	y	g	0.8	n
16	m	26	y	g	2.8	n
17	m	37	n	.	3.8	n
18	m	38	y	g	.	n
19	f	29	n	.	3.8	n
20	f	28	y	vg	1.8	n
21	f	28	y	m	2.8	y
22	f	28	y	vg	1.8	y
23	f	38	y	g	2.8	n
24	f	27	y	m	1.8	y
25	m	27	n	.	2.8	y

Fungsi `tablecol()` mengembalikan nilai kolom tabel, melewatkannya setiap baris dengan nilai `NAN("." dalam file)`, dan indeks kolom, yang berisi nilai-nilai ini.

```
>{c,i}=tablecol(MT,[5,6]);
```

Kita dapat menggunakan ini untuk mengekstrak kolom dari tabel untuk tabel baru.

```
>j=[1,5,6]; writetable(MT[i,j],labc=hd[j],ctok=[2],tok=tok)
```

Person	Evaluation	Tip
2	g	1.8
3	g	1.8
6	g	2.8
7	vg	2.8
9	vg	1.8
11	vg	1.8
12	g	1.8
13	vg	1.8
14	g	1.8
15	g	0.8
16	g	2.8
20	vg	1.8
21	m	2.8
22	vg	1.8
23	g	2.8
24	m	1.8

Tentu saja, kita perlu mengekstrak tabel itu sendiri dari daftar Tabel dalam kasus ini.

```
>MT=Table[1];
```

Tentu saja, kita juga dapat menggunakan untuk menentukan nilai rata-rata kolom atau nilai statistik lainnya.

```
>mean(tablecol(MT,6))
```

2.175

Fungsi `getstatistics()` mengembalikan elemen dalam vektor, dan jumlahnya. Kami menerapkannya pada nilai "m" dan "f" di kolom kedua tabel kami.

```
>{xu,count}=getstatistics(tablecol(MT,2)); xu, count,
```

```
[1, 3]  
[12, 13]
```

Kami dapat mencetak hasilnya dalam tabel baru.

```
>writetable(count',labr=tok[xu])
```

```
m      12  
f      13
```

Fungsi selectable() mengembalikan tabel baru dengan nilai dalam satu kolom yang dipilih dari vektor indeks. Pertama kita mencari indeks dari dua nilai kita di tabel token.

```
>v:=indexof(tok, ["g", "vg"])
```

```
[5, 6]
```

Sekarang kita dapat memilih baris tabel, yang memiliki salah satu nilai dalam v di baris ke-5.

```
>MT1:=MT [selectrows(MT, 5, v)]; i:=sortedrows(MT1, 5);
```

Sekarang kita dapat mencetak tabel, dengan nilai yang diekstrak dan diurutkan di kolom ke-5.

```
>writetable(MT1[i], labc=hd, ctok=ctok, tok=tok, wc=7);
```

Person	Sex	Age	Titanic	Evaluation	Tip	Problem
2	f	23	y	g	1.8	n
3	f	26	y	g	1.8	y
6	m	28	y	g	2.8	y
18	m	38	y	g	.	n
16	m	26	y	g	2.8	n
15	f	31	y	g	0.8	n
12	m	32	y	g	1.8	n
23	f	38	y	g	2.8	n
14	f	25	y	g	1.8	y
9	f	24	y	vg	1.8	y

7	f	31	y	vg	2.8	n
20	f	28	y	vg	1.8	n
22	f	28	y	vg	1.8	y
13	m	29	y	vg	1.8	y
11	f	23	y	vg	1.8	y

Untuk statistik berikutnya, kami ingin menghubungkan dua kolom tabel. Jadi kami mengekstrak kolom 2 dan 4 dan mengurutkan tabel.

```
>i=sortedrows(MT,[2,4]); ...
> writetable(tablecol(MT[i],[2,4])',ctok=[1,2],tok=tok)
```

m	n
m	n
m	n
m	n
m	n
m	n
m	n
m	y
m	y
m	y
m	y
m	y
f	n
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y
f	y

Dengan getstatistics(), kita juga bisa menghubungkan hitungan dalam dua kolom tabel satu sama lain.

```
>MT24=tablecol(MT,[2,4]); ...
>{xu1,xu2,count}=getstatistics(MT24[1],MT24[2]); ...
>writetable(count,labr=tok[xu1],labc=tok[xu2])
```

	n	y
m	7	5
f	1	12

Tabel dapat ditulis ke file.

```
>filename="test.dat"; ...
>writetable(count,labr=tok[xu1],labc=tok[xu2],file=filename);
```

Kemudian kita bisa membaca tabel dari file tersebut.

```
>{MT2,hd,tok2,hdr}=readtable(filename,>clabs,>rlabs); ...
>writetable(MT2,labr=hdr,labc=hd)
```

	n	y
m	7	5
f	1	12

Dan hapus file tersebut.

```
>fileremove(filename);
```

Distribusi

Dengan plot2d, terdapat metode yang sangat mudah untuk memplot sebaran data eksperimen.

```
>p=normal(1,1000); //1000 random normal-distributed sample p
>plot2d(p,distribution=20,style="\\""); // plot the random sample p
>plot2d("qnormal(x,0,1)",add=1); // add the standard normal distribution pl
```

Harap perhatikan perbedaan antara plot batang (sampel) dan kurva normal (distribusi nyata). Masukkan kembali tiga perintah untuk melihat hasil pengambilan sampel lainnya.

Berikut adalah perbandingan 10 simulasi dari 1000 nilai terdistribusi normal menggunakan apa yang disebut plot kotak. Plot ini menunjukkan median, kuartil 25% dan 75%, nilai minimal dan maksimal, dan outlier.

```
>p=normal(10,1000); boxplot(p):
```

Untuk membangkitkan bilangan bulat acak, Euler memiliki intrarandom. Mari kita simulasi lemparan dadu dan plot distribusinya.

Kita menggunakan fungsi getmultiplicities(v,x), yang menghitung seberapa sering elemen v muncul di x. Kemudian kita memplot hasilnya menggunakan columnplot().

```
>k=intrandom(1,6000,6); ...
>columnspplot(getmultiplicities(1:6,k)); ...
>ygrid(1000,color=red):
```

Sementara intrandom(n,m,k) mengembalikan bilangan bulat yang terdistribusi secara seragam dari 1 ke k, dimungkinkan untuk menggunakan distribusi bilangan bulat lain yang diberikan dengan randpint().

Dalam contoh berikut, probabilitas untuk 1,2,3 masing-masing adalah 0,4,0,1,0,5.

```
>randpint(1,1000,[0.4,0.1,0.5]); getmultiplicities(1:3,%)
```

```
[378, 102, 520]
```

Euler dapat menghasilkan nilai acak dari lebih banyak distribusi. Coba lihat referensinya. Misalnya, kami mencoba distribusi eksponensial. Variabel acak kontinu X dikatakan memiliki distribusi eksponensial, jika PDF-nya diberikan oleh

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0,$$

dengan parameter

$$\lambda = \frac{1}{\mu}, \quad \mu \text{ adalah rata-rata, dan dilambangkan dengan } X \sim \text{Eksponensial}(\lambda).$$

```
>plot2d(randexponential(1,1000,2),>distribution):
```

Untuk banyak distribusi, Euler dapat menghitung fungsi distribusi dan inversnya.

```
>plot2d("normaldis",-4,4):
```

Berikut ini adalah salah satu cara untuk memplot kuantil.

```
>plot2d("qnormal(x,1,1.5)",-4,6); ...  
>plot2d("qnormal(x,1,1.5)",a=2,b=5,>add,>filled):
```

$$\text{normaldis}(x,m,d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

Probabilitas untuk berada di area hijau adalah sebagai berikut.

```
>normaldis(5,1,1.5)-normaldis(2,1,1.5)
```

0.248662156979

Ini dapat dihitung secara numerik dengan integral berikut.

$$\int_2^5 \frac{1}{1.5\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-1}{1.5})^2} dx.$$

```
>gauss("qnormal(x,1,1.5)",2,5)
```

0.248662156979

Mari kita bandingkan distribusi binomial dengan distribusi normal mean dan deviasi yang sama. Fungsi invbindis() memecahkan interpolasi linier antara nilai bilangan bulat.

```
>invbindis(0.95,1000,0.5), invnormaldis(0.95,500,0.5*sqrt(1000))
```

525.516721219
526.007419394

Fungsi qdis() adalah kepadatan distribusi chi-kuadrat. Seperti biasa, Euler memetakan vektor ke fungsi ini. Jadi kita mendapatkan plot dari semua distribusi chi-kuadrat dengan derajat 5 sampai 30 dengan mudah dengan cara berikut.

```
>plot2d("qchidis(x,(5:5:50)'),0,50):
```

Euler memiliki fungsi yang akurat untuk mengevaluasi distribusi. Mari kita periksa chidis() dengan integral.

Penamaan mencoba untuk konsisten. Misalnya.,

- distribusi chi-kuadrat adalah chidis(),
- fungsi kebalikannya adalah invchidis(),
- densitasnya adalah qchidis().

Pelengkap distribusi (ekor atas) adalah chicdis().

```
>chidis(1.5,2), integrate("qchidis(x,2)",0,1.5)
```

```
0.527633447259  
0.527633447259
```

Distribusi Diskrit

Untuk menentukan distribusi diskrit Anda sendiri, Anda dapat menggunakan metode berikut. Pertama kita mengatur fungsi distribusi.

```
>wd = 0 | ((1:6)+[-0.01,0.01,0,0,0,0])/6
```

```
[0, 0.165, 0.335, 0.5, 0.666667, 0.833333, 1]
```

Artinya dengan probabilitas wd[i+1]-wd[i] kita menghasilkan nilai acak i.

Ini hampir merupakan distribusi yang seragam. Mari kita tentukan generator angka acak untuk ini. Fungsi find(v,x) menemukan nilai x dalam vektor v. Fungsi ini juga berlaku untuk vektor x.

```
>function wrongdice (n,m) := find(wd,random(n,m))
```

Kesalahannya sangat halus sehingga kami melihatnya hanya dengan iterasi yang sangat banyak.

```
>columnsplot(getmultiplicities(1:6,wrongdice(1,1000000)):
```

Berikut adalah fungsi sederhana untuk memeriksa distribusi seragam dari nilai 1...K dalam v. Kami menerima hasilnya, jika untuk semua frekuensi

$$\left| f_i - \frac{1}{K} \right| < \frac{\delta}{\sqrt{n}}.$$

```
>function checkrandom (v, delta=1) ...  
  
    K=max(v); n=cols(v);  
    fr=getfrequencies(v,1:K);  
    return max(fr/n-1/K)<delta/sqrt(n);  
endfunction
```

Memang fungsi menolak distribusi seragam.

```
>checkrandom(wrongdice(1,1000000))
```

0

Dan itu menerima generator acak bawaan.

```
>checkrandom(intrandom(1,1000000,6))
```

1

Kita dapat menghitung distribusi binomial. Pertama ada binomials() , yang mengembalikan probabilitas i atau kurang hit dari n percobaan.

```
>bindis(410,1000,0.4)
```

0.751401349654

Fungsi Beta terbalik digunakan untuk menghitung interval kepercayaan Clopper-Pearson untuk parameter p. Level default adalah alfa.

Arti interval ini adalah jika p berada di luar interval, hasil pengamatan 410 dalam 1000 jarang terjadi.

```
>clopperpearson(410,1000)
```

[0.37932, 0.441212]

Perintah berikut adalah cara langsung untuk mendapatkan hasil di atas. Tapi untuk n besar, penjumlahan langsungnya tidak akurat dan lambat.

```
>p=0.4; i=0:410; n=1000; sum(bin(n,i)*p^i*(1-p)^(n-i))
```

0.751401349655

Omong-omong, `invbinsum()` menghitung kebalikan dari `binomials()`.

```
>invbindis(0.75,1000,0.4)
```

409.932733047

Di Bridge, kami mengasumsikan 5 kartu beredar (dari 52) dengan dua tangan (26 kartu). Mari kita hitung probabilitas distribusi yang lebih buruk dari 3:2 (mis. 0:5, 1:4, 4:1, atau 5:0).

```
>2*hypergeomsum(1,5,13,26)
```

0.321739130435

Ada juga simulasi distribusi multinomial.

```
>randmultinomial(10,1000,[0.4,0.1,0.5])
```

381	100	519
376	91	533
417	80	503
440	94	466
406	112	482
408	94	498
395	107	498
399	96	505
428	87	485
400	99	501

Merencanakan Data

Untuk memplot data, kami mencoba hasil pemilu Jerman sejak 1990, yang diukur dalam jumlah kursi.

```
>BW := [ ...  
>1990,662,319,239,79,8,17; ...  
>1994,672,294,252,47,49,30; ...  
>1998,669,245,298,43,47,36; ...  
>2002,603,248,251,47,55,2; ...  
>2005,614,226,222,61,51,54; ...  
>2009,622,239,146,93,68,76; ...  
>2013,631,311,193,0,63,64];
```

Untuk partai, kami menggunakan rangkaian nama.

```
>P := ["CDU/CSU", "SPD", "FDP", "Gr", "Li"];
```

Mari kita cetak persentasenya dengan baik.

Pertama, kami mengekstrak kolom yang diperlukan. Kolom 3 sampai 7 adalah kursi masing-masing partai, dan kolom 2 adalah jumlah kursi. kolom adalah tahun pemilihan.

```
>BT := BW[, 3:7]; BT := BT / sum(BT); YT := BW[, 1]';
```

Kemudian kami mencetak statistik dalam bentuk tabel. Kami menggunakan nama sebagai tajuk kolom, dan tahun sebagai tajuk untuk baris. Lebar default untuk kolom adalah wc=10, tetapi kami lebih memilih hasil yang lebih padat. Kolom akan diperluas untuk label kolom, jika perlu.

```
>writetable(BT * 100, wc=6, dc=0, >fixed, labc=P, labr=YT)
```

	CDU/CSU	SPD	FDP	Gr	Li
1990	48	36	12	1	3
1994	44	38	7	7	4
1998	37	45	6	7	5
2002	41	42	8	9	0
2005	37	36	10	8	9
2009	38	23	15	11	12
2013	49	31	0	10	10

Perkalian matriks berikut mengekstrak jumlah persentase dari dua partai besar yang menunjukkan bahwa partai-partai kecil telah mendapatkan rekaman di parlemen hingga tahun 2009.

```
>BT1 := (BT[, 1:2])' * 100
```

```
[84.29, 81.25, 81.1659, 82.7529, 72.9642, 61.8971, 79.8732]
```

Ada juga plot statistik sederhana. Kami menggunakananya untuk menampilkan garis dan titik secara bersamaan. Alternatifnya adalah memanggil plot2d dua kali dengan >add.

```
>statplot(YT, BT1, "b"):
```

Tentukan beberapa warna untuk masing-masing pihak.

```
>CP:=[rgb(0.5,0.5,0.5),red,yellow,green,rgb(0.8,0,0)];
```

Sekarang kita bisa memplot hasil pemilu 2009 dan mengubahnya menjadi satu plot menggunakan gambar. Kita dapat menambahkan vektor kolom ke setiap plot.

```
>figure(2,1); ...
>figure(1); columnsplot(BW[6,3:7],P,color=CP); ...
>figure(2); columnsplot(BW[6,3:7]-BW[5,3:7],P,color=CP); ...
>figure(0):
```

Plot data menggabungkan deretan data statistik dalam satu plot.

```
>J:=BW[,1]'; DP:=BW[,3:7]'; ...
>dataplot(YT,BT',color=CP); ...
>labelbox(P,colors=CP,styles="[]",>points,w=0.2,x=0.3,y=0.4):
```

Plot kolom 3D menunjukkan deretan data statistik dalam bentuk kolom. Kami menyediakan label untuk baris dan kolom. sudut adalah sudut pandang.

```
>columnsplot3d(BT,scols=P,srows=YT, ...
> angle=30°,ccols=CP):
```

Representasi lainnya adalah plot mozaik. Perhatikan bahwa kolom plot mewakili kolom matriks di sini. Karena panjang label CDU/CSU, kami mengambil jendela yang lebih kecil dari biasanya.

```
>shrinkwindow(>smaller); ...
>mosaicplot(BT',srows=YT,scols=P,color=CP,style="#"); ...
>shrinkwindow():
```

Kita juga bisa membuat diagram lingkaran. Karena hitam dan kuning membentuk koalisi, kami menyusun ulang elemennya.

```
>i=[1,3,5,4,2]; piechart(BW[6,3:7][i],color=CP[i],lab=P[i]):
```

Ini jenis plot lainnya.

```
>starplot(normal(1,10)+4,lab=1:10,>rays) :
```

Beberapa plot di plot2d bagus untuk statika. Berikut adalah plot impuls dari data acak, terdistribusi secara seragam di [0,1].

```
>plot2d(makeimpulse(1:10,random(1,10)),>bar) :
```

Tetapi untuk data yang terdistribusi secara eksponensial, kita mungkin memerlukan plot logaritmik.

```
>logimpulseplot(1:10,-log(random(1,10))*10) :
```

Fungsi columnplot() lebih mudah digunakan, karena hanya membutuhkan vektor nilai. Selain itu, ia dapat mengatur labelnya ke apa pun yang kita inginkan, kita telah mendemonstrasikannya di tutorial ini.

Ini adalah aplikasi lain, di mana kami menghitung karakter dalam sebuah kalimat dan memplot statistik.

```
>v=strtochar("the quick brown fox jumps over the lazy dog"); ...
>w=ascii("a"):ascii("z"); x=getmultiplicities(w,v); ...
>cw=[]; for k=w; cw=cw|char(k); end; ...
>columnsplot(x,lab=cw,width=0.05) :
```

Dimungkinkan juga untuk mengatur sumbu secara manual.

```
>n=10; p=0.4; i=0:n; x=bin(n,i)*p^i*(1-p)^(n-i); ...
>columnsplot(x,lab=i,width=0.05,<frame,<grid); ...
>yaxis(0,0:0.1:1,style="->",>left); xaxis(0,style="."); ...
>label("p",0,0.25), label("i",11,0); ...
>textbox(["Binomial distribution","with p=0.4"]):
```

Berikut ini adalah cara memplot frekuensi bilangan dalam vektor.

Kami membuat vektor bilangan acak bilangan bulat 1 hingga 6.

```
>v:=intrandom(1,10,10)
```

```
[8, 5, 8, 8, 6, 8, 8, 3, 5, 5]
```

Kemudian ekstrak angka unik di v.

```
>vu:=unique(v)
```

```
[3, 5, 6, 8]
```

Dan plot frekuensi dalam plot kolom.

```
>columnsplot(getmultiplicities(vu,v),lab=vu,style="/"):
```

Kami ingin menunjukkan fungsi untuk distribusi nilai empiris.

```
>x=normal(1,20);
```

Fungsi empdist(x,vs) membutuhkan array nilai yang diurutkan. Jadi kita harus mengurutkan x sebelum kita dapat menggunakannya.

```
>xs=sort(x);
```

Kemudian kami memplot distribusi empiris dan beberapa batang kepadatan menjadi satu plot. Alih-alih plot batang untuk distribusi, kali ini kami menggunakan plot gigi gergaji.

```
>figure(2,1); ...
>figure(1); plot2d("empdist",-4,4;xs); ...
>figure(2); plot2d(histo(x,v=-4:0.2:4,<bar)); ...
>figure(0):
```

Plot pencar mudah dilakukan di Euler dengan plot titik biasa. Grafik berikut menunjukkan bahwa X dan X+Y jelas berkorelasi positif.

```
>x=normal(1,100); plot2d(x,x+rotright(x),>points,style=".."):
```

Seringkali, kami ingin membandingkan dua sampel dari distribusi yang berbeda. Ini dapat dilakukan dengan plot kuantil-kuantil.

Untuk pengujian, kami mencoba distribusi student-t dan distribusi eksponensial.

```
>x=randt(1,1000,5); y=randnormal(1,1000,mean(x),dev(x)); ...
>plot2d("x",r=6,style="--",yl="normal",xl="student-t",>vertical); ...
>plot2d(sort(x),sort(y),>points,color=red,style="x",>add):
```

Plot jelas menunjukkan bahwa nilai terdistribusi normal cenderung lebih kecil di ujung ekstrim.

Jika kita memiliki dua distribusi dengan ukuran berbeda, kita dapat memperluas yang lebih kecil atau mengecilkan yang lebih besar. Fungsi berikut ini baik untuk keduanya. Dibutuhkan nilai median dengan persentase antara 0 dan 1.

```
>function medianexpand (x,n) := median(x,p=linspace(0,1,n-1));
```

Mari kita bandingkan dua distribusi yang sama.

```
>x=random(1000); y=random(400); ...
>plot2d("x",0,1,style="--"); ...
>plot2d(sort(medianexpand(x,400)),sort(y),>points,color=red,style="x",>add)
```

Regresi dan Korelasi

Regresi linier dapat dilakukan dengan fungsi polyfit() atau berbagai fungsi fit.

Sebagai permulaan, kami menemukan garis regresi untuk data univariat dengan polyfit(x,y,1).

```
>x=1:10; y=[2,3,1,5,6,3,7,8,9,8]; writetable(x' | y',labc=["x","y"])
```

x	y
1	2
2	3
3	1
4	5
5	6
6	3
7	7
8	8
9	9
10	8

Kami ingin membandingkan kecocokan yang tidak berbobot dan berbobot. Pertama koefisien fit linier.

```
>p=polyfit(x,y,1)
```

```
[0.733333, 0.812121]
```

Sekarang koefisien dengan bobot yang menekankan nilai terakhir.

```
>w &= "exp(-(x-10)^2/10)"; pw=polyfit(x,y,1,w=w(x))
```

```
[4.71566, 0.38319]
```

Kami memasukkan semuanya ke dalam satu plot untuk poin dan garis regresi, dan untuk bobot yang digunakan.

```
>figure(2,1); ...
>figure(1); statplot(x,y,"b",xl="Regression"); ...
> plot2d("evalpoly(x,p)",>add,color=blue,style="--"); ...
> plot2d("evalpoly(x,pw)",5,10,>add,color=red,style="--"); ...
>figure(2); plot2d(w,1,10,>filled,style="/",fillcolor=red,xl=w); ...
>figure(0):
```

Sebagai contoh lain kita membaca survei siswa, umur mereka, umur orang tua mereka dan jumlah saudara dari sebuah file.

Tabel ini berisi "m" dan "f" di kolom kedua. Kami menggunakan variabel tok2 untuk menyetel terjemahan yang tepat alih-alih membiarkan readtable() mengumpulkan terjemahan.

```
>{MS,hd}:=readtable("table1.dat",tok2:=[ "m", "f" ]); ...
>writetable(MS,labc=hd,tok2:=[ "m", "f" ]);
```

Person	Sex	Age	Mother	Father	Siblings
1	m	29	58	61	1
2	f	26	53	54	2
3	m	24	49	55	1
4	f	25	56	63	3
5	f	25	49	53	0
6	f	23	55	55	2
7	m	23	48	54	2
8	m	27	56	58	1
9	m	25	57	59	1

10	m	24	50	54	1
11	f	26	61	65	1
12	m	24	50	52	1
13	m	29	54	56	1
14	m	28	48	51	2
15	f	23	52	52	1
16	m	24	45	57	1
17	f	24	59	63	0
18	f	23	52	55	1
19	m	24	54	61	2
20	f	23	54	55	1

Bagaimana usia bergantung satu sama lain? Kesan pertama berasal dari sebar berpasangan.

```
>scatterplots(tablecol(MS, 3:5), hd[3:5]):
```

Jelas terlihat bahwa usia ayah dan ibu saling bergantung. Mari kita tentukan dan plot garis regresi.

```
>cs:=MS[, 4:5]'; ps:=polyfit(cs[1], cs[2], 1)
```

```
[17.3789, 0.740964]
```

Ini jelas model yang salah. Garis regresi adalah $s=17+0,74t$, di mana t adalah umur ibu dan s adalah umur ayah. Perbedaan usia mungkin sedikit bergantung pada usia, tetapi tidak terlalu banyak.

Sebaliknya, kami mencurigai fungsi seperti $s=a+t$. Maka a adalah rata-rata dari s-t. Ini adalah perbedaan usia rata-rata antara ayah dan ibu.

```
>da:=mean(cs[2]-cs[1])
```

```
3.65
```

Mari kita plot ini menjadi satu plot pencar.

```
>plot2d(cs[1], cs[2], >points); ...
>plot2d("evalpoly(x,ps)", color=red, style=".",>add); ...
>plot2d("x+da", color=blue,>add):
```

Berikut adalah plot kotak dari dua zaman. Ini hanya menunjukkan, bahwa usianya berbeda.

```
>boxplot(cs, ["mothers", "fathers"]):
```

Menariknya, perbedaan median tidak sebesar perbedaan rata-rata.

```
>median(cs[2])-median(cs[1])
```

1.5

Koefisien korelasi menunjukkan korelasi positif.

```
>correl(cs[1],cs[2])
```

0.7588307236

Korelasi peringkat adalah ukuran untuk urutan yang sama di kedua vektor. Ini juga cukup positif.

```
>rankcorrel(cs[1],cs[2])
```

0.758925292358

Membuat Fungsi baru

Tentu saja, bahasa EMT dapat digunakan untuk memprogram fungsi-fungsi baru. Misalnya, kita mendefinisikan fungsi skewness.

$$\text{sk}(x) = \frac{\sqrt{n} \sum_i (x_i - m)^3}{(\sum_i (x_i - m)^2)^{3/2}}$$

di mana m adalah rata-rata dari x.

```
>function skew (x:vector) ...
```

```
m=mean(x);  
return sqrt(cols(x))*sum((x-m)^3)/(sum((x-m)^2))^(3/2);  
endfunction
```

Seperti yang Anda lihat, kita dapat dengan mudah menggunakan bahasa matriks untuk mendapatkan implementasi yang sangat singkat dan efisien. Mari kita coba fungsi ini.

```
>data=normal(20); skew(normal(10))
```

```
-0.198710316203
```

Ini adalah fungsi lain, yang disebut koefisien kemiringan Pearson.

```
>function skew1 (x) := 3*(mean(x)-median(x))/dev(x)
>skew1(data)
```

```
-0.0801873249135
```

Simulasi Monte Carlo

Euler dapat digunakan untuk mensimulasikan kejadian acak. Kita telah melihat contoh sederhana di atas. Ini satu lagi, yang mensimulasikan 1000 kali 3 lemparan dadu, dan meminta distribusi jumlahnya.

```
>ds:=sum(intrandom(1000,3,6))'; fs=getmultiplicities(3:18,ds)
```

```
[5, 17, 35, 44, 75, 97, 114, 116, 143, 116, 104, 53, 40,
22, 13, 6]
```

Kita bisa merencanakan ini sekarang.

```
>columnsplot(fs,lab=3:18):
```

Untuk menentukan distribusi yang diharapkan tidak begitu mudah. Kami menggunakan rekursi lanjutan untuk ini.

Fungsi berikut menghitung banyaknya cara bilangan k dapat dinyatakan sebagai jumlah dari n bilangan dalam rentang 1 sampai m. Ini bekerja secara rekursif dengan cara yang jelas.

```
>function map countways (k; n, m) ...
if n==1 then return k>=1 && k<=m
else
  sum=0;
  loop 1 to m; sum=sum+countways(k-#,n-1,m); end;
  return sum;
end;
endfunction
```

Inilah hasil lemparan dadu sebanyak tiga kali.

```
>cw=countways(3:18,3,6)
```

```
[1, 3, 6, 10, 15, 21, 25, 27, 27, 25, 21, 15, 10, 6, 3,  
1]
```

Kami menambahkan nilai yang diharapkan ke plot.

```
>plot2d(cw/6^3*1000,>add); plot2d(cw/6^3*1000,>points,>add):
```

Untuk simulasi lain, penyimpangan nilai rata-rata n 0-1-variabel acak terdistribusi normal adalah $1/\sqrt{n}$.

```
>longformat; 1/sqrt(10)
```

```
0.316227766017
```

Mari kita periksa ini dengan simulasi. Kami menghasilkan 10.000 kali 10 vektor acak.

```
>M=normal(10000,10); dev(mean(M)')
```

```
0.319493614817
```

```
>plot2d(mean(M)',>distribution):
```

Median dari 10 bilangan acak terdistribusi 0-1-normal memiliki deviasi yang lebih besar.

```
>dev(median(M)')
```

```
0.374460271535
```

Karena kita dapat dengan mudah membuat jalan acak, kita dapat mensimulasikan proses Wiener. Kami mengambil 1000 langkah dari 1000 proses. Kami kemudian memplot standar deviasi dan rata-rata langkah ke-n dari proses ini bersama dengan nilai yang diharapkan dalam warna merah.

```
>n=1000; m=1000; M=cumsum(normal(n,m)/sqrt(m)); ...
>t=(1:n)/n; figure(2,1); ...
>figure(1); plot2d(t,mean(M)'); plot2d(t,0,color=red,>add); ...
>figure(2); plot2d(t,dev(M)'); plot2d(t,sqrt(t),color=red,>add); ...
>figure(0):
```

Tes

Tes adalah alat penting dalam statistik. Di Euler, banyak tes yang diterapkan. Semua tes ini mengembalikan kesalahan yang kami terima jika kami menolak hipotesis nol.

Sebagai contoh, kami menguji lemparan dadu untuk distribusi seragam. Pada 600 lemparan, kami mendapat nilai berikut, yang kami masukkan ke uji chi-square.

```
>chitest([90,103,114,101,103,89],dup(100,6)')
```

0.498830517952

Tes chi-kuadrat juga memiliki mode yang menggunakan simulasi Monte Carlo untuk menguji statistik. Hasilnya harus hampir sama. Parameter `>p` menginterpretasikan vektor-y sebagai vektor probabilitas.

```
>chitest([90,103,114,101,103,89],dup(1/6,6)',>p,>montecarlo)
```

0.526

Kesalahan ini terlalu besar. Jadi kita tidak bisa menolak pemerataan distribusi. Ini tidak membuktikan bahwa dadu kami adil. Tapi kita tidak bisa menolak hipotesis kita.

Selanjutnya kami menghasilkan 1000 lemparan dadu menggunakan generator angka acak, dan melakukan pengujian yang sama.

```
>n=1000; t=random([1,n*6]); chitest(count(t*6,6),dup(n,6)')
```

0.528028118442

Mari kita uji nilai rata-rata 100 dengan uji-t.

```
>s=200+normal([1,100])*10; ...
>ttest(mean(s),dev(s),100,200)
```

0.200555093084

Fungsi ttest() membutuhkan nilai rata-rata, simpangan, jumlah data, dan nilai rata-rata untuk diuji.

Sekarang mari kita periksa dua pengukuran untuk rata-rata yang sama. Kami menolak hipotesis bahwa mereka memiliki rata-rata yang sama, jika hasilnya <0,05.

```
>tcomparedata(normal(1,10),normal(1,10))
```

0.210166540976

Jika kami menambahkan bias ke satu distribusi, kami mendapat lebih banyak penolakan. Ulangi simulasi ini beberapa kali untuk melihat efeknya.

```
>tcomparedata(normal(1,10),normal(1,10)+2)
```

0.000282252410978

Dalam contoh berikutnya, kami menghasilkan 20 lemparan dadu acak 100 kali dan menghitungnya. Harus ada rata-rata $20/6=3,3$.

```
>R=random(100,20); R=sum(R*6<=1)'; mean(R)
```

3.1

Kami sekarang membandingkan jumlah satu dengan distribusi binomial. Pertama kita memplot distribusi satuan.

```
>plot2d(R,distribution=max(R)+1,even=1,style="\\");

>t=count(R,21);
```

Kemudian kami menghitung nilai yang diharapkan.

```
>n=0:20; b=bin(20,n)*(1/6)^n*(5/6)^(20-n)*100;
```

Kita harus mengumpulkan beberapa angka untuk mendapatkan kategori yang cukup besar.

```
>t1=sum(t[1:2])|t[3:7]|sum(t[8:21]); ...
>b1=sum(b[1:2])|b[3:7]|sum(b[8:21]);
```

Uji chi-square menolak hipotesis bahwa distribusi kita adalah distribusi binomial, jika hasilnya <0,05.

```
>chitest(t1,b1)
```

0.674181062604

Contoh berikut berisi hasil dari dua kelompok orang (pria dan wanita, katakanlah) memilih satu dari enam partai.

```
>A=[23,37,43,52,64,74;27,39,41,49,63,76]; ...  
> writetable(A,wc=6,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	23	37	43	52	64	74
f	27	39	41	49	63	76

Kami ingin menguji independensi suara dari jenis kelamin. Tes tabel chi^2 melakukan ini. Hasilnya terlalu besar untuk menolak kemerdekaan. Jadi kami tidak bisa mengatakan, jika pemungutan suara tergantung pada jenis kelamin dari data tersebut.

```
>tabletest(A)
```

0.990701632326

Berikut adalah tabel yang diharapkan, jika kita mengasumsikan frekuensi pemungutan suara yang diamati.

```
>writetable(expectedtable(A),wc=6,dc=1,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	24.9	37.9	41.9	50.3	63.3	74.7
f	25.1	38.1	42.1	50.7	63.7	75.3

Kita dapat menghitung koefisien kontingensi yang dikoreksi. Karena sangat mendekati 0, kami menyimpulkan bahwa pemungutan suara tidak bergantung pada jenis kelamin.

```
>contingency(A)
```

0.0427225484717

Beberapa Tes Lagi

Selanjutnya kami menggunakan analisis varians (F-test) untuk menguji tiga sampel data yang terdistribusi normal untuk nilai rata-rata yang sama. Metode tersebut dinamakan ANOVA (analysis of variance). Di Euler, fungsi varanalysis() digunakan.

```
>x1=[109,111,98,119,91,118,109,99,115,109,94]; mean(x1),
```

106.545454545

```
>x2=[120,124,115,139,114,110,113,120,117]; mean(x2),
```

119.111111111

```
>x3=[120,112,115,110,105,134,105,130,121,111]; mean(x3)
```

116.3

```
>varanalysis(x1,x2,x3)
```

0.0138048221371

Ini berarti, kami menolak hipotesis dengan nilai rata-rata yang sama. Kami melakukan ini dengan probabilitas kesalahan 1,3%.

Ada juga uji median yang menolak sampel data dengan distribusi rata-rata yang berbeda menguji median sampel bersatu.

```
>a=[56,66,68,49,61,53,45,58,54];
>b=[72,81,51,73,69,78,59,67,65,71,68,71];
>mediantest(a,b)
```

0.0241724220052

Tes lain tentang kesetaraan adalah tes peringkat. Ini jauh lebih tajam daripada tes median.

```
>ranktest(a,b)
```

0.00199969612469

Dalam contoh berikut, kedua distribusi memiliki rata-rata yang sama.

```
>ranktest(random(1,100),random(1,50)*3-1)
```

0.3598701243

Mari kita coba mensimulasikan dua perlakuan a dan b yang diterapkan pada orang yang berbeda.

```
>a=[8.0,7.4,5.9,9.4,8.6,8.2,7.6,8.1,6.2,8.9];  
>b=[6.8,7.1,6.8,8.3,7.9,7.2,7.4,6.8,6.8,8.1];
```

Tes signum memutuskan, jika a lebih baik dari b.

```
>signtest(a,b)
```

0.0546875

Ini terlalu banyak kesalahan. Kita tidak dapat menolak bahwa a sama baiknya dengan b.

Tes Wilcoxon lebih tajam dari tes ini, tetapi bergantung pada nilai kuantitatif perbedaannya.

```
>wilcoxon(a,b)
```

0.0296680599405

Mari kita coba dua tes lagi menggunakan rangkaian yang dihasilkan.

```
>>wilcoxon(normal(1,20),normal(1,20)-1)
```

0.0076200305916

```
>>wilcoxon(normal(1,20),normal(1,20))
```

0.5

Angka Acak

Berikut ini adalah tes untuk generator angka acak. Euler menggunakan generator yang sangat bagus, jadi kita tidak perlu berharap ada masalah.

Pertama kami menghasilkan sepuluh juta angka acak di [0,1].

```
>n:=10000000; r:=random(1,n);
```

Selanjutnya kita menghitung jarak antara dua angka kurang dari 0,05.

```
>a:=0.05; d:=differences(nonzeros(r<a));
```

Akhirnya, kami memplot berapa kali, setiap jarak terjadi, dan membandingkannya dengan nilai yang diharapkan.

```
>m=getmultiplicities(1:100,d); plot2d(m); ...
> plot2d("n*(1-a)^(x-1)*a^2",color=red,>add):
```

Hapus datanya.

```
>remvalue n;
```

Pengantar untuk Pengguna Proyek R

Jelas, EMT tidak bersaing dengan R sebagai paket statistik. Namun, ada banyak prosedur dan fungsi statistik yang tersedia di EMT juga. Jadi EMT dapat memenuhi kebutuhan dasar. Lagi pula, EMT hadir dengan paket numerik dan sistem aljabar komputer.

Notebook ini cocok untuk Anda jika sudah familiar dengan R, namun perlu mengetahui perbedaan sintaks EMT dan R. Kami mencoba memberikan gambaran umum tentang hal-hal yang jelas dan kurang jelas yang perlu Anda ketahui.

Selain itu, kami mencari cara untuk bertukar data antara kedua sistem.

Perhatikan bahwa ini adalah pekerjaan yang sedang berjalan.

Sintaks Dasar

Hal pertama yang Anda pelajari di R adalah membuat vektor. Di EMT, perbedaan utamanya adalah operator : dapat mengambil ukuran langkah. Apalagi daya ikatnya rendah.

```
>n=10; 0:n/20:n-1
```

```
[0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5,
 7, 7.5, 8, 8.5, 9]
```

Fungsi c() tidak ada. Dimungkinkan untuk menggunakan vektor untuk menggabungkan berbagai hal.

Contoh berikut, seperti banyak lainnya, dari "Introduction to R" yang disertakan dengan proyek R. Jika Anda membaca PDF ini, Anda akan menemukan bahwa saya mengikuti jalannya dalam tutorial ini.

```
>x=[10.4, 5.6, 3.1, 6.4, 21.7]; [x,0,x]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 0, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Operator titik dua dengan ukuran langkah EMT diganti dengan fungsi seq() di R. Kita bisa menulis fungsi ini di EMT.

```
>function seq(a,b,c) := a:b:c; ...  
>seq(0,-0.1,-1)
```

```
[0, -0.1, -0.2, -0.3, -0.4, -0.5, -0.6, -0.7, -0.8, -0.9, -1]
```

Fungsi rep() dari R tidak ada di EMT. Untuk input vektor, dapat ditulis sebagai berikut.

```
>function rep(x:vector,n:index) := flatten(dup(x,n)); ...  
>rep(x,2)
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Perhatikan bahwa "=" atau ":=" digunakan untuk tugas. Operator "->" digunakan untuk unit di EMT.

```
>125km -> " miles"
```

```
77.6713990297 miles
```

The "<->" operator for assignment is misleading anyway, and not a good idea of R. The following will compare a and -4 in EMT.

```
>a=2; a<-4
```

```
0
```

Di R, "a<-4<3" berfungsi, tetapi "a<-4<-3" tidak. Saya juga memiliki ambiguitas serupa di EMT, tetapi mencoba menghilangkannya sedikit demi sedikit.

EMT dan R memiliki vektor tipe boolean. Namun dalam EMT, angka 0 dan 1 digunakan untuk mewakili salah dan benar. Di R, nilai benar dan salah tetap bisa digunakan dalam aritmatika biasa seperti di EMT.

```
>x<5, %*x
```

```
[0, 0, 1, 0, 0]
[0, 0, 3.1, 0, 0]
```

EMT melempar kesalahan atau menghasilkan NAN tergantung pada bendera "kesalahan".

```
>errors off; 0/0, isNaN(sqrt(-1)), errors on;
```

```
NAN
1
```

String sama di R dan EMT. Keduanya berada di lokal saat ini, bukan di Unicode.

Di R ada paket untuk Unicode. Di EMT, sebuah string dapat berupa string Unicode. String unicode dapat diterjemahkan ke pengkodean lokal dan sebaliknya. Selain itu, u"..." dapat berisi entitas HTML.

```
>u"\u00f6; Ren\u00e9 Grothmann"
```

© René Grothmann

Berikut ini mungkin atau mungkin tidak ditampilkan dengan benar di sistem Anda sebagai A dengan titik dan garis di atasnya. Itu tergantung pada font yang Anda gunakan.

```
>chartoutf([480])
```

Penggabungan string dilakukan dengan "+" atau "|". Itu bisa termasuk angka, yang akan dicetak dalam format saat ini.

```
>"pi = "+pi
```

```
pi = 3.14159265359
```

Pengindeksan

Sebagian besar waktu, ini akan berfungsi seperti di R.

Tetapi EMT akan menginterpretasikan indeks negatif dari belakang vektor, sedangkan R menginterpretasikan $x[n]$ sebagai x tanpa elemen ke-n.

```
>x, x[1:3], x[-2]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]  
[10.4, 5.6, 3.1]  
6.4
```

Perilaku R dapat dicapai dalam EMT dengan `drop()`.

```
>drop(x, 2)
```

```
[10.4, 3.1, 6.4, 21.7]
```

Vektor logis tidak diperlakukan berbeda sebagai indeks di EMT, berbeda dengan R. Anda perlu mengekstraksi elemen bukan nol terlebih dahulu di EMT.

```
>x, x>5, x[nonzeros(x>5)]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]  
[1, 1, 0, 1, 1]  
[10.4, 5.6, 6.4, 21.7]
```

Sama seperti di R, vektor indeks dapat berisi pengulangan.

```
>x[[1,2,2,1]]
```

```
[10.4, 5.6, 5.6, 10.4]
```

Tetapi nama untuk indeks tidak dimungkinkan di EMT. Untuk paket statistik, hal ini sering diperlukan untuk memudahkan akses ke elemen vektor.

Untuk meniru perilaku ini, kita dapat mendefinisikan fungsi sebagai berikut.

```
>function sel(v,i,s) := v[indexof(s,i)]; ...  
>s=["first","second","third","fourth"]; sel(x,[ "first","third"],s)
```

```

Trying to overwrite protected function sel!
Error in:
function sel v,i,s) := v[indexof(s,i)]; . . .
^

Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; . . .
^

Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; . . .
^
[10.4, 3.1]

```

Tipe Data

EMT memiliki lebih banyak tipe data tetap daripada R. Jelas, di R terdapat vektor yang tumbuh. Anda dapat menyetel vektor numerik kosong *v* dan menetapkan nilai ke elemen *v*[17]. Ini tidak mungkin di EMT.

Berikut ini agak tidak efisien.

```
>v=[]; for i=1 to 10000; v=v|i; end;
```

EMT sekarang akan membuat vektor dengan *v* dan *i* ditambahkan pada tumpukan dan menyalin vektor itu kembali ke variabel global *v*.

Semakin efisien pra-mendefinisikan vektor.

```
>v=zeros(10000); for i=1 to 10000; v[i]=i; end;
```

Untuk mengubah jenis tanggal di EMT, Anda dapat menggunakan fungsi seperti *complex()*.

```
>complex(1:4)
```

```
[ 1+0i , 2+0i , 3+0i , 4+0i ]
```

Konversi ke string hanya dimungkinkan untuk tipe data dasar. Format saat ini digunakan untuk penggabungan string sederhana. Tapi ada fungsi seperti print() atau frac(). Untuk vektor, Anda dapat dengan mudah menulis fungsi Anda sendiri.

```
>function tostr (v) ...
```

```
s=[";
loop 1 to length(v);
    s=s+print(v[#,2,0]);
    if #<length(v) then s=s+",";
endif;
end;
return s+"]";
endfunction
```

```
>tostr(linspace(0,1,10))
```

```
[0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90,1.00]
```

Untuk komunikasi dengan Maxima, terdapat fungsi convertm xm(), yang juga dapat digunakan untuk memformat vektor untuk output.

```
>convertm xm(1:10)
```

```
[1,2,3,4,5,6,7,8,9,10]
```

Untuk Latex, perintah tex dapat digunakan untuk mendapatkan perintah Latex.

```
>tex(&[1,2,3])
```

```
\left[ 1 , 2 , 3 \right]
```

Faktor dan Tabel

Dalam pengantar R ada contoh dengan apa yang disebut faktor.

Berikut ini adalah daftar wilayah dari 30 negara bagian.

```
>austates = ["tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", ...
>"qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas", ...
>"sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa", ...
>"sa", "act", "nsw", "vic", "vic", "act"];
```

Asumsikan, kita memiliki pendapatan yang sesuai di setiap negara bagian.

```
>incomes = [60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, ...
>61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46, ...
>59, 46, 58, 43];
```

Sekarang, kami ingin menghitung rata-rata pendapatan di wilayah tersebut. Menjadi program statistik, R memiliki factor() dan tappy() untuk ini.

EMT dapat melakukannya dengan menemukan indeks wilayah di daftar unik wilayah.

```
>auterr=sort(unique(austates)); f=indexofsorted(auterr,austates)
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Pada saat itu, kita dapat menulis fungsi loop kita sendiri untuk melakukan sesuatu hanya untuk satu faktor.

Atau kita bisa meniru fungsi tapply() dengan cara berikut.

```
>function map_tappl (i; f$call, cat, x) ...
```

```
u=sort(unique(cat));
f=indexof(u,cat);
return f$(x[nonzeros(f==indexof(u,i))]);
endfunction
```

Ini sedikit tidak efisien, karena menghitung wilayah unik untuk setiap i, tetapi berhasil.

```
>tappl(auterr,"mean",austates,incomes)
```

```
[44.5, 57.3333333333, 55.5, 53.6, 55, 60.5, 56, 52.25]
```

Perhatikan bahwa ini berfungsi untuk setiap vektor wilayah.

```
>tappl(["act","nsw"],"mean",austates,incomes)
```

```
[44.5, 57.3333333333]
```

Sekarang, paket statistik EMT mendefinisikan tabel seperti pada R. Fungsi `readtable()` dan `writetable()` dapat digunakan untuk input dan output.

Sehingga kita bisa mencetak rata-rata pendapatan negara di daerah dengan cara yang bersahabat.

```
>writetable(tapply(auterr,"mean",austates,incomes),labc=auterr,wc=7)
```

act	nsw	nt	qld	sa	tas	vic	wa
44.5	57.33	55.5	53.6	55	60.5	56	52.25

Kami juga dapat mencoba meniru perilaku R sepenuhnya.

Faktor jelas harus disimpan dalam kumpulan dengan jenis dan kategori (negara bagian dan teritori dalam contoh kita). Untuk EMT, kami menambahkan indeks yang telah dihitung sebelumnya.

```
>function makef (t) ...
```

```
## Factor data
## Returns a collection with data t, unique data, indices.
## See: tapply
u=sort(unique(t));
return {{t,u,indexofsorted(u,t)}};
endfunction
```

```
>statef=makef(austates);
```

Sekarang elemen ketiga dari koleksi akan berisi indeks.

```
>statef[3]
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Sekarang kita bisa meniru `tapply()` dengan cara berikut. Ini akan mengembalikan tabel sebagai kumpulan data tabel dan judul kolom.

```
>function tapply (t:vector,tf,f$:call) ...
```

```

## Makes a table of data and factors
## tf : output of makef()
## See: makef
uf=tf[2]; f=tf[3]; x=zeros(length(uf));
for i=1 to length(uf);
    ind=nonzeros(f==i);
    if length(ind)==0 then x[i]=NAN;
    else x[i]=f$(t[ind]);
    endif;
end;
return {{x,uf}};
endfunction

```

Kami tidak menambahkan banyak pengecekan tipe di sini. Satu-satunya tindakan pencegahan menyangkut kategori (faktor) tanpa data. Tetapi orang harus memeriksa panjang t yang benar dan kebenaran koleksi tf.

Tabel ini dapat dicetak sebagai tabel dengan writetable().

```
>writetable(tapply(incomes,statef,"mean"),wc=7)
```

act	nsw	nt	qld	sa	tas	vic	wa
44.5	57.33	55.5	53.6	55	60.5	56	52.25

Array

EMT hanya memiliki dua dimensi untuk array. Tipe datanya disebut matriks. Namun, akan mudah untuk menulis fungsi untuk dimensi yang lebih tinggi atau pustaka C untuk ini.

R memiliki lebih dari dua dimensi. Di R array adalah vektor dengan bidang dimensi.

Dalam EMT, vektor adalah matriks dengan satu baris. Itu dapat dibuat menjadi matriks dengan redim().

```
>shortformat; X=redim(1:20,4,5)
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Ekstraksi baris dan kolom, atau sub-matriks, sangat mirip dengan R.

```
>X[,2:3]
```

2	3
7	8
12	13
17	18

Namun, dalam R dimungkinkan untuk menetapkan daftar indeks spesifik vektor ke suatu nilai. Hal yang sama dimungkinkan di EMT hanya dengan satu putaran.

```
>function setmatrixvalue (M, i, j, v) ...
loop 1 to max(length(i),length(j),length(v))
  M[i#{},j#{}] = v#;
end;
endfunction
```

Kami mendemonstrasikan ini untuk menunjukkan bahwa matriks dilewatkan dengan referensi di EMT. Jika Anda tidak ingin mengubah matriks asli M, Anda perlu menyalinnya ke dalam fungsi.

```
>setmatrixvalue(X,1:3,3:-1:1,0); X,
```

1	2	0	4	5
6	0	8	9	10
0	12	13	14	15
16	17	18	19	20

Produk luar di EMT hanya dapat dilakukan di antara vektor. Ini otomatis karena bahasa matriks. Satu vektor harus berupa vektor kolom dan yang lainnya vektor baris.

```
>(1:5)*(1:5)'
```

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Dalam pengantar PDF untuk R ada contoh, yang menghitung distribusi ab-cd untuk a,b,c,d dipilih dari 0 sampai n secara acak. Solusi dalam R adalah membentuk matriks 4 dimensi dan menjalankan table() di atasnya.

Tentu saja, ini bisa dicapai dengan satu putaran. Tapi loop tidak efektif di EMT atau R. Di EMT, kita bisa menulis loop di C dan itu akan menjadi solusi tercepat.

Tapi kami ingin meniru perilaku R. Untuk ini, kami perlu meratakan perkalian ab dan membuat matriks ab-cd.

```
>a=0:6; b=a'; p=flatten(a*b); q=flatten(p-p'); ...
>u=sort(unique(q)); f=getmultiplicities(u,q); ...
>statplot(u,f,"h"):
```

Selain perkalian yang tepat, EMT dapat menghitung frekuensi dalam vektor.

```
>getfrequencies(q,-50:10:50)
```

```
[0, 23, 132, 316, 602, 801, 333, 141, 53, 0]
```

Cara paling mudah untuk memplot ini sebagai distribusi adalah sebagai berikut.

```
>plot2d(q,distribution=11):
```

Tetapi juga memungkinkan untuk melakukan pra-perhitungan hitungan dalam interval yang dipilih sebelumnya. Tentu saja, berikut ini menggunakan getfrequencies() secara internal.

Karena fungsi histo() mengembalikan frekuensi, kita perlu menskalakannya sehingga integral di bawah grafik batang adalah 1.

```
>{x,y}=histo(q,v=-55:10:55); y=y/sum(y)/differences(x); ...
>plot2d(x,y,>bar,style="/"):
```

Daftar

EMT memiliki dua jenis daftar. Salah satunya adalah daftar global yang bisa berubah, dan yang lainnya adalah tipe daftar yang tidak bisa diubah. Kami tidak peduli dengan daftar global di sini.

Jenis daftar yang tidak dapat diubah disebut koleksi di EMT. Ini berperilaku seperti struktur di C, tetapi elemennya hanya diberi nomor dan tidak diberi nama.

```
>L={ {"Fred","Flintstone",40,[1990,1992]} }
```

```
Fred
Flintstone
40
[1990, 1992]
```

Saat ini elemen tidak memiliki nama, meskipun nama dapat diatur untuk tujuan khusus. Mereka diakses oleh nomor.

```
> (L[4])[2]
```

1.992

File Input dan Output (Membaca dan Menulis Data)

Anda sering ingin mengimpor matriks data dari sumber lain ke EMT. Tutorial ini memberitahu Anda tentang banyak cara untuk mencapai hal ini. Fungsi sederhana adalah writematrix() dan readmatrix().

Mari kita tunjukkan cara membaca dan menulis vektor real ke file.

```
>a=random(1,100); mean(a), dev(a),
```

```
0.47241  
0.272
```

Untuk menulis data ke file, kami menggunakan fungsi writematrix().

Karena pengantar ini kemungkinan besar ada di direktori, di mana pengguna tidak memiliki akses tulis, kami menulis data ke direktori home pengguna. Untuk buku catatan sendiri, hal ini tidak diperlukan, karena file data akan ditulis ke dalam direktori yang sama.

```
>filename="test.dat";
```

Sekarang kita menulis vektor kolom a' ke file. Ini menghasilkan satu nomor di setiap baris file.

```
>writematrix(a',filename);
```

Untuk membaca data, kami menggunakan readmatrix().

```
>a=readmatrix(filename)';
```

Dan hapus file tersebut.

```
>fileremove(filename);  
>mean(a), dev(a),
```

```
0.47241  
0.272
```

Fungsi writematrix() atau writetable() dapat dikonfigurasi untuk bahasa lain.

Misalnya, jika Anda memiliki sistem bahasa Indonesia (titik desimal dengan koma), Excel Anda memerlukan nilai dengan koma desimal yang dipisahkan oleh titik koma dalam file csv (defaultnya adalah nilai yang dipisahkan koma). File berikut "test.csv" akan muncul di folder cuurent Anda.

```
>filename="test.csv"; ...  
>writematrix(random(5,3),file=filename,separator=",");
```

Anda sekarang dapat membuka file ini dengan Excel bahasa Indonesia secara langsung.

```
>fileremove(filename);
```

Terkadang kami memiliki string dengan token seperti berikut ini.

```
>s1:="f m m f m m m f f f m m f"; ...  
>s2:="f f f m m f f";
```

Untuk menandai ini, kami mendefinisikan vektor token.

```
>tok:=[ "f", "m" ]
```

```
f  
m
```

Kemudian kita dapat menghitung berapa kali setiap token muncul dalam string, dan memasukkan hasilnya ke dalam tabel.

```
>M:=getmultiplicities(tok,strtokens(s1))_ ...  
> getmultiplicities(tok,strtokens(s2));
```

Tulis tabel dengan header token.

```
>writetable(M, labc=tok, labr=1:2, wc=8)
```

	f	m
1	6	7
2	5	2

Untuk statika, EMT dapat membaca dan menulis tabel.

```
>file="test.dat"; open(file,"w"); ...
>writeln("A,B,C"); writematrix(random(3,3)); ...
>close();
```

The file looks like this.

```
>printfile(file)
```

```
A,B,C
0.6172797453910012, 0.1349437164469696, 0.9765569046173882
0.8793694482866689, 0.754131576093465, 0.5573849749647652
0.3088149680985474, 0.6428098853858375, 0.01934472827441148
```

Fungsi readtable() dalam bentuknya yang paling sederhana dapat membaca ini dan mengembalikan kumpulan nilai dan baris heading.

```
>L=readtable(file,>list);
```

Koleksi ini dapat dicetak dengan writetable() ke notebook, atau ke file.

```
>writetable(L,wc=10,dc=5)
```

A	B	C
0.61728	0.13494	0.97656
0.87937	0.75413	0.55738
0.30881	0.64281	0.01934

Matriks nilai adalah elemen pertama dari L. Perhatikan bahwa mean() dalam EMT menghitung nilai rata-rata dari baris matriks.

```
>mean(L[1])
```

```
0.57626  
0.7303  
0.32366
```

File CSV

Pertama, mari kita menulis matriks ke dalam file. Untuk hasilnya, kami membuat file di direktori kerja saat ini.

```
>file="test.csv"; ...  
>M=random(3,3); writematrix(M,file);
```

Berikut adalah isi dari file ini.

```
>printfile(file)
```

```
0.9591333160765152, 0.3217404262851627, 0.3029060479006795  
0.2434324611405652, 0.5496398069520404, 0.5496362073901471  
0.09188706512115921, 0.6495983173213233, 0.5372140438590169
```

CSV ini dapat dibuka pada sistem bahasa Inggris ke dalam Excel dengan klik dua kali. Jika Anda mendapatkan file seperti itu di sistem Jerman, Anda perlu mengimpor data ke Excel dengan memperhatikan titik desimal.

Tetapi titik desimal juga merupakan format default untuk EMT. Anda dapat membaca matriks dari file dengan readmatrix().

```
>readmatrix(file)
```

```
0.95913 0.32174 0.30291  
0.24343 0.54964 0.54964  
0.091887 0.6496 0.53721
```

Dimungkinkan untuk menulis beberapa matriks ke satu file. Perintah open() dapat membuka file untuk ditulis dengan parameter "w". Standarnya adalah "r" untuk membaca.

```
>open(file, "w"); writematrix(M); writematrix(M'); close();
```

Matriks dipisahkan oleh garis kosong. Untuk membaca matriks, buka file dan panggil readmatrix() beberapa kali.

```
>open(file); A=readmatrix(); B=readmatrix(); A==B, close();
```

1	0	0
0	1	0
0	0	1

Di Excel atau spreadsheet serupa, Anda dapat mengekspor matriks sebagai CSV (nilai yang dipisahkan koma). Di Excel 2007, gunakan "simpan sebagai" dan "format lain", lalu pilih "CSV". Pastikan, tabel saat ini hanya berisi data yang ingin Anda ekspor.

Ini sebuah contoh.

```
>printfile("excel-data.csv")
```

0;1000;1000
1;1051,271096;1072,508181
2;1105,170918;1150,273799
3;1161,834243;1233,67806
4;1221,402758;1323,129812
5;1284,025417;1419,067549
6;1349,858808;1521,961556
7;1419,067549;1632,31622
8;1491,824698;1750,6725
9;1568,312185;1877,610579
10;1648,721271;2013,752707

Seperti yang Anda lihat, sistem Jerman saya menggunakan titik koma sebagai pemisah dan koma desimal. Anda dapat mengubahnya di pengaturan sistem atau di Excel, tetapi tidak perlu membaca matriks ke dalam EMT.

Cara termudah untuk membaca ini ke Euler adalah readmatrix(). Semua koma diganti dengan titik dengan parameter >koma. Untuk CSV bahasa Inggris, hilangkan saja parameter ini.

```
>M=readmatrix("excel-data.csv",>comma)
```

0	1000	1000
1	1051.3	1072.5
2	1105.2	1150.3
3	1161.8	1233.7
4	1221.4	1323.1
5	1284	1419.1

```

6      1349.9      1522
7      1419.1      1632.3
8      1491.8      1750.7
9      1568.3      1877.6
10     1648.7      2013.8

```

Let us plot this.

```
>plot2d(M' [1],M' [2:3],>points,color=[red,green]):
```

Ada cara yang lebih mendasar untuk membaca data dari file. Anda dapat membuka file dan membaca angka baris demi baris. Fungsi getvectorline() akan membaca angka dari baris data. Secara default, ini mengharapkan titik desimal. Tapi itu juga bisa menggunakan koma desimal, jika Anda memanggil setdecimaldot(",") sebelum Anda menggunakan fungsi ini.

Fungsi berikut adalah contoh untuk ini. Itu akan berhenti di akhir file atau baris kosong.

```
>function myload (file) ...
```

```

open(file);
M=[];
repeat
  until eof();
  v=getvectorline(3);
  if length(v)>0 then M=M_v; else break; endif;
end;
return M;
close(file);
endfunction

```

```
>myload(file)
```

```

0.95913   0.32174   0.30291
0.24343   0.54964   0.54964
0.091887  0.6496    0.53721

```

Dimungkinkan juga untuk membaca semua angka dalam file itu dengan getvector().

```
>open(file); v=getvector(10000); close(); redim(v[1:9],3,3)
```

```

0.95913   0.32174   0.30291
0.24343   0.54964   0.54964
0.091887  0.6496    0.53721

```

Thus it is very easy to save a vector of values, one value in each line and read back this vector.

```
>v=random(1000); mean(v)
```

0.4925

```
>writematrix(v',file); mean(readmatrix(file)')
```

0.4925

Menggunakan Tabel

Tabel dapat digunakan untuk membaca atau menulis data numerik. Sebagai contoh, kami menulis tabel dengan tajuk baris dan kolom ke file.

```
>file="test.tab"; M=random(3,3); ...
>open(file,"w"); ...
>writetable(M,separator=",",labc=["one","two","three"]); ...
>close(); ...
>printfile(file)
```

one	two	three
0.89,	0.57,	0.02
0.35,	1,	0.52
0.72,	0.18,	0.95

Ini dapat diimpor ke Excel.

Untuk membaca file di EMT, kami menggunakan readtable().

```
>{M,headings}=readtable(file,>clabs); ...
>writetable(M,labc=headings)
```

one	two	three
0.89	0.57	0.02
0.35	1	0.52
0.72	0.18	0.95

Menganalisis Garis

Anda bahkan dapat mengevaluasi setiap baris dengan tangan. Misalkan, kita memiliki garis dengan format berikut.

```
>line="2020-11-03,Tue,1'114.05"
```

2020-11-03, Tue, 1'114.05

Pertama kita dapat menandai garis.

```
>vt=strtoks(line)
```

2020-11-03

Tue

1'114.05

Kemudian kita dapat mengevaluasi setiap elemen garis menggunakan evaluasi yang sesuai.

```
>day(vt[1]), ...
>indexof(["mon","tue","wed","thu","fri","sat","sun"],tolower(vt[2])), ...
>strrepl(vt[3], "'", "")()
```

7.3816e+05

2

1114

Menggunakan ekspresi reguler, dimungkinkan untuk mengekstraksi hampir semua informasi dari sebaris data.

Asumsikan kita memiliki baris berikut sebuah dokumen HTML.

```
>line="<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>"
```

<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>

Untuk mengekstrak ini, kami menggunakan ekspresi reguler, yang mencari

- tanda kurung tutup >,
- string apa pun yang tidak mengandung tanda kurung dengan

sub-pertandingan "(...)".

- braket pembuka dan penutup menggunakan solusi terpendek,
- sekali lagi string apa pun yang tidak mengandung tanda kurung,
- dan tanda kurung buka <.

Ekspresi reguler agak sulit dipelajari tetapi sangat kuat.

```
>{pos,s,vt}=strxfind(line,">([^\>]+)<.+?>([^\>]+)<" );
```

Hasilnya adalah posisi kecocokan, string yang cocok, dan vektor string untuk sub-kecocokan.

```
>for k=1:length(vt); vt[k](), end;
```

```
1145.5  
5.6
```

Ini adalah fungsi yang membaca semua item numerik antara <td> dan </td>.

```
>function readtd (line) ...
```

```
v=[]; cp=0;  
repeat  
  {pos,s,vt}=strxfind(line,<td.*?>(.+?)</td>,cp);  
  until pos==0;  
  if length(vt)>0 then v=v|vt[1]; endif;  
  cp=pos+strlen(s);  
end;  
return v;  
endfunction
```

```
>readtd(line+"<td>non-numerical</td>")
```

```
1145.45  
5.6  
-4.5  
non-numerical
```

Situs web atau file dengan URL dapat dibuka di EMT dan dapat dibaca baris demi baris. Dalam contoh, kami membaca versi terkini dari situs EMT. Kami menggunakan ekspresi reguler untuk memindai "Versi ..." dalam judul.

```
>function readversion () ...
```

```
urlopen("http://www.euler-math-toolbox.de/Programs/Changes.html");
repeat
    until urleof();
    s=urlgetline();
    k=strfind(s,"Version ",1);
    if k>0 then substring(s,k,strfind(s,<,k)-1), break; endif;
end;
urlclose();
endfunction
```

```
>readversion
```

Version 2022-05-18

Input dan Output Variabel

Anda dapat menulis variabel dalam bentuk definisi Euler ke file atau ke baris perintah.

```
>writevar(pi, "mypi");
```

```
mypi = 3.141592653589793;
```

Untuk pengujian, kami membuat file Euler di direktori kerja EMT.

```
>file="test.e"; ...
>writevar(random(2,2), "M", file); ...
>printfile(file, 3)
```

```
M = [ ..
0.395371271114125, 0.533449516791482;
0.6914986603788036, 0.5888399658964002];
```

We can now load the file. It will define the matrix M.

```
>load(file); show M,
```

```
M =  
0.39537 0.53345  
0.6915 0.58884
```

By the way, jika writevar() digunakan pada variabel, itu akan mencetak definisi variabel dengan nama variabel ini.

```
>writevar(M); writevar(inch$)
```

```
M = [ ..  
0.395371271114125, 0.533449516791482;  
0.6914986603788036, 0.5888399658964002];  
inch$ = 0.0254;
```

Kami juga dapat membuka file baru atau menambahkan file yang sudah ada. Dalam contoh kami menambahkan file yang dihasilkan sebelumnya.

```
>open(file, "a"); ...  
>writevar(random(2,2), "M1"); ...  
>writevar(random(3,1), "M2"); ...  
>close();  
>load(file); show M1; show M2;
```

```
M1 =  
0.82529 0.39421  
0.87526 0.31661  
M2 =  
0.42648  
0.56656  
0.7148
```

Untuk menghapus file apa pun gunakan filerremove().

```
>filerremove(file);
```

Vektor baris dalam file tidak memerlukan koma, jika setiap angka berada di baris baru. Mari kita buat file seperti itu, menulis setiap baris satu per satu dengan writeln().

```

>open(file, "w"); writeln("M = ["); ...
>for i=1 to 5; writeln("'" +random()); end; ...
>writeln("]"); close(); ...
>printfile(file)

```

```

M = [
0.590934164929
0.423328852739
0.677100375552
0.664538356236
0.628889443099
];

```

```

>load(file); M

```

```

[0.59093, 0.42333, 0.6771, 0.66454, 0.62889]

```

LATIHAN SOAL

Buatlah tabel data penjualan laptop dan grafik kenaikan dan penurunan berdasarkan jumlah penjualan unit laptop dari tahun 2019-2022 dari data berikut :

HP : [4300,4800,5000,4900]

Lenovo : [4500,5000,4900,5100]

Asus : [4900,4200,5200,5000]

Toshiba : [4000,4300,5000,5300]

Acer : [3800,4900,5100,5500]

TABEL DATA PENJUALAN DARI TAHUN 2019-2022

```

>BW := [ ...
>2019,4300,4500,4900,4000,3800; ...
>2020,4800,5000,4200,4300,4900; ...
>2021,5000,4900,5200,5000,5100; ...
>2022,4900,5100,5000,5300,5500];
>P:=[ "HP", "Lenovo", "Asus", "Toshiba", "Acer" ];
>BT:=BW[,2:6]; YT:=BW[,1]';
>writetable(BT,wc=6,dc=0,>fixed,labc=P,labr=YT)

```

	HP	Lenovo	Asus	Toshiba	Acer
2019	4300	4500	4900	4000	3800
2020	4800	5000	4200	4300	4900
2021	5000	4900	5200	5000	5100
2022	4900	5100	5000	5300	5500

Grafik kenaikan dan penurunan jumlah penjualan Laptop dari tahun 2019-2022

```
>BT1:=(BT.[1;1;1;1;1])'
```

```
[21500, 23200, 25200, 25800]
```

```
>statplot(YT,BT1,"b"):  
>BT2:=BT1/3
```

```
[7166.67, 7733.33, 8400, 8600]
```

```
>statplot(YT,BT2,"b"):
```

TERIMAKASIH
