

Tugas Pekan 16

rahmakomalla

30 November 2023

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

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

$$x^2 - 1 \over x - 1 = x + 1$$

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

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.

```
>(1+sqrt(5))/2
```

```
1.61803398875
```

```
>i=1:10
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>u=[1,2,3];  
>y=exp(u)
```

```
[2.71828, 7.38906, 20.0855]
```

```
>z=log(y)
```

```
[1, 2, 3]
```

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.

```
>H=zeros(4,4);  
>for i=1:4; for j=1:4; H[i,j]=1/(i+j-1); end; end;...  
>H
```

1	0.5	0.333333	0.25
0.5	0.333333	0.25	0.2
0.333333	0.25	0.2	0.166667
0.25	0.2	0.166667	0.142857

```
>a=12.34; x=a/2; tol=1e-6;  
>repeat;x=(x+a/x)/2; until abs(a-x^2)<tol;end;  
>x
```

```
3.51283362132
```

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 perlu ditulis dengan mengaktifkan easyunits.

```
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 mengatrtamilan 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 "defformat" atau "reset".

```
>longestformat; pi, defformat; 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, 0.193585,  
0.463387, 0.095153, 0.595017]
```

```
>normal(10)
```

```
[-0.495418, 1.6463, -0.390056, -1.98151, 3.44132, 0.308178, -0.733427,  
-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 imajiner

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

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 imajiner 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 imajiner 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 dikerjakan oleh Maxima.

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

$$(b + a)^2$$

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

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

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

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

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

```
>&(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 fungsi tri
```

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

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)
```

$$7$$

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

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

$$b^2 + (2a - 1)b + a^2 - a + 1$$

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

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

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

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

Tampilan Matematika Simbolik dengan LaTeX

Anda dapat menampilkan hasil perhitunagn 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), $factor(x^2+5*x+6)
>$solve(a*x^2+b*x+c,x) // rumus abc
>$ (a^2-b^2)/(a+b), $ratsimp(%)
```

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.

SOAL Sebuah kota kecil dengan penduduk 1000 orang terkena suatu

wabah penyakit menular. Jumlah orang yang sakit setelah t hari semenjak menyebarnya wabah penyakit dapat dimodelkan dengan fungsi sebagai berikut :

$$P(t) = 1000 \frac{1}{1 + 999e^{-0.603t}}$$

```
>p=1000/(1+999*exp(-0.603*10))
```

```
293.850719578
```

```
>round(p)
```

```
294
```

```
[a4paper,10pt]article  
eumat
```

EMT untuk Perhitungan Aljabar

Pada notebook ini Anda belajar menggunakan EMT untuk melakukan berbagai perhitungan terkait dengan materi atau topik dalam Aljabar. Kegiatan yang harus Anda lakukan adalah sebagai berikut:

- Membaca secara cermat dan teliti notebook ini;
- Menerjemahkan teks bahasa Inggris ke bahasa Indonesia;
- Mencoba contoh-contoh perhitungan (perintah EMT) dengan cara meng-ENTER setiap perintah EMT yang ada (pindahkan kursor ke baris perintah)
- Jika perlu Anda dapat memodifikasi perintah yang ada dan memberikan keterangan/penjelasan tambahan terkait hasilnya.
- Menyisipkan baris-baris perintah baru untuk mengerjakan soal-soal Aljabar dari file PDF yang saya berikan;
- Memberi catatan hasilnya.
- Jika perlu tuliskan soalnya pada teks notebook (menggunakan format LaTeX).
- Gunakan tampilan hasil semua perhitungan yang eksak atau simbolik dengan format LaTeX. (Seperti contoh-contoh pada notebook ini.)

Contoh pertama

Menyederhanakan bentuk aljabar:

$$6x^{-3}y^5 - 7x^2y^{-9}$$

```
>&6*x^(-3)*y^5-7*x^2*y^(-9)
```

$$-42 \frac{1}{xy^4}$$

Baris perintah berikut hanya akan mencetak hasil ekspresi, bukan tugas atau perintah

Menjabarkan:

$$(6x^{-3}+y^5)(-7x^2-y^{-9})$$

```
>&showev('expand((6*x^(-3)+y^5)*(-7*x^2-y^(-9))))
```

$$\text{expand}\left(\left(-\frac{1}{y^9} - 7x^2\right)(y^5 + 6x^3 - 7x^2y^5 - \frac{1}{y^4} - \frac{6}{x^3})\right)$$

penyelesaian soal-soal

1. menjabarkan

$$(6xy^3)(9x^4y^2)$$

```
>&showev('expand((6*x*y^3)*(9*x^4*y^2)))
```

$$\text{expand}(54x^5y^5) = 54x^5y^5$$

2. menjabarkan

$$(2x+3y+4)(2x+3y-4)$$

```
>&showev('expand((2*x+3*y+4)*(2*x+3*y-4))')
```

$$\begin{aligned} \text{expand}((3y+2x-4)(3y+2x+4)) \\ = 9y^2 + 12xy + 4x^2 - 16 \end{aligned}$$

3. menjabarkan

$$(-5m^4n^2)(6m^2n^3)$$

```
>&showev('expand((-5*m^4*n^2)*(6*m^2*n^3))')
```

$$\text{expand}(-30m^6n^5) = -30m^6n^5$$

4. menjabarkan

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

```
>&showev('expand((2*x+3*y)^2)')
```

$$\begin{aligned} \text{expand}((3y+2x)^2) = 9y^2 + 12xy + 4 \\ x^2 \end{aligned}$$

5. menyederhanakan

$$(5x^2+4xy-3y^2+2)-(9x^2-4xy+2y^2-1)$$

```
>&((5*x^2+4*x*y-3*y^2+2)-(9*x^2-4*x*y+2*y^2-1))
```

$$-5y^2+8xy-4x^2+3$$

Baris Perintah

baris perintah euler terdiri dari satu atau beberapa perintah euler yang diikuti oleh titik koma ";" atau koma ",". Titik koma mencegah pencetakan hasil. koma setelah perintah terakhir dapat dihilangkan. Baris perintah berikut hanya akan mencetak hasil ekspresi, bukan tugas atau perintah

```
>r:=2; h:=4; pi*r^2*h/3
```

16.7551608191

perintah harus dipisahkan dengan yang kosong. Baris perintah berikut dicetak dua hasilnya

```
>pi*2*r*h, %2*pi*r*h // Ingat tanda % menyatakan hasil perhitungan terakhir sebelumnya
```

50.2654824574

100.530964915

baris perintah dijalankan sesuai urutan yang ditekan pengguna kembali. jadi akan mendapatkan a nilai baru setiap kali menjalankan baris kedua

```
>x := 1;  
>x := cos(x) // nilai cosinus (x dalam radian)
```

0.540302305868

```
>x := cos(x)
```

0.857553215846

jika dua baris dihubungkan dengan "...". kedua baris akan selalu dijalankan secara bersamaan

```
>x := 1.5; ...  
>x := (x+2/x)/2, x := (x+2/x)/2, x := (x+2/x)/2,
```

```
1.41666666667  
1.41421568627  
1.41421356237
```

ini juga merupakan cara terbaik untuk menyebarkan perintah panjang pada dua baris atau lebih. dengan menekan Ctrl+return untuk membagi baris menjadi dua pada posisi kursor saat ini, atau Ctrl+back untuk bergabung dengan garis.

Untuk melipat semua multi garis tekan Ctrl+L. maka hanya baris berikutnya yang akan terlihat, jika salah satunya mempunyai fokus

untuk melipat satu multi-garis, mulailah baris pertama dengan "+"

```
>%+ x=4+5; ...
```

garis yang dimulai dengan % tidak akan terlihat sama sekali

```
81
```

Euler mendukung loop di baris perintah, asalkan cocok menjadi satu baris atau multi baris. dalam program, batasan ini tidak berlaku. untuk info lebih lanjut baca pendahuluan berikut

```
>x=1; for i=1 to 5; x := (x+2/x)/2, end; // menghitung akar 2
```

```
1.5  
1.41666666667  
1.41421568627  
1.41421356237  
1.41421356237
```

tidak masalah menggunakan multi baris. pastikan baris diakhir dengan "..."

```
>x := 1.5; // comments go here before the ...  
>repeat xnew:=(x+2/x)/2; until xnew~x; ...  
>  x := xnew; ...  
>end; ...  
>x,
```

```
1.41421356237
```

struktur bersyarat juga berfungsi

```
>if E^pi>pi^E; then "Thought so!", endif;
```

Thought so!

saat menjalankan perintah, kursor dapat berada di posisi mana pun di baris perintah. Anda dapat kembali ke perintah sebelumnya atau melompat ke perintah berikutnya dengan tombol panah. Atau anda dapat mengklik bagian komentar diatas perintah untuk membuka perintah. saat menggerakkan kursor di sepanjang garis, pasangan tanda kurung atau tanda kurung buka dan tutup akan disorot. perhatikan juga baris status. setelah tanda kurung buka dari fungsi `sqrt()`, baris status akan menampilkan teks bantuan untuk fungsi tersebut. Jalankan perintah dengan kunci kembali

```
>sqrt(sin(10°)/cos(20°))
```

0.429875017772

untuk melihat bantuan untuk perintah baru, buka jendela bantuan dengan `f1`. Di sana, anda dapat memasukkan teks untuk dicari. Pada baris kosong, bantuan untuk jendela bantuan akan ditampilkan. Anda dapat menekan `escape` untuk menghapus garis atau menutup jendela bantuan. Anda dapat mengklik dua kali pada perintah apa pun untuk membuka bantuan untuk perintah ini. Coba klik dua kali perintah `exp` dibawah ini pada baris perintah

```
>exp(log(2.5))
```

2.5

Anda juga dapat menyalin dan menempel di Euler menggunakan `Ctrl-C` dan `Ctrl-V`. Untuk menandai teks, seret mouse atau gunakan `shift` bersamaan dengan tombol kursor apapun. Selain itu anda dapat menyalin tanda kurung yang disorot.

penyelesaian soal-soal

1. jika diketahui $x=5$ dan $y=10$, selesaikan

$$x = (5y)^{\frac{1}{x^2}}, \quad x = \left(\frac{x^4}{x}\right)y, \quad x = \left(\frac{x+5}{2}\right)y$$

```
>x:=5; y:=10;...  
>x := (5*y/x)^2, x := (x^4/x)*y, x := (x+5/2)*y
```

100
10000000
100000025

2. jika diketahui $\pi=3,14$ dan $r=50$, carilah

$$\pi r^2$$

```
>pi := 3.14 ; r:=50;...  
>%pi*r^2
```

7850

Dasar Syntax

Euler mengetahui fungsi matematika biasa. Seperti yang anda lihat di atas, fungsi trigonometri bekerja dalam radian atau derajat. untuk mengonversi ke derajat, tambahkan simbol derajat (dengan tombol F7) ke nilainya, atau gunakan fungsi $\text{rad}(x)$. Fungsi akar kuadrat disebut sqrt di Euler. Tentu saja, $x^{(1/2)}$ juga dimungkinkan untuk menyetel variabel, gunakan "=" atau ":=". Agar jelas, pendahuluan ini menggunakan bentuk yang terakhir. spasi tidaklah penting. tapi jarak antar perintah diharapkan. Beberapa perintah dalam satu baris dipisahkan dengan ",", "aa" atau ";". Titik koma menahan output perintah. Di akhir baris perintah, ",", " diasumsikan jika ",", " hilang

```
>g:=9.81; t:=2.5; 1/2*g*t^2
```

30.65625

emt menggunakan sintaks pemrograman untuk ekspresi.

$$e^2 \cdot \left(\frac{1}{3+4 \log(0.6)} + \frac{1}{7} \right)$$

Anda harus mengatur tanda kurung yang benar dan menggunakan "/" untuk pecahan. Perhatikan tanda kurung yang disorot untuk mendapatkan bantuan. Perhatikan bahwa konstanta Euler e diberi nama E dalam EMT.

```
>E^2*(1/(3+4*log(0.6))+1/7)
```

8.77908249441

untuk menghitung ekspresi rumit seperti ini

$$\left(\frac{\frac{1}{7} + \frac{1}{8} + \frac{2}{3}}{\frac{1}{3} + \frac{1}{2}} \right)^2 \pi$$

kamu harus memasukkannya dalam formulir baris

```
>((1/7 + 1/8 + 2) / (1/3 + 1/2))^2 * pi
```

```
23.2553846939
```

Letakkan tanda kurung dengan hati-hati di sekitar sub-ekspresi yang perlu dihitung terlebih dahulu. EMT membantu anda dalam menyorot ekspresi yang mengakhiri tanda kurung tutup. Anda juga harus memasukkan nama "pi" untuk huruf Yunani pi.

Hasil perhitungan ini berupa bilangan floating point. ini secara default dicetak dengan akurasi sekitar 12 digit. Di baris perintah berikut, kita juga mempelajari bagaimana kita bisa merujuk ke hasil sebelumnya dalam baris yang sama

```
>1/3+1/7, fraction %
```

```
0.47619047619
```

```
10/21
```

Perintah Euler dapat berupa ekspresi atau perintah primitif. Ekspresi terbuat dari operator dan fungsi. Jika perlu, harus berisi tanda kurung untuk memaksakan urutan eksekusi yang benar. Jika ragu, memasang braket adalah ide yang bagus. Perhatikan bahwa EMT menampilkan tanda kurung buka dan tutup saat mengedit baris perintah.

```
>(cos(pi/4)+1)^3*(sin(pi/4)+1)^2
```

```
14.5002323902
```

Operator numerik Euler meliputi

- + unary or operator tambah (plus)
- unary or operator pengurangan (minus)
- *, /
- . produk matriks
- a^b pangkat untuk a positif atau bilangan bulat b (a**b juga

berfungsi)

N! operasi faktorial

dan masih banyak lagi.

Berikut beberapa fungsi yang mungkin Anda perlukan. masih banyak lagi.

- sin,cos,tan,atan,asin,acos,rad,deg
- log,exp,log10,sqrt,logbase
- bin,logbin,logfac,mod,floor,ceil,round,abs,sign
- conj,re,im,arg,conj,real,complex
- beta,betai,gamma,complexgamma,ellrf,ellf,ellrd,elle
- bitand,bitor,bitxor,bitnot

beberapa perintah memiliki alias, e.g. ln i log.


```
>ln(E^2), arctan(tan(0.5))
```

2
0.5

$$>\sin(30^\circ)$$

0.5

pastikan untuk menggunakan tanda kurung (tanda kurung bulat), setiap kali ada keraguan tentang urutan eksekusi! Berikut ini $(2^3)^4$ tidak sama dengan yang merupakan default untuk 2^3^4 di EMT (beberapa sistem numerik melakukannya dengan cara lain)

$$> 2^{3^4}, \quad (2^3)^4, \quad 2^{(3^4)}$$

```
2.41785163923e+24
4096
2.41785163923e+24
```

Bilangan Real

Tipe data primer pada Euler adalah bilangan real. Real dipresentasikan dalam format IEEE dengan akurasi sekitar 16 digit desimal

```
>longest 1/3
```

0.333333333333333333

Representasi ganda internal membutuhkan 8 byte

```
> printdual(1/3)
```

[illegible]

```
>printhex(1/3)
```

$$5.555555555555554 \times 10^{-1}$$

Strings

sebuah string di Euler didefinisikan dengan "..."

```
>"A string can contain anything."
```

A string can contain anything.

string dapat digabungkan dengan | atau dengan +. Ini juga berfungsi dengan angka, yang dalam hal ini diubah menjadi string

```
>"The area of the circle with radius " + 2 + " cm is " + pi*4 + " cm^2."
```

The area of the circle with radius 2 cm is 12.56 cm².

fungsi print juga mengubah angka menjadi string. Ini dapat memerlukan sejumlah digit dan sejumlahnya tempat(0 untuk keluaran padat), dan optimalnya satuan

```
>"Golden Ratio : " + print((1+sqrt(5))/2,5,0)
```

Golden Ratio : 1.61803

Ada string khusus none yang tidak dicetak. Itu dikembalikan oleh beberapa fungsi, ketika hasilnya tidak menjadi masalah. (ini dikembalikan secara otomatis, jika fungsi tidak memiliki pernyataan return)

```
>none
```

Ada string khusus none yang tidak dicetak. Untuk mengonversi string menjadi angka, cukup evaluasi saja. Ini juga berfungsi untuk ekspresi (lihat di bawah).

```
>"1234.5"()
```

1234.5

Untuk mendefinisikan vektor string, gunakan notasi vektor [...].

```
>v=["affe","charlie","bravo"]
```

affe
charlie
bravo

Vektor string kosong dilambangkan dengan [tidak ada]. Vektor string dapat digabungkan.

```
>w=[none]; w|v|v
```

affe
charlie
bravo
affe
charlie
bravo

String dapat berisi karakter Unicode. Secara internal, string ini berisi kode UTF-8. Untuk menghasilkan string seperti itu, gunakan `u"..."` dan salah satu entitas HTML. String Unicode dapat digabungkan seperti string lainnya.

```
>u"&alpha; = " + 45 + u"&deg;" // pdfLaTeX mungkin gagal menampilkan secara benar
```

= 45°

I

Di komentar, entitas yang sama seperti `, etc.` dapat digunakan. Ini mungkin merupakan alternatif cepat untuk lateks. (Detail lebih lanjut di komentar di bawah)

Ada beberapa fungsi untuk membuat atau menganalisis string unicode. Fungsi `strtochar()` akan mengenali string unicode, dan menerjemahkannya dengan benar

```
>v=strtochar(u"&Auml; is a German letter")
```

```
[196, 32, 105, 115, 32, 97, 32, 71, 101, 114, 109, 97, 110,
 32, 108, 101, 116, 116, 101, 114]
```

hasilnya adalah vektor angka unicode. Fungsi kebalikannya adalah `chartoutf()`

```
>v[1]=strtochar(u"&Uuml;")[1]; chartoutf(v)
```

Ü is a German letter

fungsi `utf()` dapat menerjemahkan string dengan entitas dalam variabel menjadi string unicode

```
>s="We have &alpha;=&beta;."; utf(s) // pdfLaTeX mungkin gagal menampilkan secara benar
```

We have =.

Dimungkinkan juga untuk menggunakan entitas numerik

```
>u"&#196;hnliches"
```

Ähnliches

Nilai Boolean

Nilai Boolean diwakili dengan `1=true` atau `0=false` di Euler. String dapat dibandingkan, seperti halnya angka.

```
>2<1, "apel"<"banana"
```

0

1

"and" adalah operator "&&" dan "or" adalah operator "||", seperti dalam bahasa C. (kata "and" dan "or" hanya dapat digunakan dalam kondisi "if".)

```
>2<E && E<3
```

1

operator Boolean mematuhi aturan bahasa matriks

```
>(1:10)>5, nonzeros(%)
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
[6, 7, 8, 9, 10]
```

Anda dapat menggunakan fungsi `nonzeros()` untuk mengekstrak elemen tertentu dari vektor. Dalam contoh ini, kita menggunakan kondisi `isprime(n)`.

```
>N=2|3:2:99 // N berisi elemen 2 dan bilangan2 ganjil dari 3 s.d. 99
```

```
[2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29,
31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57,
59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85,
87, 89, 91, 93, 95, 97, 99]
```

```
>N[nonzeros(isprime(N))] //pilih anggota2 N yang prima
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Format Keluaran

Format keluaran default EMT mencetak 12 digit. Untuk memastikan bahwa kami melihat defaultnya, kami mengatur ulang formatnya

```
>defformat; pi
```

3.14

Secara internal, EMT menggunakan standar IEEE untuk bilangan ganda dengan sekitar 16 digit desimal. Untuk melihat jumlah digit secara lengkap gunakan perintah "longestformat", atau kita gunakan operator "longest" untuk menampilkan hasilnya dalam format terpanjang.

```
>longest pi
```

3.14

Berikut adalah representasi heksadesimal internal dari bilangan ganda.

```
>printhex(pi)
```

```
3.23D70A3D70A3E*16^0
```

Format keluaran dapat diubah secara permanen dengan perintah format.

```
>format(12,5); 1/3, pi, sin(1)
```

```
0.33333  
3.14000  
0.84147
```

Standarnya adalah format(12).

```
>format(12); 1/3
```

```
0.333333333333
```

Fungsi seperti "shortestformat", "shortformat", "longformat" berfungsi untuk vektor dengan cara berikut.

```
>shortestformat; random(3,8)
```

```
0.66    0.2    0.89    0.28    0.53    0.31    0.44    0.3  
0.28    0.88    0.27    0.7    0.22    0.45    0.31    0.91  
0.19    0.46    0.095    0.6    0.43    0.73    0.47    0.32
```

The default format for scalars is format(12). But this can be changed.

```
>setscalarformat(5); pi
```

```
3.14
```

Fungsi "longestformat" mengatur format skalar

```
>longestformat; pi
```

```
3.14
```

Sebagai referensi, berikut adalah daftar format keluaran terpenting.

```
format terpendek, format pendek, format panjang, format terpanjang  
format(panjang,digit) format bagus(panjang)  
frakformat(panjang)  
merusak format
```

Akurasi internal EMT adalah sekitar 16 tempat desimal, yang merupakan standar IEEE. Nomor disimpan dalam format internal ini.

Namun format keluaran EMT dapat diatur dengan cara yang fleksibel.

```
>longestformat; pi,
```

3.14

```
>format(10,5); pi
```

3.14000

standarnya adalah `deformat()`.

```
>deformat; // default
```

ada operator pendek yang hanya mencetak satu nilai. operator "longest" akan mencetak semua digit nomor yang valid

```
>longest pi^2/2
```

4.9298

Ada juga operator singkat untuk mencetak hasil dalam format pecahan. Kami sudah menggunakannya di atas.

```
>fraction 1+1/2+1/3+1/4
```

25/12

Karena format internal menggunakan cara biner untuk menyimpan angka, nilai 0,1 tidak akan direpresentasikan secara tepat. Kesalahannya bertambah sedikit, seperti yang Anda lihat pada perhitungan berikut.

```
>longest 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1-1
```

-1.110223024625157e-16

Tetapi dengan "format panjang" default Anda tidak akan menyadarinya. Untuk kenyamanan, keluaran angka yang sangat kecil adalah 0.

```
>0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1-1
```

0

Ekspresi

String atau nama dapat digunakan untuk menyimpan ekspresi matematika, yang dapat dievaluasi dengan EMT. Untuk ini, gunakan tanda kurung setelah ekspresi. Jika Anda ingin menggunakan string sebagai ekspresi, gunakan konvensi untuk menamainya "fx" atau "fxy" dll. Ekspresi lebih diutamakan daripada fungsi.

Variabel global dapat digunakan dalam evaluasi.

```
>r:=2; fx:="pi*r^2"; longest fx()
```

12.56

Parameter ditetapkan ke x, y, dan z dalam urutan itu. Parameter tambahan dapat ditambahkan menggunakan parameter yang ditetapkan.

```
>fx:="a*sin(x)^2"; fx(5,a=-1)
```

-0.919535764538

Perhatikan bahwa ekspresi akan selalu menggunakan variabel global, meskipun ada variabel dalam fungsi dengan nama yang sama. (Jika tidak, evaluasi ekspresi dalam fungsi dapat memberikan hasil yang sangat membingungkan bagi pengguna yang memanggil fungsi tersebut.)

```
>at:=4; function f(expr,x,at) := expr(x); ...  
>f("at*x^2",3,5) // computes 4*3^2 not 5*3^2
```

36

Jika Anda ingin menggunakan nilai lain untuk "at" selain nilai global, Anda perlu menambahkan "at=value".

```
>at:=4; function f(expr,x,a) := expr(x,at=a); ...  
>f("at*x^2",3,5)
```

45

Sebagai referensi, kami mencatat bahwa koleksi panggilan (dibahas di tempat lain) dapat berisi ekspresi. Jadi kita bisa membuat contoh di atas sebagai berikut.

```
>at:=4; function f(expr,x) := expr(x); ...  
>f({"at*x^2",at=5},3)
```

45

Ekspresi dalam x sering digunakan seperti halnya fungsi.

Perhatikan bahwa mendefinisikan fungsi dengan nama yang sama seperti ekspresi simbolik global akan menghapus variabel ini untuk menghindari kebingungan antara ekspresi simbolik dan fungsi.

```
>f &= 5*x;
>function f(x) := 6*x;
>f(2)
```

12

Berdasarkan konvensi, ekspresi simbolik atau numerik harus diberi nama fx , fx_y , dll. Skema penamaan ini tidak boleh digunakan untuk fungsi.

```
>fx &= diff(x^x,x); &fx
```

$x^x(\log x + 1)$

A special form of an expression allows any variable as an unnamed parameter to the evaluation of the expression, not just "x", "y" etc. For this, start the expression with "@(variables) ...".

```
>"@(a,b) a^2+b^2", %(4,5)
```

@(a,b) a^2+b^2
41

This allows to manipulate expressions in other variables for functions of EMT which need an expression in "x".

The most elementary way to define a simple function is to store its formula in a symbolic or numerical expression. If the main variable is x, the expression can be evaluated just like a function.

As you see in the following example, global variables are visible during the evaluation.

```
>fx &= x^3-a*x; ...
>a=1.2; fx(0.5)
```

-0.475

All other variables in the expression can be specified in the evaluation using an assigned parameter.

```
>fx(0.5,a=1.1)
```

-0.425

An expression needs not be symbolic. This is necessary, if the expression contains functions, which are only known in the numerical kernel, not in Maxima.

Symbolic Mathematics

EMT does symbolic math with the help of Maxima. For details, start with the following tutorial, or browse the reference for Maxima. Experts in Maxima should note that there are differences in the syntax between the original syntax of Maxima and the default syntax of symbolic expressions in EMT. Symbolic math is integrated seamlessly into Euler with `&`. Any expression starting with `&` is a symbolic expression. It is evaluated and printed by Maxima.

First of all, Maxima has an "infinite" arithmetic which can handle very large numbers.

```
>&44!
```

```
2658271574788448768043625811014615890319638528000000000
```

This way, you can compute large results exactly. Let us compute

$$C(44,10) = 44! \frac{1}{34! \cdot 10!}$$

```
>& 44!/(34!*10!) // nilai C(44,10)
```

```
2481256778
```

Tentu saja, Maxima memiliki fungsi yang lebih efisien untuk ini (seperti halnya bagian numerik EMT).

```
>binomial(44,10) //menghitung C(44,10) menggunakan fungsi binomial()
```

```
2481256778
```

Untuk mempelajari lebih lanjut tentang fungsi tertentu, klik dua kali padanya. Misalnya, coba klik dua kali pada "`&binomial`" di baris perintah sebelumnya. Ini membuka dokumentasi Maxima yang disediakan oleh penulis program tersebut.

Anda akan mengetahui bahwa cara berikut juga bisa dilakukan.

$$C(x, 3) = x! \frac{1}{(x-3)!3!} = \frac{(x-2)(x-1)x}{6}$$

```
>binomial(x,3) // C(x,3)
```

$$(x-2)(x-1)x\overline{6}$$

If you want to replace x with any specific value use "with".

```
>&binomial(x,3) with x=10 // substitusi x=10 ke C(x,3)
```

120

That way you can use a solution of an equation in another equation.

Symbolic expressions are printed by Maxima in 2D form. The reason for this is a special symbolic flag in the string.

As you will have seen in previous and following examples, if you have LaTeX installed, you can print a symbolic expression with Latex. If not, the following command will issue an error message.

To print a symbolic expression with LaTeX, use \$ in front of & (or you may omit &) before the command. Do not run the Maxima commands with \$, if you don't have LaTeX installed.

```
>(3+x)/(x^2+1)
```

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

Symbolic expressions are parsed by Euler. If you need a complex syntax in one expression, you can enclose the expression in "...". To use more than a simple expression is possible, but strongly discouraged.

```
>&"v := 5; v^2"
```

For completeness, we remark that symbolic expressions can be used in programs, but need to be enclosed in quotes. Moreover, it is much more effective to call Maxima at compile time if possible.

```
>$expand((1+x)^4), $factor(diff(%,x)) // diff: turunan, factor: faktor
```

$$x^4 + 4x^3 + 6x^2 + 4x + 1$$

$$4(x+1)^3$$

Again, % refers to the previous result.
To make things easier we save the solution to a symbolic variable.
Symbolic variables are defined with "&=".

```
>fx &= (x+1)/(x^4+1); $fx
```

$$x+1 \frac{1}{x^4+1}$$

Symbolic expressions can be used in other symbolic expressions.

```
>&factor(diff(fx,x))
```

$$-3x^4 - 4x^3 + 1 \frac{1}{(x^4+1)^2}$$

A direct input of Maxima commands is available too. Start the command line with ":::". The syntax of Maxima is adapted to the syntax of EMT (called the "compatibility mode").

```
>&factor(20!)
```

2432902008176640000

```
>::: factor(10!)
```

$$2^8 3^4 5^2 7^2$$

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

$$2^{18} 3^8 5^4 7^2 11 13 17 19$$

If you are an expert in Maxima, you may wish to use the original syntax of Maxima. You can do this with "::::".

```
>:::: av:g av^2;
```

$$g^2$$

```
>fx &= x^3*exp(x), fx
```

$$x^3 E^x$$

```
x^3e^x
```

Such variables can be used in other symbolic expressions. Note, that in the following command the right hand side of &= is evaluated before the assignment to Fx.

```
>&(fx with x=5), %, &float(%)
```

$$125 E^5$$

```
125e^5
```

18551.64488782208

```
>fx(5)
```

18551.6448878

For the evaluation of an expression with specific values of the variables, you can use the "with" operator.

The following command line also demonstrates that Maxima can evaluate an expression numerically with `float()`.

```
>&(fx with x=10)-(fx with x=5), &float(%)
```

$$1000 E^{10} - 125 E^5$$

2.20079141499189e+7

```
>factor(diff(fx,x,2))
```

$$x(x^2 + 6x + 6)e^x$$

To get the Latex code for an expression, you can use the `tex` command.

```
>tex(fx)
```

$$x^3 \backslash, e^{\{x\}}$$

Symbolic expressions can be evaluated just like numerical expressions.

```
>fx(0.5)
```

0.206090158838

In symbolic expressions, this does not work, since Maxima does not support it. Instead, use the "with" syntax (a nicer form of the `at(...)` command of Maxima).

```
>&fx with x=1/2
```

$$\sqrt{e_8}$$

The assignment can also be symbolic.

```
>&fx with x=1+t
```

$$(t+1)^3 e^t + 1$$

The command `solve` solves symbolic expressions for a variable in Maxima. The result is a vector of solutions.

```
>&solve(x^2+x=4,x)
```

$$\left[x = \frac{-\sqrt{17}-1}{2}, x = \frac{\sqrt{17}-1}{2} \right]$$

Compare with the numerical "solve" command in Euler, which needs a start value, and optionally a target value.

```
>solve("x^2+x",1,y=4)
```

1.56155281281

Nilai numerik dari solusi simbolik dapat dihitung dengan evaluasi hasil simbolik. Euler akan membacakan tugas `x=dst`. Jika Anda tidak memerlukan hasil numerik untuk perhitungan lebih lanjut, Anda juga dapat membiarkan Maxima menemukan nilai numeriknya.

```
>sol &= solve(x^2+2*x=4,x); $sol, sol(), $float(sol)
```

$$\left[x = -\sqrt{5}-1, x = \sqrt{5}-1 \right]$$

[-3.23607, 1.23607]

$$\left[x = -3.23606797749979, x = 1.23606797749979 \right]$$

To get a specific symbolic solution, one can use "with" and an index.

```
>&solve(x^2+x=1,x), x2 &= x with %[2]; $&x2
```

$$\left[x = \frac{-\sqrt{5}-1}{2}, x = \frac{\sqrt{5}-1}{2} \right]$$

$$\sqrt{5}-1_2$$

To solve a system of equations, use a vector of equations. The result is a vector of solutions.

```
>sol &= solve([x+y=3,x^2+y^2=5],[x,y]); $&sol, $&x*y with sol[1]
```

$$\left[\begin{matrix} x=2 \\ y=1 \end{matrix} \right], \left[\begin{matrix} x=1 \\ y=2 \end{matrix} \right]$$

$$2$$

Ekspresi simbolis dapat memiliki bendera, yang menunjukkan perlakuan khusus di Maxima. Beberapa flag dapat digunakan sebagai perintah juga, yang lainnya tidak. Bendera ditambahkan dengan "|" (bentuk yang lebih bagus dari "ev(...,flags)")

```
>& diff((x^3-1)/(x+1),x) //turunan bentuk pecahan
```

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

```
>& diff((x^3-1)/(x+1),x) | ratsimp //menyederhanakan pecahan
```

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

```
>&factor(%)
```

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

penyelesaian soal-soal

1. mencari faktor 10!

```
> ::factor(10!)
```

$$\begin{matrix} 8 & 4 & 2 \\ 2 & 3 & 5 & 7 \end{matrix}$$

2. mencari faktor 30!

```
> ::factor(30!)
```

$$\begin{matrix} 26 & 14 & 7 & 4 & 2 & 2 \\ 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 & 29 \end{matrix}$$

2. mencari faktor 30! 10!

```
> ::factor(30!)*(10!)
```

$$\begin{matrix} & & & & 2 & 2 \\ 16807 & 17179869184 & 387420489 & 1953125 & 11 & 13 & 17 & 19 & 23 & 29 \end{matrix}$$

Fungsi

Dalam EMT, fungsi adalah program yang didefinisikan dengan perintah "function. Ini bisa berupa fungsi satu baris atau fungsi multibaris.

Fungsi satu baris dapat berupa numerik atau simbolik. Fungsi satu baris numerik didefinisikan oleh ":=".


```
>function f(x) := x*sqrt(x^2+1)
```

For an overview, we show all possible definitions for one-line functions. A function can be evaluated just like any built-in Euler function.

```
>f(2)
```

```
4.472135955
```

This function will work for vectors too, obeying the matrix language of Euler, since the expressions used in the function are vectorized.

```
>f(0:0.1:1)
```

```
[0, 0.100499, 0.203961, 0.313209, 0.430813, 0.559017, 0.699714,
0.854459, 1.0245, 1.21083, 1.41421]
```

Functions can be plotted. Instead of expressions, we need only provide the function name.

In contrast to symbolic or numerical expressions, the function name must be provided in a string.

```
>solve("f",1,y=1)
```

```
0.786151377757
```

By default, if you need to overwrite a built-in function, you must add the keyword "overwrite". Overwriting built-in functions is dangerous and can cause problems for other functions depending on them.

You can still call the built-in function as "_...", if it is function in the Euler core.

```
>function overwrite sin (x) := _sin(x°) // redine sine in degrees
>sin(45)
```

```
0.707106781187
```

We better remove this redefinition of sin.

```
>forget sin; sin(pi/4)
```

```
0.706825181105
```

Default Parameters

Numerical function can have default parameters.

```
>function f(x,a=1) := a*x^2
```

Omitting this parameter uses the default value.

```
>f(4)
```

16

Setting it overwrites the default value.

```
>f(4,5)
```

80

An assigned parameter overwrite it too. This is used by many Euler functions like plot2d, plot3d.

```
>f(4,a=1)
```

16

If a variable is not a parameter, it must be global. One-line functions can see global variables.

```
>function f(x) := a*x^2  
>a=6; f(2)
```

24

But an assigned parameter overrides the global value.

If the argument is not in the list of pre-defined parameters, it must be declared with "!="

```
>f(2,a:=5)
```

20

Symbolic functions are defined with "&=". They are defined in Euler and Maxima, and work in both worlds. The defining expression is run through Maxima before the definition.

```
>function g(x) &= x^3-x*exp(-x); &g(x)
```

$x^3 - xe^{-x}$

Symbolic functions can be used in symbolic expressions.

```
>diff(g(x),x), % with x=4/3
```

$$xe^x - x - e^x - x + 3x^2$$

$$e^x - 4 \frac{1}{3 + \frac{16}{3}}$$

They can also be used in numerical expressions. Of course, this will only work if EMT can interpret everything inside the function.

```
>g(5+g(1))
```

178.635099908

They can be used to define other symbolic functions or expressions.

```
>function G(x) &= factor(integrate(g(x),x)); &G(c) // integrate: mengintegralkan
```

$$e^x - c (c^4 e^c + 4c + 4) \frac{1}{4}$$

```
>solve(&g(x),0.5)
```

0.703467422498

The following works too, since Euler uses the symbolic expression in the function g, if it does not find a symbolic variable g, and if there is a symbolic function g.

```
>solve(&g,0.5)
```

0.703467422498

```
>function P(x,n) &= (2*x-1)^n; &P(x,n)
```

$$(2x - 1)^n$$

```
>function Q(x,n)  &= (x+2)^n;  &Q(x,n)
```

$$(x + 2)^n$$

```
>$&P(x,4) ,  $&expand(%)
```

$$(2x - 1)^4$$

$$16x^4 - 32x^3 + 24x^2 - 8x + 1$$

```
>P(3,4)
```

$$625$$

```
>$&P(x,4) + Q(x,3) ,  $&expand(%)
```

$$(2x - 1)^4 + (x + 2)^3$$

$$16x^4 - 31x^3 + 30x^2 + 4x + 9$$

```
>$&P(x,4) - Q(x,3) ,  $&expand(%) ,  $&factor(%)
```

$$(2x - 1)^4 - (x + 2)^3$$

$$16x^4 - 33x^3 + 18x^2 - 20x - 7$$

$$16x^4 - 33x^3 + 18x^2 - 20x - 7$$

```
>$P(x,4)*Q(x,3), $expand(%), $factor(%)
```

$$(x + 2)^3 (2x - 1)^4$$

$$16x^7 + 64x^6 + 24x^5 - 120x^4 - 15x^3 + 102x^2 - 52x + 8$$

$$(x + 2)^3 (2x - 1)^4$$

```
>$P(x,4)/Q(x,1), $expand(%), $factor(%)
```

$$(2x - 1)^4 \frac{1}{x+2}$$

$$16x^4 \frac{1}{x+2} - \frac{32x^3}{x+2} + \frac{24x^2}{x+2} - 8$$

$$x \frac{1}{x+2} + \frac{1}{x+2}$$

$$(2x - 1)^4 \frac{1}{x+2}$$

```
>function f(x) &= x^3-x; &f(x)
```

x^3-x

With `&=` the function is symbolic, and can be used in other symbolic expressions.

```
>&integrate(f(x),x)
```

$x^4 \frac{1}{4 - \frac{x^2}{2}}$

With `:=` the function is numerical. A good example is a definite integral like

$f(x) = \int_0^x t^t dt,$

which can not be evaluated symbolically.

If we redefine the function with the keyword "map" it can be used for vectors `x`. Internally, the function is called for all values of `x` once, and the results are stored in a vector.

```
>function map f(x) := integrate("x^x",1,x)
>f(0:0.5:2)
```

```
[-0.783431, -0.410816, 0, 0.676863, 2.05045]
```

Functions can have default values for parameters.

```
>function mylog (x,base=10) := ln(x)/ln(base);
```

Now the function can be called with or without a parameter "base".

```
>mylog(100), mylog(2^6.7,2)
```

```
2
6.7
```

Moreover, it is possible to use assigned parameters.

```
>mylog(E^2,base=E)
```

2

Often, we want to use functions for vectors at one place, and for individual elements at other places. This is possible with vector parameters.

```
>function f([a,b]) &= a^2+b^2-a*b+b; $&f(a,b), $&f(x,y)
```

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

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

Such a symbolic function can be used for symbolic variables.
But the function can also be used for a numerical vector.

```
>v=[3,4]; f(v)
```

17

There are also purely symbolic functions, which cannot be used numerically.

```
>function lapl(expr,x,y) &= diff(expr,x,2)+diff(expr,y,2)//turunan parsial kedua
```

$\text{diff}(\text{expr}, y, 2) + \text{diff}(\text{expr}, x, 2)$

```
>$&realpart((x+I*y)^4), $&lapl(%,x,y)
```

$y^4-6x^2y^2+x^4$

0

Namun tentu saja, mereka dapat digunakan dalam ekspresi simbolik atau dalam definisi fungsi simbolik.

```
>function f(x,y) &= factor(lapl((x+y^2)^5,x,y)); &f(x,y)
```

$$10(y^2+x)^3(9y^2+x+2)$$

Untuk meringkas

- &= mendefinisikan fungsi simbolik,
- := mendefinisikan fungsi numerik,
- &&= mendefinisikan fungsi simbolik murni.

Memecahkan Ekspresi

Ekspresi dapat diselesaikan secara numerik dan simbolis.

Untuk menyelesaikan ekspresi sederhana dari satu variabel, kita dapat menggunakan fungsi `solve()`. Dibutuhkan nilai awal untuk memulai pencarian. Secara internal, `solve()` menggunakan metode secant.

```
>solve("x^2-2",1)
```

1.41421356237

This works for symbolic expression too. Take the following function.

```
>&solve(x^2=2,x)
```

$$\left[x = -\sqrt{2} , x = \sqrt{2} \right]$$

```
>&solve(x^2-2,x)
```

$$\left[x = -\sqrt{2} , x = \sqrt{2} \right]$$

```
>&solve(a*x^2+b*x+c=0,x)
```


$$\left[x = \frac{-\sqrt{b^2 - 4ac} - b}{2a}, x = \frac{-\sqrt{b^2 - 4ac} + b}{2a} \right]$$

```
> solve([a*x+b*y=c, d*x+e*y=f], [x, y])
```

$$\left[\left[x = -\frac{ce}{b(d-5) - ae}, y = \frac{c(d-5)}{b(d-5) - ae} \right] \right]$$

```
> px &= 4*x^8+x^7-x^4-x; &px
```

$$4x^8 + x^7 - x^4 - x$$

Sekarang kita mencari titik yang polinomialnya adalah 2. Dalam solve(), nilai target default y=0 dapat diubah dengan variabel yang ditetapkan.

Kami menggunakan y=2 dan memeriksa dengan mengevaluasi polinomial pada hasil sebelumnya.

```
> solve(px, 1, y=2), px(%)
```

```
0.966715594851
2
```

Solving a symbolic expression in symbolic form returns a list of solutions. We use the symbolic solver solve() provided by Maxima.

```
> sol &= solve(x^2-x-1, x); &sol
```

$$\left[x = \frac{1 - \sqrt{5}}{2}, x = \frac{\sqrt{5} + 1}{2} \right]$$

The easiest way to get the numerical values is to evaluate the solution numerically just like an expression.

```
> longest sol()
```

```
-0.6180339887498949      1.618033988749895
```

To use the solutions symbolically in other expressions, the easiest way is "with".

```
> $x^2 with sol[1], $expand(x^2-x-1 with sol[2])
```

$$\frac{(\sqrt{5} - 1)^2}{4}$$

0

Solving systems of equations symbolically can be done with vectors of equations and the symbolic solver `solve()`. The answer is a list of lists of equations.

```
>&solve([x+y=2, x^3+2*y+x=4], [x, y])
```

$$[[x = -1, y = 3], [x = 1, y = 1], [x = 0, y = 2]]$$

The function `f()` can see global variables. But often we want to use local parameters.

$$a^x - x^a = 0.1$$

with `a=3`.

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

One way to pass the additional parameter to `f()` is to use a list with the function name and the parameters (the other way are semicolon parameters).

```
>solve({{"f", 3}}, 2, y=0.1)
```

2.54116291558

This does also work with expressions. But then, a named list element has to be used. (More on lists in the tutorial about the syntax of EMT).

```
>solve({{"x^a-a^x", a=3}}, 2, y=0.1)
```

2.54116291558

Menyelesaikan Pertidaksamaan

Untuk menyelesaikan pertidaksamaan, EMT tidak akan dapat melakukannya, melainkan dengan bantuan Maxima, artinya secara eksak (simbolik). Perintah Maxima yang digunakan adalah `fourier_elim()`, yang harus dipanggil dengan perintah `"load(fourier_elim)"` terlebih dahulu.

```
>&load(fourier_elim)
```

```
C:/Program Files/Euler x64/maxima/share/maxima/5.35.1/share/f\
ourier_elim/fourier_elim.lisp
```

```
>&fourier_elim([x^2 - 1>0], [x]) // x^2-1 > 0
```

$$[1 < x] \vee [x < -1]$$

```
>&fourier_elim([x^2 - 1<0],[x]) // x^2-1 < 0
```

$$[-1 < x, x < 1]$$

```
>&fourier_elim([x^2 - 1 # 0],[x]) // x^-1 <> 0
```

$$[-1 < x, x < 1] \vee [1 < x] \vee [x < -1]$$

```
>&fourier_elim([x # 6],[x])
```

$$[x < 6] \vee [6 < x]$$

```
>&fourier_elim([x < 1, x > 1],[x]) // tidak memiliki penyelesaian
```

emptyset

```
>&fourier_elim([minf < x, x < inf],[x]) // solusinya R
```

universalset

```
>&fourier_elim([x^3 - 1 > 0],[x])
```

$$[1 < x, x^2 + x + 1 > 0] \vee [x < 1, -x^2 - x - 1 > 0]$$

```
>&fourier_elim([cos(x) < 1/2],[x]) // ??? gagal
```

$$[1 - 2 \cos x > 0]$$

```
>&fourier_elim([y-x < 5, x - y < 7, 10 < y],[x,y]) // sistem pertidaksamaan
```

$$[y - 5 < x, x < y + 7, 10 < y]$$

```
>&fourier_elim([y-x < 5, x - y < 7, 10 < y],[y,x])
```

$$[\max(10, x - 7) < y, y < x + 5, 5 < x]$$

```
>&fourier_elim((x + y < 5) and (x - y > 8), [x, y])
```

$$\left[y + 8 < x, x < 5 - y, y < -\frac{3}{2} \right]$$

```
>&fourier_elim(((x + y < 5) and x < 1) or (x - y > 8), [x, y])
```

$$[y + 8 < x] \vee [x < \min(1, 5 - y)]$$

```
>&fourier_elim([max(x, y) > 6, x # 8, abs(y-1) > 12], [x, y])
```

```
[6 < x, x < 8, y < - 11] or [8 < x, y < - 11]
or [x < 8, 13 < y] or [x = y, 13 < y] or [8 < x, x < y, 13 < y]
or [y < x, 13 < y]
```

```
>&fourier_elim([(x+6)/(x-9) <= 6], [x])
```

$$[x = 12] \vee [12 < x] \vee [x < 9]$$

Bahasa Matriks

Dokumentasi inti EMT berisi pembahasan rinci tentang bahasa matriks Euler.

Vektor dan matriks dimasukkan dengan tanda kurung siku, elemen dipisahkan dengan koma, baris dipisahkan dengan titik koma.

```
>A=[1,2;3,4]
```

1	2
3	4

Hasil kali matriks dilambangkan dengan titik.

```
>b=[3;4]
```

3
4

```
>b' // transpose b
```

[3, 4]

```
>inv(A) //inverse A
```

```
      -2      1  
1.5      -0.5
```

```
>A.b //perkalian matriks
```

```
11  
25
```

```
>A.inv(A)
```

```
      1      0  
      0      1
```

Poin utama dari bahasa matriks adalah semua fungsi dan operator bekerja elemen demi elemen.

```
>A.A
```

```
      7      10  
15      22
```

```
>A^2 //perpangkatan elemen2 A
```

```
      1      4  
      9      16
```

```
>A.A.A
```

```
      37      54  
      81      118
```

```
>power(A,3) //perpangkatan matriks
```

```
      37      54  
      81      118
```

```
>A/A //pembagian elemen-elemen matriks yang seletak
```

```
      1      1  
      1      1
```

```
>A/b //pembagian elemen2 A oleh elemen2 b kolom demi kolom (karena b vektor kolom)
```

0.333333	0.666667
0.75	1

```
>A\b // hasilkali invers A dan b, A(-1)b
```

-2
2.5

```
>inv(A) .b
```

-2
2.5

```
>A\A //A(-1)A
```

1	0
0	1

```
>inv(A) .A
```

1	0
0	1

```
>A*A //perkalin elemen-elemen matriks seletak
```

1	4
9	16

Ini bukan hasil kali matriks, melainkan perkalian elemen demi elemen. Hal yang sama juga berlaku untuk vektor.

```
>b^2 // perpangkatan elemen-elemen matriks/vektor
```

9
16

Jika salah satu operan adalah vektor atau skalar, maka operan tersebut diperluas secara alami.

```
>2*A
```

2	4
6	8

Misalnya, jika operan adalah vektor kolom, elemennya diterapkan ke semua baris A.

```
> [1,2] * A
```

1	4
3	8

Jika ini adalah vektor baris, maka diterapkan ke semua kolom A.

```
> A * [2,3]
```

2	6
6	12

Kita dapat membayangkan perkalian ini seolah-olah vektor baris v telah diduplikasi untuk membentuk matriks yang berukuran sama dengan A.

```
> dup([1,2],2) // dup: menduplikasi/menggandakan vektor [1,2] sebanyak 2 kali (baris)
```

1	2
1	2

```
> A * dup([1,2],2)
```

1	4
3	8

Hal ini juga berlaku untuk dua vektor dimana yang satu adalah vektor baris dan yang lainnya adalah vektor kolom. Kita menghitung $i*j$ untuk i,j dari 1 sampai 5. Caranya adalah dengan mengalikan 1:5 dengan transposenya. Bahasa matriks Euler secara otomatis menghasilkan tabel nilai.

```
> (1:5) * (1:5)' // hasilkali elemen-elemen vektor baris dan vektor kolom
```

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

Sekali lagi, ingatlah bahwa ini bukan produk matriks!

```
> (1:5) . (1:5)' // hasilkali vektor baris dan vektor kolom
```

55

```
> sum((1:5) * (1:5)) // sama hasilnya
```

55

Bahkan operator seperti $<$ atau $==$ bekerja dengan cara yang sama.

```
>(1:10)<6 // menguji elemen-elemen yang kurang dari 6
```

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

Misalnya, kita dapat menghitung jumlah elemen yang memenuhi kondisi tertentu dengan fungsi `sum()`.

```
>sum((1:10)<6) // banyak elemen yang kurang dari 6
```

```
5
```

Euler memiliki operator perbandingan, seperti "=", yang memeriksa kesetaraan.

Kita mendapatkan vektor 0 dan 1, dimana 1 berarti benar.

```
>t=(1:10)^2; t==25 //menguji elemen2 t yang sama dengan 25 (hanya ada 1)
```

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

Dari vektor tersebut, "bukan nol" memilih elemen bukan nol.

Dalam hal ini, kita mendapatkan indeks semua elemen lebih besar dari 50.

```
>nonzeros(t>50) //indeks elemen2 t yang lebih besar daripada 50
```

```
[8, 9, 10]
```

Tentu saja, kita dapat menggunakan vektor indeks ini untuk mendapatkan nilai yang sesuai dalam `t`.

```
>t[nonzeros(t>50)] //elemen2 t yang lebih besar daripada 50
```

```
[64, 81, 100]
```

As an example, let us find all squares of the numbers 1 to 1000, which are 5 modulo 11 and 3 modulo 13.

```
>t=1:1000; nonzeros(mod(t^2,11)==5 && mod(t^2,13)==3)
```

```
[4, 48, 95, 139, 147, 191, 238, 282, 290, 334, 381, 425,  
433, 477, 524, 568, 576, 620, 667, 711, 719, 763, 810, 854,  
862, 906, 953, 997]
```

EMT is not completely effective for integer computations. It uses double precision floating point internally. However, it is often very useful.

We can check for primality. Let us find out, how many squares plus 1 are primes.

```
>t=1:1000; length(nonzeros(isprime(t^2+1)))
```

```
112
```

The function `nonzeros()` works only for vectors. For matrices, there is `mnonzeros()`.


```
>seed(2); A=random(3,4)
```

0.765761	0.401188	0.406347	0.267829
0.13673	0.390567	0.495975	0.952814
0.548138	0.006085	0.444255	0.539246

It returns the indices of the elements, which are not zeros.

```
>k=mnnonzeros(A<0.4) //indeks elemen2 A yang kurang dari 0,4
```

1	4
2	1
2	2
3	2

These indices can be used to set the elements to some value.

```
>mset(A,k,0) //mengganti elemen2 suatu matriks pada indeks tertentu
```

0.765761	0.401188	0.406347	0
0	0	0.495975	0.952814
0.548138	0	0.444255	0.539246

The function `mset()` can also set the elements at the indices to the entries of some other matrix.

```
>mset(A,k,-random(size(A)))
```

0.765761	0.401188	0.406347	-0.126917
-0.122404	-0.691673	0.495975	0.952814
0.548138	-0.483902	0.444255	0.539246

And it is possible to get the elements in a vector.

```
>mget(A,k)
```

```
[0.267829, 0.13673, 0.390567, 0.006085]
```

Another useful function is `extrema`, which returns the minimal and maximal values in each row of the matrix and their positions.

```
>ex=extrema(A)
```

0.267829	4	0.765761	1
0.13673	1	0.952814	4
0.006085	2	0.548138	1

We can use this to extract the maximal values in each row.

```
>ex[:,3]'
```

```
[0.765761, 0.952814, 0.548138]
```

This, of course, is the same as the function `max()`.

```
>max(A)'
```

```
[0.765761, 0.952814, 0.548138]
```

But with `mget()`, we can extract the indices and use this information to extract the elements at the same positions from another matrix.

```
>j=(1:rows(A))' | ex[,4], mget(-A,j)
```

```
      1      1  
      2      4  
      3      1  
[-0.765761, -0.952814, -0.548138]
```

Fungsi Matriks Lainnya (Matriks Bangunan)

Untuk membangun sebuah matriks, kita dapat menumpuk satu matriks di atas matriks lainnya. Jika keduanya tidak memiliki jumlah kolom yang sama, maka kolom yang lebih pendek akan diisi dengan 0.

```
>v=1:3; v_v
```

```
      1      2      3  
      1      2      3
```

Demikian pula, kita dapat melampirkan matriks ke matriks lain secara berdampingan, jika keduanya mempunyai jumlah baris yang sama.

```
>A=random(3,4); A|v'
```

```
0.032444 0.0534171 0.595713 0.564454 1  
0.83916 0.175552 0.396988 0.83514 2  
0.0257573 0.658585 0.629832 0.770895 3
```

Jika jumlah barisnya tidak sama, matriks yang lebih pendek diisi dengan 0.

Ada pengecualian untuk aturan ini. Bilangan real yang melekat pada suatu matriks akan digunakan sebagai kolom yang diisi dengan bilangan real tersebut.

```
>A|1
```

```
0.032444 0.0534171 0.595713 0.564454 1  
0.83916 0.175552 0.396988 0.83514 1  
0.0257573 0.658585 0.629832 0.770895 1
```

It is possible to make a matrix of row and column vectors.

```
>[v;v]
```

1	2	3
1	2	3

```
>[v',v']
```

1	1
2	2
3	3

The main purpose of this is to interpret a vector of expressions for column vectors.

```
>"[x,x^2]"(v')
```

1	1
2	4
3	9

To get the size of A, we can use the following functions.

```
>C=zeros(2,4); rows(C), cols(C), size(C), length(C)
```

```
2
4
[2, 4]
4
```

For vectors, there is length().

```
>length(2:10)
```

```
9
```

There are many other functions, which generate matrices.

```
>ones(2,2)
```

1	1
1	1

Ini juga dapat digunakan dengan satu parameter. Untuk mendapatkan vektor dengan bilangan selain 1, gunakan yang berikut ini.

```
>ones(5)*6
```

```
[6, 6, 6, 6, 6]
```

Matriks bilangan acak juga dapat dihasilkan dengan acak (distribusi seragam) atau normal (distribusi Gauß).

```
>random(2,2)
```

0.66566	0.831835
0.977	0.544258

Berikut adalah fungsi lain yang berguna, yang merestrukturisasi elemen matriks menjadi matriks lain.

```
>redim(1:9,3,3) // menyusun elemen2 1, 2, 3, ..., 9 ke bentuk matriks 3x3
```

1	2	3
4	5	6
7	8	9

Dengan fungsi berikut, kita dapat menggunakan fungsi ini dan fungsi dup untuk menulis fungsi rep(), yang mengulangi vektor sebanyak n kali.

```
>function rep(v,n) := redim(dup(v,n),1,n*cols(v))
```

Let us test.

```
>rep(1:3,5)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

The function multdup() duplicates elements of a vector.

```
>multdup(1:3,5), multdup(1:3,[2,3,2])
```

```
[1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]
[1, 1, 2, 2, 2, 3, 3]
```

The functions flipx() and flipy() revert the order of the rows or columns of a matrix. I.e., the function flipx() flips horizontally.

```
>flipx(1:5) //membalik elemen2 vektor baris
```

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

For rotations, Euler has rotright() and rotright().

```
>rotright(1:5) // memutar elemen2 vektor baris
```

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

A special function is drop(v,i), which removes the elements with the indices in i from the vector v.

```
>drop(10:20,3)
```

```
[10, 11, 13, 14, 15, 16, 17, 18, 19, 20]
```

Perhatikan bahwa vektor i di drop(v,i) mengacu pada indeks elemen di v, bukan nilai elemen. Jika Anda ingin menghapus elemen, Anda perlu mencari elemennya terlebih dahulu. Fungsi indexof(v,x) dapat digunakan untuk mencari elemen x dalam vektor yang diurutkan v.

```
>v=primes(50), i=indexof(v,10:20), drop(v,i)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
[0, 5, 0, 6, 0, 0, 0, 7, 0, 8, 0]
[2, 3, 5, 7, 23, 29, 31, 37, 41, 43, 47]
```

Seperti yang Anda lihat, tidak ada salahnya memasukkan indeks di luar rentang (seperti 0), indeks ganda, atau indeks yang tidak diurutkan.

```
>drop(1:10,shuffle([0,0,5,5,7,12,12]))
```

```
[1, 2, 3, 4, 6, 8, 9, 10]
```

There are some special functions to set diagonals or to generate a diagonal matrix.
We start with the identity matrix.

```
>A=id(5) // matriks identitas 5x5
```

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Then we set the lower diagonal (-1) to 1:4.

```
>setdiag(A,-1,1:4) //mengganti diagonal di bawah diagonal utama
```

1	0	0	0	0
1	1	0	0	0
0	2	1	0	0
0	0	3	1	0
0	0	0	4	1

Note that we did not change the matrix A. We get a new matrix as result of setdiag().
Here is a function, which returns a tri-diagonal matrix.

```
>function tridiag (n,a,b,c) := setdiag(setdiag(b*id(n),1,c),-1,a); ...
>tridiag(5,1,2,3)
```

2	3	0	0	0
1	2	3	0	0
0	1	2	3	0
0	0	1	2	3
0	0	0	1	2

The diagonal of a matrix can also be extracted from the matrix. To demonstrate this, we restructure the vector 1:9 to a 3x3 matrix.

```
>A=redim(1:9,3,3)
```

1	2	3
4	5	6
7	8	9

Now we can extract the diagonal.

```
>d=getdiag(A,0)
```

```
[1, 5, 9]
```

E.g. We can divide the matrix by its diagonal. The matrix language takes care that the column vector d is applied to the matrix row by row.

```
>fraction A/d'
```

1	2	3
4/5	1	6/5
7/9	8/9	1

Vektorisasi

Hampir semua fungsi di Euler juga berfungsi untuk input matriks dan vektor, jika hal ini masuk akal. Misalnya, fungsi `sqrt()` menghitung akar kuadrat dari semua elemen vektor atau matriks.

```
>sqrt(1:3)
```

```
[1, 1.41421, 1.73205]
```

So you can easily create a table of values. This is one way to plot a function (the alternative uses an expression).

```
>x=1:0.01:5; y=log(x)/x^2; // terlalu panjang untuk ditampilkan
```

Dengan ini dan operator titik dua $a:\text{delta}:b$, vektor nilai fungsi dapat dihasilkan dengan mudah.

Pada contoh berikut, kita menghasilkan vektor nilai $t[i]$ dengan jarak 0,1 dari -1 hingga 1. Kemudian kita menghasilkan vektor nilai fungsi

lateks: $s = t^3 - t$ Dengan ini dan operator titik dua $a:\text{delta}:b$, vektor nilai fungsi dapat dihasilkan dengan mudah.

Pada contoh berikut, kita menghasilkan vektor nilai $t[i]$ dengan jarak 0,1 dari -1 hingga 1. Kemudian kita menghasilkan vektor nilai fungsi

lateks: $s = t^3 - t$

```
>t=-1:0.1:1; s=t^3-t
```

```
[0, 0.171, 0.288, 0.357, 0.384, 0.375, 0.336, 0.273, 0.192,
0.099, 0, -0.099, -0.192, -0.273, -0.336, -0.375, -0.384,
-0.357, -0.288, -0.171, 0]
```

EMT memperluas operator untuk skalar, vektor, dan matriks dengan cara yang jelas.

Misalnya, vektor kolom dikali vektor baris diperluas ke matriks, jika operator diterapkan. Berikut ini, v' adalah vektor yang dialihkan (vektor kolom).

```
>shortest (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

Note, that this is quite different from the matrix product. The matrix product is denoted with a dot "." in EMT.

```
>(1:5).(1:5)'
```

55

By default, row vectors are printed in a compact format.

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

[1, 2, 3, 4]

For matrices the special operator . denotes matrix multiplication, and A' denotes transposing. A 1x1 matrix can be used just like a real number.

```
>v:=[1,2]; v.v', %^2
```

5
25

To transpose a matrix we use the apostrophe.

```
>v=1:4; v'
```

1
2
3
4

So we can compute matrix A times vector b.

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

30
70

Note that v is still a row vector. So v'.v is different from v.v'.

```
>v'.v
```

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

$v \cdot v'$ computes the norm of v squared for row vectors v . The result is a 1x1 vector, which works just like a real number.

```
>v.v'
```

30

There is also the function `norm` (along with many other function of Linear Algebra).

```
>norm(v)^2
```

30

Operator dan fungsi mematuhi bahasa matriks Euler.

Berikut ringkasan peraturannya.

- Suatu fungsi yang diterapkan pada vektor atau matriks diterapkan pada setiap elemen.
- Operator yang mengoperasikan dua matriks dengan ukuran yang sama diterapkan secara berpasangan pada elemen-elemen matriks.
- Jika kedua matriks mempunyai dimensi yang berbeda, keduanya diekspansi secara wajar sehingga mempunyai ukuran yang sama.

Misalnya, nilai skalar dikalikan vektor dengan mengalikan nilai setiap elemen vektor. Atau matriks dikalikan vektor (dengan `*`, bukan `.`) memperluas vektor ke ukuran matriks dengan menduplikasinya.

Berikut ini adalah kasus sederhana dengan operator `^`.

```
>[1,2,3]^2
```

[1, 4, 9]

Here is a more complicated case. A row vector times a column vector expands both by duplicating.

```
>v:=[1,2,3]; v*v'
```

1	2	3
2	4	6
3	6	9

Note that the scalar product uses the matrix product, not the `*`!

```
>v.v'
```

14

Ada banyak fungsi matriks. Kami memberikan daftar singkat. Anda harus membaca dokumentasi untuk informasi lebih lanjut tentang perintah ini.

```
sum,prod menghitung jumlah dan hasil kali baris
cumsum,cumprod melakukan hal yang sama secara kumulatif
menghitung nilai ekstrem setiap baris
extreme mengembalikan vektor dengan informasi ekstrem
diag(A,i) mengembalikan diagonal ke-i
setdiag(A,i,v) menyetel diagonal ke-i
id(n) matriks identitas
det(A) determinannya
charpoly(A) polinomial karakteristik
nilai eigen(A) nilai eigen
```



```
>v*v, sum(v*v), cumsum(v*v)
```

```
[1, 4, 9]
14
[1, 5, 14]
```

The `:` operator generates an equally spaced row vector, optionally with a step size.

```
>1:4, 1:2:10
```

```
[1, 2, 3, 4]
[1, 3, 5, 7, 9]
```

To concatenate matrices and vectors there are the operators `"|"` and `"_"`.

```
>[1,2,3]|[4,5], [1,2,3]_1
```

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

The elements of a matrix are referred with `"A[i,j]"`.

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

```
6
```

For row or column vectors, `v[i]` is the *i*-th element of the vector. For matrices, this returns the complete *i*-th row of the matrix.

```
>v:=[2,4,6,8]; v[3], A[3]
```

```
6
[7, 8, 9]
```

The indices can also be row vectors of indices. `:` denotes all indices.

```
>v[1:2], A[:,2]
```

```
[2, 4]
      2
      5
      8
```

A short form for `:` is omitting the index completely.

```
>A[,2:3]
```

```
      2      3
      5      6
      8      9
```

Untuk tujuan vektorisasi, elemen matriks dapat diakses seolah-olah elemen tersebut adalah vektor.

```
>A{4}
```

4

Matriks juga dapat diratakan menggunakan fungsi `redim()`. Ini diimplementasikan dalam fungsi `flatten()`.

```
>redim(A,1,prod(size(A))), flatten(A)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Untuk menggunakan matriks pada tabel, mari kita atur ulang ke format default, dan hitung tabel nilai sinus dan kosinus. Perhatikan bahwa sudut dinyatakan dalam radian secara default.

```
>deformat; w=0°:45°:360°; w=w'; deg(w)
```

```
0
45
90
135
180
225
270
315
360
```

Now we append columns to a matrix.

```
>M = deg(w) |w|cos(w) |sin(w)
```

0	0	1	0
45	0.785398	0.707107	0.707107
90	1.5708	0	1
135	2.35619	-0.707107	0.707107
180	3.14159	-1	0
225	3.92699	-0.707107	-0.707107
270	4.71239	0	-1
315	5.49779	0.707107	-0.707107
360	6.28319	1	0

Dengan menggunakan bahasa matriks, kita dapat menghasilkan beberapa tabel dari beberapa fungsi sekaligus.

Dalam contoh berikut, kita menghitung t_j^i untuk i dari 1 hingga n . Kita mendapatkan sebuah matriks, yang setiap barisnya merupakan tabel t^i untuk satu i . Yaitu, matriks memiliki elemen lateks: $a_{\{i,j\}} = t_j^i$, $\quad 1 \leq j \leq 101, \quad 1 \leq i \leq n$

Fungsi yang tidak berfungsi untuk masukan vektor harus "divektorkan". Hal ini dapat dicapai dengan kata kunci "peta" dalam definisi fungsi. Kemudian fungsi tersebut akan dievaluasi untuk setiap elemen parameter vektor.

Integrasi numerik `integral()` hanya berfungsi untuk batas interval skalar. Jadi kita perlu membuat vektorisasinya.

```
>function map f(x) := integrate("x^x",1,x)
```

Kata kunci "map" membuat vektorisasi fungsi tersebut. Fungsinya sekarang akan berfungsi untuk vektor bilangan.

```
>f([1:5])
```

```
[0, 2.05045, 13.7251, 113.336, 1241.03]
```

Sub-Matrices and Matrix-Elements

To access a matrix element, use the bracket notation.

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

	1	2	3
	4	5	6
	7	8	9
5			

We can access a complete line of a matrix.

```
>A[2]
```

```
[4, 5, 6]
```

In case of row or column vectors, this returns an element of the vector.

```
>v=1:3; v[2]
```

```
2
```

Untuk memastikan, Anda mendapatkan baris pertama untuk matriks 1xn dan mxn, tentukan semua kolom menggunakan indeks kedua yang kosong.

```
>A[2,]
```

```
[4, 5, 6]
```

Jika indeks adalah vektor dari indeks, Euler akan mengembalikan baris matriks yang sesuai. Di sini kita menginginkan baris pertama dan kedua A.

```
>A[[1,2]]
```

1	2	3
4	5	6

Kita bahkan dapat menyusun ulang A menggunakan vektor indeks. Tepatnya, kita tidak mengubah A di sini, namun menghitung versi A yang disusun ulang.

```
>A[[3,2,1]]
```

7	8	9
4	5	6
1	2	3

Trik indeks juga berfungsi dengan kolom.

Contoh ini memilih semua baris A dan kolom kedua dan ketiga.

```
>A[1:3,2:3]
```

2	3
5	6
8	9

Untuk singkatan ":" menunjukkan semua indeks baris atau kolom.

```
>A[:,3]
```

3
6
9

Alternatifnya, biarkan indeks pertama kosong.

```
>A[,2:3]
```

2	3
5	6
8	9

Kita juga bisa mendapatkan baris terakhir A.

```
>A[-1]
```

```
[7, 8, 9]
```

Sekarang mari kita ubah elemen A dengan menetapkan submatriks A ke beberapa nilai. Ini sebenarnya mengubah matriks A yang disimpan.

```
>A[1,1]=4
```

4	2	3
4	5	6
7	8	9

Kita juga dapat memberikan nilai pada baris A.

```
>A[1]=[-1,-1,-1]
```

-1	-1	-1
4	5	6
7	8	9

We can even assign to a sub-matrix if it has the proper size.

```
>A[1:2,1:2]=[5,6;7,8]
```

5	6	-1
7	8	6
7	8	9

Selain itu, beberapa jalan pintas diperbolehkan.

```
>A[1:2,1:2]=0
```

0	0	-1
0	0	6
7	8	9

Peringatan: Indeks di luar batas mengembalikan matriks kosong, atau pesan kesalahan, bergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Namun perlu diingat bahwa indeks negatif dapat digunakan untuk mengakses elemen matriks yang dihitung dari akhir.

```
>A[4]
```

```
Row index 4 out of bounds!  
Error in:  
A[4] ...  
      ^
```

Menyortir dan Mengacak

Fungsi `sort()` mengurutkan vektor baris.

```
>sort([5,6,4,8,1,9])
```

```
[1, 4, 5, 6, 8, 9]
```

Seringkali perlu mengetahui indeks vektor yang diurutkan dalam vektor aslinya. Ini dapat digunakan untuk menyusun ulang vektor lain dengan cara yang sama.

Mari kita mengacak sebuah vektor.

```
>v=shuffle(1:10)
```

```
[4, 5, 10, 6, 8, 9, 1, 7, 2, 3]
```

Indeks berisi urutan v.

```
>{vs,ind}=sort(v); v[ind]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Ini juga berfungsi untuk vektor string.

```
>s=["a","d","e","a","aa","e"]
```

```
a
d
e
a
aa
e
```

```
>{ss,ind}=sort(s); ss
```

```
a
a
aa
d
e
e
```

Seperti yang Anda lihat, posisi entri ganda agak acak.

```
>ind
```

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

Fungsi unik mengembalikan daftar elemen unik vektor yang diurutkan.

```
>intrandom(1,10,10), unique(%)
```

```
[4, 4, 9, 2, 6, 5, 10, 6, 5, 1]
[1, 2, 4, 5, 6, 9, 10]
```

Ini juga berfungsi untuk vektor string.

```
>unique(s)
```

```
a
aa
d
e
```

Aljabar linier

EMT memiliki banyak sekali fungsi untuk menyelesaikan masalah sistem linier, sistem sparse, atau regresi. Untuk sistem linier $Ax=b$, Anda dapat menggunakan algoritma Gauss, matriks invers, atau linear fit. Operator $A \backslash b$ menggunakan versi algoritma Gauss.

```
>A=[1,2;3,4]; b=[5;6]; A\b
```

```
-4
4.5
```

Contoh lain, kita membuat matriks berukuran 200x200 dan jumlah baris-barisnya. Kemudian kita selesaikan $Ax=b$ menggunakan matriks invers. Kami mengukur kesalahan sebagai deviasi maksimal semua elemen dari 1, yang tentu saja merupakan solusi yang tepat.

```
>A=normal(200,200); b=sum(A); longest totalmax(abs(inv(A).b-1))
```

8.790745908981989e-13

Jika sistem tidak mempunyai solusi, kecocokan linier meminimalkan norma kesalahan $Ax=b$.

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

1	2	3
4	5	6
7	8	9

Penentu matriks ini adalah 0.

```
>det(A)
```

0

Matriks Simbolik

Maxima memiliki matriks simbolik. Tentu saja Maxima dapat digunakan untuk permasalahan aljabar linier sederhana seperti itu. Kita dapat mendefinisikan matriks untuk Euler dan Maxima dengan $\&:=$, lalu menggunakannya dalam ekspresi simbolik. Bentuk [...] yang biasa untuk mendefinisikan matriks dapat digunakan di Euler untuk mendefinisikan matriks simbolik.

```
>A &:= [a,1,1;1,a,1;1,1,a]; A
```

$$\begin{pmatrix} a & 1 & 1 \\ 1 & a & 1 \\ 1 & 1 & a \end{pmatrix}$$

```
>$det(A), $factor(%)
```

$$a(a^2 - 1) - 2a + 2$$

$$(a - 1)^2(a + 2)$$

```
>&invert(A) with a=0
```

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

```
>A = [1,a;b,2]; A
```

$$\begin{pmatrix} 1 & a \\ b & 2 \end{pmatrix}$$

Seperti semua variabel simbolik, matriks ini dapat digunakan dalam ekspresi simbolik lainnya.

```
>$det(A-x*ident(2)), $solve(%,x)
```

$$(1-x)(2-x)-ab$$

$$\left[x = \frac{3 - \sqrt{4ab+1}}{2}, x = \frac{\sqrt{4ab+1}+3}{2} \right]$$

Nilai eigen juga dapat dihitung secara otomatis. Hasilnya adalah sebuah vektor dengan dua vektor nilai eigen dan multiplisitas.

```
>eigenvalues([a,1;1,a])
```

$$[[a-1, a+1], [1, 1]]$$

Untuk mengekstrak vektor eigen tertentu memerlukan pengindeksan yang cermat.

```
>eigenvectors([a,1;1,a]), &%[2][1][1]
```

$$[[[a-1, a+1], [1, 1]], [[[1, -1]], [[1, 1]]]]$$

$$[1, -1]$$

Symbolic matrices can be evaluated in Euler numerically just like other symbolic expressions.

```
>A(a=4,b=5)
```

$$\begin{pmatrix} 1 & 4 \\ 5 & 2 \end{pmatrix}$$

In symbolic expressions, use with.

```
>&A with [a=4,b=5]
```

$$\begin{pmatrix} 1 & 4 \\ 5 & 2 \end{pmatrix}$$

Access to rows of symbolic matrices work just like with numerical matrices.

```
>&A[1]
```

$$[1, a]$$

A symbolic expression can contain an assignment. And that changes the matrix A.

```
>&A[1,1]:=t+1; &A
```

$$\begin{pmatrix} t+1 & a \\ b & 2 \end{pmatrix}$$

Ada fungsi simbolik di Maxima untuk membuat vektor dan matriks. Untuk ini, lihat dokumentasi Maxima atau tutorial tentang Maxima di EMT.

```
>v &= makelist(1/(i+j),i,1,3); v
```

$$\left[\frac{1}{j+1}, \frac{1}{j+2}, \frac{1}{j+3} \right]$$

```
>B &:= [1,2;3,4]; $B, $&invert(B)
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$
$$\begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

Hasilnya dapat dievaluasi secara numerik dalam Euler. Untuk informasi lebih lanjut tentang Maxima, lihat pengenalan Maxima.

```
>&invert(B)()
```

$$\begin{matrix} -2 & 1 \\ 1.5 & -0.5 \end{matrix}$$

Euler juga memiliki fungsi kuat `xinv()`, yang melakukan upaya lebih besar dan mendapatkan hasil yang lebih tepat.

Perhatikan, bahwa dengan `&:=` matriks B telah didefinisikan sebagai simbolik dalam ekspresi simbolik dan numerik dalam ekspresi numerik. Jadi kita bisa menggunakannya di sini.

```
>longest B.xinv(B)
```

$$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

E.g. Misalnya. nilai eigen dari A dapat dihitung secara numerik.

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

$$[16.1168, -1.11684, 0]$$

Atau secara simbolis. Lihat tutorial tentang Maxima untuk detailnya.

```
>&eigenvalues (@A)
```

$$\left[\left[\frac{15 - 3\sqrt{33}}{2}, \frac{3\sqrt{33} + 15}{2}, 0 \right], [1, 1, 1] \right]$$

Nilai Numerik dalam Ekspresi simbolik

Ekspresi simbolis hanyalah string yang berisi ekspresi. Jika kita ingin mendefinisikan nilai untuk ekspresi simbolik dan ekspresi numerik, kita harus menggunakan "&:=".

```
>A &:= [1,pi;4,5]
```

$$\begin{array}{cc} 1 & 3.14 \\ 4 & 5 \end{array}$$

Masih terdapat perbedaan antara bentuk numerik dan simbolik. Saat mentransfer matriks ke bentuk simbolik, pendekatan pecahan untuk real akan digunakan.

```
>&A
```

$$\begin{pmatrix} 1 & \frac{157}{50} \\ 4 & 5 \end{pmatrix}$$

Untuk menghindari hal ini, ada fungsi "mxmset(variabel)".

```
>mxmset (A) ; &A
```

$$\begin{pmatrix} 1 & 3.14 \\ 4 & 5 \end{pmatrix}$$

Maxima juga dapat menghitung dengan bilangan floating point, bahkan dengan bilangan mengambang besar dengan 32 digit. Namun evaluasinya jauh lebih lambat.

```
>$&bfloat(sqrt(2)), $&float(sqrt(2))
```

$$1.4142135623730950488016887242097_B \times 10^0$$

$$1.414213562373095$$

Ketepatan angka floating point besar dapat diubah.

```
>&fpprec:=100; &bfloat(pi)
```

$$3.14159265358979323846264338327950288419716939937510582097494 \backslash \\ 4592307816406286208998628034825342117068b0$$

Variabel numerik dapat digunakan dalam ekspresi simbolik apa pun menggunakan "@var". Perhatikan bahwa ini hanya diperlukan, jika variabel telah didefinisikan dengan "==" atau "=" sebagai variabel numerik.

```
>B:=[1,pi;3,4]; &det (@B)
```

−5.42

Demo - Suku Bunga

Di bawah ini, kami menggunakan Euler Math Toolbox (EMT) untuk menghitung suku bunga. Kami melakukannya secara numerik dan simbolis untuk menunjukkan kepada Anda bagaimana Euler dapat digunakan untuk memecahkan masalah kehidupan nyata.

Asumsikan Anda memiliki modal awal sebesar 5.000 (katakanlah dalam dolar).

```
>K=5000
```

5000

Sekarang kami mengasumsikan tingkat bunga 3% per tahun. Mari kita tambahkan satu tarif sederhana dan hitung hasilnya.

```
>K*1.03
```

5150

Euler juga akan memahami sintaks berikut.

```
>K+K*3%
```

5150

Namun lebih mudah menggunakan faktor tersebut

```
>q=1+3%, K*q
```

1.03

5150

Selama 10 tahun, kita cukup mengalikan faktor-faktornya dan mendapatkan nilai akhir dengan tingkat bunga majemuk.

```
>K*q^10
```

6719.58189672

Untuk keperluan kita, kita dapat mengatur formatnya menjadi 2 digit setelah titik desimal.

```
>format(12,2); K*q^10
```

6719.58

Mari kita cetak yang dibulatkan menjadi 2 digit dalam satu kalimat lengkap.

```
>"Starting from " + K + "$ you get " + round(K*q^10,2) + "$."
```

Starting from 5000\$ you get 6719.58\$.

Bagaimana jika kita ingin mengetahui hasil antara dari tahun 1 sampai tahun ke 9? Untuk ini, bahasa matriks Euler sangat membantu. Anda tidak perlu menulis satu perulangan, tetapi cukup masuk

```
>K*q^(0:10)
```

Real 1 x 11 matrix

5000.00 5150.00 5304.50 5463.64 ...

Bagaimana keajaiban ini terjadi? Pertama, ekspresi 0:10 mengembalikan vektor bilangan bulat.

```
>short 0:10
```

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Kemudian semua operator dan fungsi di Euler dapat diterapkan pada vektor elemen demi elemen. Jadi

```
>short q^(0:10)
```

[1, 1.03, 1.0609, 1.0927, 1.1255, 1.1593, 1.1941, 1.2299,
1.2668, 1.3048, 1.3439]

adalah vektor faktor q^0 sampai q^{10} . Ini dikalikan dengan K, dan kita mendapatkan vektor nilainya.

```
>VK=K*q^(0:10);
```

Tentu saja, cara realistis untuk menghitung suku bunga ini adalah dengan membulatkan ke sen terdekat setiap tahunnya. Mari kita tambahkan fungsi untuk ini.

```
>function oneyear (K) := round(K*q,2)
```

Mari kita bandingkan kedua hasil tersebut, dengan dan tanpa pembulatan.

```
>longest oneyear(1234.57), longest 1234.57*q
```

1271.61
1271.6071

Sekarang tidak ada rumus sederhana untuk tahun ke-n, dan kita harus mengulanginya selama bertahun-tahun. Euler memberikan banyak solusi untuk ini.

Cara termudah adalah fungsi iterate, yang mengulangi fungsi tertentu beberapa kali.

```
>VKr=iterate("oneyear",5000,10)
```

Real 1 x 11 matrix

```
5000.00    5150.00    5304.50    5463.64    ...
```

Kami dapat mencetaknya dengan cara yang ramah, menggunakan format kami dengan tempat desimal tetap.

```
>VKr'
```

```
5000.00
5150.00
5304.50
5463.64
5627.55
5796.38
5970.27
6149.38
6333.86
6523.88
6719.60
```

Untuk mendapatkan elemen vektor tertentu, kami menggunakan indeks dalam tanda kurung siku.

```
>VKr[2], VKr[1:3]
```

```
5150.00
5000.00    5150.00    5304.50
```

Anehnya, kita juga bisa menggunakan vektor indeks. Ingatlah bahwa 1:3 menghasilkan vektor [1,2,3].

Mari kita bandingkan elemen terakhir dari nilai yang dibulatkan dengan nilai penuh.

```
>VKr[-1], VK[-1]
```

```
6719.60
6719.58
```

Perbedaannya sangat kecil. **Memecahkan Persamaan**

Sekarang kita mengambil fungsi yang lebih maju, yang menambahkan tingkat uang tertentu setiap tahunnya.

```
>function onepay (K) := K*q+R
```

Kita tidak perlu menentukan q atau R untuk definisi fungsi. Hanya jika kita menjalankan perintah, kita harus mendefinisikan nilai-nilai ini. Kami memilih $R=200$.

```
>R=200; iterate("onipay",5000,10)
```

```
Real 1 x 11 matrix
      5000.00      5350.00      5710.50      6081.82      ...
```

Bagaimana jika kita menghapus jumlah yang sama setiap tahun?

```
>R=-200; iterate("onipay",5000,10)
```

```
Real 1 x 11 matrix
      5000.00      4950.00      4898.50      4845.45      ...
```

Kami melihat uangnya berkurang. Jelasnya, jika kita hanya mendapat bunga sebesar 150 pada tahun pertama, namun menghapus 200, kita kehilangan uang setiap tahunnya.

Bagaimana kita dapat menentukan berapa tahun uang tersebut akan bertahan? Kita harus menulis satu lingkaran untuk ini. Cara termudah adalah dengan melakukan iterasi cukup lama.

```
>VKR=iterate("onipay",5000,50)
```

```
Real 1 x 51 matrix
      5000.00      4950.00      4898.50      4845.45      ...
```

Dengan menggunakan bahasa matriks, kita dapat menentukan nilai negatif pertama dengan cara berikut.

```
>min(nonzeros(VKR<0))
```

```
48.00
```

Alasannya adalah bukan nol ($VKR<0$) mengembalikan vektor indeks i , dengan $VKR[i]<0$, dan \min menghitung indeks minimal.

Karena vektor selalu dimulai dengan indeks 1, maka jawabannya adalah 47 tahun.

Fungsi `iterate()` memiliki satu trik lagi. Ini dapat mengambil kondisi akhir sebagai argumen. Kemudian akan mengembalikan nilai dan jumlah iterasi.

```
>{x,n}=iterate("onipay",5000,till="x<0"); x, n,
```

```
-19.83
47.00
```

Mari kita coba menjawab pertanyaan yang lebih ambigu. Asumsikan kita mengetahui bahwa nilainya adalah 0 setelah 50 tahun. Berapa tingkat bunganya?

Ini adalah pertanyaan yang hanya bisa dijawab secara numerik. Di bawah ini, kita akan mendapatkan rumus yang diperlukan. Kemudian Anda akan melihat bahwa tidak ada rumus yang mudah untuk menentukan tingkat suku bunga. Namun untuk saat ini, kami menargetkan solusi numerik.

Langkah pertama adalah mendefinisikan fungsi yang melakukan iterasi sebanyak n kali. Kami menambahkan semua parameter ke fungsi ini.

```
>function f(K,R,P,n) := iterate("x*(1+P/100)+R",K,n;P,R)[-1]
```

Iterasinya sama seperti di atas

$$x_{n+1} = x_n \cdot \left(1 + \frac{P}{100}\right) + R$$

Namun kami tidak lagi menggunakan nilai global R dalam ekspresi kami. Fungsi seperti iterate() memiliki trik khusus di Euler. Anda dapat meneruskan nilai variabel dalam ekspresi sebagai parameter titik koma. Dalam hal ini P dan R.

Apalagi kami hanya tertarik pada nilai terakhir. Jadi kita ambil indeks [-1].

Mari kita coba tes.

```
>f(5000,-200,3,47)
```

-19.83

Sekarang kita bisa menyelesaikan masalah kita.

```
>solve("f(5000,-200,x,50)",3)
```

3.15

Rutinitas penyelesaian menyelesaikan ekspresi=0 untuk variabel x. Jawabannya adalah 3,15% per tahun. Kami mengambil nilai awal 3% untuk algoritma. Fungsi solve() selalu membutuhkan nilai awal.

Kita dapat menggunakan fungsi yang sama untuk menyelesaikan pertanyaan berikut: Berapa banyak yang dapat kita keluarkan per tahun sehingga modal awal habis setelah 20 tahun dengan asumsi tingkat bunga 3% per tahun.

```
>solve("f(5000,x,3,20)",-200)
```

-336.08

Perhatikan bahwa Anda tidak dapat menyelesaikan jumlah tahun, karena fungsi kami mengasumsikan n sebagai nilai bilangan bulat.

Solusi Simbolis Masalah Suku Bunga

Kita dapat menggunakan bagian simbolis dari Euler untuk mempelajari masalahnya. Pertama kita mendefinisikan fungsi onepay() kita secara simbolis.

```
>function op(K) &= K*q+R; &op(K)
```

$$R + q K$$

Sekarang kita dapat mengulanginya.

```
>$&op(op(op(op(K)))) , $&expand(%)
```

$$q (q (q (R + q K) + R) + R) + R$$

$$q^3 R + q^2 R + q R + R + q^4 K$$

Kami melihat sebuah pola. Setelah n periode yang kita miliki

$$K_n = q^n K + R(1 + q + \dots + q^{n-1}) = q^n K + \frac{q^n - 1}{q - 1} R$$

Rumusnya adalah rumus jumlah geometri yang diketahui Maxima.

```
>sum(q^k,k,0,n-1); & % = ev(% ,simpsum)
```

$$\sum_{k=0}^{n-1} q^k = \frac{q^n - 1}{q - 1}$$

Ini agak rumit. Jumlahnya dievaluasi dengan tanda "simpsum" untuk mengurangnya menjadi hasil bagi. Mari kita membuat fungsi untuk ini.

```
>function fs(K,R,P,n) &= (1+P/100)^n*K + ((1+P/100)^n-1)/(P/100)*R; &fs(K,R,P,n)
```

$$\frac{100 \left(\left(\frac{P}{100} + 1 \right)^n - 1 \right) R}{P} + K \left(\frac{P}{100} + 1 \right)^n$$

Fungsinya sama dengan fungsi f kita sebelumnya. Tapi ini lebih efektif.

```
>longest f(5000,-200,3,47), longest fs(5000,-200,3,47)
```

```
-19.82504734650985
-19.82504734652684
```

Sekarang kita dapat menggunakannya untuk menanyakan waktu n . Kapan modal kita habis? Perkiraan awal kami adalah 30 tahun.

```
>solve("fs(5000,-330,3,x)",30)
```

```
20.51
```

Jawaban ini mengatakan akan menjadi negatif setelah 21 tahun.

Kita juga dapat menggunakan sisi simbolis Euler untuk menghitung rumus pembayaran.

Asumsikan kita mendapatkan pinjaman sebesar K , dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama) meninggalkan sisa hutang sebesar K_n (pada saat pembayaran terakhir). Rumusnya jelas

```
>equ &= fs(K,R,P,n)=Kn; &equ
```

$$\frac{100 \left(\left(\frac{P}{100} + 1 \right)^n - 1 \right) R}{P} + K \left(\frac{P}{100} + 1 \right)^n = K_n$$

Biasanya rumus ini diberikan dalam bentuk

$$i = \frac{P}{100}$$

```
>equ &= (equ with P=100*i); &equ
```

$$\frac{((i+1)^n - 1) R}{i} + (i+1)^n K = K_n$$

Kita dapat menyelesaikan nilai R secara simbolis.

```
>&solve(equ,R)
```

$$\left[R = \frac{i K n - i (i+1)^n K}{(i+1)^n - 1} \right]$$

Seperti yang Anda lihat dari rumusnya, fungsi ini mengembalikan kesalahan floating point untuk $i=0$. Euler tetap merencanakannya.

Tentu saja, kami memiliki batasan berikut.

```
>&limit(R(5000,0,x,10),x,0)
```

$$\lim_{x \rightarrow 0} R(5000, 0, x, 10)$$

Yang jelas tanpa bunga kita harus membayar kembali 10 tarif 500.

Persamaan tersebut juga dapat diselesaikan untuk n . Akan terlihat lebih bagus jika kita menerapkan beberapa penyederhanaan padanya.

```
>fn &= solve(equ,n) | ratsimp; &fn
```

$$\left[n = \frac{\log\left(\frac{R+iKn}{R+iK}\right)}{\log(i+1)} \right]$$

article
eumat

Menggambar Grafik 2D dengan EMT

Notebook ini menjelaskan tentang cara menggambar berbagai kurva dan grafik 2D dengan software EMT. EMT menyediakan fungsi `plot2d()` untuk menggambar berbagai kurva dan grafik dua dimensi (2D).

Plot Dasar

Ada fungsi plot yang sangat mendasar. Terdapat koordinat layar yang selalu berkisar antara 0 hingga 1024 di setiap sumbu, tidak peduli apakah layarnya berbentuk persegi atau tidak. Semut terdapat koordinat plot, yang dapat diatur dengan `setplot()`. Pemetaan antar koordinat bergantung pada jendela plot saat ini. Misalnya, `shrinkwindow()` default memberikan ruang untuk label sumbu dan judul plot.

Dalam contoh ini, kita hanya menggambar beberapa garis acak dengan berbagai warna. Untuk rincian tentang fungsi-fungsi ini, pelajari fungsi inti EMT.

```
>clg; // clear screen
>window(0,0,1024,1024); // use all of the window
>setplot(0,1,0,1); // set plot coordinates
>hold on; // start overwrite mode
>n=100; X=random(n,2); Y=random(n,2); // get random points
>colors=rgb(random(n),random(n),random(n)); // get random colors
>loop 1 to n; color(colors[#]); plot(X[#],Y[#]); end; // plot
>hold off; // end overwrite mode
>insimg; // insert to notebook
```



```
>reset;
```

Grafik perlu ditahan, karena perintah `plot()` akan menghapus jendela plot.

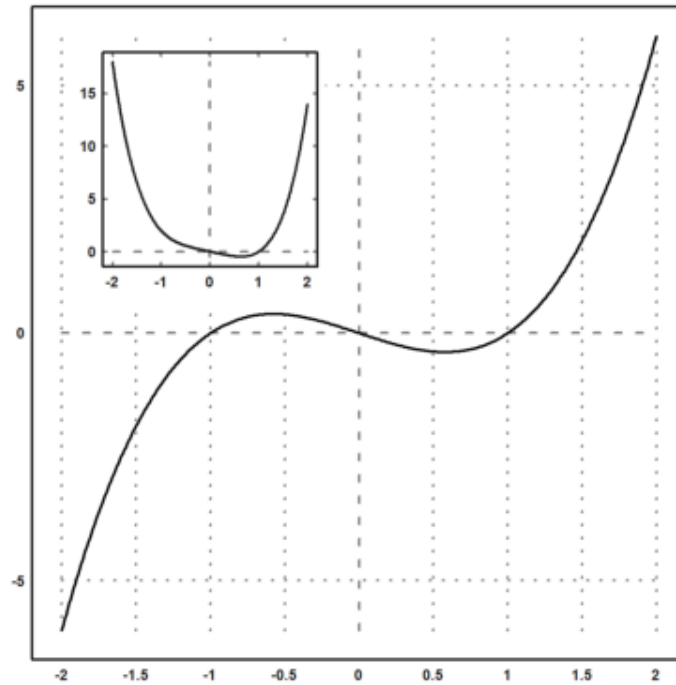
`reset()` digunakan untuk menghapus semua yang telah dilakukan sebelumnya agar tidak mempengaruhi perintah berikutnya.

Untuk menampilkan gambar hasil plot di layar notebook,

- perintah `plot2d()` dapat diakhiri dengan titik dua (:).
- perintah `plot2d()` diakhiri dengan titik koma (;), dilanjutkan menggunakan perintah `insimg()` untuk menampilkan gambar hasil plot.
- `insimg()` digunakan juga untuk mengatur ukuran figur

Contoh lain, saat menggambar plot sebagai sisipan di plot lain. Hal ini dilakukan dengan mendefinisikan jendela plot yang lebih kecil. Perhatikan bahwa jendela ini tidak memberikan ruang untuk label sumbu di luar jendela plot. Kita harus menambahkan beberapa margin untuk ini sesuai kebutuhan. Perhatikan bahwa kita menyimpan dan memulihkan jendela penuh, dan menahan plot saat ini sementara kita memplot inset.

```
>plot2d("x^3-x");
>xw=200; yw=100; ww=300; hw=300;
>ow=window();
>window(xw,yw,xw+ww,yw+hw);
>hold on;
>barclear(xw-50,yw-10,ww+60,ww+60);
>plot2d("x^4-x",grid=6):
```



```
>hold off;  
>window(ow);
```

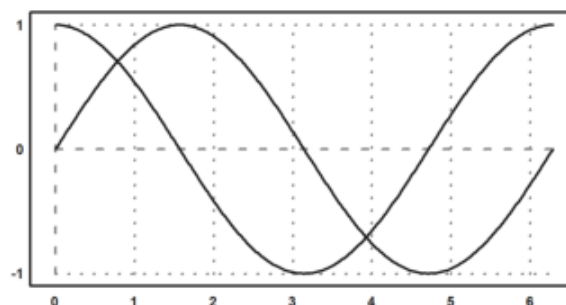
Plot dengan banyak gambar dicapai dengan cara yang sama. Ada fungsi utilitas `figure()` untuk ini.

Aspek Plot

Plot default menggunakan jendela plot persegi. Dapat juga diubah dengan `aspect()`. Jangan lupa untuk mengatur ulang aspeknya nanti. misalnya `aspect(5)`. `aspect()` digunakan untuk menentukan rasio. Aspect dapat juga diubah default di menu dengan "Set Aspect" ke rasio aspek tertentu atau ke ukuran jendela grafik saat ini.

Tapi `aspect` juga bisa diubah hanya untuk satu plot. Untuk ini, ukuran area plot saat ini diubah, dan jendela diatur sehingga label memiliki cukup ruang.

```
>aspect(2); // rasio panjang dan lebar 2:1  
>plot2d(["sin(x)", "cos(x)"], 0, 2pi):
```



```
>aspect();  
>reset;
```

Fungsi `reset()` mengembalikan default plot termasuk rasio aspek.

Plot 2D di Euler

EMT Math Toolbox memiliki plot dalam 2D, baik untuk data maupun fungsi. EMT menggunakan fungsi `plot2d`. Fungsi ini dapat memplot fungsi dan data.

Dimungkinkan untuk membuat plot di Maxima menggunakan Gnuplot atau dengan Python menggunakan Math Plot Lib.

Euler dapat membuat plot 2D

- ekspresi
- fungsi, variabel, atau kurva berparameter,
- vektor nilai x-y,
- awan titik di pesawat,
- kurva implisit dengan level atau wilayah level.
- Fungsi kompleks

Gaya plot mencakup berbagai gaya untuk garis dan titik, plot batang, dan plot berbayang.

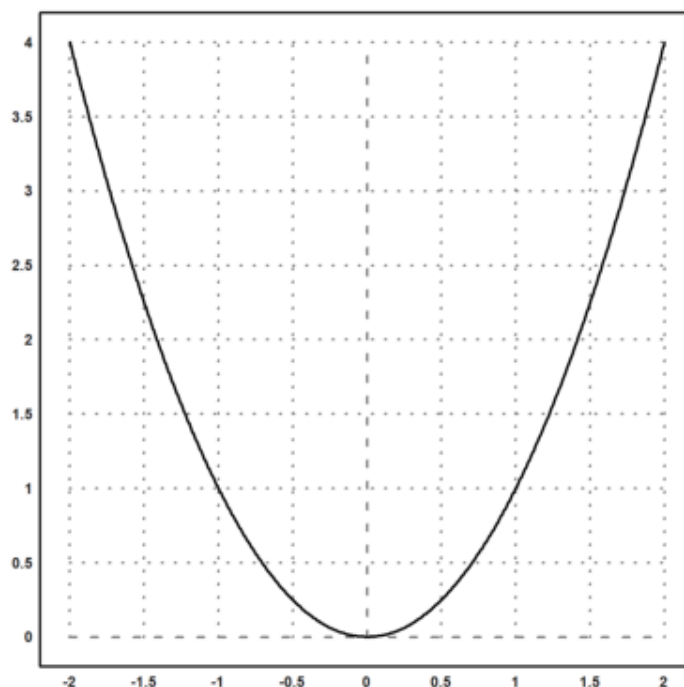
Plot Ekspresi atau Variabel

Ekspresi tunggal dalam "x" (misalnya " $4x^2$ ") atau nama suatu fungsi (misalnya "f") menghasilkan grafik fungsi tersebut.

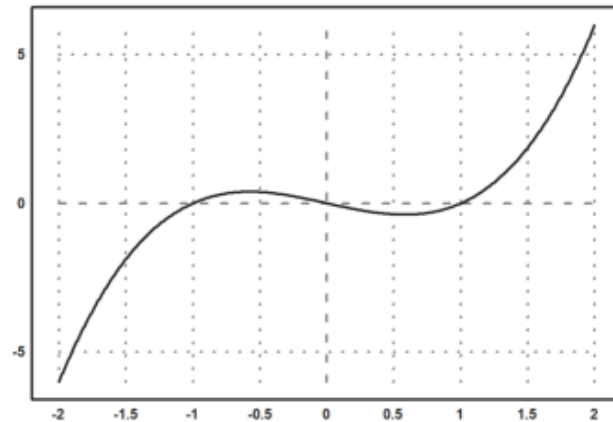
Berikut adalah contoh paling dasar, yang menggunakan rentang default dan menetapkan rentang y yang tepat agar sesuai dengan plot fungsinya.

Catatan: Jika Anda mengakhiri baris perintah dengan titik dua ":", plot akan dimasukkan ke dalam jendela teks. Jika tidak, tekan TAB untuk melihat plot jika jendela plot tertutup.

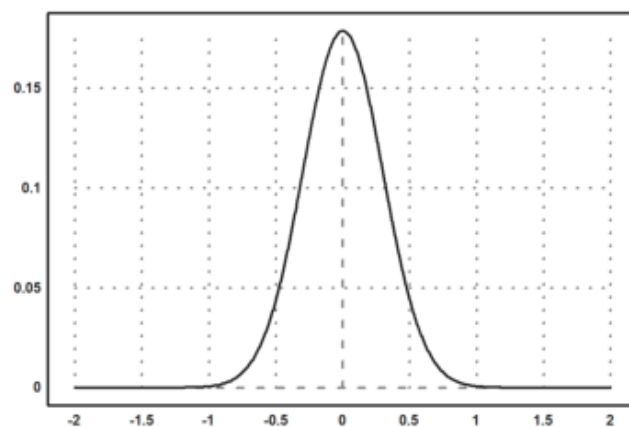
```
>plot2d("x^2"):
```



```
>aspect(1.5); plot2d("x^3-x"):
```



```
>a:=5.6; plot2d("exp(-a*x^2)/a"); insimg(30); // menampilkan gambar hasil plot setinggi 25
```

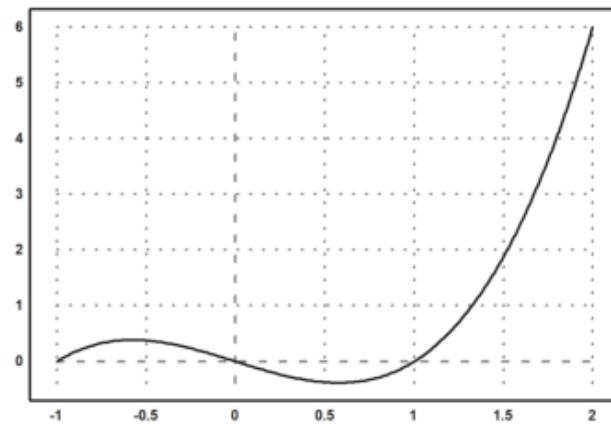


Dari beberapa contoh sebelumnya dapat dilihat bahwa gambar plot aslinya menggunakan sumbu X dengan rentang nilai dari -2 sampai dengan 2. Untuk mengubah jarak nilai X dan Y, dapat ditambahkan nilai-nilai batas X (dan Y) di belakang ekspresi yang digambar.

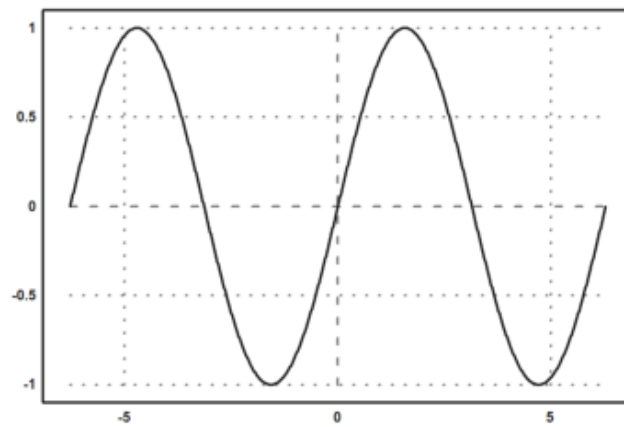
Rentang plot diatur dengan parameter yang ditetapkan sebagai berikut

- 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)

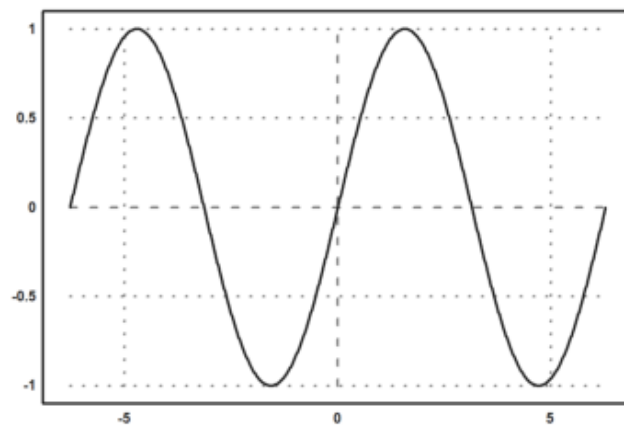
```
>plot2d("x^3-x",-1,2):
```



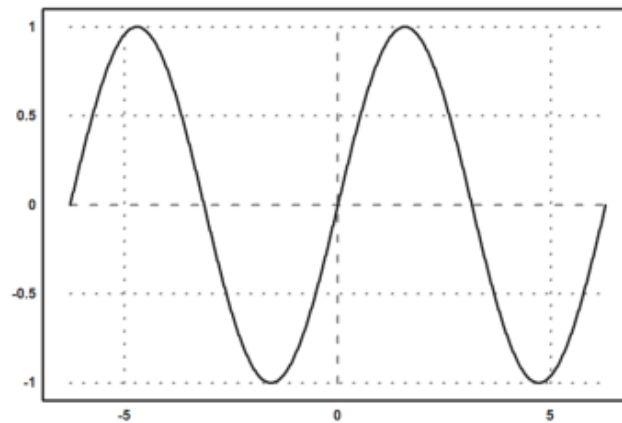
```
>plot2d("sin(x)",-2*pi,2*pi): //plot sin(x) interval [-2pi, 2pi]
```



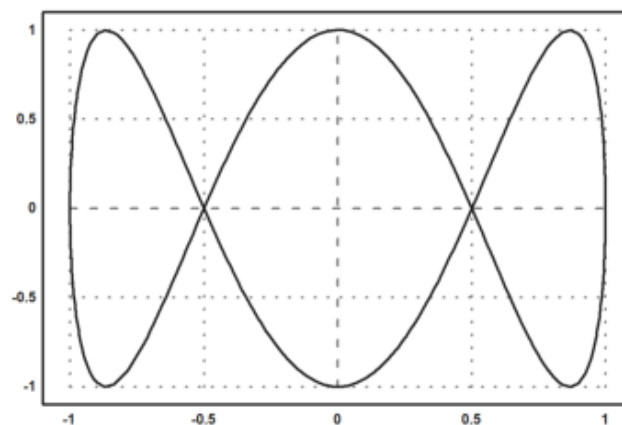
```
>plot2d("sin(x)",-2*pi,2*pi): // plot sin(x) pada interval [-2pi, 2pi]
```



```
>plot2d("sin(x)",-2*pi,2*pi): // plot sin(x) pada interval [-2pi, 2pi]
```



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



Alternatif untuk titik dua adalah perintah `insimg(baris)`, yang menyisipkan plot yang menempati sejumlah baris teks tertentu.

Dalam opsi, plot dapat diatur agar muncul

- di jendela terpisah yang dapat diubah ukurannya,
- di jendela buku catatan.

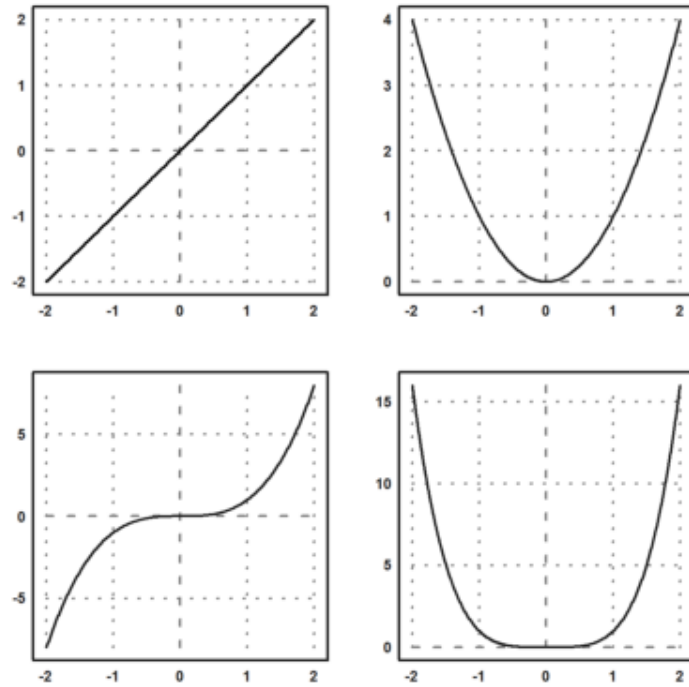
Lebih banyak gaya dapat dicapai dengan perintah plot tertentu.

Bagaimanapun, tekan tombol tabulator untuk melihat plotnya, jika tersembunyi.

Untuk membagi jendela menjadi beberapa plot, gunakan perintah `figure()`. Dalam contoh, memplot x^1 hingga x^4 menjadi 4 bagian jendela.

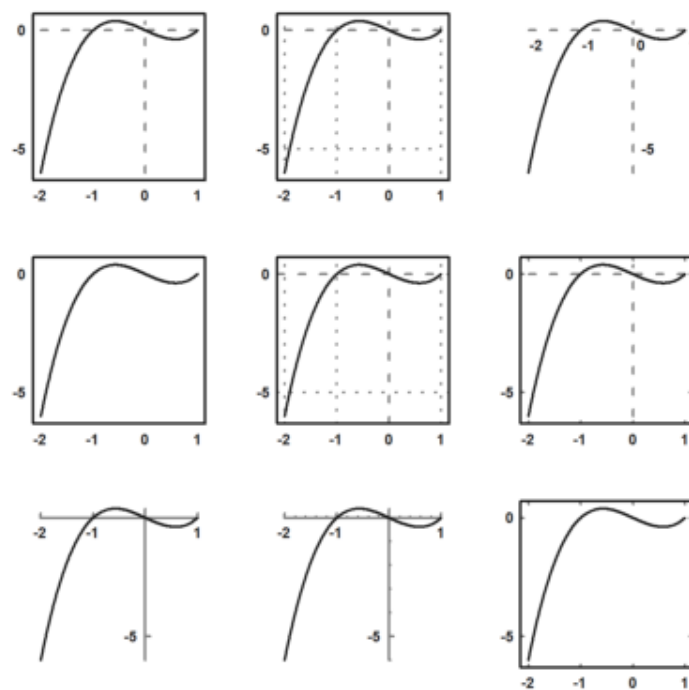
`images(0)` mengatur ulang jendela default.

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



Di `plot2d()`, ada gaya alternatif yang tersedia dengan `grid=x`. Untuk gambaran umum, kami menampilkan berbagai gaya kisi dalam satu gambar (lihat contoh di bawah untuk perintah `figure()`). Gaya `grid=0` tidak disertakan. Ini tidak menunjukkan kisi dan bingkai.

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

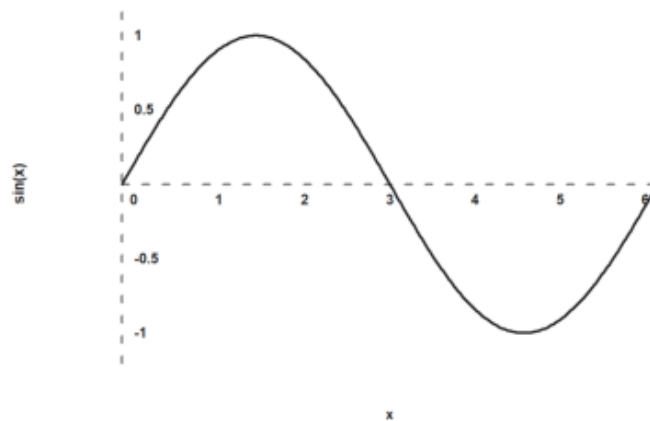


Jika argumen pada `plot2d()` adalah ekspresi yang diikuti oleh empat angka, angka-angka tersebut adalah rentang x dan y untuk plot tersebut.

Alternatifnya, a , b , c , d dapat ditentukan sebagai parameter yang ditetapkan sebagai $a=\dots$ dll.

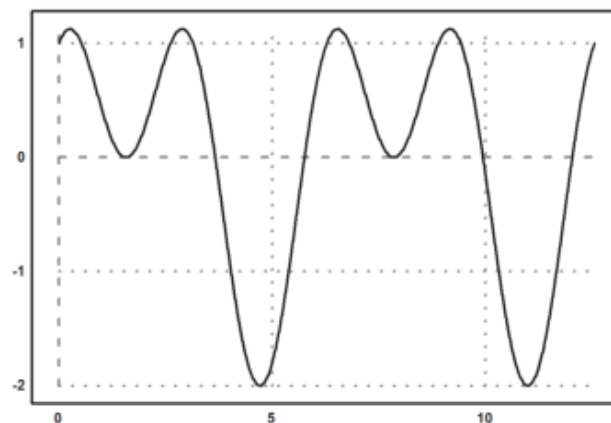
Pada contoh berikut, akan diperlihatkan mengubah gaya kisi, menambahkan label, dan menggunakan label vertikal untuk sumbu y .

```
>aspect(1.5); plot2d("sin(x)",0,2pi,-1.2,1.2,grid=3,xl="x",yl="sin(x)") :
```



dari plot diatas terdapat `aspect(1.5)` yang merupakan perbandingan $y:x$

```
>plot2d("sin(x)+cos(2*x)",0,4pi):
```

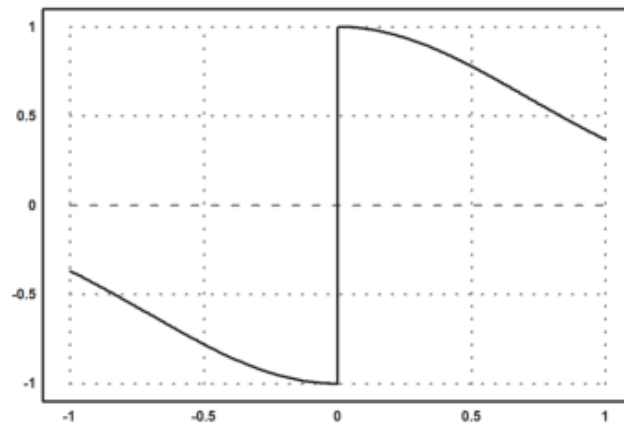


Gambar yang dihasilkan dengan memasukkan plot ke dalam jendela teks disimpan di direktori yang sama dengan buku catatan, secara default di subdirektori bernama "images". Mereka juga digunakan oleh ekspor HTML.

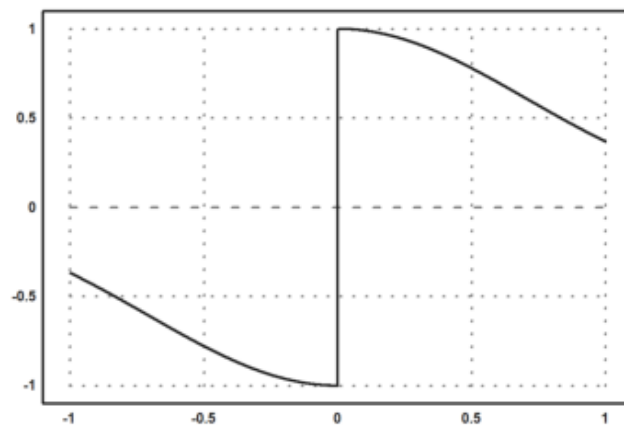
Cukup menandai gambar apa saja dan menyalinnya ke clipboard dengan Ctrl-C. Tentu saja, dapat diekspor grafik saat ini dengan fungsi di menu File.

Fungsi atau ekspresi di `plot2d` dievaluasi secara adaptif. Agar lebih cepat, nonaktifkan plot adaptif dengan "`<adaptive`" dan tentukan jumlah subinterval dengan $n=\dots$. Hal ini hanya diperlukan dalam kasus yang jarang terjadi.

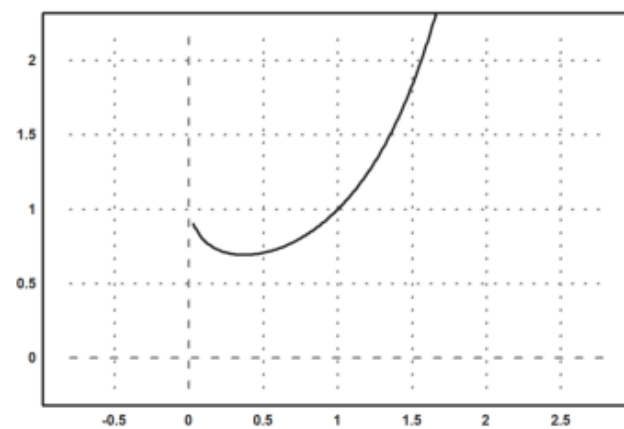
```
>plot2d("sign(x)*exp(-x^2)",-1,1):
```



```
>plot2d("sign(x)*exp(-x^2)",-1,1; <adaptive, n=10000):
```

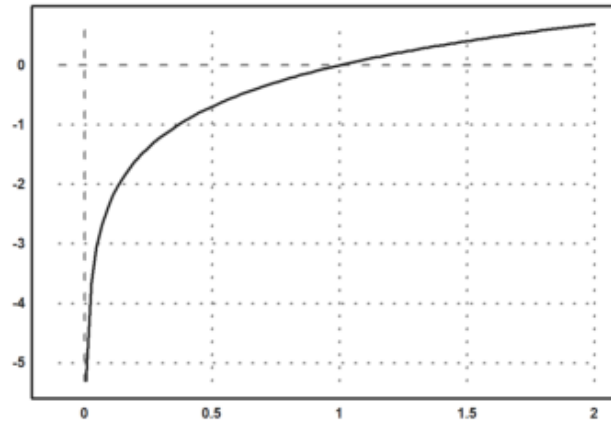


```
>plot2d("x^x",r=1.2,cx=1,cy=1):
```



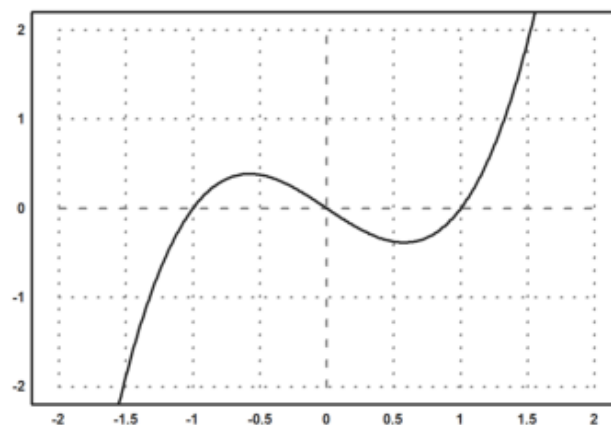
Perhatikan bahwa x^x tidak ditentukan untuk $x \leq 0$. Fungsi `plot2d` menangkap kesalahan ini, dan mulai membuat plot segera setelah fungsinya ditentukan. Ini berfungsi untuk semua fungsi yang mengembalikan NAN di luar jangkauan definisinya.

```
>plot2d("log(x)", -0.1, 2):
```

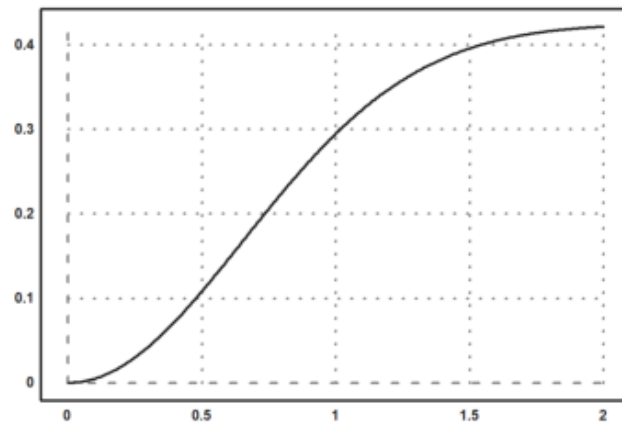


Parameter `square=true` (atau `>square`) memilih rentang y secara otomatis sehingga hasilnya adalah jendela plot persegi. Perhatikan bahwa secara default, Euler menggunakan spasi persegi di dalam jendela plot.

```
>plot2d("x^3-x", >square): //plot logaritma
```

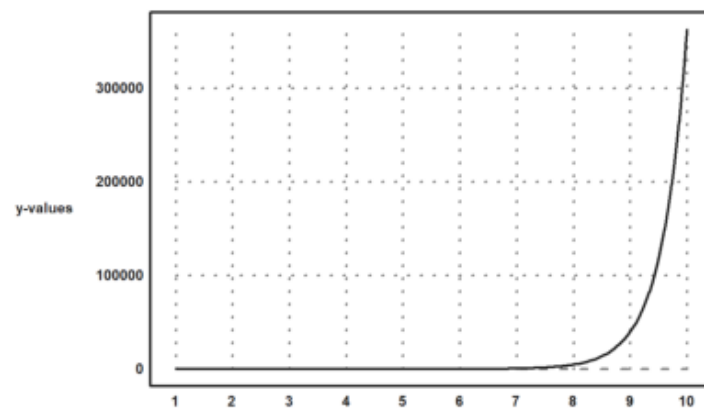


```
>plot2d(''integrate("sin(x)*exp(-x^2)", 0, x)'', 0, 2): // plot integral
```



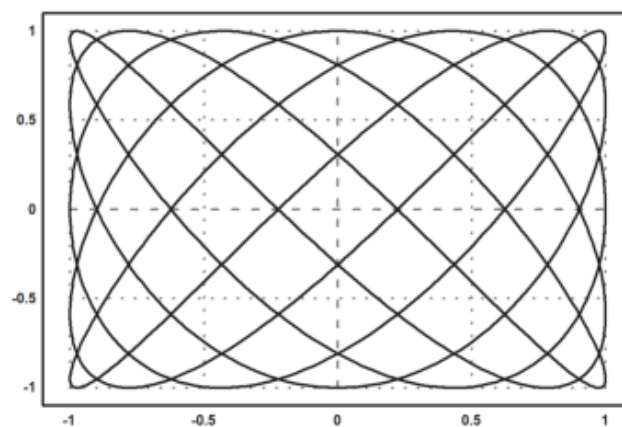
Jika Anda memerlukan lebih banyak ruang untuk label y, panggil `shrinkwindow()` dengan parameter lebih kecil, atau tetapkan nilai positif untuk "lebih kecil" di `plot2d()`.

```
>plot2d("gamma(x) ",1,10,yl="y-values",smaller=6,<vertical):
```

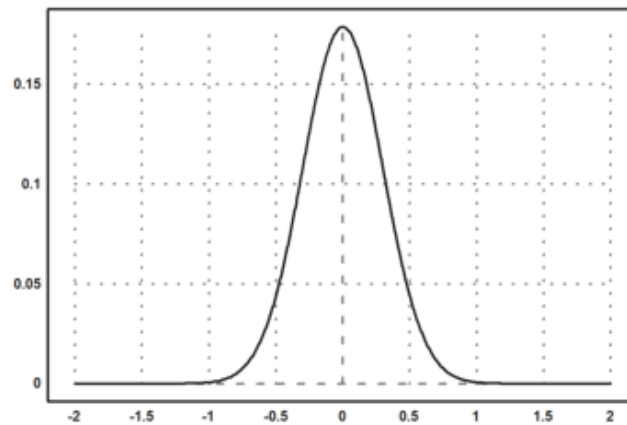


Ekspresi simbolik juga dapat digunakan karena disimpan sebagai ekspresi string sederhana.

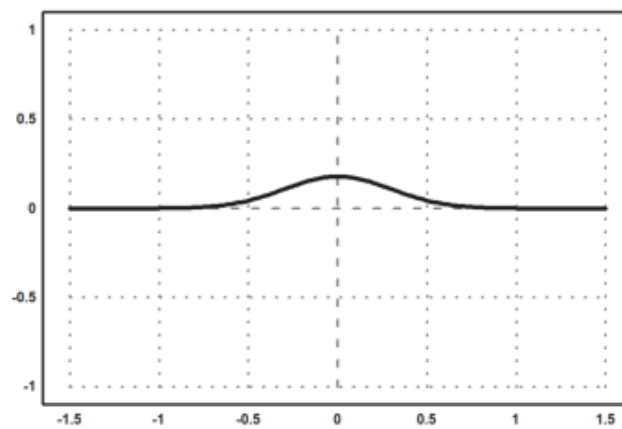
```
>x=linspace(0,2pi,1000); plot2d(sin(5x),cos(7x)):
```



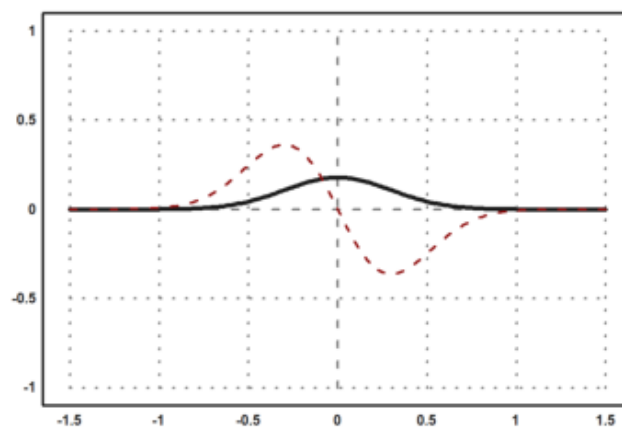
```
>a:=5.6; expr &= exp(-a*x^2)/a; // mendefinisikan ekspresi
>plot2d(expr,-2,2): // plot from -2 to 2
```



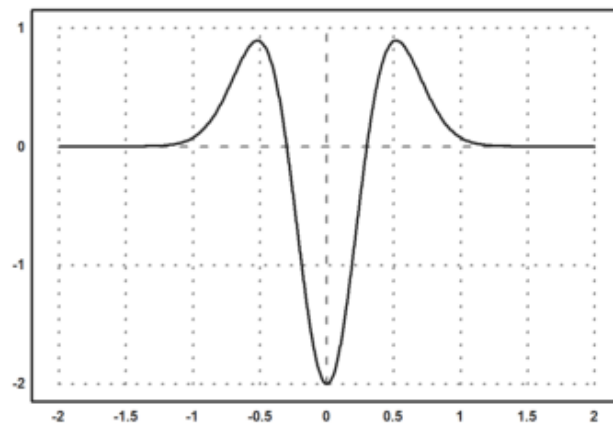
```
>plot2d(expr,r=1,thickness=2): // plot di persegi sekitar (0,0)
```



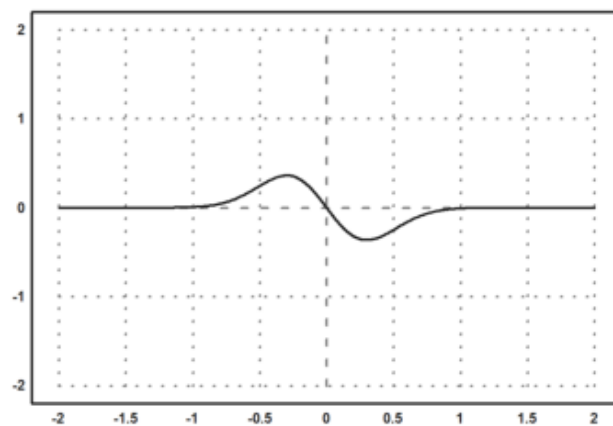
```
>plot2d(&diff(expr,x),>add,style="--",color=red): // penambahan plot lain
```



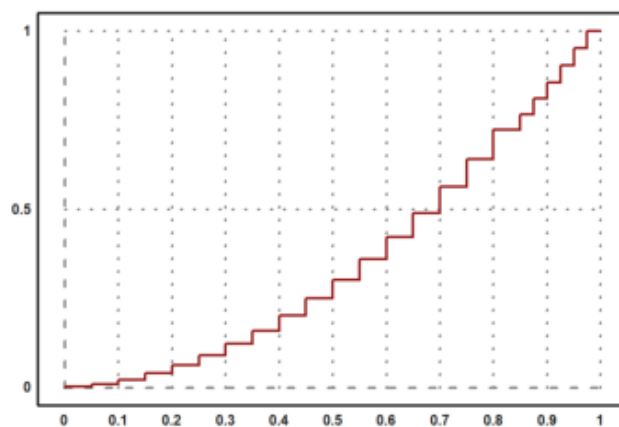
```
>plot2d(&diff(expr,x,2),a=-2,b=2,c=-2,d=1): // plot dalam persegi panjang
```



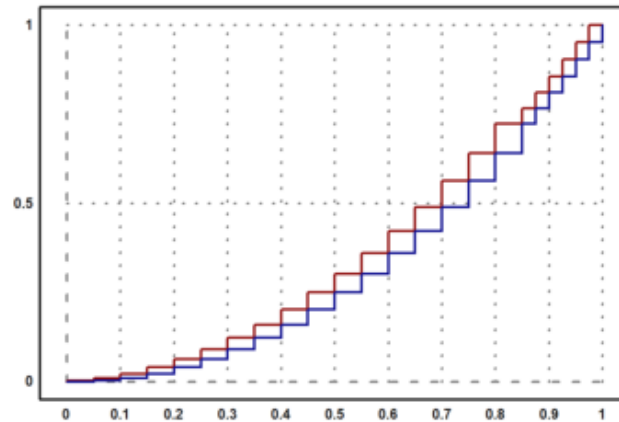
```
>plot2d(&diff(expr,x),a=-2,b=2,>square): // untuk plot tetap persegi
```



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



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

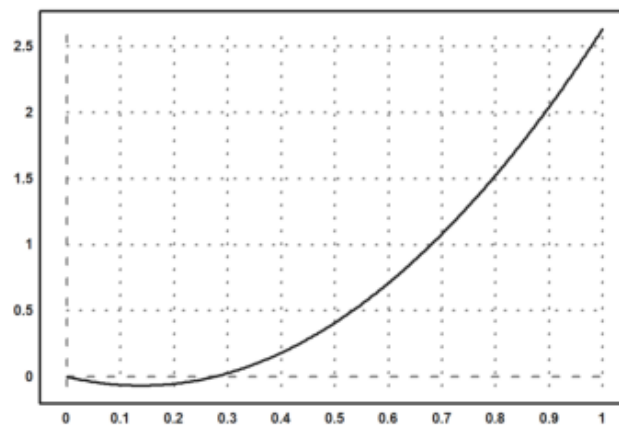


Fungsi dalam satu Parameter

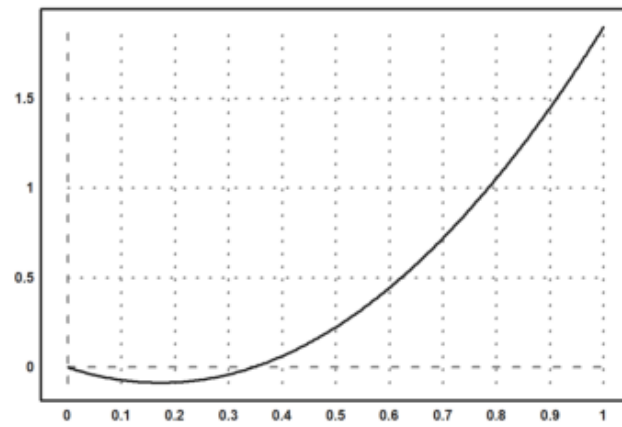
Fungsi plot yang paling penting untuk plot planar adalah `plot2d()`. Fungsi ini diimplementasikan dalam bahasa Euler di file "plot.e", yang dimuat di awal program.

Berikut beberapa contoh penggunaan suatu fungsi. Seperti biasa di EMT, fungsi yang berfungsi untuk fungsi atau ekspresi lain, Anda bisa meneruskan parameter tambahan (selain x) yang bukan variabel global ke fungsi dengan parameter titik koma atau dengan kumpulan panggilan.

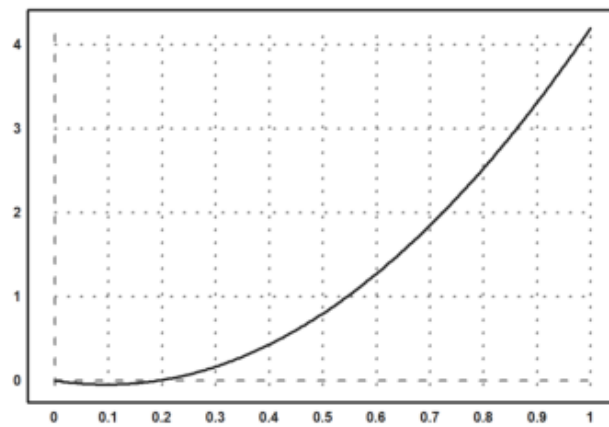
```
>function f(x,a) := x^2/a+a*x^2-x; // mendefinisikan fungsi
>a=0.3; plot2d("f",0,1;a): // plot dengan a=0.3
```



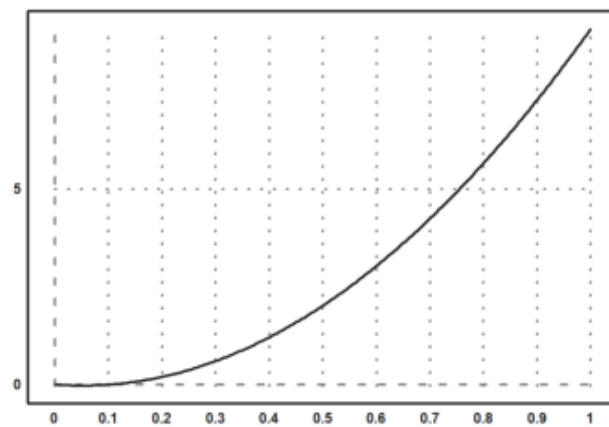
```
>plot2d("f",0,1;0.4): // plot dengan a=0.4
```



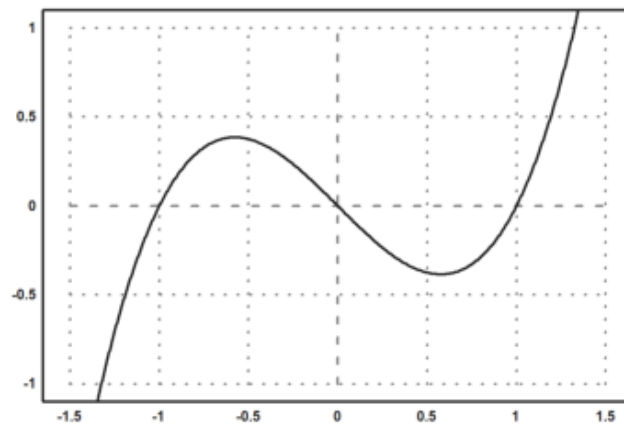
```
>plot2d({{"f",0.2}},0,1): // plot dengan a=0.2
```



```
>plot2d({{"f(x,b)",b=0.1}},0,1): // plot dengan 0.1
```




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



Berikut ini ringkasan fungsi yang diterima

- ekspresi atau ekspresi simbolik di x
- fungsi atau fungsi simbolik dengan nama "f"
- fungsi simbolik hanya dengan nama f

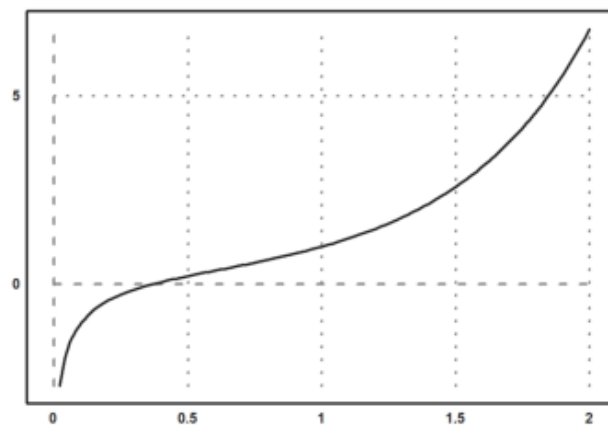
Fungsi plot2d() juga menerima fungsi simbolik.

Untuk fungsi simbolik, namanya saja yang berfungsi.

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

$$x^x (\log(x) + 1)$$

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

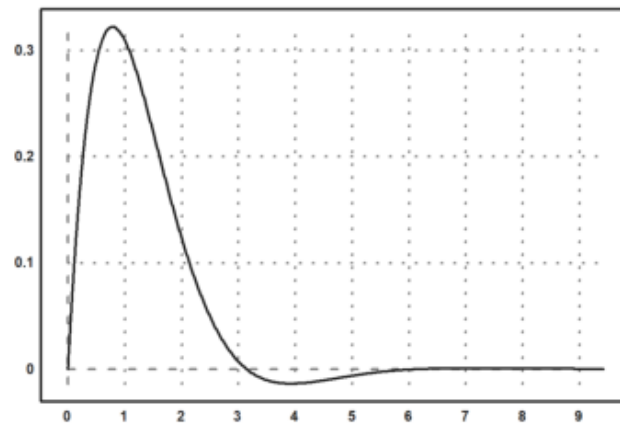


Tentu saja, untuk ekspresi atau ekspresi simbolik, nama variabel sudah cukup untuk memplotnya.

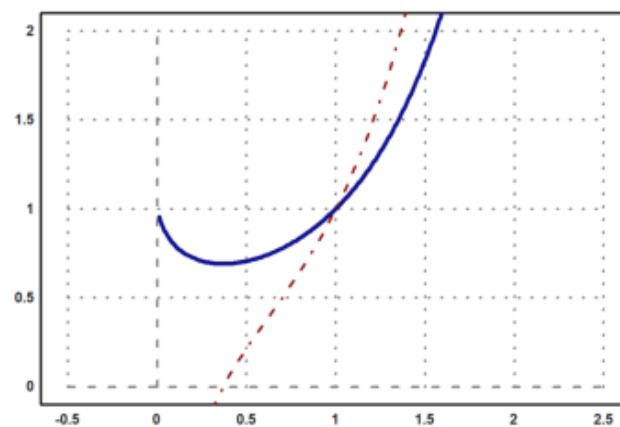
```
>expr &= sin(x)*exp(-x)
```

$$E^{-x} \sin(x)$$

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



```
>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="-.-"):
```



Untuk gaya garis ada berbagai pilihan.

- gaya="...". Pilih dari "-", "--", "-.", ":", ":", ":", "-.-".

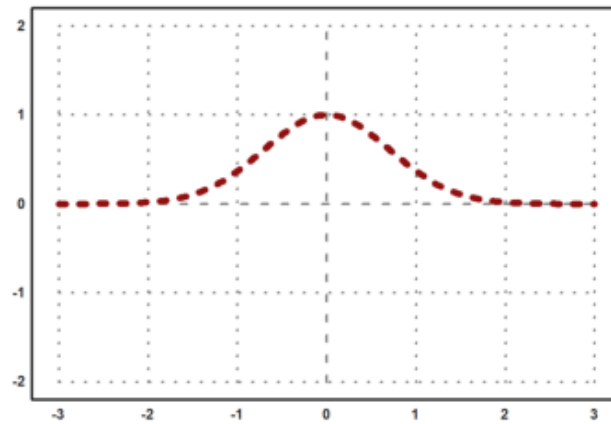
- Warna: Lihat di bawah untuk warna.

- ketebalan: Defaultnya adalah 1.

Warna dapat dipilih sebagai salah satu warna default, atau sebagai warna RGB.

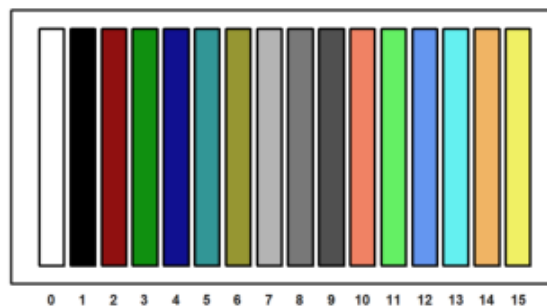
- 0..15: indeks warna default.
- konstanta warna: putih, hitam, merah, hijau, biru, cyan, zaitun, abu-abu muda, abu-abu, abu-abu tua, oranye, hijau muda, pirus, biru muda, oranye muda, kuning
- `rgb(merah,hijau,biru)`: parameternya real di $[0,1]$.

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



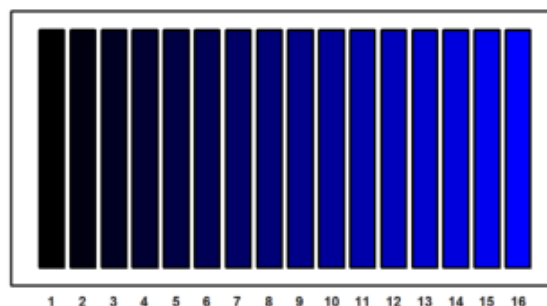
Berikut adalah tampilan warna EMT yang telah ditentukan sebelumnya.

```
>aspect(2); columnsplot(ones(1,16),lab=0:15,grid=0,color=0:15):
```



tapi kamu dapat menggunakan berbagai warna

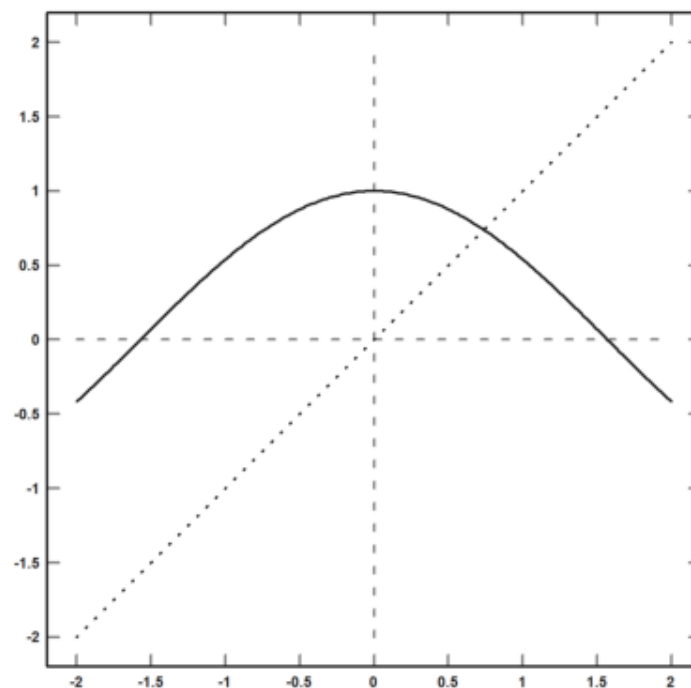
```
>columnsplot(ones(1,16),grid=0,color=rgb(0,0,linspace(0,1,15))):
```



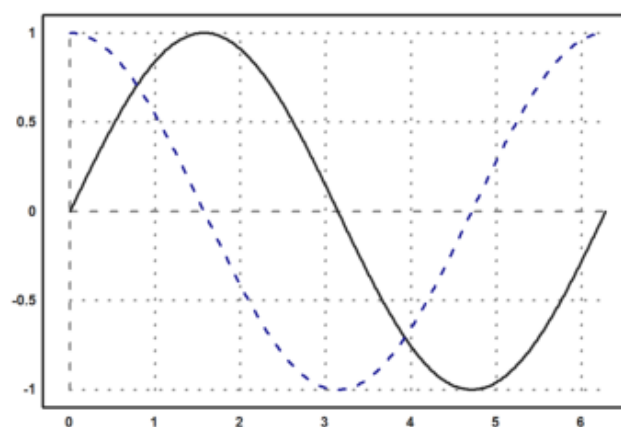
Menggambar Beberapa Kurva dalam satu bidang koordinat

Plot lebih dari satu fungsi (multiple function) ke dalam satu jendela dapat dilakukan dengan berbagai cara. Salah satu metodenya adalah menggunakan ">add" untuk beberapa panggilan ke plot2d secara keseluruhan, kecuali panggilan pertama. Kami telah menggunakan fitur ini pada contoh di atas.

```
>aspect(); plot2d("cos(x)",r=2,grid=6); plot2d("x",style=".",>add):
```

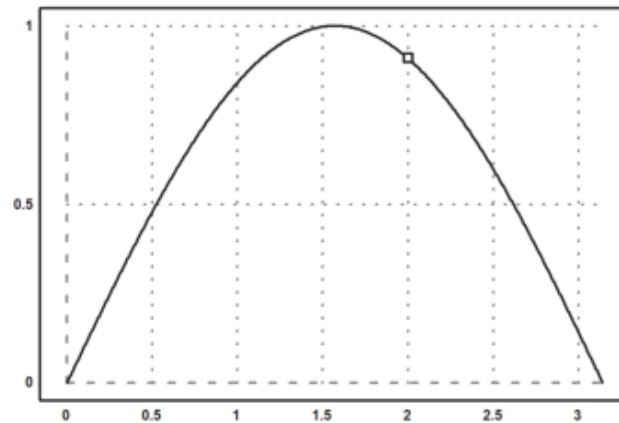


```
>aspect(1.5); plot2d("sin(x)",0,2pi); plot2d("cos(x)",color=blue,style="--",>add):
```



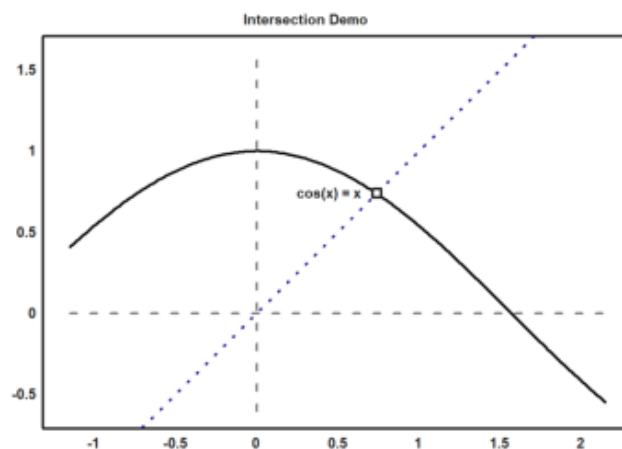
Salah satu kegunaan ">add" adalah untuk menambahkan titik pada kurva.

```
>plot2d("sin(x)",0,pi); plot2d(2,sin(2),>points,>add):
```



Kita tambahkan titik perpotongan dengan label (pada posisi "cl" untuk kiri tengah), dan masukkan hasilnya ke dalam buku catatan. Kami juga menambahkan judul pada plot.

```
>plot2d(["cos(x)", "x"],r=1.1,cx=0.5,cy=0.5, ...
> color=[black,blue],style=["-", "."], ...
> grid=1);
>x0=solve("cos(x)-x",1); ...
> plot2d(x0,x0,>points,>add,title="Intersection Demo"); ...
> label("cos(x) = x",x0,x0,pos="cl",offset=20):
```



Dalam demo berikut, kita memplot fungsi $\sin(x)=\sin(x)/x$ dan ekspansi Taylor ke-8 dan ke-16. Kami menghitung perluasan ini menggunakan Maxima melalui ekspresi simbolik.

Plot ini dilakukan dalam perintah multi-baris berikut dengan tiga panggilan ke plot2d(). Yang kedua dan ketiga memiliki kumpulan tanda >add, yang membuat plot menggunakan rentang sebelumnya.

Kami menambahkan kotak label yang menjelaskan fungsinya.

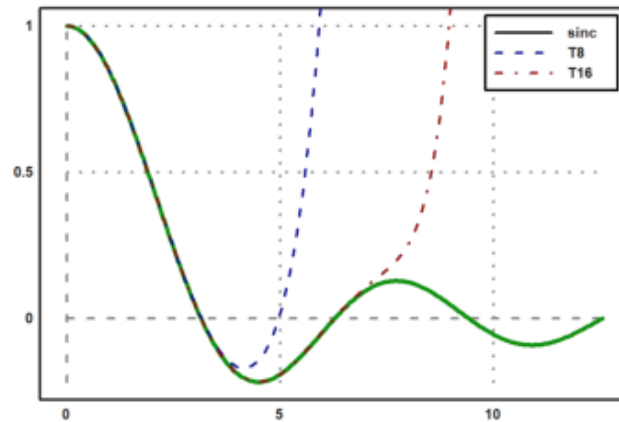
```
>taylor(sin(x)/x,x,0,4)
```

$$\frac{x^4}{120} - \frac{x^2}{6} + 1$$

```

>plot2d("sinc(x)",0,4pi,color=green,thickness=2); ...
> plot2d(&taylor(sin(x)/x,x,0,8),>add,color=blue,style="--"); ...
> plot2d(&taylor(sin(x)/x,x,0,16),>add,color=red,style="-.-"); ...
> labelbox(["sinc","T8","T16"],styles=["-","--","-.-"], ...
> colors=[black,blue,red]):

```



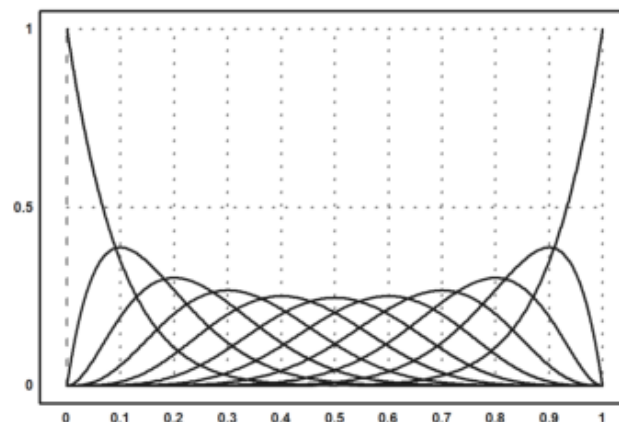
Dalam contoh berikut, kami menghasilkan Polinomial Bernstein.

$$B_i(x) = \binom{n}{i} x^i (1-x)^{n-i}$$

```

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

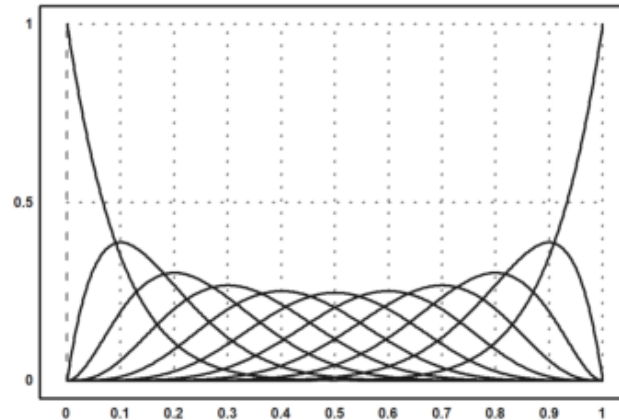
```



Cara kedua adalah dengan menggunakan pasangan matriks bernilai x dan matriks bernilai y yang berukuran sama.

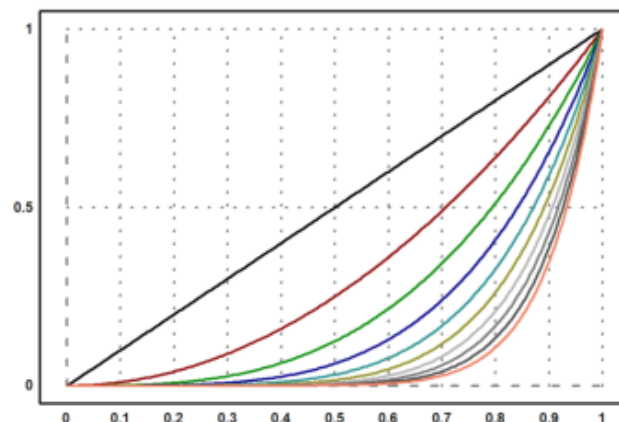
Kami menghasilkan matriks nilai dengan satu Polinomial Bernstein di setiap baris. Untuk ini, kita cukup menggunakan vektor kolom i. Lihat pendahuluan tentang bahasa matriks untuk mempelajari lebih detail.

```
>x=linspace(0,1,500);
>n=10; k=(0:n)'; // n is row vector, k is column vector
>y=bin(n,k)*x^k*(1-x)^(n-k); // y is a matrix then
>plot2d(x,y):
```



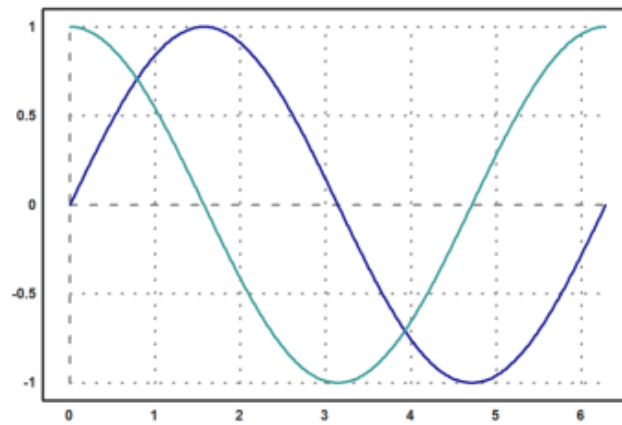
Perhatikan bahwa parameter warna dapat berupa vektor. Kemudian setiap warna digunakan untuk setiap baris matriks.

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

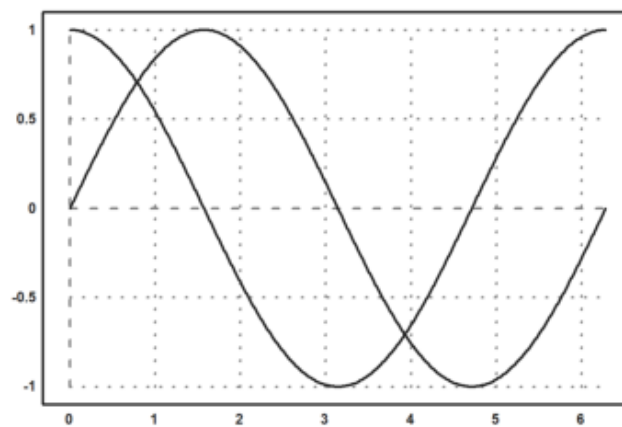


Metode lain adalah menggunakan vektor ekspresi (string). Anda kemudian dapat menggunakan susunan warna, susunan gaya, dan susunan ketebalan dengan panjang yang sama.

```
>plot2d(["sin(x)", "cos(x)"], 0, 2pi, color=4:5):
```



```
>plot2d(["sin(x)","cos(x)"],0,2pi): // plot vektor ekspresi
```



Kita bisa mendapatkan vektor seperti itu dari Maxima menggunakan makelist() dan mxm2str().

```
>v &= makelist(binomial(10,i)*x^i*(1-x)^(10-i),i,0,10) // membuat list
```

```

      10      9      8 2      7 3
      (1 - x) , 10 (1 - x) x, 45 (1 - x) x , 120 (1 - x) x ,
      6 4      5 5      4 6      3 7
210 (1 - x) x , 252 (1 - x) x , 210 (1 - x) x , 120 (1 - x) x ,
      2 8      9 10
45 (1 - x) x , 10 (1 - x) x , x ]
```

```
>mxm2str(v) // untuk mendapatkan vektor string dari vektor simbolik
```

```

(1-x)^10
10*(1-x)^9*x
45*(1-x)^8*x^2
120*(1-x)^7*x^3
```

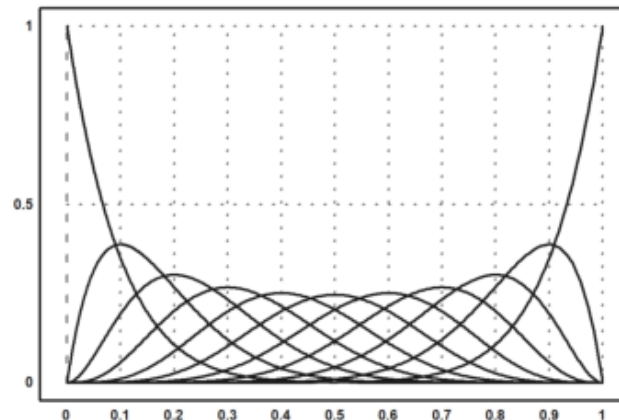


```

210*(1-x)^6*x^4
252*(1-x)^5*x^5
210*(1-x)^4*x^6
120*(1-x)^3*x^7
45*(1-x)^2*x^8
10*(1-x)*x^9
x^10

```

```
>plot2d(mxm2str(v),0,1): // plot fungsi
```

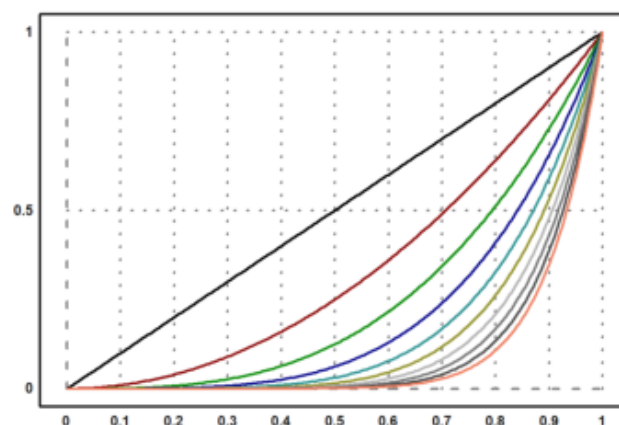


Alternatif lain adalah dengan menggunakan bahasa matriks Euler.

Jika suatu ekspresi menghasilkan matriks fungsi, dengan satu fungsi di setiap baris, semua fungsi tersebut akan diplot ke dalam satu plot.

Untuk ini, gunakan vektor parameter dalam bentuk vektor kolom. Jika array warna ditambahkan maka akan digunakan untuk setiap baris plot.

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

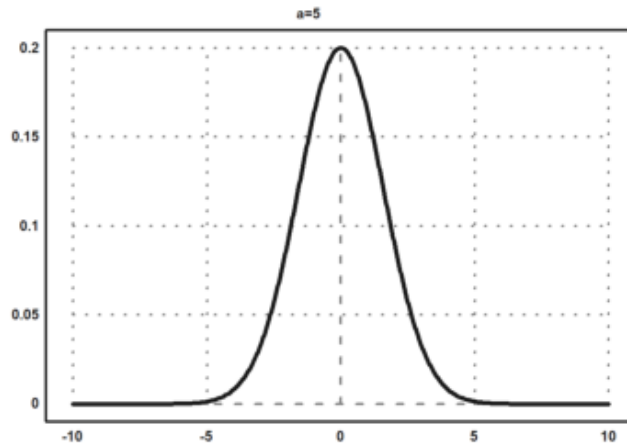


Ekspresi dan fungsi satu baris dapat melihat variabel global.

Jika Anda tidak dapat menggunakan variabel global, Anda perlu menggunakan fungsi dengan parameter tambahan, dan meneruskan parameter ini sebagai parameter titik koma.

Berhati-hatilah, untuk meletakkan semua parameter yang ditetapkan di akhir perintah plot2d. Dalam contoh ini kita meneruskan a=5 ke fungsi f, yang kita plot dari -10 hingga 10.

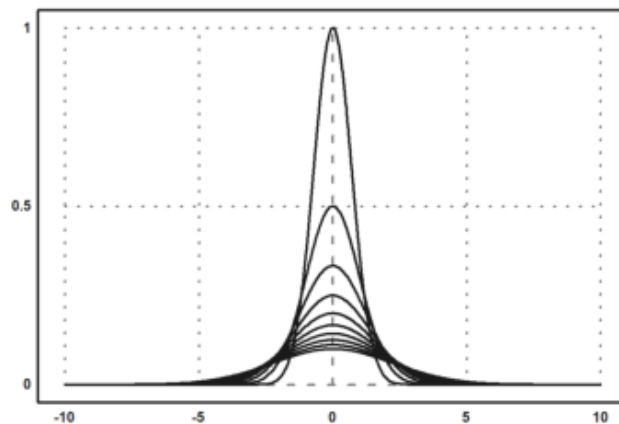
```
>function f(x,a) := 1/a*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5"):
```



Alternatifnya, gunakan koleksi dengan nama fungsi dan semua parameter tambahan. Daftar khusus ini disebut kumpulan panggilan, dan ini adalah cara yang lebih disukai untuk meneruskan argumen ke suatu fungsi yang kemudian diteruskan sebagai argumen ke fungsi lain.

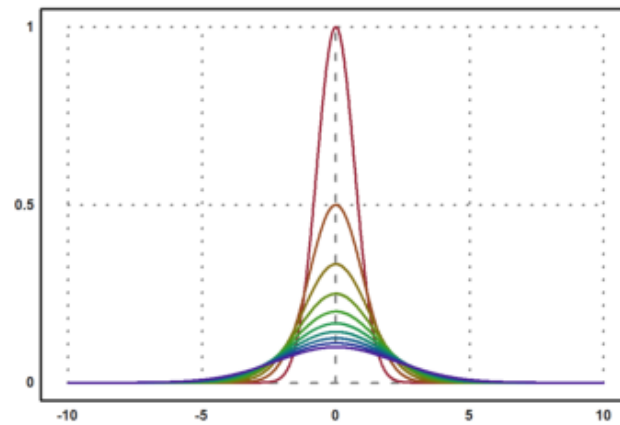
Pada contoh berikut, kita menggunakan loop untuk memplot beberapa fungsi (lihat tutorial tentang pemrograman loop/pengulangan).

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



Kita dapat mencapai hasil yang sama dengan cara berikut menggunakan bahasa matriks EMT. Setiap baris matriks $f(x,a)$ merupakan satu fungsi. Selain itu, kita dapat mengatur warna untuk setiap baris matriks. Klik dua kali pada fungsi `getspectral()` untuk penjelasannya.

```
>x=-10:0.01:10; a=(1:10)'; plot2d(x,f(x,a),color=getspectral(a/10)):
```



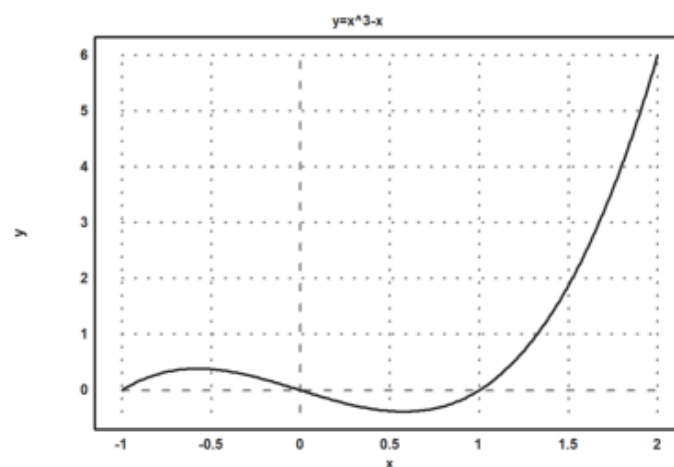
Label Teks

Dekorasi sederhana pun bisa

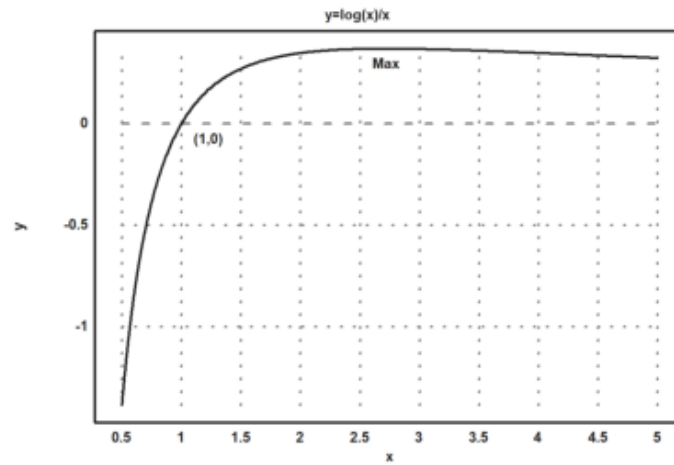
- judul dengan judul = "..."
- label x dan y dengan xl="...", yl="..."
- label teks lain dengan label("...",x,y)

Perintah label akan memplot ke plot saat ini pada koordinat plot (x,y). Hal ini memerlukan argumen posisional.

```
>plot2d("x^3-x",-1,2,title="y=x^3-x",yl="y",xl="x") :
```

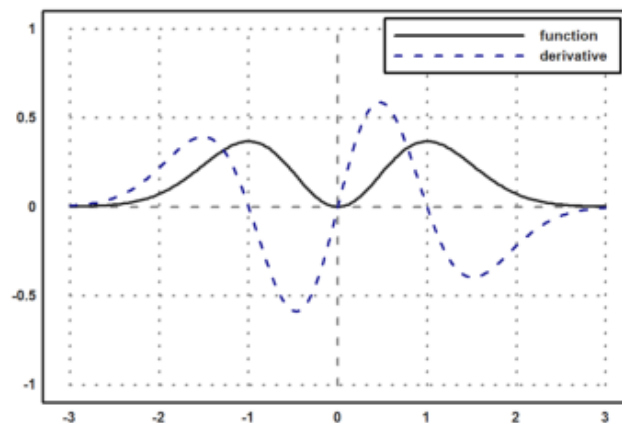


```
>expr := "log(x)/x"; ...
> plot2d(expr,0.5,5,title="y="+expr,xl="x",yl="y"); ...
> label("(1,0)",1,0); label("Max",E,expr(E),pos="lc") :
```



Ada juga fungsi `labelbox()`, yang dapat menampilkan fungsi dan teks. Dibutuhkan vektor string dan warna, satu item untuk setiap fungsi.

```
>function f(x) &= x^2*exp(-x^2); ...
>plot2d(&f(x),a=-3,b=3,c=-1,d=1); ...
>plot2d(&diff(f(x),x),>add,color=blue,style="--"); ...
>labelbox(["function","derivative"],styles=["-", "--"], ...
> colors=[black,blue],w=0.4):
```

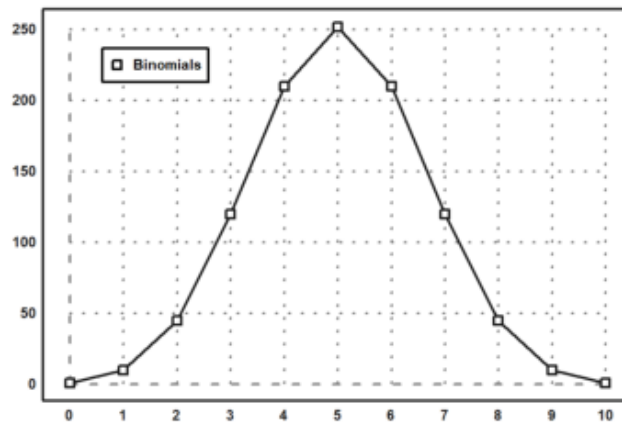


Kotak ini berlabuh di kanan atas secara default, tetapi `>left` berlabuh di kiri atas. Anda dapat memindahkannya ke tempat mana pun yang Anda suka. Posisi jangkar berada di pojok kanan atas kotak, dan angkanya merupakan pecahan dari ukuran jendela grafis. Lebarinya otomatis.

Untuk plot titik, kotak label juga berfungsi. Tambahkan parameter `>points`, atau vektor bendera, satu untuk setiap label.

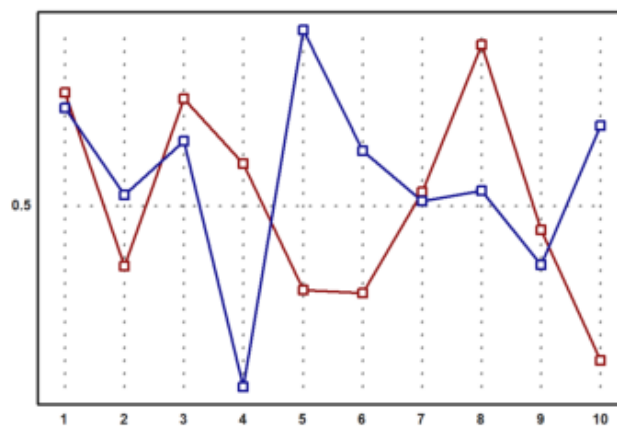
Pada contoh berikut, hanya ada satu fungsi. Jadi kita bisa menggunakan string sebagai pengganti vektor string. Kami mengatur warna teks menjadi hitam untuk contoh ini.

```
>n=10; plot2d(0:n,bin(n,0:n),>addpoints); ...
>labelbox("Binomials",styles="[]",>points,x=0.1,y=0.1, ...
>tcolor=black,>left):
```



Gaya plot ini juga tersedia di `statplot()`. Seperti di `plot2d()` warna dapat diatur untuk setiap baris plot. Masih banyak lagi plot khusus untuk keperluan statistik (lihat tutorial tentang statistik).

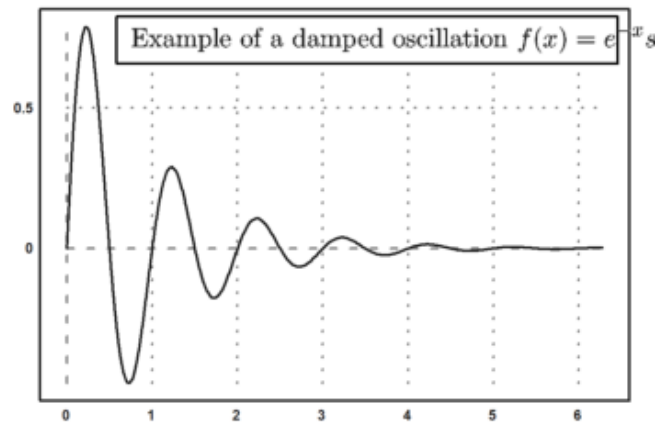
```
>statplot(1:10,random(2,10),color=[red,blue]):
```



Fitur serupa adalah fungsi `textbox()`.

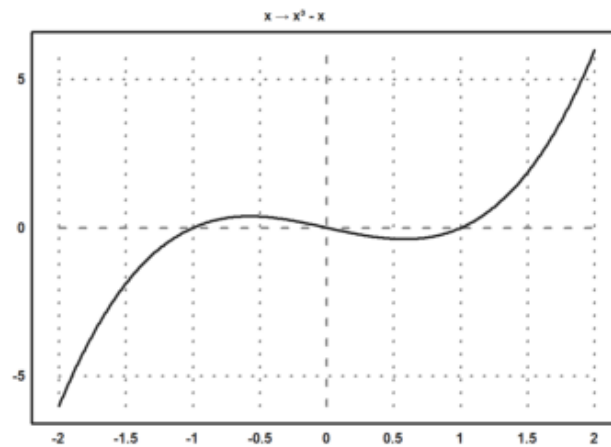
Lebar nya secara default adalah lebar maksimal baris teks. Tapi itu bisa diatur oleh pengguna juga.

```
>function f(x) &= exp(-x)*sin(2*pi*x); ...
>plot2d("f(x)",0,2pi); ...
>textbox(latex("\text{Example of a damped oscillation}\ f(x)=e^{-x}\sin(2\pi x)"),w=0.85):
```



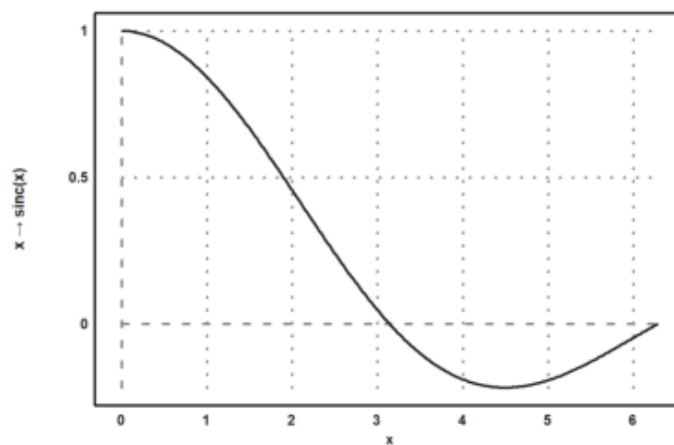
Label teks, judul, kotak label, dan teks lainnya dapat berisi string Unicode (lihat sintaks EMT untuk mengetahui lebih lanjut tentang string Unicode).

```
>plot2d("x^3-x",title=u"x \rarr; x&sup3; - x"):
```



Label pada sumbu x dan y bisa vertikal, begitu juga dengan sumbunya.

```
>plot2d("sinc(x)",0,2pi,xl="x",yl=u"x \rarr; sinc(x)",>vertical):
```



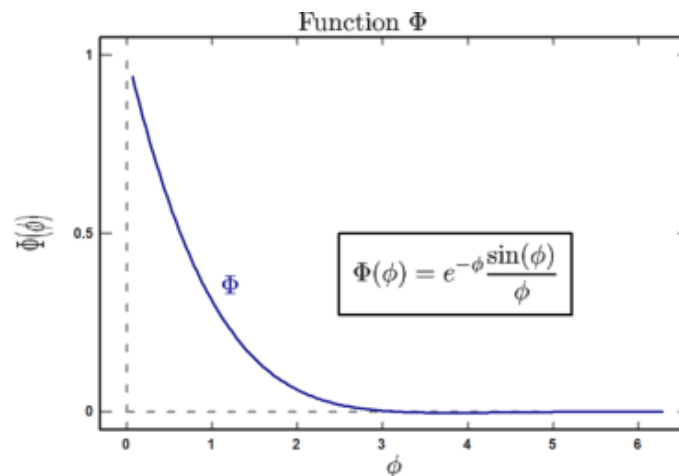
LaTeX

Anda juga dapat memplot rumus LaTeX jika Anda telah menginstal sistem LaTeX. Saya merekomendasikan MiKTeX. Jalur ke biner "latex" dan "dvipng" harus berada di jalur sistem, atau Anda harus mengatur LaTeX di menu opsi.

Perhatikan, penguraian LaTeX lambat. Jika Anda ingin menggunakan LaTeX dalam plot animasi, Anda harus memanggil `latex()` sebelum loop satu kali dan menggunakan hasilnya (gambar dalam matriks RGB).

Pada plot berikut, kami menggunakan LaTeX untuk label x dan y, label, kotak label, dan judul plot.

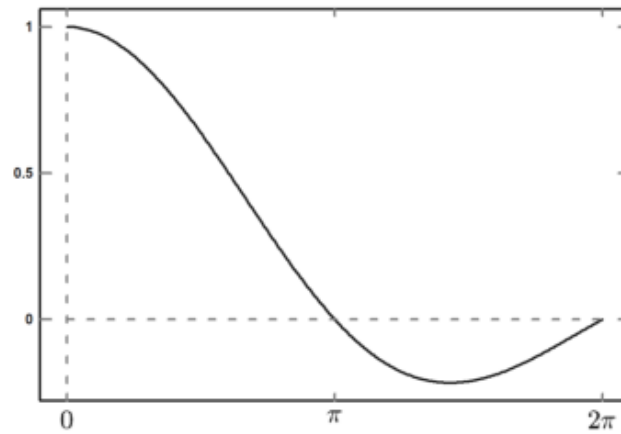
```
>plot2d("exp(-x)*sin(x)/x",a=0,b=2pi,c=0,d=1,grid=6,color=blue, ...
> title=latex("\text{Function \Phi$}"), ...
> xl=latex("\phi"),yl=latex("\Phi(\phi)")); ...
>textbox( ...
> latex("\Phi(\phi) = e^{-\phi} \frac{\sin(\phi)}{\phi}",x=0.8,y=0.5); ...
>label(latex("\Phi",color=blue),1,0.4):
```



Seringkali, kita menginginkan spasi dan label teks yang tidak konformal pada sumbu x. Kita bisa menggunakan `xaxis()` dan `yaxis()` seperti yang akan kita tunjukkan nanti.

Cara termudah adalah membuat plot kosong dengan bingkai menggunakan `grid=4`, lalu menambahkan grid dengan `ygrid()` dan `xgrid()`. Pada contoh berikut, kami menggunakan tiga string LaTeX untuk label pada sumbu x dengan `xtick()`.

```
>plot2d("sinc(x)",0,2pi,grid=4,<ticks); ...
>ygrid(-2:0.5:2,grid=6); ...
>xgrid([0:2]*pi,<ticks,grid=6); ...
>xtick([0,pi,2pi],["0","\pi","2\pi"],>latex):
```



Tentu saja fungsinya juga bisa digunakan.

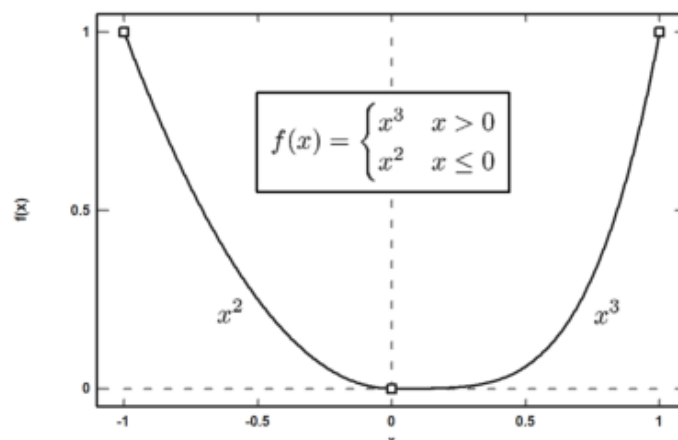
```
>function map f(x) ...
```

```
    if x>0 then return x^4
    else return x^2
    endif
endfunction
```

Parameter "peta" membantu menggunakan fungsi untuk vektor. Untuk plot, itu tidak perlu. Tapi untuk menunjukkan vektorisasi itu berguna, kita menambahkan beberapa poin penting ke plot di $x=-1$, $x=0$ dan $x=1$.

Pada plot berikut, kami juga memasukkan beberapa kode LaTeX. Kami menggunakannya untuk dua label dan kotak teks. Tentu saja, Anda hanya bisa menggunakannya LaTeX jika Anda telah menginstal LaTeX dengan benar.

```
>plot2d("f",-1,1,xl="x",yl="f(x)",grid=6); ...
>plot2d([-1,0,1],f([-1,0,1]),>points,>add); ...
>label(latex("x^3"),0.72,f(0.72)); ...
>label(latex("x^2"),-0.52,f(-0.52),pos="ll"); ...
>textbox( ...
> latex("f(x)=\begin{cases} x^3 & x>0 \\ x^2 & x \le 0 \end{cases}"), ...
> x=0.7,y=0.2):
```



Interaksi pengguna

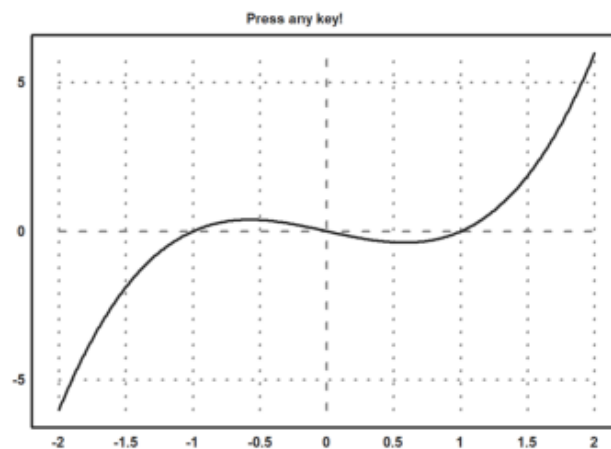
Saat memplot suatu fungsi atau ekspresi, parameter `>pengguna` memungkinkan pengguna untuk memperbesar dan menggeser plot dengan tombol kursor atau mouse. Pengguna bisa

- perbesar dengan + atau -
- pindahkan plot dengan tombol kursor
- pilih jendela plot dengan mouse
- atur ulang tampilan dengan spasi
- keluar dengan kembali

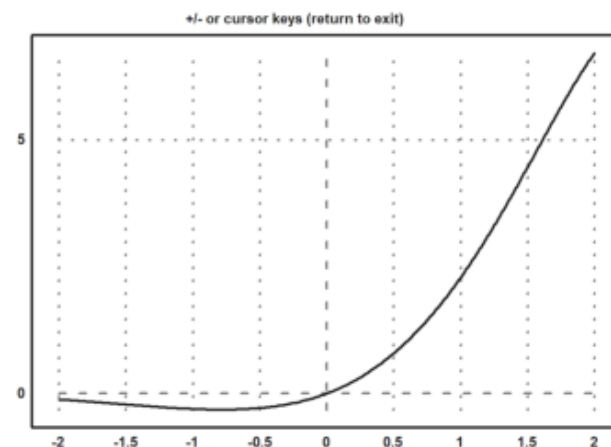
Tombol spasi akan mengatur ulang plot ke jendela plot aslinya.

Saat memplot data, flag `>user` hanya akan menunggu penekanan tombol.

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



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



Berikut ini menunjukkan cara interaksi pengguna tingkat lanjut (lihat tutorial tentang pemrograman untuk detailnya).

Fungsi bawaan `mousedrag()` menunggu aktivitas mouse atau keyboard. Ini melaporkan mouse ke bawah, gerakan mouse atau mouse ke atas, dan penekanan tombol. Fungsi `dragpoints()` memanfaatkan ini, dan memungkinkan pengguna menyeret titik mana pun dalam plot.

Kita membutuhkan fungsi plot terlebih dahulu. Misalnya, kita melakukan interpolasi pada 5 titik dengan polinomial. Fungsi tersebut harus diplot ke dalam area plot yang tetap.

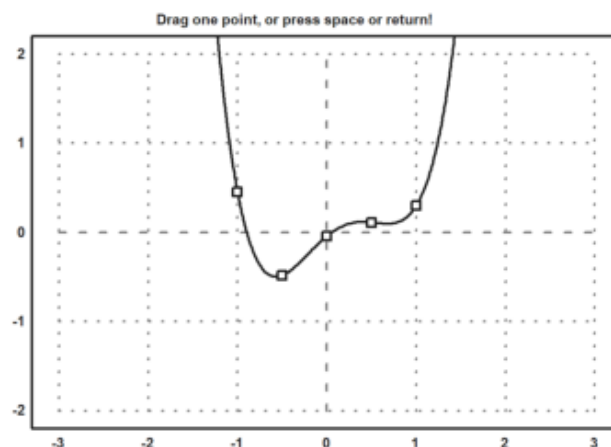
```
>function plotf(xp,yp,select) ...

    d=interp(xp,yp);
    plot2d("interpval(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 nilainya secara global.

Sekarang kita menghasilkan beberapa nilai acak, dan membiarkan pengguna menyeret titiknya.

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



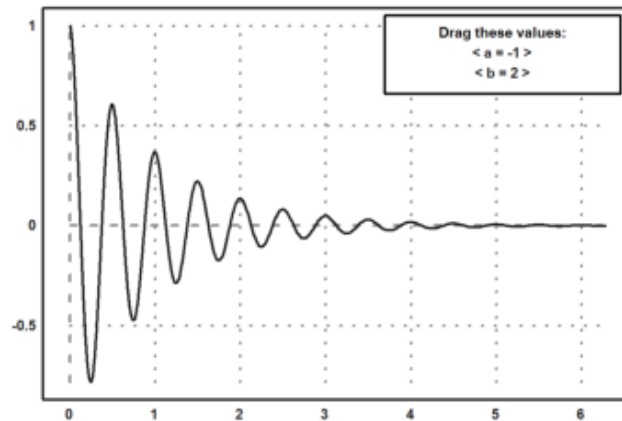
Ada juga fungsi yang memplot fungsi lain bergantung pada vektor parameter, dan memungkinkan pengguna menyesuaikan parameter ini.

Pertama kita membutuhkan fungsi plot.

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

Kemudian kita memerlukan nama untuk parameter, nilai awal dan matriks rentang $n \times 2$, opsional garis judul. Ada penggeser interaktif, yang dapat menetapkan nilai oleh pengguna. Fungsi `dragvalues()` menyediakan ini.

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



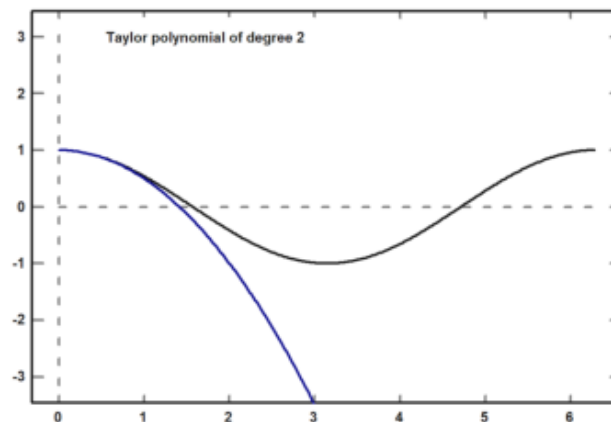
Dimungkinkan untuk membatasi nilai yang diseret menjadi bilangan bulat. Sebagai contoh, kita menulis fungsi plot, yang memplot polinomial Taylor berderajat n ke fungsi kosinus.

```
>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 kita izinkan derajat n bervariasi dari 0 hingga 20 dalam 20 perhentian. Hasil dragvalues() digunakan untuk memplot sketsa dengan n ini, dan untuk memasukkan plot ke dalam buku catatan.

```
>nd=dragvalues("plotf","degree",2,[0,20],20,y=0.8, ...
> heading="Drag the value:"); ...
>plotf(nd):
```



Berikut ini adalah demonstrasi sederhana dari fungsinya. Pengguna dapat menggambar jendela plot, meninggalkan jejak titik.

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

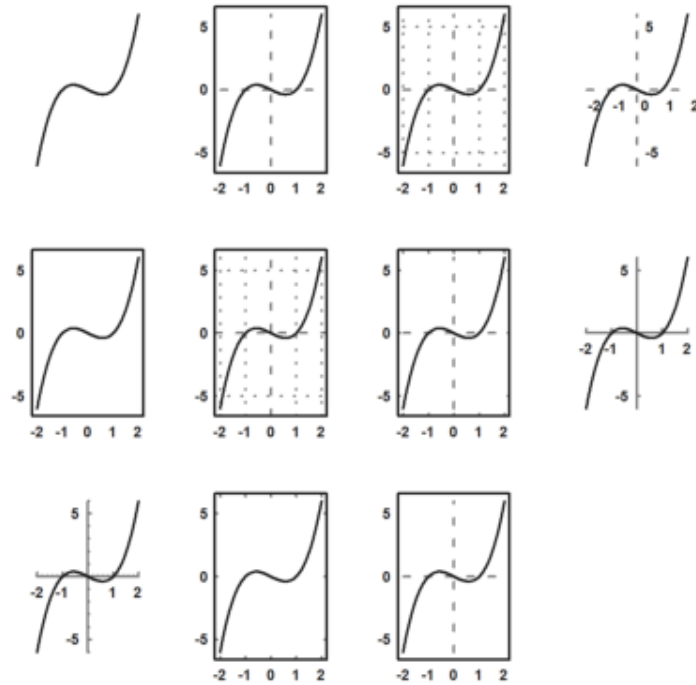
```
    plot2d(none,r=1,title="Drag with the mouse, or press any key!");
    start=0;
    repeat
        {flag,m,time}=mousedrag();
        if flag==0 then return; endif;
        if flag==2 then
            hold on; mark(m[1],m[2]); hold off;
        endif;
    end
endfunction
```

```
>dragtest // lihat hasilnya dan cobalah lakukan!
```

****Gaya Plot 2D**

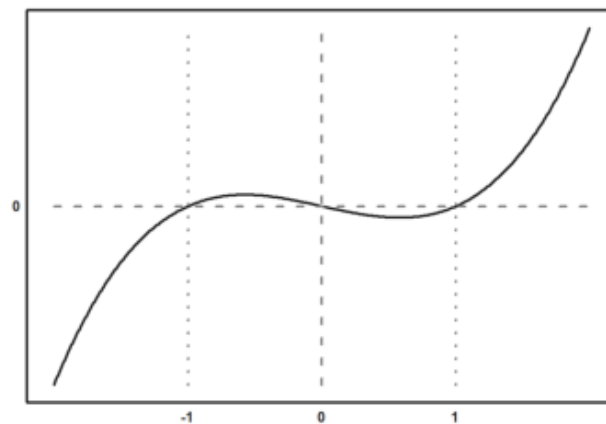
Secara default, EMT menghitung tick sumbu otomatis dan menambahkan label ke setiap tick. Ini dapat diubah dengan parameter grid. Gaya default sumbu dan label dapat diubah. Selain itu, label dan judul dapat ditambahkan secara manual. Untuk menyetel ulang ke gaya default, gunakan reset().

```
>aspect();
>figure(3,4); ...
> figure(1); plot2d("x^3-x",grid=0); ... // tidak ada bingkai kisi atau sumbu
> figure(2); plot2d("x^3-x",grid=1); ... // sumbu x-y
> figure(3); plot2d("x^3-x",grid=2); ... // default ticks
> figure(4); plot2d("x^3-x",grid=3); ... // sumbu x-y dengan label didalamnya
> figure(5); plot2d("x^3-x",grid=4); ... // no ticks, only labels
> figure(6); plot2d("x^3-x",grid=5); ... // default, but no margin
> figure(7); plot2d("x^3-x",grid=6); ... // axes only
> figure(8); plot2d("x^3-x",grid=7); ... // axes only, ticks at axis
> figure(9); plot2d("x^3-x",grid=8); ... // axes only, finer ticks at axis
> figure(10); plot2d("x^3-x",grid=9); ... // default, small ticks inside
> figure(11); plot2d("x^3-x",grid=10); ...// no ticks, axes only
> figure(0):
```



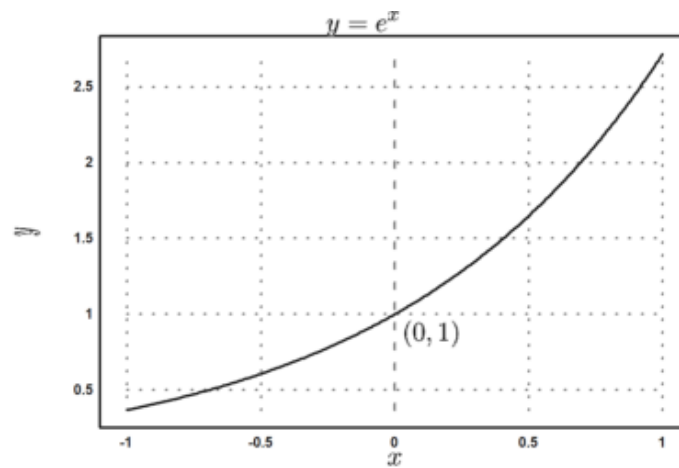
The parameter `<frame` switches off the frame, and `framecolor=blue` sets the frame to a blue color. If you want your own ticks, you can use `style=0`, and add everything later.

```
>aspect(1.5);
>plot2d("x^3-x",grid=0); // plot
>frame; xgrid([-1,0,1]); ygrid(0): // menambahkan bingkai dan kisi
```



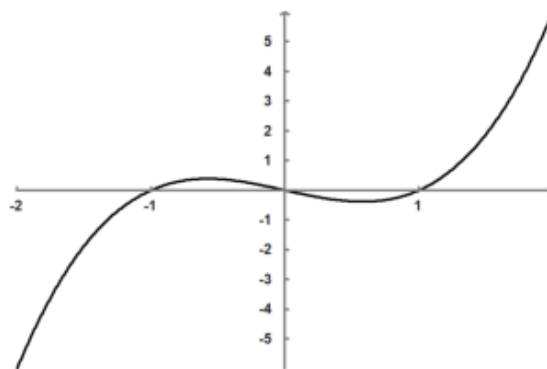
Untuk judul plot dan label sumbu, lihat contoh berikut.

```
>plot2d("exp(x)",-1,1);
>textcolor(black); // mengubah warna teks hitam
>title(latex("y=e^x")); // judul diatas plot
>xlabel(latex("x")); // "x" sumbu x
>ylabel(latex("y"),>vertical); // vertikal "y" untuk sumbu-y
>label(latex("(0,1)"),0,1,color=blue): // memberi label pada suatu titik
```



Sumbu dapat digambar secara terpisah dengan `xaxis()` dan `yaxis()`.

```
>plot2d("x^3-x", <grid, <frame);
>xaxis(0, xx=-2:1, style="->"); yaxis(0, yy=-5:5, style="->"):
```

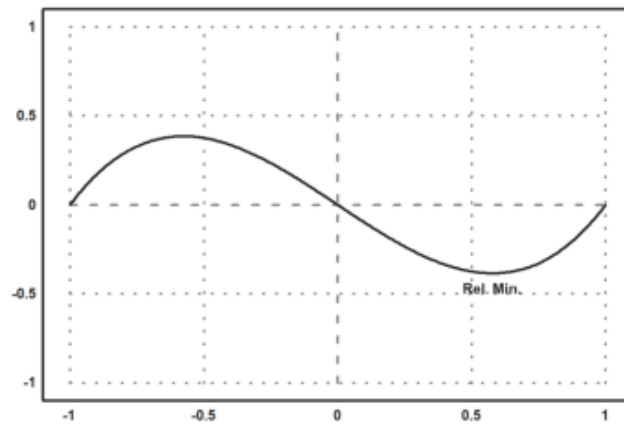


Teks pada plot dapat diatur dengan `label()`. Dalam contoh berikut, "lc" berarti bagian tengah bawah. Ini menetapkan posisi label relatif terhadap koordinat plot.

```
>function f(x) &= x^3-x
```

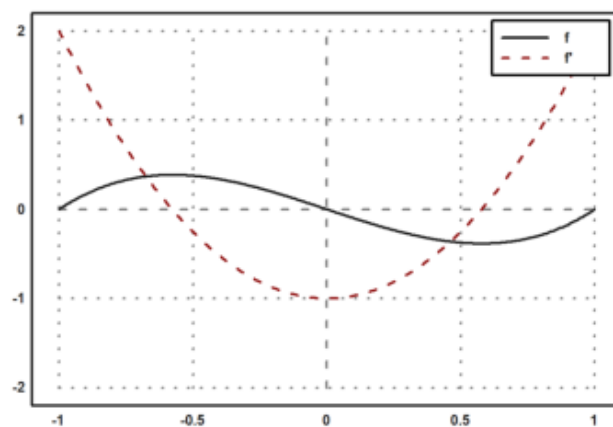
$$x^3 - x$$

```
>plot2d(f, -1, 1, >square);
>x0=fmin(f, 0, 1); // menghitung titik minimum
>label("Rel. Min.", x0, f(x0), pos="lc"): // menambahkan label
```

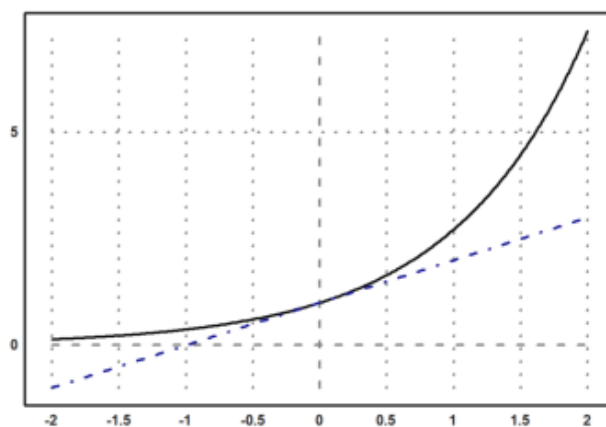


Ada juga kotak teks.

```
>plot2d(&f(x),-1,1,-2,2); // fungsi f(x)
>plot2d(&diff(f(x),x),>add,style="--",color=red); // turunan
>labelbox(["f","f'"],["-","--"],[black,red]): // kotak label
```



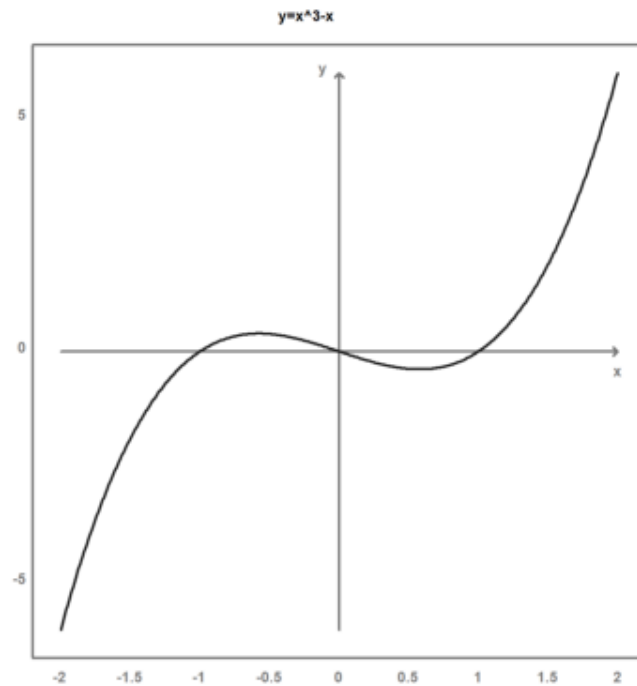
```
>plot2d(["exp(x)","1+x"],color=[black,blue],style=["-","-."]):
```



```

> gridstyle("-", color=gray, textcolor=gray, framecolor=gray); ...
> plot2d("x^3-x", grid=1); ...
> setttitle("y=x^3-x", color=black); ...
> label("x", 2, 0, pos="bc", color=gray); ...
> label("y", 0, 6, pos="cl", color=gray); ...
> reset():

```



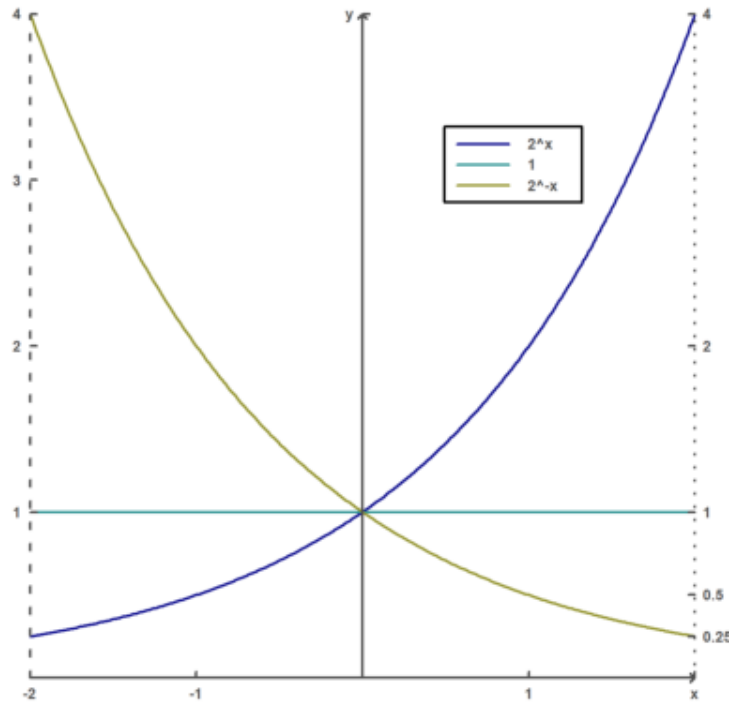
Untuk kontrol lebih lanjut, sumbu x dan sumbu y dapat dilakukan secara manual.

Perintah `fullwindow()` memperluas jendela plot karena kita tidak lagi memerlukan tempat untuk label di luar jendela plot. Gunakan `shrinkwindow()` atau `reset()` untuk menyetel ulang ke default.

```

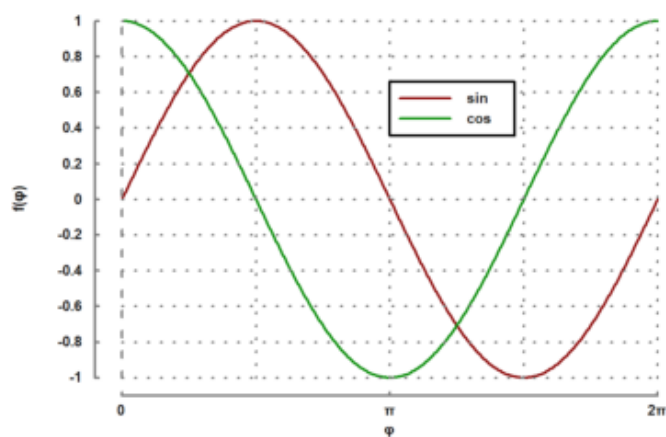
> fullwindow; ...
> gridstyle(color=darkgray, textcolor=darkgray); ...
> plot2d(["2^x", "1", "2^(-x)"], a=-2, b=2, c=0, d=4, <grid, color=4:6, <frame); ...
> xaxis(0, -2:1, style="->"); xaxis(0, 2, "x", <axis); ...
> yaxis(0, 4, "y", style="->"); ...
> yaxis(-2, 1:4, >left); ...
> yaxis(2, 2^(-2:2), style=".", <left); ...
> labelbox(["2^x", "1", "2^-x"], colors=4:6, x=0.8, y=0.2); ...
> reset:

```

Berikut adalah contoh lain, di mana string Unicode digunakan dan sumbunya berada di luar area plot.

```
>aspect(1.5);
>plot2d(["sin(x)", "cos(x)"], 0, 2pi, color=[red, green], <grid, <frame); ...
> xaxis(-1.1, (0:2)*pi, xt=["0", u"&pi;", u"2&pi;"], style="-", >ticks, >zero); ...
> xgrid((0:0.5:2)*pi, <ticks); ...
> yaxis(-0.1*pi, -1:0.2:1, style="-", >zero, >grid); ...
> labelbox(["sin", "cos"], colors=[red, green], x=0.5, y=0.2, >left); ...
> xlabel(u"&phi;"); ylabel(u"f(&phi;)"):
```



Merencanakan Data 2D

Jika x dan y adalah vektor data, maka data tersebut akan digunakan sebagai koordinat x dan y pada suatu kurva. Dalam hal ini, a , b , c , dan d , atau radius r dapat ditentukan, atau jendela plot akan menyesuaikan secara otomatis dengan data. Alternatifnya, `>persegi` dapat diatur untuk mempertahankan rasio aspek persegi.

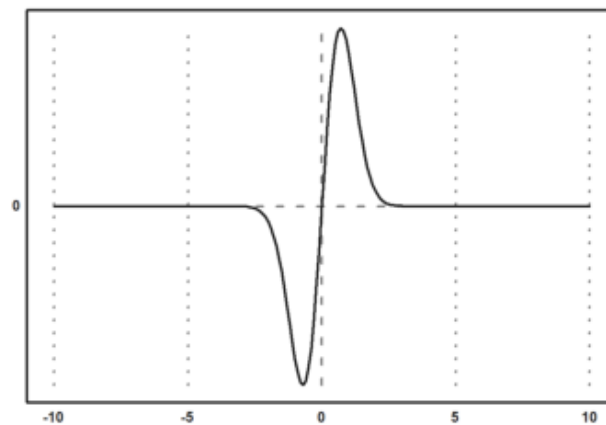
Merencanakan ekspresi hanyalah singkatan dari plot data. Untuk plot data, Anda memerlukan satu atau beberapa baris nilai x, dan satu atau beberapa baris nilai y. Dari rentang dan nilai x, fungsi plot2d akan menghitung data yang akan diplot, secara default dengan evaluasi fungsi yang adaptif. Untuk plot titik gunakan ">titik", untuk garis dan titik campuran gunakan ">addpoints".

Tapi Anda bisa memasukkan data secara langsung.

- Gunakan vektor baris untuk x dan y untuk satu fungsi.
- Matriks untuk x dan y diplot baris demi baris.

Berikut adalah contoh dengan satu baris untuk x dan y.

```
>x=-10:0.1:10; y=exp(-x^2)*x; plot2d(x,y):
```

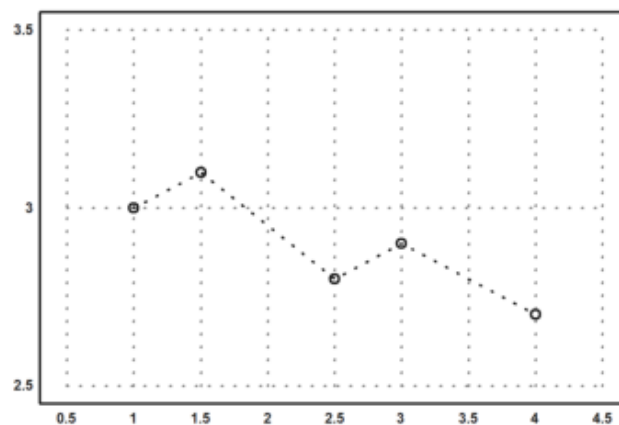


Data juga dapat diplot sebagai poin. Gunakan points=true untuk ini. Plotnya berfungsi seperti poligon, tetapi hanya menggambar sudutnya saja.

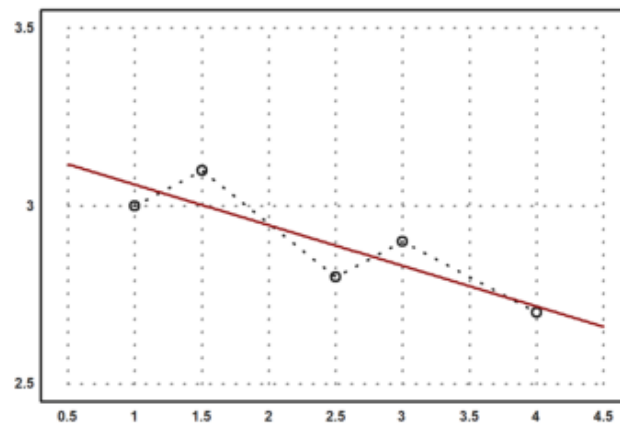
- style="...": Pilih dari "[", "<>", "o", ":", "...", "+", "*", "[]", "<>", "o", "...", "", "|".

Untuk memplot kumpulan titik, gunakan >titik. Jika warna merupakan vektor warna, masing-masing titik mendapat warna berbeda. Untuk matriks koordinat dan vektor kolom, warna diterapkan pada baris matriks. Parameter >addpoints menambahkan titik ke segmen garis untuk plot data.

```
>xdata=[1,1.5,2.5,3,4]; ydata=[3,3.1,2.8,2.9,2.7]; // data
>plot2d(xdata,ydata,a=0.5,b=4.5,c=2.5,d=3.5,style="."); // garis
>plot2d(xdata,ydata,>points,>add,style="o"): // menambahkan titik
```



```
>p=polyfit(xdata,ydata,1); // mendapatkan garis regresi
>plot2d("polyval(p,x)",>add,color=red): // tambahkan plot garis
```



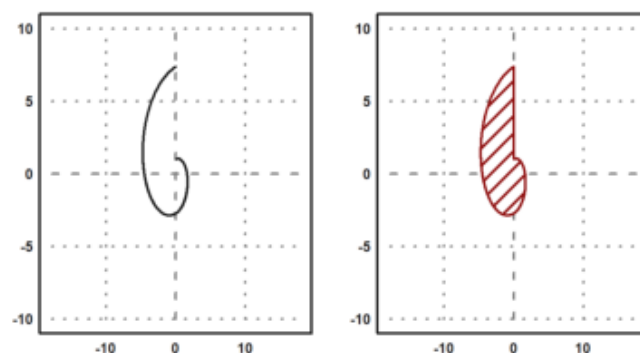
Menggambar Daerah Yang Dibatasi Kurva

Plot data sebenarnya berbentuk poligon. Kita juga dapat memplot kurva atau kurva terisi.

- terisi=benar mengisi plot.
- style="...": Pilih dari "", "/", "\", "\/".
- Fillcolor : Lihat di atas untuk mengetahui warna yang tersedia.

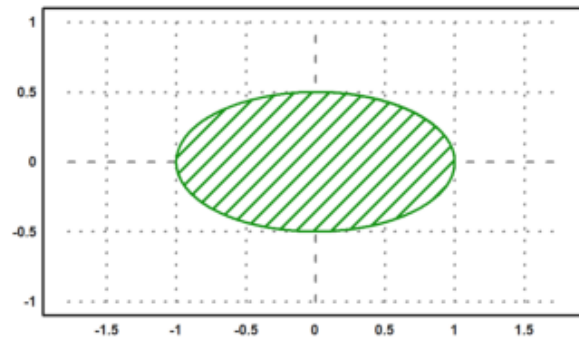
Warna isian ditentukan oleh argumen "fillcolor", dan pada <outline opsional, mencegah menggambar batas untuk semua gaya kecuali gaya default.

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

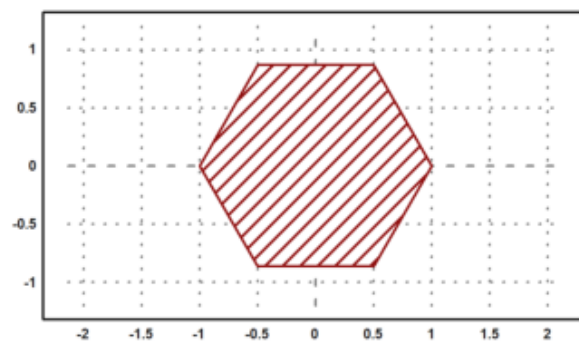


Dalam contoh berikut kita memplot elips terisi dan dua segi enam terisi menggunakan kurva tertutup dengan 6 titik dengan gaya isian berbeda.

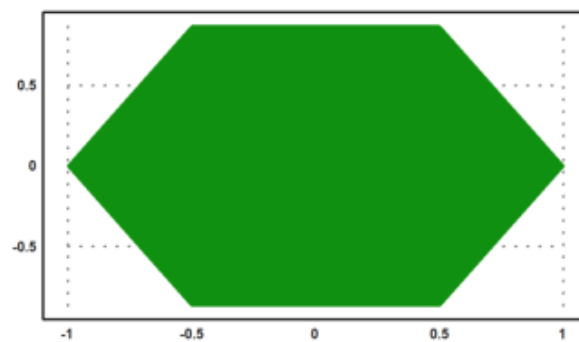
```
>x=linspace(0,2pi,1000); plot2d(sin(x),cos(x)*0.5,r=1,>filled,style="/"):
```



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

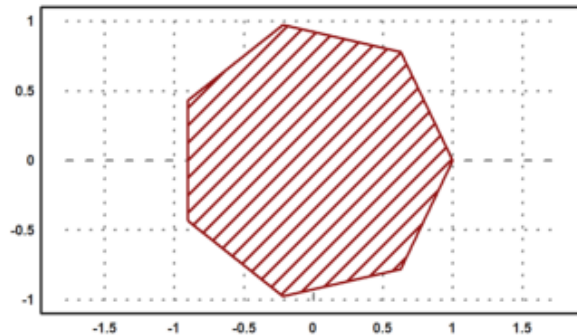


```
>t=linspace(0,2pi,6); plot2d(cos(t),sin(t),>filled,style="#"):
```



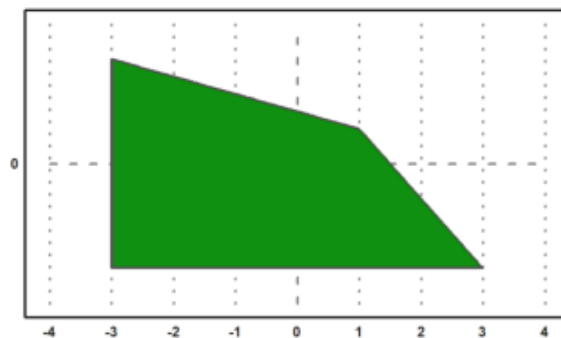
Contoh lainnya adalah septagon yang kita buat dengan 7 titik pada lingkaran satuan.

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



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

```
>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):
```

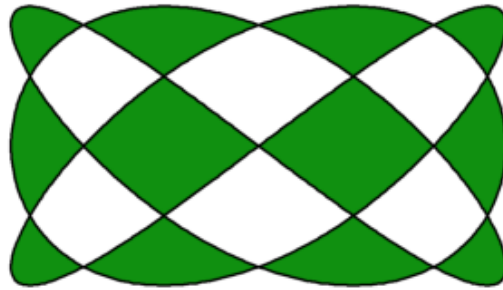


Poin utama dari bahasa matriks adalah memungkinkan pembuatan tabel fungsi dengan mudah.

```
>t=linspace(0,2pi,1000); x=cos(3*t); y=sin(4*t);
```

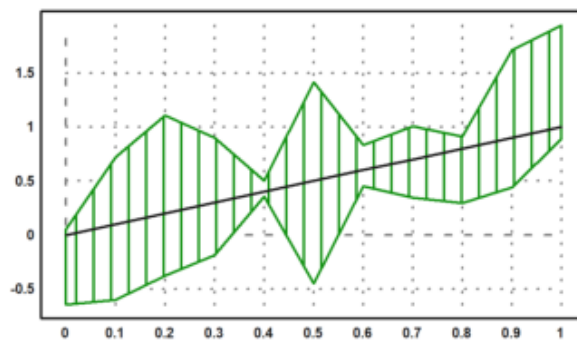
Kami sekarang memiliki nilai vektor x dan y . `plot2d()` dapat memplot nilai-nilai ini sebagai kurva yang menghubungkan titik-titik tersebut. Plotnya bisa diisi. Pada kasus ini memberikan hasil yang bagus karena aturan belitan, yang digunakan untuk isi.

```
>plot2d(x,y,<grid,<frame,>filled):
```



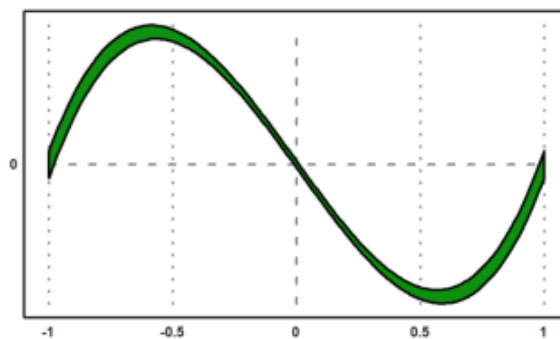
Vektor interval diplot terhadap nilai x sebagai wilayah terisi antara nilai interval yang lebih rendah dan lebih tinggi. Hal ini dapat berguna untuk memplot kesalahan perhitungan. Tapi itu bisa juga dapat digunakan untuk memplot kesalahan statistik.

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



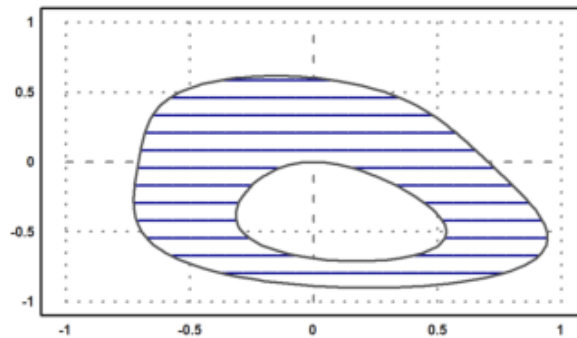
Jika x adalah vektor yang diurutkan, dan y adalah vektor interval, maka plot2d akan memplot rentang interval yang terisi pada bidang. Gaya isianya sama dengan gaya poligon.

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



Dimungkinkan untuk mengisi wilayah nilai untuk fungsi tertentu. Untuk ini, level harus berupa matriks 2xn. Baris pertama adalah batas bawah dan baris kedua berisi batas atas.

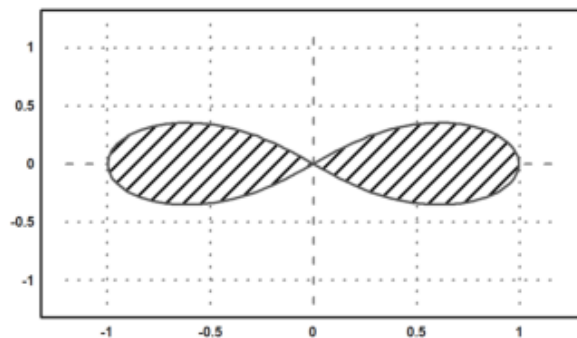
```
>expr := "2*x^2+x*y+3*y^4+y"; // mendefinisikan ekspresi f(x,y)
>plot2d(expr,level=[0;1],style="-",color=blue): // 0 <= f(x,y) <= 1
```



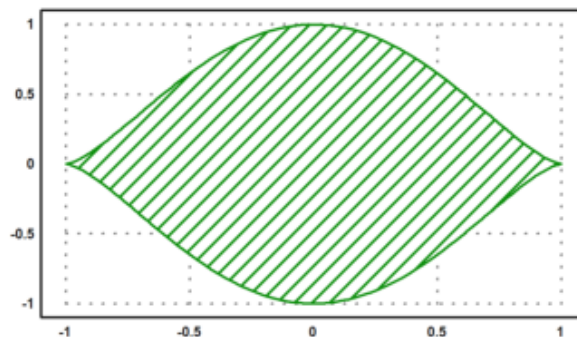
Kita juga dapat mengisi rentang nilai seperti

$$-1 \leq (x^2 + y^2)^2 - x^2 + y^2 \leq 0.$$

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



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



Grafik Fungsi Parametrik

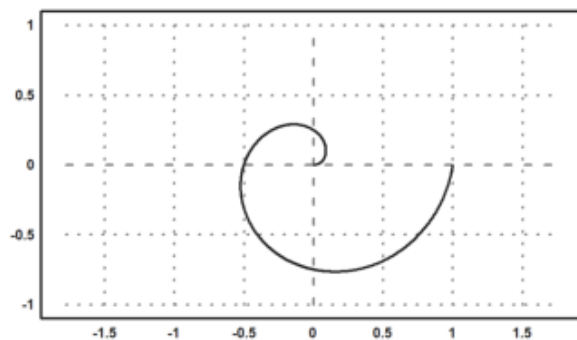
Nilai x tidak perlu diurutkan. (x,y) hanya menggambarkan sebuah kurva. Jika x diurutkan, kurva tersebut merupakan grafik suatu fungsi.

Dalam contoh berikut, kita memplot spiral

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

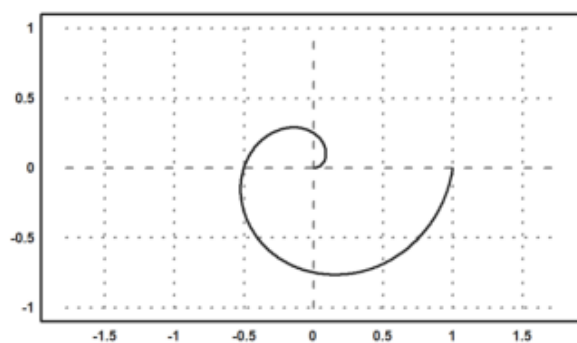
Kita perlu menggunakan banyak titik untuk tampilan yang halus atau fungsi adaptif() untuk mengevaluasi ekspresi (lihat fungsi adaptif() untuk lebih jelasnya).

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

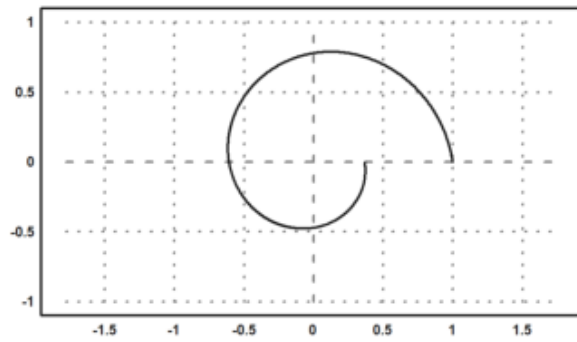


Sebagai alternatif, dimungkinkan untuk menggunakan dua ekspresi untuk kurva. Berikut ini plot kurva yang sama seperti di atas.

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



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

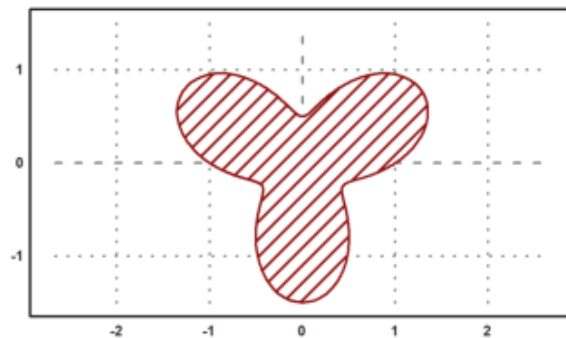
In the next example, we plot the curve

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

with

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

```
>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):
```



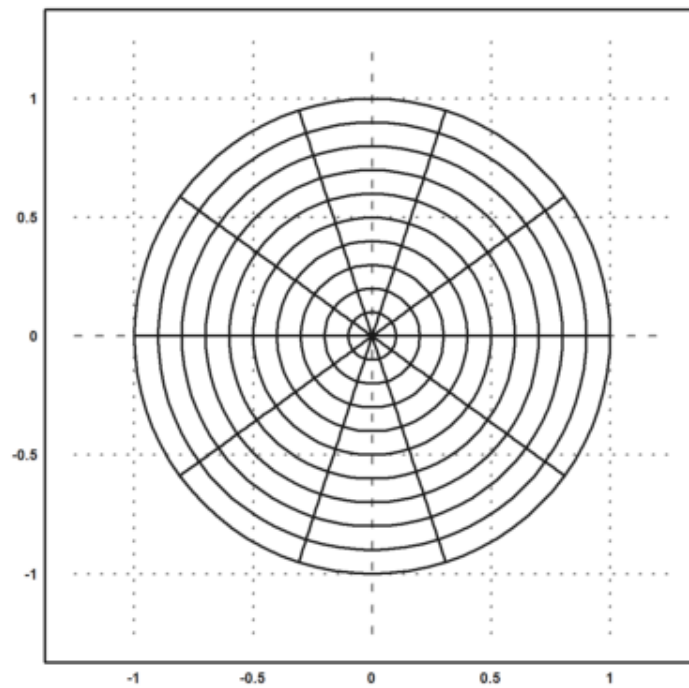
Menggambar Grafik Bilangan Kompleks

Serangkaian bilangan kompleks juga dapat diplot. Kemudian titik-titik grid akan dihubungkan. Jika sejumlah garis kisi ditentukan (atau vektor garis kisi berukuran 1x2) dalam argumen `cgrid`, hanya garis kisi tersebut yang terlihat.

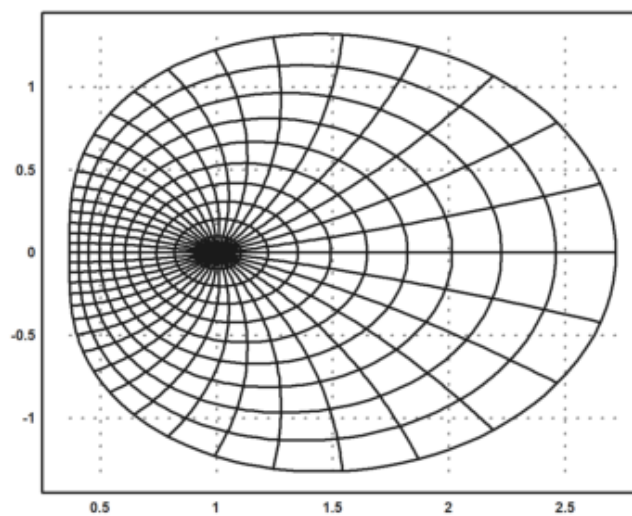
Matriks bilangan kompleks secara otomatis akan diplot sebagai kisi-kisi pada bidang kompleks.

Pada contoh berikut, kita memplot gambar lingkaran satuan di bawah fungsi eksponensial. Parameter `cgrid` menyembunyikan beberapa kurva grid.

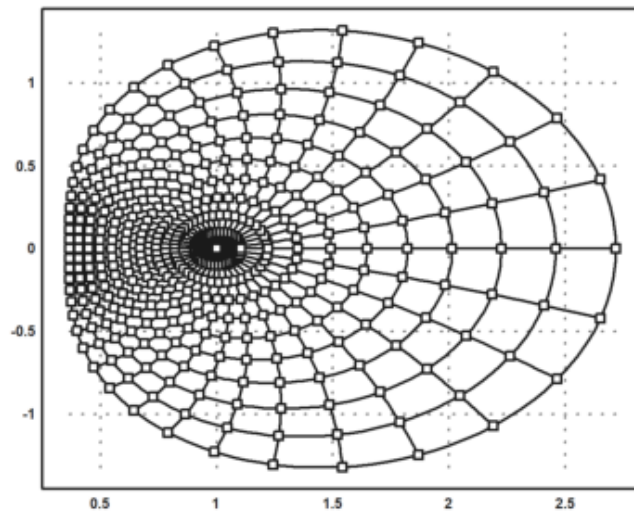
```
>aspect(); r=linspace(0,1,50); a=linspace(0,2pi,80)'; z=r*exp(I*a);...
>plot2d(z,a=-1.25,b=1.25,c=-1.25,d=1.25,cgrid=10):
```



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



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

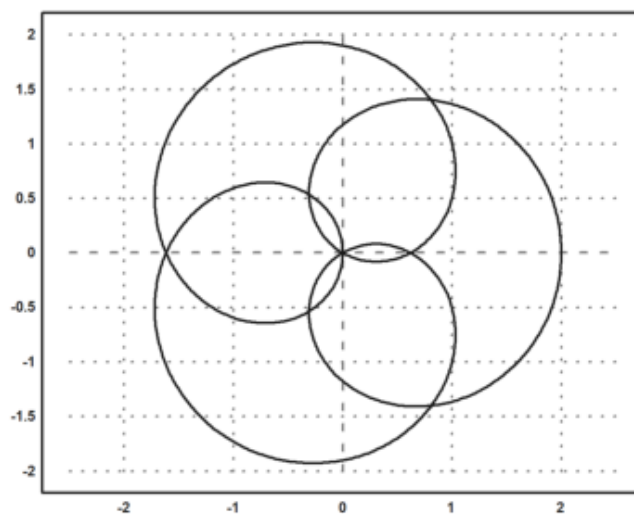


Vektor bilangan kompleks secara otomatis diplot sebagai kurva pada bidang kompleks dengan bagian nyata dan bagian imajiner.

Dalam contoh, kita memplot lingkaran satuan dengan

$$\gamma(t) = e^{it}$$

```
>t=linspace(0,2pi,1000); ...
>plot2d(exp(I*t)+exp(4*I*t),r=2):
```



Plot Statistik

Ada banyak fungsi yang dikhususkan pada plot statistik. Salah satu plot yang sering digunakan adalah plot kolom.

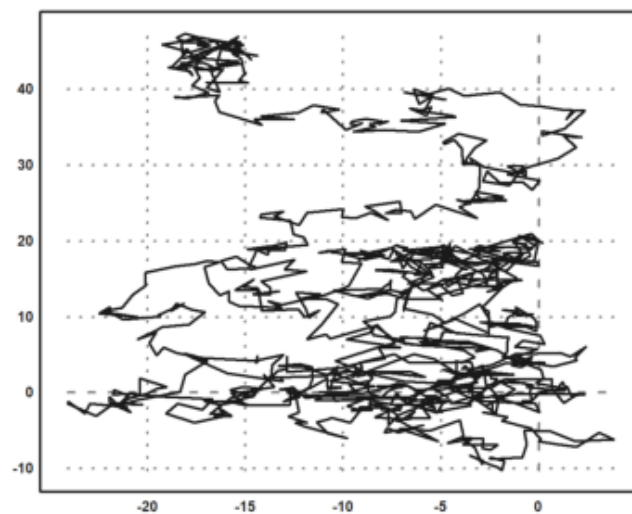
Jumlah kumulatif dari nilai terdistribusi normal 0-1 menghasilkan jalan acak.

```
>plot2d(cumsum(randnormal(1,1000))):
```

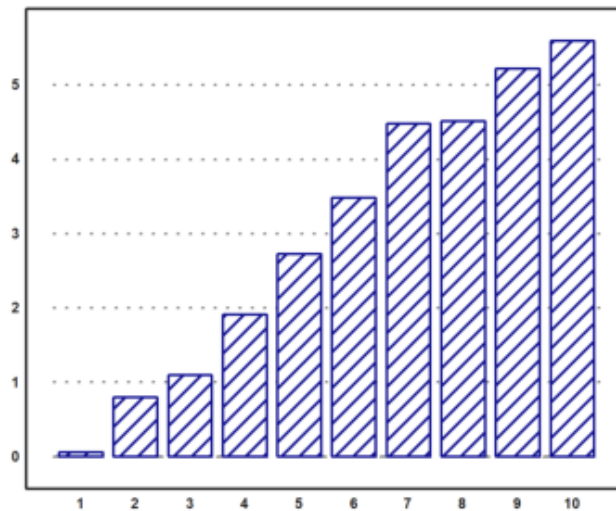


Penggunaan dua baris menunjukkan jalan dalam dua dimensi.

```
>X=cumsum(randnormal(2,1000)); plot2d(X[1],X[2]):
```

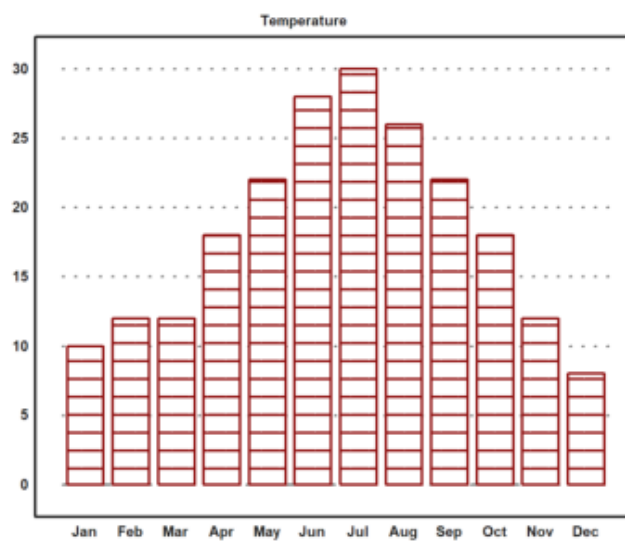


```
>columnplot(cumsum(random(10)),style="/",color=blue):
```

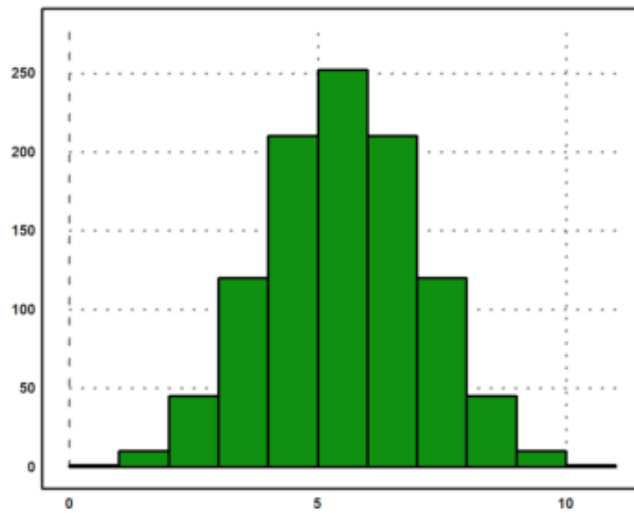


Itu juga dapat menampilkan string sebagai label.

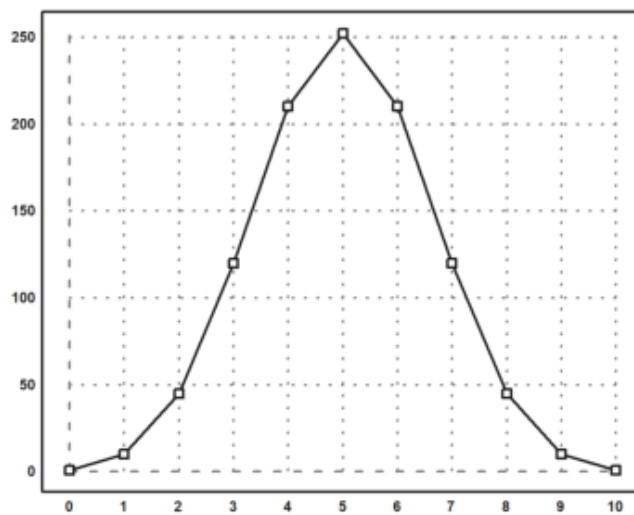
```
>months=["Jan", "Feb", "Mar", "Apr", "May", "Jun", ...
> "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];
>values=[10,12,12,18,22,28,30,26,22,18,12,8];
>columnsplot(values,lab=months,color=red,style="-");
>title("Temperature"):
```



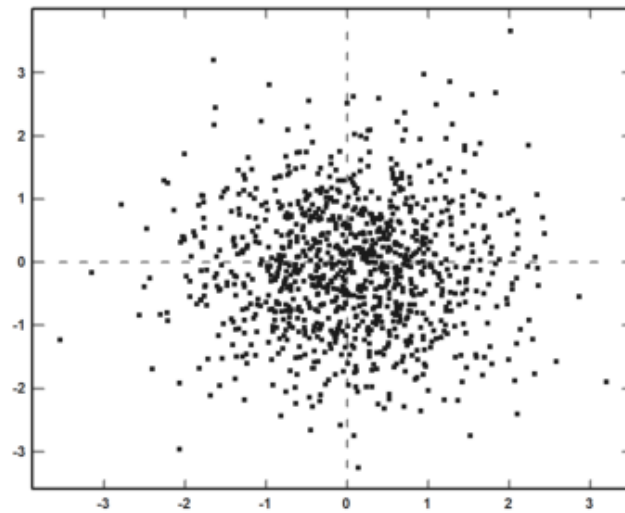
```
>k=0:10;
>plot2d(k,bin(10,k),>bar):
```



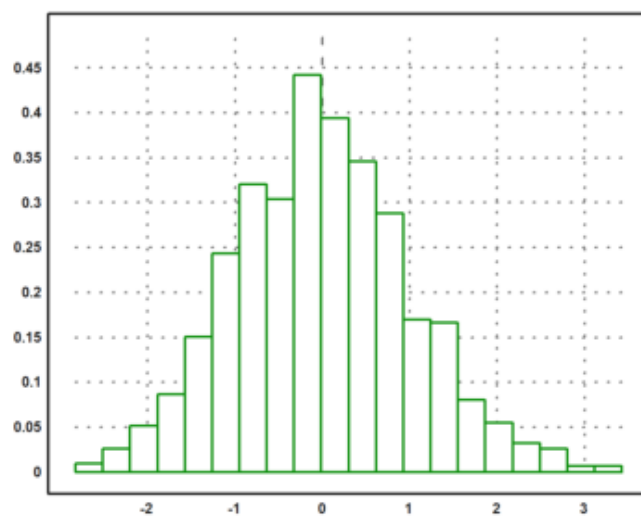
```
>plot2d(k,bin(10,k)); plot2d(k,bin(10,k),>points,>add):
```



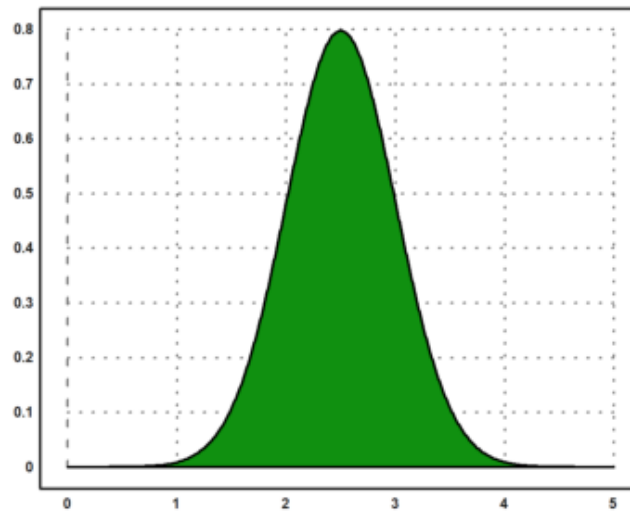
```
>plot2d(normal(1000),normal(1000),>points,grid=6,style=".."):
```



```
>plot2d(normal(1,1000),>distribution,style="O"):
```

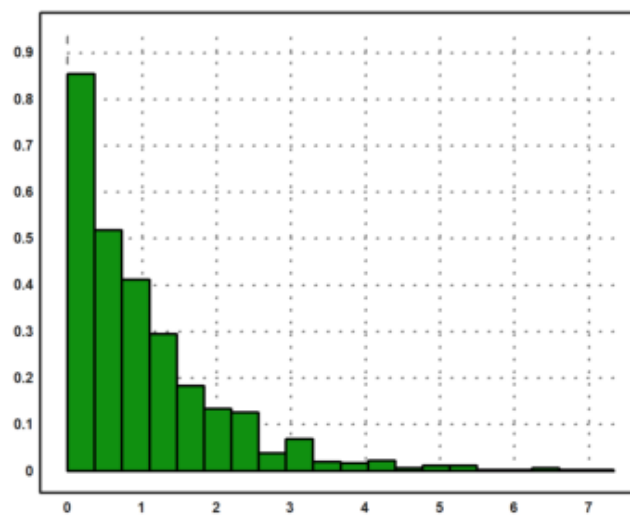


```
>plot2d("qnormal",0,5;2.5,0.5,>filled):
```



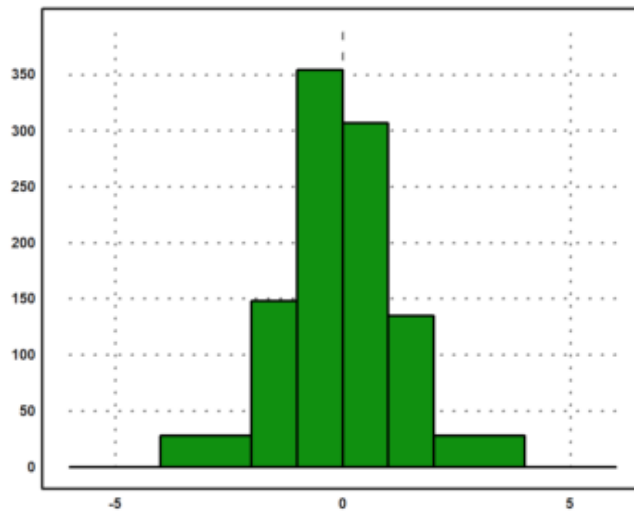
Untuk memplot distribusi statistik eksperimental, Anda dapat menggunakan `distribution=n` dengan `plot2d`.

```
>w=randexponential(1,1000); // distribusi eksponen
>plot2d(w,>distribution): // atau distribusi=n dengan n adalah interval
```



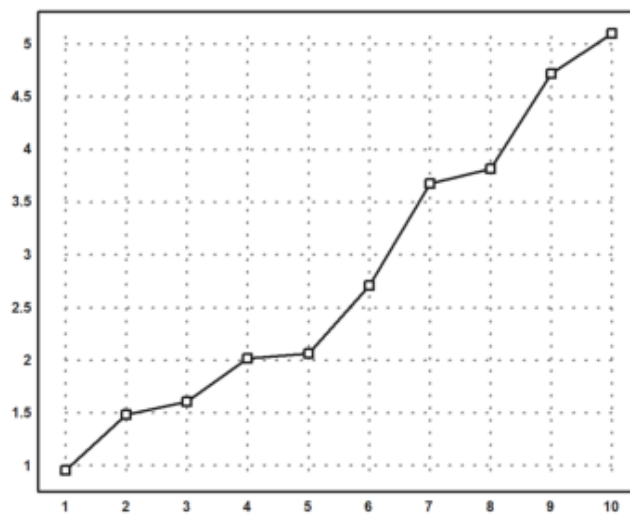
Atau Anda dapat menghitung distribusi dari data dan memplot hasilnya dengan `>bar` di `plot3d`, atau dengan `plot kolom`.

```
>w=normal(1000); // distribusi normal 0-1
>{x,y}=histo(w,10,v=[-6,-4,-2,-1,0,1,2,4,6]); // batas interval v
>plot2d(x,y,>bar):
```

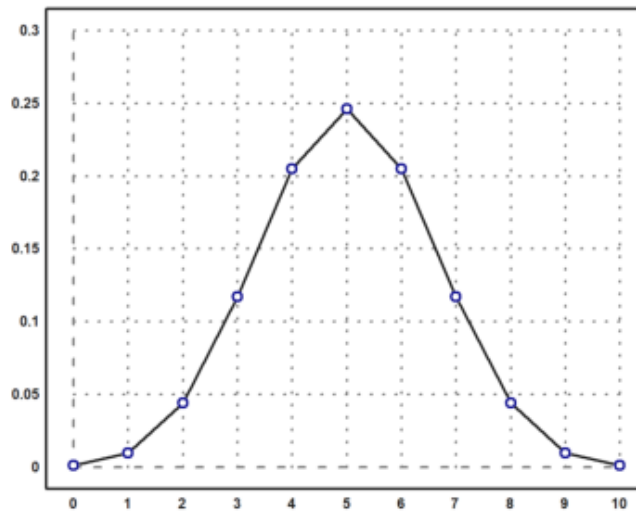



Fungsi `statplot()` mengatur gaya dengan string sederhana.

```
>statplot(1:10,cumsum(random(10)),"b"):
```



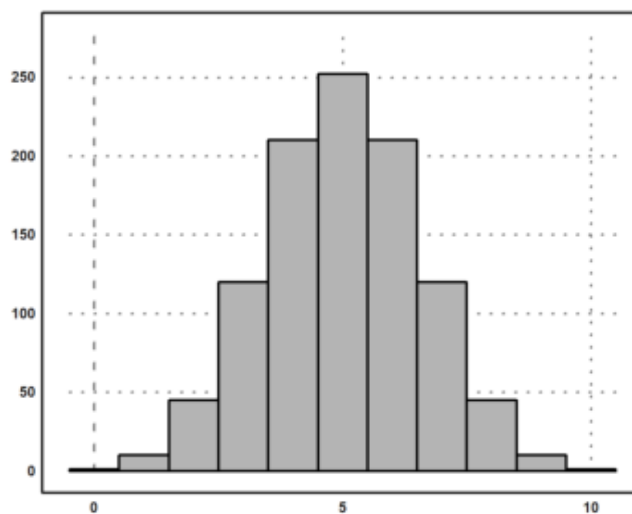
```
>n=10; i=0:n; ...
>plot2d(i,bin(n,i)/2^n,a=0,b=10,c=0,d=0.3); ...
>plot2d(i,bin(n,i)/2^n,points=true,style="ow",add=true,color=blue):
```



Selain itu, data dapat diplot sebagai batang. Dalam hal ini, x harus diurutkan dan satu elemen lebih panjang dari y. Batangnya akan memanjang dari $x[i]$ hingga $x[i+1]$ dengan nilai $y[i]$. Jika x berukuran sama dengan y, maka x akan diperpanjang satu elemen dengan spasi terakhir.

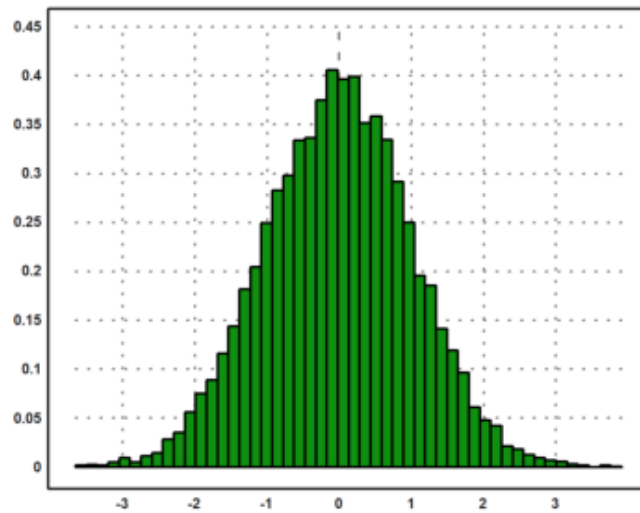
Gaya isian dapat digunakan seperti di atas.

```
>n=10; k=bin(n,0:n); ...
>plot2d(-0.5:n+0.5,k,bar=true,fillcolor=lightgray):
```

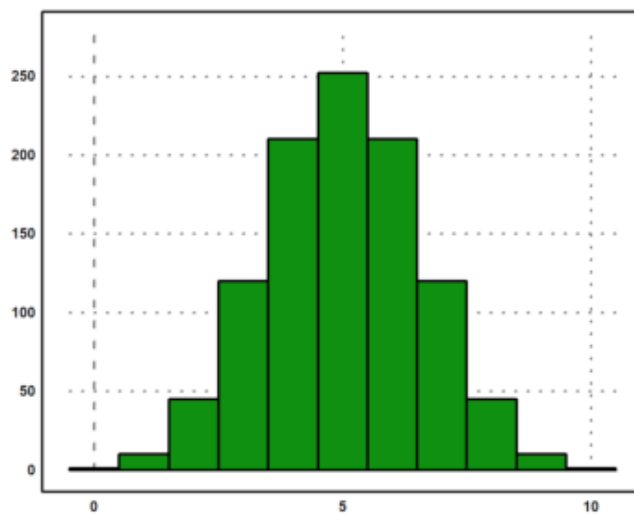


Data untuk plot batang (batang=1) dan histogram (histogram=1) dapat diberikan secara eksplisit dalam xv dan yv, atau dapat dihitung dari distribusi empiris dalam xv dengan `>distribusi` (atau `distribusi=n`). Histogram nilai xv akan dihitung secara otomatis dengan `>histogram`. Jika `>even` ditentukan, nilai xv akan dihitung dalam interval bilangan bulat.

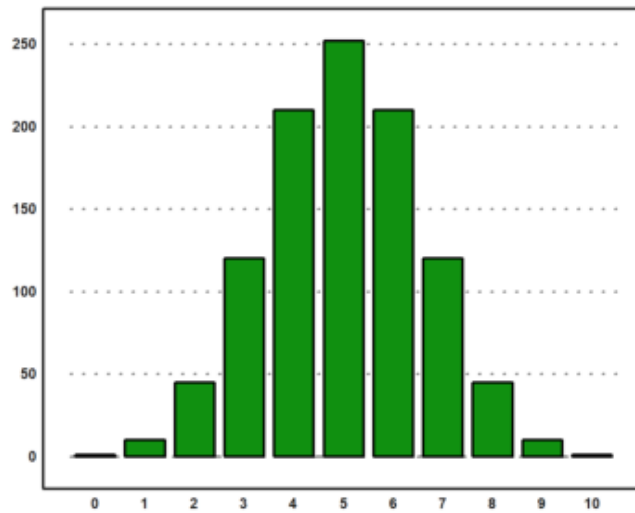
```
>plot2d(normal(10000),distribution=50):
```



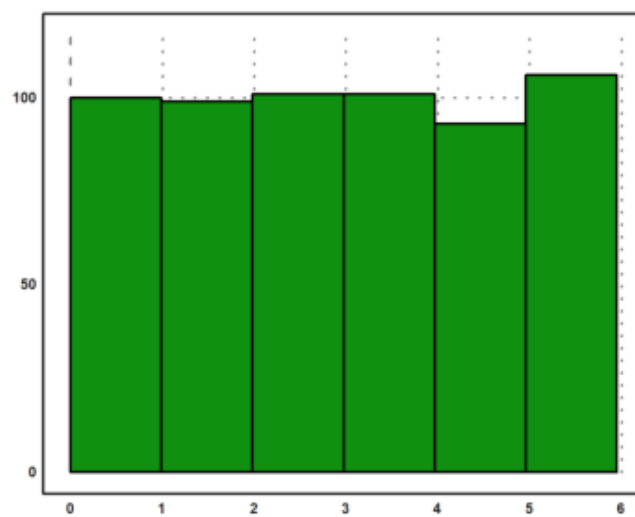
```
>k=0:10; m=bin(10,k); x=(0:11)-0.5; plot2d(x,m,>bar):
```



```
>columnsplo(m,k):
```

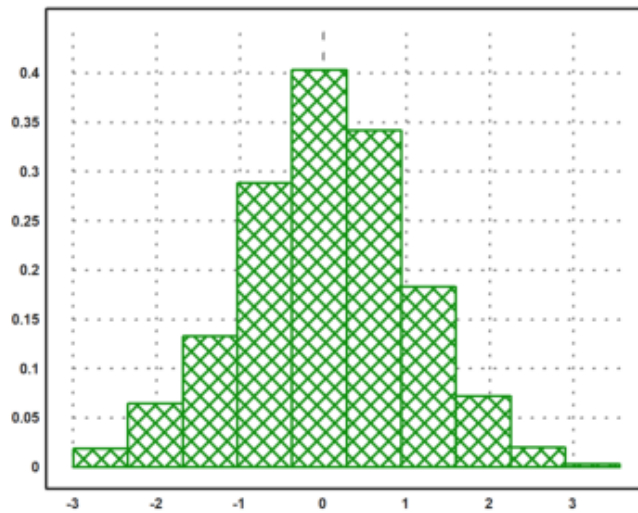


```
>plot2d(random(600)*6,histogram=6):
```



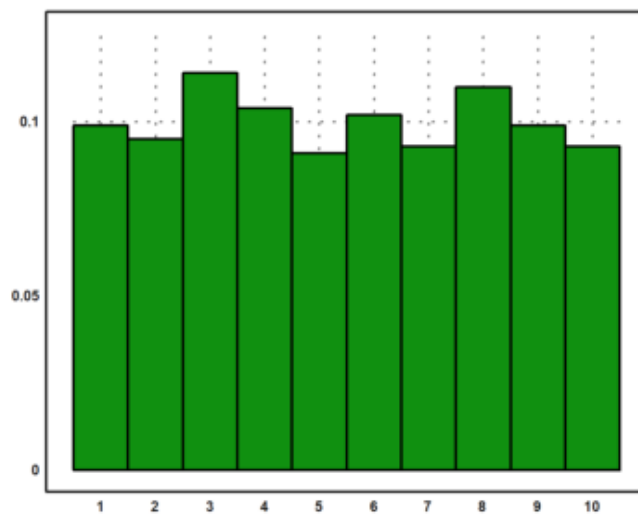
Untuk distribusi, terdapat parameter `distribution=n`, yang menghitung nilai secara otomatis dan mencetak distribusi relatif dengan `n` sub-interval.

```
>plot2d(normal(1,1000),distribution=10,style="\/"): 
```



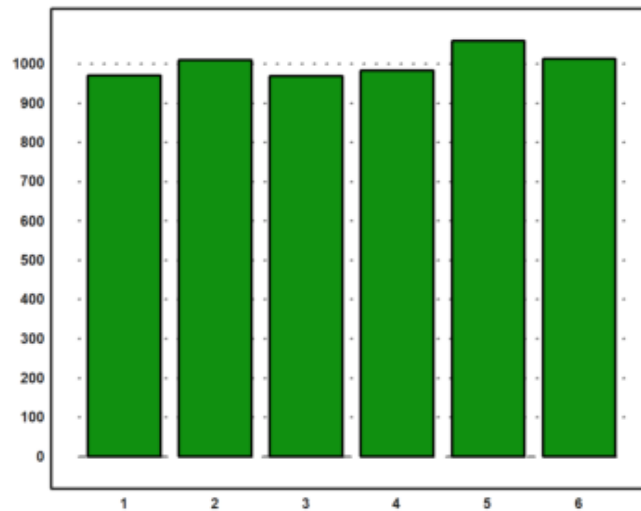
Dengan parameter `even=true`, ini akan menggunakan interval bilangan bulat.

```
>plot2d(intrandom(1,1000,10),distribution=10,even=true):
```

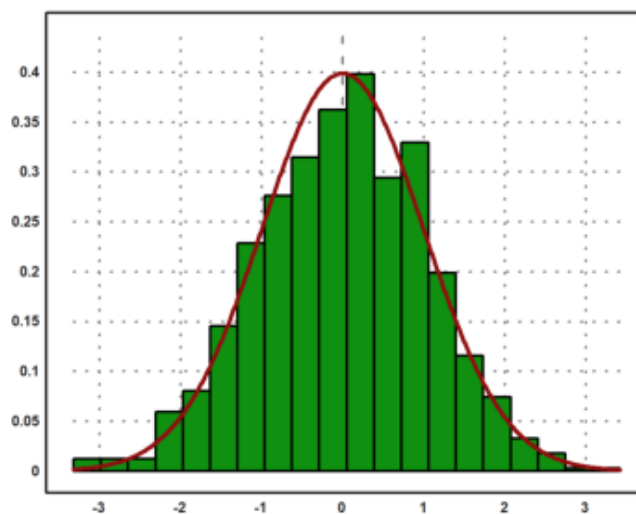


Perhatikan bahwa ada banyak plot statistik yang mungkin berguna. Silahkan lihat tutorial tentang statistik.

```
>columnplot(getmultiplicities(1:6,intrandom(1,6000,6))):
```

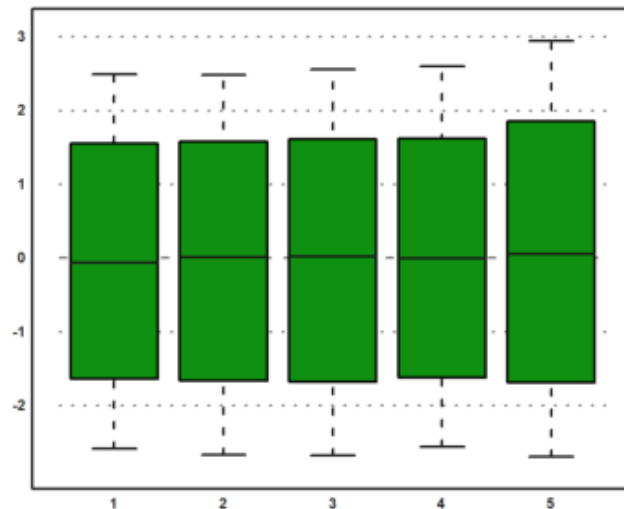


```
>plot2d(normal(1,1000),>distribution); ...
> plot2d("qnormal(x)",color=red,thickness=2,>add):
```



Ada juga banyak plot khusus untuk statistik. Plot kotak menunjukkan kuartil distribusi ini dan banyak outlier. Menurut definisinya, outlier dalam plot kotak adalah data yang melebihi 1,5 kali rentang 50% tengah plot.

```
>M=normal(5,1000); boxplot(quartiles(M)):
```



Fungsi Implisit

Plot implisit menunjukkan penyelesaian garis level $f(x,y)=\text{level}$, dengan "level" dapat berupa nilai tunggal atau vektor nilai. Jika level = "auto", akan ada garis level nc, yang akan tersebar 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 parameter x dan y, atau alternatifnya, xv dapat berupa matriks nilai.

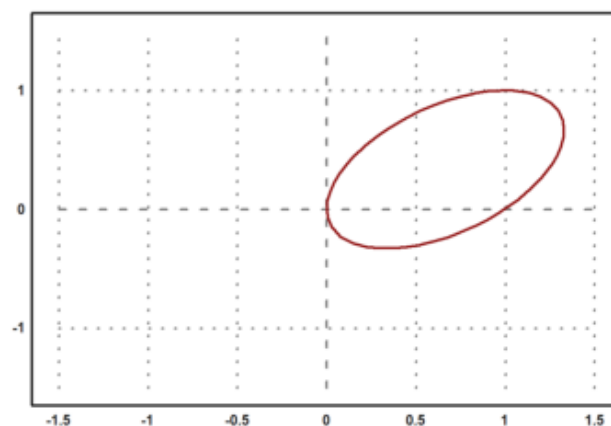
Euler dapat menandai garis level

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

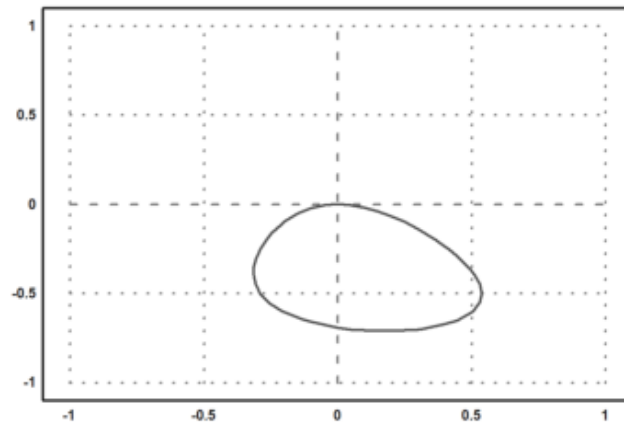
dari fungsi apa pun.

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

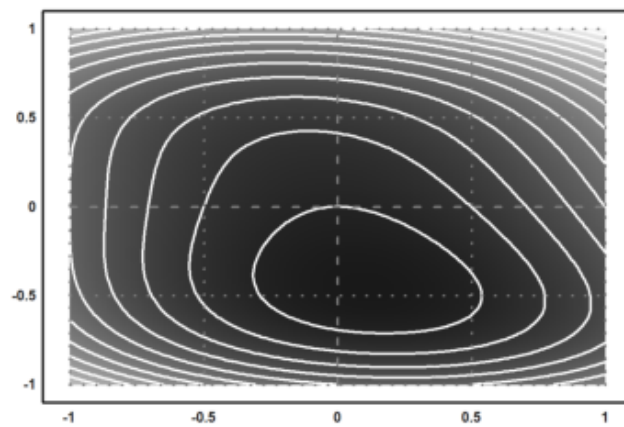
```
>aspect(1.5);
>plot2d("x^2+y^2-x*y-x",r=1.5,level=0,contourcolor=red):
```



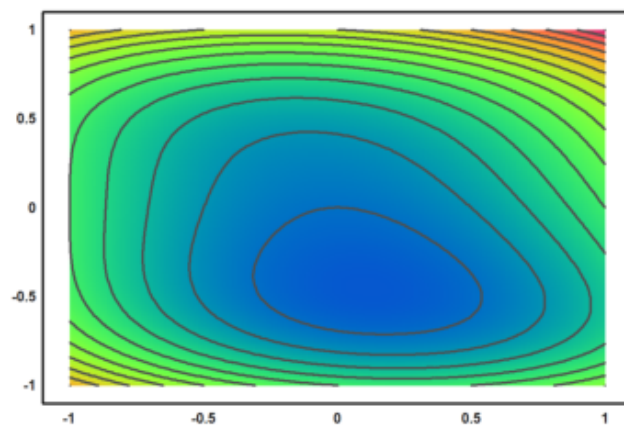
```
>expr := "2*x^2+x*y+3*y^4+y"; // definisi ekspresi f(x,y)
>plot2d(expr,level=0): // Solusi dari f(x,y)=0
```



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

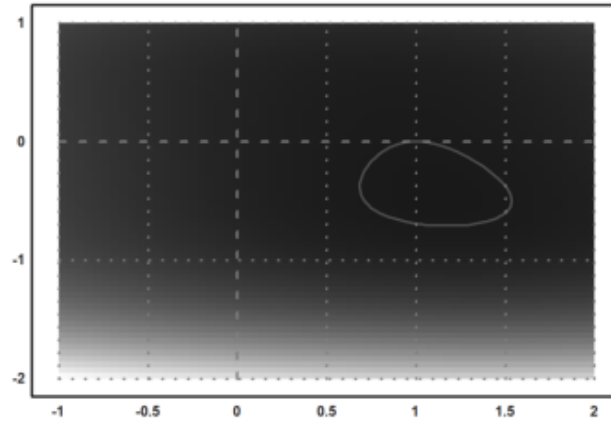


```
>plot2d(expr,level=0:0.5:20,>hue,>spectral,n=200,grid=4): // lebih bagus
```

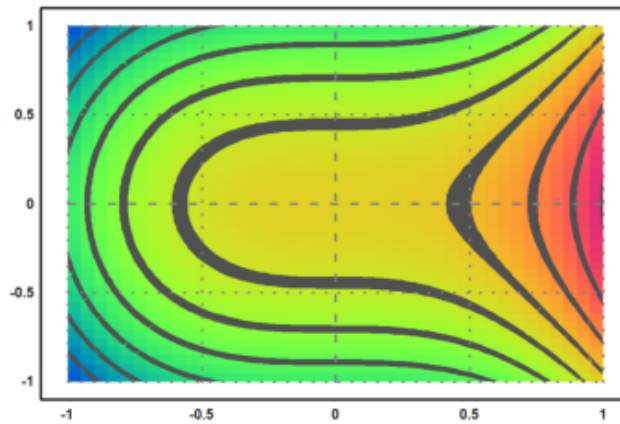


Ini juga berfungsi untuk plot data. Namun Anda harus menentukan rentangnya untuk label sumbu.

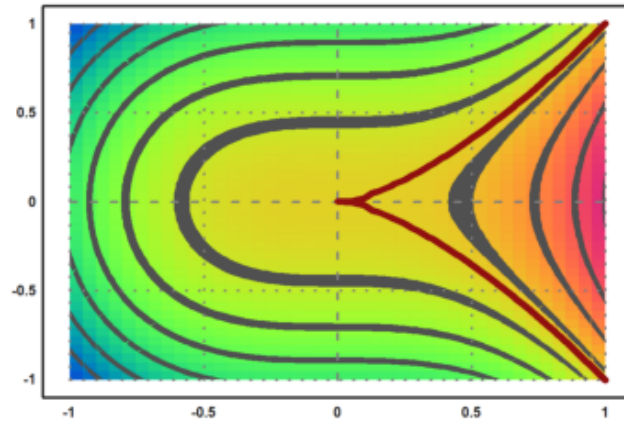
```
>x=-2:0.05:1; y=x'; z=expr(x,y);  
>plot2d(z,level=0,a=-1,b=2,c=-2,d=1,>hue):
```



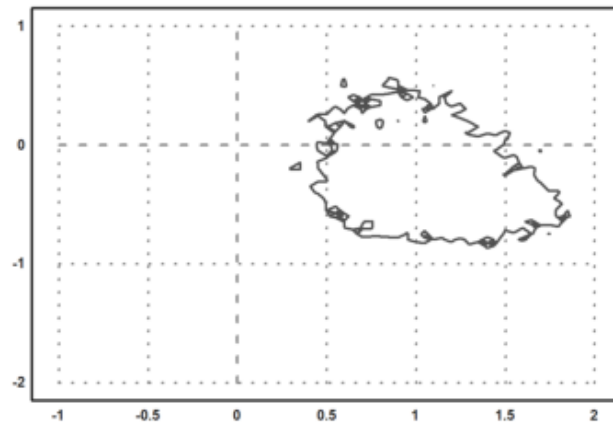
```
>plot2d("x^3-y^2",>contour,>hue,>spectral):
```



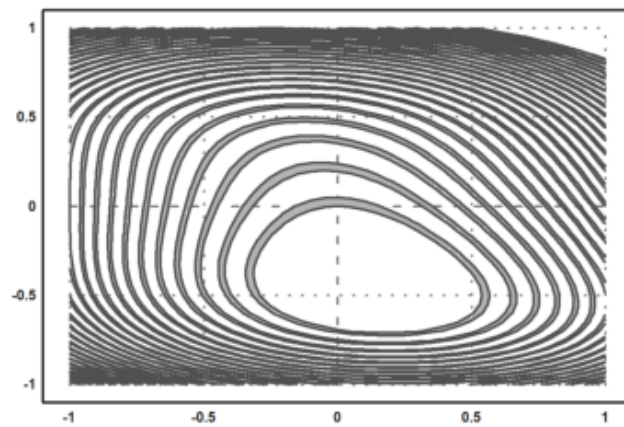
```
>plot2d("x^3-y^2",level=0,contourwidth=3,>add,contourcolor=red):
```



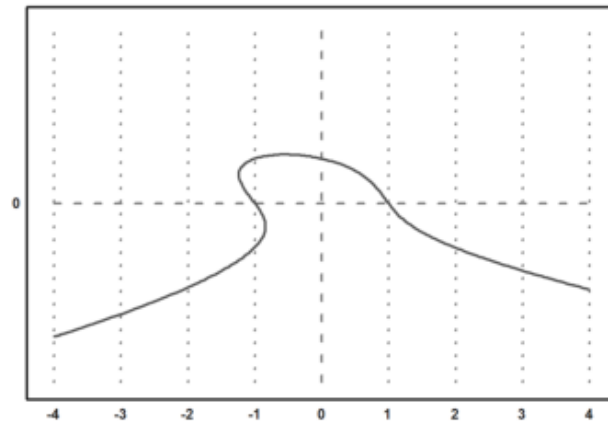
```
>z=z+normal(size(z))*0.2;
>plot2d(z,level=0.5,a=-1,b=2,c=-2,d=1):
```



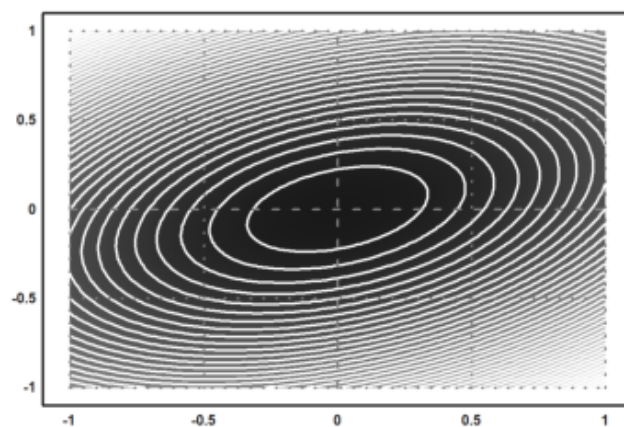
```
>plot2d(expr,level=[0:0.2:5;0.05:0.2:5.05],color=lightgray):
```



```
>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):
```



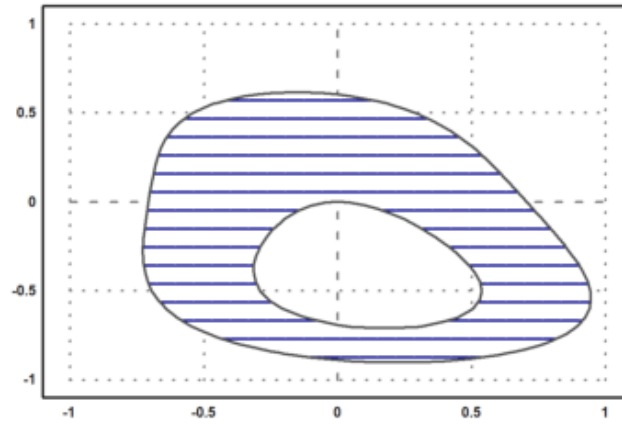
Dimungkinkan juga untuk mengisi set

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

dengan rentang level.

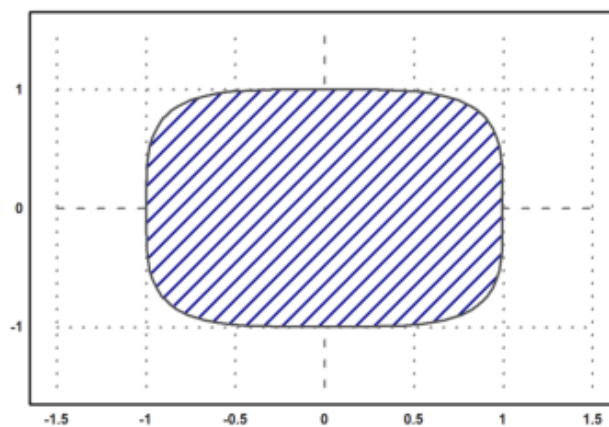
Dimungkinkan untuk mengisi wilayah nilai untuk fungsi tertentu. Untuk ini, level harus berupa matriks 2xn. Baris pertama adalah batas bawah dan baris kedua berisi batas atas.

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

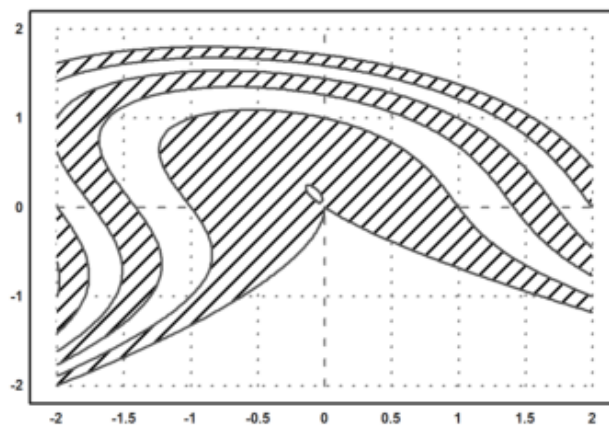


Plot implisit juga dapat menunjukkan rentang level. Maka level harus berupa matriks interval level $2 \times n$, di mana baris pertama berisi awal dan baris kedua berisi akhir setiap interval. Alternatifnya, vektor baris sederhana dapat digunakan untuk level, dan parameter `dl` memperluas nilai level ke interval.

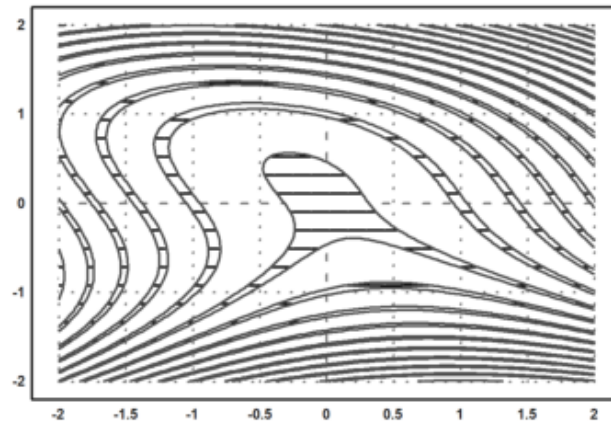
```
>plot2d("x^4+y^4",r=1.5,level=[0;1],color=blue,style="/") :
```



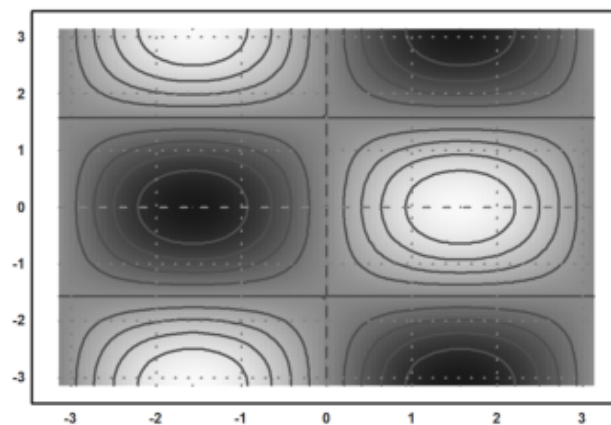
```
>plot2d("x^2+y^3+x*y",level=[0,2,4;1,3,5],style="/",r=2,n=100) :
```



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



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

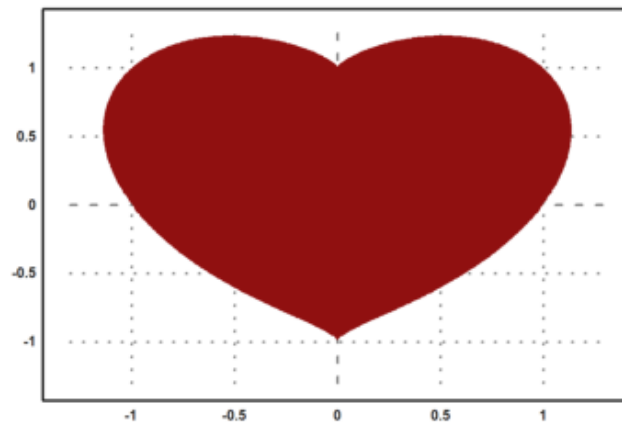


Dimungkinkan juga untuk menandai suatu wilayah

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

Hal ini dilakukan dengan menambahkan level dengan dua baris.

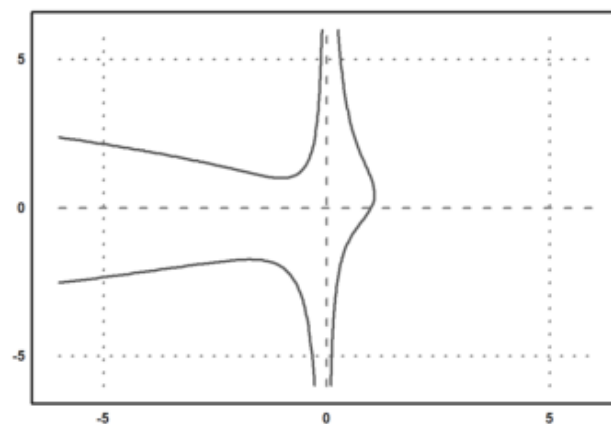
```
>plot2d("(x^2+y^2-1)^3-x^2*y^3", r=1.3, ...  
> style="#", color=red, <outline, ...  
> level=[-2;0], n=100):
```



Dimungkinkan untuk menentukan level tertentu. Misalnya, kita dapat memplot solusi persamaan seperti

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

```
>plot2d("x^3-x*y+x^2*y^2",r=6,level=1,n=100):
```



```
>function starplot1 (v, style="/", color=green, lab=none) ...
```

```

if !holding() then clg; endif;
w=window(); window(0,0,1024,1024);
h=holding(1);
r=max(abs(v))*1.2;
setplot(-r,r,-r,r);
n=cols(v); t=linspace(0,2pi,n);
v=v|v[1]; c=v*cos(t); s=v*sin(t);
cl=barcolor(color); st=barstyle(style);
loop 1 to n
  polygon([0,c[#],c[#+1]], [0,s[#],s[#+1]],1);
  if lab!=none then
    rlab=v[#]+r*0.1;
    {col,row}=toscreen(cos(t[#])*rlab, sin(t[#])*rlab);
    ctext("'" + lab[#], col, row-textheight()/2);
  end
end

```

```

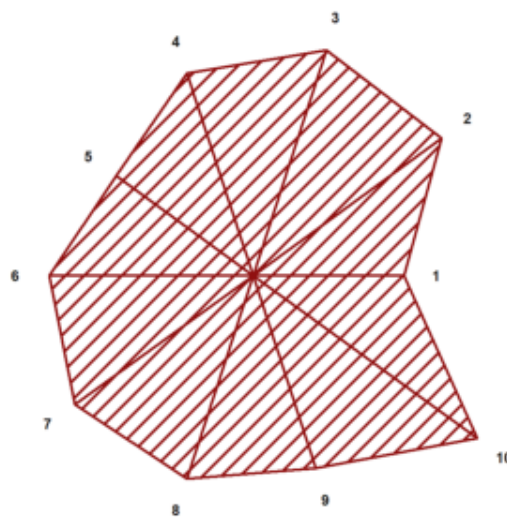
    endif;
end;
barcolor(cl); barstyle(st);
holding(h);
window(w);
endfunction

```

Tidak ada tanda centang kotak atau sumbu di sini. Selain itu, kami menggunakan jendela penuh untuk plot-nya.

Kami memanggil reset sebelum kami menguji plot ini untuk mengembalikan default grafis. Ini tidak perlu dilakukan jika Anda yakin plot Anda berhasil.

```
>reset; starplot1(normal(1,10)+5,color=red,lab=1:10):
```



Terkadang, Anda mungkin ingin merencanakan sesuatu yang plot2d tidak bisa lakukan, tapi hampir.

Dalam fungsi berikut, kita membuat plot impuls logaritmik. plot2d dapat melakukan plot logaritmik, tetapi tidak untuk batang impuls.

```
>function logimpulseplot1 (x,y) ...
```

```

{x0,y0}=makeimpulse(x,log(y)/log(10));
plot2d(x0,y0,>bar,grid=0);
h=holding(1);
frame();
xgrid(ticks(x));
p=plot();
for i=-10 to 10;
    if i<=p[4] and i>=p[3] then
        ygrid(i,yt="10^"+i);
    endif;
end;
end;

```

```

    holding(h);
endfunction

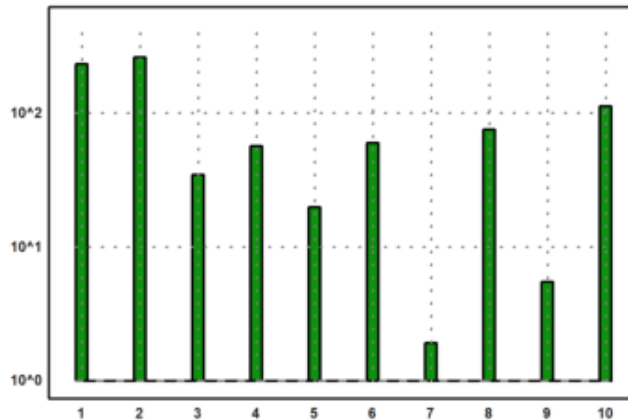
```

Mari kita uji dengan nilai yang terdistribusi secara eksponensial.

```

>aspect(1.5); x=1:10; y=-log(random(size(x)))*200; ...
>logimpulseplot1(x,y):

```



Mari kita menganimasikan kurva 2D menggunakan plot langsung. Perintah `plot(x,y)` hanya memplot kurva ke dalam jendela plot. `setplot(a,b,c,d)` menyetel jendela ini.

Fungsi `wait(0)` memaksa plot muncul di jendela grafis. Jika tidak, pengundian ulang akan dilakukan dalam interval waktu yang jarang.

```

>function animliss (n,m) ...

```

```

    t=linspace(0,2pi,500);
    f=0;
    c=framecolor(0);
    l=linewidth(2);
    setplot(-1,1,-1,1);
    repeat
        clg;
        plot(sin(n*t),cos(m*t+f));
        wait(0);
        if testkey() then break; endif;
        f=f+0.02;
    end;
    framecolor(c);
    linewidth(l);
endfunction

```

Tekan tombol apa saja untuk menghentikan animasi ini.

```

>animliss(2,3); // lihat hasilnya, jika sudah puas, tekan ENTER

```

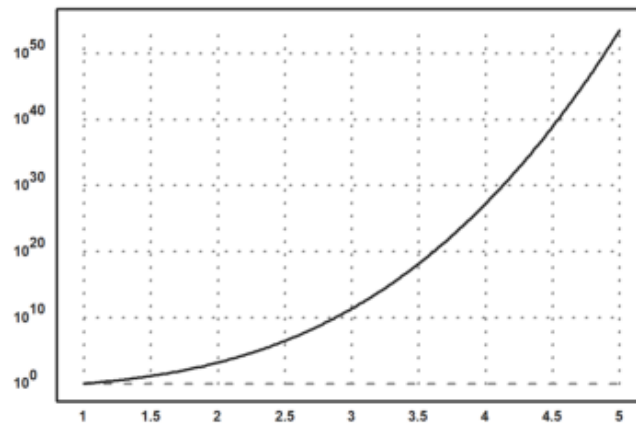
Plot Logaritmik

EMT menggunakan parameter "logplot" untuk skala logaritmik.

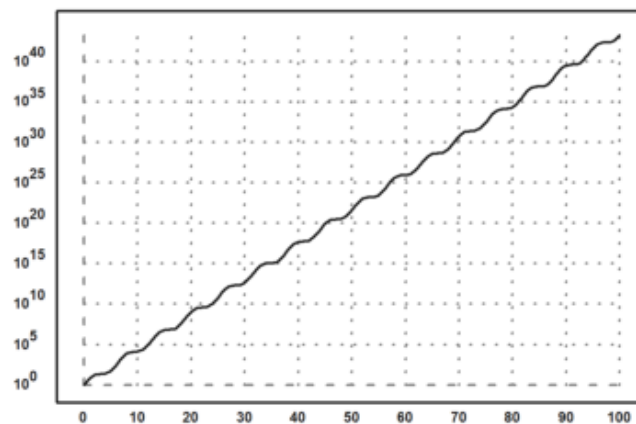
Plot logaritma dapat diplot menggunakan skala logaritma di y dengan logplot=1, atau menggunakan skala logaritma di x dan y dengan logplot=2, atau di x dengan logplot=3.

- logplot=1: y-logaritma
- logplot=2: x-y-logaritma
- logplot=3: x-logaritma

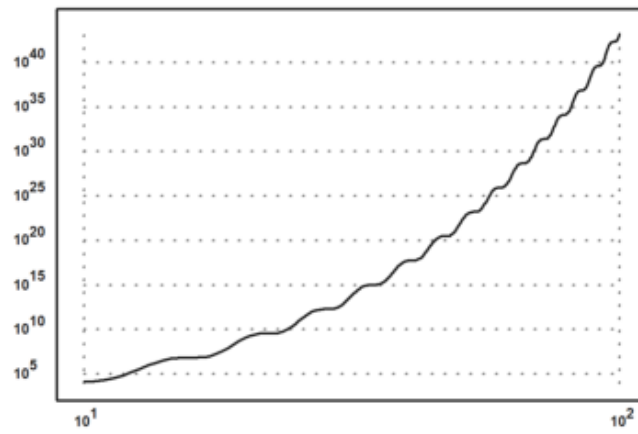
```
>plot2d("exp(x^3-x)*x^2",1,5,logplot=1):
```



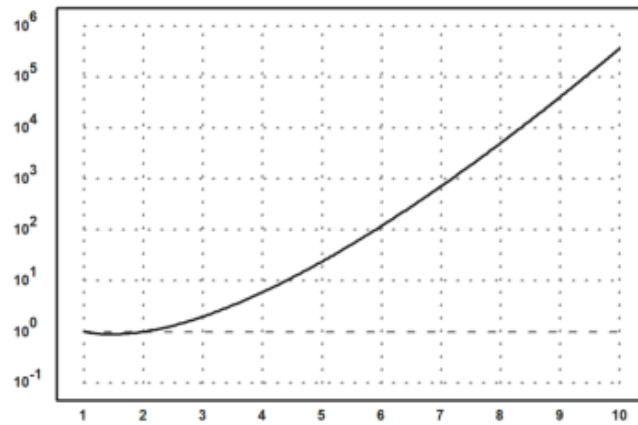
```
>plot2d("exp(x+sin(x))",0,100,logplot=1):
```



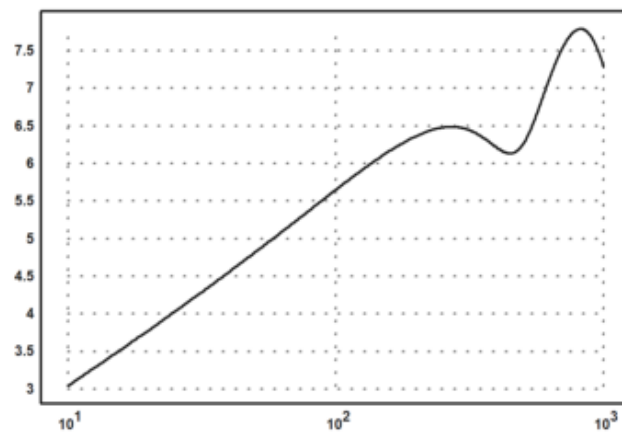
```
>plot2d("exp(x+sin(x))",10,100,logplot=2):
```



```
>plot2d("gamma(x) ",1,10,logplot=1):
```

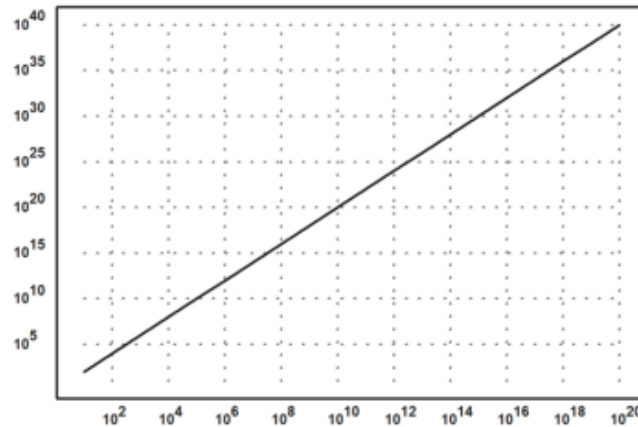


```
>plot2d("log(x*(2+sin(x/100))) ",10,1000,logplot=3):
```



Ini juga berfungsi dengan plot data.

```
>x=10^(1:20); y=x^2-x;  
>plot2d(x,y,logplot=2) :
```



Soal-soal

1. Suatu perusahaan yang mempunyai 3 pabrik akan memproduksi 2 jenis produk. Pabrik 1 dapat menghasilkan satu unit produk I selama 1 jam dan produk II selama 2 jam. Pabrik 2 dapat menghasilkan satu unit produk I selama 1 jam dan menghasilkan produk II selama 1 jam. Pabrik 3 dapat menghasilkan satu unit produk I selama 3 jam dan menghasilkan satu unit produk II selama 2 jam. Kapasitas produksi pabrik 1 setiap pekannya adalah dapat beroperasi paling lama 10 jam, pabrik 2 paling lama 6 jam, dan pabrik 3 paling lama 16 jam. Adapun keuntungan produk I per unit adalah 3 dan produk II adalah 5. Perusahaan tersebut ingin memaksimalkan laba yang diperoleh dengan keterbatasan kapasitas produksi setiap pabriknya. Masalahnya adalah berapa unit masing-masing produk I dan produk II yang harus diproduksi?

penyelesaian:

Dijadikan model matematika

-fungsi tujuan:

$$f(x, y) = 3x + 5y$$

-Kendala

$$x + 2y$$

$$x + y$$

$$3x + 2y$$

Pabrik 1 dapat beroperasi ≤ 10 jam

Pabrik 2 dapat beroperasi ≤ 6 jam

Pabrik 3 dapat beroperasi ≤ 16 jam

dan diperoleh kendala:

$$x + 2y \leq 10$$

$$x + y \leq 6$$

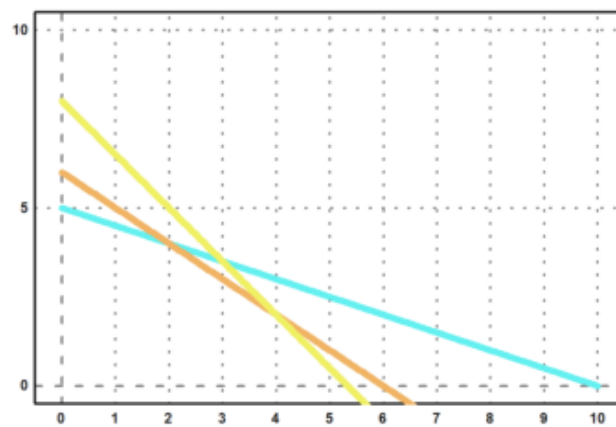
$$3x + 2y \leq 16$$

$$x \geq 0$$

$$y \geq 0$$

Kemudian akan digambar grafik

```
>plot2d(["(10-x)/2", "(6-x)", "(3*x-16)/-2"]...  
>,0,10,0,10,color=13:15,thickness=3,grid=5,):
```



Mencari perpotongan bisa dengan menggunakan solve
Kemudian kendala dibuat kedalam bentuk matriks koefisien

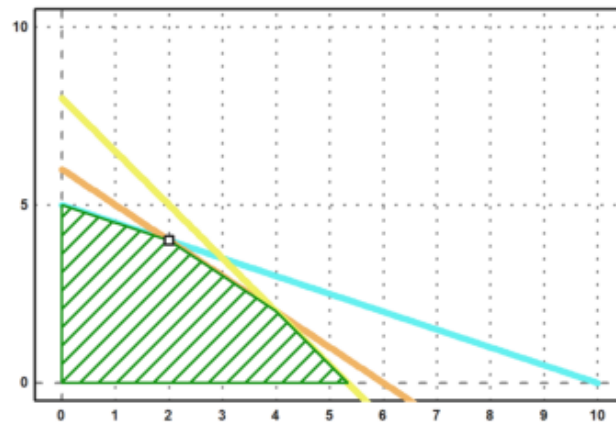
```
>A=[1,2;1,1;3,2]; b=[10;6;16];
```

Metode simpleks untuk maksimisasi atau minimisasi fungsi linier
fungsi dengan kendala linier, dan variabel yang dibatasi atau tidak dibatasi variabel. Fungsi ini memanggil
algoritme Simpleks bawaan.

```
>x=simplex(A,b,[3,5],>max); fraction x
```

2
4

```
>xa=feasibleArea(A,b);  
>plot2d(xa[1],xa[2],>filled,style="/",a=0,b=10,c=0,d=10,>add);  
>plot2d(x[1],x[2],>add,>points):
```



Kemudian tinggal di substitusikan saja titik (2,4) ke dalam fungsi tujuan maka ditemukan nilai optimal

$$3x + 5y$$

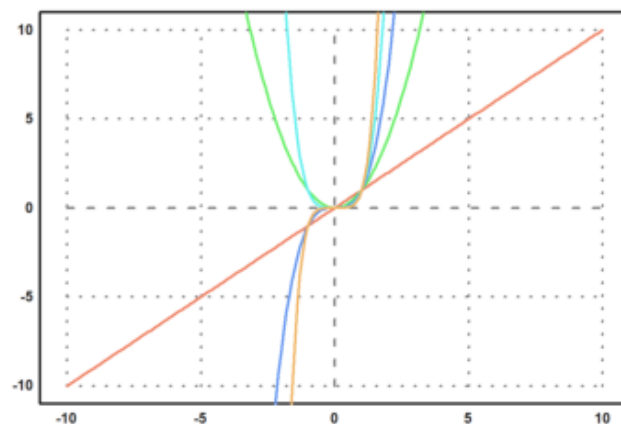
$$3 * 2 + 5 * 4$$

$$26$$

Jadi maksimal pabrik memperoleh laba adalah 26 dan produk 1 dibuat sebanyak 2 dan produk 2 dibuat sebanyak 4

2. Gambar kurva x^i dengan i adalah bilangan asli 1 sampai 5
penyelesaian:

```
>plot2d(["x", "x^2", "x^3", "x^4", "x^5"], a=-10, b=10, c=-10, d=10, color=10:14) :
```



Rujukan Lengkap Fungsi plot2d()

```
function plot2d (xv, yv, btest, a, b, c, d, xmin, xmax, r, n, ..
logplot, grid, frame, framecolor, square, color, thickness, style, ..
auto, add, user, delta, points, addpoints, pointstyle, bar, histogram, ..
distribution, even, steps, own, adaptive, hue, level, contour, ..
nc, filled, fillcolor, outline, title, xl, yl, maps, contourcolor, ..
contourwidth, ticks, margin, clipping, cx, cy, insimg, spectral, ..
cgrid, vertical, smaller, dl, niveau, levels)
```

Multipurpose plot function for plots in the plane (2D plots). This function can do plots of functions of one variables, data plots, curves in the plane, bar plots, grids of complex numbers, and implicit plots of functions of two variables.

Parameters

x,y : equations, functions or data vectors

a,b,c,d : Plot area (default $a=-2,b=2$)

r : if r is set, then $a=cx-r$, $b=cx+r$, $c=cy-r$, $d=cy+r$

r can be a vector $[rx,ry]$ or a vector $[rx1,rx2,ry1,ry2]$.

$xmin,xmax$: range of the parameter for curves

$auto$: Determine y-range automatically (default)

$square$: if true, try to keep square x-y-ranges

n : number of intervals (default is adaptive)

$grid$: 0 = no grid and labels,

1 = axis only,
2 = normal grid (see below for the number of grid lines)
3 = inside axis
4 = no grid
5 = full grid including margin
6 = ticks at the frame
7 = axis only
8 = axis only, sub-ticks

$frame$: 0 = no frame

$framecolor$: color of the frame and the grid

$margin$: number between 0 and 0.4 for the margin around the plot

$color$: Color of curves. If this is a vector of colors,

it will be used for each row of a matrix of plots. In the case of point plots, it should be a column vector. If a row vector or a full matrix of colors is used for point plots, it will be used for each data point.

$thickness$: line thickness for curves

This value can be smaller than 1 for very thin lines.

$style$: Plot style for lines, markers, and fills.

For points use
"[]", "<>", ".", "..", "...",
"*", "+", "|", "-", "o"
"[]#", "<>#", "o#" (filled shapes)
"[]w", "<>w", "ow" (non-transparent)
For lines use
"-", "--", "-.", ".", "-.-", "-.-", "->"
For filled polygons or bar plots use
"#", "#O", "O", "/", "\", "\/",
"+", "|", "-", "t"

points : plot single points instead of line segments
 addpoints : if true, plots line segments and points
 add : add the plot to the existing plot
 user : enable user interaction for functions
 delta : step size for user interaction
 bar : bar plot (x are the interval bounds, y the interval values)
 histogram : plots the frequencies of x in n subintervals
 distribution=n : plots the distribution of x with n subintervals
 even : use inter values for automatic histograms.
 steps : plots the function as a step function (steps=1,2)
 adaptive : use adaptive plots (n is the minimal number of steps)
 level : plot level lines of an implicit function of two variables
 outline : draws boundary of level ranges.
 If the level value is a 2xn matrix, ranges of levels will be drawn
 in the color using the given fill style. If outline is true, it
 will be drawn in the contour color. Using this feature, regions of
 f(x,y) between limits can be marked.
 hue : add hue color to the level plot to indicate the function

value

contour : Use level plot with automatic levels
 nc : number of automatic level lines
 title : plot title (default "")
 xl, yl : labels for the x- and y-axis
 smaller : if >0, there will be more space to the left for labels.
 vertical :

Turns vertical labels on or off. This changes the global variable
 verticallabels locally for one plot. The value 1 sets only vertical
 text, the value 2 uses vertical numerical labels on the y axis.

filled : fill the plot of a curve
 fillcolor : fill color for bar and filled curves
 outline : boundary for filled polygons
 logplot : set logarithmic plots

1 = logplot in y,
 2 = logplot in xy,
 3 = logplot in x

own :

A string, which points to an own plot routine. With >user, you get
 the same user interaction as in plot2d. The range will be set
 before each call to your function.

maps : map expressions (0 is faster), functions are always mapped.
 contourcolor : color of contour lines
 contourwidth : width of contour lines
 clipping : toggles the clipping (default is true)
 title :

This can be used to describe the plot. The title will appear above the plot. Moreover, a label for the x and y axis can be added with `xl="string"` or `yl="string"`. Other labels can be added with the functions `label()` or `labelbox()`. The title can be a unicode string or an image of a Latex formula.

cgrid :

Determines the number of grid lines for plots of complex grids. Should be a divisor of the the matrix size minus 1 (number of subintervals). `cgrid` can be a vector `[cx,cy]`.

Overview

The function can plot

- expressions, call collections or functions of one variable,
- parametric curves,
- x data against y data,
- implicit functions,
- bar plots,
- complex grids,
- polygons.

If a function or expression for `xv` is given, `plot2d()` will compute values in the given range using the function or expression. The expression must be an expression in the variable `x`. The range must be defined in the parameters `a` and `b` unless the default range should be used. The y-range will be computed automatically, unless `c` and `d` are specified, or a radius `r`, which yields the range `r,r`

for `x` and `y`. For plots of functions, `plot2d` will use an adaptive evaluation of the function by default. To speed up the plot for complicated functions, switch this off with `<adaptive`, and optionally decrease the number of intervals `n`. Moreover, `plot2d()` will by default use mapping. I.e., it will compute the plot element for element. If your expression or your functions can handle a vector `x`, you can switch that off with `<maps` for faster evaluation.

Note that adaptive plots are always computed element for element.

If functions or expressions for both `xv` and for `yv` are specified, `plot2d()` will compute a curve with the `xv` values as x-coordinates and the `yv` values as y-coordinates. In this case, a range should be defined for the parameter using `xmin`, `xmax`. Expressions contained in strings must always be expressions in the parameter variable `x`.