Menggambar Grafik 2D dengan EMT

Nama : Sabrina Royani Winahyu
NIM : 24030130013

Notebook ini menjelaskan tentang cara menggambar berbagaikurva dan grafik 2D dengan software EMT. EMT menyediakan fungsi plot2d() untuk menggambar berbagai kurva dan grafik dua dimensi (2D).

Plot Dasar

Terdapat beberapa fungsi dasar untuk plotting. Ada koordinat layar (screen coordinates), yang selalu berada dalam rentang 0 hingga 1024 di setiap sumbu, terlepas dari apakah layar berbentuk persegi atau tidak. Selain itu, ada juga koordinat plot (plot coordinates), yang dapat diatur menggunakan setplot(). Pemetaan antara kedua jenis koordinat ini bergantung pada jendela plot yang sedang digunakan. Sebagai contoh, fungsi default shrinkwindow() memberikan ruang untuk label sumbu dan judul grafik.

Pada contoh ini, kita hanya menggambar beberapa garis acak dengan berbagai warna. Untuk detail lebih lanjut tentang fungsi-fungsi ini, silakan pelajari fungsi inti dari 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;
```

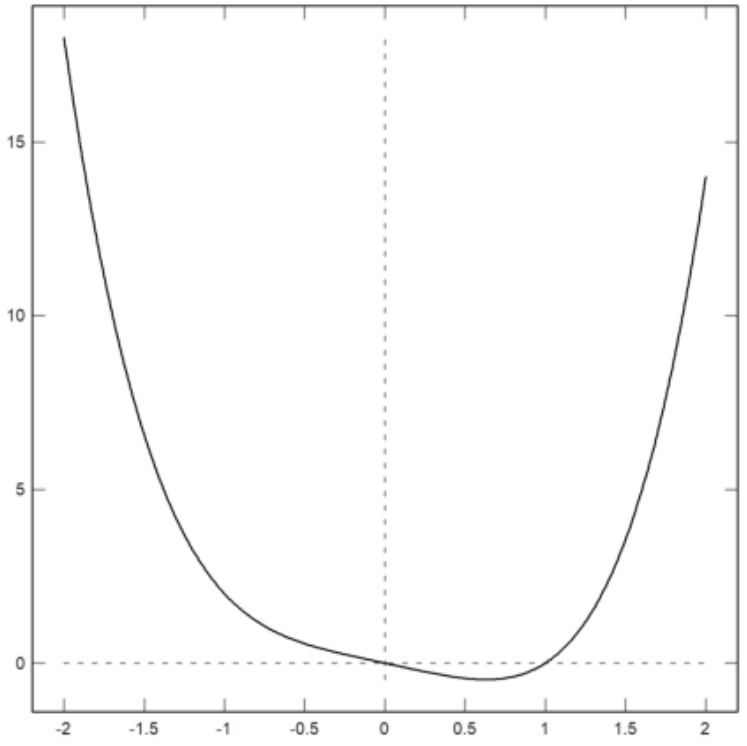
Kita perlu menahan tampilan grafik (hold the graphics), karena perintah ‘plot()‘ akan menghapus (clear) jendela plot.

Untuk menghapus semua yang telah kita buat, kita bisa menggunakan perintah ‘reset()‘.

Untuk menampilkan gambar hasil plot di layar notebook, perintah ‘plot2d()‘ dapat diakhiri dengan titik dua (‘:‘). Cara lainnya adalah mengakhiri perintah ‘plot2d()‘ dengan titik koma (‘;‘), lalu menggunakan perintah ‘insimg()‘ untuk menampilkan gambar hasil plot.

Sebagai contoh lainnya, kita menggambar sebuah plot sebagai sisipan (inset) di dalam plot lainnya. Ini dilakukan dengan mendefinisikan jendela plot yang lebih kecil. Perlu diperhatikan bahwa jendela ini tidak menyediakan ruang untuk label sumbu di luar area plot. Kita harus menambahkan margin seperlunya untuk itu. Perhatikan juga bahwa kita menyimpan dan mengembalikan jendela penuh, serta menahan plot saat ini ketika kita membuat plot sisipan.

```
>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);
```

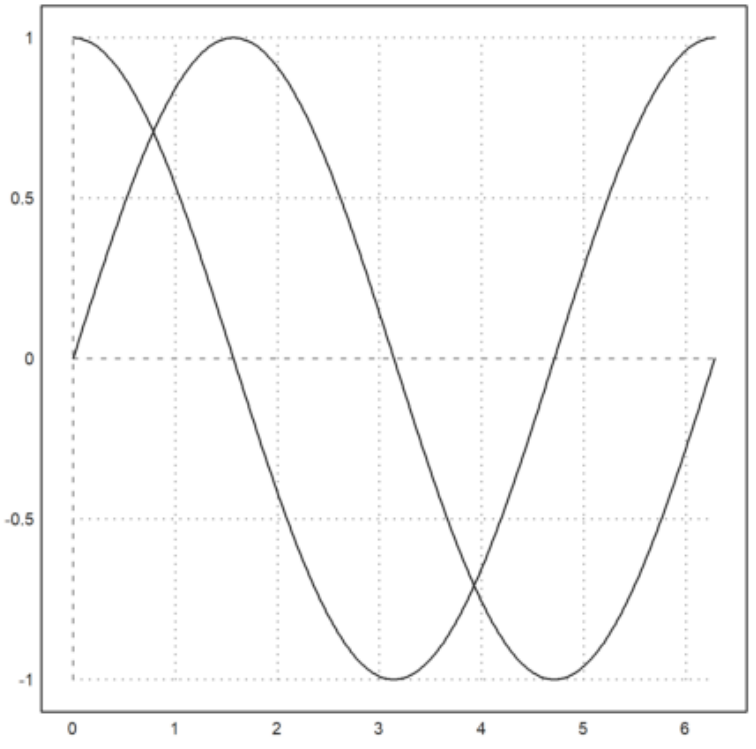
Sebuah plot dengan beberapa gambar dapat dibuat dengan cara yang sama. Untuk ini tersedia fungsi utilitas figure().

Aspek Plot

Secara bawaan, plot menggunakan jendela berbentuk persegi. Anda dapat mengubahnya dengan fungsi aspect(). Jangan lupa untuk mengatur ulang aspek setelahnya. Anda juga bisa mengubah pengaturan bawaan tersebut di menu dengan memilih "Set Aspect" ke rasio aspek tertentu atau ke ukuran saat ini dari jendela grafik.

Namun, Anda juga dapat mengubahnya hanya untuk satu plot saja. Untuk hal ini, ukuran area plot saat ini akan diubah, dan jendela akan diatur agar label memiliki ruang yang cukup.

```
>aspect(2); // rasio panjang dan lebar 2:1
>plot2d(["sin(x)","cos(x)"],0,2pi):
```




```
>aspect();
>reset;
```

Fungsi reset() mengembalikan pengaturan plot ke bawaan, termasuk rasio aspek.

Plot 2D di Euler

EMT Math Toolbox memiliki plot 2D, baik untuk data maupun fungsi. EMT menggunakan fungsi plot2d. Fungsi ini dapat digunakan untuk memplot fungsi maupun data.

Dimungkinkan juga untuk membuat plot di Maxima menggunakan Gnuplot atau di Python menggunakan Matplotlib.

Euler dapat membuat plot 2D dari:

- ekspresi,
- fungsi, variabel, atau kurva parametrik,
- vektor nilai x-y,
- kumpulan titik dalam bidang,
- kurva implisit dengan level atau daerah level,
- fungsi kompleks.

Gaya plot mencakup berbagai gaya untuk garis dan titik, plot batang, serta plot berarsir.

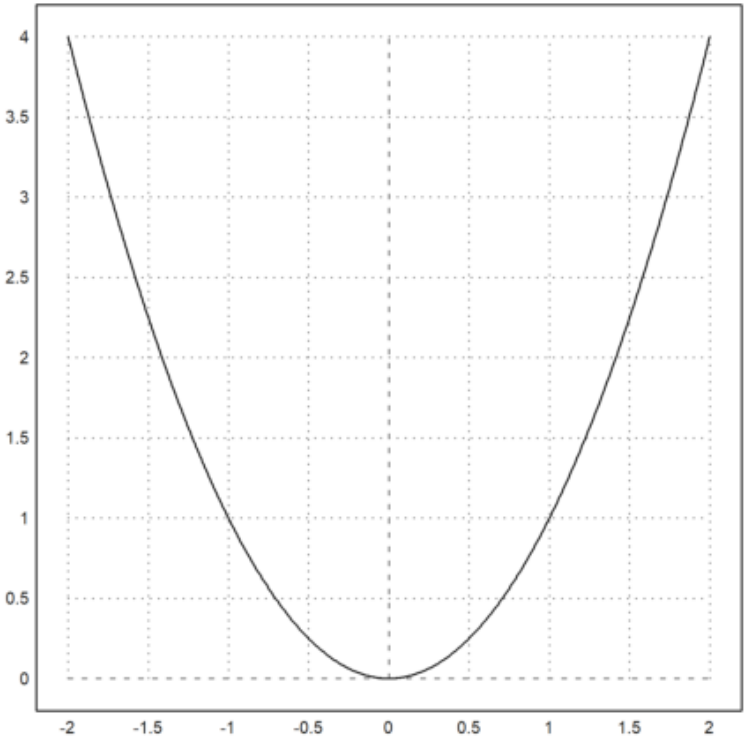
Plot Ekspresi atau Variabel

Sebuah ekspresi tunggal dalam ”x” (misalnya ‘”4*x^2”‘) atau nama fungsi (misalnya ‘”f”‘) akan menghasilkan grafik dari fungsi tersebut.

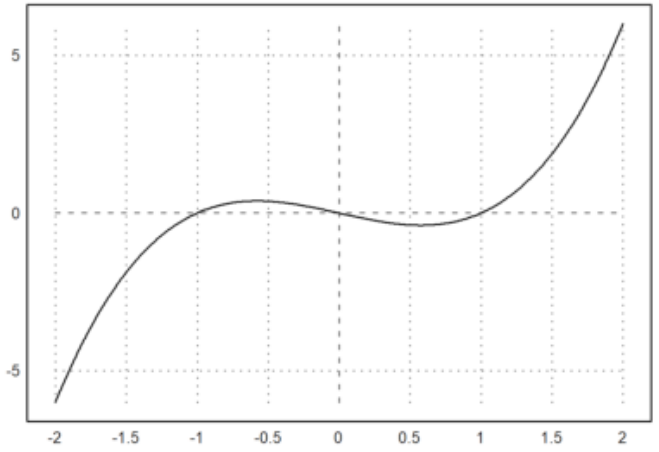
Berikut contoh paling dasar, yang menggunakan rentang bawaan dan menetapkan rentang-y yang sesuai agar grafik fungsi dapat ditampilkan dengan baik.

Catatan: Jika Anda mengakhiri baris perintah dengan titik dua ‘”:”‘, maka plot akan dimasukkan ke dalam jendela teks. Jika tidak, tekan TAB untuk melihat plot jika jendela plot tertutup.

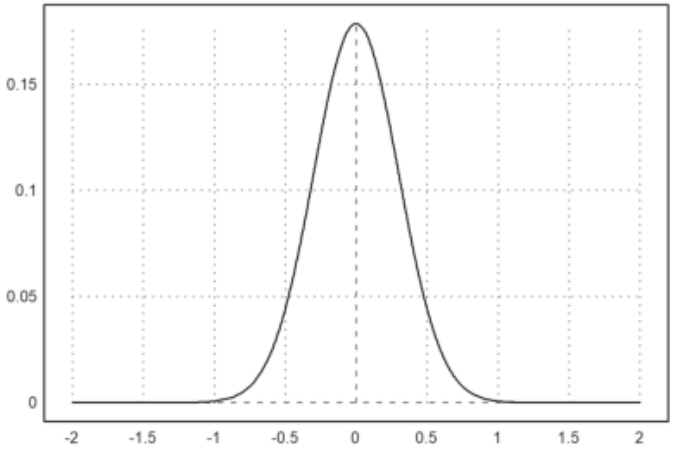
```
>plot2d("x^2"):
```




```
>aspect(1.5); plot2d("x^3-x"):
```




```
>a:=5.6; plot2d("exp(-a*x^2)/a"); insimg(30); // menampilkan gambar hasil plot setinggi 25 baris
```

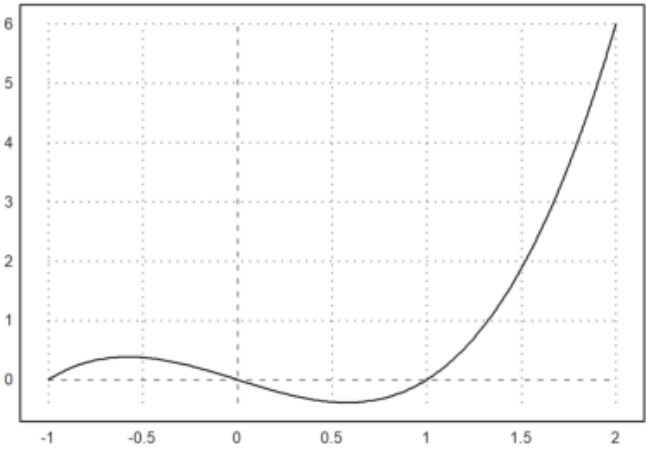


Dari beberapa contoh sebelumnya Anda dapat melihat bahwa aslinya gambar plot menggunakan sumbu X dengan rentang nilai dari -2 sampai dengan 2. Untuk mengubah rentang nilai X dan Y, Anda dapat menambahkan nilai-nilai batas X (dan Y) di belakang ekspresi yang digambar.

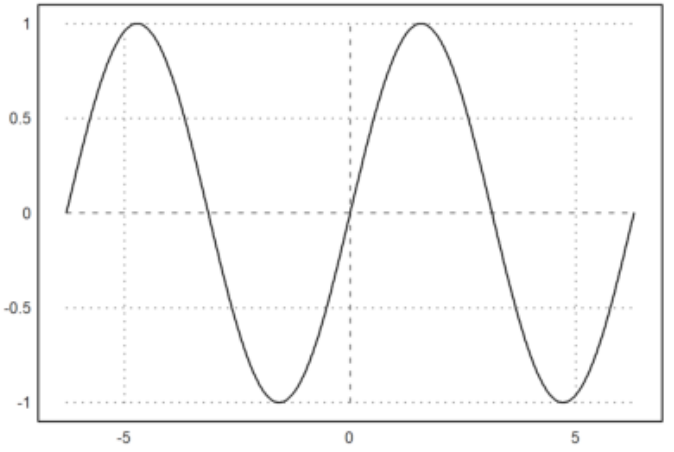
The plot range is set with the following assigned parameters

- a,b: x-range (default -2,2)
- c,d: y-range (default: scale with values)
- r: alternatively a radius around the plot center
- cx,cy: the coordinates of the plot center (default 0,0)

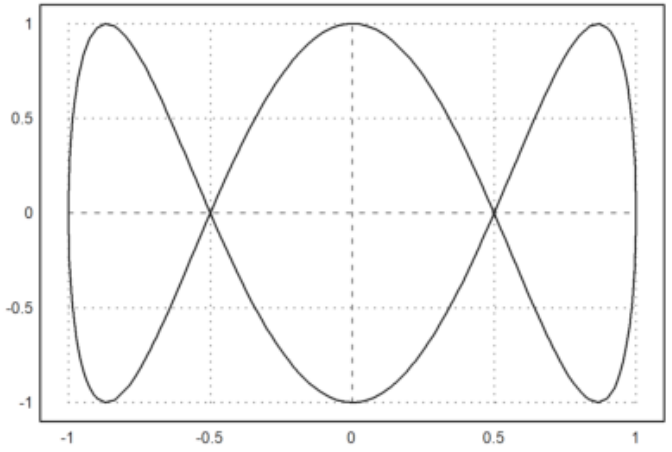
```
>plot2d("x^3-x",-1,2):
```




```
>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 lain dari penggunaan titik dua adalah perintah insimg(lines), yang menyisipkan plot dengan menempati sejumlah baris teks tertentu.

Dalam opsi, plot dapat diatur agar muncul:

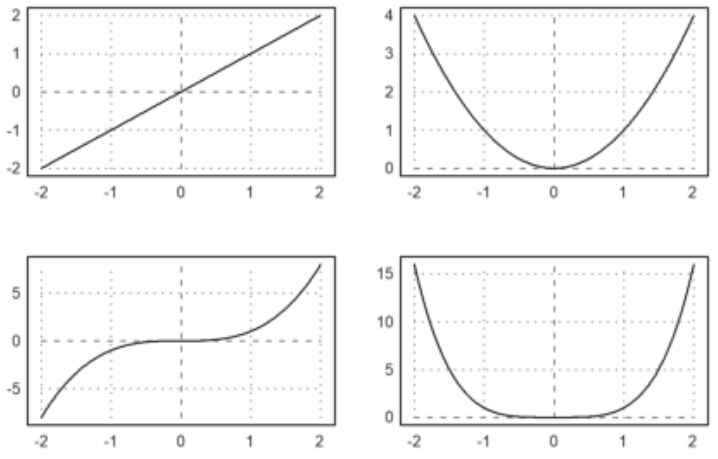
- di jendela terpisah yang dapat diubah ukurannya,
- di jendela notebook.

Lebih banyak gaya dapat dicapai dengan perintah plot tertentu.

Dalam semua kasus, tekan tombol Tab untuk melihat plot jika tersembunyi.

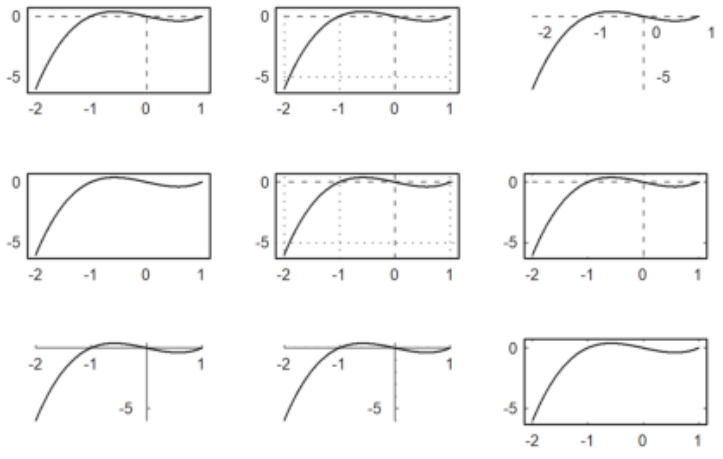
Untuk membagi jendela menjadi beberapa plot, gunakan perintah figure(). Pada contoh, kita memplot $xˆ1$ hingga $xˆ4$ ke dalam 4 bagian jendela. figure(0) mengembalikan jendela ke pengaturan bawaan.

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



Dalam plot2d(), tersedia gaya alternatif dengan grid=x. Untuk memberikan gambaran, berbagai gaya grid ditampilkan dalam satu gambar (lihat penjelasan perintah figure() di bawah).
Gaya grid=0 tidak disertakan. Gaya ini tidak menampilkan grid maupun bingkai.

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

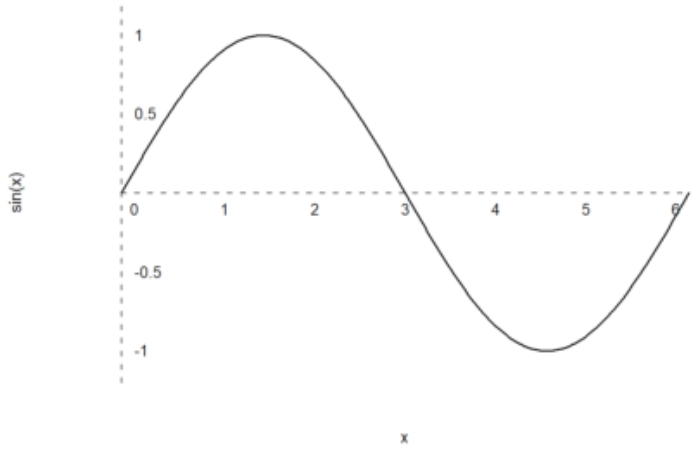


Jika argumen pada plot2d() berupa sebuah ekspresi yang diikuti oleh empat angka, maka angka-angka tersebut adalah rentang-x dan rentang-y untuk plot.

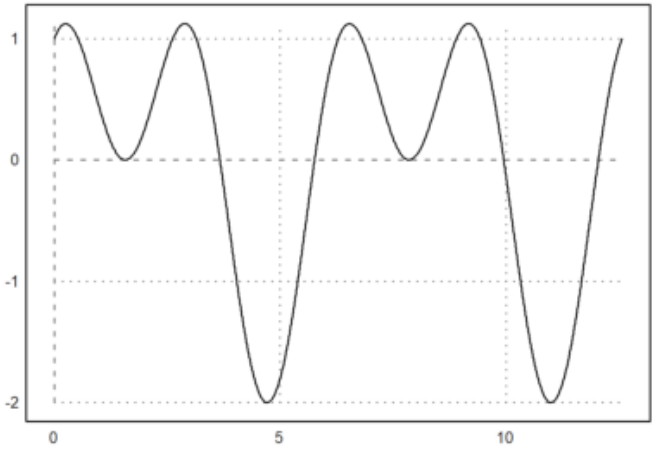
Sebagai alternatif, nilai a, b, c, d dapat ditentukan sebagai parameter yang diberi nilai, misalnya a=... dan seterusnya.

Pada contoh berikut, kita mengubah gaya grid, 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)"):
```




```
>plot2d("sin(x)+cos(2*x)",0,4pi):
```

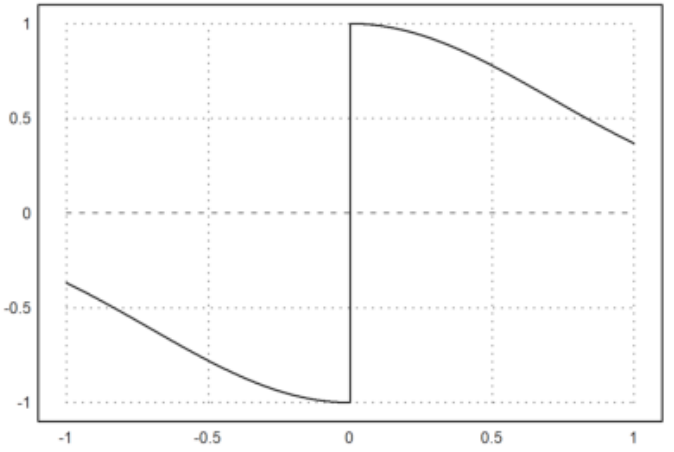


Gambar yang dihasilkan dengan menyisipkan plot ke dalam jendela teks akan disimpan di direktori yang sama dengan notebook, secara bawaan di dalam subdirektori bernama "images". Gambar-gambar tersebut juga digunakan dalam ekspor HTML.

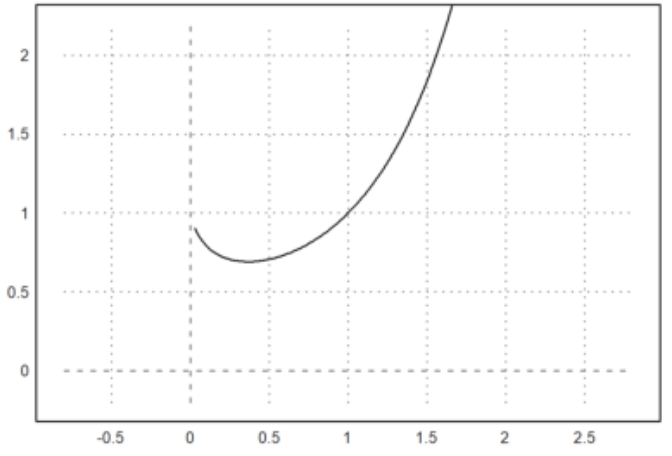
Anda dapat dengan mudah menandai gambar apa pun dan menyalinnya ke clipboard dengan Ctrl+C. Tentu saja, Anda juga bisa mengekspor grafik saat ini menggunakan fungsi yang ada di menu File.

Fungsi atau ekspresi dalam plot2d dievaluasi secara adaptif. Untuk mempercepat, Anda dapat mematikan plot adaptif dengan \<adaptive dan menentukan jumlah subinterval dengan n=.... Hal ini biasanya hanya diperlukan pada kasus-kasus tertentu saja.

```
>plot2d("sign(x)*exp(-x^2)",-1,1,<adaptive,n=10000):
```

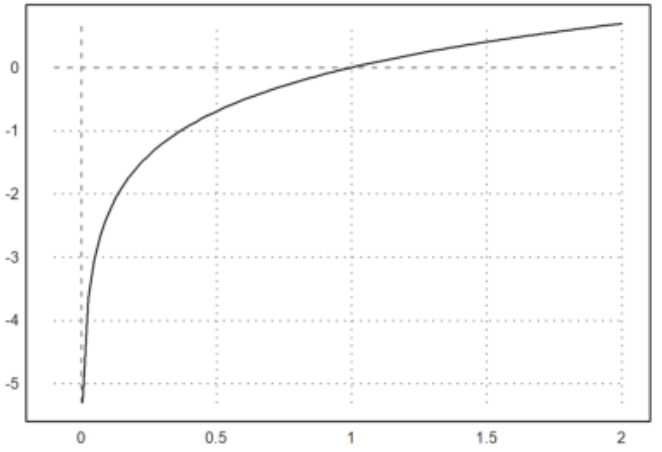



```
>plot2d("x^x",r=1.2,cx=1,cy=1):
```



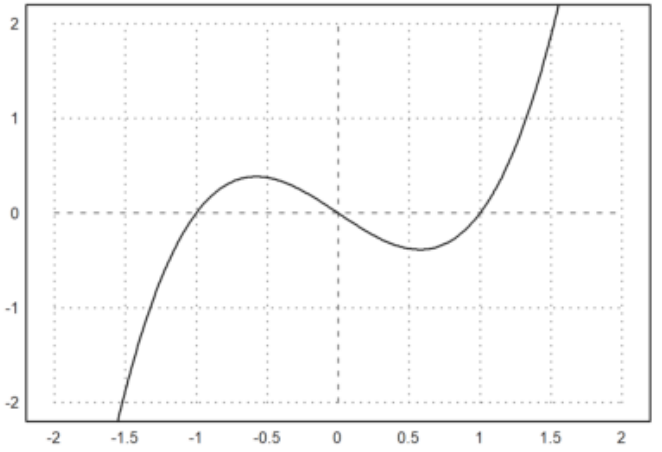
Perhatikan bahwa xˆx tidak terdefinisi untuk x=0. Fungsi plot2d akan mengenali kesalahan ini, dan mulai menggambar grafik hanya ketika fungsi tersebut sudah terdefinisi. Hal ini berlaku juga untuk semua fungsi yang menghasilkan nilai NAN jika berada di luar daerah definisinya.

```
>plot2d("log(x)",-0.1,2):
```

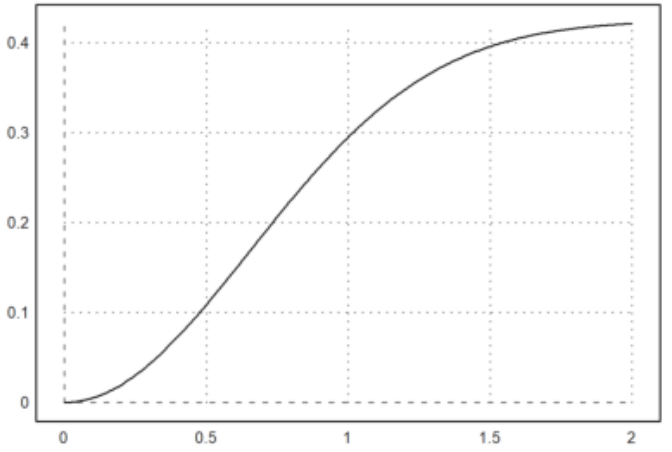


Parameter square=true (atau >square) akan secara otomatis memilih rentang sumbu y sehingga hasilnya berubah jendela grafik berbentuk persegi. Perhatikan bahwa secara default, Euler menggunakan ruang persegi di dalam jendela plot

```
>plot2d("x^3-x",>square):
```

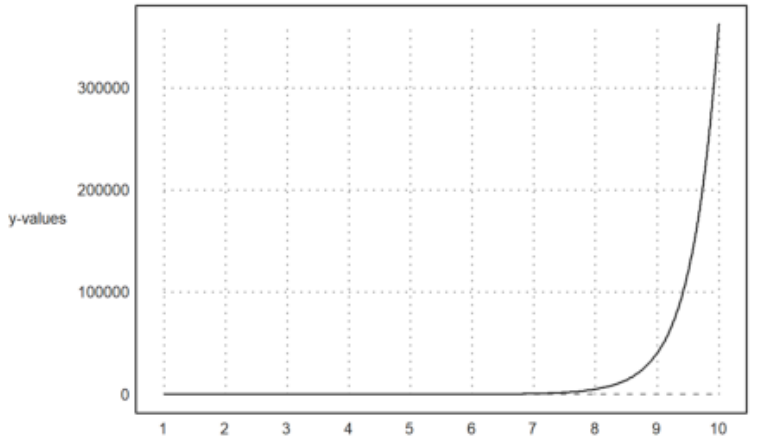



```
>plot2d(''integrate("sin(x)*exp(-x^2)",0,x)'',0,2): // plot integral
```



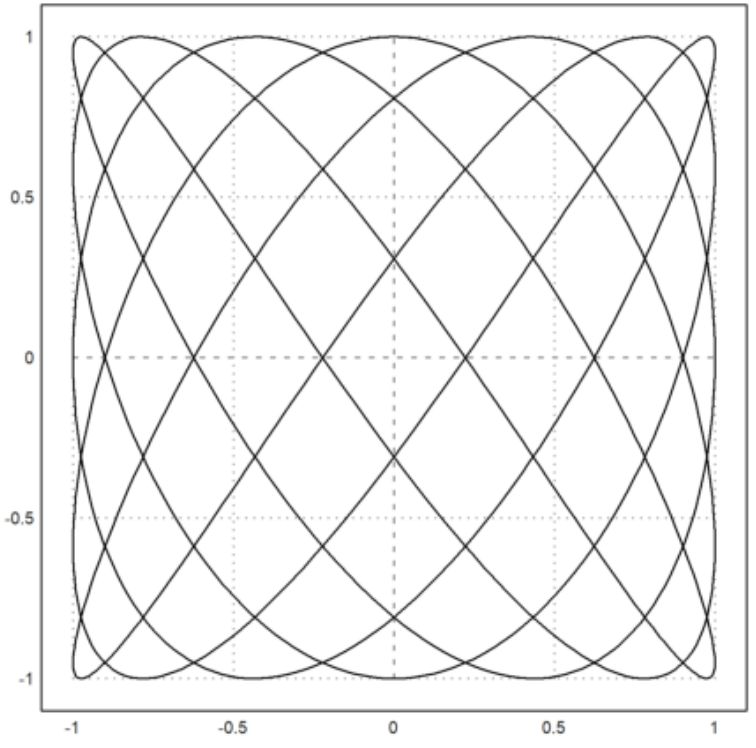
Jika membutuhkan lebih banyak ruang untuk label sumbu y, gunakan perintah shrinkwindow() dengan parameter smaller, atau berikan nilai positif pada opsi smaller di plot2d().

```
>plot2d("gamma(x)",1,10,yl="y-values",smaller=6,<vertical):
```

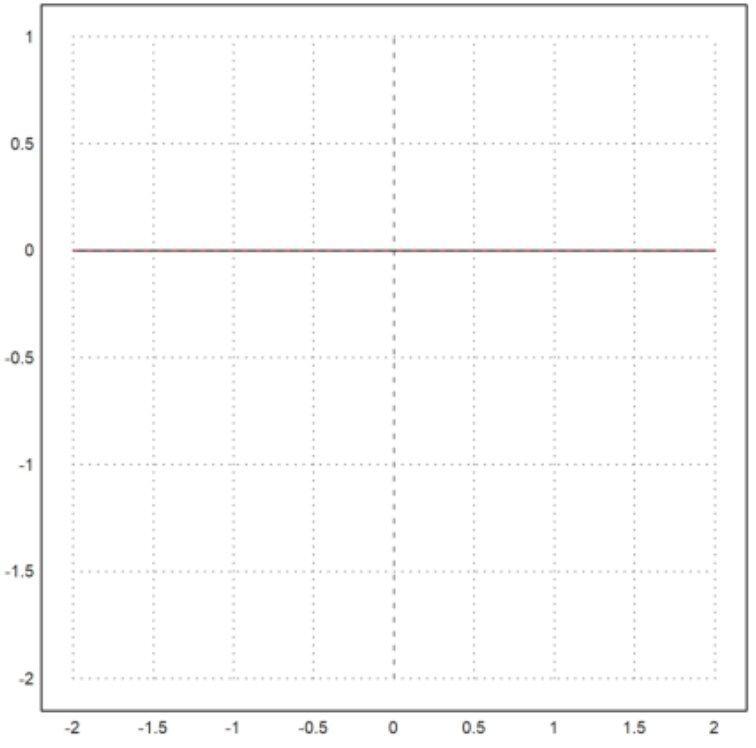


Eskpresi simbolik juga bisa digunakan, karena disimpan sebagai ekspresi string sederhana.

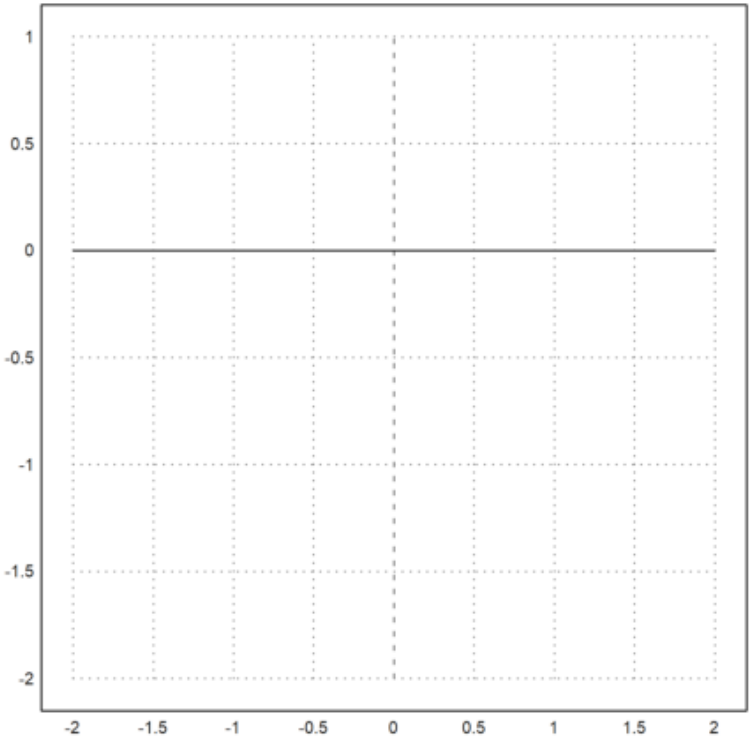
```
>x=linspace(0,2pi,1000); plot2d(sin(5x),cos(7x)):
```



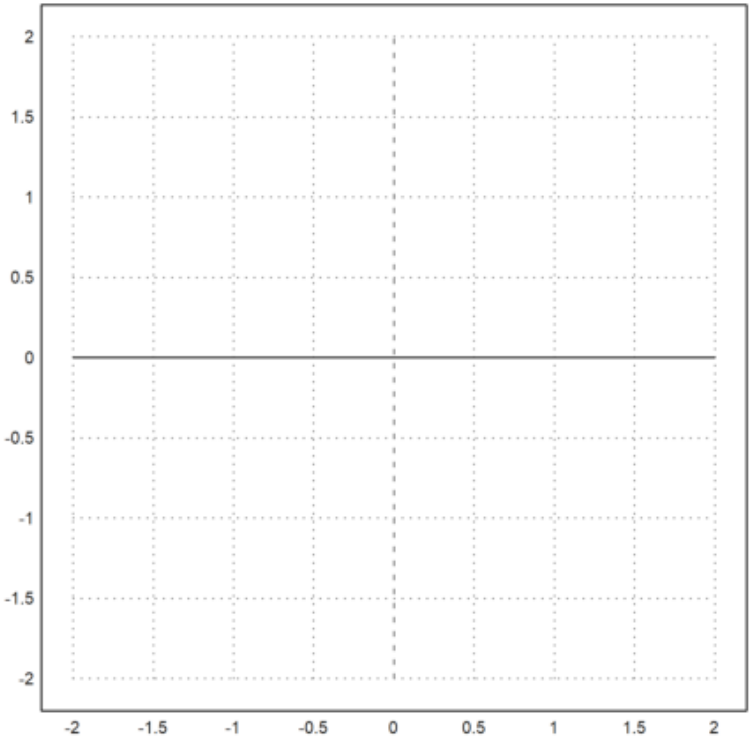

```
>a:=5.6; expr &= exp(-a*x^2)/a; // define expression
>plot2d(expr,-2,2): // plot from -2 to 2
>plot2d(expr,r=1,thickness=2): // plot in a square around (0,0)
>plot2d(&diff(expr,x),>add,style="--",color=red): // add another plot
```



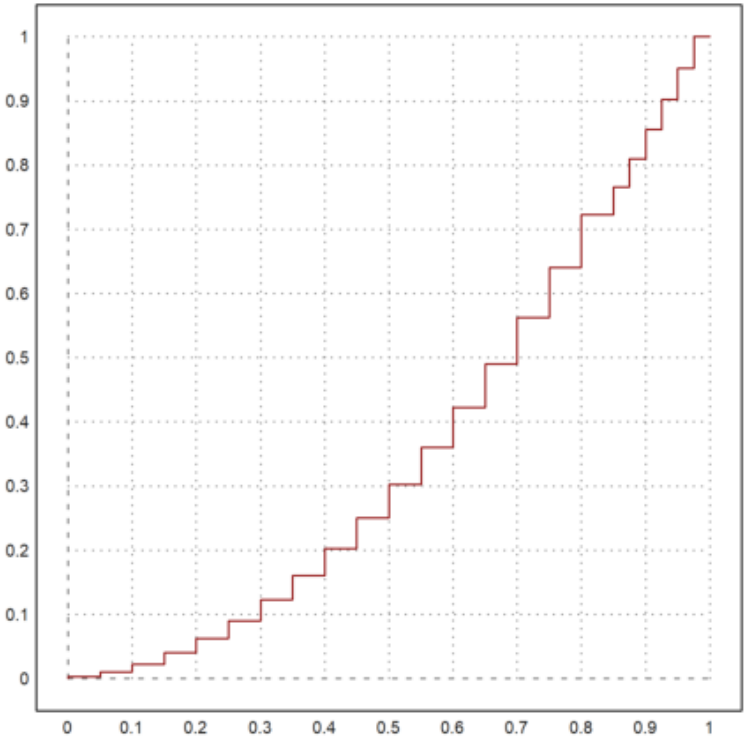

```
>plot2d(&diff(expr,x,2),a=-2,b=2,c=-2,d=1): // plot in rectangle
```



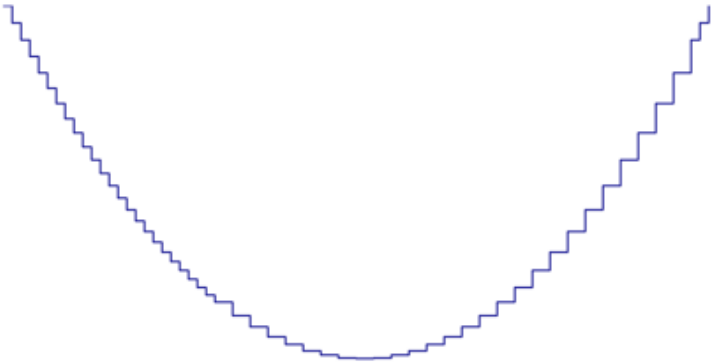

```
>plot2d(&diff(expr,x),a=-2,b=2,>square): // keep plot square
```




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




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

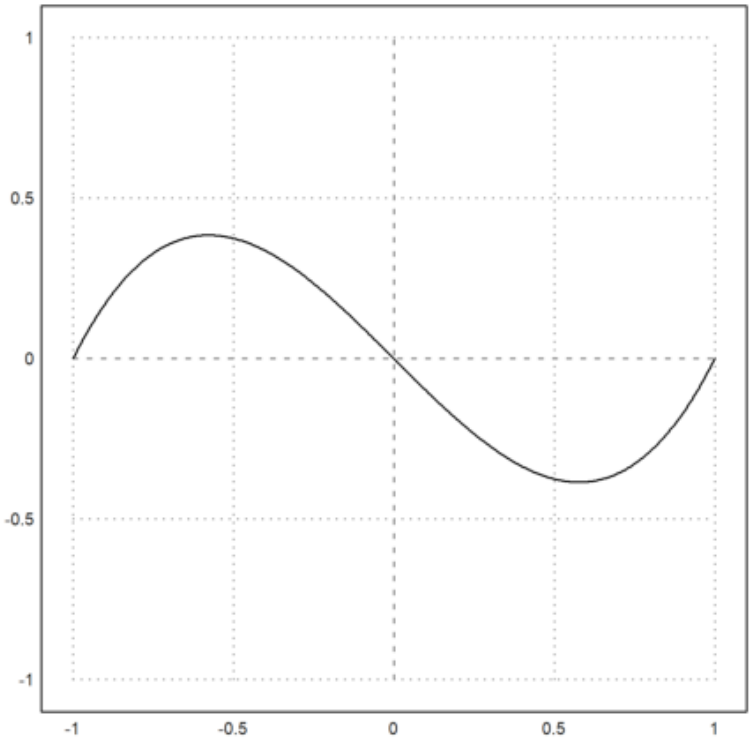


Fungsi dengan Satu Parameter

Fungsi plot yang paling penting untuk grafik bidang datar adalah plot2d(). Fungsi ini diimplementasikan dalam bahasa Euler pada file "plot.e", yang dimuat saat awal program dijalankan.

Berikut beberapa contoh penggunaan fungsi. Seperti biasa dalam EMT, fungsi yang bekerja untuk fungsi lain atau ekspresi dapat diberikan parameter tambahan (selain x), yang bukan merupakan variabel global, ke dalam fungsi dengan parameter titik koma atau dengan call collection.

```
>function f(x,a) := x^2/a+a*x^2-x; // define a function
>a=0.3; plot2d("f",0,1;a): // plot with a=0.3
>plot2d("f",0,1;0.4): // plot with a=0.4
>plot2d({{"f",0.2}},0,1): // plot with a=0.2
>plot2d({{"f(x,b)",b=0.1}},0,1): // plot with 0.1
>function f(x) := x^3-x; ...
>plot2d("f",r=1):
```



Berikut adalah ringkasan fungsi yang diterima:

-ekspresi simbolik dalam x
-fungsi simbolik dengan nama seperti ”f”
-fungsi simbolik hanya dengan menyebutkan nama f

Fungsi plot2d() juga menerima fungsi simbolik. Untuk fungsi simbolik, cukup menggunakan namanya saja.

```
>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 saja sudah cukup untuk menggambarkannya.

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

$$\sin(x) \ E^{-x}$$

```
>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="-.-"):
```

```
Syntax error in expression, or unfinished expression!
Error in expression: 'diff(f(x),x,1)
 %ploteval:
    y0=f$(x[1],args());
adaptiveevalone:
    s=%ploteval(g$,t;args());
Try "trace errors" to inspect local variables after errors.
plot2d:
    dw/n,dw/n^2,dw/n,auto;args());
```

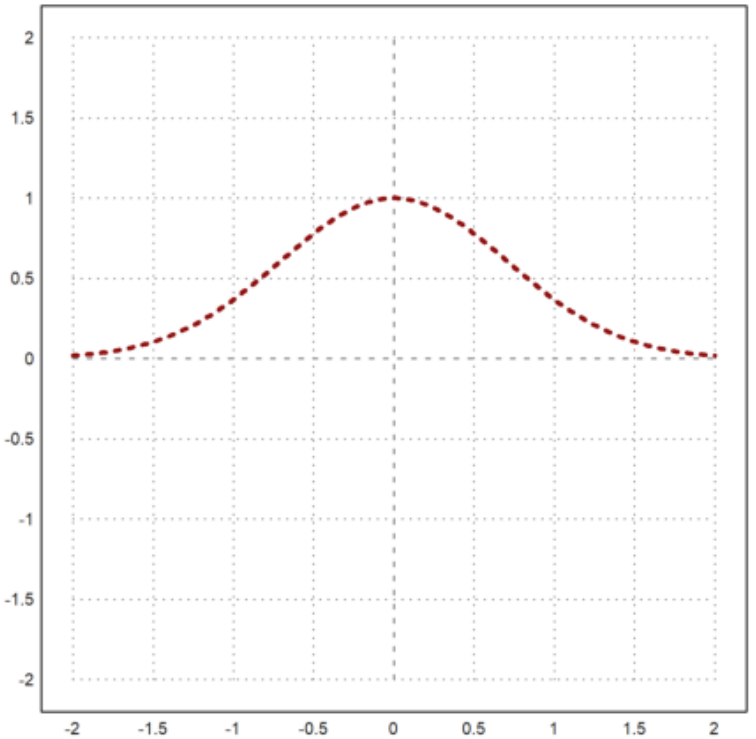
Untuk gaya garis terdapat berbagai pilihan.

-style="...". Pilihan: "-", "–", "-.", ".", ".-.", "-.-".
-color: lihat bagian di bawah untuk warna.
-thickness: default bernilai 1.

Warna dapat dipilih dari salah satu warna bawaan, atau menggunakan warna RGB.

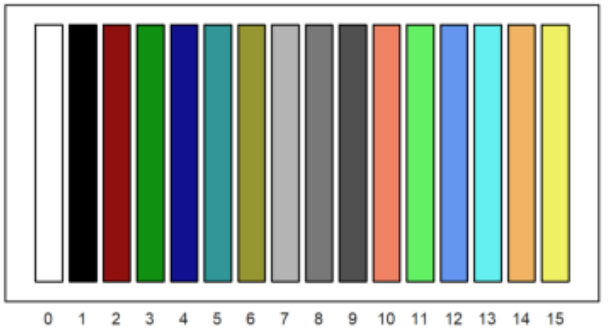
-0..15: indeks warna bawaan.
-konstanta warna: white, black, red, green, blue, cyan, olive, lightgray, gray, darkgray, orange, lightgreen, turquoise, lightblue, lightorange, yellow
-rgb(red,green,blue): parameter berupa bilangan riil dalam [0,1].

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



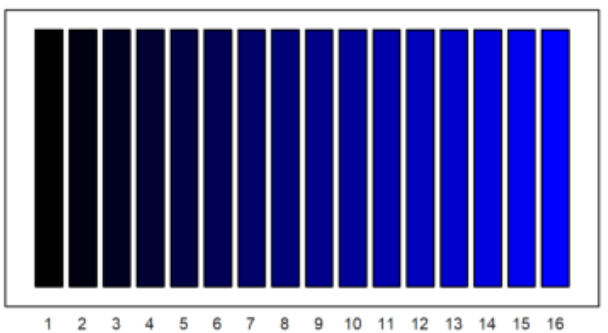
Berikut ini adalah tampilan warna-warna bawaan EMT.

```
>aspect(2); columnsplot(ones(1,16),lab=0:15,grid=0,color=0:15):
```



Tetapi Anda dapat menggunakan warna apa pun.

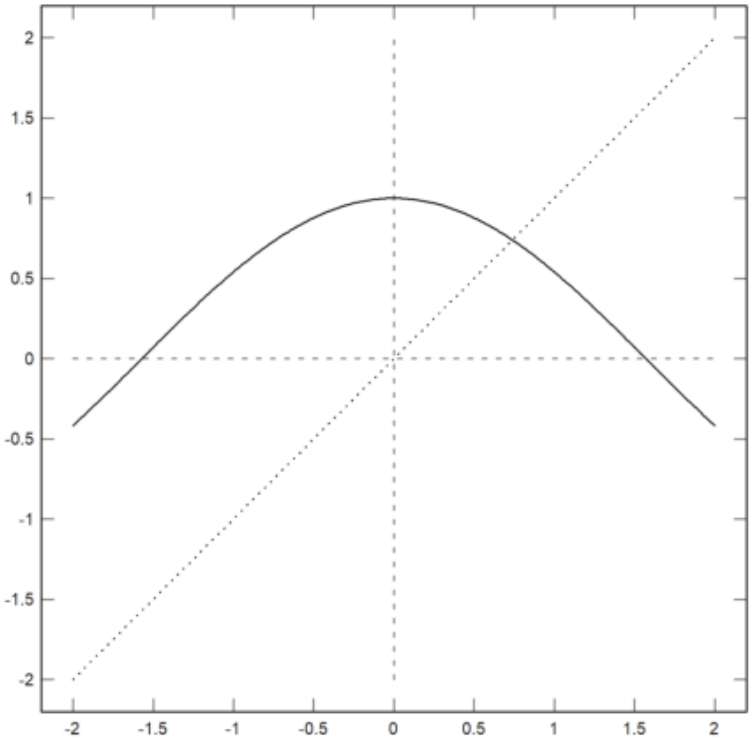
```
>columnsplot(ones(1,16),grid=0,color=rgb(0,0,linspace(0,1,15))):
```



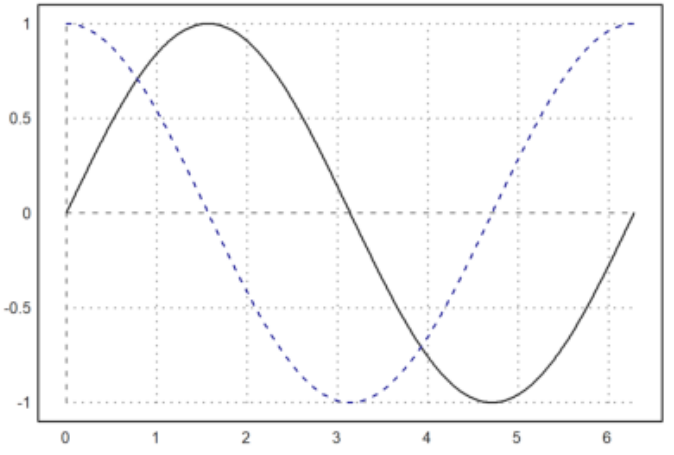
Menggambar Beberapa Kurva pada Bidang Koordinat yang Sama

Untuk memplot lebih dari satu fungsi (banyak fungsi) dalam satu jendela dapat dilakukan dengan beberapa cara. Salah satu metode adalah dengan menggunakan >add pada beberapa pemanggilan plot2d, kecuali pada pemanggilan pertama. Fitur ini sudah kita gunakan pada contoh-contoh sebelumnya.

```
>aspect(); plot2d("cos(x)",r=2,grid=6); plot2d("x",style=".",>add):
```

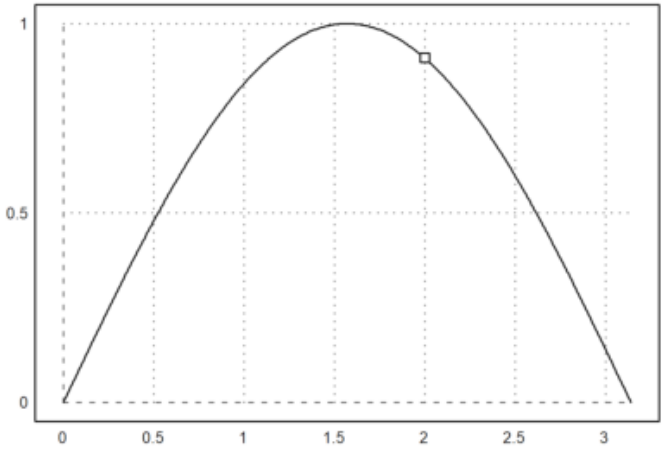



```
>aspect(1.5); plot2d("sin(x)",0,2pi); plot2d("cos(x)",color=blue,style="--",>add):
```



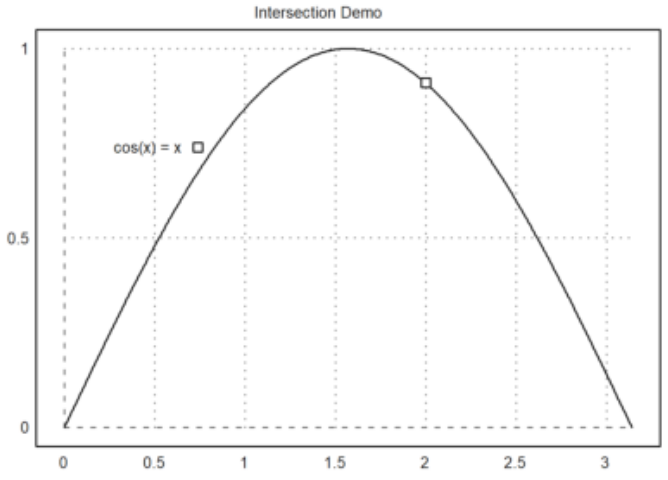
Salah satu kegunaan >add adalah untuk menambahkan titik pada kurva.

```
>plot2d("sin(x)",0,pi); plot2d(2,sin(2),>points,>add):
```



Kita menambahkan titik potong dengan sebuah label (pada posisi "cl" untuk center left), dan menyisipkan hasilnya ke dalam notebook. Kita juga menambahkan judul pada plot tersebut.

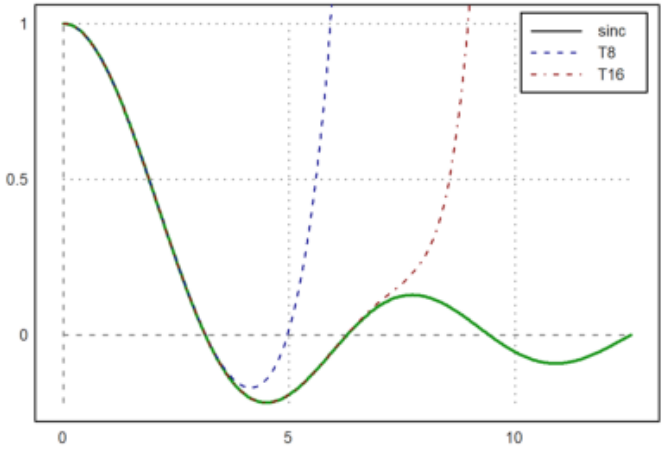
```
>plot2d(["cos(x)","x"],r=1.1,cx=0.5,cy=0.5, ...
>  color=[black,blue],style=["-","."], ...
>  grid=1);
>x0=solve("cos(x)-x",1);  ...
>  plot2d(x0,x0,>points,>add,title="Intersection Demo");  ...
>  label("cos(x) = x",x0,x0,pos="cl",offset=20):
```



Dalam demo berikut, kita memplot fungsi sinc(x) = sin(x)/x serta deret Taylor ke-8 dan ke-16 dari fungsi tersebut. Perhitungan ekspansi ini dilakukan menggunakan Maxima melalui ekspresi simbolik.
Plot ini dibuat dengan perintah multi-baris menggunakan tiga pemanggilan plot2d(). Pemanggilan kedua dan ketiga menggunakan flag >add, sehingga plot-plot tersebut memakai rentang yang sama dengan sebelumnya.

Kita juga menambahkan sebuah kotak label untuk menjelaskan fungsi-fungsi tersebut.

```
>$taylor(sin(x)/x,x,0,4)
>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]):
```



Pada contoh berikut, kita 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;
```

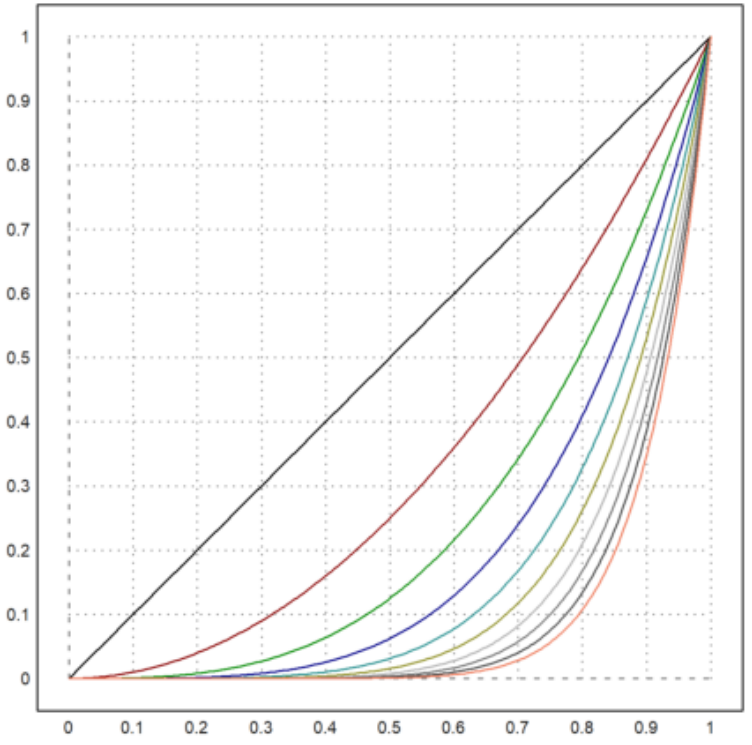
Metode kedua adalah dengan menggunakan sepasang matriks berupa nilai-nilai x dan nilai-nilai y dengan ukuran yang sama.

Kita menghasilkan sebuah matriks nilai dengan satu Polinomial Bernstein pada setiap baris. Untuk itu, kita cukup menggunakan sebuah vektor kolom dari i. Silakan lihat bagian pengantar 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):
```

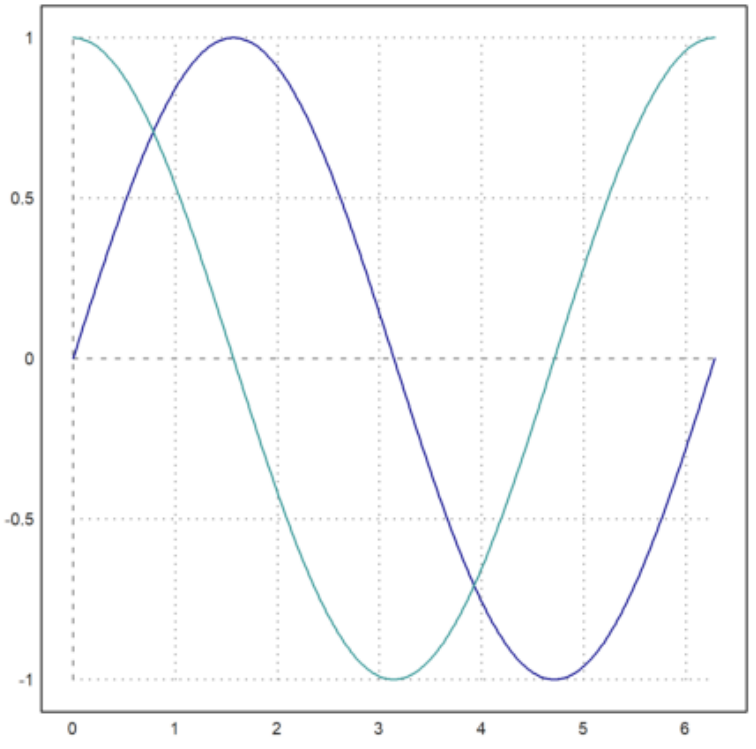
Perlu dicatat bahwa parameter color dapat berupa sebuah vektor. Dengan demikian, setiap warna akan digunakan untuk setiap baris dari matriks.

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



Metode lain adalah dengan menggunakan sebuah vektor berisi ekspresi (string). Dengan cara ini, Anda dapat menggunakan array warna, array gaya, dan array ketebalan dengan panjang yang sama.

```
>plot2d(["sin(x)","cos(x)"],0,2pi,color=4:5):
```




```
>plot2d(["sin(x)","cos(x)"],0,2pi): // plot vector of expressions
```

Kita dapat memperoleh vektor seperti itu dari Maxima dengan menggunakan makelist() dan mxm2str().

```
>v &= makelist(binomial(10,i)*x^i*(1-x)^(10-i),i,0,10) // make list
```

$$\left[(1-x)^{10},\ 10\,(1-x)^{9}\,x,\ 45\,(1-x)^{8}\,x^{2},\ 120\,(1-x)^{7}\,x^{3},\ 210\,(1-x)^{6}\,x^{4},\ 252\,(1-x)^{5}\,x^{5},\ 210\,(1-x)^{4}\,x^{6},\ 120\,(1-x)^{3}\,x^{7},\ 45\,(1-x)^{2}\,x^{8},\ 10\,(1-x)\,x^{9},\ x^{10}\right]$$

```
>mxm2str(v) // get a vector of strings from the symbolic vector
```

```
(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
```

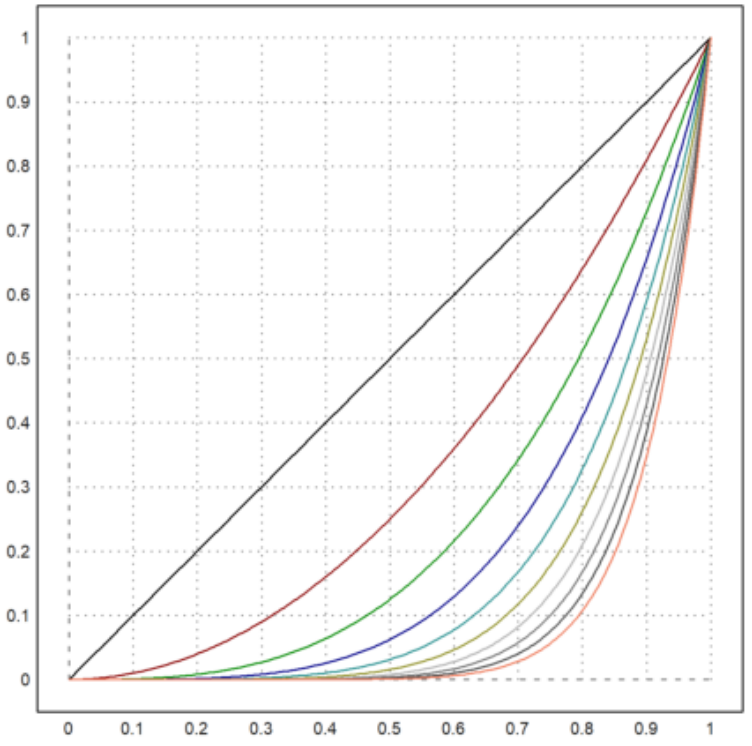
```
>plot2d(mxm2str(v),0,1): // plot functions
```

Alternatif lain adalah menggunakan bahasa matriks dari Euler.

Jika suatu ekspresi menghasilkan sebuah matriks fungsi, dengan satu fungsi pada setiap baris, maka semua fungsi tersebut akan dipetakan ke dalam satu plot.

Untuk itu, gunakan vektor parameter dalam bentuk vektor kolom. Jika ditambahkan array warna, maka array tersebut akan digunakan untuk setiap baris pada plot.

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

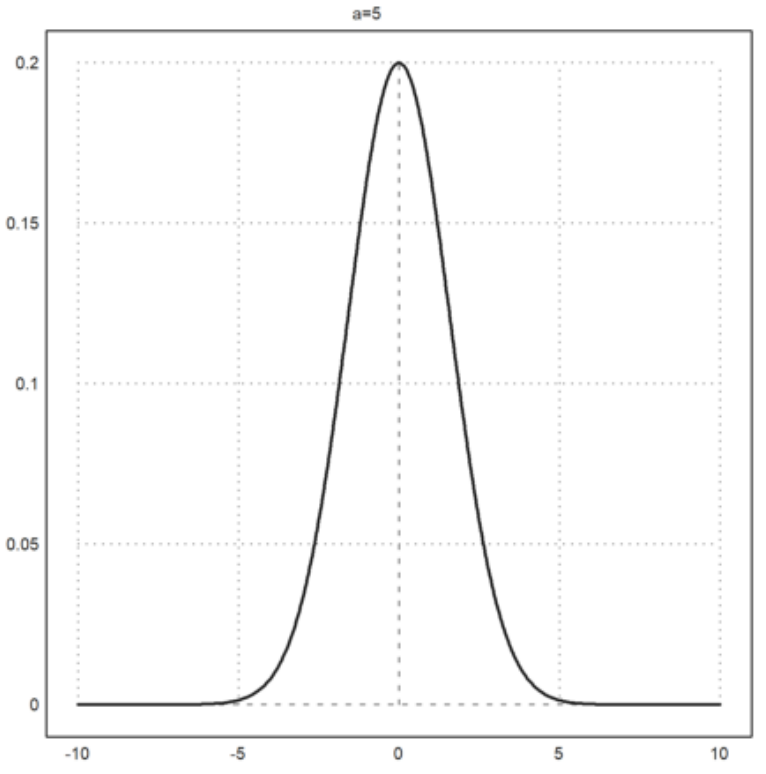


Ekspresi dan fungsi satu baris dapat melihat variabel global.

Jika Anda tidak dapat menggunakan variabel global, Anda perlu menggunakan fungsi dengan parameter tambahan, dan meneruskan parameter tersebut sebagai parameter titik koma.

Perhatikan untuk meletakkan semua parameter yang diberikan di bagian akhir perintah plot2d. Pada contoh, kita meneruskan a=5 ke fungsi f, yang kemudian kita plot dari -10 hingga 10.

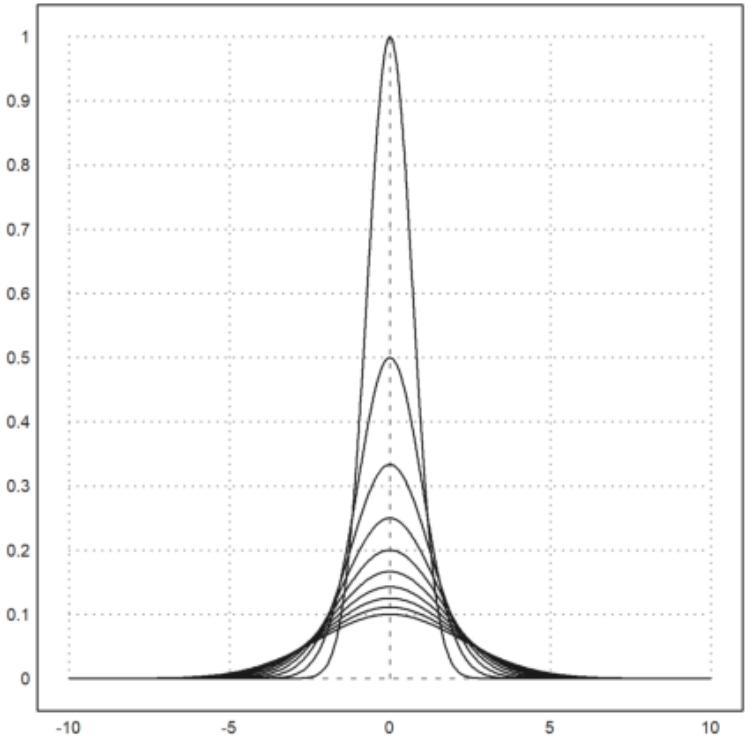
```
>function f(x,a) := 1/a*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5"):
```



Sebagai alternatif, gunakan sebuah koleksi berisi nama fungsi dan semua parameter tambahan. Daftar khusus ini disebut call collection, dan merupakan cara yang lebih disarankan untuk meneruskan argumen ke sebuah fungsi yang dirinya sendiri diteruskan sebagai argumen ke fungsi lain.

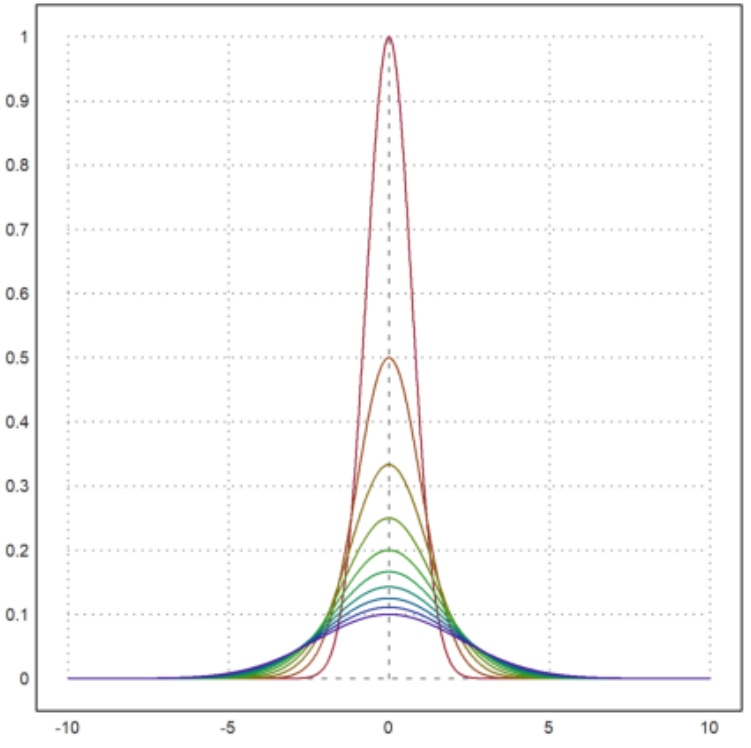
Pada contoh berikut, kita menggunakan sebuah loop untuk memplot beberapa fungsi (lihat tutorial tentang pemrograman untuk loops).

```
>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) mewakili satu fungsi. Selain itu, kita dapat menetapkan warna untuk setiap baris matriks. Klik ganda pada fungsi getspectral() untuk penjelasan.

```
>x=-10:0.01:10; a=(1:10)'; plot2d(x,f(x,a),color=getspectral(a/10)):
```



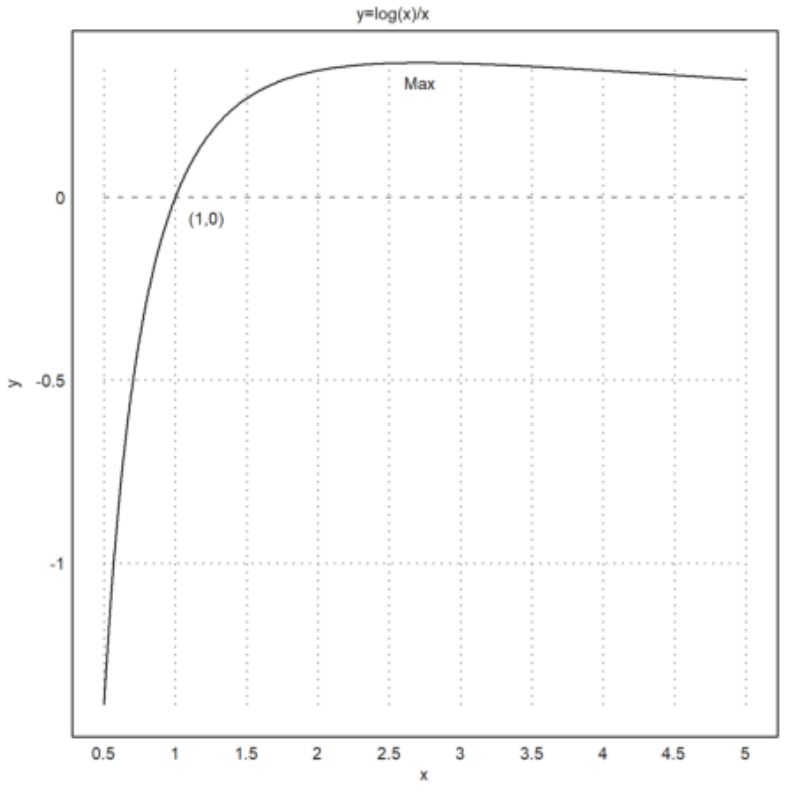
Label Teks

Dekorasi dalam grafik bisa berupa:

judul menggunakan title=”...”
label sumbu x dan y dengan xl=”...” serta yl=”...”
label teks lain memakai label(”...”,x,y)

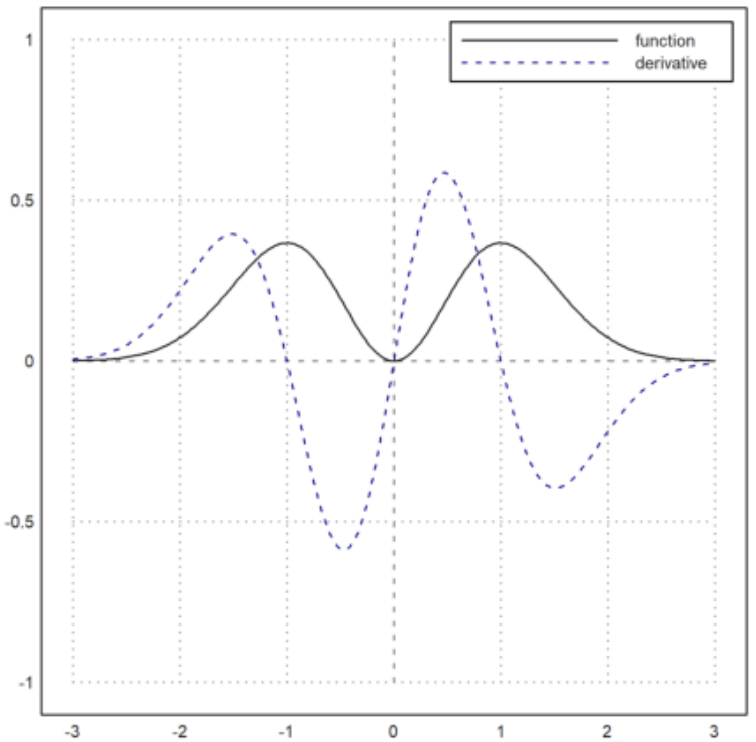
Perintah label akan menambahkan teks ke grafik pada titik koordinat (x,y). Selain itu, perintah ini juga dapat diberi tambahan argumen untuk menentukan posisinya.

```
>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"):
```



Terdapat pula fungsi labelbox(), yang berfungsi menampilkan teks beserta kurva. Fungsi ini membutuhkan masukan berupa vektor string dan warna, di mana masing-masing pasangan digunakan untuk setiap fungsi yang ditampilkan.

```
>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):
```

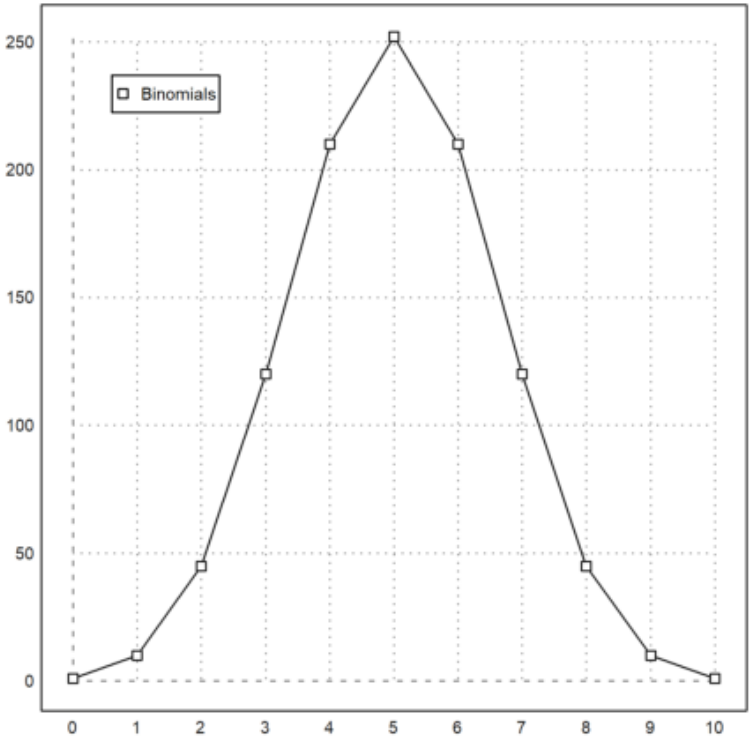


Secara bawaan, kotak label ditempatkan di bagian kanan atas, namun dengan opsi >left, posisinya akan berpindah ke kiri atas. Kotak ini bisa dipindahkan ke lokasi lain sesuai kebutuhan. Titik acuannya berada di sudut kanan atas kotak, sedangkan nilai koordinat ditentukan sebagai pecahan dari ukuran jendela grafik. Lebar kotak akan menyesuaikan secara otomatis.

Pada grafik berupa titik, kotak label tetap dapat digunakan. Untuk itu, tambahkan parameter >points atau gunakan vektor penanda, masing-masing sesuai dengan jumlah label yang ada.

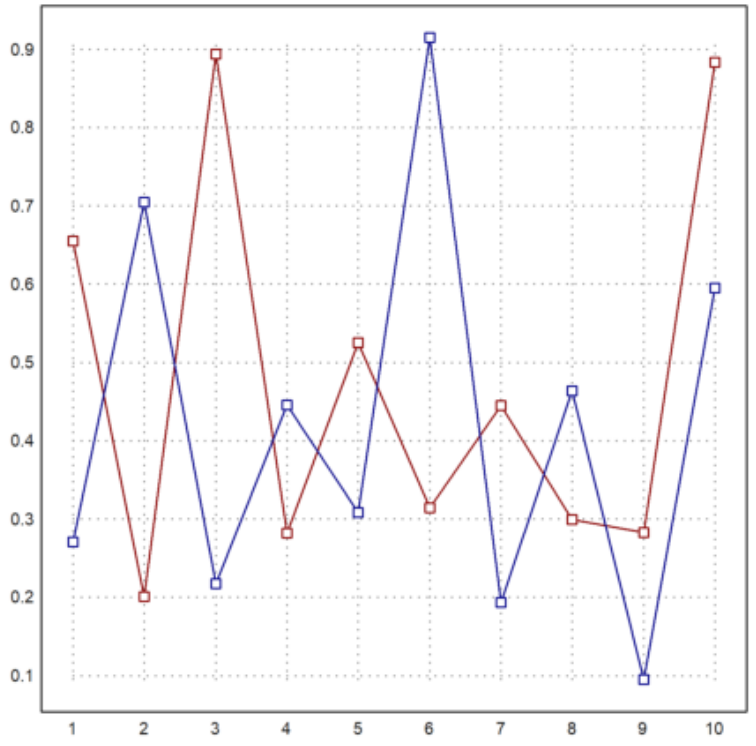
Pada contoh berikut hanya digunakan satu fungsi, sehingga cukup memakai string biasa, bukan vektor string. Warna teks diatur menjadi hitam untuk ilustrasi ini.

```
>n=10; plot2d(0:n,bin(n,0:n),>addpoints); ...
>labelbox("Binomials",styles="[]",>points,x=0.1,y=0.1, ...
>tcolor=black,>left):
```



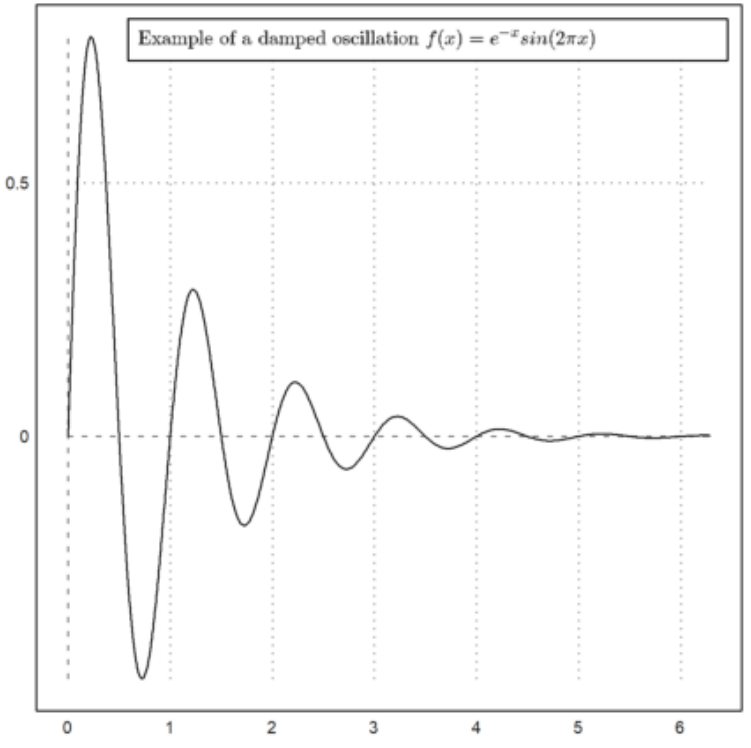
Selain itu, gaya plot yang dipakai juga tersedia dalam statplot(). Sama seperti pada plot2d(), pengguna bisa mengatur warna tiap garis plot. Ada pula jenis plot khusus yang ditujukan untuk analisis statistik (lihat panduan statistik untuk detailnya).

```
>statplot(1:10,random(2,10),color=[red,blue]):
```



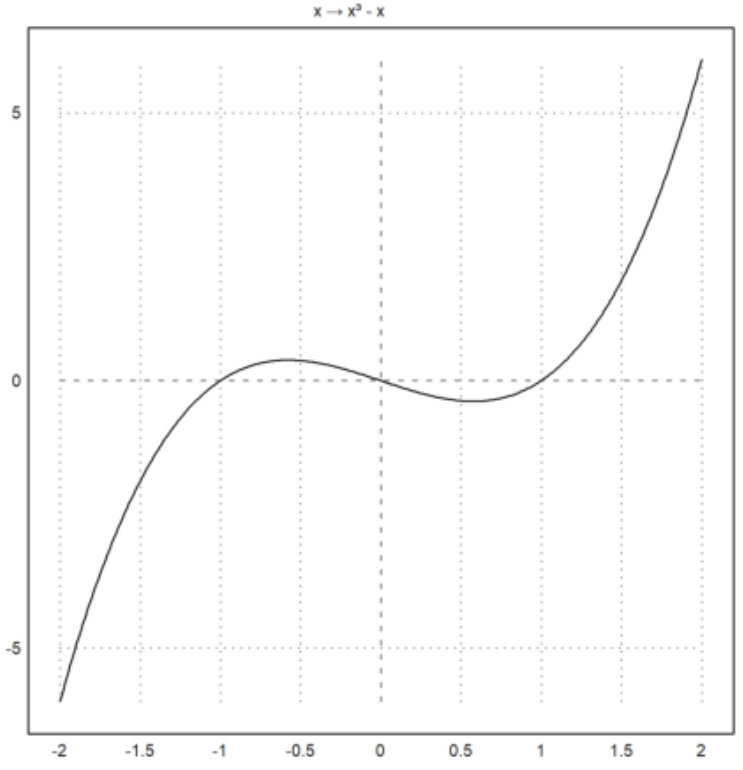
Fungsi lain yang serupa adalah textbox().
Lebar kotak teks secara default mengikuti panjang baris terpanjang, tetapi pengguna juga bisa menentukannya sendiri.

```
>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):
```



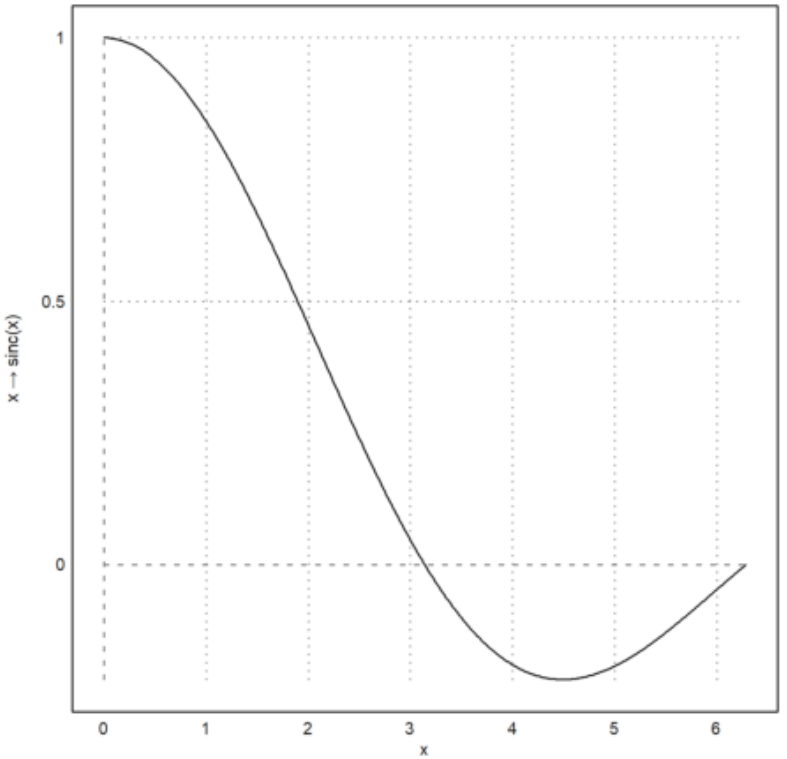
Label teks, judul, kotak label, maupun teks lainnya dapat dituliskan dengan string Unicode (lihat dokumentasi EMT untuk detail penggunaannya).

```
>plot2d("x^3-x",title=u"x &rarr; x&sup3; - x"):
```



Selain itu, label sumbu x maupun y dapat dibuat vertikal, begitu juga orientasi sumbunya.

```
>plot2d("sinc(x)",0,2pi,xl="x",yl=u"x &rarr; sinc(x)",>vertical):
```

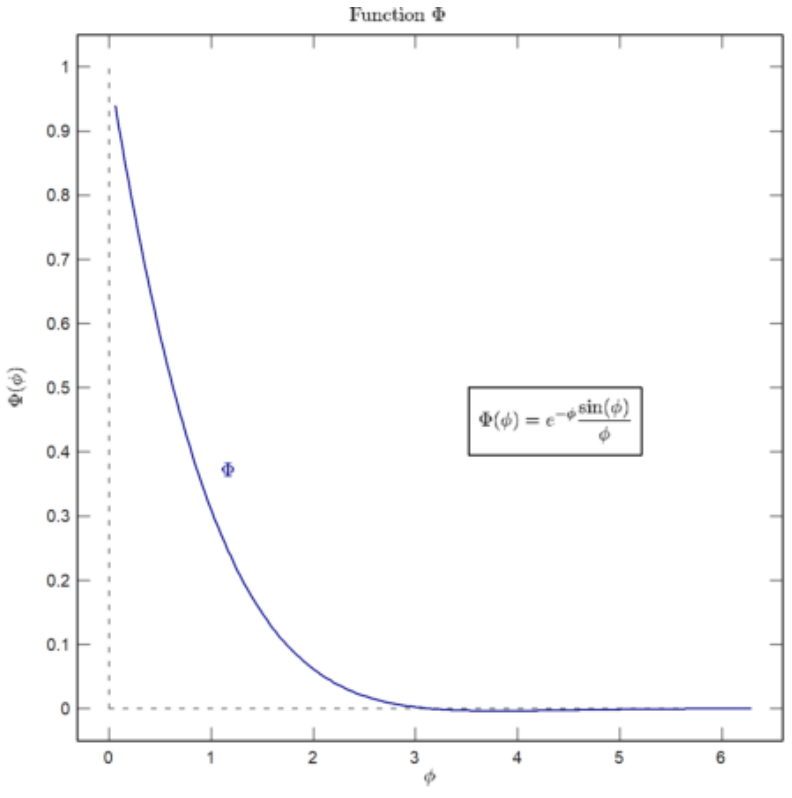


Anda juga dapat mem-plot rumus LaTeX jika sudah menginstal sistem LaTeX. Saya merekomendasikan MiKTeX. Jalur ke file biner "latex" dan "dvipng" harus ada di *system path*, atau Anda harus mengatur LaTeX melalui menu opsi.

Perlu dicatat bahwa proses *parsing* LaTeX cukup lambat. Jika Anda ingin menggunakan LaTeX dalam plot animasi, sebaiknya panggil fungsi 'latex()' sekali sebelum loop, lalu gunakan hasilnya (sebuah gambar dalam matriks RGB).

Pada plot berikut, kita menggunakan LaTeX untuk label sumbu-x dan sumbu-y, sebuah label, kotak label, serta judul plot.

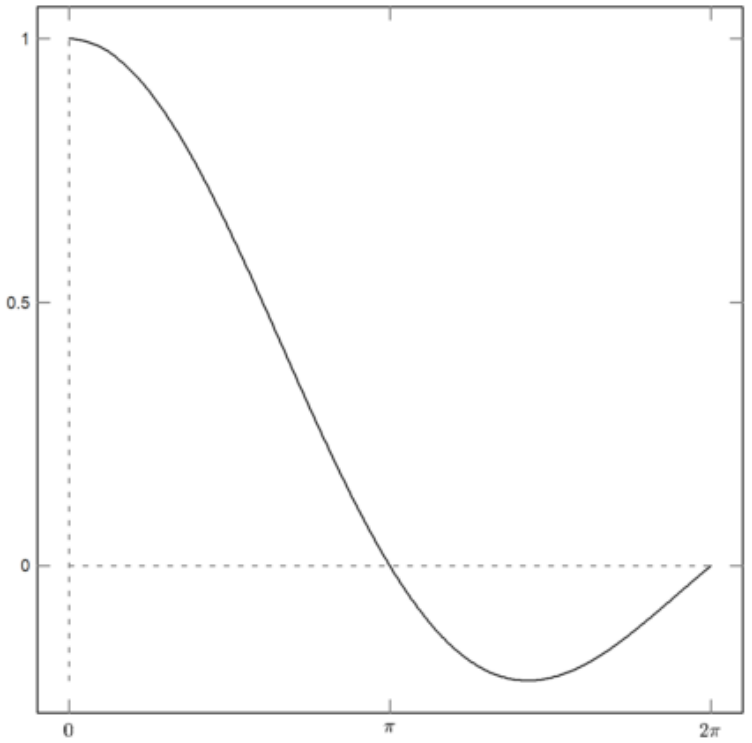
```
>plot2d("exp(-x)*sin(x)/x",a=0,b=2pi,c=0,d=1,grid=6,color=blue, ...
>  title=latex("\text{Function $\Phi$}"), ...
>  xl=latex("\phi"),yl=latex("\Phi(\phi)")); ...
>textbox( ...
>  latex("\Phi(\phi) = e^{-\phi} \frac{\sin(\phi)}{\phi}"),x=0.8,y=0.5); ...
>label(latex("\Phi",color=blue),1,0.4):
```



Sering kali, kita menginginkan jarak yang tidak konformal (*non-conformal spacing*) serta label teks pada sumbu-x. Kita dapat menggunakan ‘xaxis()‘ dan ‘yaxis()‘ seperti yang akan ditunjukkan nanti.

Cara termudah adalah membuat plot kosong dengan bingkai menggunakan ‘grid=4‘, lalu menambahkan garis kisi dengan ‘ygrid()‘ dan ‘xgrid()‘. Pada contoh berikut, kita menggunakan tiga string LaTeX sebagai 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):
```



Of course, functions can also be used.

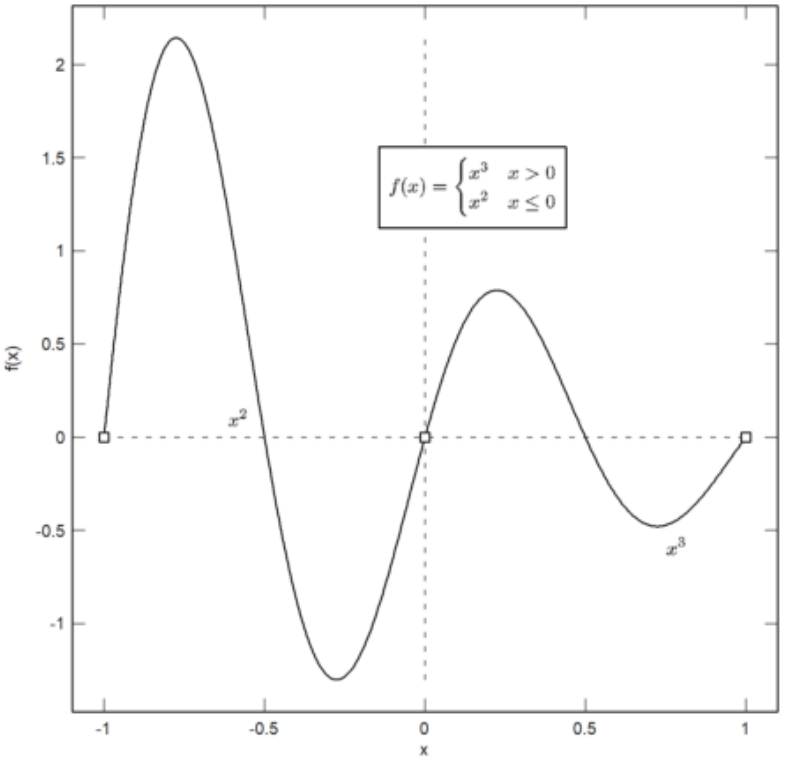
```
>function map f(x) ...
```

```
  if x>0 then return x^4
  else return x^2
  endif
  endfunction
```

Parameter "map" membantu penggunaan fungsi untuk vektor. Untuk plot, sebenarnya tidak diperlukan. Namun, untuk menunjukkan bahwa vektorisasi itu berguna, kita tambahkan beberapa titik penting pada plot di $x=-1$, $x=0$, dan $x=1$.

Pada plot berikut, kita juga memasukkan beberapa kode LaTeX. Kode tersebut digunakan untuk dua label dan sebuah kotak teks. Tentu saja, Anda hanya dapat menggunakan LaTeX jika LaTeX sudah terinstal 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):
```


$f(x) = \begin{cases} x^3 & x > 0 \\ x^2 & x \leq 0 \end{cases}$
x^2
x^3
f(x)
x
2
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Saat mem-plot sebuah fungsi atau ekspresi, parameter ‘>user‘ memungkinkan pengguna untuk melakukan zoom dan menggeser plot menggunakan tombol kursor atau mouse. Pengguna dapat:

- melakukan zoom dengan tombol ‘+‘ atau ‘-‘
- menggeser plot dengan tombol kursor
- memilih jendela plot dengan mouse
- mengatur ulang tampilan dengan tombol spasi
- keluar dengan tombol return

Tombol spasi akan mengembalikan plot ke jendela plot semula.

Saat mem-plot sebuah data, flag ‘>user‘ hanya akan menunggu penekanan tombol (keystroke).

```
>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)"):
```

Contoh berikut menunjukkan cara lanjutan untuk interaksi pengguna (lihat tutorial tentang pemrograman untuk detail lebih lanjut).

Fungsi bawaan ‘mousedrag()‘ menunggu event dari mouse atau keyboard. Fungsi ini melaporkan klik mouse, pergerakan mouse, pelepasan klik mouse, serta penekanan tombol. Fungsi ‘dragpoints()‘ memanfaatkan hal ini, dan memungkinkan pengguna untuk menyeret (drag) titik apa pun dalam sebuah plot.

Pertama, kita memerlukan fungsi plot. Sebagai contoh, kita lakukan interpolasi pada 5 titik menggunakan polinomial. Fungsi tersebut harus mem-plot ke dalam area plot yang tetap (fixed plot area).

```
>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 pada plot2d (d dan xp), yang diteruskan ke evaluasi fungsi interp(). Tanpa ini, kita harus menuliskan fungsi plotinterp() terlebih dahulu, dengan mengakses nilai secara global.

Sekarang kita akan menghasilkan beberapa nilai acak, lalu membiarkan pengguna menyeret (drag) titik-titik tersebut.

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

Ada juga sebuah fungsi yang mem-plot fungsi lain yang bergantung pada sebuah vektor parameter, dan memungkinkan pengguna untuk menyesuaikan parameter-parameter tersebut.

Pertama, kita memerlukan fungsi plotnya.

```
>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 sebuah matriks nx2 yang berisi rentang nilai, serta secara opsional sebuah baris judul (heading line).

Terdapat slider interaktif yang memungkinkan pengguna mengatur nilai parameter. Fungsi ‘dragvalues()‘ menyediakan fasilitas ini.

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

Dimungkinkan untuk membatasi nilai yang digeser (dragged values) hanya pada bilangan bulat. Sebagai contoh, kita dapat menuliskan sebuah fungsi plot yang mem-plot polinomial Taylor dengan derajat n terhadap 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 mengizinkan derajat n bervariasi dari 0 hingga 20 dalam 20 langkah. Hasil dari dragvalues() digunakan untuk mem-plot sketsa dengan nilai n tersebut, lalu menyisipkan plot ke dalam notebook.

```
>nd=dragvalues("plotf","degree",2,[0,20],20,y=0.8, ...
>   heading="Drag the value:"); ...
>plotf(nd):
```

Berikut ini adalah demonstrasi sederhana dari fungsi tersebut. Pengguna dapat menggambar di atas jendela plot, sehingga meninggalkan jejak titik-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!
```

Secara bawaan (default), EMT menghitung tanda sumbu (axis ticks) secara otomatis dan menambahkan label pada setiap tanda. Hal ini dapat diubah dengan parameter ‘grid‘. Gaya bawaan dari sumbu dan label juga dapat dimodifikasi. Selain itu, label dan judul dapat ditambahkan secara manual. Untuk mengembalikan ke gaya bawaan, gunakan ‘reset()‘.

```
>aspect();
>figure(3,4); ...
> figure(1); plot2d("x^3-x",grid=0); ... // no grid, frame or axis
> figure(2); plot2d("x^3-x",grid=1); ... // x-y-axis
> figure(3); plot2d("x^3-x",grid=2); ... // default ticks
> figure(4); plot2d("x^3-x",grid=3); ... // x-y- axis with labels inside
> 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):
```

Parameter <frame mematikan bingkai (frame), dan framecolor=blue mengatur bingkai menjadi berwarna biru.
Jika Anda ingin membuat tanda sumbu (ticks) sendiri, Anda dapat menggunakan style=0, lalu menambahkan semuanya secara manual setelahnya.

```
>aspect(1.5);
>plot2d("x^3-x",grid=0); // plot
>frame; xgrid([-1,0,1]); ygrid(0): // add frame and grid
```

Untuk judul plot dan label pada sumbu, lihat contoh berikut.

```
>plot2d("exp(x)",-1,1);
>textcolor(black); // set the text color to black
>title(latex("y=e^x")); // title above the plot
>xlabel(latex("x")); // "x" for x-axis
>ylabel(latex("y"),>vertical); // vertical "y" for y-axis
>label(latex("(0,1)"),0,1,color=blue): // label a point
```

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(). Pada contoh berikut, "lc" berarti lower center. Kode ini menentukan posisi label relatif terhadap koordinat plot.

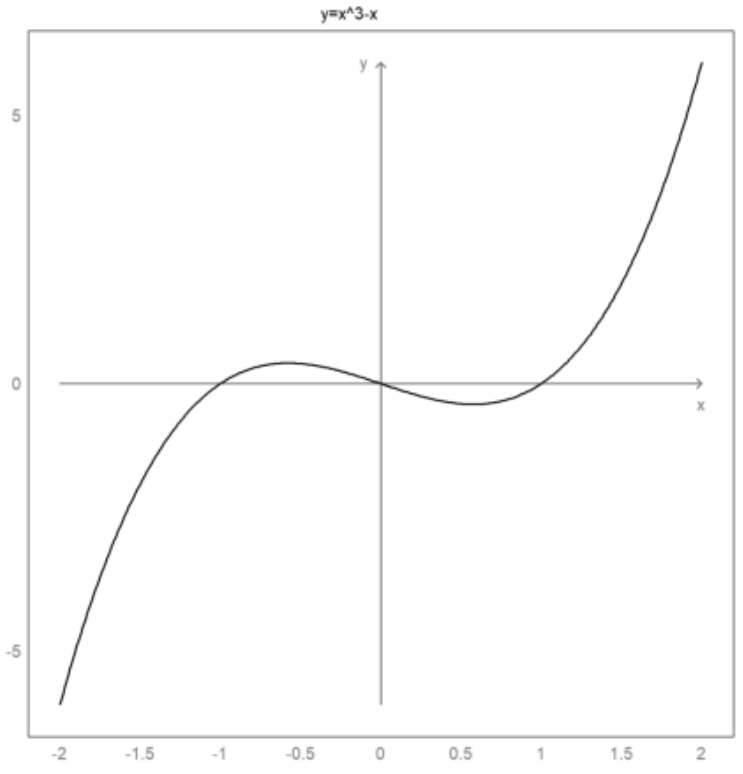
```
>function f(x) &= x^3-x
```

$$x^3 - x$$

```
>plot2d(f,-1,1,>square);
>x0=fmin(f,0,1); // compute point of minimum
>label("Rel. Min.",x0,f(x0),pos="lc"): // add a label there
```

There are also text boxes.

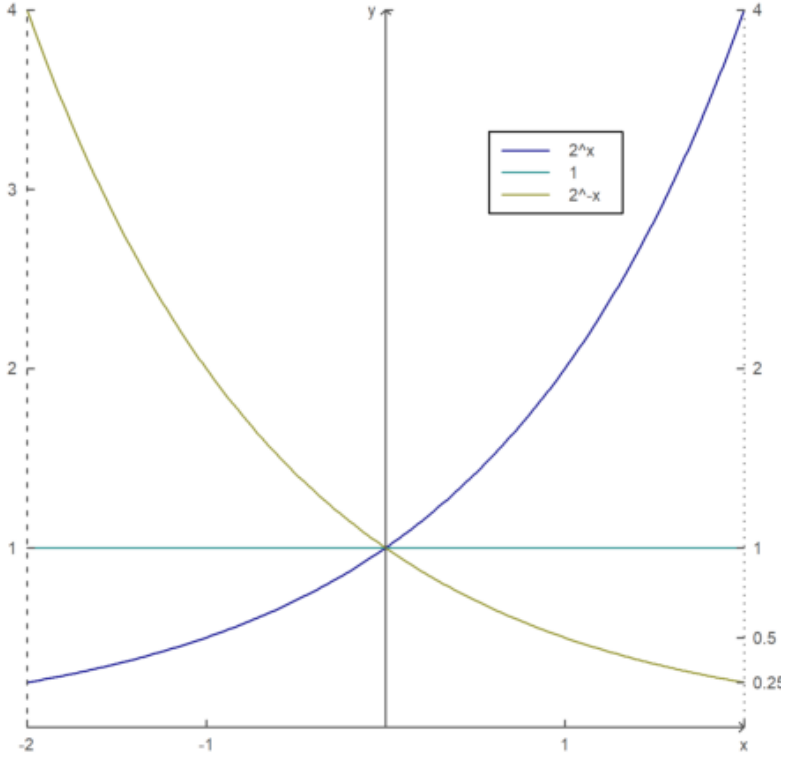
```
>plot2d(&f(x),-1,1,-2,2); // function
>plot2d(&diff(f(x),x),>add,style="--",color=red); // derivative
>labelbox(["f","f'"],["-","--"],[black,red]): // label box
>plot2d(["exp(x)","1+x"],color=[black,blue],style=["-","-."]):
>gridstyle("->",color=gray,textcolor=gray,framecolor=gray);  ...
> plot2d("x^3-x",grid=1);   ...
> settitle("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 yang lebih lanjut, sumbu-x dan sumbu-y dapat dibuat secara manual.

Perintah fullwindow() memperluas jendela plot karena kita tidak lagi membutuhkan ruang untuk label di luar jendela plot. Gunakan shrinkwindow() atau reset() untuk mengembalikan ke pengaturan bawaan.

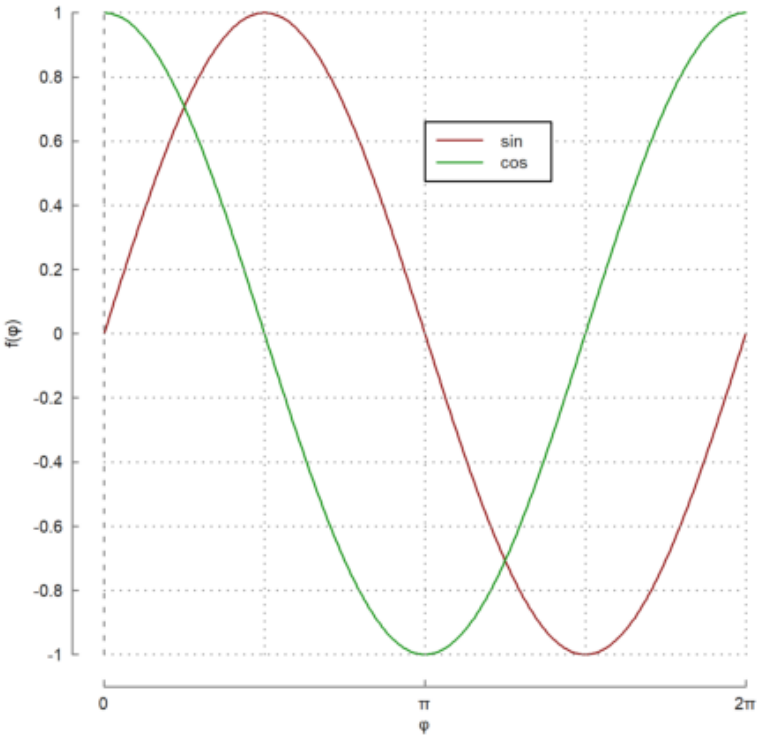
```
>fullwindow; ...
> gridstyle(color=darkgray,textcolor=darkgray); ...
> plot2d(["2^x","1","2^(-x)"],a=-2,b=2,c=0,d=4,<grid,color=4:6,<frame); ...
> xaxis(0,-2:1,style="->"); xaxis(0,2,"x",<axis); ...
> yaxis(0,4,"y",style="->"); ...
> yaxis(-2,1:4,>left); ...
> yaxis(2,2^(-2:2),style=".",<left); ...
> labelbox(["2^x","1","2^-x"],colors=4:6,x=0.8,y=0.2); ...
> reset:
```



Berikut contoh lain, di mana string Unicode digunakan dan sumbu diletakkan 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;)"):
```



Memplot Data 2D

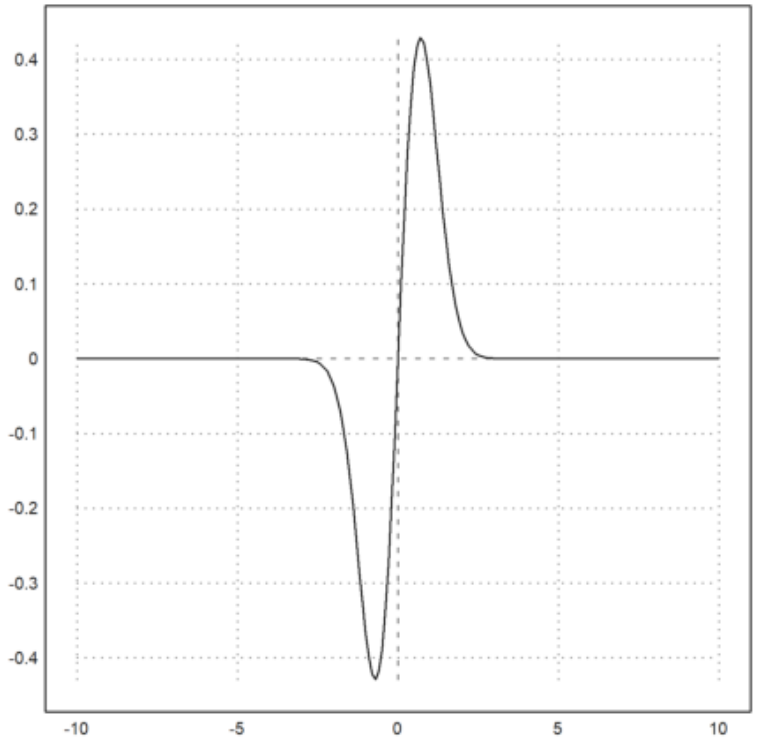
Jika x dan y berupa vektor data, maka data tersebut akan digunakan sebagai koordinat x dan y dari sebuah kurva. Dalam kasus ini, a, b, c, d, atau sebuah radius r dapat ditentukan, atau jendela plot akan menyesuaikan secara otomatis dengan data. Sebagai alternatif, dapat digunakan >square untuk menjaga rasio aspek persegi.
Memplot sebuah ekspresi hanyalah bentuk singkatan dari plot data. Untuk plot data, Anda memerlukan satu atau lebih baris nilai-x, dan satu atau lebih baris nilai-y. Dari rentang dan nilai-x tersebut, fungsi plot2d akan menghitung data yang akan diplot, secara bawaan dengan evaluasi adaptif dari fungsi.
Untuk plot titik gunakan ">points".
Untuk gabungan garis dan titik gunakan ">addpoints".

Namun, Anda juga bisa langsung memasukkan data.

- Gunakan vektor baris untuk x dan y jika hanya satu fungsi.
- Matriks untuk x dan y akan diplot baris demi baris.

Berikut contoh dengan satu baris untuk x dan y.

```
>x=-10:0.1:10; y=exp(-x^2)*x; plot2d(x,y):
```



Data juga dapat dipetakan sebagai titik. Gunakan ‘points=true‘ untuk ini. Plot bekerja mirip dengan poligon, tetapi hanya menggambar sudut-sudutnya.

‘style=”...”‘: Pilih dari ‘”[]”‘, ‘”<>”‘, ‘”o”‘, ‘”.”‘, ‘”.. ”‘, ‘”+”‘, ‘”*”‘, ‘”[]”‘, ‘”<> ”‘, ‘”o”‘, ‘”..”‘, ‘””‘, ‘”|”‘.

Untuk memplot himpunan titik, gunakan ‘>points‘. Jika warna berupa vektor warna, maka setiap titik akan mendapat warna yang berbeda. Untuk sebuah matriks koordinat dan vektor kolom, warna berlaku pada setiap baris dari 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="."); // lines
>plot2d(xdata,ydata,>points,>add,style="o"): // add points
>p=polyfit(xdata,ydata,1); // get regression line
>plot2d("polyval(p,x)",>add,color=red): // add plot of line
```

Menggambar Daerah yang Dibatasi Kurva

Plot data pada dasarnya adalah poligon. Kita juga dapat memplot kurva atau kurva yang diisi (*filled curves*).

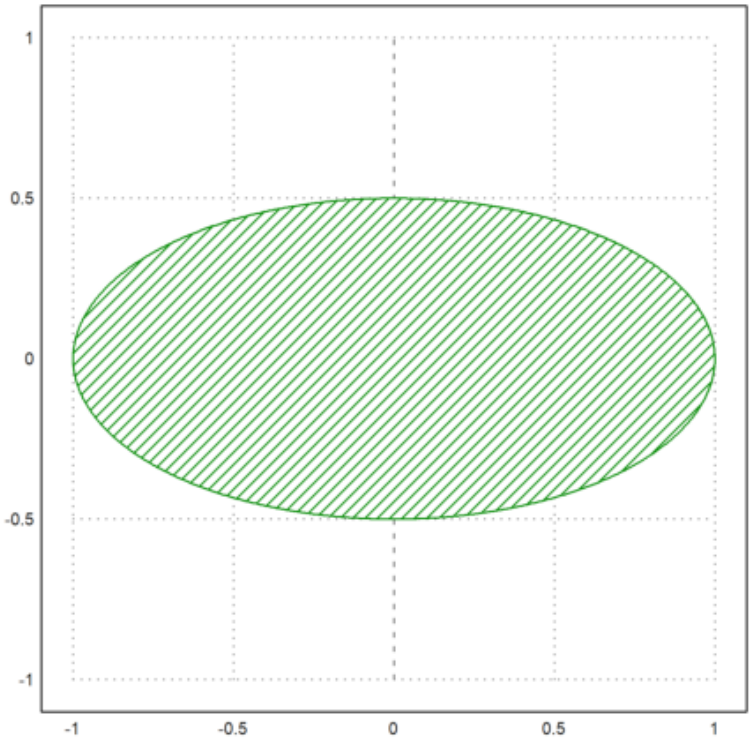
- ‘filled=true‘ mengisi area plot.
- ‘style="..."‘: Pilih dari ‘""‘, ‘"/"‘, ‘"\"‘, ‘"\/"‘.
- ‘fillcolor‘: Lihat daftar warna yang tersedia pada bagian sebelumnya.

Warna isian ditentukan oleh argumen ‘fillcolor‘, dan opsi ‘<outline‘ dapat digunakan untuk mencegah penggambaran garis batas untuk semua gaya, kecuali gaya bawaan (default).

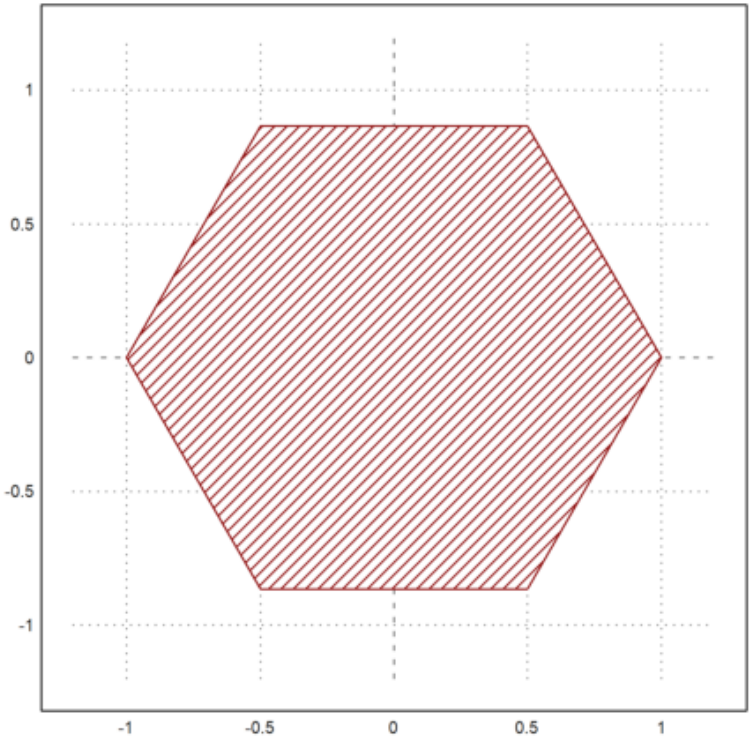
```
>t=linspace(0,2pi,1000); // parameter for curve
>x=sin(t)*exp(t/pi); y=cos(t)*exp(t/pi); // x(t) and y(t)
>figure(1,2); aspect(16/9)
>figure(1); plot2d(x,y,r=10); // plot curve
>figure(2); plot2d(x,y,r=10,>filled,style="/",fillcolor=red); // fill curve
>figure(0):
```

Pada contoh berikut, kita memplot sebuah elips terisi (filled ellipse) dan dua heksagon terisi menggunakan kurva tertutup dengan 6 titik, masing-masing dengan gaya isian (fill style) yang berbeda.

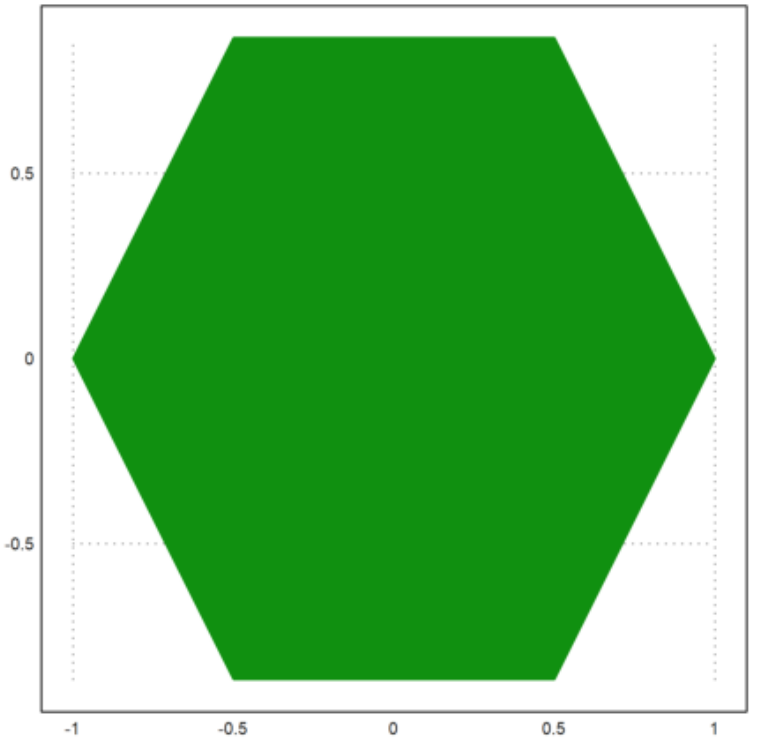
```
>x=linspace(0,2pi,1000); plot2d(sin(x),cos(x)*0.5,r=1,>filled,style="/"):
```




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

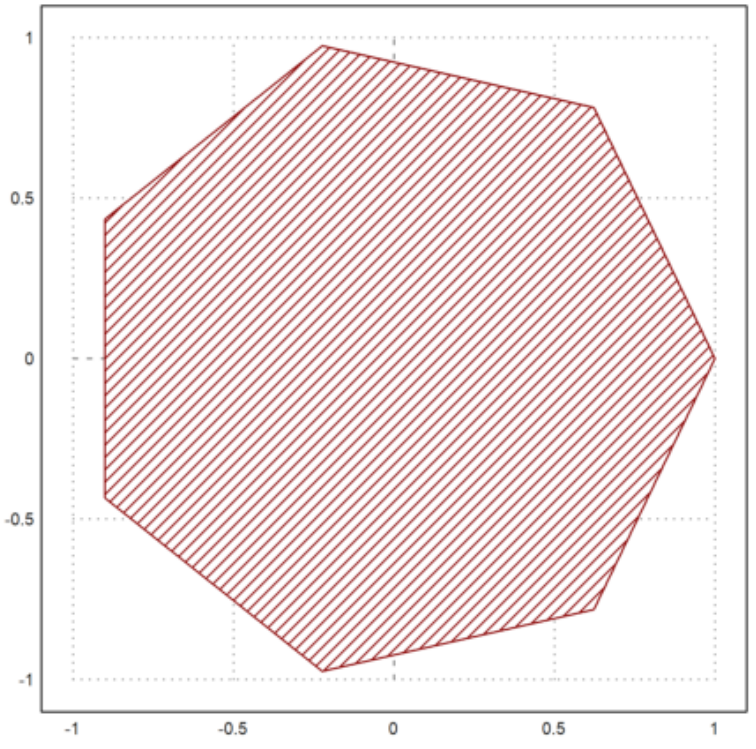



```
>t=linspace(0,2pi,6); plot2d(cos(t),sin(t),>filled,style="#"):
```



Contoh lainnya adalah sebuah segi tujuh (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 ini adalah himpunan nilai maksimum dari empat kondisi linear yang kurang dari atau sama dengan 3. Bentuk umumnya adalah $A[k] \cdot v \leq 3$ untuk semua baris matriks A. Untuk mendapatkan sudut yang lebih halus, kita menggunakan nilai 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):
```

The main point of the matrix language is that it allows to generate tables of functions easily.

```
>t=linspace(0,2pi,1000); x=cos(3*t); y=sin(4*t);
```

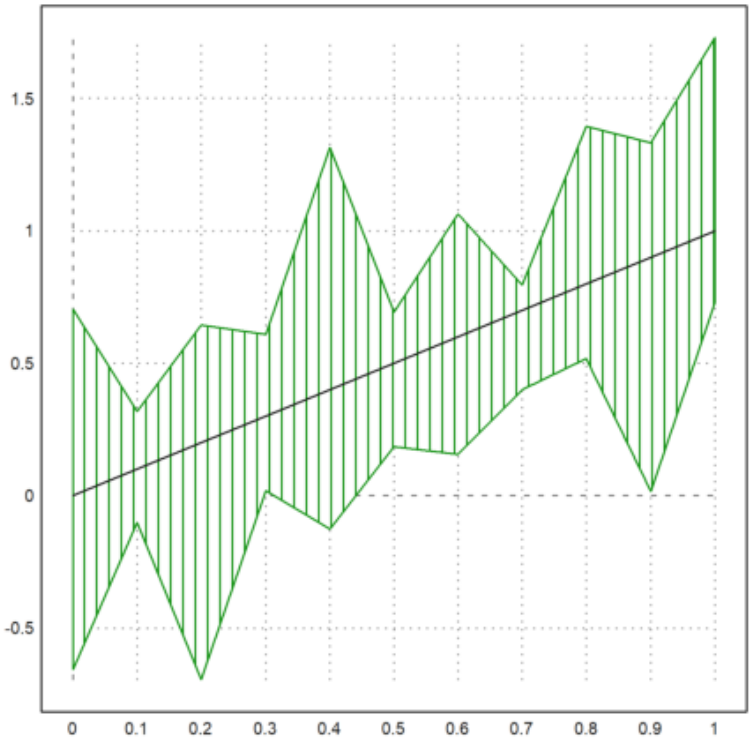
We now have vectors x and y of values. plot2d() can plot these values as a curve connecting the points. The plot can be filled. In this case this yields a nice result due to the winding rule, which is used for the fill.

```
>plot2d(x,y,<grid,<frame,>filled):
```

A vector of intervals is plotted against x values as a filled region between lower and upper values of the intervals.

This is can be useful to plot errors of the computation. But it can also be used to plot statistical errors.

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



If x is a sorted vector, and y is a vector of intervals, then plot2d will plot the filled ranges of the intervals in the plane.The fill styles are the same as the styles of polygons.

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

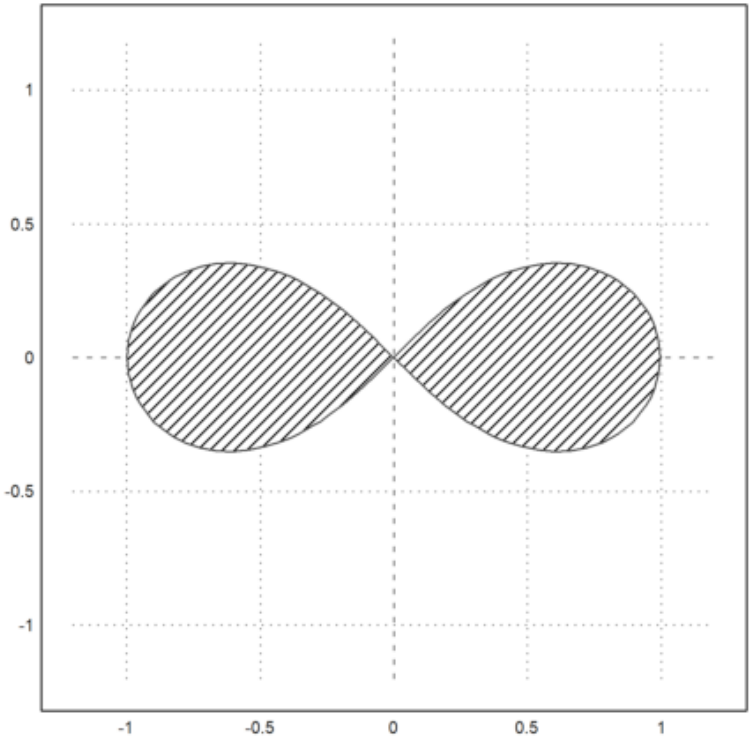
It is possible to fill regions of values for a specific function. For this, level must be a 2xn matrix. The first row are the lower bounds and the second row contains the upper bounds.

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

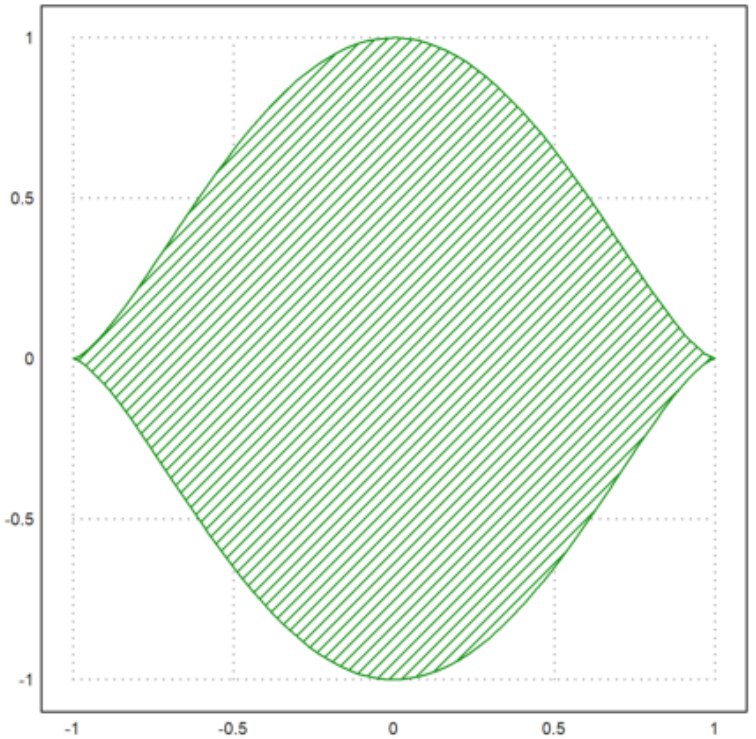
We can also fill ranges of values like

$$-1 \leq (x^2+y^2)^2 - x^2 + y^2 \leq 0.$$

```
>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="/"):
```


1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Grafik Fungsi Parametrik

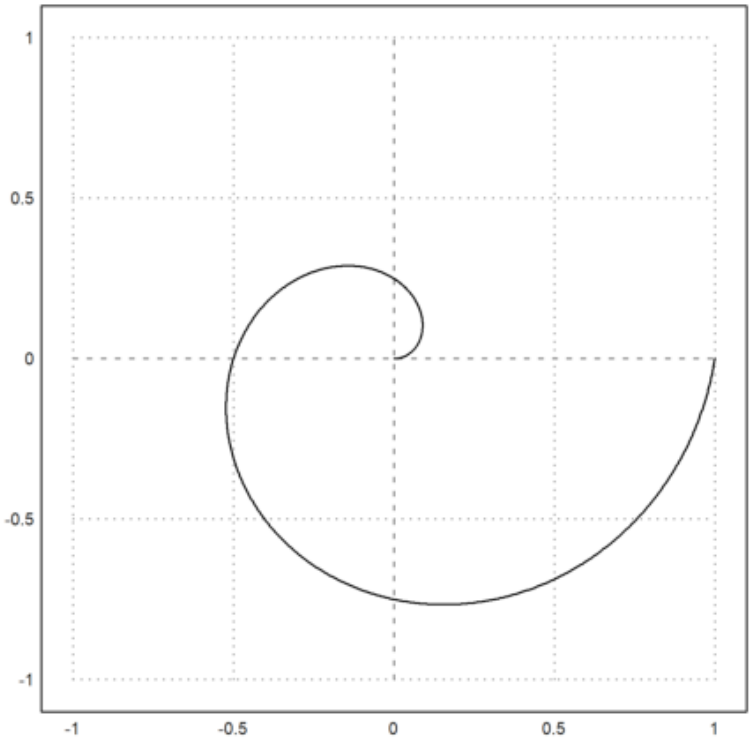
Nilai-nilai x tidak harus terurut. Pasangan (x,y) hanya mendeskripsikan sebuah kurva. Jika x terurut, maka kurva tersebut merupakan grafik dari suatu fungsi.

Pada contoh berikut, kita memplot spiral

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

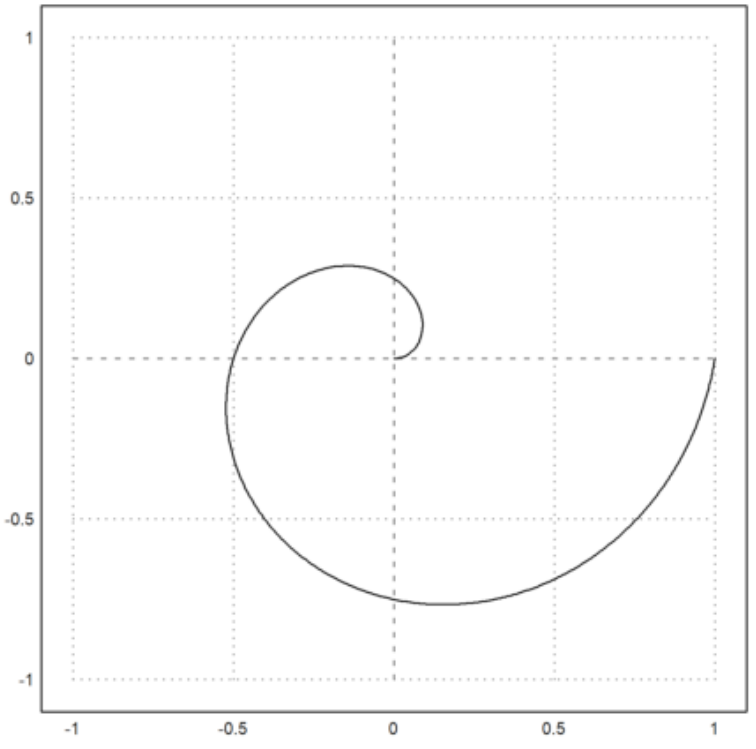
Kita perlu menggunakan banyak titik agar hasilnya terlihat mulus, atau menggunakan fungsi ‘adaptive()‘ untuk mengevaluasi ekspresi (lihat penjelasan fungsi ‘adaptive()‘ untuk detail lebih lanjut).

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



Alternatively, it is possible to use two expressions for curves. The following plots the same curve as above.

```
>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(2pi*t); y=r*sin(2pi*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):
```

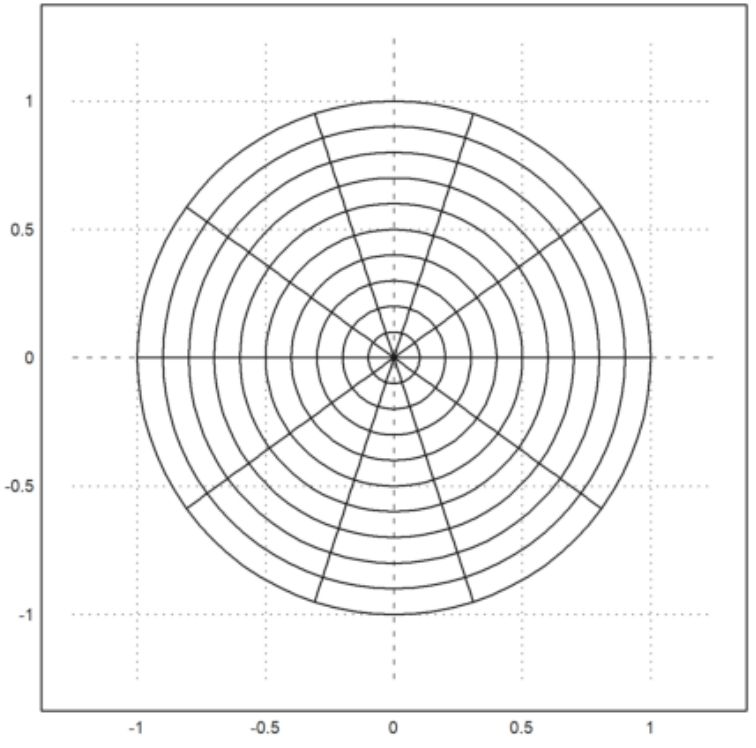
Menggambar Grafik Bilangan Kompleks

An array of complex numbers can also be plotted. Then the grid points will be connected. If a number of grid lines is specified (or a 1x2 vector of grid lines) in the argument cgrid only those grid lines are visible.

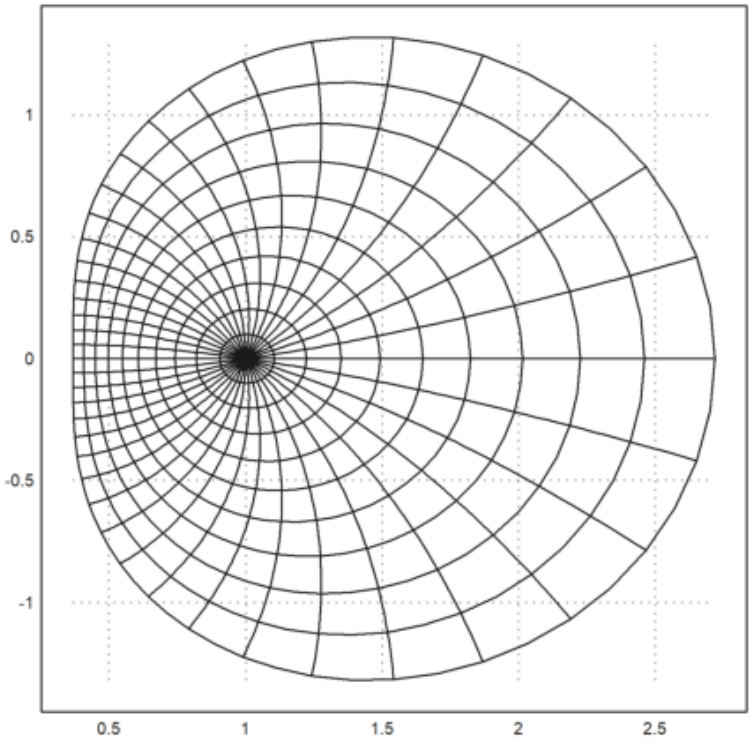
A matrix of complex numbers will automatically plot as a grid in the complex plane.

In the following example, we plot the image of the unit circle under the exponential function. The cgrid parameter hides some of the grid curves.

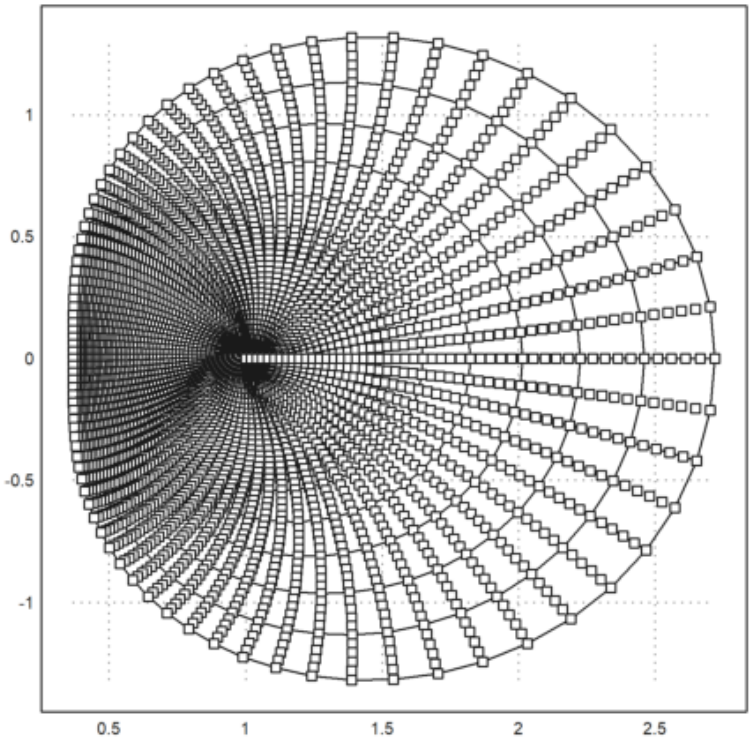
```
>aspect(); r=linspace(0,1,50); a=linspace(0,2pi,80)'; z=r*exp(I*a);...
>plot2d(z,a=-1.25,b=1.25,c=-1.25,d=1.25,cgrid=10):
```




```
>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):
```

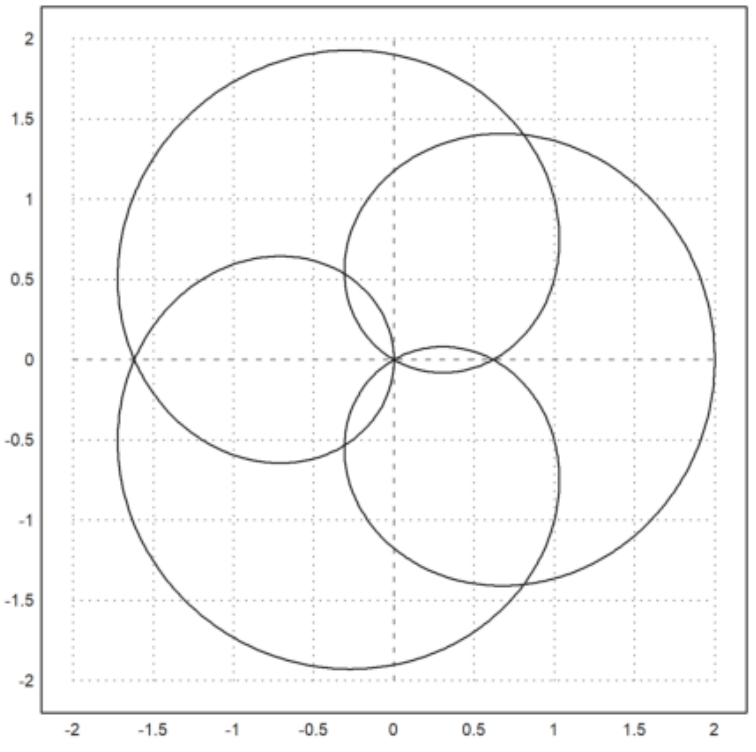


A vector of complex numbers is automatically plotted as a curve in the complex plane with real part and imaginary part.

In the example, we plot the unit circle with

$$\gamma(t) = e^{it}$$

```
>t=linspace(0,2pi,1000); ...
>plot2d(exp(I*t)+exp(4*I*t),r=2):
```



Statistical Plots

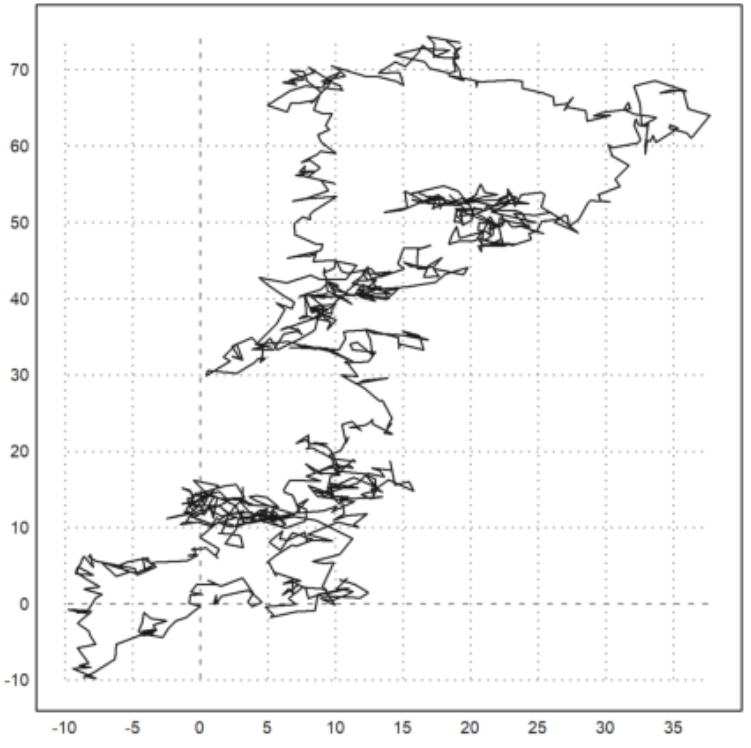
There are many functions which are specialized on statistical plots. One of the often used plots is a column plot.

A cumulative sum of a 0-1-normal distributed values produces a random walk.

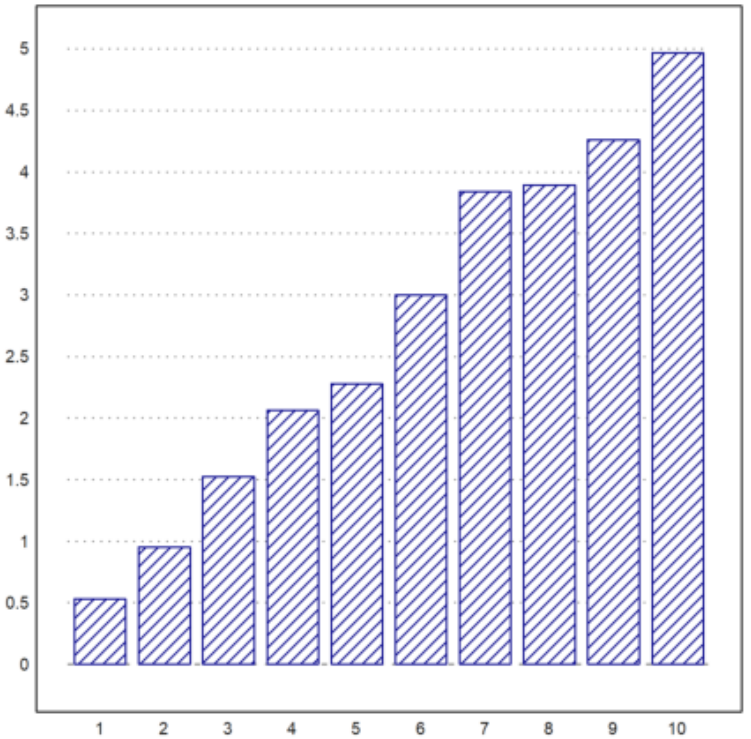
```
>plot2d(cumsum(randnormal(1,1000))):
```

Using two rows shows a walk in two dimensions.

```
>X=cumsum(randnormal(2,1000)); plot2d(X[1],X[2]):
```

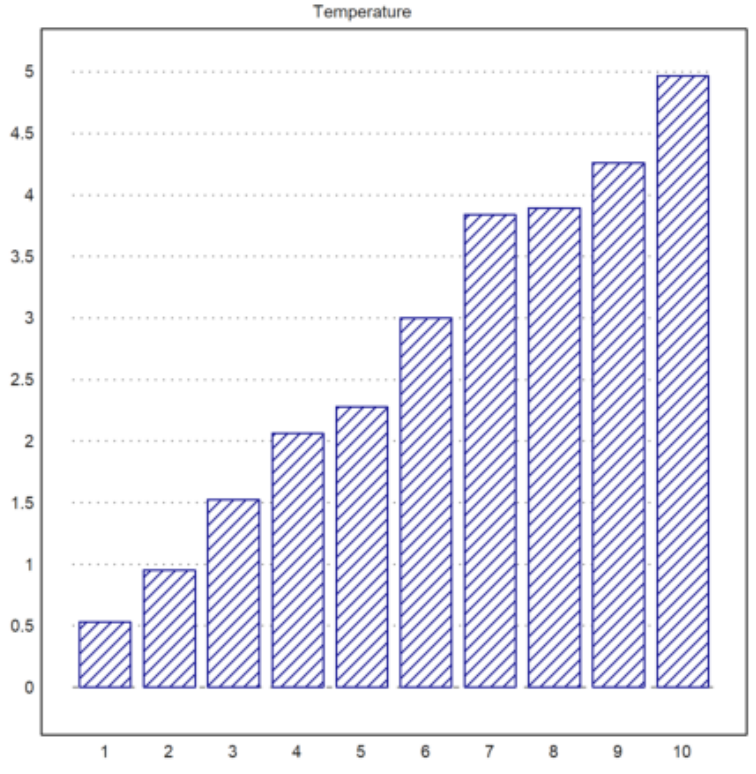



```
>columnsplot(cumsum(random(10)),style="/",color=blue):
```



It can also show strings as labels.

```
>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):
```

To plot an experimental statistical distribution, you can use distribution=n with plot2d.

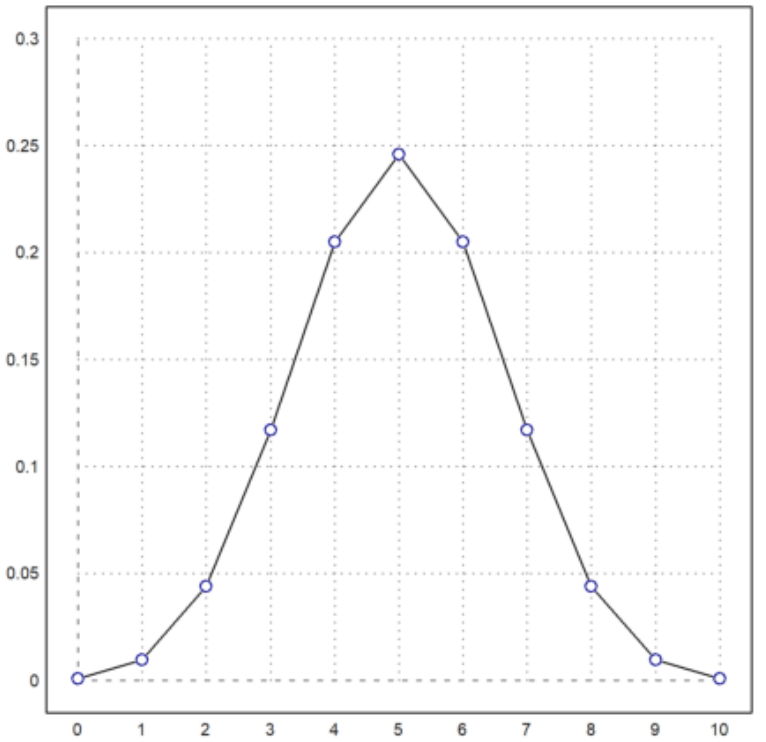
```
>w=randexponential(1,1000); // exponential distribution
>plot2d(w,>distribution): // or distribution=n with n intervals
```

Or you can compute the distribution from the data and plot the result with >bar in plot3d, or with a column plot.

```
>w=normal(1000); // 0-1-normal distribution
>{x,y}=histo(w,10,v=[-6,-4,-2,-1,0,1,2,4,6]); // interval bounds v
>plot2d(x,y,>bar):
```

The statplot() function sets the style with a simple string.

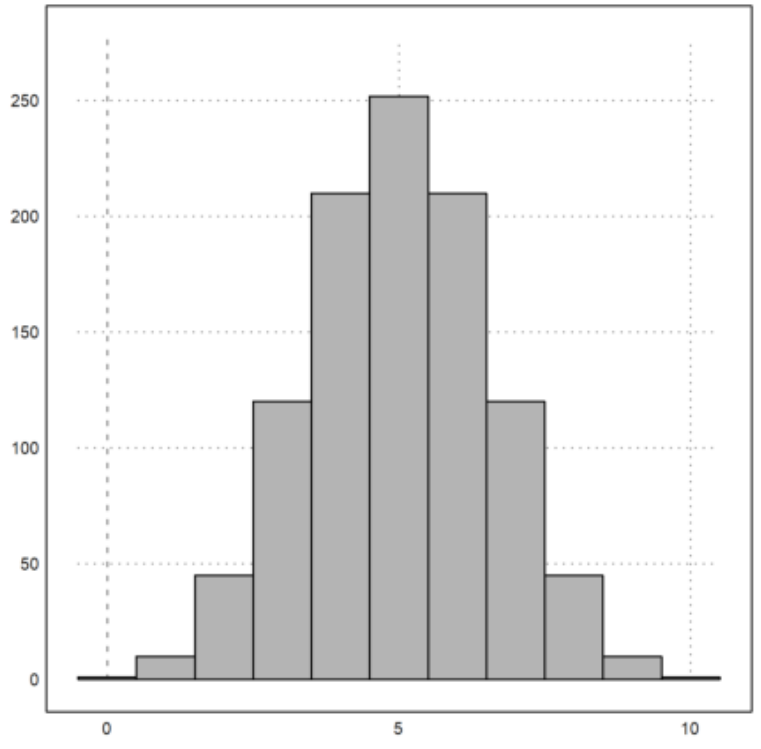
```
>statplot(1:10,cumsum(random(10)),"b"):
>n=10; i=0:n; ...
>plot2d(i,bin(n,i)/2^n,a=0,b=10,c=0,d=0.3); ...
>plot2d(i,bin(n,i)/2^n,points=true,style="ow",add=true,color=blue):
```



Moreover, data can be plotted as bars. In this case, x should be sorted and one element longer than y. The bars will extend from x[i] to x[i+1] with values y[i]. If x has the same size as y, it will be extended by one element with the last spacing.

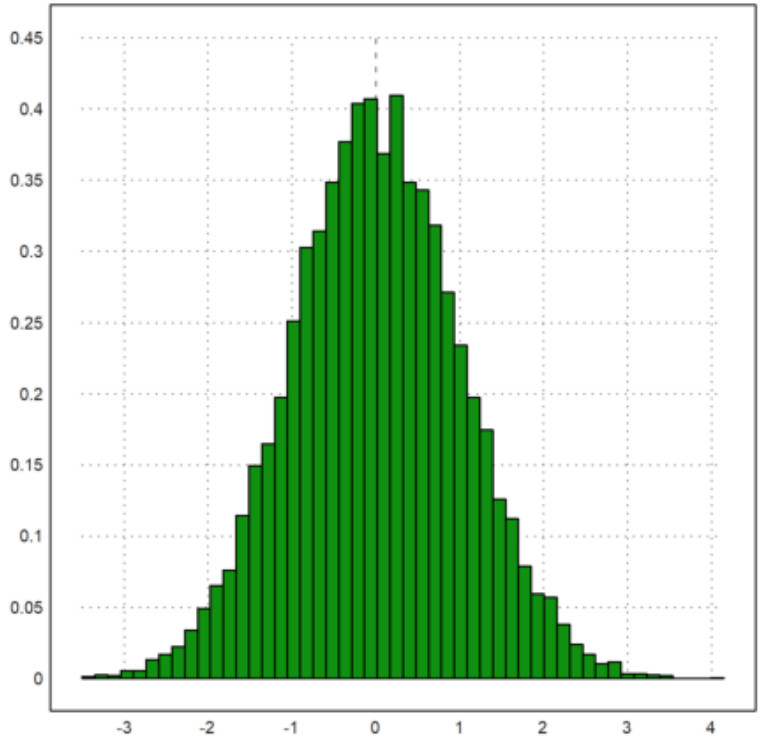
Fill styles can be used just as above.

```
>n=10; k=bin(n,0:n); ...
>plot2d(-0.5:n+0.5,k,bar=true,fillcolor=lightgray):
```

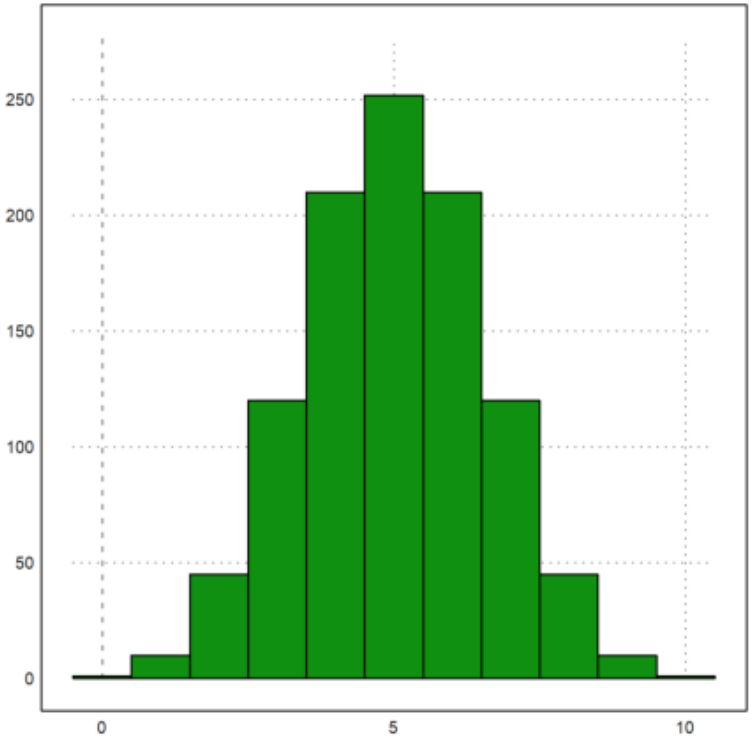


Data untuk diagram batang (bar plots) (bar=1) dan histogram (histogram=1) dapat diberikan secara eksplisit dalam xv dan yv, atau dapat dihitung dari distribusi empiris dalam xv dengan >distribution (atau distribution=n). Histogram dari nilai xv akan dihitung secara otomatis dengan >histogram. Jika >even ditentukan, maka nilai xv akan dihitung dalam interval bilangan bulat.

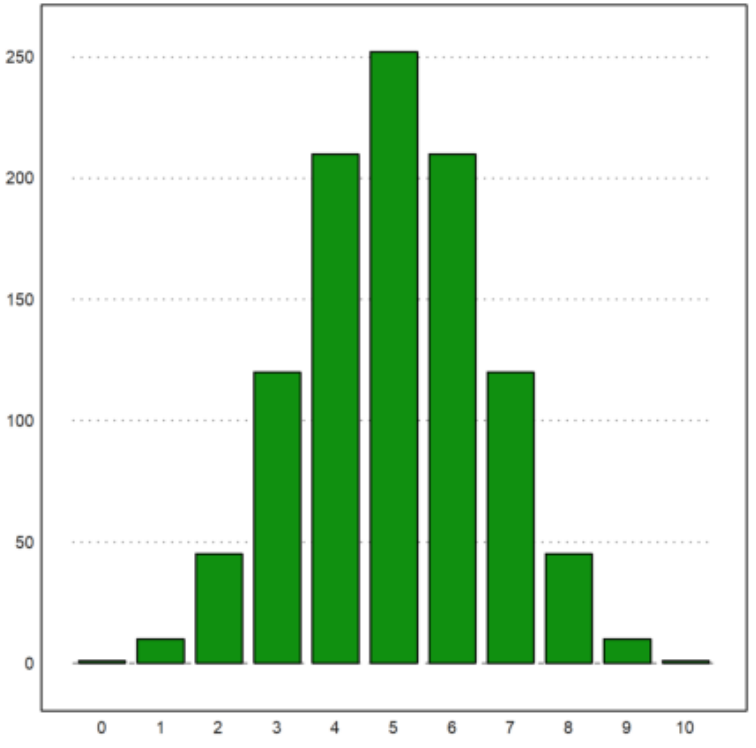
```
>plot2d(normal(10000),distribution=50):
```



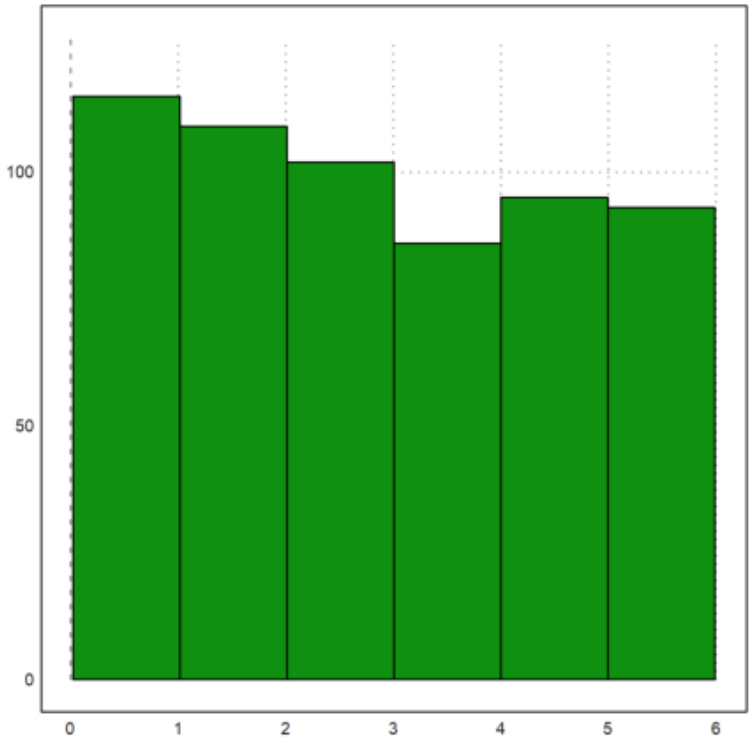

```
>k=0:10; m=bin(10,k); x=(0:11)-0.5; plot2d(x,m,>bar):
```




```
>columnsplot(m,k):
```

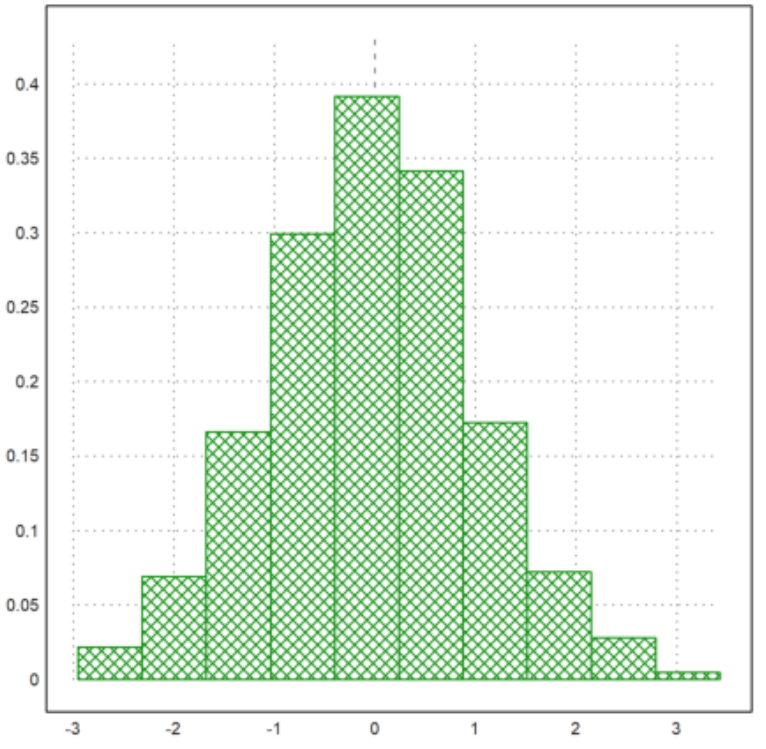



```
>plot2d(random(600)*6,histogram=6):
```



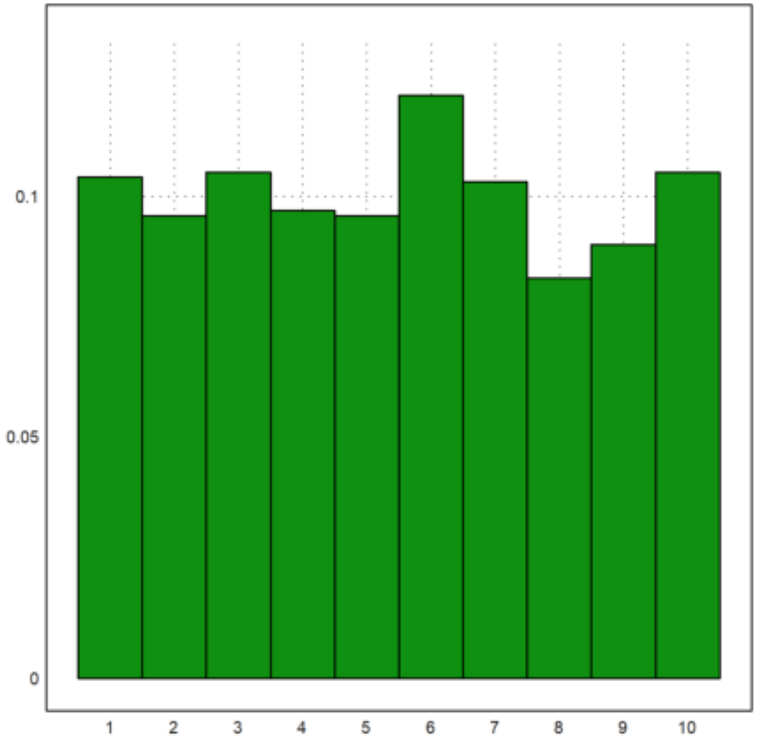
For distributions, there is the parameter distribution=n, which counts values automatically and prints the relative distribution with n sub-intervals.

```
>plot2d(normal(1,1000),distribution=10,style="\/"):
```



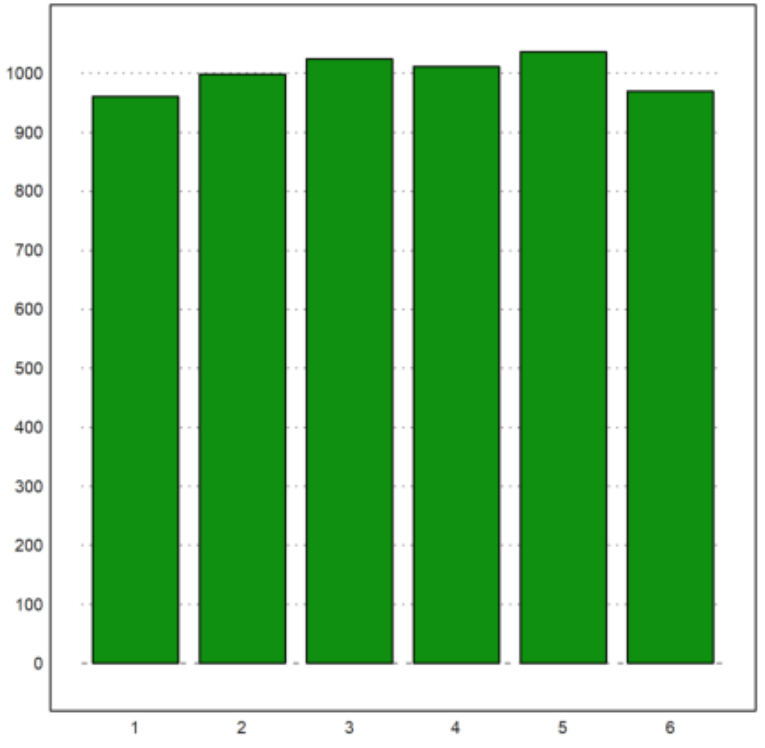
With the parameter even=true, this will use integer intervals.

```
>plot2d(intrandom(1,1000,10),distribution=10,even=true):
```

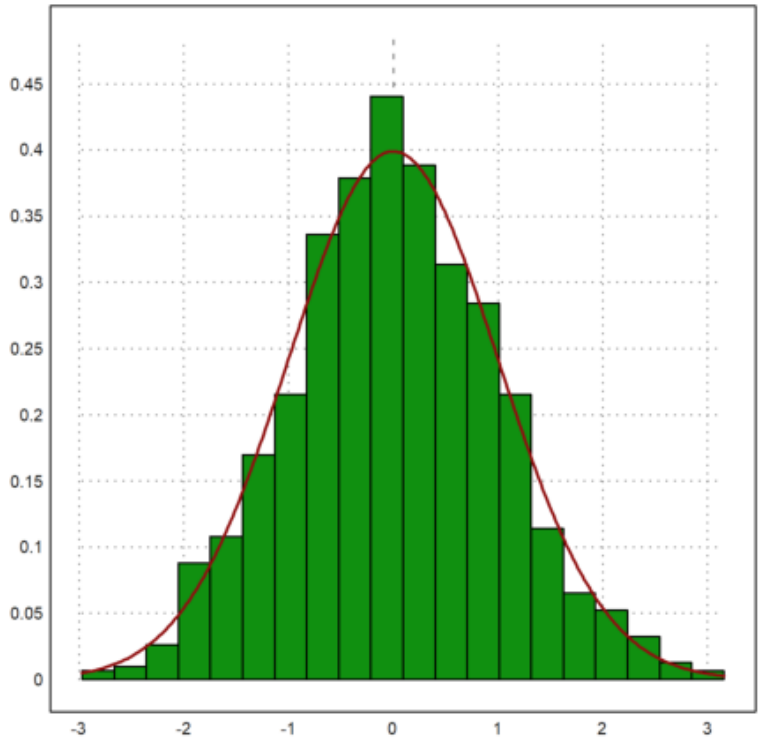


Note that there are many statistical plots, which might be useful. Have a look at the tutorial about statistics.

```
>columnsplot(getmultiplicities(1:6,intrandom(1,6000,6))):
```

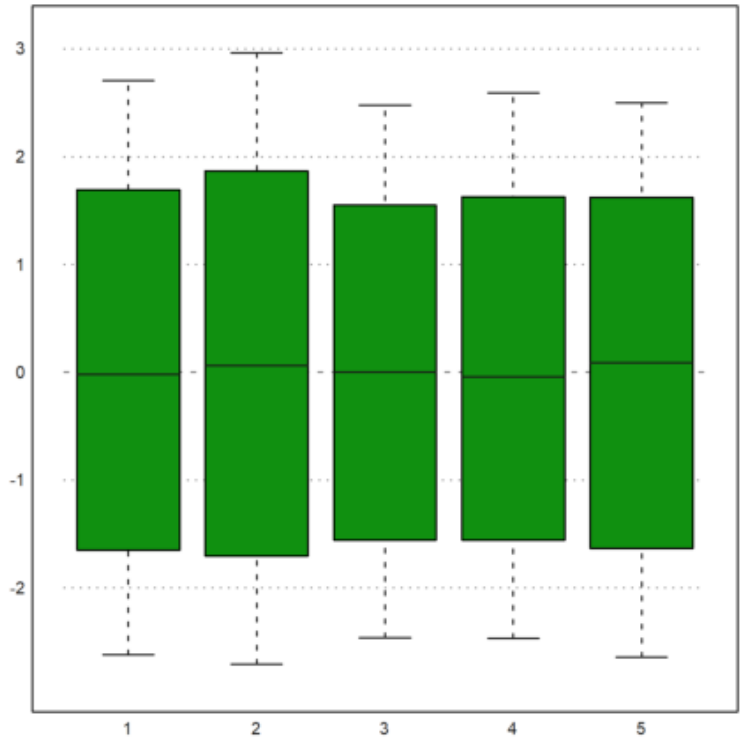



```
>plot2d(normal(1,1000),>distribution); ...
>  plot2d("qnormal(x)",color=red,thickness=2,>add):
```



There are also many special plots for statistics. A boxplot shows the quartiles of this distribution and lots of outliers. By definition, outliers in a boxplot are data which exceed 1.5 times the middle 50% range of the plot.

```
>M=normal(5,1000); boxplot(quartiles(M)):
```


3
2
1
0
-1
-2
1
2
3
4
5

Implicit Functions

Implicit plots show level lines solving f(x,y)=level, where "level" can be a single value or a vector of values. If level="auto", there will be nc level lines, which will spread between the minimum and the maximum of the function evenly. Darker or lighter color can be added with >hue to indicate value of the function. For implicit functions, xv must be a function or an expression of the parameters x and y, or, alternatively, xv can be a matrix of values.

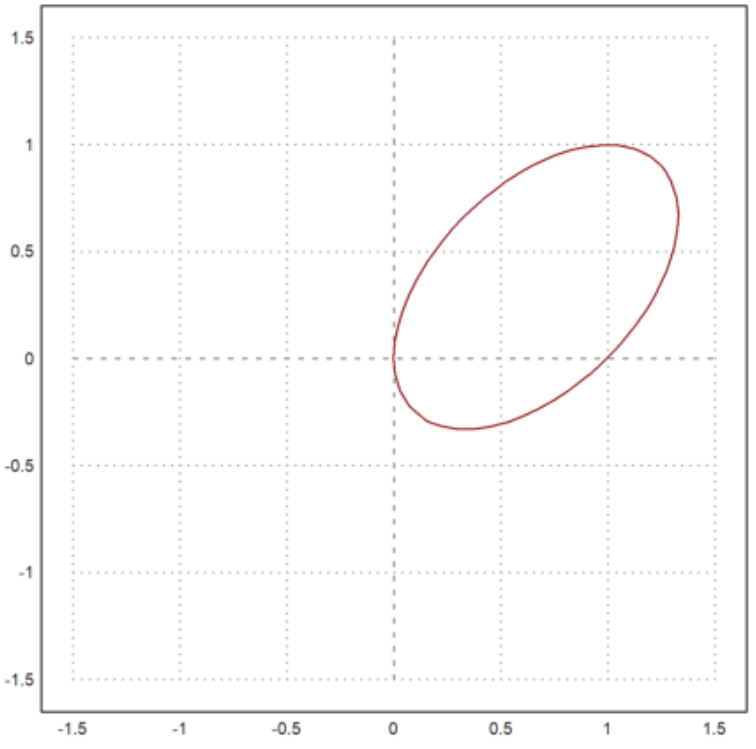
Euler can mark the level lines

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

of any function.

To draw the set f(x,y)=c for one or more constants c you can use plot2d() with its implicit plots in the plane. The parameter for c is level=c, where c can be vector of level lines. Additionally, a color scheme can be drawn in the background to indicate the value of the function for each point in the plot. The parameter "n" determines the fineness of the plot.

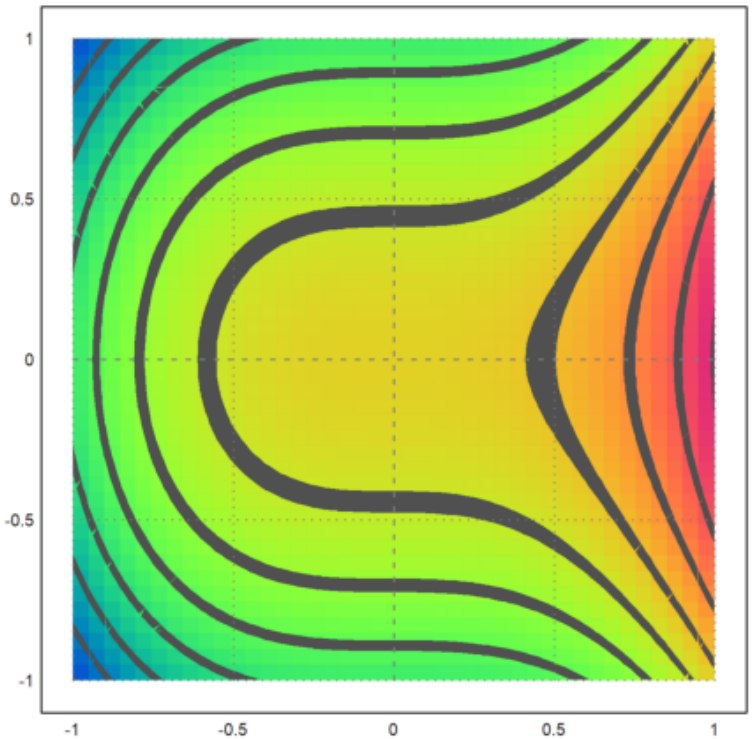
```
>aspect(1.5);
>plot2d("x^2+y^2-x*y-x",r=1.5,level=0,contourcolor=red):
```



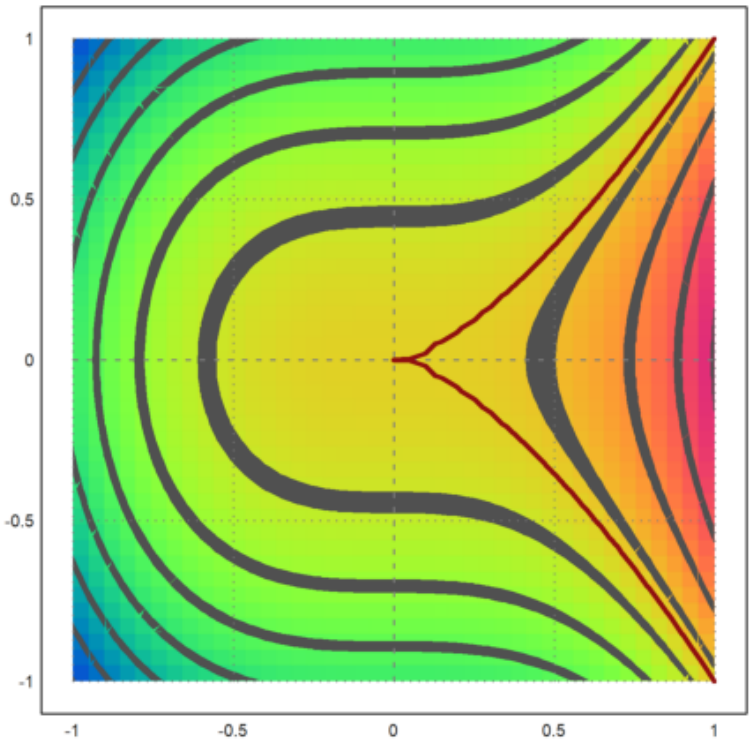

```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr,level=0): // Solutions of f(x,y)=0
>plot2d(expr,level=0:0.5:20,>hue,contourcolor=white,n=200): // nice
>plot2d(expr,level=0:0.5:20,>hue,>spectral,n=200,grid=4): // nicer
```

This works for data plots too. But you will have to specify the ranges for the axis labels.

```
>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):
```

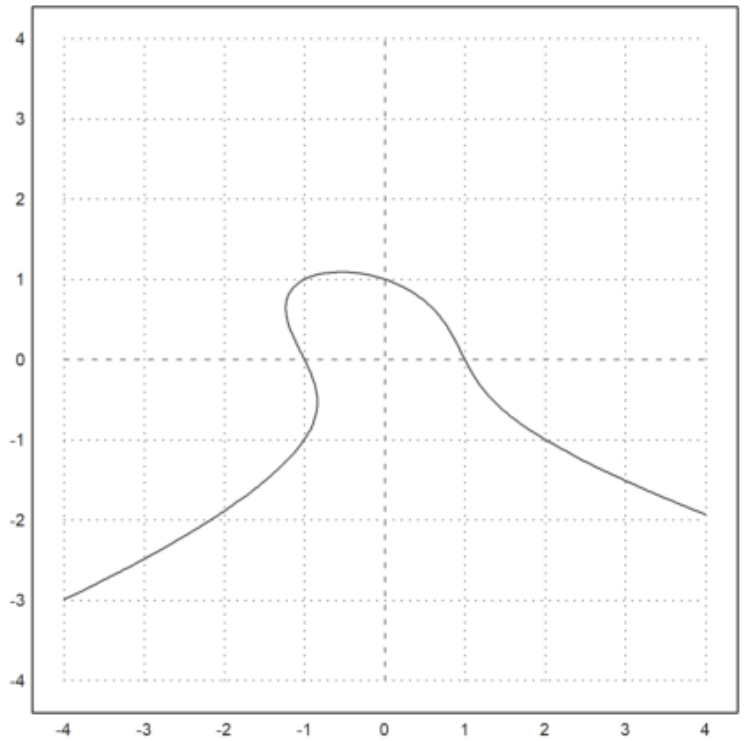


```
>plot2d("x^3-y^2",level=0,contourwidth=3,>add,contourcolor=red):
```

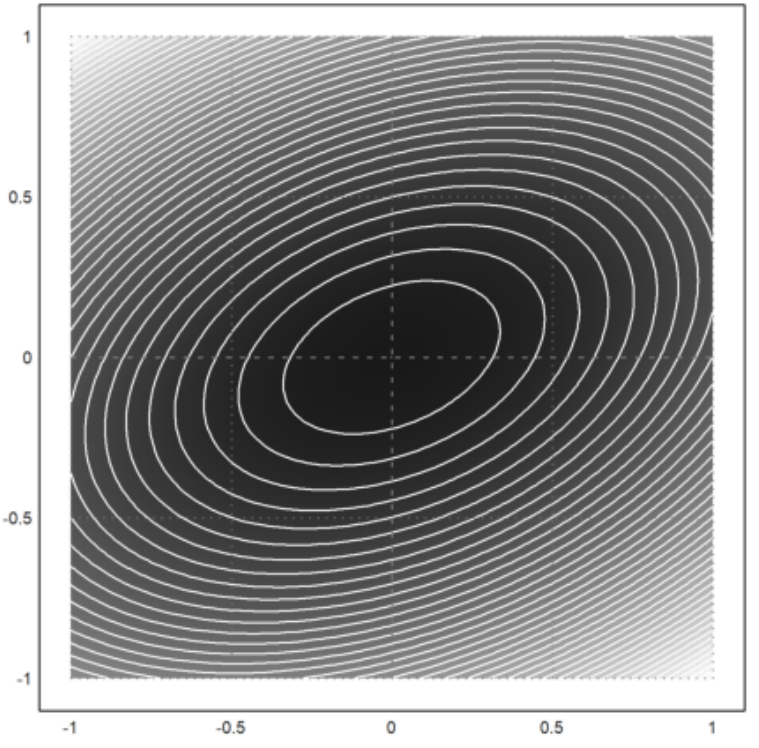



```
>z=z+normal(size(z))*0.2;
>plot2d(z,level=0.5,a=-1,b=2,c=-2,d=1):
```

```
>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):
```



It is also possible to fill the set

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

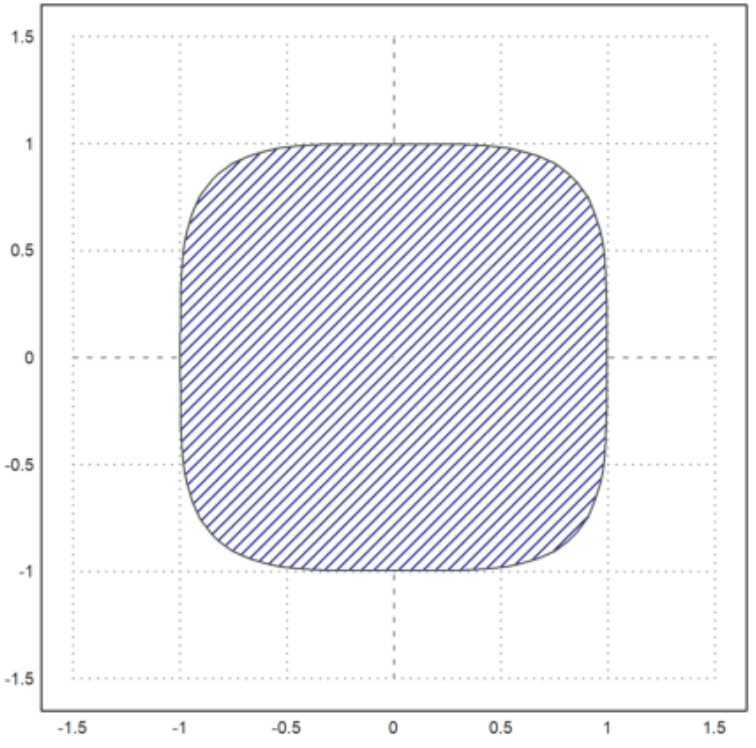
with a level range.

It is possible to fill regions of values for a specific function. For this, level must be a 2xn matrix. The first row are the lower bounds and the second row contains the upper bounds.

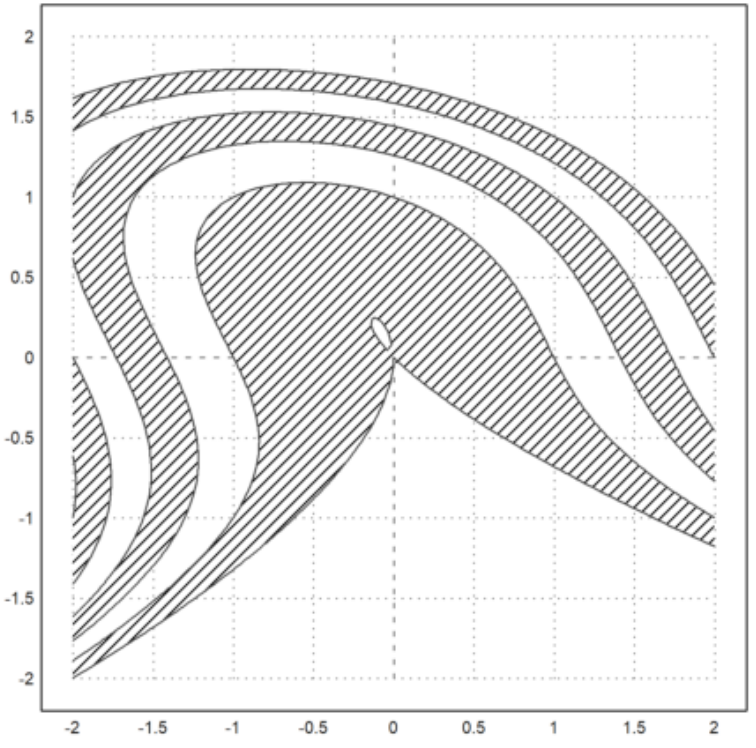
```
>plot2d(expr,level=[0;1],style="-",color=blue): // 0 <= f(x,y) <= 1
```

Implicit plots can also show ranges of levels. Then level must be a 2xn matrix of level intervals, where the first row contains the start and the second row the end of each interval. Alternatively, a simple row vector can be used for level, and a parameter dl extends the level values to intervals.

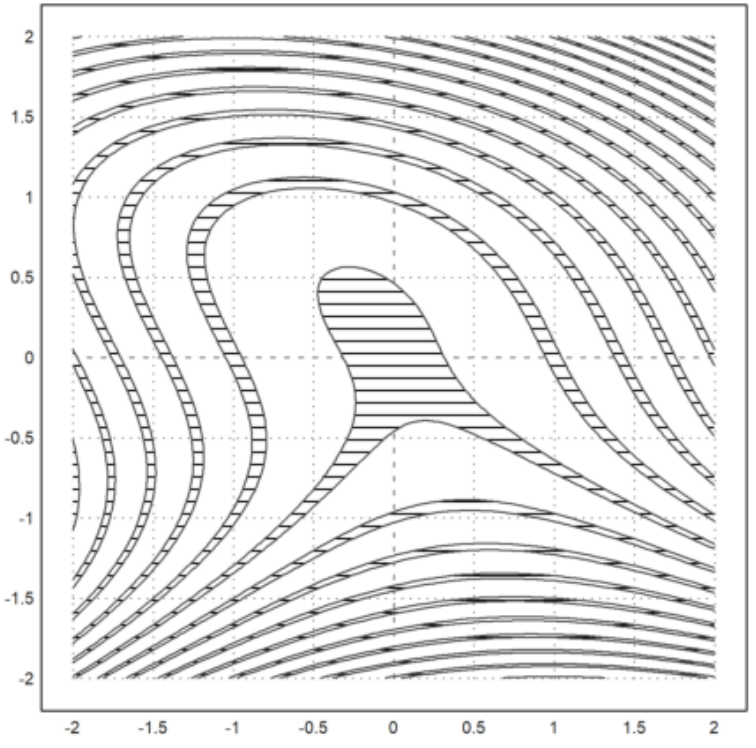
```
>plot2d("x^4+y^4",r=1.5,level=[0;1],color=blue,style="/"):
```



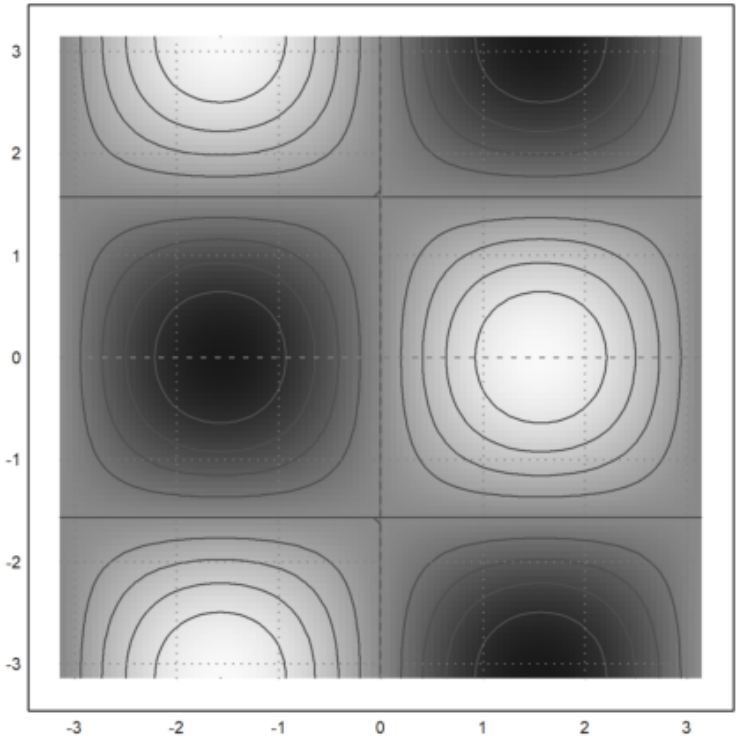

```
>plot2d("x^2+y^3+x*y",level=[0,2,4;1,3,5],style="/",r=2,n=100):
```




```
>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):
```

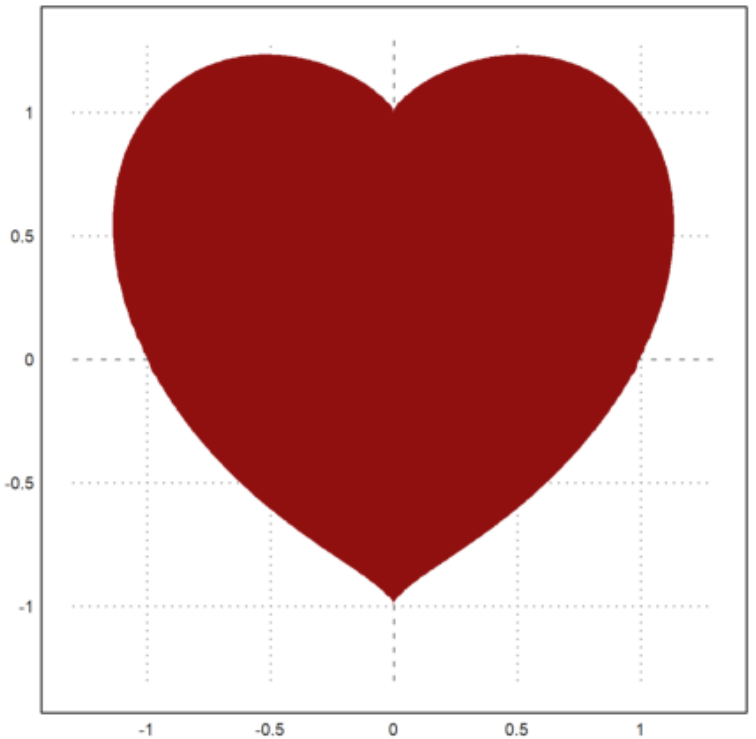


It is also possible to mark a region

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

This is done by adding a level with two rows.

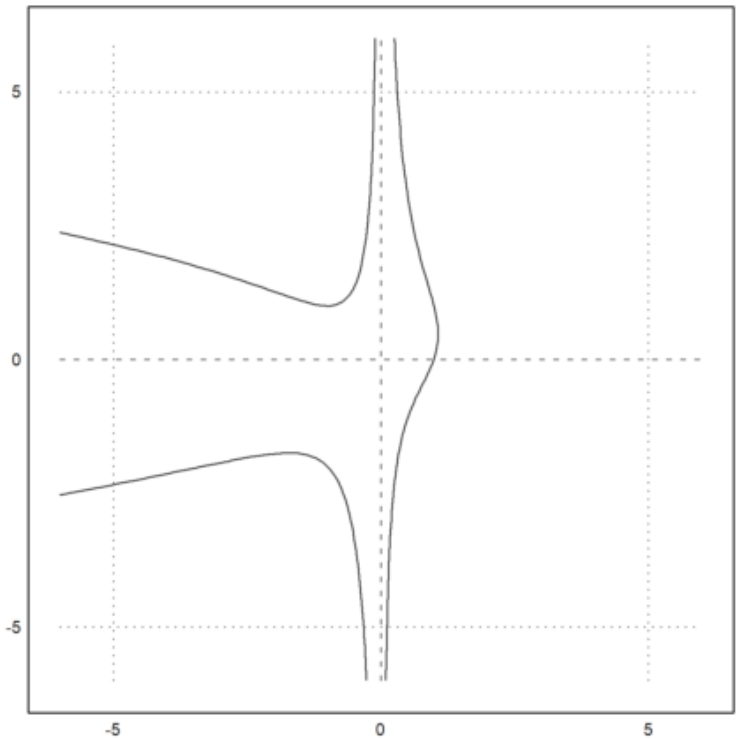
```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...
>  style="#",color=red,<outline, ...
>  level=[-2;0],n=100):
```



It is possible to specify a specific level. E.g., we can plot the solution of an equation like

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

```
>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);
    endif;
  end;
  barcolor(cl); barstyle(st);
  holding(h);
  window(w);
endfunction
```

There is no grid or axis ticks here. Moreover, we use the full window for the plot.

We call reset before we test this plot to restore the graphics defaults. This is not necessary, if you are sure that your plot works.

```
>reset; starplot1(normal(1,10)+5,color=red,lab=1:10):
```

Sometimes, you may want to plot something that plot2d cannot do, but almost.

In the following function, we do a logarithmic impulse plot. plot2d can do logarithmic plots, but not for impulse bars.

```
>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;
    holding(h);
  endfunction
```

Let us test it with exponentially distributed values.

```
>aspect(1.5); x=1:10; y=-log(random(size(x)))*200; ...
>logimpulseplot1(x,y):
```

Let us animate a 2D curve using direct plots. The plot(x,y) command simply plots a curve into the plot window. setplot(a,b,c,d) sets this window.

The wait(0) function forces the plot to appear on the graphics windows. Otherwise, the redraw takes place in sparse time intervals.

```
>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
```

Press any key to stop this animation.

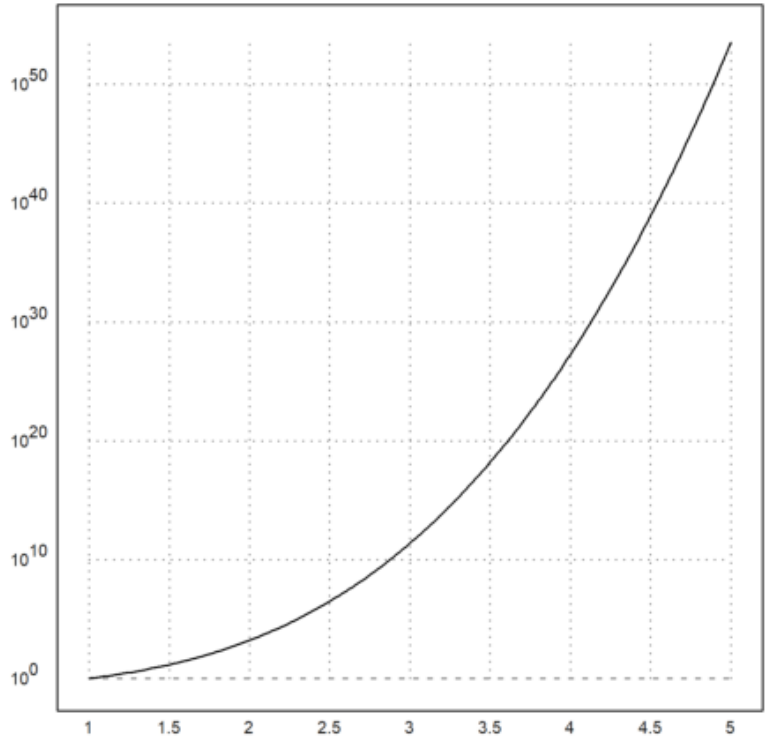
```
>animliss(2,3); // lihat hasilnya, jika sudah puas, tekan ENTER
```

Plot Logaritmik

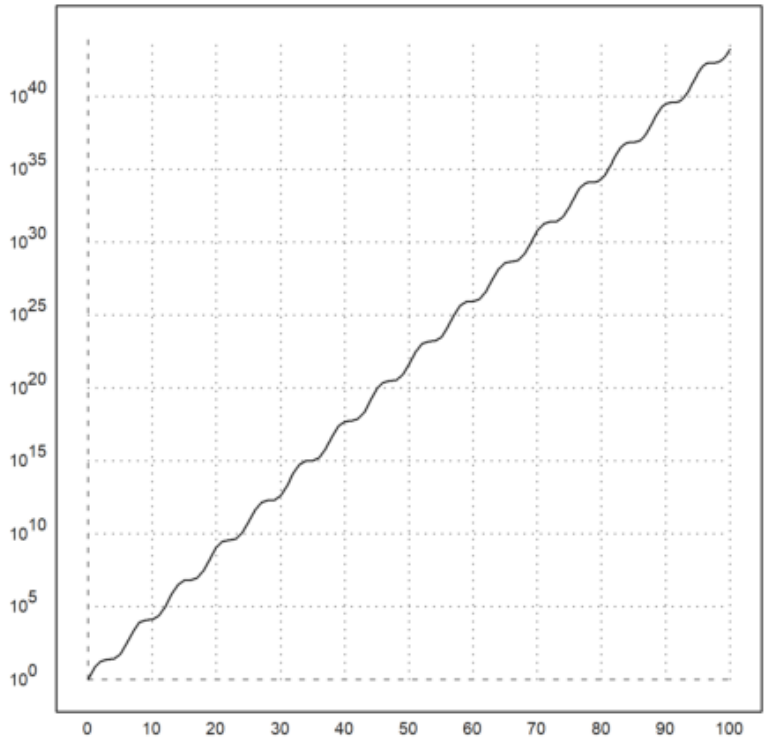
EMT menggunakan parameter ‘”logplot”‘ untuk skala logaritmik.
Plot logaritmik dapat dibuat dengan menggunakan skala logaritmik pada:

- sumbu-y dengan ‘logplot=1‘,
- sumbu-x dan y dengan ‘logplot=2‘, atau
- sumbu-x dengan ‘logplot=3‘.
- ‘logplot=1‘: skala logaritmik pada y
- ‘logplot=2‘: skala logaritmik pada x dan y
- ‘logplot=3‘: skala logaritmik pada x

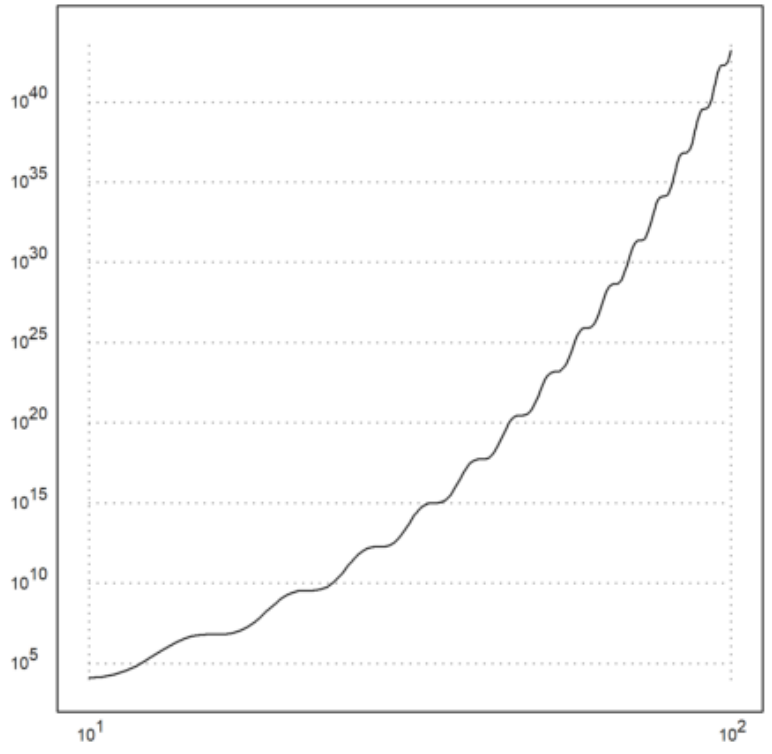
```
>plot2d("exp(x^3-x)*x^2",1,5,logplot=1):
```



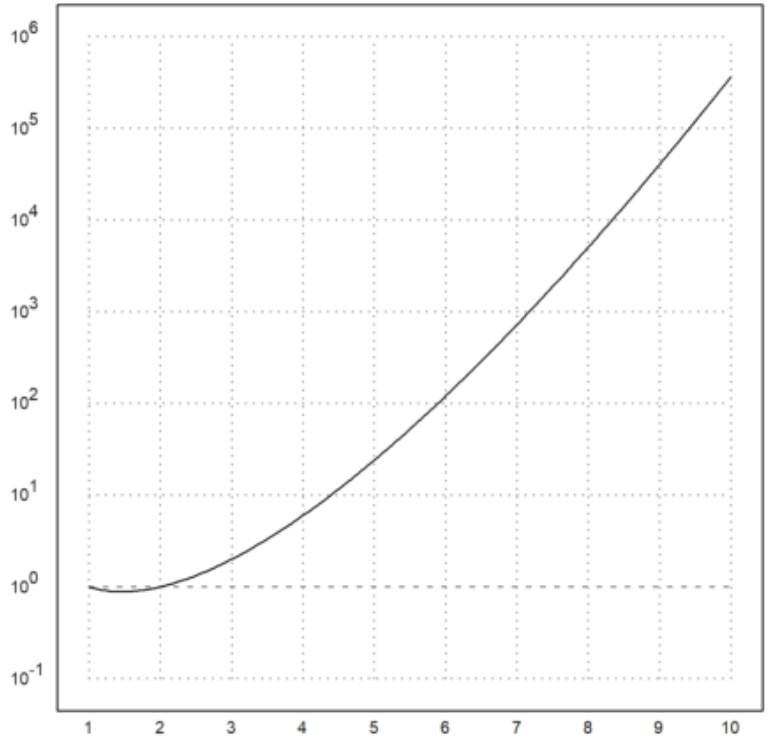

```
>plot2d("exp(x+sin(x))",0,100,logplot=1):
```



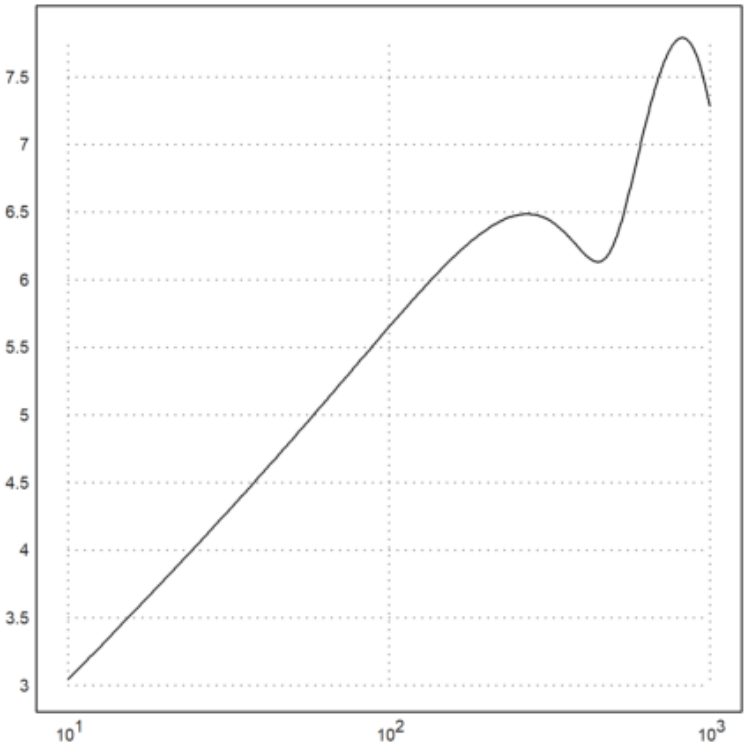

```
>plot2d("exp(x+sin(x))",10,100,logplot=2):
```




```
>plot2d("gamma(x)",1,10,logplot=1):
```




```
>plot2d("log(x*(2+sin(x/100)))",10,1000,logplot=3):
```



This does also work with data plots.

```
>x=10^(1:20); y=x^2-x;
>plot2d(x,y,logplot=2):
```

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
"#", "#0", "0", "/", "\", "\/",
"+", "|", "-", "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.