

TUGAS PEKAN 16 APLIKASI KOMPUTER

BAB LaTeX dan Markdown



Stevany Amelia Christable
22305144022
Matematika E 2022

**PRODI MATEMATIKA
DEPARTEMEN PENDIDIKAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI YOGYAKARTA
2023**

DAFTAR ISI

1	KB Pekan 3 : Menggunakan EMT untuk menyelesaikan masalah-masalah Aljabar	2
2	KB Pekan 4: Menggunakan EMT untuk menggambar grafik 2 dimensi (2D)	56
3	KB Pekan 5: Menggunakan EMT untuk menggambar grafik 3 dimensi (3D)	132
4	KB Pekan 6-7: Menggunakan EMT untuk kalkulus	170
5	KB Pekan 8: Menggunakan EMT untuk Geometri	214
6	KB Pekan 10; Menggunakan EMT untuk Statistika	275

BAB 1

KB PEKAN 3 : MENGGUNAKAN EMT UNTUK MENYELESAIKAN MASALAH-MASALAH ALJABAR

[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:

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

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

Variable j\$ not found!

Error in:

```
j$&6*x^(-3)*y^5*-7*x^2*y^(-9) ...  
^
```

Menjabarkan:

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

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

$$\text{expand}\left(\left(-\frac{1}{y^9} - 7x^2\right)\left(y^5 + \frac{6}{x^3}\right)\right) = -7x^2y^5 - \frac{1}{y^4} - \frac{6}{x^3y^9} - \frac{42}{x}$$

Baris perintah Euler terdiri dari satu atau beberapa perintah Euler 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 format.

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

16.7551608191

Perintah harus dipisahkan dengan yang kosong. Baris perintah berikut mencetak dua hasilnya.

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

50.2654824574
100.530964915

Baris perintah dieksekusi dalam urutan yang ditekan pengguna kembali. Jadi Anda mendapatkan nilai baru setiap kali Anda menjalankan baris kedua.

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

0.540302305868

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

0.857553215846

Jika dua garis terhubung dengan "..." kedua garis akan selalu dieksekusi 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 yang baik untuk menyebarkan perintah panjang pada dua atau lebih baris. Anda dapat menekan Ctrl+Return untuk membagi garis menjadi dua pada posisi kursor saat ini, atau Ctrl+Back untuk menggabungkan garis.

Untuk melipat semua multi-garis tekan Ctrl + L. Kemudian garis-garis berikutnya hanya akan terlihat, jika salah satunya memiliki fokus. Untuk melipat satu multi-baris, mulailah baris pertama dengan "%+".

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

Garis yang dimulai dengan %% tidak akan terlihat sama sekali.

81

Euler mendukung loop di baris perintah, selama mereka masuk ke dalam satu baris atau multi-baris. Dalam program, pembatasan ini tidak berlaku, tentu saja. Untuk informasi lebih lanjut, lihat pengantar berikut.

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

```
1.5  
1.4166666667  
1.41421568627  
1.41421356237  
1.41421356237
```

Tidak apa-apa untuk menggunakan multi-line. Pastikan baris diakhiri 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 Anda 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 ke bagian komentar di atas perintah untuk menuju ke perintah.

Saat Anda menggerakkan kursor di sepanjang garis, pasangan tanda kurung atau kurung buka dan tutup akan disorot. Juga, perhatikan baris status. Setelah kurung buka fungsi sqrt(), baris status akan menampilkan teks bantuan untuk fungsi tersebut. Jalankan perintah dengan tombol kembali.

```
>sqrt(sin(10°)/cos(20°))
```

```
0.429875017772
```

Untuk melihat bantuan untuk perintah terbaru, 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 untuk menutup jendela bantuan.

Anda dapat mengklik dua kali pada perintah apa pun untuk membuka bantuan untuk perintah ini. Coba klik dua kali perintah exp di bawah ini di baris perintah.

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

2.5

Anda dapat menyalin dan menempel di Euler juga. Gunakan Ctrl-C dan Ctrl-V untuk ini. Untuk menandai teks, seret mouse atau gunakan shift bersama dengan tombol kursor apa pun. Selain itu, Anda dapat menyalin tanda kurung yang disorot.

Basic Syntax

Euler tahu 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 rad(x). Fungsi akar kuadrat disebut kuadrat dalam Euler. Tentu saja, $x^{(1/2)}$ juga dimungkinkan.

Untuk menyetel variabel, gunakan "=" atau ":=". Demi kejelasan, pengantar ini menggunakan bentuk yang terakhir. Spasi tidak masalah. Tapi ruang antara perintah diharapkan.

Beberapa perintah dalam satu baris dipisahkan dengan "," atau ";". Titik koma menekan output dari 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. Memasuki

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

Anda harus mengatur tanda kurung yang benar dan menggunakan / untuk pecahan. Perhatikan tanda kurung yang disorot untuk 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

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

Anda harus memasukkannya dalam bentuk baris.

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

23.2671801626

Letakkan tanda kurung dengan hati-hati di sekitar sub-ekspresi yang perlu dihitung terlebih dahulu. EMT membantu Anda dengan menyorot ekspresi bahwa braket penutup selesai. Anda juga harus memasukkan nama "pi" untuk huruf Yunani pi.

Hasil dari perhitungan ini adalah bilangan floating point. Secara default dicetak dengan akurasi sekitar 12 digit. Di baris perintah berikut, kita juga belajar 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 dibuat dari operator dan fungsi. Jika perlu, itu harus mengandung tanda kurung untuk memaksa urutan eksekusi yang benar. Jika ragu, memasang braket adalah ide yang bagus. Perhatikan bahwa EMT menunjukkan tanda kurung buka dan tutup saat mengedit baris perintah.

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

```
14.4978445072
```

Operator numerik Euler meliputi

```
+ unary or operator plus  
- unary or operator minus  
*, /  
. the matrix product  
a^b power for positive a or integer b (a**b works too)  
n! the factorial operator
```

dan masih banyak lagi.

Berikut adalah beberapa fungsi yang mungkin Anda butuhkan. Ada 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 for log.

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

```
2  
0.5
```

```
>sin(30°)
```

```
0.5
```

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

```
>2^3^4, (2^3)^4, 2^(3^4)
```

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

Real Numbers

Tipe data utama dalam Euler adalah bilangan real. Real direpresentasikan dalam format IEEE dengan akurasi sekitar 16 digit desimal.

```
>longest 1/3
```

```
0.3333333333333333
```

Representasi ganda internal membutuhkan 8 byte.

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

```
1.01010101010101010101010101010101010101010101010101010101010101*2^-2
```

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

```
5.555555555554*16^-1
```

Strings

Sebuah string dalam 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 dikonversi menjadi string dalam kasus itu.

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

```
The area of the circle with radius 2 cm is 12.5663706144 cm^2.
```

Fungsi print juga mengonversi angka menjadi string. Ini dapat mengambil sejumlah digit dan sejumlah tempat (0 untuk keluaran padat), dan secara optimal satu unit.

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

Golden Ratio : 1.61803

Ada string khusus tidak ada, yang tidak dicetak. Itu dikembalikan oleh beberapa fungsi, ketika hasilnya tidak masalah. (Ini dikembalikan secara otomatis, jika fungsi tidak memiliki pernyataan pengembalian.)

```
>none
```

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 [none]. 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

Dalam komentar, entitas yang sama seperti , dll dapat digunakan. Ini mungkin alternatif cepat untuk Lateks. (Lebih detail 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" ;hnliches"
```

 hnliches

Boolean Values

Nilai Boolean direpresentasikan dengan 1=true atau 0=false dalam Euler. String dapat dibandingkan, seperti halnya angka.

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

```
0  
1
```

"dan" adalah operator "&&" dan "atau" adalah operator "||", seperti dalam bahasa C. (Kata-kata "dan" dan "atau" hanya dapat digunakan dalam kondisi untuk "jika".)

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

```
1
```

Operator Boolean mematuhi aturan bahasa matriks.

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

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

Anda dapat menggunakan fungsi bukan nol() untuk mengekstrak elemen tertentu dari vektor. Dalam contoh, kami menggunakan isprima bersyarat(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]
```

Output Formats

Format output default EMT mencetak 12 digit. Untuk memastikan bahwa kami melihat default, kami mengatur ulang format.

```
>defformat; pi
```

```
3.14159265359
```

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

```
>longest pi
```

```
3.141592653589793
```

Berikut adalah representasi heksadesimal internal dari bilangan ganda.

```
>printhex(pi)
```

```
3.243F6A8885A30*16^0
```

Format output dapat diubah secara permanen dengan perintah format.

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

```
0.33333
3.14159
0.84147
```

Standarnya adalah format (12).

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

```
0.333333333333
```

Fungsi seperti "shortestformat", "shortformat", "longformat" bekerja 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
```

Format default untuk skalar adalah format (12). Tapi ini bisa diubah.

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

```
3.1416
```

Fungsi "format terpanjang" mengatur format skalar juga.

```
>longestformat; pi
```

```
3.141592653589793
```

Untuk referensi, berikut adalah daftar format output yang paling penting.

```
format terpendek format pendek format panjang, format terpanjang  
format(panjang,digit) format baik(panjang)  
fracformat (panjang)  
mengubah bentuk
```

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

Tetapi format output EMT dapat diatur dengan cara yang fleksibel.

```
>longestformat; pi,
```

```
3.141592653589793
```

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

```
3.14159
```

Standarnya adalah deformat().

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

Ada operator pendek yang hanya mencetak satu nilai. Operator "terpanjang" akan mencetak semua digit angka yang valid.

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

```
4.934802200544679
```

Ada juga operator pendek 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 dengan tepat. Kesalahan bertambah sedikit, seperti yang Anda lihat dalam 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 melihat ini. Untuk kenyamanan, output dari angka yang sangat kecil adalah 0.

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

0

Expressions

String atau nama dapat digunakan untuk menyimpan ekspresi matematika, yang dapat dievaluasi oleh EMT. Untuk ini, gunakan tanda kurung setelah ekspresi. Jika Anda bermaksud 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()
```

12.56637061435917

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, bahkan jika 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" daripada 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

Untuk referensi, kami berkomentar 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 fungsi.

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

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

12

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

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

Bentuk khusus dari ekspresi memungkinkan variabel apa pun sebagai parameter tanpa nama untuk evaluasi ekspresi, bukan hanya "x", "y" dll. Untuk ini, mulai ekspresi dengan "@(variabel) ...".

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

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

41

Ini memungkinkan untuk memanipulasi ekspresi dalam variabel lain untuk fungsi EMT yang membutuhkan ekspresi dalam "x".

Cara paling dasar untuk mendefinisikan fungsi sederhana adalah dengan menyimpan rumusnya dalam ekspresi simbolis atau numerik. Jika variabel utama adalah x, ekspresi dapat dievaluasi seperti fungsi.

Seperi yang Anda lihat dalam contoh berikut, variabel global terlihat selama evaluasi.

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

-0.475

Semua variabel lain dalam ekspresi dapat ditentukan dalam evaluasi menggunakan parameter yang ditetapkan.

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

-0.425

Sebuah ekspresi tidak perlu simbolis. Ini diperlukan, jika ekspresi berisi fungsi, yang hanya diketahui di kernel numerik, bukan di Maxima.

Symbolic math

EMT melakukan matematika simbolis dengan bantuan Maxima. Untuk detailnya, mulailah dengan tutorial berikut, atau telusuri referensi untuk Maxima. Para ahli di Maxima harus mencatat bahwa ada perbedaan sintaks antara sintaks asli Maxima dan sintaks default ekspresi simbolik di EMT.

Matematika simbolik terintegrasi dengan mulus ke dalam Euler dengan &. Ekspresi apa pun yang dimulai dengan & adalah ekspresi simbolis. Itu dievaluasi dan dicetak oleh Maxima.

Pertama-tama, Maxima memiliki aritmatika "tak terbatas" yang dapat menangani angka yang sangat besar.

```
>$&44!
```

Dengan cara ini, Anda dapat menghitung hasil yang besar dengan tepat. Mari kita hitung

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

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

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

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

To learn more about a specific function double click on it. E.g., try double clicking on "&binomial" in the previous command line. This opens the documentation of Maxima as provided by the authors of that program. You will learn that the following works too.

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

Anda akan belajar bahwa yang berikut ini juga berfungsi.

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

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

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

Jika Anda ingin mengganti x dengan nilai tertentu, gunakan "dengan".

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

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.

Dengan begitu Anda dapat menggunakan solusi persamaan dalam persamaan lain.

Ekspresi simbolik dicetak oleh Maxima dalam bentuk 2D. Alasan untuk ini adalah bendera simbolis khusus dalam string.

Seperti yang akan Anda lihat pada contoh sebelumnya dan berikut, jika Anda telah menginstal LaTeX, Anda dapat mencetak ekspresi simbolis dengan Lateks. Jika tidak, perintah berikut akan mengeluarkan pesan kesalahan.

Untuk mencetak ekspresi simbolis dengan LaTeX, gunakan \$ di depan & (atau Anda dapat menghilangkan &) sebelum perintah. Jangan menjalankan perintah Maxima dengan \$, jika Anda tidak menginstal LaTeX.

```
>$ (3+x) / (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.

Ekspresi simbolik diuraikan oleh Euler. Jika Anda membutuhkan sintaks yang kompleks dalam satu ekspresi, Anda dapat menyertakan ekspresi dalam "...". Untuk menggunakan lebih dari ekspresi sederhana adalah mungkin, tetapi sangat tidak disarankan.

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

25

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.

Untuk kelengkapan, kami menyatakan bahwa ekspresi simbolik dapat digunakan dalam program, tetapi perlu diapit dalam tanda kutip. Selain itu, jauh lebih efektif untuk memanggil Maxima pada waktu kompilasi jika memungkinkan.

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

Sekali lagi, % mengacu pada hasil sebelumnya.

Untuk mempermudah, kami menyimpan solusi ke variabel simbolik. Variabel simbolik didefinisikan dengan "&=".

Sekali lagi, % mengacu pada hasil sebelumnya.

Untuk mempermudah, kami menyimpan solusi ke variabel simbolik. Variabel simbolik didefinisikan dengan "&=".

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

Symbolic expressions can be used in other symbolic expressions.
Ekspresi simbolik dapat digunakan dalam ekspresi simbolik lainnya.

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

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").

Masukan langsung dari perintah Maxima juga tersedia. Mulai baris perintah dengan "::". Sintaks Maxima disesuaikan dengan sintaks EMT (disebut "mode kompatibilitas").

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

2432902008176640000

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

8 4 2
2 3 5 7

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

18 8 4 2
2 3 5 7 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 "::". Jika Anda ahli dalam Maxima, Anda mungkin ingin menggunakan sintaks asli Maxima. Anda dapat melakukannya dengan "::".

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

2
g

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

3 x
x E

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.

Variabel tersebut dapat digunakan dalam ekspresi simbolik lainnya. Perhatikan, bahwa dalam perintah berikut sisi kanan &= dievaluasi sebelum penugasan ke Fx.

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

```
5  
125 E
```

```
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().

Untuk evaluasi ekspresi dengan nilai variabel tertentu, Anda dapat menggunakan operator "with". Baris perintah berikut juga menunjukkan bahwa Maxima dapat mengevaluasi ekspresi secara numerik dengan float().

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

```
10  
1000 E - 125 E
```

```
2.20079141499189e+7
```

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

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

Untuk mendapatkan kode Lateks untuk ekspresi, Anda dapat menggunakan perintah tex.

```
>tex(fx)
```

```
x^3\, e^{x }
```

Symbolic expressions can be evaluated just like numerical expressions.

Ekspresi simbolik dapat dievaluasi seperti ekspresi numerik.

```
>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).

Dalam ekspresi simbolis, ini tidak berfungsi, karena Maxima tidak mendukungnya. Sebagai gantinya, gunakan sintaks "with" (bentuk yang lebih bagus dari perintah at(...) dari Maxima).

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

Penugasan juga bisa bersifat simbolis.

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

The command solve solves symbolic expressions for a variable in Maxima. The result is a vector of solutions. Perintah solve memecahkan ekspresi simbolik untuk variabel di Maxima. Hasilnya adalah vektor solusi.

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

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

Bandingkan dengan perintah numerik "selesaikan" di Euler, yang membutuhkan nilai awal, dan secara opsional nilai target.

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

1.56155281281

The numerical values of the symbolic solution can be computed by evaluation of the symbolic result. Euler will read over the assignments x= etc. If you do not need the numerical results for further computations you can also let Maxima find the numerical values.

Nilai numerik dari solusi simbolik dapat dihitung dengan evaluasi hasil simbolis. Euler akan membaca tugas x= dll. Jika Anda tidak memerlukan hasil numerik untuk perhitungan lebih lanjut, Anda juga dapat memberikan Maxima menemukan nilai numerik.

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

[-3.23607, 1.23607]

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

Untuk mendapatkan solusi simbolis tertentu, seseorang dapat menggunakan "dengan" dan indeks.

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

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

Untuk menyelesaikan sistem persamaan, gunakan vektor persamaan. Hasilnya adalah vektor solusi.

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

Symbolic expressions can have flags, which indicate a special treatment in Maxima. Some flags can be used as commands too, others can't. Flags are appended with "!" (a nicer form of "ev(...,flags)")

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

```
>$& diff((x^3-1)/(x+1),x) //turunan bentuk pecahan  
>$& diff((x^3-1)/(x+1),x) ! ratsimp //menyederhanakan pecahan  
>$&factor(%)
```

Functions

In EMT, functions are programs defined with the command "function". It can be a one-line function or multiline function.

A one-line function can be numerical or symbolic. A numerical one-line function is defined by ":=".

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

Fungsi satu baris dapat berupa numerik atau simbolis. 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.

Untuk gambaran umum, kami menunjukkan semua kemungkinan definisi untuk fungsi satu baris. Suatu fungsi dapat dievaluasi sama seperti fungsi Euler bawaan lainnya.

```
>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.

Fungsi ini akan bekerja untuk vektor juga, dengan mematuhi bahasa matriks Euler, karena ekspresi yang digunakan dalam fungsi divektorkan.

```
>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.

Fungsi dapat diplot. Alih-alih ekspresi, kita hanya perlu memberikan nama fungsi.

Berbeda dengan ekspresi simbolik atau numerik, nama fungsi harus diberikan dalam 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.

Secara default, jika Anda perlu menimpa fungsi bawaan, Anda harus menambahkan kata kunci "menimpa". Menimpa fungsi bawaan berbahaya dan dapat menyebabkan masalah untuk fungsi lain tergantung pada fungsi tersebut.

Anda masih dapat memanggil fungsi bawaan sebagai "_...", jika itu adalah fungsi di inti Euler.

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

0.707106781187

We better remove this redefinition of sin.

Lebih baik kita menghapus redefinisi dosa ini.

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

0.707106781187

Default Parameters

Numerical function can have default parameters.

Fungsi numerik dapat memiliki parameter default.

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

Omitting this parameter uses the default value.

Menghilangkan parameter ini menggunakan nilai default.

```
>f(4)
```

16

Setting it overwrites the default value.

Menyetelnya akan menimpa nilai default.

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

80

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

Parameter yang ditetapkan menimpanya juga. Ini digunakan oleh banyak fungsi Euler seperti 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.

Jika suatu variabel bukan parameter, itu harus global. Fungsi satu baris dapat melihat variabel global.

```
>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 ":="!

Tetapi parameter yang ditetapkan menimpa nilai global.

Jika argumen tidak ada dalam daftar parameter yang telah ditentukan sebelumnya, argumen tersebut harus dideklarasikan dengan ":="!

```
>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.

Fungsi simbolik didefinisikan dengan "&=". Mereka didefinisikan dalam Euler dan Maxima, dan bekerja di kedua dunia. Ekspresi yang mendefinisikan dijalankan melalui Maxima sebelum definisi.

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

Symbolic functions can be used in symbolic expressions.

Fungsi simbolik dapat digunakan dalam ekspresi simbolik.

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

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

Mereka juga dapat digunakan dalam ekspresi numerik. Tentu saja, ini hanya akan berfungsi jika EMT dapat menginterpretasikan semua yang ada di dalam fungsi tersebut.

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

178.635099908

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

Mereka dapat digunakan untuk mendefinisikan fungsi atau ekspresi simbolis lainnya.

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

Berikut ini juga berfungsi, karena Euler menggunakan ekspresi simbolis dalam fungsi g, jika tidak menemukan variabel simbolik g, dan jika ada fungsi simbolis g.

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

0.703467422498

```
>function P(x,n) &= (2*x-1)^n; $&P(x,n)
>function Q(x,n) &= (x+2)^n; $&Q(x,n)
>$&P(x,4), $&expand(%)
>P(3,4)
```

625

```
>$&P(x,4)+Q(x,3), $&expand(%)
>$&P(x,4)-Q(x,3), $&expand(%), $&factor(%)
>$&P(x,4)*Q(x,3), $&expand(%), $&factor(%)
>$&P(x,4)/Q(x,1), $&expand(%), $&factor(%)
>function f(x) &= x^3-x; $&f(x)
```

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

Dengan &= fungsinya adalah simbolis, dan dapat digunakan dalam simbol lainnya ekspres

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

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

Dengan := fungsinya numerik. Contoh yang baik adalah integral tak tentu seperti

$$f(x) = \int_1^x t^t dt,$$

which can not be evaluated symbolically.

yang tidak dapat dinilai secara simbolis.

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.

Jika kita mendefinisikan kembali fungsi dengan kata kunci "peta" dapat digunakan untuk vektor x. Secara internal, fungsi dipanggil untuk semua nilai x satu kali, dan hasilnya disimpan dalam vektor.

```
>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.

Fungsi dapat memiliki nilai default untuk parameter.

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

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

Sekarang fungsi dapat dipanggil dengan atau tanpa parameter "basis".

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

```
2  
6.7
```

Moreover, it is possible to use assigned parameters.

Selain itu, dimungkinkan untuk menggunakan parameter yang ditetapkan.

```
>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.

Seringkali, kita ingin menggunakan fungsi untuk vektor di satu tempat, dan untuk individu elemen di tempat lain. Ini dimungkinkan dengan parameter vektor.

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

Such a symbolic function can be used for symbolic variables.

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

Fungsi simbolik seperti itu dapat digunakan untuk variabel simbolik.

Tetapi fungsi tersebut juga dapat digunakan untuk vektor numerik.

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

```
17
```

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

Ada juga fungsi simbolis murni, yang tidak dapat digunakan secara numerik.

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

```
diff(expr, y, 2) + diff(expr, x, 2)
```

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

But of course, they can be used in symbolic expressions or in the definition of symbolic functions.

Tetapi 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)
```

Untuk meringkas

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

Solving Expressions

Expressions can be solved numerically and symbolically.

To solve a simple expression of one variable, we can use the solve() function. It needs a start value to start the search. Internally, solve() uses the secant method.

Ekspresi dapat diselesaikan secara numerik dan simbolis.

Untuk menyelesaikan ekspresi sederhana dari satu variabel, kita dapat menggunakan fungsi solve(). Perlu 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.

Ini juga berfungsi untuk ekspresi simbolis. Ambil fungsi berikut.

```
>$&solve(x^2=2,x)
>$&solve(x^2-2,x)
>$&solve(a*x^2+b*x+c=0,x)
>$&solve([a*x+b*y=c,d*x+e*y=f],[x,y])
>px &= 4*x^8+x^7-x^4-x; $&px
```

Now we search the point, where the polynomial is 2. In solve(), the default target value $y=0$ can be changed with an assigned variable.

We use $y=2$ and check by evaluating the polynomial at the previous result.

Sekarang kita cari titiknya, di mana polinomialnya adalah 2. Dalam solve(), defaultnya nilai target $y=0$ 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.

Memecahkan ekspresi simbolis dalam bentuk simbolis mengembalikan daftar solusi. Kami menggunakan pemecah simbolik solve() yang disediakan oleh Maxima.

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

The easiest way to get the numerical values is to evaluate the solution numerically just like an expression. Cara termudah untuk mendapatkan nilai numerik adalah dengan mengevaluasi solusi secara numerik seperti ekspresi.

```
>longest sol()
```

```
-0.6180339887498949 1.618033988749895
```

Untuk menggunakan solusi secara simbolis dalam ekspresi lain, cara termudah adalah "with".

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

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.

Memecahkan sistem persamaan secara simbolis dapat dilakukan dengan vektor persamaan dan solver simbolis solve(). Jawabannya adalah daftar daftar persamaan.

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

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

Fungsi f() dapat melihat variabel global. Namun seringkali kita ingin menggunakan parameter lokal.

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

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).

Salah satu cara untuk meneruskan parameter tambahan ke f() adalah dengan menggunakan daftar dengan nama fungsi dan parameter (sebaliknya adalah parameter titik koma).

```
>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).

Ini juga bekerja dengan ekspresi. Tapi kemudian, elemen daftar bernama harus digunakan. (Lebih lanjut tentang daftar di tutorial tentang sintaks 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
```

```
>$&fourier_elim([x^2 - 1>0],[x]) // x^2-1 > 0
>$&fourier_elim([x^2 - 1<0],[x]) // x^2-1 < 0
>$&fourier_elim([x^2 - 1 # 0],[x]) // x^-1 <> 0
>$&fourier_elim([x # 6],[x])
>$&fourier_elim([x < 1, x > 1],[x]) // tidak memiliki penyelesaian
>$&fourier_elim([minf < x, x < inf],[x]) // solusinya R
>$&fourier_elim([x^3 - 1 > 0],[x])
>$&fourier_elim([cos(x) < 1/2],[x]) // ??? gagal
>$&fourier_elim([y-x < 5, x - y < 7, 10 < y],[x,y]) // sistem pertidaksamaan
>$&fourier_elim([y-x < 5, x - y < 7, 10 < y],[y,x])
>$&fourier_elim((x + y < 5) and (x - y >8),[x,y])
>$&fourier_elim(((x + y < 5) and x < 1) or (x - y >8),[x,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])
```

The Matrix Language

Dokumentasi EMT core berisi diskusi rinci pada bahasa matriks Euler.

Vektor dan matrices dimasukkan dengan tanda kurung persegi, elemen dipisahkan oleh koma, baris dipisahkan oleh semikum.

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

1	2
3	4

Produk 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 bahwa semua fungsi dan operator bekerja elemen untuk 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
```

$$\begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array}$$

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

$$\begin{array}{cc} 0.333333 & 0.666667 \\ 0.75 & 1 \end{array}$$

```
>A\b // hasil kali invers A dan b, A^(-1)b
```

$$\begin{array}{c} -2 \\ 2.5 \end{array}$$

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

$$\begin{array}{c} -2 \\ 2.5 \end{array}$$

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

$$\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}$$

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

$$\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}$$

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

$$\begin{array}{cc} 1 & 4 \\ 9 & 16 \end{array}$$

Ini bukan produk matriks, tetapi perkalian elemen demi elemen. Hal yang sama berlaku untuk vektor.

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

$$\begin{array}{c} 9 \\ 16 \end{array}$$

Jika salah satu operan adalah vektor atau skalar, itu 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 itu adalah vektor baris, itu diterapkan ke semua kolom A.

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

2	6
6	12

Seseorang dapat membayangkan perkalian ini seolah-olah vektor baris v telah digandakan untuk membentuk matriks dengan ukuran yang 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

Ini juga berlaku untuk dua vektor di mana satu adalah vektor baris dan yang lainnya adalah vektor kolom. Kami menghitung i^*j untuk i,j dari 1 hingga 5. Caranya adalah dengan mengalikan 1:5 dengan transposnya. 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, ingat bahwa ini bukan produk matriks!

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

```
>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]

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. Kami mendapatkan vektor 0 dan 1, di mana 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]

Dari vektor seperti itu, "bukan nol" memilih elemen bukan nol.

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

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

[64, 81, 100]

Sebagai contoh, mari kita cari semua kuadrat dari angka 1 hingga 1000, yaitu 5 modulo 11 dan 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 tidak sepenuhnya efektif untuk perhitungan bilangan bulat. Ini menggunakan titik mengambang presisi ganda secara internal. Namun, seringkali sangat berguna.

Kita dapat memeriksa keutamaan. Mari kita cari tahu, berapa banyak kuadrat ditambah 1 adalah bilangan prima.

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

112

Fungsi bukan nol() hanya berfungsi untuk vektor. Untuk matriks, ada 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

Ini mengembalikan indeks elemen, yang bukan nol

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

1	4
2	1
2	2
3	2

Indeks ini dapat digunakan untuk mengatur elemen ke beberapa nilai.

```
>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

Fungsi mset() juga dapat mengatur elemen pada indeks ke entri dari beberapa matriks lainnya.

```
>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

Dan dimungkinkan untuk mendapatkan elemen dalam vektor.
ntri dari beberapa matriks lainnya.

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

[0.267829, 0.13673, 0.390567, 0.006085]

Fungsi lain yang berguna adalah ekstrem, yang mengembalikan nilai minimal dan maksimal di setiap baris matriks dan posisinya.

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

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

Kita dapat menggunakan ini untuk mengekstrak nilai maksimal di setiap baris.

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

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

Ini, tentu saja, sama dengan fungsi max().

```
max() .
```

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

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

Tetapi dengan mget(), kita dapat mengekstrak indeks dan menggunakan informasi ini untuk mengekstrak elemen pada posisi yang sama dari matriks lain.
at berguna.

Kita dapat memeriksa keutamaan. Mari kita cari tahu, berapa banyak kuadrat ditambah 1 adalah bilangan prima.

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

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

Other Matrix Functions (Building Matrix)

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

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

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

Demikian juga, kita dapat melampirkan matriks ke yang lain secara berdampingan, jika keduanya memiliki 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 mereka tidak memiliki jumlah baris yang sama, matriks yang lebih pendek diisi dengan 0.

Ada pengecualian untuk aturan ini. Bilangan real yang dilampirkan pada 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

Dimungkinkan untuk membuat matriks vektor baris dan kolom.

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

1	2	3
1	2	3

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

1	1
2	2
3	3

Tujuan utama dari ini adalah untuk menafsirkan vektor ekspresi untuk vektor kolom.

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

1	1
2	4
3	9

Untuk mendapatkan ukuran A, kita dapat menggunakan fungsi berikut.

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

2	
4	
[2,	4]
4	

Untuk vektor, ada panjang().

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

9

Ada banyak fungsi lain, yang menghasilkan matriks.

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

1	1
1	1

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

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

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

Juga matriks bilangan acak dapat dihasilkan dengan acak (distribusi seragam) atau normal (distribusi Gau).

vektor n kali.

```
>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 elemen 1, 2, 3, ..., 9 ke bentuk matriks 3x3
```

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

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

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

Mari kita uji.

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

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

Fungsi multdup() menduplikasi elemen vektor.

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

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

Fungsi flipx() dan flipy() mengembalikan urutan baris atau kolom matriks. Yaitu, fungsi flipx() membalik secara horizontal.

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

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

Untuk rotasi, Euler memiliki rotleft() dan rotright().

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

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

Sebuah fungsi khusus adalah `drop(v,i)`, yang menghilangkan elemen dengan indeks di `i` dari vektor `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 harus menemukan elemennya terlebih dahulu. Fungsi `indexof(v,x)` dapat digunakan untuk mencari elemen `x` dalam vektor terurut `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, 7, 0, 8, 0]
[2, 3, 5, 7, 23, 29, 31, 37, 41, 43, 47]
```

Seperti yang Anda lihat, tidak ada salahnya untuk 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]
```

Ada beberapa fungsi khusus untuk mengatur diagonal atau untuk menghasilkan matriks diagonal. Kita mulai dengan matriks identitas.

```
>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

Kemudian kita atur diagonal bawah (-1) menjadi 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

Perhatikan bahwa kami tidak mengubah matriks A. Kami mendapatkan matriks baru sebagai hasil dari setdiag().

Berikut adalah fungsi, yang mengembalikan matriks tri-diagonal.
ngsi indexof(v,x) dapat
digunakan untuk mencari elemen x dalam vektor terurut v.

```
>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

Diagonal suatu matriks juga dapat diekstraksi dari matriks tersebut. Untuk mendemonstrasikan ini, kami merestrukturisasi vektor 1:9 menjadi matriks 3x3.

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

1	2	3
4	5	6
7	8	9

Sekarang kita dapat mengekstrak diagonal.

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

[1, 5, 9]

Misalnya. Kita dapat membagi matriks dengan diagonalnya. Bahasa matriks memperhatikan bahwa vektor kolom diterapkan ke matriks baris demi baris.

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

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

Vectorization

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

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

[1, 1.41421, 1.73205]

Jadi Anda dapat dengan mudah membuat tabel nilai. Ini adalah salah satu cara untuk memplot suatu fungsi (alternatifnya menggunakan ekspresi).

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

Dengan ini dan operator titik dua a:delta:b, vektor nilai fungsi dapat dihasilkan dengan mudah. Pada contoh berikut, kita membangkitkan vektor nilai $t[i]$ dengan spasi 0,1 dari -1 hingga 1. Kemudian kita membangkitkan vektor nilai fungsi

$$s = t^3 - t$$

```
>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 dikalikan vektor baris menjadi matriks, jika operator diterapkan. Berikut ini, v' adalah vektor yang ditransposisikan (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
```

Perhatikan, bahwa ini sangat berbeda dari produk matriks. Produk matriks dilambangkan dengan titik $\cdot\cdot\cdot$ di EMT.

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

55

Secara default, vektor baris dicetak dalam format yang ringkas.

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

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

Untuk matriks operator khusus . menunjukkan perkalian matriks, dan A' menunjukkan transpos. Matriks 1x1 dapat digunakan seperti bilangan real.

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

```
5  
25
```

Untuk mentranspos matriks kita menggunakan apostrof.

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

```
1  
2  
3  
4
```

Jadi kita dapat menghitung matriks A kali vektor b.

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

```
30  
70
```

Perhatikan bahwa v masih merupakan vektor baris. Jadi $v' \cdot v$ berbeda dari $v \cdot v'$.

```
>v'.v
```

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

$v \cdot v'$ menghitung norma v kuadrat untuk vektor baris v. Hasilnya adalah vektor 1×1 , yang bekerja seperti bilangan real.

```
>v.v'
```

```
30
```

Ada juga fungsi norma (bersama dengan banyak fungsi lain dari Aljabar Linier).

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

```
30
```

Operator dan fungsi mematuhi bahasa matriks Euler.

Berikut ringkasan aturannya.

- Fungsi yang diterapkan ke vektor atau matriks diterapkan ke setiap elemen.
- Operator yang beroperasi pada dua matriks dengan ukuran yang sama diterapkan berpasangan ke elemen matriks.
- Jika kedua matriks memiliki dimensi yang berbeda, keduanya diperluas dengan cara yang masuk akal, sehingga memiliki ukuran yang sama.

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

Berikut ini adalah kasus sederhana dengan operator ^.

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

```
[1, 4, 9]
```

Berikut adalah kasus yang lebih rumit. Vektor baris dikalikan dengan vektor kolom mengembang keduanya dengan menduplikasi.

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

1	2	3
2	4	6
3	6	9

Perhatikan bahwa produk skalar menggunakan produk matriks, bukan *!

```
>v.v'
```

14

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

```
sum,prod computes the sum and products of the rows  
cumsum,cumprod does the same cumulatively  
computes the extremal values of each row  
extrema returns a vector with the extremal information  
diag(A,i) returns the i-th diagonal  
setdiag(A,i,v) sets the i-th diagonal  
id(n) the identity matrix  
det(A) the determinant  
charpoly(A) the characteristic polynomial  
eigenvalues(A) the eigenvalues
```

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

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

Operator : menghasilkan vektor baris spasi yang sama, opsional dengan ukuran langkah.

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

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

Untuk menggabungkan matriks dan vektor ada operator " | " dan " _ ".

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

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

Unsur-unsur matriks disebut dengan "A[i,j]".

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

6

Untuk vektor baris atau kolom, $v[i]$ adalah elemen ke- i dari vektor. Untuk matriks, ini mengembalikan baris ke- i lengkap dari matriks.
tang perintah ini.

```
sum,prod computes the sum and products of the rows  
cumsum,cumprod does the same cumulatively  
computes the extremal values of each row  
extrema returns a vector with the extremal information  
diag(A,i) returns the  $i$ -th diagonal  
setdiag(A,i,v) sets the  $i$ -th diagonal  
id(n) the identity matrix  
det(A) the determinant  
charpoly(A) the characteristic polynomial  
eigenvalues(A) the eigenvalues
```

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

6

[7, 8, 9]

Indeks juga bisa menjadi vektor baris dari indeks. : menunjukkan semua indeks.

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

[2, 4]

2
5
8

Bentuk singkat untuk : adalah menghilangkan indeks sepenuhnya

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

2	3
5	6
8	9

Untuk tujuan vektorisasi, elemen matriks dapat diakses seolah-olah mereka 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 untuk tabel, mari kita reset ke format default, dan menghitung tabel nilai sinus dan kosinus. Perhatikan bahwa sudut dalam radian secara default.

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

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

Sekarang kita menambahkan kolom ke matriks.

```
>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 to n . Kami mendapatkan matriks, di mana setiap baris adalah tabel t^i untuk satu i . I.e., the matrix has the elements latex: $a_{i,j} = t_j^i$, $\quad 1 \leq j \leq 101$, $1 \leq i \leq n$

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

Integrasi numerik terintegrasi() hanya berfungsi untuk batas interval skalar. Jadi kita perlu membuat vektor.

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

Kata kunci "peta" membuat vektor fungsi. Fungsinya sekarang akan bekerja untuk vektor bilangan

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

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

Sub-Matrices and Matrix-Elements

Untuk mengakses elemen matriks, gunakan notasi braket.

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

1	2	3
4	5	6
7	8	9
5		

Kita dapat mengakses satu baris matriks yang lengkap.

```
>A[2]
```

[4, 5, 6]

Dalam kasus vektor baris atau kolom, ini mengembalikan elemen vektor.

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

2

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

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

[4, 5, 6]

Jika indeks adalah vektor indeks, Euler akan mengembalikan baris matriks yang sesuai.

Di sini kita menginginkan baris pertama dan kedua dari A

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

1	2	3
4	5	6

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

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

7	8	9
4	5	6
1	2	3

Trik indeks bekerja dengan kolom juga.

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.

baris A dan kolom kedua dan ketiga

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

3
6
9

Atau, biarkan indeks pertama kosong.

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

2	3
5	6
8	9

Kita juga bisa mendapatkan baris terakhir dari 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 tersimpan.

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

4	2	3
4	5	6
7	8	9

Kita juga bisa mendapatkan baris terakhir dari A.

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

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

Kami bahkan dapat menetapkan sub-matriks jika memiliki ukuran yang tepat.

```
>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, tergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Ingat, bagaimanapun, 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] ...
^
```

Sorting and Shuffling

Fungsi sort() mengurutkan vektor baris.

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

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

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

Mari kita mengocok vektor.

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

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

Indeks berisi urutan yang tepat dari v.

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

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

Ini bekerja untuk vektor string juga.

```
>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.
andom.

```
>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 bekerja untuk vektor string juga.

```
>unique(s)
```

```
a  
aa  
d  
e
```

Linear Algebra

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

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

```
-4  
4.5
```

Untuk contoh lain, kami membuat matriks 200x200 dan jumlah 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 benar.

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

```
8.790745908981989e-13
```

Jika sistem tidak memiliki 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

Determinan matriks ini adalah 0.

```
>det(A)
```

0

Symbolic Matrices

Maxima memiliki matriks simbolis. Tentu saja, Maxima dapat digunakan untuk masalah aljabar linier sederhana seperti itu. Kita dapat mendefinisikan matriks untuk Euler dan Maxima dengan &:=, dan kemudian menggunakan dalam ekspresi simbolis. Bentuk [...] biasa untuk mendefinisikan matriks dapat digunakan di Euler untuk mendefinisikan matriks simbolik.

```
>A &= [a,1,1;1,a,1;1,1,a]; $A  
>$&det(A), $&factor(%)  
>$&invert(A) with a=0  
>A &= [1,a;b,2]; $A
```

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

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

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

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

Untuk mengekstrak vektor eigen tertentu perlu pengindeksan yang cermat.

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

[1, - 1]

Matriks simbolik dapat dievaluasi dalam Euler secara numerik seperti ekspresi simbolik lainnya.

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

1	4
5	2

Dalam ekspresi simbolik, gunakan dengan.

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

Akses ke baris matriks simbolik bekerja seperti halnya dengan matriks numerik. simbolik lainnya.

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

Ekspresi simbolis dapat berisi tugas. Dan itu mengubah matriks A.

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

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

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

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

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

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

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

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

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

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

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

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)
```

Numerical Values in symbolic Expressions

Ekspresi simbolis hanyalah string yang berisi ekspresi. Jika kita ingin mendefinisikan nilai baik untuk ekspresi simbolik maupun ekspresi numerik, kita harus menggunakan "&:=". sebagai numerik dalam ekspresi numerik. Jadi kita bisa menggunakannya di sini.

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

```
1      3.14159  
4          5
```

Masih ada perbedaan antara bentuk numerik dan simbolik. Saat mentransfer matriks ke bentuk simbolis, pendekatan fraksional untuk real akan digunakan.

```
>$&A
```

Untuk menghindarinya, ada fungsi "m xmset(variable)".

```
>m xmset (A); $&A
```

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

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

Ketepatan angka floating point besar dapat diubah.
be changed.

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

```
3.14159265358979323846264338327950288419716939937510582097494\  
4592307816406286208998628034825342117068b0
```

Variabel numerik dapat digunakan dalam ekspresi simbolis 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)
```

Demo - Interest Rates

Di bawah ini, kami menggunakan Euler Math Toolbox (EMT) untuk perhitungan 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 5000 (katakanlah dalam dolar).

```
>K=5000
```

5000

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

```
>K*1.03
```

5150

Euler akan memahami sintaks berikut juga.

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

5150

Tetapi lebih mudah menggunakan faktornya.

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

1.03

5150

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

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

6719.58189672

Untuk tujuan kita, kita dapat mengatur format menjadi 2 digit setelah titik desimal.
ku bunga
majemuk.

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

6719.58

Mari kita cetak yang dibulatkan menjadi 2 digit dalam 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 9? Untuk ini, bahasa matriks Euler sangat membantu. Anda tidak harus menulis loop, tetapi cukup masukkan

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

Real 1 x 11 matrix

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

Bagaimana keajaiban ini bekerja? 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 dalam Euler dapat diterapkan pada elemen vektor untuk 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 kami mendapatkan vektor nilai. Jadi

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

Tentu saja, cara realistik untuk menghitung suku bunga ini adalah dengan membulatkan ke sen terdekat setelah setiap tahun. Mari kita tambahkan fungsi untuk ini. api cukup masukkan

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

Mari kita bandingkan dua hasil, 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 mengulang selama bertahun-tahun. Euler memberikan banyak solusi untuk ini.

Cara termudah adalah iterasi fungsi, yang mengulangi fungsi tertentu beberapa kali. emecahkan masalah kehidupan nyata.

Asumsikan Anda memiliki modal awal 5000 (katakanlah dalam dolar).

```
>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 tertentu dari vektor, 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. Ingat 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.

Solving Equations

Sekarang kita mengambil fungsi yang lebih maju, yang menambahkan tingkat uang tertentu setiap tahun. ulatkan dengan nilai penuh.

```
>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("onepay",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("onepay",5000,10)
```

Real 1 x 11 matrix

```
5000.00 4950.00 4898.50 4845.45 ...
```

Kami melihat bahwa uang berkurang. Jelas, jika kita hanya mendapatkan 150 bunga di tahun pertama, tetapi menghapus 200, kita kehilangan uang setiap tahun.

Bagaimana kita bisa menentukan berapa tahun uang itu akan bertahan? Kita harus menulis loop untuk ini. Cara termudah adalah dengan iterasi cukup lama.

```
>VKR=iterate("onepay",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
```

Alasan untuk ini adalah bahwa bukan nol(VKR<0) mengembalikan vektor indeks i, di mana VKR[i]<0, dan min menghitung indeks minimal.

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

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

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

```
-19.83  
47.00
```

Mari kita coba menjawab pertanyaan yang lebih ambigu. Asumsikan kita tahu bahwa nilainya adalah 0 setelah 50 tahun. Apa yang akan menjadi tingkat bunga?

Ini adalah pertanyaan yang hanya bisa dijawab dengan angka. Di bawah ini, kita akan mendapatkan formula yang diperlukan. Kemudian Anda akan melihat bahwa tidak ada formula yang mudah untuk tingkat bunga. Tapi untuk saat ini, kami bertujuan untuk 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

$$x_{n+1} = x_n \cdot \left(1 + \frac{P}{100}\right) + R$$

Tapi 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.

Selain itu, kami hanya tertarik pada nilai terakhir. Jadi kita ambil indeks [-1].

Mari kita coba tes

```
>f(5000,-200,3,47)
```

-19.83

Now we can solve our problem.

```
>solve("f(5000,-200,x,50)",3)
```

3.15

Rutin memecahkan memecahkan 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 integer.

Symbolic Solutions to the Interest Rate Problem

Kita dapat menggunakan bagian simbolik dari Euler untuk mempelajari masalah tersebut. Pertama kita mendefinisikan fungsi onepay() kita secara simbolis

```
>function op(K) &= K*q+R; \$&op(K)
```

Kita sekarang dapat mengulangi ini.

```
>\$&op(op(op(op(K)))) , \$&expand(%)
```

Kami melihat sebuah pola. Setelah n periode yang kita miliki

atax: $K_n = q^n K + R(1+q+\dots+q^{n-1}) = q^n K + \frac{q^n - 1}{q - 1} R$

Rumusnya adalah rumus untuk jumlah geometri, yang diketahui Maxima. simbolik dari Euler untuk mempelajari masalah tersebut. Pertama kita mendefinisikan fungsi onepay() kita secara simbolis

```
>\$sum(q^k,k,0,n-1); \$& % = ev(% , simpsum)
```

Ini agak rumit. Jumlahnya dievaluasi dengan bendera "simpsum" untuk menguranginya 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)
```

Fungsi tersebut melakukan hal yang sama seperti fungsi f kita sebelumnya. Tapi itu lebih efektif.

```
>longest f(5000,-200,3,47), longest fs(5000,-200,3,47)
```

```
-19.82504734650985  
-19.82504734652684
```

Kita sekarang dapat menggunakan sisa simbolis Euler untuk menanyakan waktu n. Kapan modal kita habis? Dugaan awal kami adalah 30 tahun.

```
>solve ("fs(5000,-330,3,x)",30)
```

```
20.51
```

Jawaban ini mengatakan bahwa itu akan menjadi negatif setelah 21 tahun.

Kita juga dapat menggunakan sisa simbolis Euler untuk menghitung formula pembayaran.

Asumsikan kita mendapatkan pinjaman sebesar K, dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama) meninggalkan sisa hutang sebesar Kn (pada saat pembayaran terakhir). Rumus untuk ini jelas

msi tingkat

bunga 3% per tahun.

```
>equ &= fs(K,R,P,n)=Kn; $&equ
```

Biasanya rumus ini diberikan dalam bentuk

$$i = \frac{P}{100}$$

```
>equ &= (equ with P=100*i); $&equ
```

Kita dapat memecahkan tingkat R secara simbolis.

```
>$&solve(equ,R)
```

Seperti yang Anda lihat dari rumus, fungsi ini mengembalikan kesalahan titik mengambang untuk i=0. Euler tetap merencanakannya.

Tentu saja, kami memiliki batasan berikut.

a mendapatkan pinjaman sebesar K, dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama) meninggalkan sisa hutang sebesar Kn (pada saat pembayaran terakhir). Rumus untuk ini jelas

msi tingkat

bunga 3% per tahun.

```
> $&limit(R(5000,0,x,10),x,0)
```

Jelas, tanpa bunga kita harus membayar kembali 10 tarif 500.

Persamaan juga dapat diselesaikan untuk n. Kelihatannya lebih bagus, jika kita menerapkan beberapa penyederhanaan untuk itu.

injaman sebesar K, dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama) meninggalkan sisa hutang sebesar Kn (pada saat pembayaran terakhir). Rumus untuk ini jelas

msi tingkat

bunga 3% per tahun.

```
> fn &= solve(equ,n) | ratsimp; $&fn
```

BAB 2

KB PEKAN 4: MENGGUNAKAN EMT UNTUK MENGAMBAR GRAFIK 2 DIMENSI (2D)

[a4paper,10pt]article eumat

Menggambar Grafik 2D dengan EMT

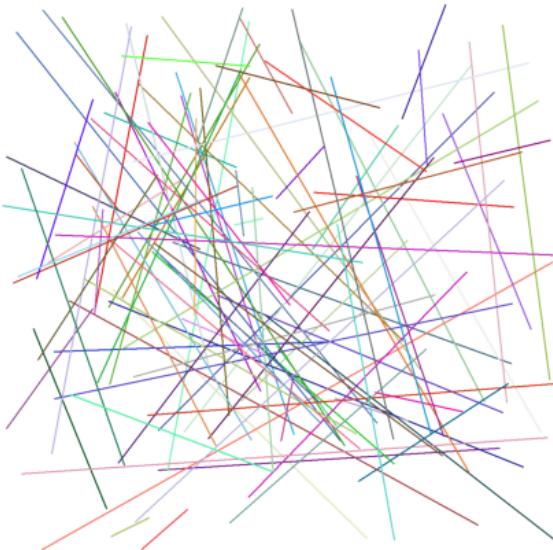
Notebook ini menjelaskan tentang cara menggambar berbagai kurva dan grafik 2D dengan software EMT. EMT menyediakan fungsi plot2d() untuk menggambar berbagai kurva dan grafik dua dimensi (2D).

Basic Plots

Ada fungsi plot yang sangat mendasar. Ada koordinat layar, yang selalu berkisar dari 0 hingga 1024 di setiap sumbu, tidak peduli apakah layarnya persegi atau tidak. Dan ada plot koordinat, yang dapat diatur dengan setplot(). Pemetaan antar koordinat bergantung pada jendela plot saat ini. Misalnya, shrinkwindow() default menyisakan ruang untuk label sumbu dan a judul plot.

Dalam contoh, kita hanya menggambar beberapa garis acak dalam berbagai warna. Untuk detail tentang ini fungsi, 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;
```

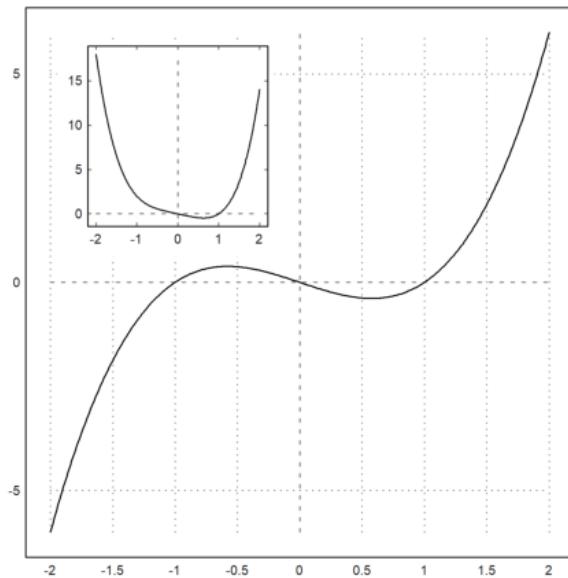
Grafik perlu ditahan, karena perintah plot() akan menghapus plot jendela.

Untuk membersihkan semuanya, we use reset().

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

Untuk contoh lain, kami menggambar plot sebagai sisipan di plot lain. Ini dilakukan untuk mendefinisikan jendela plot yang lebih kecil. Perhatikan bahwa jendela ini tidak menyediakan ruang untuk sumbu label di luar jendela plot. Kita harus menambahkan beberapa margin untuk ini sesuai kebutuhan. Perhatikan bahwa kami menyimpan dan memulihkan jendela penuh, dan menahan plot saat ini saat kami 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):
```



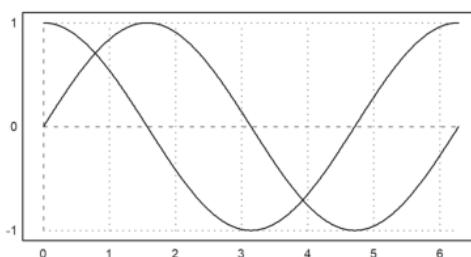
```
>hold off;
>window(ow);
```

Sebuah plot dengan beberapa angka dicapai dengan cara yang sama. Ada angka utilitas() fungsi untuk ini.
Plot Aspect

Plot default menggunakan jendela plot persegi. Anda dapat mengubah ini dengan fungsi aspek(). Jangan lupa untuk mengatur ulang aspek nanti. Anda juga dapat mengubah default ini di menu dengan "Set Aspect" ke rasio aspek tertentu atau ke ukuran jendela grafis saat ini.

Tetapi Anda juga dapat mengubahnya untuk satu plot. Untuk ini, ukuran area plot saat ini berubah, 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 memplot plot 2D dari

- ekspresi
- fungsi, variabel, atau kurva parameter,
- 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.

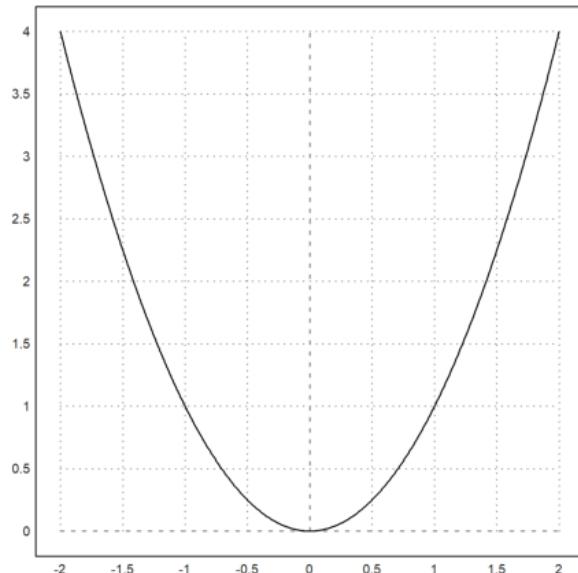
Plot Ekspresi atau Variabel

Ekspresi tunggal dalam "x" (mis. "4*x^2") atau nama fungsi (mis. "f") menghasilkan grafik fungsi.

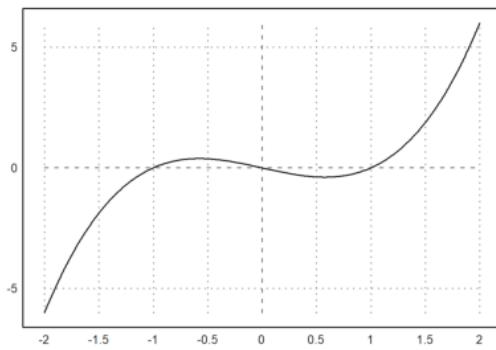
Berikut adalah contoh paling dasar, yang menggunakan rentang default dan menetapkan rentang y yang tepat agar sesuai dengan plot fungsi.

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.

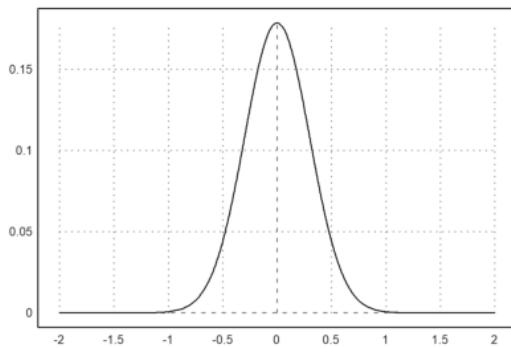
```
>plot2d("x^2") :
```



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



```
>a:=5.6; plot2d("exp(-a*x^2)/a"); insimg(30); // menampilkan gambar hasil plot setinggi 250px
```

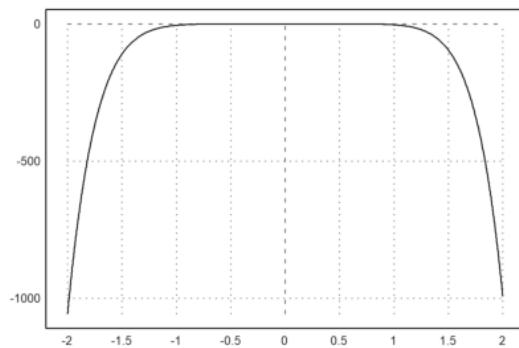


Dari beberapa contoh sebelumnya Anda dapat melihat bahwa aslinya gambar plot menggunakan sumbu X dengan rentang nilai dari -2 sampai dengan 2. Untuk mengubah rentang nilai X dan Y, Anda dapat menambahkan nilai-nilai batas X (dan Y) di belakang ekspresi yang digambar.

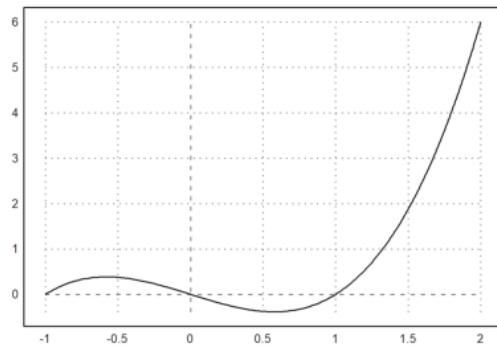
Rentang plot diatur dengan persamaan parameter berikut:

- a,b: rentang x (default -2,2)
- c,d: rentang y (default: skala dengan nilai)
- r: alternatif radius disekitar pusat plot
- cx,xy: koordinat pusat plot (default 0,0)

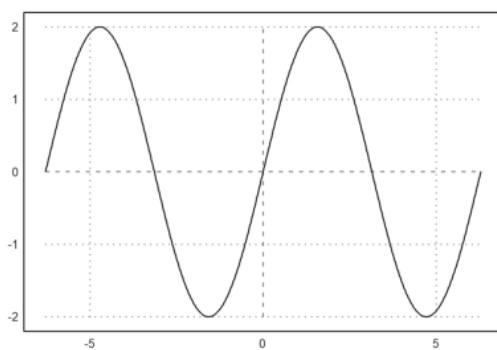
```
>plot2d("x^5-4x^8",-2,2); //plot x pangkat 5 - 4x pangkat 8 dengan interval -2 sampai 2
```



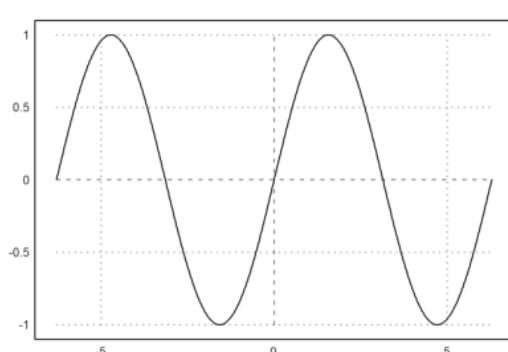
```
>plot2d("x^3-x",-1,2):
```



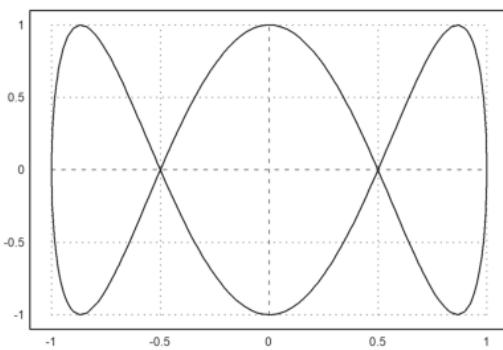
```
>plot2d("2sin(x)",-2*pi,2*pi): // plot sin(x) pada interval [-2pi, 2pi]
```



```
>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 untuk muncul

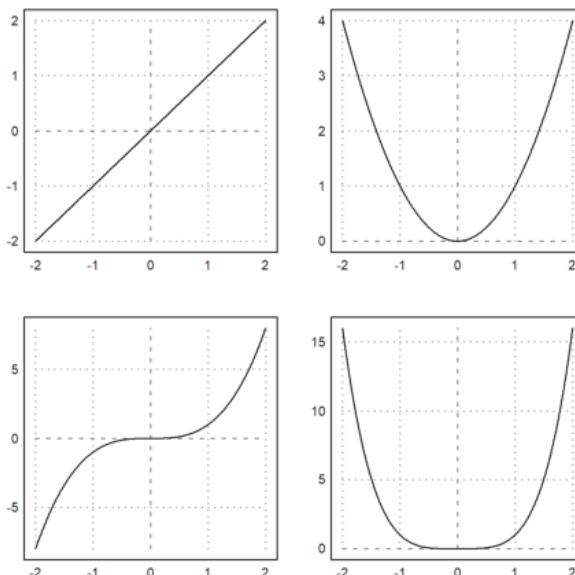
- di jendela terpisah yang dapat diubah ukurannya,
- di jendela buku catatan.

Lebih banyak gaya dapat dicapai dengan perintah plot tertentu.

Bagaimanapun, tekan tombol tabulator untuk melihat plot, jika disembunyikan.

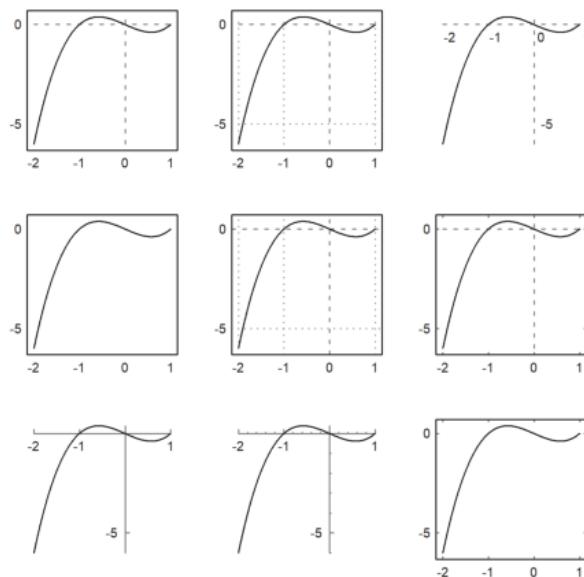
Untuk membagi jendela menjadi beberapa plot, gunakan perintah figure(). Dalam contoh, kami memplot x^1 hingga x^4 menjadi 4 bagian jendela. figure(0) mengatur ulang jendela default.

```
>reset;
>figure(2,2); ...
>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 menunjukkan berbagai gaya kisi dalam satu gambar (lihat di bawah untuk perintah figure()). Gaya kisi=0 tidak disertakan. Ini menunjukkan tidak ada grid dan tidak ada bingkai.

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

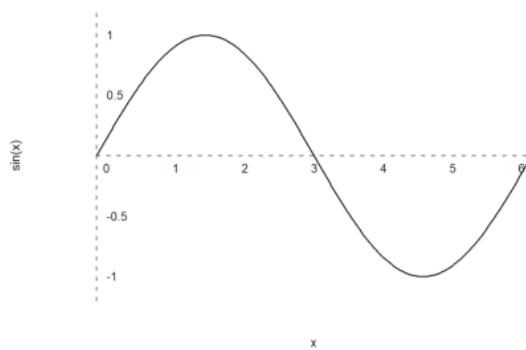


Jika argumen ke `plot2d()` adalah ekspresi yang diikuti oleh empat angka, angka-angka ini adalah rentang x dan y untuk plot.

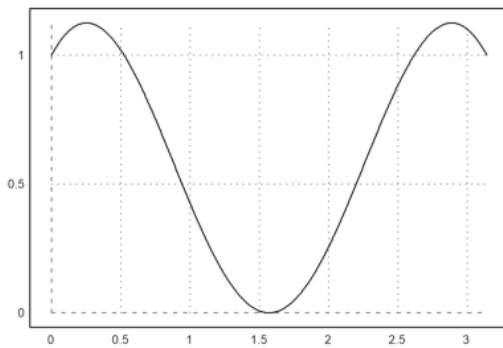
Atau, a, b, c, d dapat ditentukan sebagai parameter yang ditetapkan sebagai a=... dll.

Dalam contoh berikut, kita 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)":
```



```
>plot2d("sin(x)+cos(2*x)",0,1pi):
```

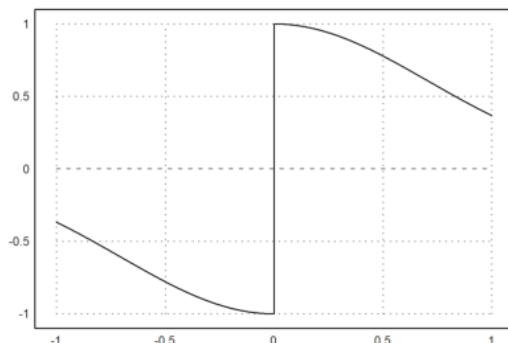


Gambar yang dihasilkan dengan memasukkan plot ke dalam jendela teks disimpan di direktori yang sama dengan buku catatan, secara default di subdirektori bernama "gambar". Mereka juga digunakan oleh ekspor HTML.

Anda cukup menandai gambar apa saja dan menyalinnya ke clipboard dengan Ctrl-C. Tentu saja, Anda juga dapat mengekspor grafik saat ini dengan fungsi di menu File.

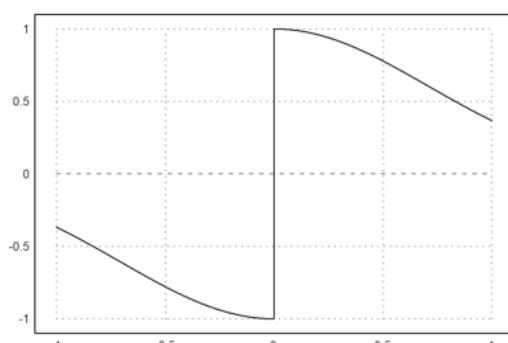
Fungsi atau ekspresi dalam plot2d dievaluasi secara adaptif. Untuk kecepatan lebih, matikan plot adaptif dengan <adaptive dan tentukan jumlah subinterval dengan n=... Ini hanya diperlukan dalam kasus yang jarang terjadi.

```
>plot2d("sign(x)*exp(-x^2)", -1, 1, <adaptive, n=5000) :
```

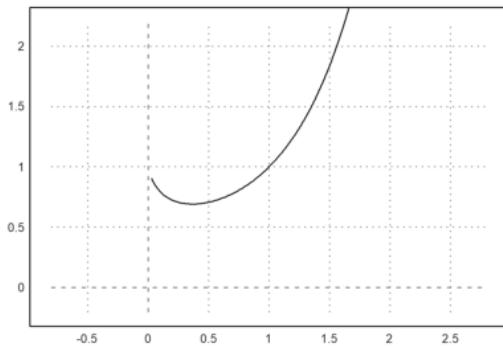


gambar sama dengan n=10000

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

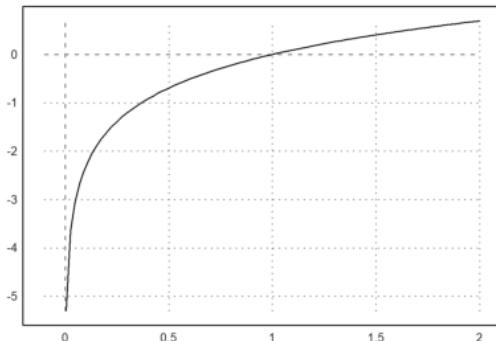


```
>plot2d("x^x", r=1.2, cx=1, cy=1):
```



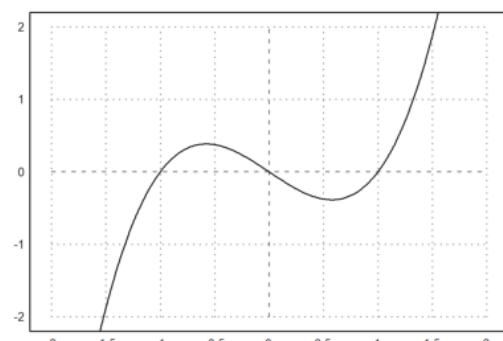
Perhatikan bahwa x^x tidak didefinisikan untuk $x \leq 0$. Fungsi plot2d menangkap kesalahan ini, dan mulai merencanakan segera setelah fungsi didefinisikan. Ini berfungsi untuk semua fungsi yang mengembalikan NAN keluar dari jangkauan definisinya.

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

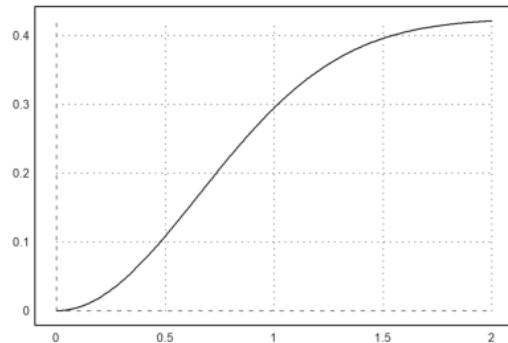


Parameter square=true (atau >square) memilih y-range secara otomatis sehingga hasilnya adalah jendela plot persegi. Perhatikan bahwa secara default, Euler menggunakan ruang persegi di dalam jendela plot.

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

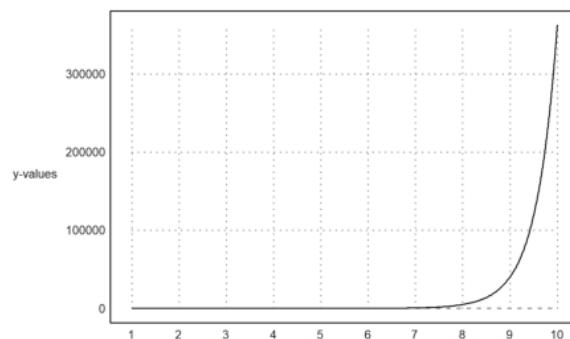


```
>plot2d(''integrate("sin(x)*exp(-x^2)",0,x)'',0,2): // plot integral
```



Jika Anda membutuhkan lebih banyak ruang untuk label-y, panggil shrinkwindow() dengan parameter yang 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.

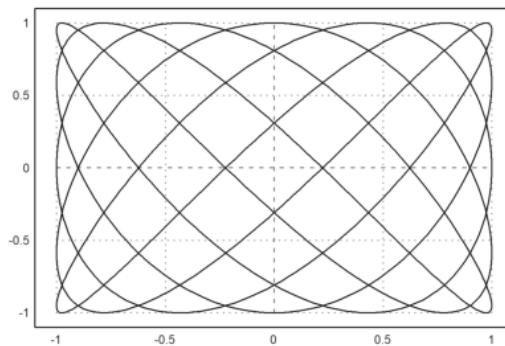
```
>plot2d("gamma(x)",1,10000,yl="y-values",smaller=100,<vertical):
```

* 1e+

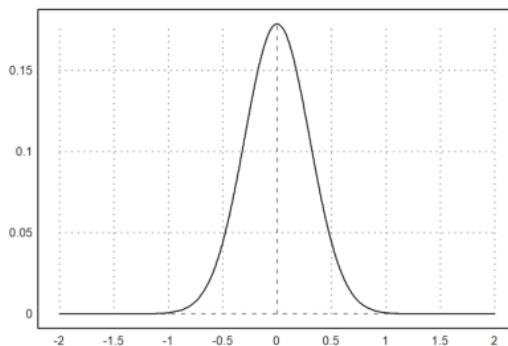
y-values 131

angkanya terlalu besar sehingga didefinisikan dengan e

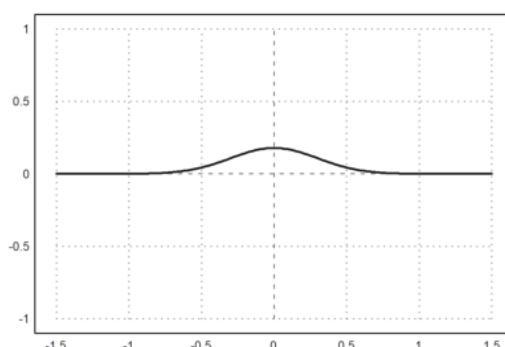
```
>x=linspace(0,2pi,1000); plot2d(sin(5x),cos(7x)):
```



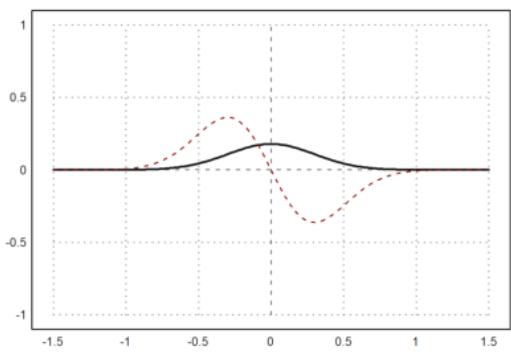
```
>a:=5.6; expr &= exp(-a*x^2)/a; // define expression  
>plot2d(expr,-2,2); // plot from -2 to 2
```



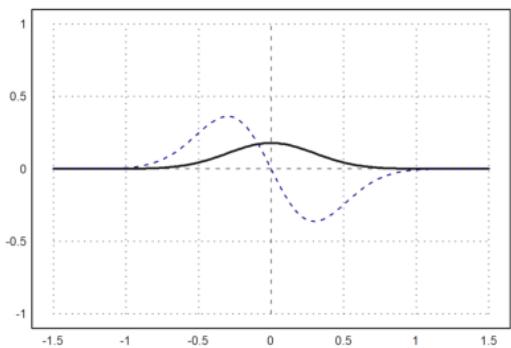
```
>plot2d(expr,r=1,thickness=2); // plot in a square around (0,0)
```



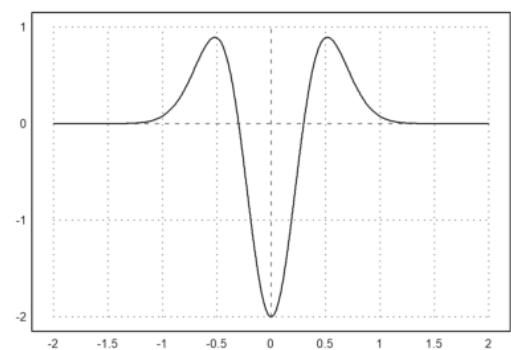
```
>plot2d(&diff(expr,x),>add,style="--",color=red); // add another plot
```



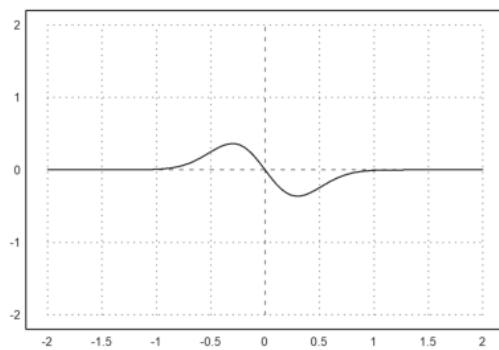
```
>plot2d(&diff(expr,x),>add,style="--",color=blue): // add another plot
```



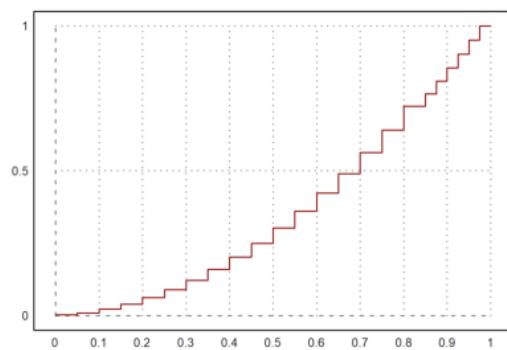
```
>plot2d(&diff(expr,x,2),a=-2,b=2,c=-2,d=1): // plot in rectangle
```



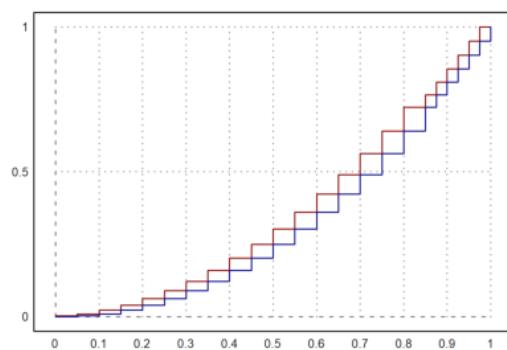
```
>plot2d(&diff(expr,x),a=-2,b=2,>square): // keep plot square
```



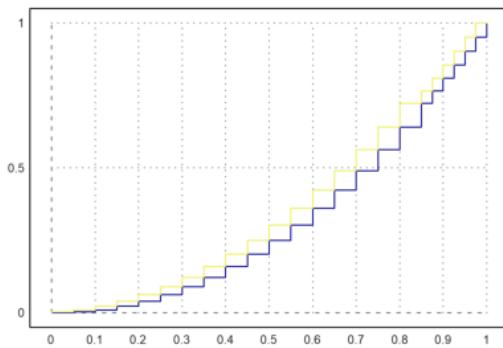
```
>plot2d("x^2",0,1,steps=1,color=red,n=10):
```



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



```
>plot2d("x^2",>add,steps=3,color=yellow,n=10):
```



Fungsi dalam satu Parameter

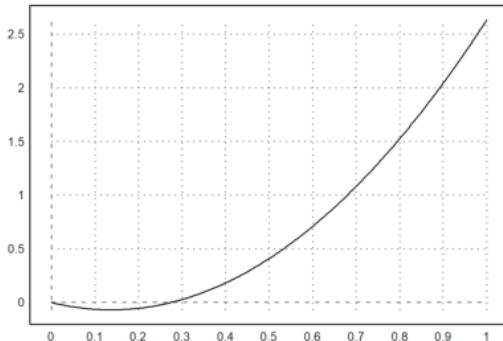
Fungsi plot yang paling penting untuk plot planar adalah `plot2d()`. Fungsi ini diimplementasikan dalam bahasa Euler dalam file "plot.e", yang dimuat di awal program.

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

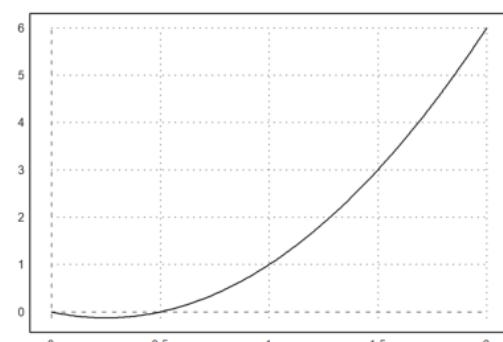
```
>function f(x,a) := x^2/a+a*x^2-x; // define a function
```

mendefinisikan fungsi

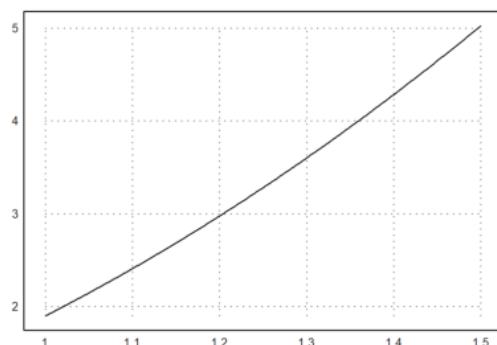
```
>a=0.3; plot2d("f",0,1;a); // plot with a=0.3
```



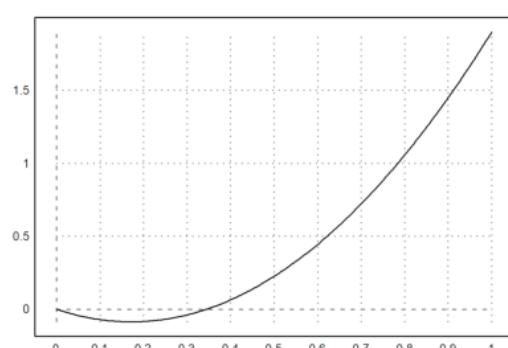
```
>a=1; plot2d("f",0,2;a); // plot dengan fungsi f, a=1
```



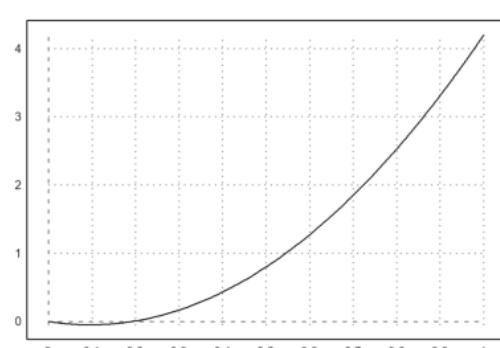
```
>plot2d("f",1,3/2;0.4): // plot with a=0.4
```



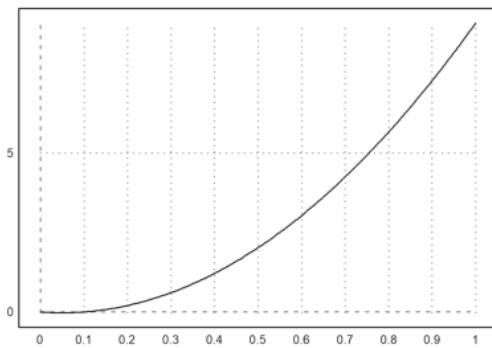
```
>plot2d("f",0,1;0.4): // plot with a=0.4
```



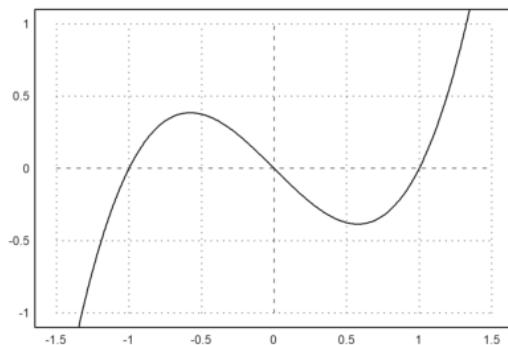
```
>plot2d({{"f",0.2}},0,1): // plot with a=0.2
```



```
>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 dari fungsi yang diterima

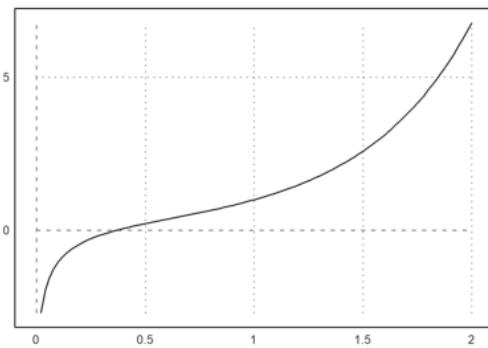
- ekspresi atau ekspresi simbolik dalam x
- fungsi atau fungsi simbolis dengan nama sebagai "f"
- fungsi simbolis hanya dengan nama f

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

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

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

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

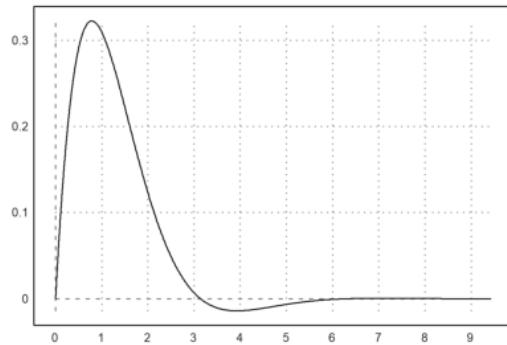


Tentu saja, untuk ekspresi atau ekspresi simbolik, nama variabel sudah cukup untuk memplotnya.

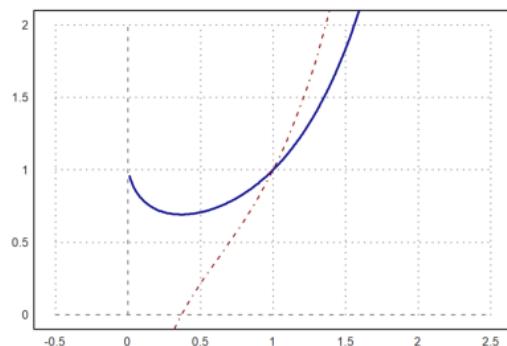
```
>expr &= sin(x) * exp(-x)
```

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

```
>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: Default adalah 1.

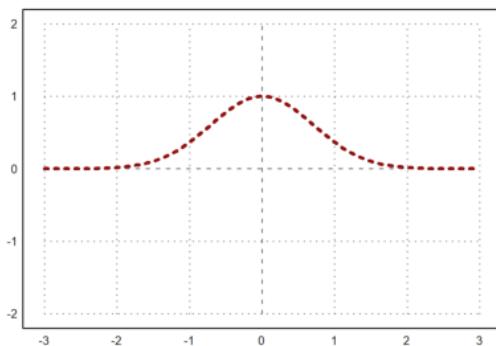
Warna dapat dipilih sebagai salah satu warna default, atau sebagai warna RGB.

- 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 terang, kuning

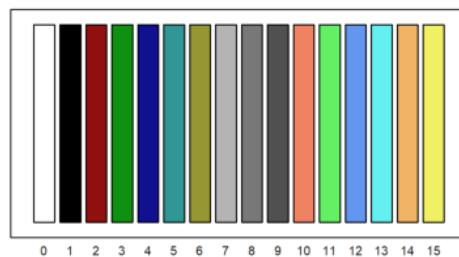
- `rgb(merah, hijau, biru)`: parameter adalah real dalam [0,1].

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



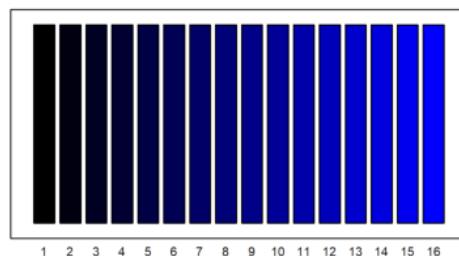
Berikut adalah tampilan warna EMT yang telah ditentukan sebelumnya.

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



Tapi Anda bisa menggunakan warna apa saja.

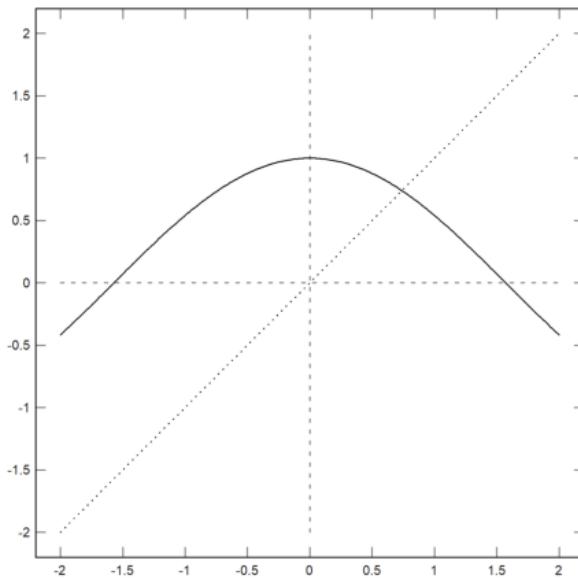
```
>columnsplot(ones(1,16), grid=0, color=rgb(0,0,linspace(0,1,15))) :
```



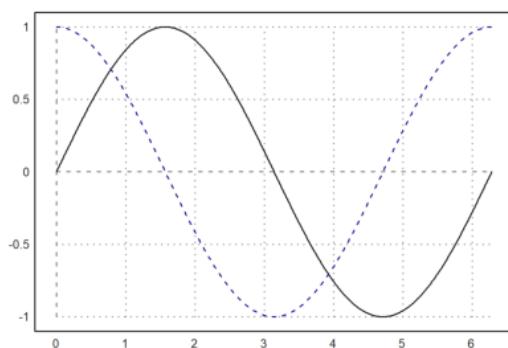
Menggambar Beberapa Kurva pada bidang koordinat yang sama

Plot lebih dari satu fungsi (multiple function) ke dalam satu jendela dapat dilakukan dengan berbagai cara. Salah satu metode menggunakan `>add` untuk beberapa panggilan ke `plot2d` secara keseluruhan, tetapi panggilan pertama. Kami telah menggunakan fitur ini dalam contoh di atas.

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

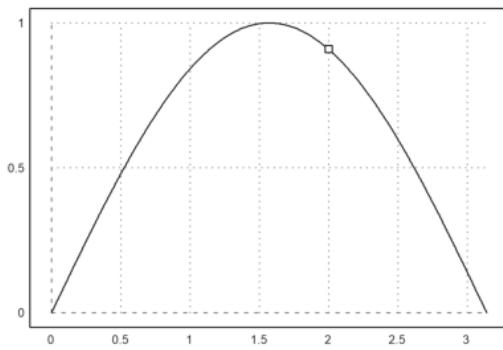


```
>aspect(1.5); plot2d("sin(x)", 0, 2pi); plot2d("cos(x)", color=blue, style="--",>add):
```



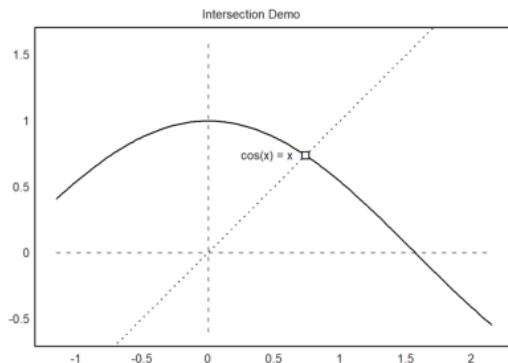
Salah satu kegunaan `>add` adalah untuk menambahkan titik pada kurva.

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



Kami menambahkan titik persimpangan dengan label (pada posisi "cl" untuk kiri tengah), dan memasukkan hasilnya ke dalam notebook. Kami juga menambahkan judul ke 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, kami memplot fungsi $\text{sinc}(x)=\sin(x)/x$ dan ekspansi Taylor ke-8 dan ke-16. Kami menghitung ekspansi ini menggunakan Maxima melalui ekspresi simbolis.

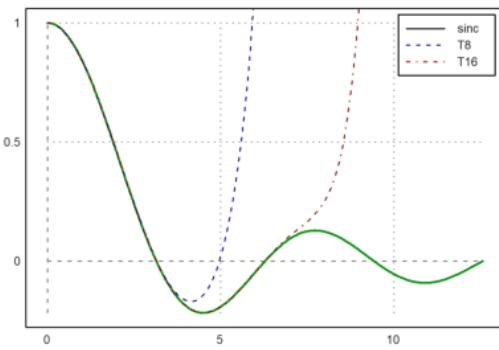
Plot ini dilakukan dalam perintah multi-baris berikut dengan tiga panggilan ke `plot2d()`. Yang kedua dan yang ketiga memiliki set flag `>add`, yang membuat plot menggunakan rentang sebelumnya.

Kami menambahkan kotak label yang menjelaskan fungsi.

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

$$\frac{x^4}{120} - \frac{x^2}{6} + 1$$

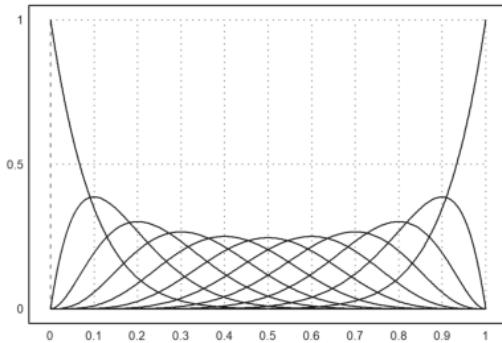
```
>plot2d("sinc(x)",0,4pi,color=green,thickness=2); ...
>  plot2d(&taylor(sin(x)/x,x,0,8),>add,color=blue,style="--"); ...
>  plot2d(&taylor(sin(x)/x,x,0,16),>add,color=red,style="-.-"); ...
>  labelbox(["sinc", "T8", "T16"], styles=["-", "--", "-.-"], ...
>    colors=[black,blue,red]):
```



Dalam contoh berikut, kami menghasilkan Bernstein-Polinomial.

$$B_i(x) = \binom{n}{i} x^i (1-x)^{n-i}$$

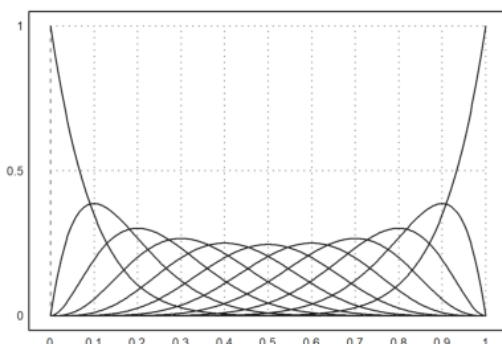
```
>plot2d("(1-x)^10",0,1); // plot first function
>for i=1 to 10; plot2d("bin(10,i)*x^i*(1-x)^(10-i)",>add); end;
>insimg;
```



Metode kedua menggunakan pasangan matriks nilai-x dan matriks nilai-y yang berukuran sama.

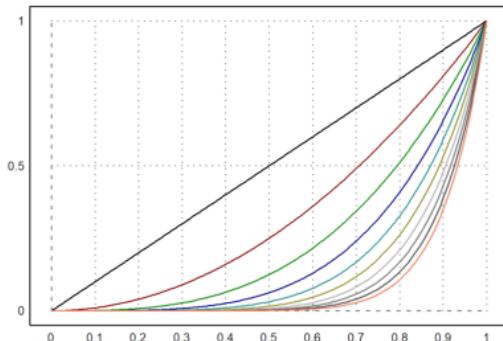
Kami menghasilkan matriks nilai dengan satu Polinomial Bernstein di setiap baris. Untuk ini, kita cukup menggunakan vektor kolom i . Lihat pengantar tentang bahasa matriks untuk mempelajari lebih detail.

```
>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):
```



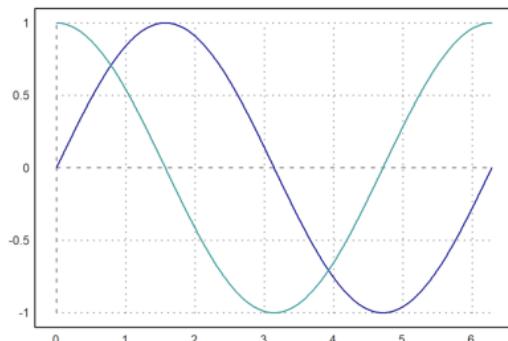
Perhatikan bahwa parameter warna dapat berupa vektor. Kemudian setiap warna digunakan untuk setiap baris matriks.

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

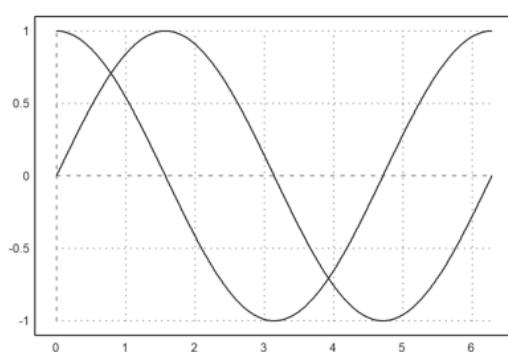


Metode lain adalah menggunakan vektor ekspresi (string). Anda kemudian dapat menggunakan larik warna, larik gaya, dan larik ketebalan dengan panjang yang sama.

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



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



Kita bisa mendapatkan vektor seperti itu dari Maxima menggunakan makelist() dan mxm2str().

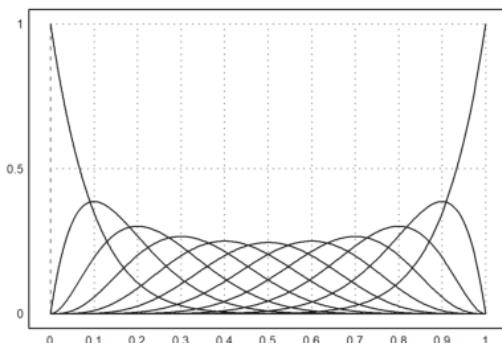
```
>v &= makelist(binomial(10,i)*x^i*(1-x)^(10-i),i,0,10) // make list
```

```
          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) // 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
```

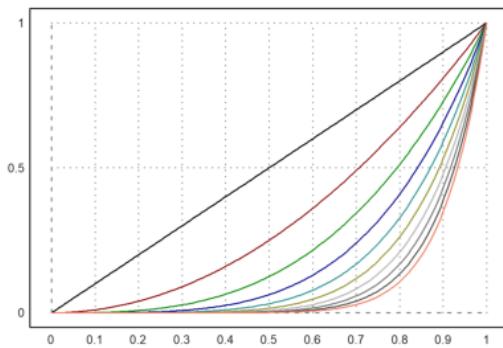


Alternatif lain adalah dengan menggunakan bahasa matriks Euler.

Jika ekspresi menghasilkan matriks fungsi, dengan satu fungsi di setiap baris, semua fungsi ini akan diplot ke dalam satu plot.

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

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

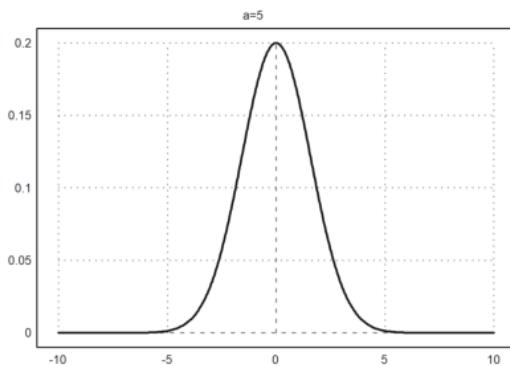


Ekspresi dan fungsi satu baris dapat melihat variabel global.

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 kita meneruskan a=5 ke fungsi f, yang kita plot dari -10 hingga 10.

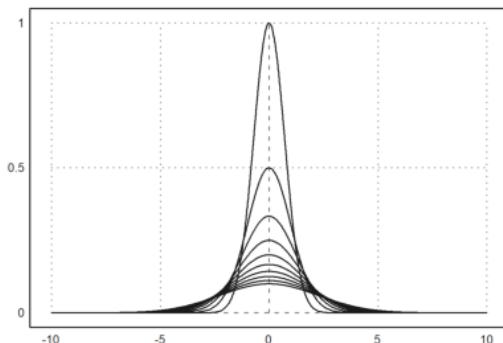
```
>function f(x,a) := 1/a*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5");
```



Atau, gunakan koleksi dengan nama fungsi dan semua parameter tambahan. Daftar khusus ini disebut koleksi panggilan, dan itu adalah cara yang lebih disukai untuk meneruskan argumen ke fungsi yang dengan sendirinya diteruskan sebagai argumen ke fungsi lain.

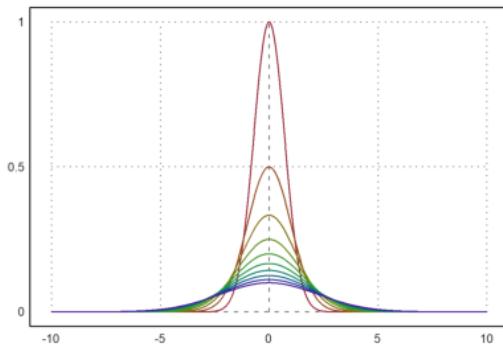
Dalam contoh berikut, kami menggunakan loop untuk memplot beberapa fungsi (lihat tutorial tentang pemrograman untuk loop).

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



Kami dapat mencapai hasil yang sama dengan cara berikut menggunakan bahasa matriks EMT. Setiap baris matriks $f(x,a)$ adalah 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)):
```



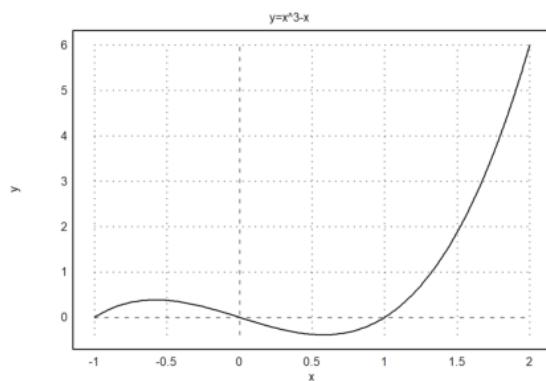
Label Teks

Dekorasi sederhana bisa

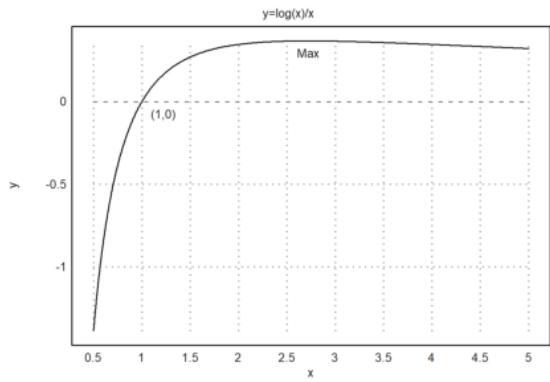
- judul dengan `judul="..."`
- x- dan y-label dengan `xl="...", yl="..."`
- label teks lain dengan `label("...",x,y)`

Perintah label akan memplot ke dalam plot saat ini pada koordinat plot (x,y). Itu bisa mengambil argumen posisi.

```
>plot2d("x^3-x",-1,2,title="y=x^3-x",yl="y",xl="x"):
```

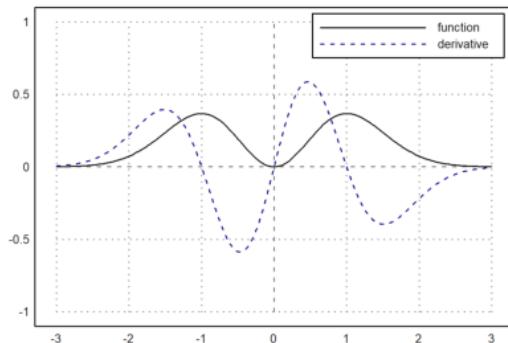


```
>expr := "log(x)/x"; ...
> plot2d(expr,0.5,5,title="y="+expr,xl="x",yl="y"); ...
> label("(1,0)",1,0); label("Max",E,expr(E),pos="lc"):
```



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):
```

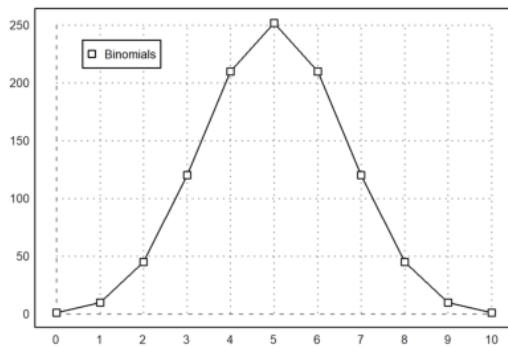


Kotak ditambatkan di kanan atas secara default, tetapi `> kiri` menambatkannya di kiri atas. Anda dapat memindahkannya ke tempat yang Anda suka. Posisi jangkar adalah sudut kanan atas kotak, dan angkanya adalah pecahan dari ukuran jendela grafik. Lebarnya otomatis.

Untuk plot titik, kotak label juga berfungsi. Tambahkan parameter `>points`, atau vektor flag, satu untuk setiap label.

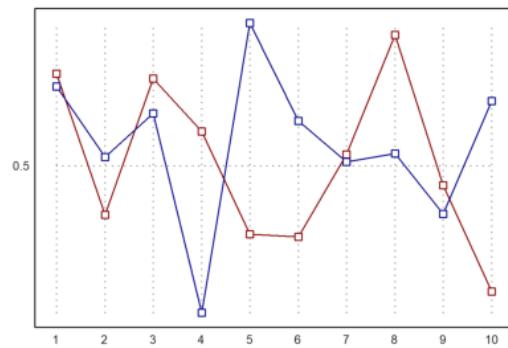
Dalam contoh berikut, hanya ada satu fungsi. Jadi kita bisa menggunakan string sebagai pengganti vektor string. Kami mengatur warna teks menjadi hitam untuk contoh ini.

```
>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. Ada lebih banyak plot khusus untuk keperluan statistik (lihat tutorial tentang statistik).

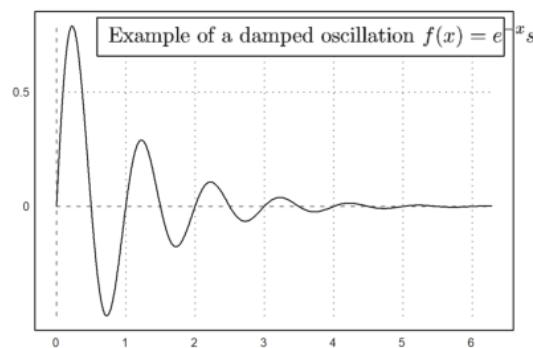
```
>statplot(1:10,random(2,10),color=[red,blue]):
```



Fitur serupa adalah fungsi textbox().

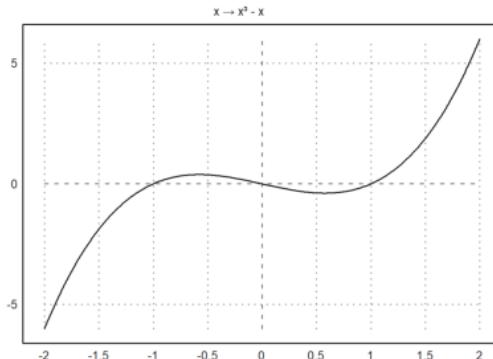
Lebar secara default adalah lebar maksimal dari baris teks. Tapi itu bisa diatur oleh pengguna juga.

```
>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):
```



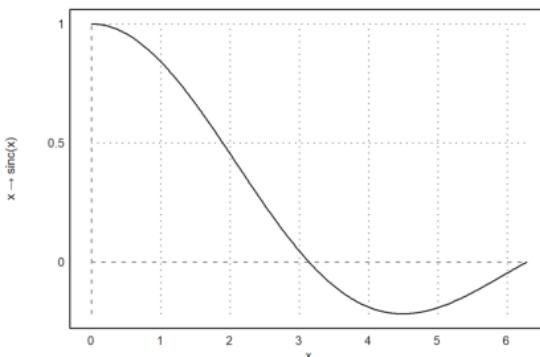
Label teks, judul, kotak label, dan teks lainnya dapat berisi string Unicode (lihat sintaks EMT untuk mengetahui lebih lanjut tentang string Unicode).

```
>plot2d("x^3-x",title=u"x &rarr; x3 - x"):
```



Label pada sumbu x dan y bisa vertikal, begitu juga sumbunya.

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



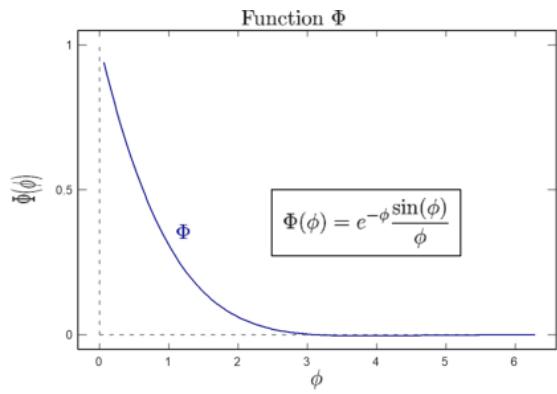
**LaTeX

Anda juga dapat memplot rumus LaTeX jika Anda telah menginstal sistem LaTeX. Saya merekomendasikan MiKTeX. Jalur ke biner "lateks" dan "dvipng" harus berada di jalur sistem, atau Anda harus mengatur LaTeX di menu opsi.

Perhatikan, bahwa penguraian LaTeX lambat. Jika Anda ingin menggunakan LaTeX dalam plot animasi, Anda harus memanggil `latex()` sebelum loop sekali dan menggunakan hasilnya (gambar dalam matriks RGB).

Dalam plot berikut, kami menggunakan LaTeX untuk label x dan y, label, kotak label, dan judul plot.

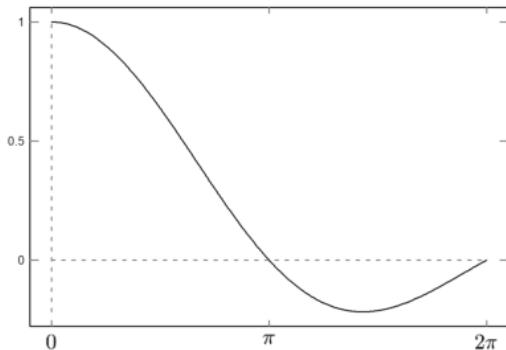
```
>plot2d("exp(-x)*sin(x)/x",a=0,b=2pi,c=0,d=1,grid=6,color=blue, ...
> title=latex("\text{Function } \Phi"), ...
> xl=latex("\phi"),yl=latex("\Phi(\phi)"); ...
>textbox( ...
> latex("\Phi(\phi) = e^{-\phi} \frac{\sin(\phi)}{\phi}"),x=0.8,y=0.5); ...
>label(latex("\Phi",color=blue),1,0.4):
```



Seringkali, kami menginginkan spasi dan label teks non-konformal pada sumbu x. Kita dapat menggunakan `xaxis()` dan `yaxis()` seperti yang akan kita tunjukkan nanti.

Cara termudah adalah dengan membuat plot kosong dengan bingkai menggunakan `grid=4`, lalu menambahkan grid dengan `ygrid()` dan `xgrid()`. Dalam 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, fungsi juga dapat digunakan.

```
>function map f(x) ...
if x>0 then return x^4
else return x^2
endif
endfunction
```

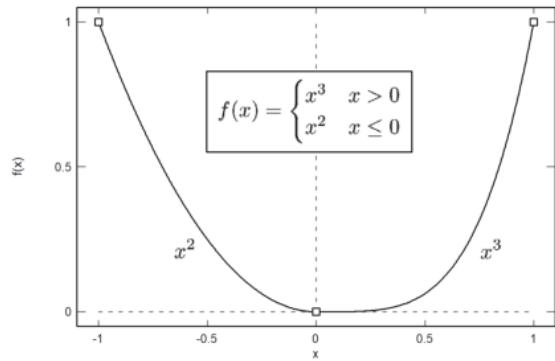
Parameter "peta" membantu menggunakan fungsi untuk vektor. Untuk plot, itu tidak perlu. Tetapi untuk menunjukkan bahwa vektorisasi berguna, kami menambahkan beberapa poin kunci ke plot di $x=-1$, $x=0$ dan $x=1$.

Pada plot berikut, kami juga memasukkan beberapa kode LaTeX. Kami menggunakanannya untuk dua label dan kotak teks. Tentu saja, Anda hanya dapat menggunakan LaTeX jika LaTeX telah 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 \leq 0 \end{cases}"), ...
>  x=0.7,y=0.2):

```



Interaksi pengguna

Saat memplot fungsi atau ekspresi, parameter `>user` memungkinkan pengguna untuk memperbesar dan menggeser plot dengan tombol kursor atau mouse. Pengguna dapat

- perbesar dengan + atau -
- pindahkan plot dengan tombol kursor
- pilih jendela plot dengan mouse
- atur ulang tampilan dengan spasi
- keluar dengan kembali

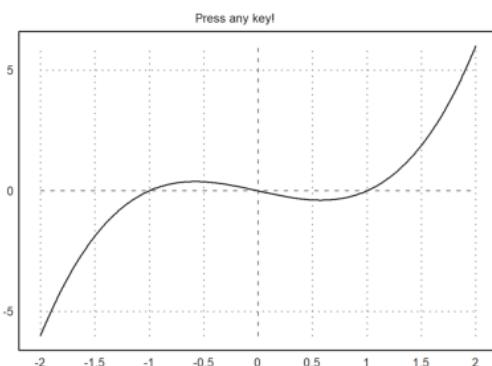
Tombol spasi akan mengatur ulang plot ke jendela plot asli.

Saat memplot data, flag `>user` hanya akan menunggu penekanan tombol.

```

>plot2d({{"x^3-a*x",a=1}},>user,title="Press any key!"):

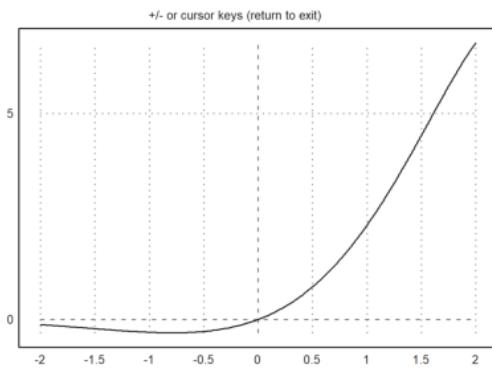
```



```

>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 event mouse atau keyboard. Ini melaporkan mouse ke bawah, mouse dipindahkan atau mouse ke atas, dan penekanan tombol. Fungsi dragpoints() memanfaatkan ini, dan memungkinkan pengguna menyeret titik mana pun dalam plot.

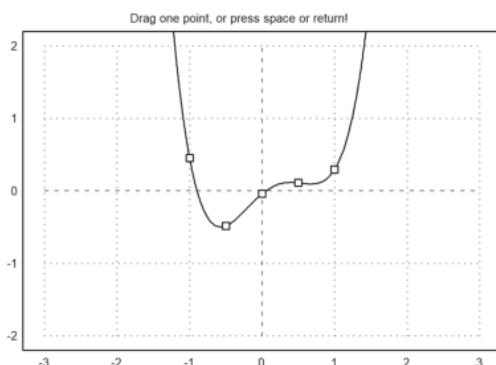
Kita membutuhkan fungsi plot terlebih dahulu. Sebagai contoh, kita interpolasi dalam 5 titik dengan polinomial. Fungsi harus diplot ke area plot tetap.

```
>function plotf(xp,yp,select) ...
d=interp(xp,yp);
plot2d("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 nilai secara global.

Sekarang kita menghasilkan beberapa nilai acak, dan membiarkan pengguna menyeret poin.

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



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

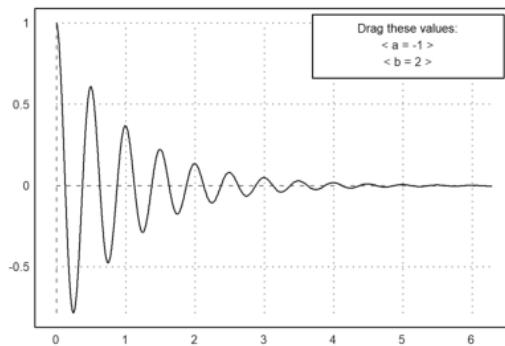
Pertama kita membutuhkan fungsi plot.

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

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

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

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



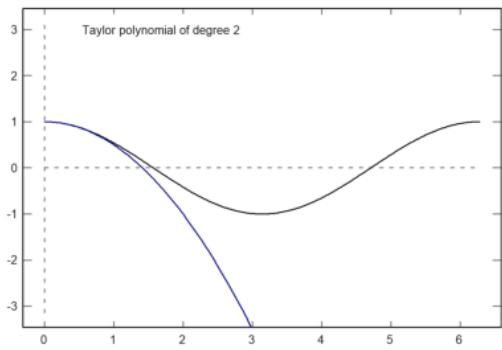
Dimungkinkan untuk membatasi nilai yang diseret ke bilangan bulat. Sebagai contoh, kita menulis fungsi plot, yang memplot polinomial Taylor derajat n ke fungsi kosinus.

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

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

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

```
>nd=dragvalues("plotf", "degree", 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, meninggalkan jejak poin.

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

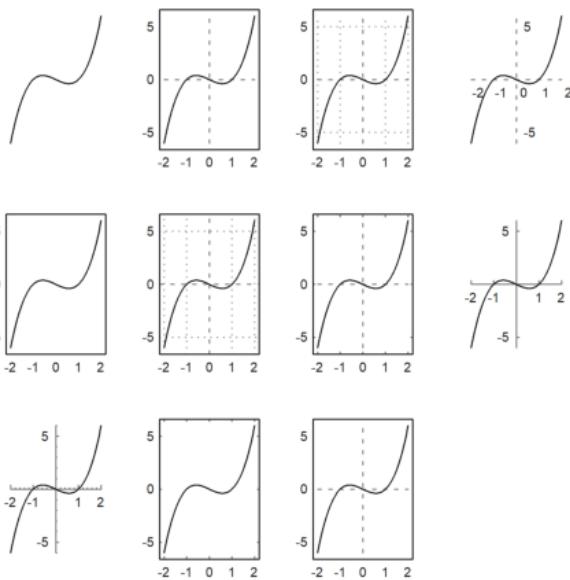
```
plot2d(none,r=1,title="Drag with the mouse, or press any key!");
start=0;
repeat
{flag,m,time}=mousedrag();
if flag==0 then return; endif;
if flag==2 then
    hold on; mark(m[1],m[2]); hold off;
endif;
end
endfunction
```

```
>dragtest // lihat hasilnya dan cobalah lakukan!
```

Gaya Plot 2D

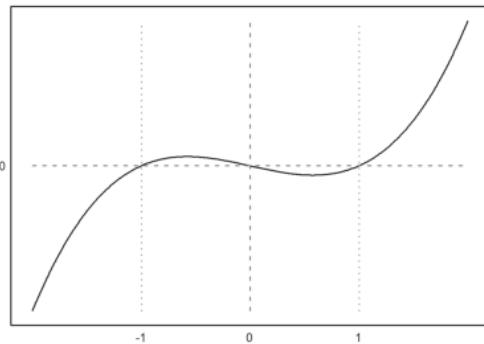
Secara default, EMT menghitung tick sumbu otomatis dan menambahkan label ke setiap tick. Ini dapat diubah dengan parameter grid. Gaya default sumbu dan label dapat dimodifikasi. Selain itu, label dan judul dapat ditambahkan secara manual. Untuk mengatur ulang ke gaya default, gunakan reset().

```
>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):
```



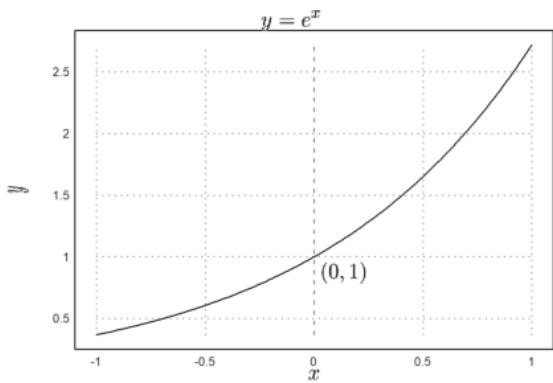
Parameter `<frame>` mematikan frame, dan `framecolor=blue` mengatur frame ke warna biru. Jika Anda ingin centang sendiri, Anda dapat menggunakan `style=0`, dan menambahkan semuanya nanti.

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



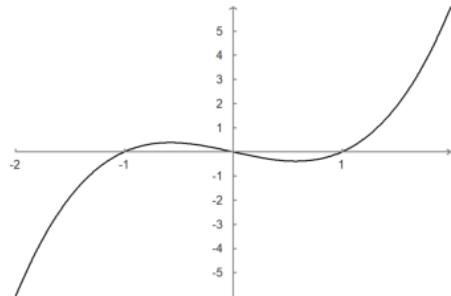
Untuk judul plot dan label sumbu, lihat contoh berikut.

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

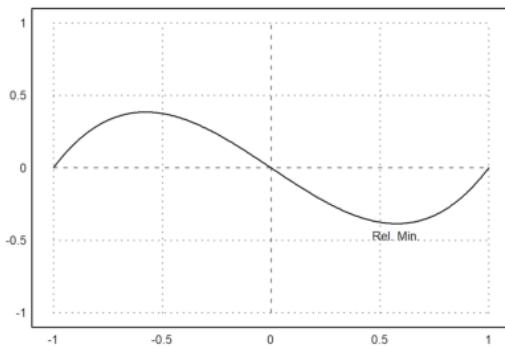


Teks pada plot dapat diatur dengan label(). Dalam contoh berikut, "lc" berarti tengah bawah. Ini mengatur posisi label relatif terhadap koordinat plot.

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

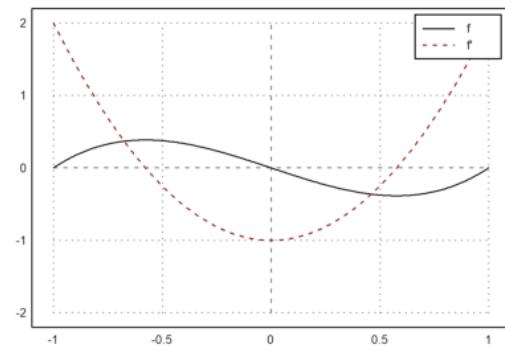
$$x^3 - x$$

```
>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
```

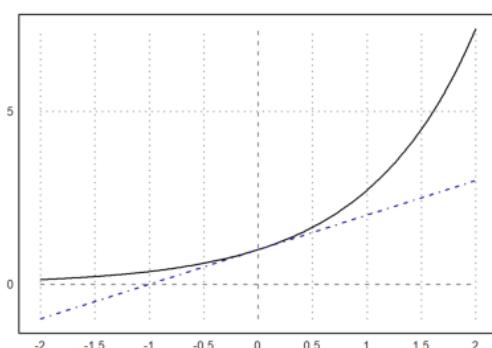


Ada juga kotak teks.

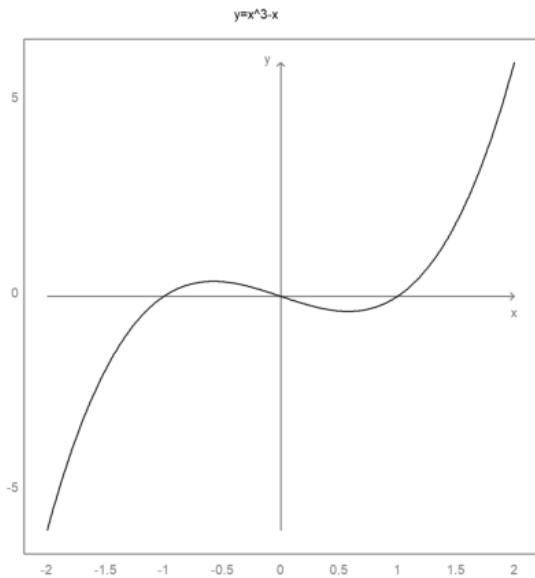
```
>plot2d(&f(x),-1,1,-2,2); // function
>plot2d(&diff(f(x),x),>add,style="--",color=red); // derivative
>labelbox(["f","f'"],["-", "--"],[black,red]): // label box
```



```
>plot2d(["exp(x)", "1+x"],color=[black,blue],style=["-", "-.-"]):
```



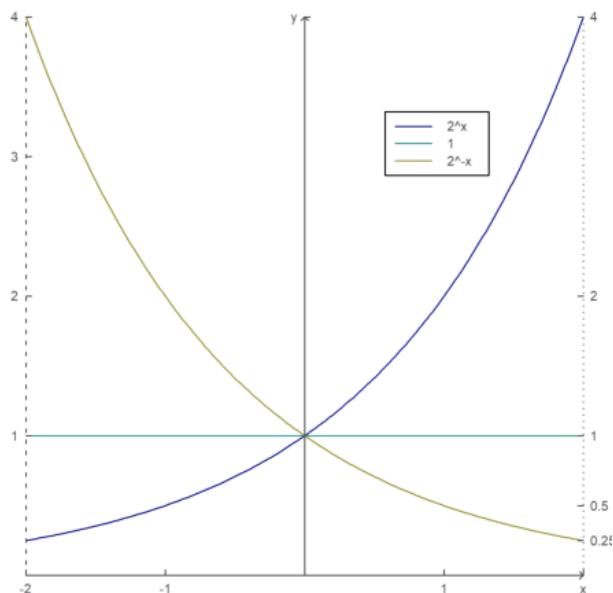
```
>gridstyle("->",color=gray,textcolor=gray,framecolor=gray); ...
> plot2d("x^3-x",grid=1); ...
> settitle("y=x^3-x",color=black); ...
> label("x",2,0,pos="bc",color=gray); ...
> label("y",0,6,pos="cl",color=gray); ...
> reset():
```



Untuk kontrol lebih, sumbu x dan sumbu y dapat dilakukan secara manual.

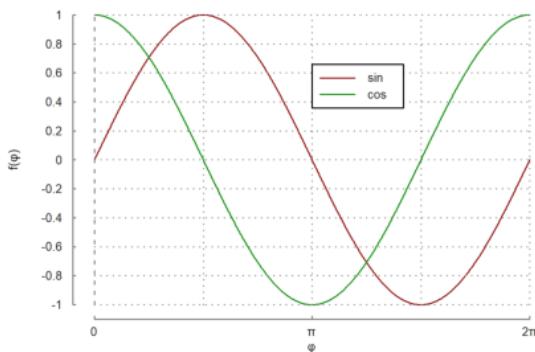
Perintah fullwindow() memperluas jendela plot karena kita tidak lagi membutuhkan tempat untuk label di luar jendela plot. Gunakan shrinkwindow() atau reset() untuk mengatur ulang ke default.

```
>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 sumbu 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, data ini akan digunakan sebagai koordinat x dan y dari suatu kurva. Dalam hal ini, a, b, c, dan d, atau radius r dapat ditentukan, atau jendela plot akan menyesuaikan secara otomatis dengan data. Atau, >persegi dapat diatur untuk menjaga rasio aspek persegi.

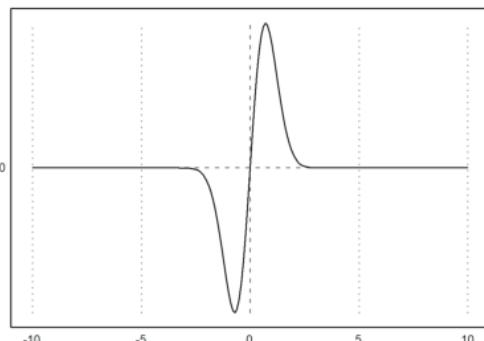
Memplot ekspresi hanyalah singkatan untuk plot data. Untuk plot data, Anda memerlukan satu atau beberapa baris nilai x, dan satu atau beberapa baris nilai y. Dari rentang dan nilai-x, fungsi plot2d akan menghitung data yang akan diplot, secara default dengan evaluasi fungsi yang adaptif. Untuk plot titik gunakan ">titik", untuk garis campuran dan titik gunakan ">tambahan".

Tetapi Anda dapat 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):
```



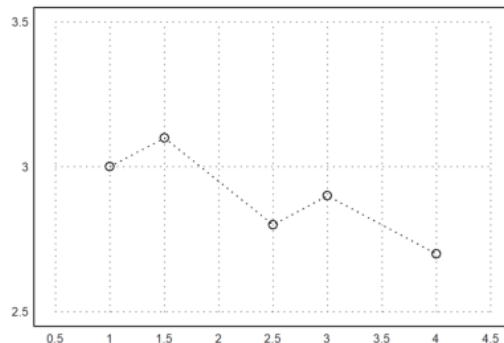
Data juga dapat diplot sebagai titik. Gunakan poin=true untuk ini. Plotnya bekerja seperti poligon, tetapi hanya menggambar sudut-sudutnya.

- style="...": Pilih dari "[", "<>", "o", ".", "..", "+", "*", "[", "<>", "o", "..", "", "|".

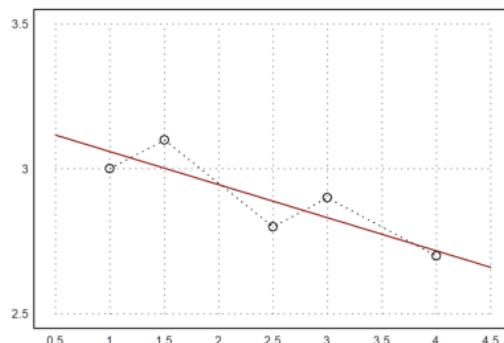
Untuk memplot set poin gunakan >points. Jika warna adalah vektor warna, setiap titik mendapat warna yang berbeda. Untuk matriks koordinat dan vektor kolom, warna berlaku untuk baris matriks.

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="."); // 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 benar-benar poligon. Kita juga dapat memplot kurva atau kurva terisi.

- terisi=true mengisi plot.

- style="...": Pilih dari "", "/", "\", "\\".

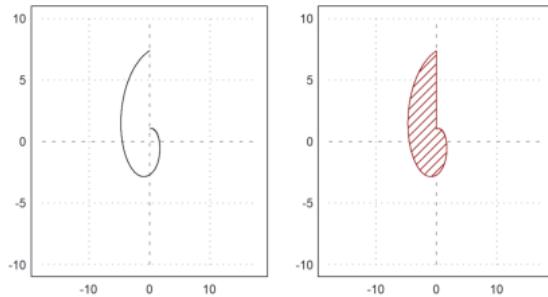
- fillcolor: Lihat di atas untuk warna yang tersedia.

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

```

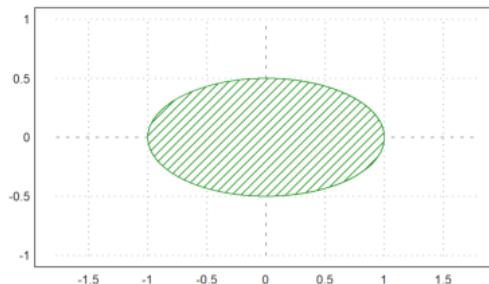


Dalam contoh berikut kami memplot elips terisi dan dua segi enam terisi menggunakan kurva tertutup dengan 6 titik dengan gaya isian berbeda.

```

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

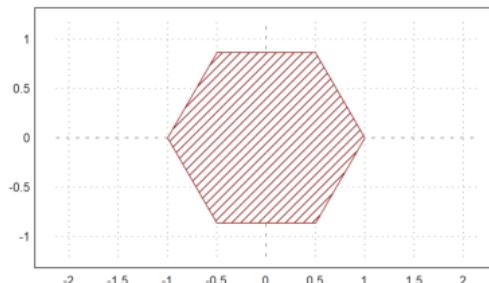
```



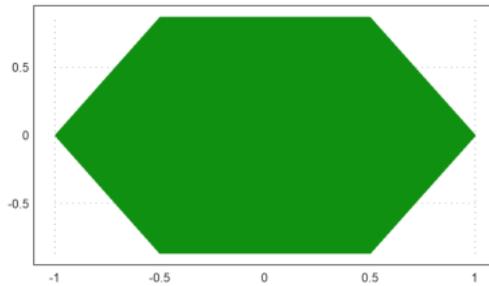
```

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

```

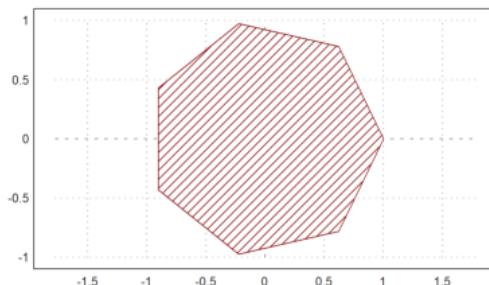


```
>t=linspace(0,2pi,6); plot2d(cos(t),sin(t),>filled,style="#"):
```



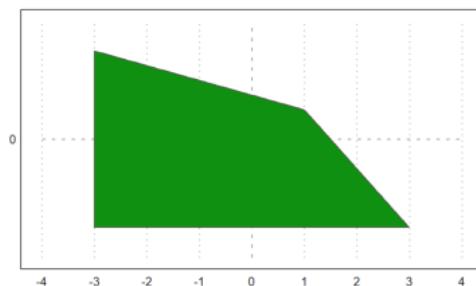
Contoh lainnya adalah segi empat, yang kita buat dengan 7 titik pada lingkaran satuan.

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



Berikut ini 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):
```

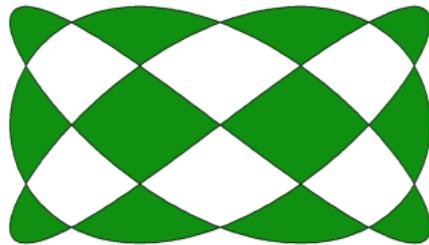


Poin utama dari bahasa matriks adalah memungkinkan untuk menghasilkan tabel fungsi dengan mudah.

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

Kami sekarang memiliki vektor x dan y nilai. plot2d() dapat memplot nilai-nilai ini sebagai kurva yang menghubungkan titik-titik. Plotnya bisa diisi. Pada kasus ini ini menghasilkan hasil yang bagus karena aturan lilitan, yang digunakan untuk isi.

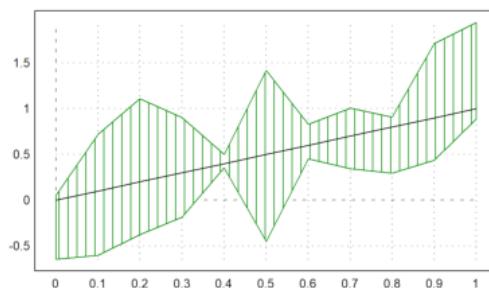
```
>plot2d(x,y,<grid,<frame,>filled):
```



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

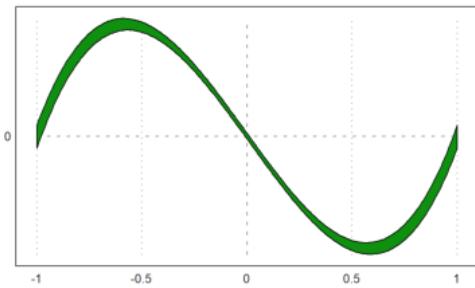
Ini dapat berguna untuk memplot kesalahan perhitungan. Tapi itu bisa juga digunakan untuk memplot kesalahan statistik.

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



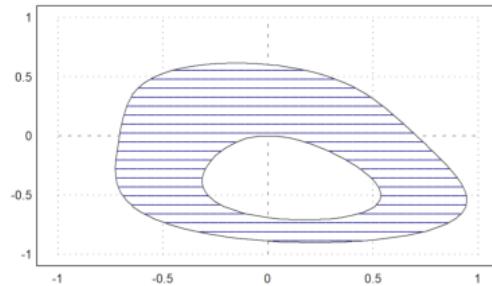
Jika x adalah vektor yang diurutkan, dan y adalah vektor interval, maka plot2d akan memplot rentang interval yang terisi dalam bidang. Gaya isian sama dengan gaya poligon.

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



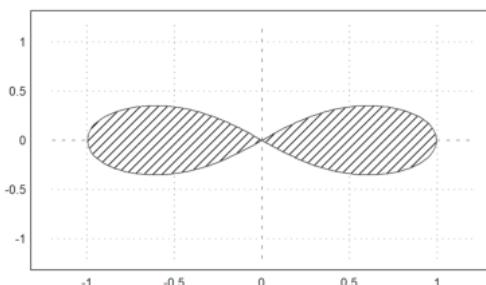
Dimungkinkan untuk mengisi wilayah nilai untuk fungsi tertentu. Untuk ini, level harus berupa matriks 2xn. Baris pertama adalah batas bawah dan baris kedua berisi batas atas.

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

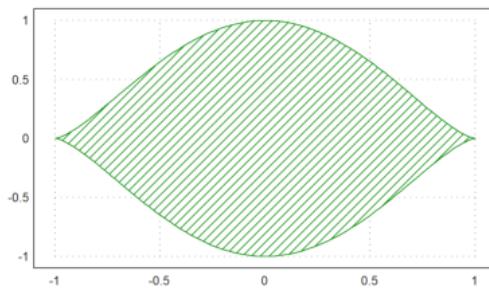


Kami juga dapat mengisi rentang nilai seperti
lateks: $-1 \leq (x^2+y^2)^2-x^2+y^2 \leq 0$.

```
>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

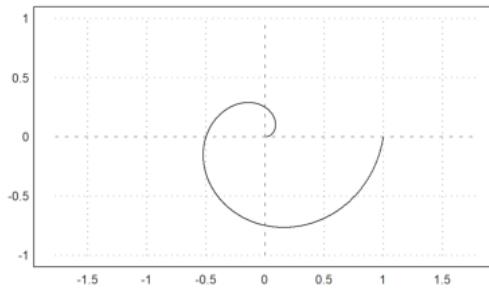
Nilai-x tidak perlu diurutkan. (x,y) hanya menggambarkan kurva. Jika x diurutkan, kurva tersebut merupakan grafik fungsi.

Dalam contoh berikut, kami memplot spiral

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

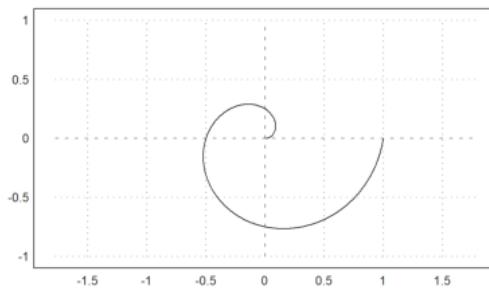
Kita perlu menggunakan banyak titik untuk tampilan yang halus atau fungsi adaptif() untuk mengevaluasi ekspresi (lihat fungsi adaptif() untuk lebih jelasnya).

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

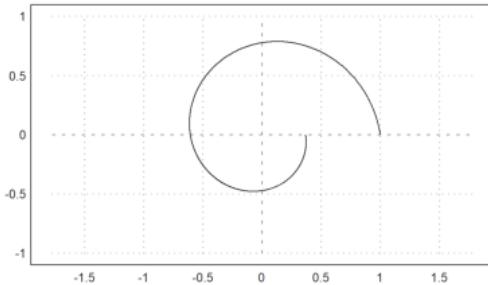


Atau, dimungkinkan untuk menggunakan dua ekspresi untuk kurva. Berikut ini plot kurva yang sama seperti di atas.

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

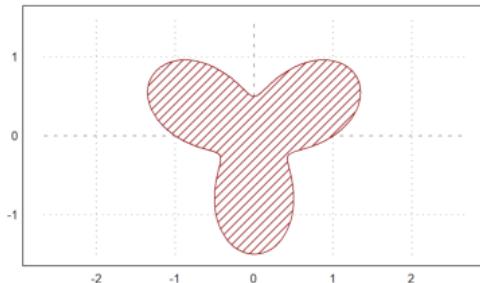


```
>t=linspace(0,1,1000); r=exp(-t); x=r*cos(2pi*t); y=r*sin(2pi*t);
>plot2d(x,y,r=1):
```



Dalam contoh berikutnya, kami memplot kurva
 lateks: $\gamma(t) = (r(t) \cos(t), r(t) \sin(t))$
 dengan
 lateks: $r(t) = 1 + \frac{\sin(3t)}{2}$.

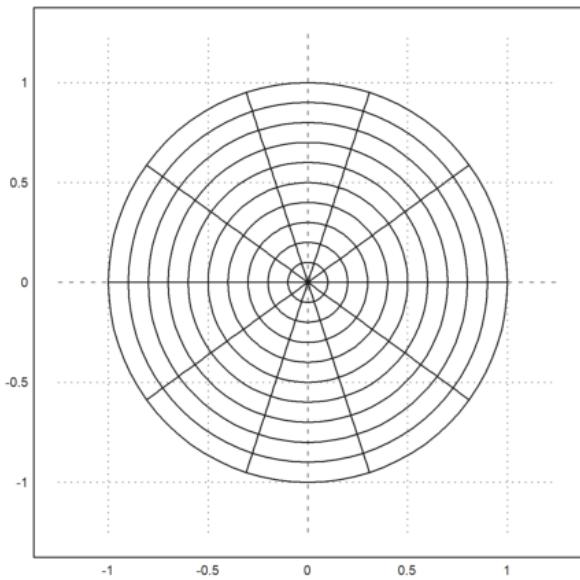
```
>t=linspace(0,2pi,1000); r=1+sin(3*t)/2; x=r*cos(t); y=r*sin(t); ...
>plot2d(x,y,>filled,fillcolor=red,style="/"',r=1.5):
```



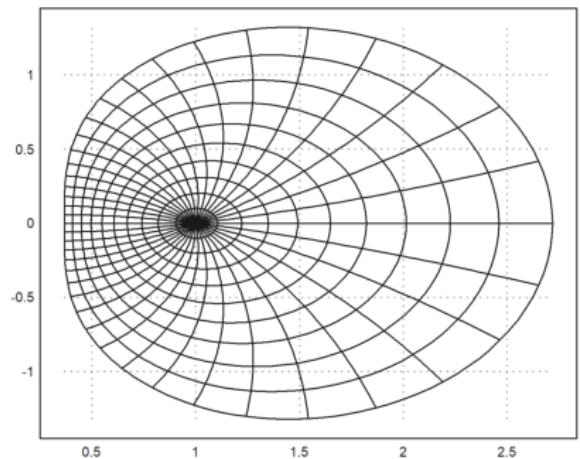
Menggambar Grafik Bilangan Kompleks

Array bilangan kompleks juga dapat diplot. Kemudian titik-titik grid akan terhubung. Jika sejumlah garis kisi ditentukan (atau vektor garis kisi 1×2) dalam argumen cgrid, hanya garis kisi tersebut yang terlihat. Matriks bilangan kompleks akan secara otomatis diplot sebagai kisi di bidang kompleks.
 Dalam contoh berikut, kami memplot gambar lingkaran satuan di bawah fungsi eksponensial. Parameter cgrid menyembunyikan beberapa kurva grid.

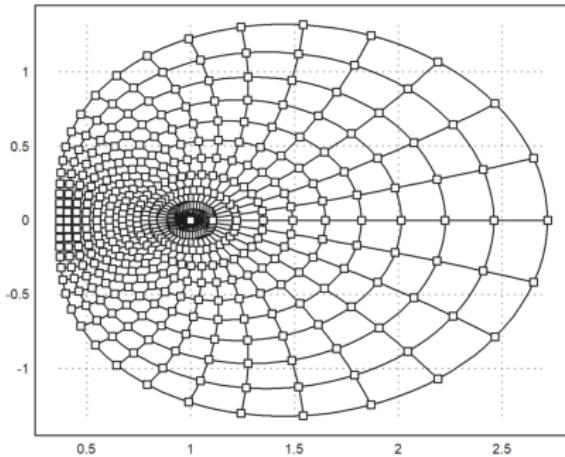
```
>aspect(); r=linspace(0,1,50); a=linspace(0,2pi,80)'; z=r*exp(I*a);...
>plot2d(z,a=-1.25,b=1.25,c=-1.25,d=1.25,cgrid=10):
```



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



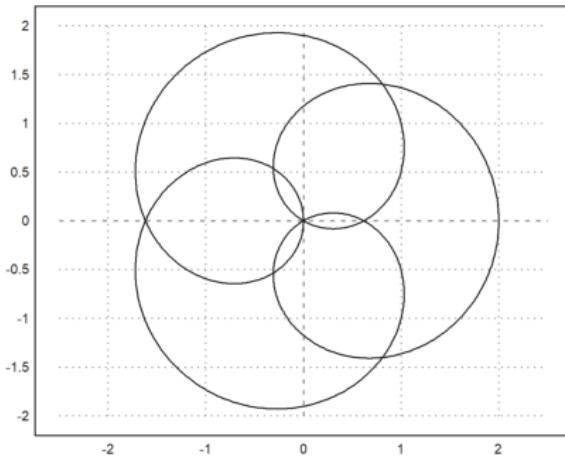
```
>r=linspace(0,1,10); a=linspace(0,2pi,40)'; z=r*exp(I*a);
>plot2d(exp(z),>points,>add):
```



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

Dalam contoh, kami memplot lingkaran satuan dengan
lateks: $\gamma(t) = e^{it}$

```
>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.

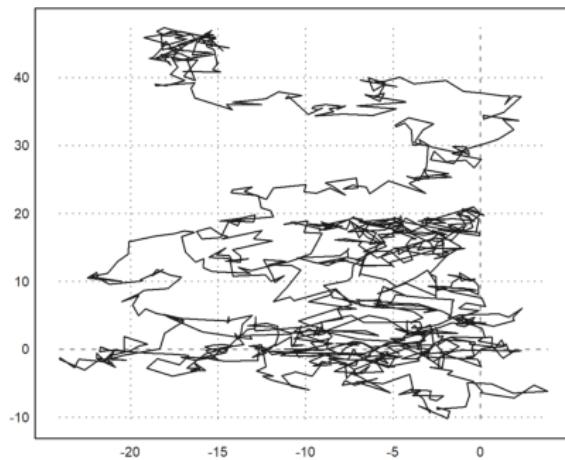
Jumlah kumulatif dari nilai terdistribusi 0-1-normal menghasilkan jalan acak.

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

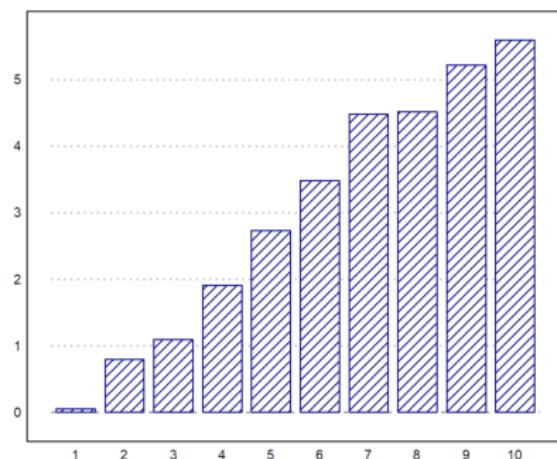


Menggunakan dua baris menunjukkan jalan dalam dua dimensi.

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

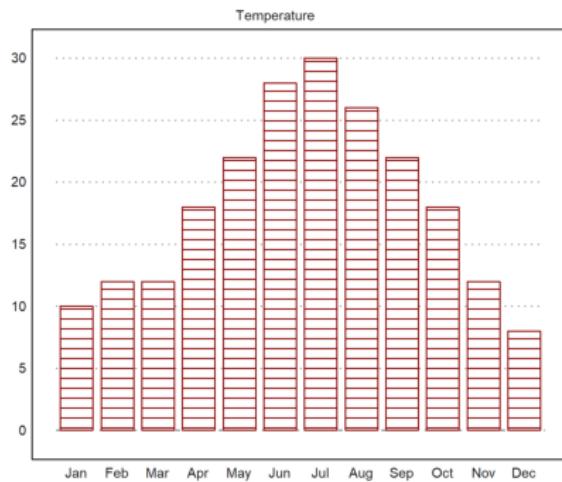


```
>columnsplot(cumsum(random(10)),style="/",color=blue):
```

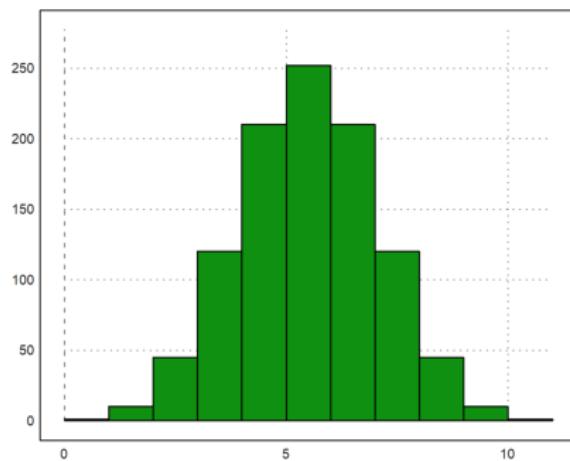


Itu juga dapat menampilkan string sebagai label.

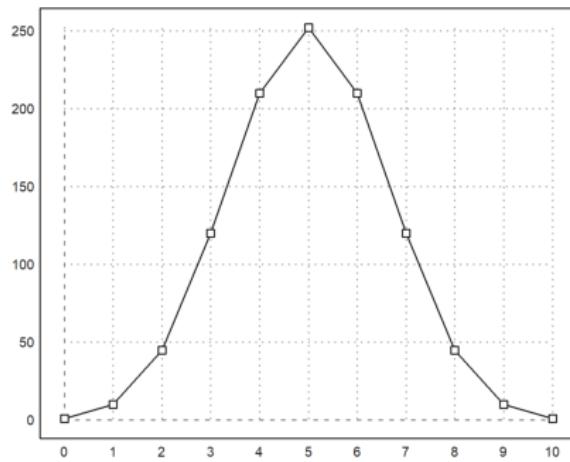
```
>months=["Jan", "Feb", "Mar", "Apr", "May", "Jun", ...
> "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];
>values=[10,12,12,18,22,28,30,26,22,18,12,8];
>columnsplot(values,lab=months,color=red,style="-");
>title("Temperature"):
```



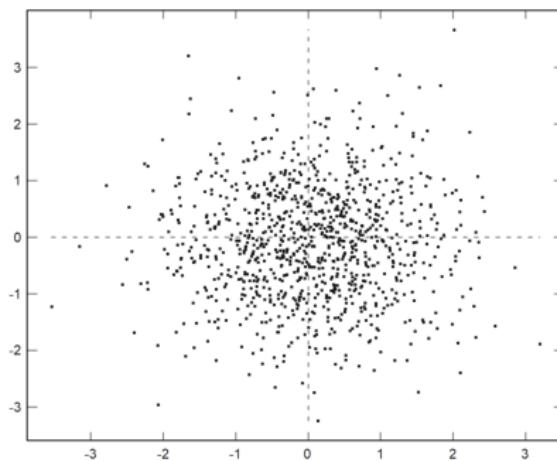
```
>k=0:10;
>plot2d(k,bin(10,k),>bar):
```



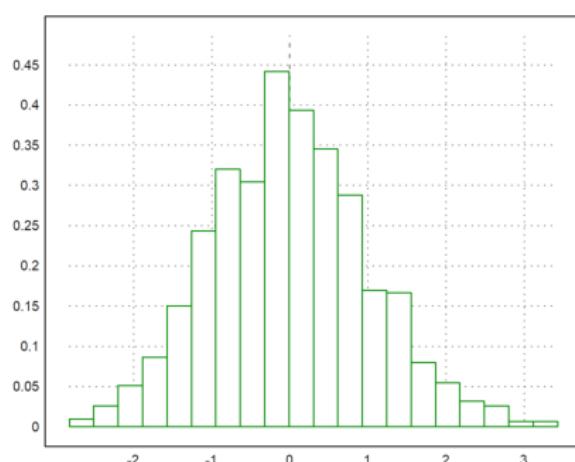
```
>plot2d(k,bin(10,k)); plot2d(k,bin(10,k),>points,>add):
```



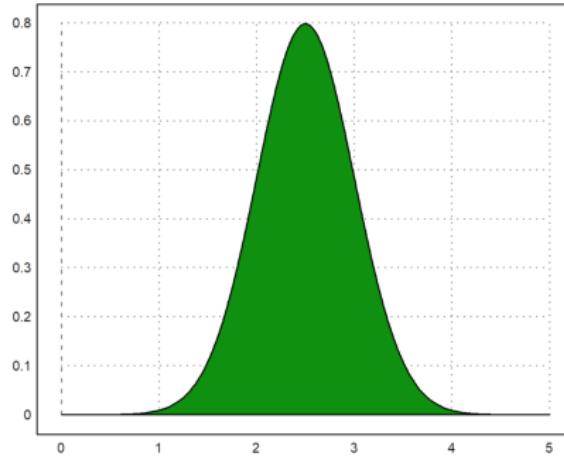
```
>plot2d(normal(1000),normal(1000),>points,grid=6,style=".."):
```



```
>plot2d(normal(1,1000),>distribution,style="O"):
```

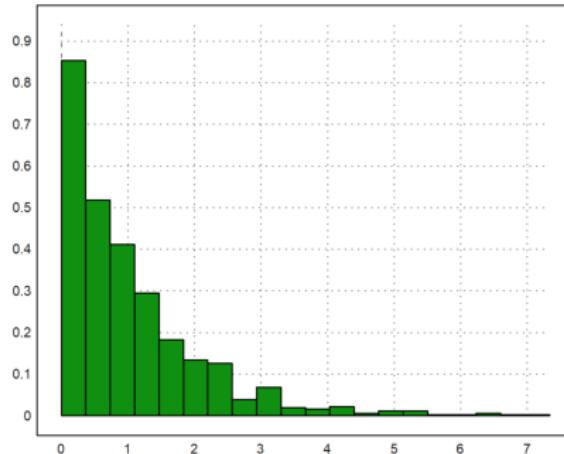


```
>plot2d("qnormal",0,5;2.5,0.5,>filled):
```



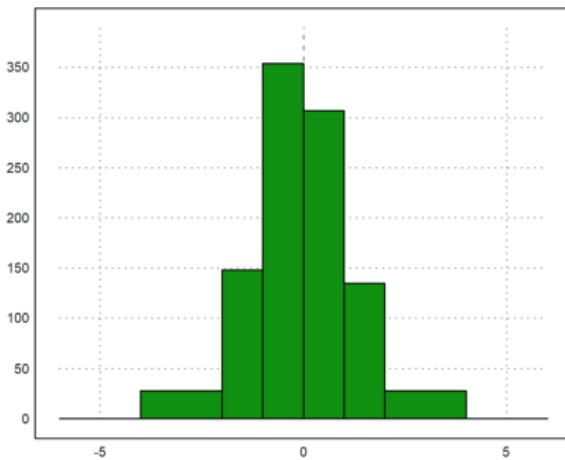
Untuk memplot distribusi statistik eksperimental, Anda dapat menggunakan distribution=n dengan plot2d.

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



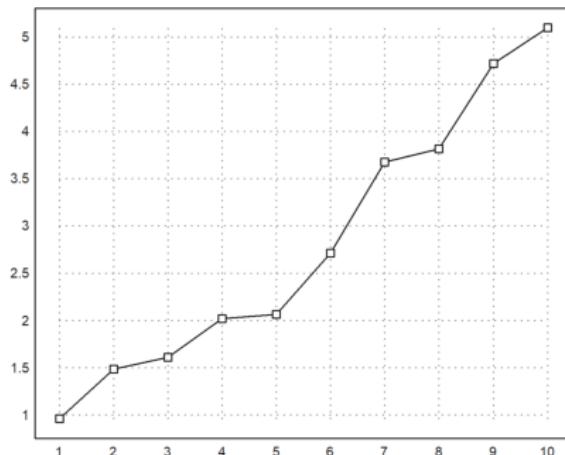
Atau Anda dapat menghitung distribusi dari data dan memplot hasilnya dengan >bar di plot3d, atau dengan plot kolom.

```
>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):
```

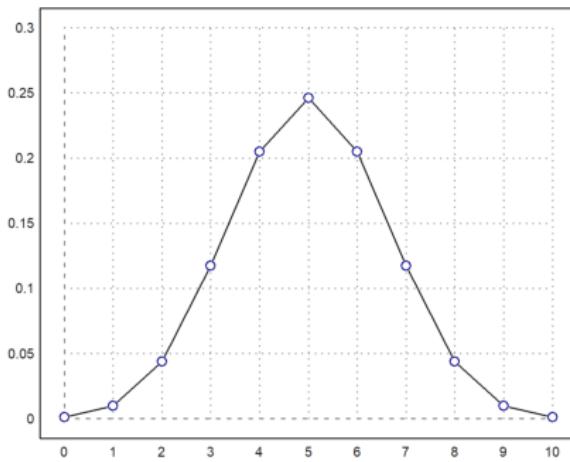


Fungsi statplot() menyetel gaya dengan string sederhana.

```
>statplot(1:10,cumsum(random(10)), "b"):
```



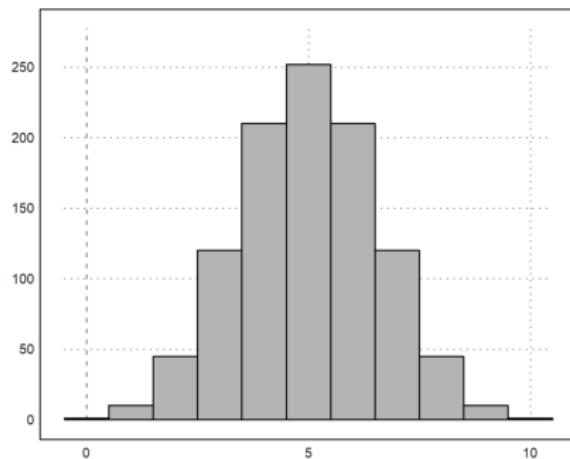
```
>n=10; i=0:n; ...
>plot2d(i,bin(n,i)/2^n,a=0,b=10,c=0,d=0.3); ...
>plot2d(i,bin(n,i)/2^n,points=true,style="ow",add=true,color=blue):
```



Selain itu, data dapat diplot sebagai batang. Dalam hal ini, x harus diurutkan dan satu elemen lebih panjang dari y. Bilah akan memanjang dari $x[i]$ ke $x[i+1]$ dengan nilai $y[i]$. Jika x memiliki ukuran yang sama dengan y, maka akan diperpanjang satu elemen dengan spasi terakhir.

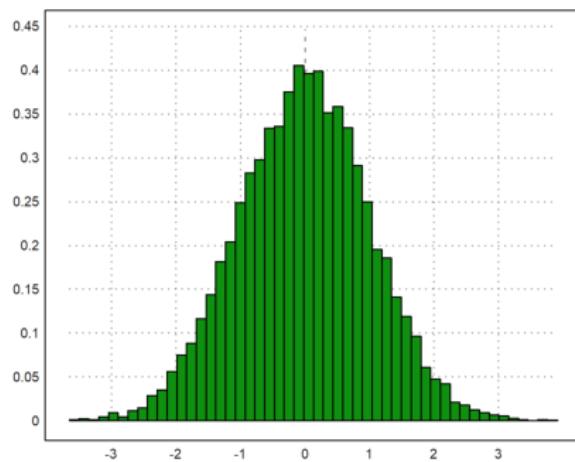
Gaya isian dapat digunakan seperti di atas.

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

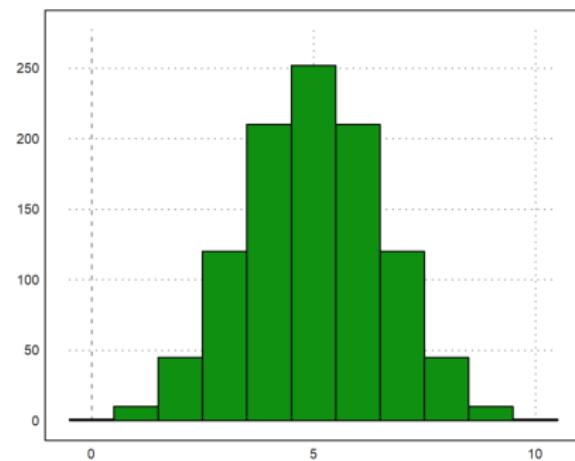


Data untuk plot batang (bar=1) dan histogram (histogram=1) dapat dinyatakan secara eksplisit dalam xv dan yv, atau dapat dihitung dari distribusi empiris dalam xv dengan >distribusi (atau distribusi=n). Histogram nilai xv akan dihitung secara otomatis dengan >histogram. Jika >genap ditentukan, nilai xv akan dihitung dalam interval bilangan bulat.

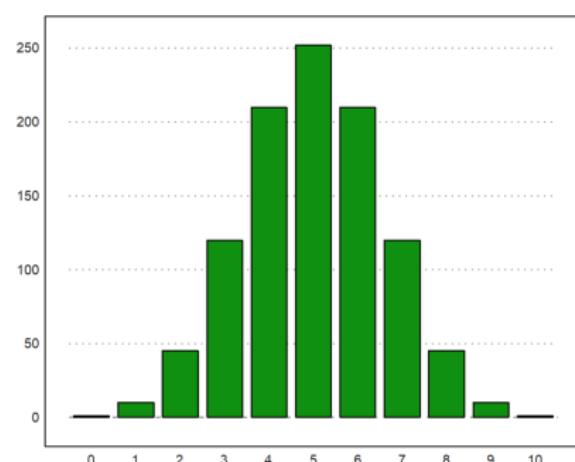
```
>plot2d(normal(10000),distribution=50):
```



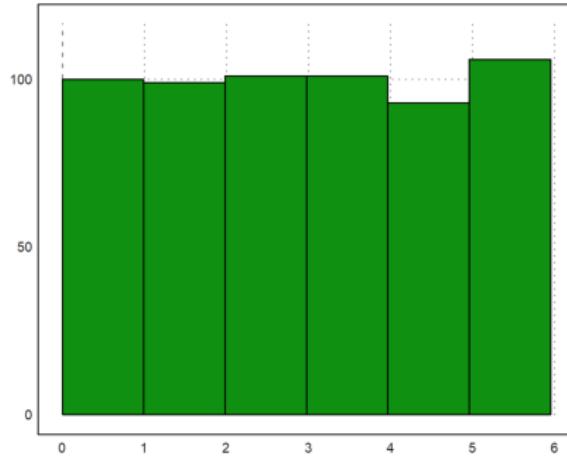
```
>k=0:10; m=bin(10,k); x=(0:11)-0.5; plot2d(x,m,>bar):
```



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

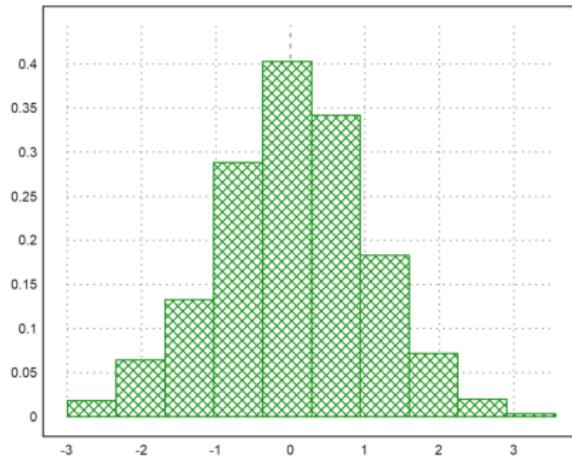


```
>plot2d(random(600)*6,histogram=6):
```



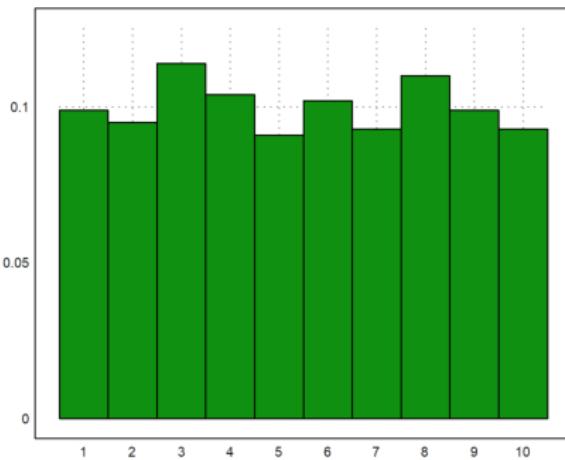
Untuk distribusi, ada parameter `distribution=n`, yang menghitung nilai secara otomatis dan mencetak distribusi relatif dengan n sub-interval.

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



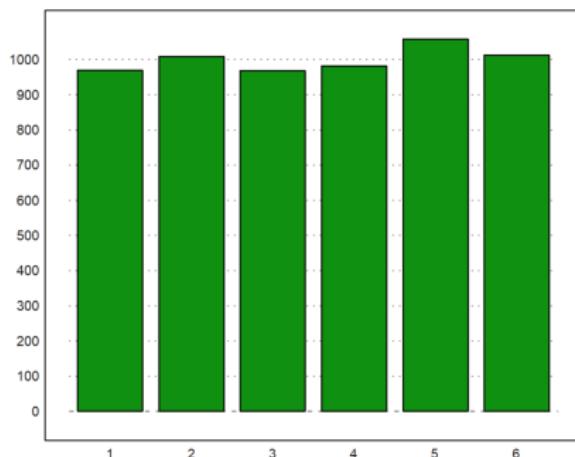
Dengan parameter `even=true`, ini akan menggunakan interval integer.

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

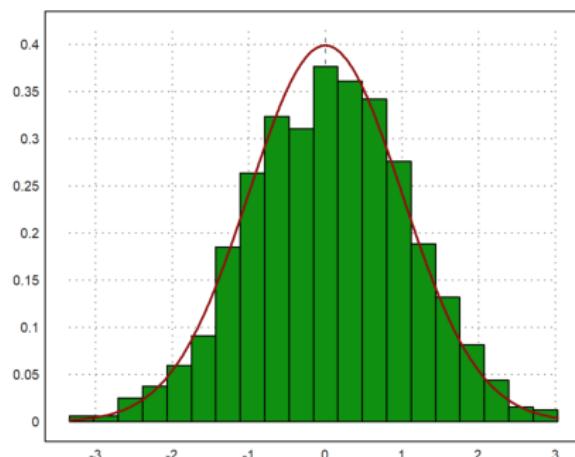


Perhatikan bahwa ada banyak plot statistik, yang mungkin berguna. Silahkan lihat tutorial tentang statistik.

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

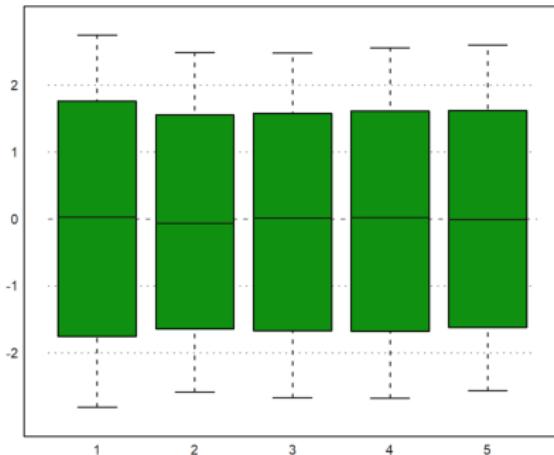


```
>plot2d(normal(1,1000),>distribution); ...
> plot2d("qnormal(x)",color=red,thickness=2,>add) :
```



Ada juga banyak plot khusus untuk statistik. Boxplot menunjukkan kuartil dari distribusi ini dan banyak outlier. Menurut definisi, outlier dalam boxplot adalah data yang melebihi 1,5 kali kisaran 50% tengah plot.

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



Fungsi Implisit

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

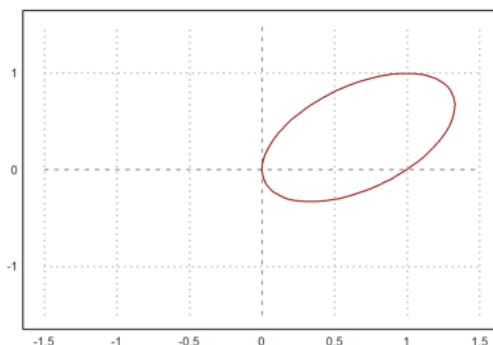
Euler dapat menandai garis level

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

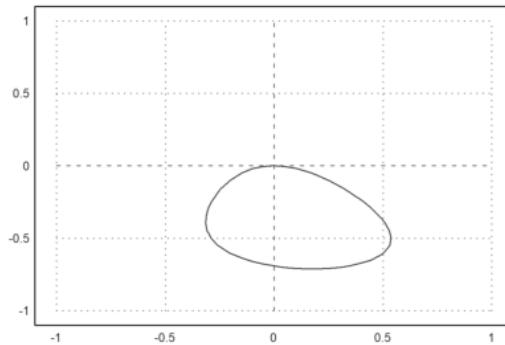
dari fungsi apapun.

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

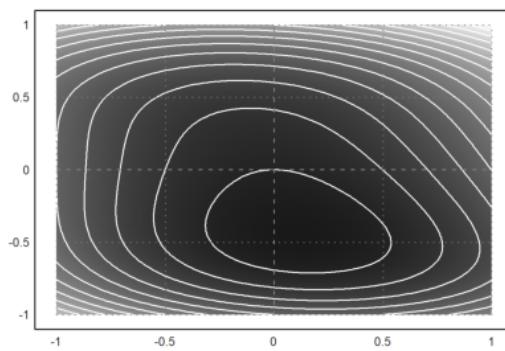
```
>aspect(1.5);
>plot2d("x^2+y^2-x*y-x", r=1.5, level=0, contourcolor=red):
```



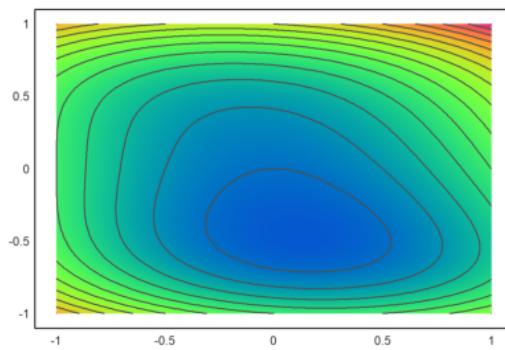
```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr,level=0); // Solutions of f(x,y)=0
```



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

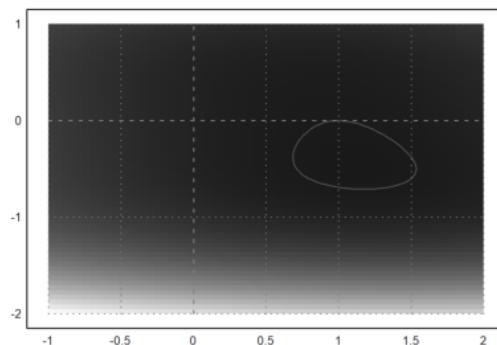


```
>plot2d(expr,level=0:0.5:20,>hue,>spectral,n=200,grid=4); // nicer
```

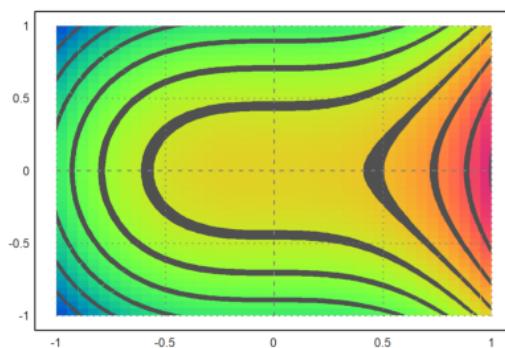


Ini berfungsi untuk plot data juga. Tetapi Anda harus menentukan rentangnya untuk label sumbu.

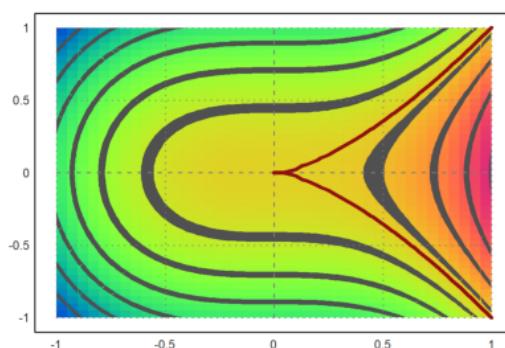
```
>x=-2:0.05:1; y=x'; z=expr(x,y);
>plot2d(z,level=0,a=-1,b=2,c=-2,d=1,>hue):
```



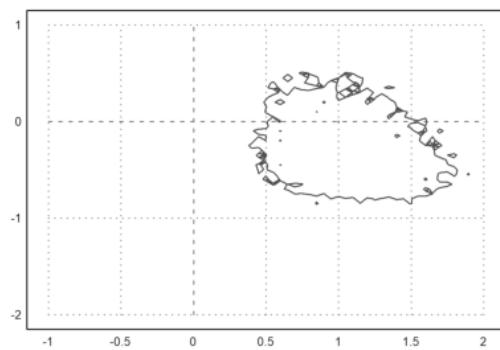
```
>plot2d("x^3-y^2",>contour,>hue,>spectral):
```



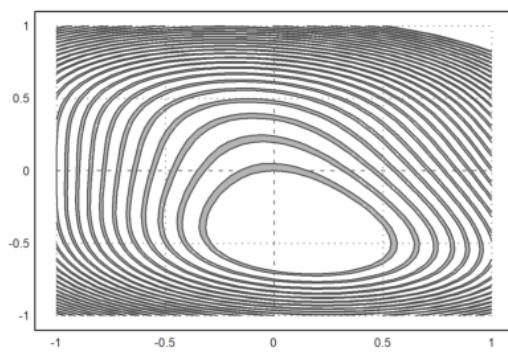
```
>plot2d("x^3-y^2",level=0,contourwidth=3,>add,contourcolor=red):
```



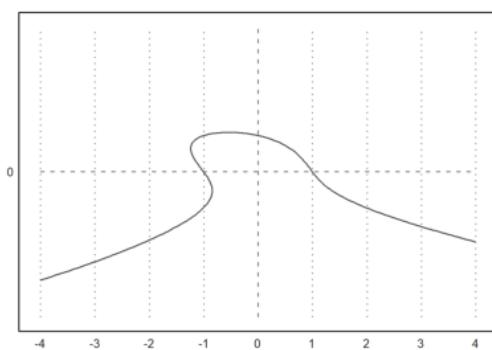
```
>z=z+normal(size(z))*0.2;
>plot2d(z,level=0.5,a=-1,b=2,c=-2,d=1):
```



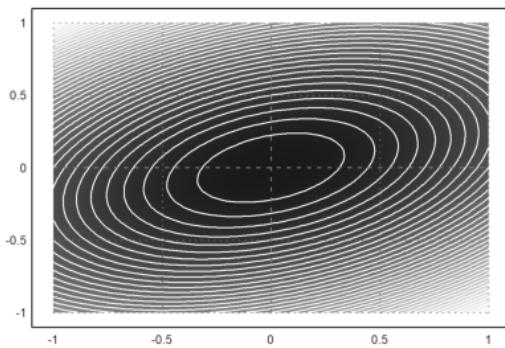
```
>plot2d(expr,level=[0:0.2:5;0.05:0.2:5.05],color=lightgray):
```



```
>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):
```



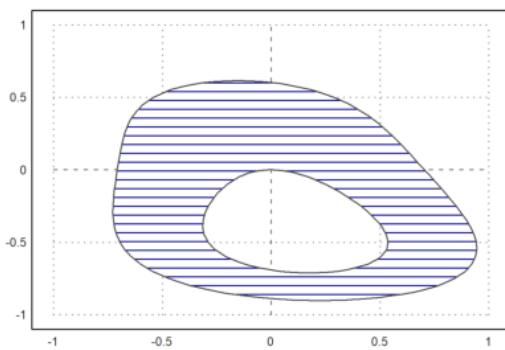
Dimungkinkan juga untuk mengisi set

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

dengan rentang tingkat.

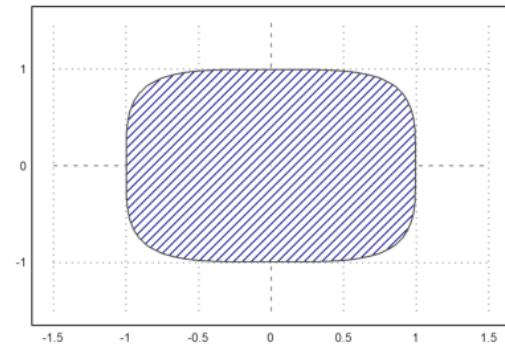
Dimungkinkan untuk mengisi wilayah nilai untuk fungsi tertentu. Untuk ini, level harus berupa matriks 2xn. Baris pertama adalah batas bawah dan baris kedua berisi batas atas.

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

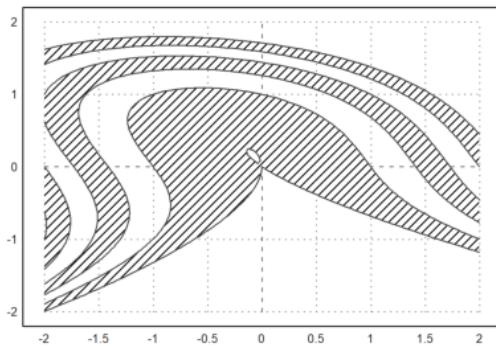


Plot implisit juga dapat menunjukkan rentang level. Kemudian level harus berupa matriks 2xn dari interval level, di mana baris pertama berisi awal dan baris kedua adalah akhir dari setiap interval. Atau, vektor baris sederhana dapat digunakan untuk level, dan parameter dl memperluas nilai level ke interval.

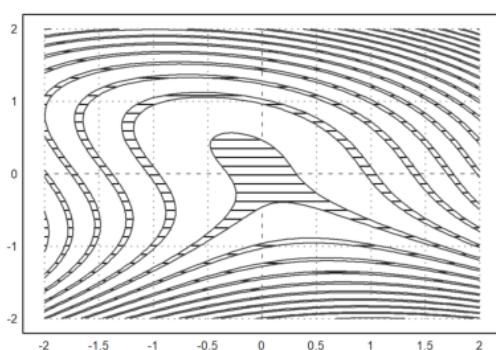
```
>plot2d("x^4+y^4",r=1.5,level=[0;1],color=blue,style="/"):
```



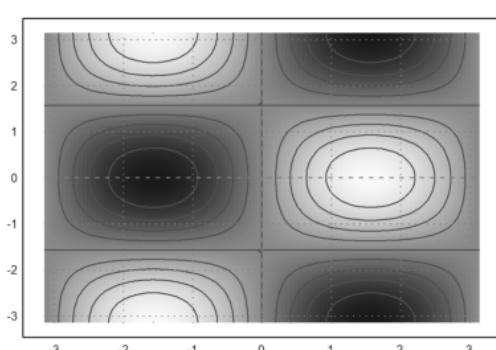
```
>plot2d("x^2+y^3+x*y",level=[0,2,4;1,3,5],style="/",r=2,n=100):
```



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

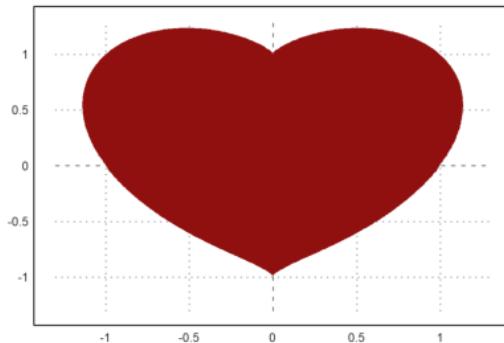


```
>plot2d("sin(x)*cos(y)",r=pi,>hue,>levels,n=100):
```



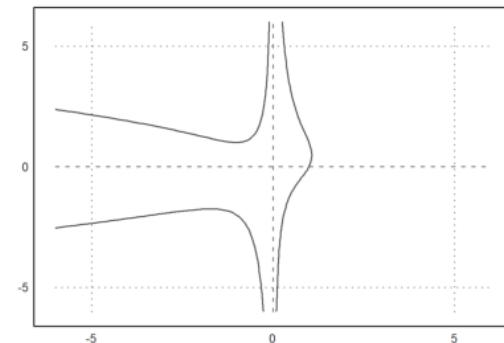
Dimungkinkan juga untuk menandai suatu wilayah
lateks: $a \leq f(x,y) \leq b$.
Ini dilakukan dengan menambahkan level dengan dua baris.

```
>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 lateks: $x^3-xy+x^2y^2=6$

```
>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);
```

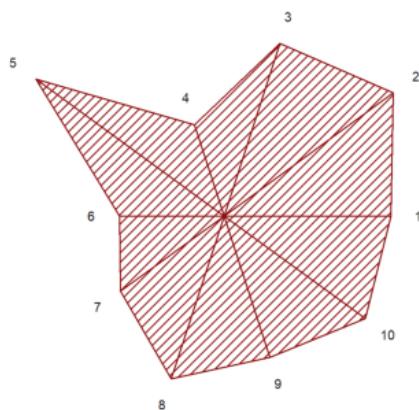
```

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

```

Tidak ada kotak atau sumbu kutu di sini. Selain itu, kami menggunakan jendela penuh untuk plot. Kami memanggil reset sebelum kami menguji plot ini untuk mengembalikan default grafis. Ini tidak perlu, jika Anda yakin plot Anda berhasil.

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



Terkadang, Anda mungkin ingin merencanakan sesuatu yang tidak dapat dilakukan plot2d, tetapi hampir. Dalam fungsi berikut, kami melakukan 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;
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 jendela plot. setplot(a,b,c,d) mengatur jendela ini.

Fungsi wait(0) memaksa plot untuk muncul di jendela grafik. Jika tidak, menggambar ulang terjadi 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 sembarang tombol untuk menghentikan animasi ini.

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

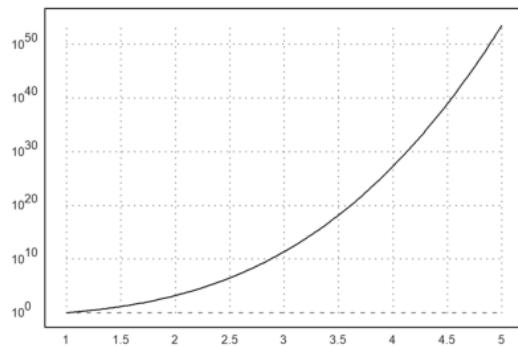
Plot Logaritmik

EMT menggunakan parameter "logplot" untuk skala logaritmik.

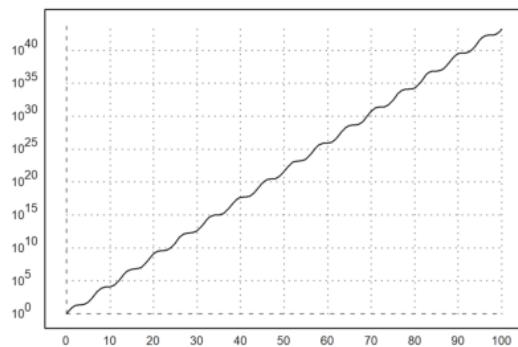
Plot logaritma dapat diplot baik menggunakan skala logaritmik dalam y dengan logplot=1, atau menggunakan skala logaritmik dalam x dan y dengan logplot=2, atau dalam x dengan logplot=3.

- logplot=1: y-logaritma
- logplot=2: x-y-logaritma
- logplot=3: x-logaritma

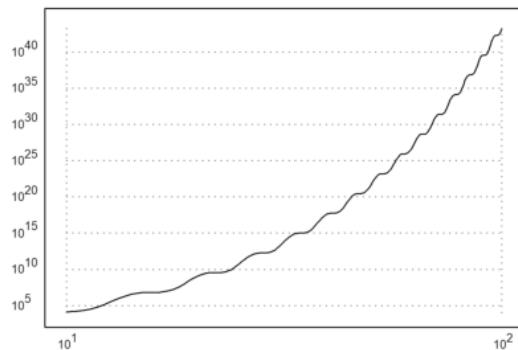
```
>plot2d("exp(x^3-x)*x^2",1,5,logplot=1):
```



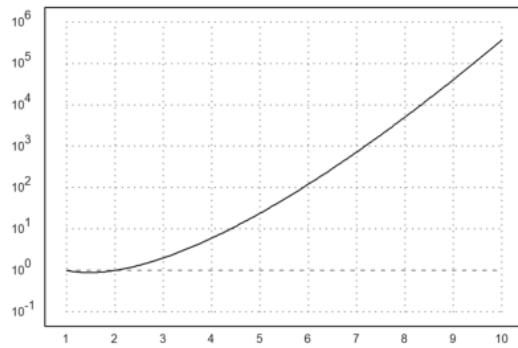
```
>plot2d("exp(x+sin(x))",0,100,logplot=1):
```



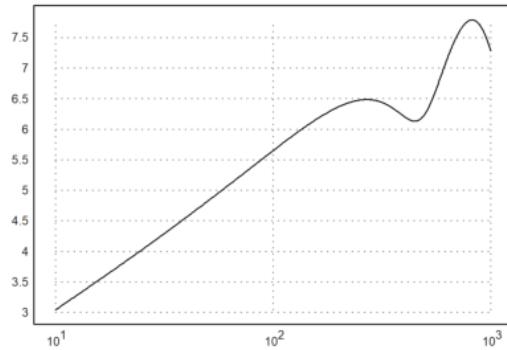
```
>plot2d("exp(x+sin(x))",10,100,logplot=2):
```



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

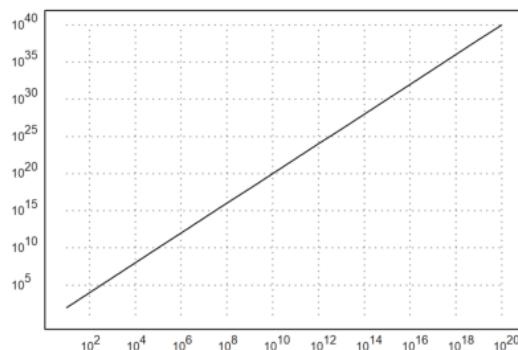


```
>plot2d("log(x*(2+sin(x/100)))",10,1000,logplot=3):
```

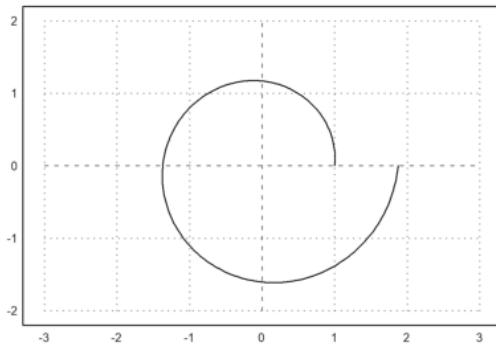


Ini juga berfungsi dengan plot data.

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

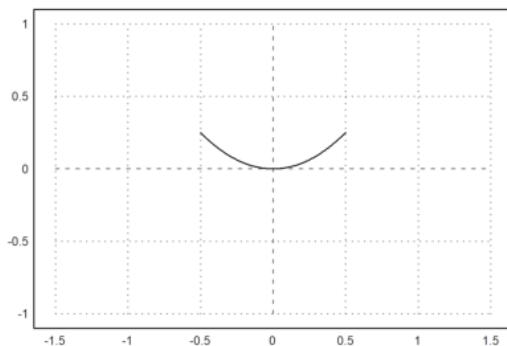


```
>a=0.1; ...
> plot2d("exp(a*x)*cos(x)", "exp(a*x)*sin(x)", r=2, xmin=0, xmax=2*pi):
```



spiral logaritmic

```
>plot2d("x^2", xmin=-1/2, xmax=1/2, r=1):
```



Program Bumi di Bawah ini

```
>load "spherical.e";
>EI=[rad(48,53.371),rad(11,11.330)]
```

```
[0.853283, 0.195282]
```

```
>sposprint(EI)
```

```
N 48°53.371' E 11°11.330'
```

```
>IN=[rad(48,45.854),rad(11,25.055)]; ND=[rad(48,44.248),rad(11,10.712)];
>sposprint(IN), sposprint(ND),
```

```
N 48°45.854' E 11°25.055'  
N 48°44.248' E 11°10.712'
```

```
>br=svector(EI,IN); deg(br[1]), br[2]*rearth(48°)->km
```

```
129.671788237  
21.766496146
```

```
>br=svector(EI,IN); deg(br[1]), br[2]*rearth(48°)->km
```

```
129.671788237  
21.766496146
```

```
>sdegprint(esdir(EI,IN))
```

```
129.67°
```

```
>asum=sangle(IN,EI,ND)+sangle(EI,IN,ND)+sangle(IN,ND,EI); deg(asum)
```

```
180.000207484
```

```
>(asum-pi)*rearth(48°)^2->" km^2"
```

```
146.771319034 km^2
```

```
>esarea(IN,EI,ND)->" km^2"
```

```
146.757696278 km^2
```

```
>v=svector(EI,IN); sposprint(saddvector(EI,v)), sposprint(IN),
```

```
N 48°45.854' E 11°25.055'  
N 48°45.854' E 11°25.055'
```

```
>sposprint(esadd(EI,esdir(EI,IN),esdist(EI,IN))), sposprint(IN),
```

```
N 48°45.854' E 11°25.055'  
N 48°45.854' E 11°25.055'
```

```
>Berlin=[52.5164°,13.3777°]; Lissabon=[38.692668°,-9.177944°];  
>sposprint(Berlin), sposprint(Lissabon)
```

```
N 52°30.984' E 13°22.662'  
N 38°41.560' W 9°10.677'
```

```
>esdist(Lissabon,Berlin)->" km"
```

2313.67485976 km

```
>degsprint(esdir(Lissabon,Berlin))
```

41°3'5.08''

```
>sposprint(esadd(Berlin,esdir(Berlin,Lissabon),esdist(Berlin,Lissabon)))
```

N 38°41.557' W 9°10.680'

```
>sposprint(Lissabon),
```

N 38°41.560' W 9°10.677'

```
>dist=esdist(Berlin,Lissabon); hd=esdir(Berlin,Lissabon);  
>p=Berlin; loop 1 to 10; p=esadd(p,hd,dist/10); end;  
>sposprint(p), skmpprint(esdist(p,Lissabon))
```

N 41°0.566' W 12°5.348'

357.810km

```
>P1=[30°,10°]; P2=[30°,50°];  
>sdegprint(esdir(P1,P2))
```

79.69°

```
>p=P1; dist=esdist(P1,P2); ...  
>loop 1 to 10; dir=esdir(p,P2); sdegprint(dir), p=esadd(p,dir,dist/10); end;
```

79.69°
81.67°
83.71°
85.78°
87.89°
90.00°
92.12°
94.22°
96.29°
98.33°

```
>skmpprint(esdist(p,P2))
```

0.203km

```
>p=Berlin; dist=esdist(Berlin,Lissabon); ...
> loop 1 to 100; p=esadd(p,esdir(p,Lissabon),dist/100); end;
>skmpprint(esdist(p,Lissabon))
```

0.057km

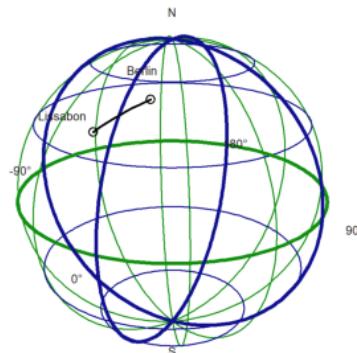
```
>load spherical; v=navigate(Berlin,Lissabon,10); ...
> loop 1 to rows(v); sposprint(v[#]), end;
```

```
N 52°30.984' E 13°22.662'
N 51°21.596' E 10°34.058'
N 50°8.382' E 7°53.992'
N 48°51.693' E 5°22.083'
N 47°31.853' E 2°57.902'
N 46°9.158' E 0°40.991'
N 44°43.881' W 1°29.117'
N 43°16.269' W 3°32.890'
N 41°46.547' W 5°30.784'
N 40°14.916' W 7°23.240'
N 38°41.560' W 9°10.677'
```

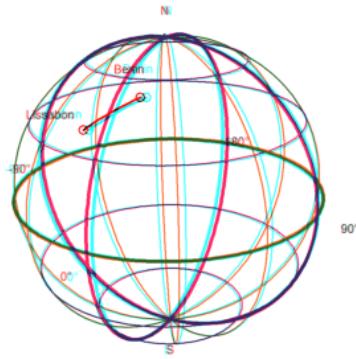
```
>function testplot ...
```

```
useglobal;
plotearth;
plotpos(Berlin,"Berlin"); plotpos(Lissabon,"Lissabon");
plotposline(v);
endfunction
```

```
>plot3d("testplot",>own,>user,zoom=4):
```



```
>plot3d("testplot",own=1,anaglyph=1,zoom=4):
```



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 .

BAB 3

KB PEKAN 5: MENGGUNAKAN EMT UNTUK MENGAMBAR GRAFIK 3 DIMENSI (3D)

[a4paper,10pt]article eumat

Menggambar Plot 3D dengan EMT

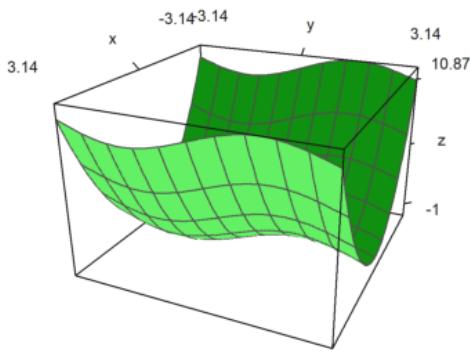
This is an introduction to 3D plots in Euler. We need a 3D plot to visualize a function of two variables. Euler draws such functions using a sorting algorithm to hide parts in the background. In general, Euler uses a central projection. The default is from the positive x-y quadrant towards the origin $x=y=z=0$, but angle=0° looks from into the direction of the y-axis. The view angle and height can be changed.

Euler can plot

- surfaces with shading and level lines or level ranges,
- clouds of points,
- parametric curves,
- implicit surfaces.

A 3D plot of a function uses plot3d. The easiest way is to plot an expression in x and y. The parameter r set the range of the plot around (0,0).

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



Functions of two Variables

For the graph of a function, use

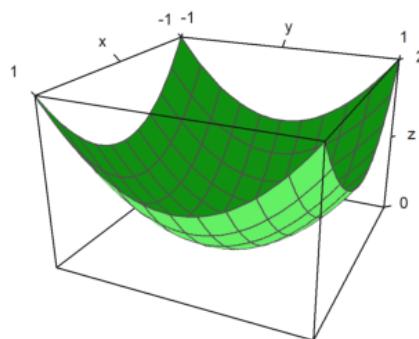
- a simple expression in x and y ,
- the name of a function of two variables,
- or data matrices.

The default is a filled wire grid with different colors on both sides. Note that the default number of grid intervals is 10, but the plot uses the default number of 40x40 rectangles to construct the surface. This can be changed.

- $n=40, n=[40,40]$: number of grid lines in each direction
- $grid=10, grid=[10,10]$: number of grid lines in each direction.

We use the default $n=40$ and $grid=10$.

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

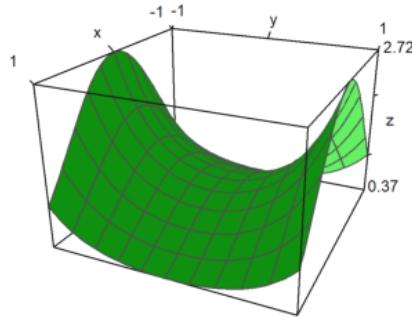


User interaction is possible with the `>user` parameter. The user can press the following keys.

- left,right,up,down: turn the viewing angle
- +,-: zoom in or out
- a: produce an anaglyph (see below)
- l: toggle turning the light source (see below)
- space: reset to default
- return: end interaction

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

Turn with the vector keys (press return to finish)



The plot range for functions can be specified with

- a,b: the x-range
- c,d: the y-range
- r: a symmetric square around (0,0).
- n: number of subintervals for the plot.

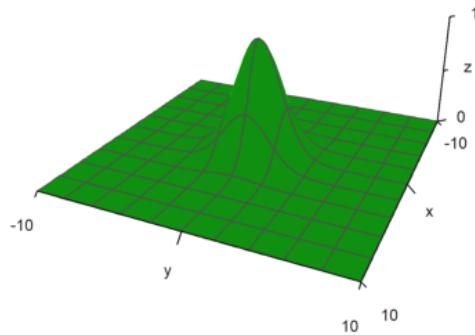
There are some parameters to scale the function or change the look of the graph.

fscale: scales to function values (default is <fscale>).

scale: number or 1x2 vector to scale into x- and y-direction.

frame: type of frame (default 1).

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



The view can be changed in many different ways.

- distance: the viewing distance to the plot.
- zoom: the zoom value.
- angle: the angle to the negative y-axis in radians.
- height: the height of the view in radians.

The default values can be inspected or changed with the function `view()`. It returns the parameters in the order above.

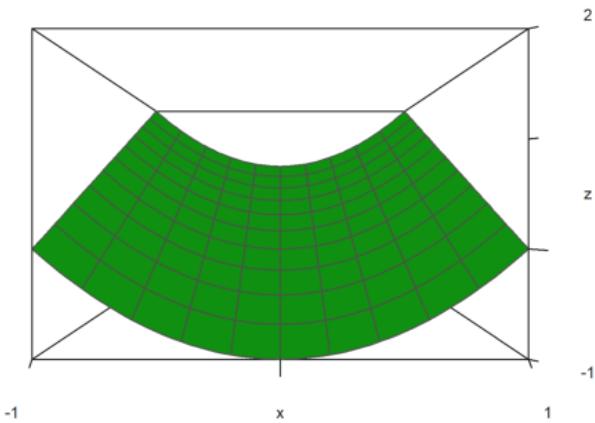
```
>view
```

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

A closer distance needs less zoom. The effect is more like a wide angle lens.

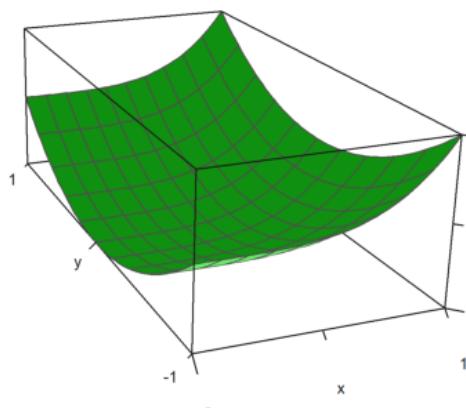
In the following example, `angle=0` and `height=0` look from the negative y-axis. The axis labels for y are hidden in this case.

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



The plot looks always to the center of the plot cube. You can move the center with the `center` parameter.

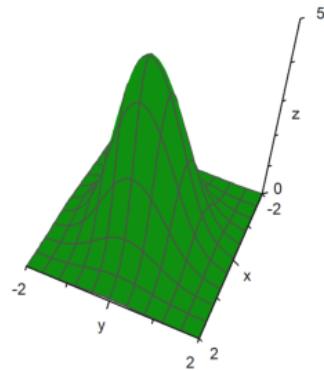
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
> center=[0.4,0,0],zoom=5):
```



The plot is scaled to fit into a unit cube for viewing. So there is no need to change the distance or zoom depending on the size of the plot. The labels refer to the actual size, however.

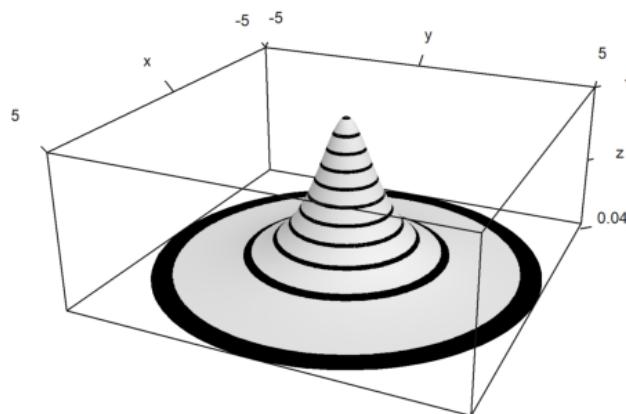
If you turn this off with scale=false, you need to take care, that the plot still fits into the plotting window, by changing the viewing distance or zoom, and moving the center.

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

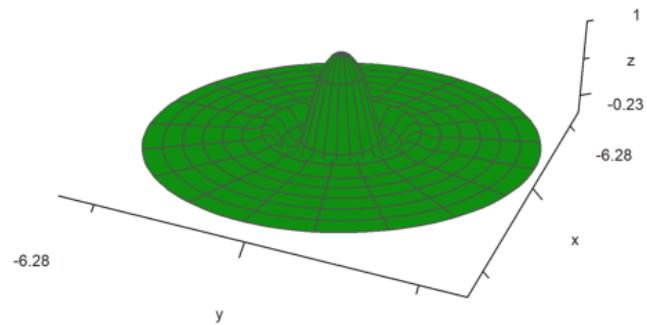


A polar plot is also available. The parameter polar=true draws a polar plot. The function must still be a function of x and y. The parameter "fscale" scales the function with an own scale. Otherwise the function is scaled to fit into a cube.

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



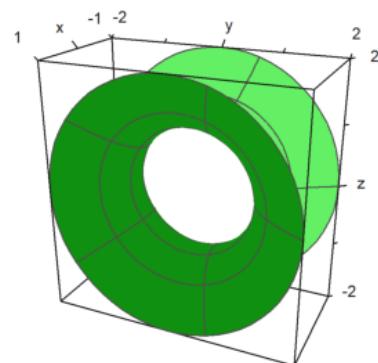
```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=2pi,frame=3,zoom=4):
```



The parameter `rotate` rotates a function in `x` around the `x`-axis.

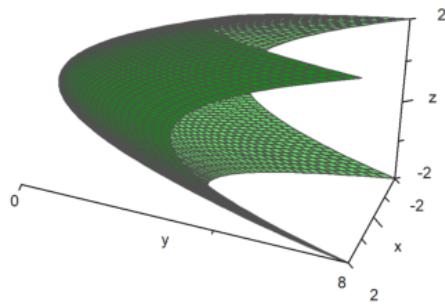
- `rotate=1`: Uses the `x`-axis
- `rotate=2`: Uses the `z`-axis

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



Here is a plot with three functions.

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



Contour Plots

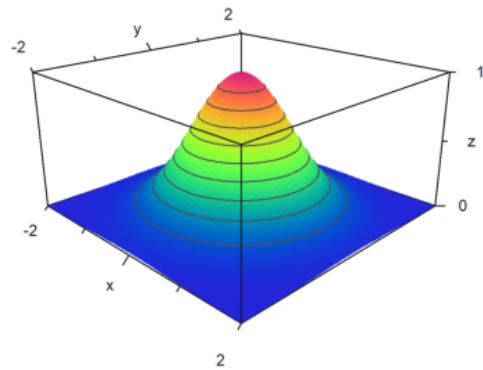
For the plot, Euler adds grid lines. Instead it is possible to use level lines and a one-color hue or a spectral colored hue. Euler can draw the heights of functions on a plot with shading. In all 3D plots Euler can produce red/cyan anaglyphs.

- >hue: Turns on light shading instead of wires.
- >contour: Plots automatic contour lines on a plot.
- level=... (or levels): A vector of values for the contour lines.

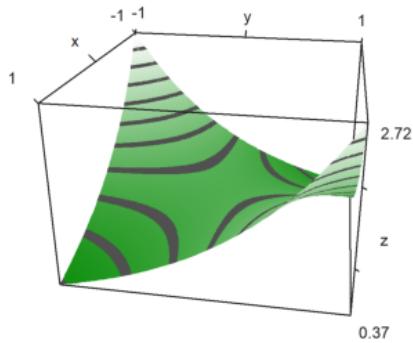
The default is level="auto", which computes some level lines automatically. As you see in the plot, the levels are in fact ranges of levels.

The default style can be changed. For the following contour plot, we use a finer grid fo 100x100 points, scale the function and the plot, and use different angle of view.

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



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

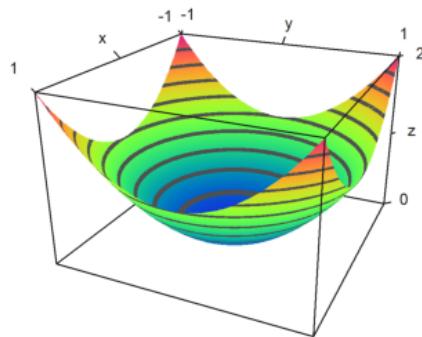


The default shading uses a gray color. But a spectral range of colors is also available.

- >spectral: Used the default spectral scheme
- color=...: Uses special colors or spectral schemes

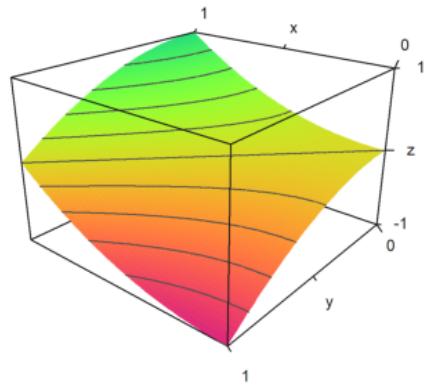
For the following plot, we use the default spectral scheme and increase the number of points to get a very smooth look.

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



Instead of automatic level lines, we can also set values of the level lines. This will produce thin level lines instead of ranges of levels.

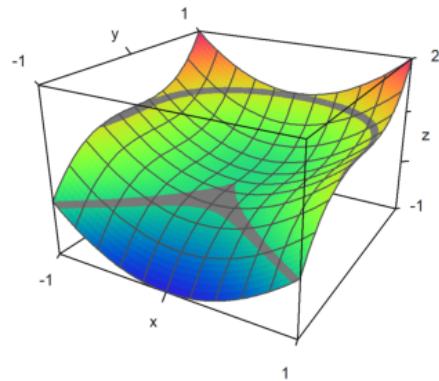
```
>plot3d("x^2-y^2",0,1,0,1,angle=220°,level=-1:0.2:1,color=redgreen):
```



In the following plot, we use two very broad level bands from -0.1 to 1, and from 0.9 to 1. This is entered as a matrix with level bounds as columns.

Moreover, we overlay a grid with 10 intervals in each direction.

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

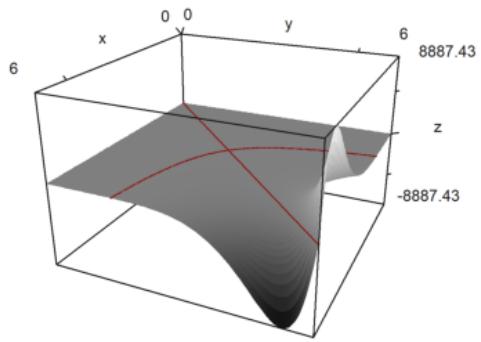


In the following example, we plot the set, where

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

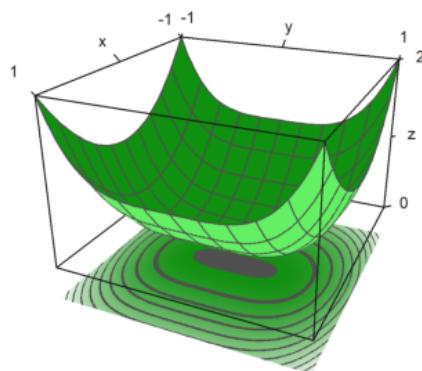
We use a single thin line for the level line.

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



It is possible to show a contour plane below the plot. A color and distance to the plot can be specified.

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



Here are a few more styles. We always turn off the frame, and use various color schemes for the plot and the grid.

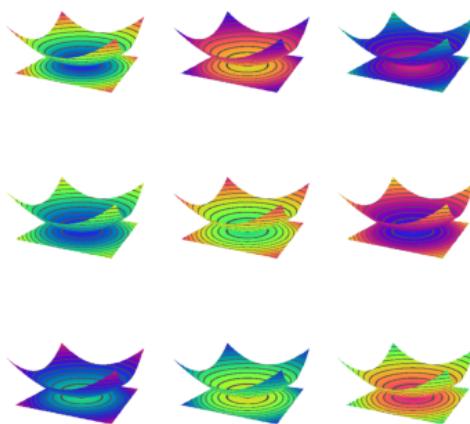
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1); ...
> plot3d(expr,<frame,>cp,cpcolor=spectral); ...
>figure(2); ...
> plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3); ...
> plot3d(expr,<frame,>contour,color=gray,nc=5,cp=3,cpcolor=greenred); ...
>figure(4); ...
> plot3d(expr,<frame,>hue,grid=10,>cp,cpcolor=gray);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



There are some other spectral schemes, numbered from 1 to 9. But you can also use the color=value, where value

- spectral: for a range from blue to red
- white: for a fainter range
- yellowblue,purplegreen,blueyellow,greenred
- blueyellow, greenpurple,yellowblue,redgreen

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



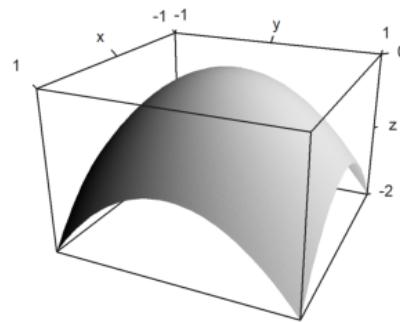
The light source can be changed with l and the cursor keys during the user interaction. It can also be set with parameters.

- light: a direction for the light
- amb: ambient light between 0 and 1

Note that the program does not make a difference between the sides of the plot. There are no shadows. For this you would need Povray.

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

Press l and cursor keys (return to exit)



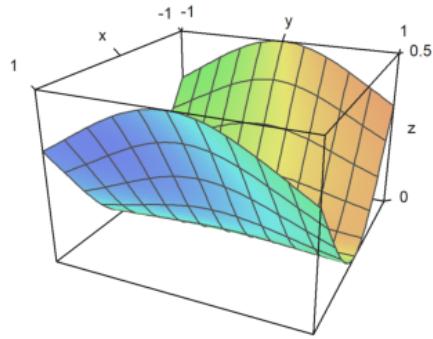
The color parameter changes the color of the surface. The color of the level lines can also be changed.

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



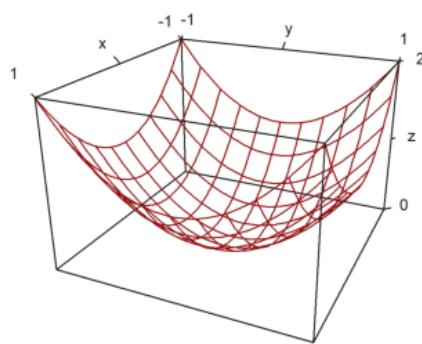
The color 0 gives a special rainbow effect.

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



The surface can also be transparent.

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



Implicit Plots

There are also implicit plots in three dimensions. Euler generates cuts through the objects. The features of `plot3d` include implicit plots. These plots show the zero set of a function in three variables.
The solutions of

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

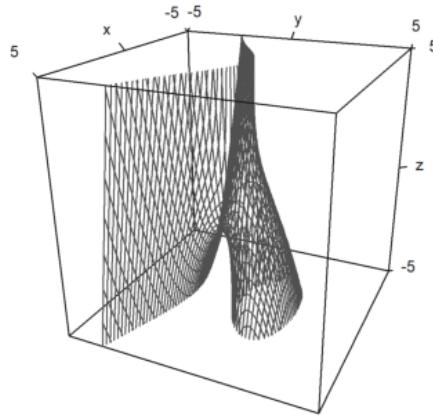
can be visualized in cuts parallel to the x-y-, the x-z- and the y-z-plane.

- `implicit=1`: cut parallel to the y-z-plane
- `implicit=2`: cut parallel to the x-z-plane
- `implicit=4`: cut parallel to the x-y-plane

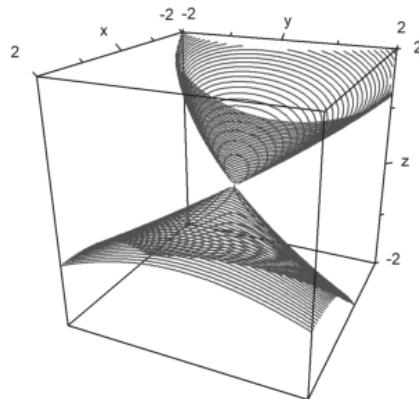
Add these values, if you like. In the example we plot

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

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



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



Plotting 3D Data

Just as plot2d, plot3d accepts data. For 3D objects, you need to provide a matrix of x-, y- and z-values, or three functions or expressions $f_x(x,y)$, $f_y(x,y)$, $f_z(x,y)$.

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

Since x, y, z are matrices, we assume that (t,s) run through a square grid. As a result, you can plot images of rectangles in space.

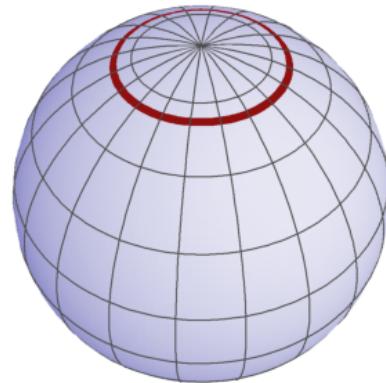
You can use the Euler matrix language to produce the coordinates effectively.

In the following example, we use a vector of t values and a column vector of s values to parameterize the surface of the ball. In the drawing we can mark regions, in our case the polar region.

```

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

```

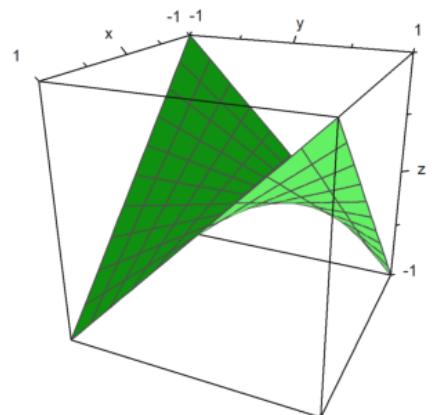


Here is an example, which is the graph of a function.

```

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

```



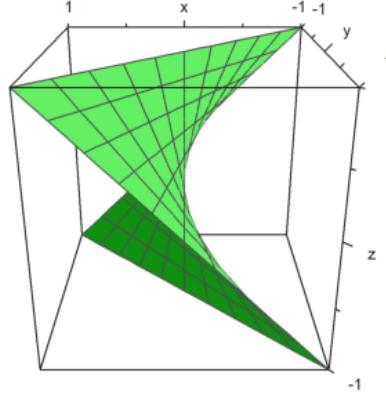
However, we can make all sorts of surfaces. Here is the same surface as a function

$$x = yz$$

```

>plot3d(t*s,t,s,angle=180°,grid=10):

```



With more effort, we can produce many surfaces.

In the following example we make a shaded view of a distorted ball. The usual coordinates for the ball are

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

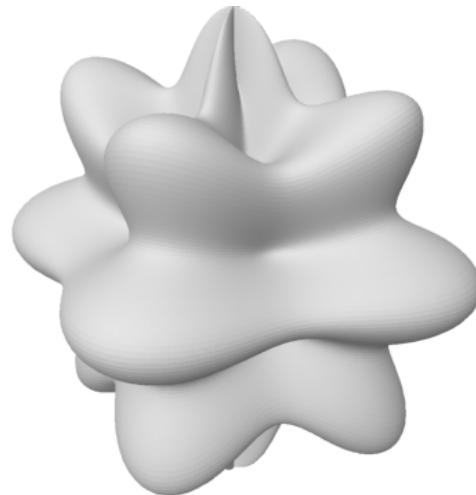
with

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

We distored this with a factor

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

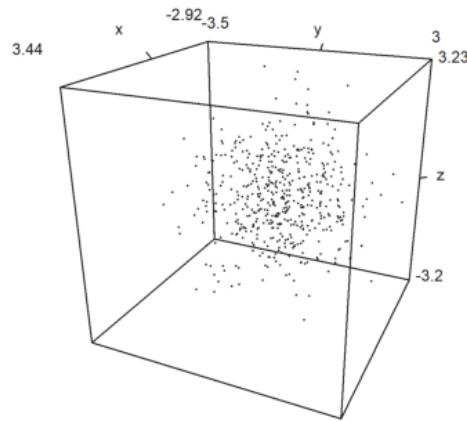
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



Of course, a point cloud is also possible. To plot point data in the space, we need three vectors for the coordinates of the points.

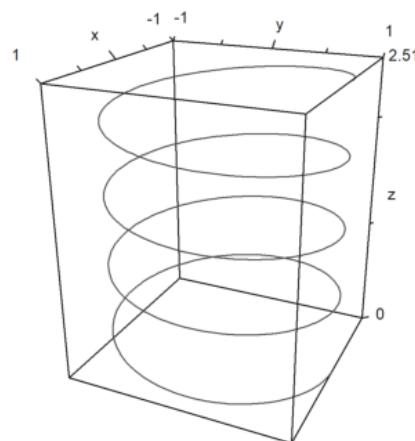
The styles are just as in plot2d with points=true;

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

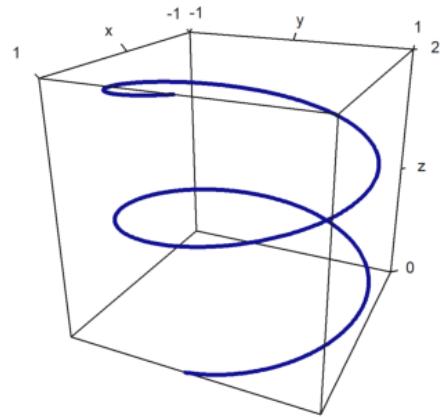


It is also possible to plot a curve in 3D. In this case, it is easier to precompute the points of the curve. For curves in the plane we use a sequence of coordinates and the parameter wire=true.

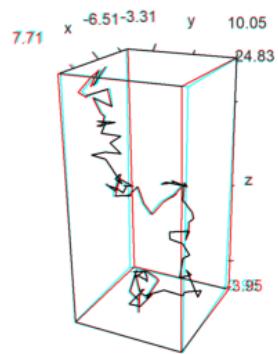
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3):
```



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

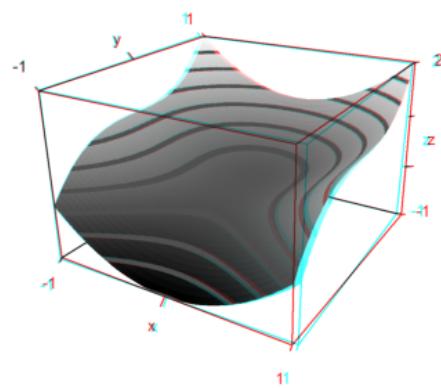


```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire) :
```



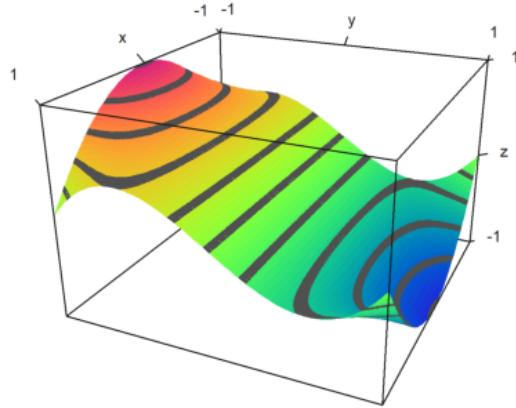
EMT can also plot in anaglyph mode. To view such a plot, you need red/cyan glasses.

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



Often, a spectral color scheme is used for plots. This emphasizes the heights of the function.

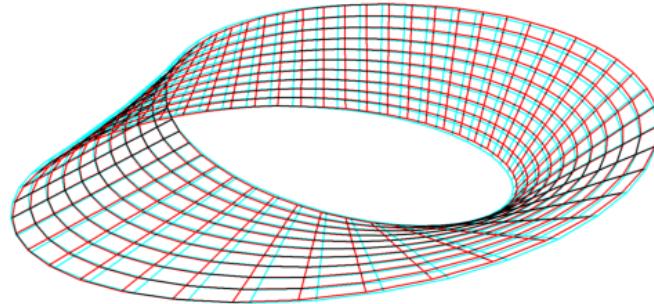
```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2):
```



Euler can plot parameterized surfaces too, when the parameters are the x-, y-, and z-values of an image of a rectangular grid in the space.

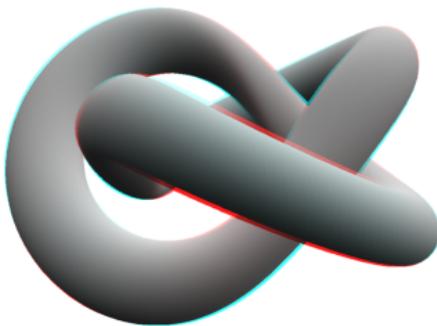
For the following demo, we setup u- and v- parameters, and generate space coordinates from these.

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



Here is a more complicated example, which is majestic with red/cyan glasses.

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



Statistical PLots

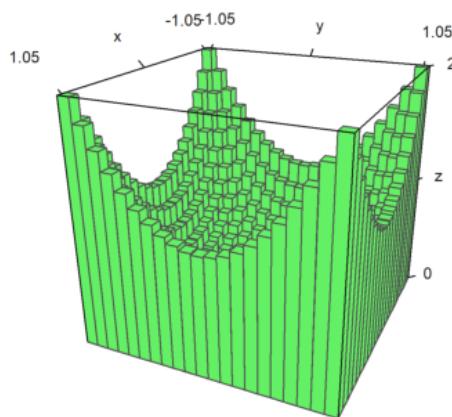
Bar plots are possible too. For this, we have to provide

- x: row vector with n+1 elements
- y: column vector with n+1 elements
- z: nxn matrix of values.

z can be larger, but only nxn values will be used.

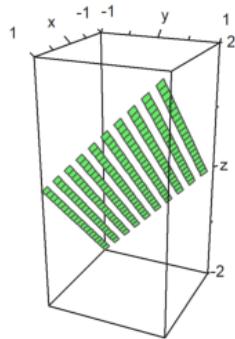
In the example, we first compute the values. Then we adjust x and y, so that the vectors center at the values used.

```
>x=-1:0.1:1; y=x'; z=x^2+y^2; ...
>xa=(x+1.1)-0.05; ya=(y-1.1)-0.05; ...
>plot3d(xa,ya,z,bar=true);
```



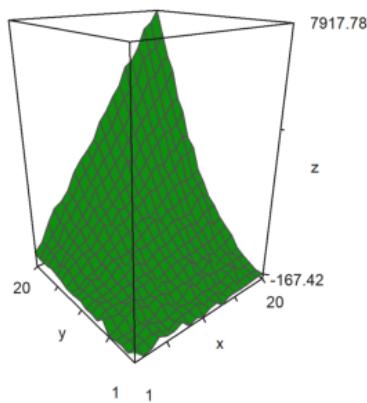
It is possible to split the plot of a surface in two or more parts.

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

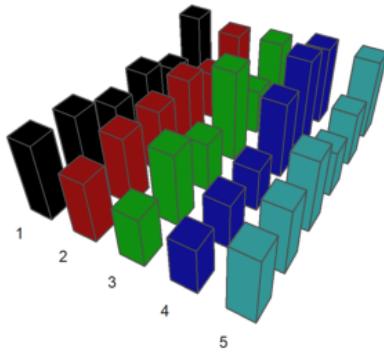


If load or generate a data matrix M from a file and need to plot it in 3D you can either scale the matrix to [-1,1] with scale(M), or scale the matrix with >zscale. This can be combined with individual scaling factors which are applied additionally.

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

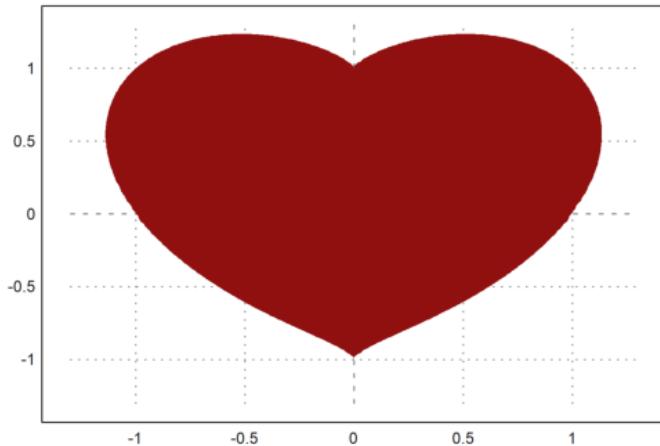


```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



Permukaan Benda Putar

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



```
>ekspresi &= (x^2+y^2-1)^3-x^2*y^3; $ekspresi
```

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

We wish to turn the heart curve around the y-axis. Here is the expression, which defines the heart:

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

Next we set

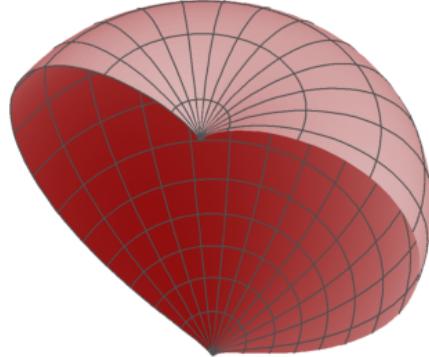
$$x = r.\cos(a), \quad y = r.\sin(a).$$

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

$$(r^2 - 1)^3 + \frac{(\sin(5a) - \sin(3a) - 2\sin a) r^5}{16}$$

This allows to define a numerical function, which solves for r, if a is given. With that function we can plot the turned heart as a parametric surface.

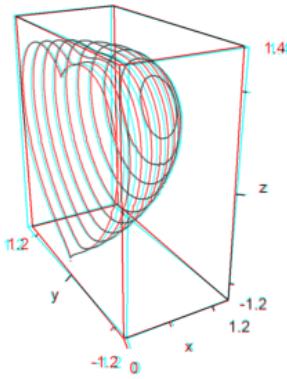
```
>function map f(a) := bisect("fr",0,2;a); ...
>t=linspace(-pi/2,pi/2,100); r=f(t); ...
>s=linspace(pi,2pi,100)'; ...
>plot3d(r*cos(t)*sin(s),r*cos(t)*cos(s),r*sin(t), ...
>>hue,<frame,color=red,zoom=4,amb=0,max=0.7,grid=12,height=50°):
```



The following is a 3D plot of the figure above rotated around the z-axis. We define the function, which describes the object.

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

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



Special 3D Plots

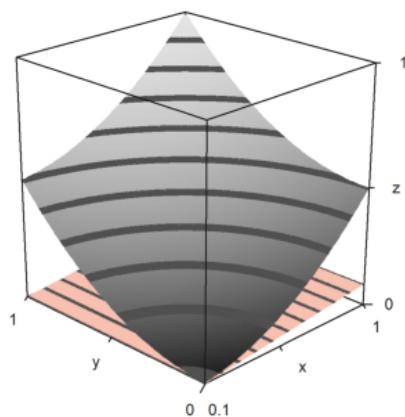
The `plot3d` function is nice to have, but it does not satisfy all needs. Besides more basic routines, it is possible to get a framed plot of any object you like.

Though Euler is not a 3D program, it can combine some basic objects. We try to visualize a paraboloid and its tangent.

```
>function myplot ...
y=0:0.01:1; x=(0.1:0.01:1)';
plot3d(x,y,0.2*(x-0.1)/2,<scale,<frame,>hue, ...
    hues=0.5,>contour,color=orange);
h=holding(1);
plot3d(x,y,(x^2+y^2)/2,<scale,<frame,>contour,>hue);
holding(h);
endfunction
```

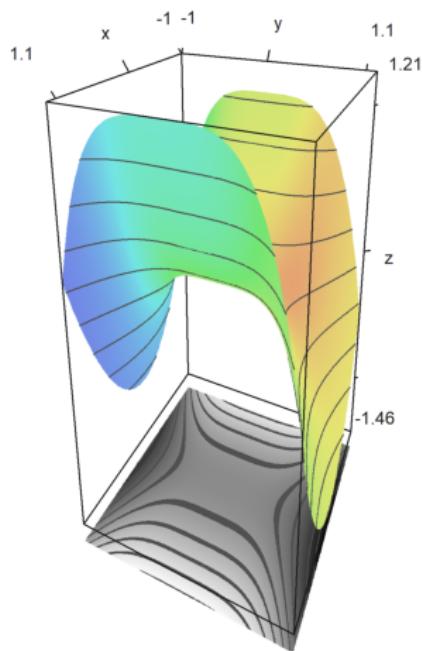
Now `framedplot()` provides the frames, and sets the views.

```
>framedplot("myplot", [0.1,1,0,1,0,1], angle=-45°, ...
> center=[0,0,-0.7], zoom=6):
```



In the same way, you can plot the contour plane manually. Note that `plot3d()` sets the window to `fullwindow()` by default, but `plotcontourplane()` assumes that.

```
>x=-1:0.02:1.1; y=x'; z=x^2-y^4;  
>function myplot (x,y,z) ...  
  
    zoom(2);  
    wi=fullwindow();  
    plotcontourplane(x,y,z,level="auto",<scale);  
    plot3d(x,y,z,>hue,<scale,>add,color=white,level="thin");  
    window(wi);  
    reset();  
endfunction  
  
>myplot(x,y,z);
```



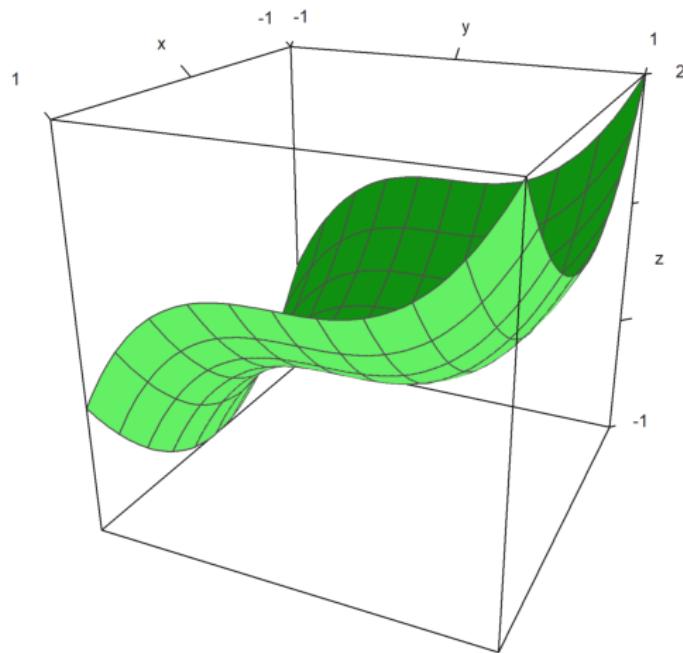
Animation

Euler can use frames to pre-compute the animation.

One function, which makes use of this technique is `rotate`. It can change the angle of view and redraw a 3D plot. The function calls `addpage()` for each new plot. Finally it animates the plots.

Please study the source of `rotate` to see more details.

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



Menggambar Povray

With the help of the Euler file povray.e, Euler can generate Povray files. The results are very nice to look at. You need to install Povray (32bit or 64bit) from <http://www.povray.org/>, and put the sub-directory "bin" of Povray into the environment path, or set the variable "defaultpovray" with full path pointing to "pvengine.exe". The Povray interface of Euler generates Povray files in the home directory of the user, and calls Povray to parse these files. The default file name is current.pov, and the default directory is eulerhome(), usually c:\Users\Username\Euler. Povray generates a PNG file, which can be loaded by Euler into a notebook. To clean up these files, use povclear().

The pov3d function is in the same spirit as plot3d. It can generate the graph of a function $f(x,y)$, or a surface with coordinates X,Y,Z in matrices, including optional level lines. This function starts the raytracer automatically, and loads the scene into the Euler notebook.

Besides pov3d(), there are many functions, which generate Povray objects. These functions return strings, containing the Povray code for the objects. To use these functions, start the Povray file with povstart(). Then use writeln(...) to write the objects to the scene file. Finally, end the file with povend(). By default, the raytracer will start, and the PNG will be inserted into the Euler notebook.

The object functions have a parameter called "look", which needs a string with Povray code for the texture and the finish of the object. The function povlook() can be used to produce this string. It has parameters for the color, the transparency, Phong Shading etc.

Note that the Povray universe has another coordinate system. This interface translates all coordinates to the Povray system. So you can keep thinking in the Euler coordinate system with z pointing vertically upwards, and x,y,z axes in right hand sense.

You need to load the povray file.

```
>load povray;
```

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

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

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

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

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

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

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

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

```
>pov3d("x^2+y^3",axiscolor=red,angle=20°, ...
>   look=povlook(blue,0.2),level=-1:0.5:1,zoom=3.8);
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

Sometimes it is necessary to prevent the scaling of the function, and scale the function by hand.

We plot the set of points in the complex plane, where the product of the distances to 1 and -1 is equal to 1.

```
>pov3d("((x-1)^2+y^2)*((x+1)^2+y^2)/40",r=1.5, ...
>   angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=45°,n=50, ...
>   <fscale,zoom=3.8);
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

Plotting with Coordinates

Instead of functions, we can plot with coordinates. As in plot3d, we need three matrices to define the object. In the example we turn a function around the z-axis.

```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,8)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,height=20°,axis=0,zoom=4,light=[10,-5,5]);
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

In the following example, we plot a damped wave. We generate the wave with the matrix language of Euler. We also show, how an additional object can be added to a pov3d scene. For the generation of objects, see the following examples. Note that plot3d scales the plot, so that it fits into the unit cube.

```
>r=linspace(0,1,80); phi=linspace(0,2pi,80)'; ...
>x=r*cos(phi); y=r*sin(phi); z=exp(-5*r)*cos(8*pi*r)/3; ...
>pov3d(x,y,z,zoom=5,axis=0,add=povsphere([0,0,0.5],0.1,povlook(green)), ...
> w=500,h=300);
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

With the advanced shading method of Povray, very few points can produce very smooth surfaces. Only at the boundaries and in shadows the trick might become obvious.

For this, we need to add normal vectors in each matrix point.

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

$$\begin{matrix} 2 & 3 \\ x & y \end{matrix}$$

The equation of the surface is $[x,y,Z]$. We compute the two derivatives to x and y of this and take the cross product as the normal.

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

We define the normal as the cross product of these derivatives, and define coordinate functions.

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

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

We use only 25 points.

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

```
Unexpected "( ". Index () not allowed in strict mode!
In Euler files, use relax to avoid this.
Error in:
pov3d(x,y,Z(x,y),angle=10°, xv=NX(x,y),yv=NY(x,y),zv=NZ(x,y) ...
^
```

The following is the Trefoil knot done by A. Busser in Povray. There is an improved version of this in the examples.

See: Examples\Trefoil Knot | Trefoil Knot

For a good look with not too many points, we add normal vectors here. We use Maxima to compute the normals for us. First, the three functions for the coordinates as symbolic expressions.

```
>X &= ((4+sin(3*y))+cos(x))*cos(2*y); ...
>Y &= ((4+sin(3*y))+cos(x))*sin(2*y); ...
>Z &= sin(x)+2*cos(3*y);
```

Then the two derivative vectors to x and y.

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

Now the normal, which is the cross product of the two derivatives.

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

We now evaluate all this numerically.

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

The normal vectors are evaluations of the symbolic expressions dn[i] for i=1,2,3. The syntax for this is &"expression"(parameters). This is an alternative to the method in the previous example, where we defined symbolic expressions NX, NY, NZ first.

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

```

Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;

```

We can also generate a grid in 3D.

```

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

```

```

Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);

```

With `povgrid()`, curves are possible.

```

>povstart(center=[0,0,1],zoom=3.6); ...
>t=linspace(0,2,1000); r=exp(-t); ...
>x=cos(2*pi*10*t)*r; y=sin(2*pi*10*t)*r; z=t; ...
>writeln(povgrid(x,y,z,povlook(red))); ...
>writeAxis(0,2,axis=3); ...
>povend();

```

```

Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);

```

Povray Objects

Above, we used `pov3d` to plot surfaces. The `povray` interface in Euler can also generate Povray objects. These objects are stored as strings in Euler, and need to be written to a Povray file.

We start the output with `povstart()`.

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

First we define the three cylinders, and store them in strings in Euler.
The functions `povx()` etc. simply returns the vector [1,0,0], which could be used instead.

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

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

```
>c1
```

```
cylinder { <-1,0,0>, <1,0,0>, 1
    texture { pigment { color rgb <0.564706,0.0627451,0.0627451> } }
    finish { ambient 0.2 }
}
```

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

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

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

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

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

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

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

```
>povend;
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

The following functions generate a fractal recursively.

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

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

```

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

```

```

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

```

Command was not allowed!

```

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

```

Differences allow cutting off one object from another. Like intersections, there are part of the CSG objects of Povray.

```

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

```

For this demonstration, we define an object in Povray, instead of using a string in Euler. Definitions are written to the file immediately.

A box coordinate of -1 just means [-1,-1,-1].

```

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

```

We can use this object in povobject(), which returns a string as usual.

```

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

```

We generate a second cube, and rotate and scale it a bit.

```

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

```

Then we take the difference of the two objects.

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

Now add three axes.

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

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

Implicit Functions

Povray can plot the set where $f(x,y,z)=0$, just like the implicit parameter in plot3d. The results looks much better, however.

The syntax for the functions is a bit different. You cannot use the output of Maxima or Euler expressions.

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

Create the implicit surface. Note the different syntax in the expression.

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

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

Mesh Object

In this example, we show how to create a mesh object, and draw it with additional information.

We like to maximize xy under the condition $x+y=1$ and demonstrate the tangential touching of the level lines.

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

We cannot store the object in a string as before, since is too large. So we define the object in a Povray file using declare. The function povtriangle() does this automatically. It can accept normal vectors just like pov3d(). The following defines the mesh object, and writes it immediately into the file.

```
>x=0:0.02:1; y=x'; z=x*y; vx=-y; vy=-x; vz=1;  
>mesh=povtriangles(x,y,z,"",vx,vy,vz);
```

Now we define two discs, which will be intersected with the surface.

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

Write the surface minus the two discs.

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

Write the two intersections.

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

Write a point at the maximum.

```
>writeln(povpoint([1/2,1/2,1/4],povlook(gray),size=2*defaultpointsize));
```

Add axes and finish.

```
>writeAxes(0,1,0,1,0,1,d=0.015); ...  
>povend();
```

```
Command was not allowed!  
exec:  
    return _exec(program,param,dir,print,hidden,wait);  
povray:  
    exec(program,params,defaulthome);  
Try "trace errors" to inspect local variables after errors.  
povend:  
    povray(file,w,h,aspect,exit);
```

Anaglyphs in Povray

To generate an anaglyph for a red/cyan glasses, Povray must run twice from different camera positions. It generates two Povray files and two PNG files, which are loaded with the function `loadanaglyph()`. Of course, you need red/cyan glasses to view the following examples properly. The function `pov3d()` has a simple switch to generate anaglyphs.

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

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
pov3d:
    if povray then povray(currentfile,w,h,w/h); endif;
```

If you create a scene with objects, you need to put the generation of the scene into a function, and run it twice with different values for the anaglyph parameter.

```
>function myscene ...
```

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

The function `povanaglyph()` does all this. The parameters are like in `povstart()` and `povend()` combined.

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

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povanaglyph:
    povray(currentfile,w,h,aspect,exit);
```

Defining own Objects

The povray interface of Euler contains a lot of objects. But you are not restricted to these. You can create own objects, which combine other objects, or are completely new objects.

We demonstrate a torus. The Povray command for this is "torus". So we return a string with this command and its parameters. Note that the torus is always centered at the origin.

```
>function povdonat (r1,r2,look="") ...
```

```
    return "torus {" + r1 + "," + r2 + look + "}";  
endfunction
```

Here is our first torus.

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

```
torus {0.8,0.2}
```

Let us use this object to create a second torus, translated and rotated.

```
>t2=povobject(t1,rotate=xrotate(90°),translate=[0.8,0,0])
```

```
object { torus {0.8,0.2}  
    rotate 90 *x  
    translate <0.8,0,0>  
}
```

Now we place these objects into a scene. For the look, we use Phong Shading.

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

```
>povend();
```

calls the Povray program. However, in case of errors, it does not display the error. You should therefore use

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

if anything did not work. This will leave the Povray window open.

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

```
Command was not allowed!  
exec:  
    return _exec(program,param,dir,print,hidden,wait);  
povray:  
    exec(program,params,defaulthome);  
Try "trace errors" to inspect local variables after errors.  
povend:  
    povray(file,w,h,aspect,exit);
```

Here is a more elaborate example. We solve

$$Ax \leq b, \quad x \geq 0, \quad c.x \rightarrow \text{Max.}$$

and show the feasible points and the optimum in a 3D plot.

```
>A=[10,8,4;5,6,8;6,3,2;9,5,6];
>b=[10,10,10,10]';
>c=[1,1,1];
```

First, let us check, if this example has a solution at all.

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

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

Yes, it has.

Next we define two objects. The first is the plane

$$a \cdot x \leq b$$

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

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

Then we define the intersection of all half spaces and a cube.

```
>function adm (A, b, r, look "") ...
```

```
ol=[];
loop 1 to rows(A); ol=ol|oneplane(A[#,b[#]); end;
ol=ol|povbox([0,0,0],[r,r,r]);
return povintersection(ol,look);
endfunction
```

We can now plot the scene.

```
>povstart(angle=120°,center=[0.5,0.5,0.5],zoom=3.5); ...
>writeln(adm(A,b,2,povlook(green,0.4))); ...
>writeAxes(0,1.3,0,1.6,0,1.5); ...
```

The following is a circle around the optimum.

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

And an error in the direction of the optimum.

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

We add text to the screen. Text is just a 3D object. We need to place and turn it according to our view.

```
>writeln(povtext("Linear Problem", [0,0.2,1.3],size=0.05,rotate=125°)); ...
>povend();
```

```
Command was not allowed!
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

More Examples

You can find some more examples for Povray in Euler in the following files.

See: Examples/Dandelin Spheres

See: Examples/Donat Math

See: Examples/Trefoil Knot

See: Examples/Optimization by Affine Scaling

BAB 4

KB PEKAN 6-7: MENGGUNAKAN EMT UNTUK KALKULUS

[a4paper,10pt]article eumat

IDENTITAS

Kalkulus dengan EMT

Materi Kalkulus mencakup di antaranya:

- Fungsi (fungsi aljabar, trigonometri, eksponensial, logaritma, komposisi fungsi)
- Limit Fungsi,
- Turunan Fungsi,
- Integral Tak Tentu,
- Integral Tentu dan Aplikasinya,
- Barisan dan Deret (kekonvergenan barisan dan deret).

EMT (bersama Maxima) dapat digunakan untuk melakukan semua perhitungan di dalam kalkulus, baik secara numerik maupun analitik (eksak).

Mendefinisikan Fungsi

Terdapat beberapa cara mendefinisikan fungsi pada EMT, yakni:

- Menggunakan format `nama_fungsi := rumus fungsi` (untuk fungsi numerik),
- Menggunakan format `nama_fungsi &= rumus fungsi` (untuk fungsi simbolik, namun dapat dihitung secara numerik),
- Menggunakan format `nama_fungsi &&= rumus fungsi` (untuk fungsi simbolik murni, tidak dapat dihitung langsung),
- Fungsi sebagai program EMT.

Setiap format harus diawali dengan perintah `function` (bukan sebagai ekspresi).

Berikut adalah beberapa contoh cara mendefinisikan fungsi.

```
>function f(x) := 2*x^2+exp(sin(x)) // fungsi numerik
>f(0), f(1), f(pi)
```

```
1  
4.31977682472  
20.7392088022
```

```
>function g(x) := sqrt(x^2-3*x) / (x+1)  
>g(3)
```

```
0
```

```
>g(0)
```

```
0
```

```
>g(1)
```

```
Floating point error!  
Error in sqrt  
Try "trace errors" to inspect local variables after errors.  
g:  
    useglobal; return sqrt(x^2-3*x) / (x+1)  
Error in:  
g(1) ...  
^
```

```
>f(g(5)) // komposisi fungsi
```

```
2.20920171961
```

```
>g(f(5))
```

```
0.950898070639
```

```
>f(0:10) // nilai-nilai f(1), f(2), ..., f(10)
```

```
[1, 4.31978, 10.4826, 19.1516, 32.4692, 50.3833, 72.7562,  
99.929, 130.69, 163.51, 200.58]
```

```
>fmap(0:10) // sama dengan f(0:10), berlaku untuk semua fungsi
```

```
[1, 4.31978, 10.4826, 19.1516, 32.4692, 50.3833, 72.7562,  
99.929, 130.69, 163.51, 200.58]
```

Misalkan kita akan mendefinisikan fungsi

$$f(x) = \begin{cases} x^3 & x > 0 \\ x^2 & x \leq 0. \end{cases}$$

Fungsi tersebut tidak dapat didefinisikan sebagai fungsi numerik secara "inline" menggunakan format `:=`, melainkan didefinisikan sebagai program. Perhatikan, kata "map" digunakan agar fungsi dapat menerima vektor sebagai input, dan hasilnya berupa vektor. Jika tanpa kata "map" fungsinya hanya dapat menerima input satu nilai.

```
>function map f(x) ...
    if x>0 then return x^3
    else return x^2
    endif;
endfunction
```

```
>f(1)
```

```
1
```

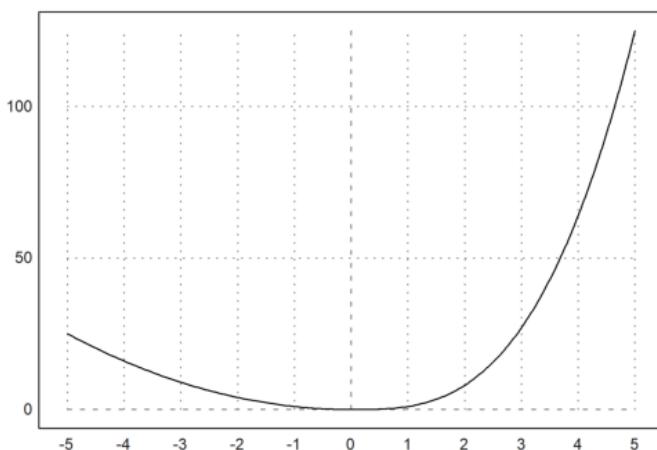
```
>f(-2)
```

```
4
```

```
>f(-5:5)
```

```
[25, 16, 9, 4, 1, 0, 1, 8, 27, 64, 125]
```

```
>aspect(1.5); plot2d("f(x)", -5, 5):
```



```
>function f(x) &= 2*x^2 // fungsi simbolik
```

$$2 \frac{x^2}{E}$$

```
>function g(x) &= 3*x+1
```

$$3 \ x + 1$$

```
>function h(x) &= f(g(x)) // komposisi fungsi
```

$$\begin{matrix} 3 & x & + & 1 \\ 2 & E \end{matrix}$$

Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, hitung beberapa nilainya, baik untuk satu nilai maupun vektor. Gambar grafik tersebut.

Juga, carilah fungsi beberapa (dua) variabel. Lakukan hal sama seperti di atas.

Jawab:

A). FUNGSI 1 VARIABEL

1. Fungsi 1

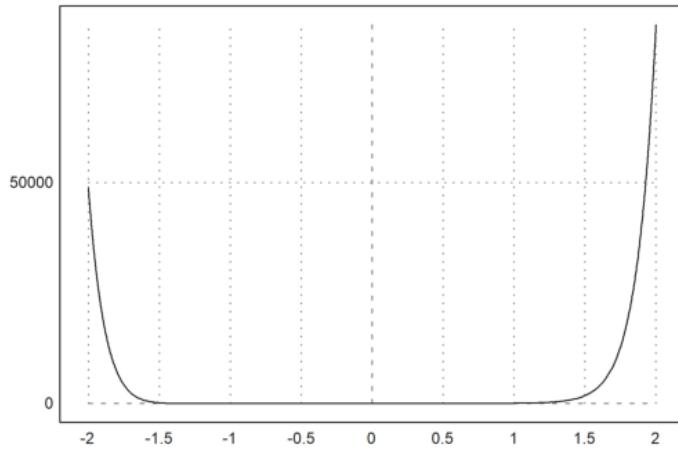
```
>function k(x) := x*(x^5+3)^3  
>k(3), k(5), k(7)
```

44660808
153027765760
3.3250729687e+13

```
>kmap(-3:3)
```

[4.1472e+07, 48778, -8, 0, 64, 85750, 4.46608e+07]

```
>plot2d("k(x)":
```



2. Fungsi 2

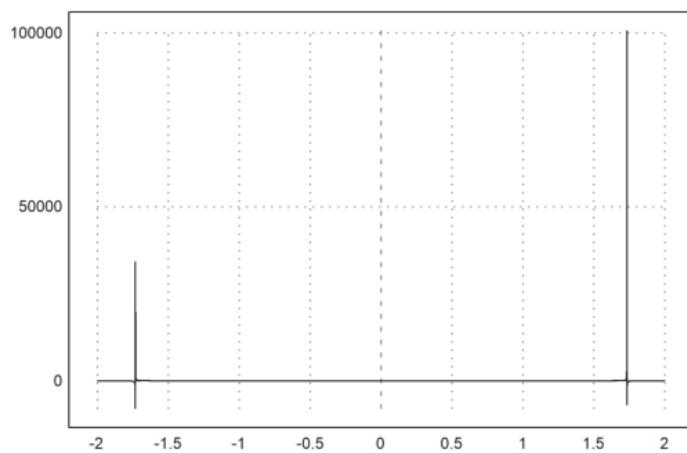
```
>function m(x) := (x)^4 / (3-x^2)
>m(2), m(-2), m(1)
```

-16
-16
0.5

```
>mmap (-5:-5)
```

-28.4090909091

```
>plot2d ("m(x) "):
```



3. Fungsi 3

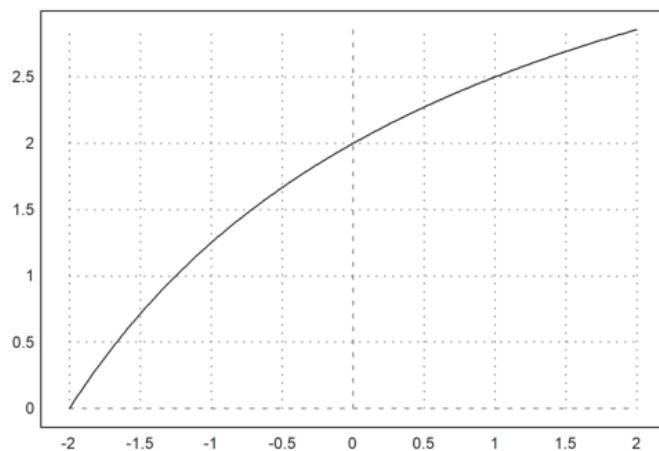
```
>function n(x) := 3*x/(x+5)+2  
>n(2), n(-1), n(-3), n(4)
```

2.85714285714
1.25
-2.5
3.33333333333

```
>nmap(2:5)
```

[2.85714, 3.125, 3.33333, 3.5]

```
>plot2d("n(x)":
```



4. Fungsi 4

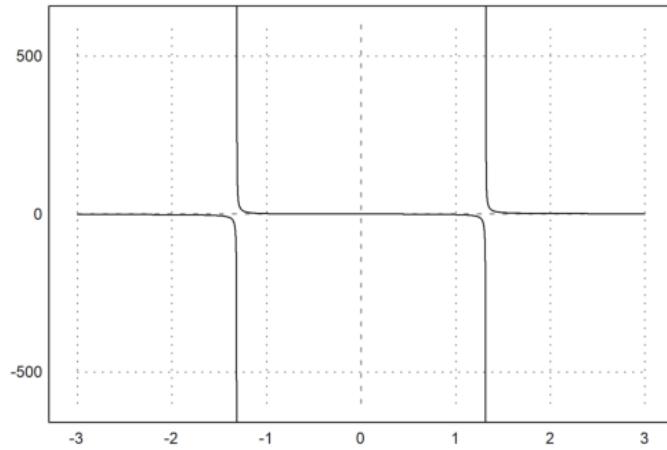
```
>function l(x) := 3*x^3/(x^4-3)  
>l(5), l(4), l(3)
```

0.602893890675
0.758893280632
1.03846153846

```
>lmap(5:8)
```

[0.602894, 0.50116, 0.429108, 0.375275]

```
>plot2d("l(x)", -3, 3, -600, 600):
```



5. Fungsi 5

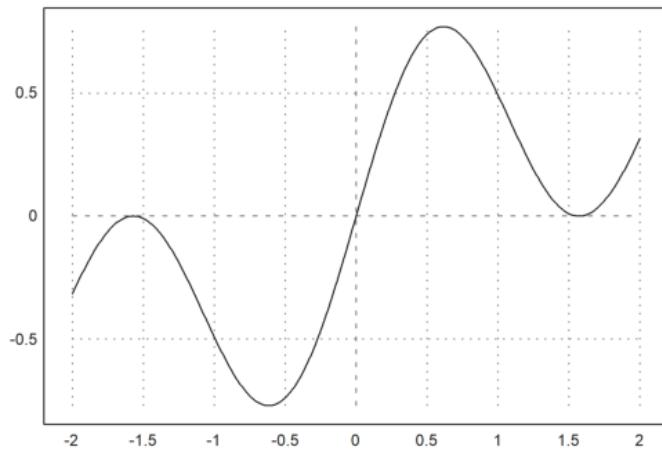
```
>function j(x) := (cos(x))*sin(2*x)
>j(pi), j(0), j(pi/3)
```

```
0
0
0.433012701892
```

```
>jmap(0:3pi)
```

```
[0, 0.491295, 0.314941, 0.276619, -0.646688, -0.154318,
-0.515201, 0.746821, 0.0418899, 0.684247]
```

```
>plot2d("j(x)":
```



6. Fungsi 6

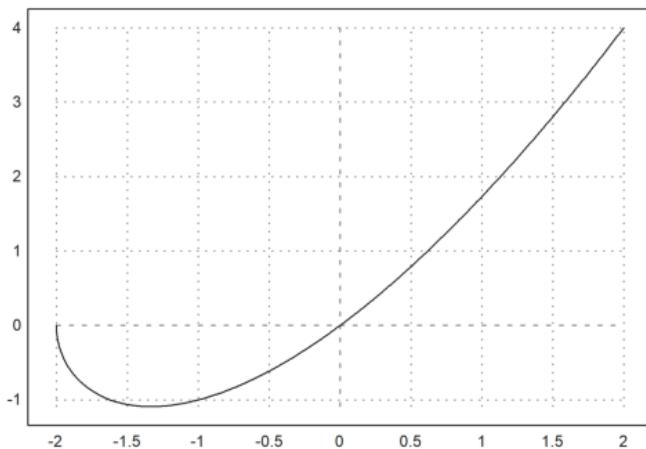
```
>function o(x) := x*sqrt(x+2)
>o(3), o(5), o(7)
```

6.7082039325
13.2287565553
21

```
>omap(3:12)
```

[6.7082, 9.79796, 13.2288, 16.9706, 21, 25.2982, 29.8496,
34.641, 39.6611, 44.8999]

```
>plot2d("o(x)":
```



B). FUNGSI 2 VARIABEL

1. Fungsi 1

```
>function a(x,y) ...
```

```
return x^2+y^2-24
endfunction
```

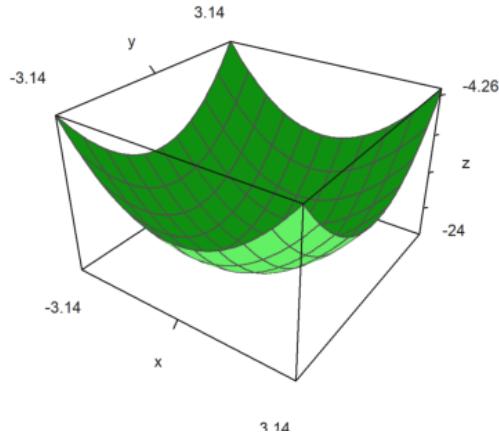
```
>a(2,1), a(5,4), a(2,4)
```

-19
17
-4

```
>amap (-2:2,3:3)
```

```
[-11, -14, -15, -14, -11]
```

```
>aspect=1.5; plot3d("a(x,y)",a=-100,b=100,c=-80,d=80,angle=35°,height=30°,r=pi,n=100):
```



2. Fungsi 2

```
>function q(x,y) ...
```

```
return y^2/(x^2/3)
endfunction
```

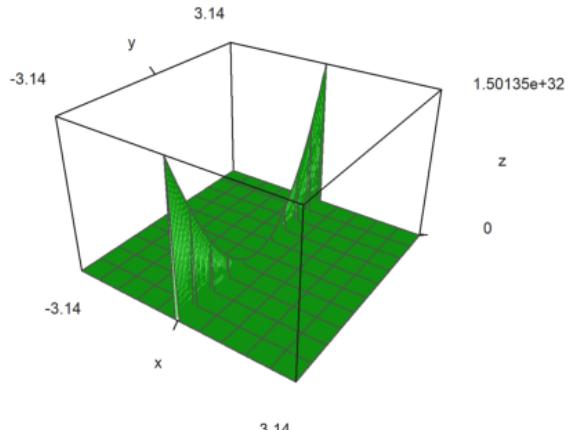
```
>q(4,2), q(2,3), q(4,3)
```

```
0.75
6.75
1.6875
```

```
>qmap (2:2,-2:2)
```

```
[3, 0.75, 0, 0.75, 3]
```

```
>aspect=1.5; plot3d("q(x,y)",a=-100,b=100,c=-80,d=80,angle=35°,height=30°,r=pi,n=100):
```



Menghitung Limit

Perhitungan limit pada EMT dapat dilakukan dengan menggunakan fungsi Maxima, yakni "limit". Fungsi "limit" dapat digunakan untuk menghitung limit fungsi dalam bentuk ekspresi maupun fungsi yang sudah didefinisikan sebelumnya. Nilai limit dapat dihitung pada sebarang nilai atau pada tak hingga (-inf, minf, dan inf). Limit kiri dan limit kanan juga dapat dihitung, dengan cara memberi opsi "plus" atau "minus". Hasil limit dapat berupa nilai, "und" (tak definisi), "ind" (tak tentu namun terbatas), "infinity" (kompleks tak hingga). Perhatikan beberapa contoh berikut. Perhatikan cara menampilkan perhitungan secara lengkap, tidak hanya menampilkan hasilnya saja.

```
>$showev('limit(1/(2*x-1),x,0))
```

$$\lim_{x \rightarrow 0} \frac{1}{2x - 1} = -1$$

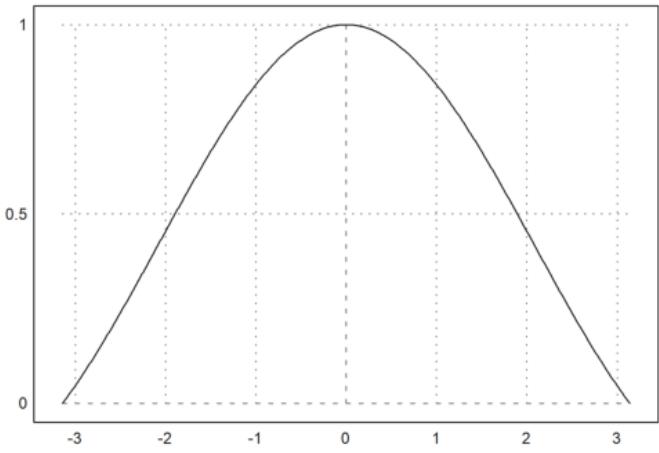
```
>$showev('limit((x^2-3*x-10)/(x-5),x,5))
```

$$\lim_{x \rightarrow 5} \frac{x^2 - 3x - 10}{x - 5} = 7$$

```
>$showev('limit(sin(x)/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

```
>plot2d("sin(x)/x",-pi,pi):
```



```
>$showev('limit(sin(x^3)/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{\sin x^3}{x} = 0$$

```
>$showev('limit(log(x), x, minf))
```

$$\lim_{x \rightarrow -\infty} \log x = \text{infinity}$$

```
>$showev('limit((-2)^x,x, inf))
```

$$\lim_{x \rightarrow \infty} (-2)^x = \text{infinity}$$

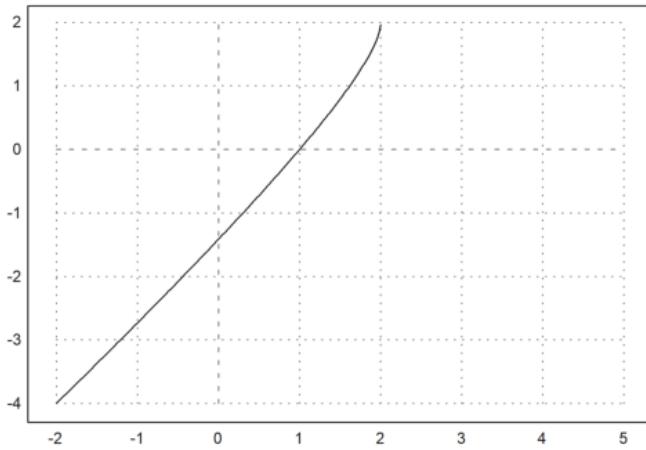
```
>$showev('limit(t-sqrt(2-t),t,2,minus))
```

$$\lim_{t \uparrow 2} t - \sqrt{2-t} = 2$$

```
>$showev('limit(t-sqrt(2-t),t,5,plus)) // Perhatikan hasilnya
```

$$\lim_{t \downarrow 5} t - \sqrt{2-t} = 5 - \sqrt{3}i$$

```
>plot2d("x-sqrt(2-x)",-2,5):
```



```
>$showev('limit((x^2-9)/(2*x^2-5*x-3),x,3))
```

$$\lim_{x \rightarrow 3} \frac{x^2 - 9}{2x^2 - 5x - 3} = \frac{6}{7}$$

```
>$showev('limit((1-cos(x))/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{1 - \cos x}{x} = 0$$

```
>$showev('limit((x^2+abs(x))/(x^2-abs(x)),x,0))
```

$$\lim_{x \rightarrow 0} \frac{|x| + x^2}{x^2 - |x|} = -1$$

```
>$showev('limit((1+1/x)^x,x,inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{1}{x} + 1 \right)^x = e$$

```
>$showev('limit((1+k/x)^x,x,inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{k}{x} + 1 \right)^x = e^k$$

```
>$showev('limit((1+x)^(1/x),x,0))
```

$$\lim_{x \rightarrow 0} (x + 1)^{\frac{1}{x}} = e$$

```
>$showev('limit((x/(x+k))^x,x,inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{x}{x+k} \right)^x = e^{-k}$$

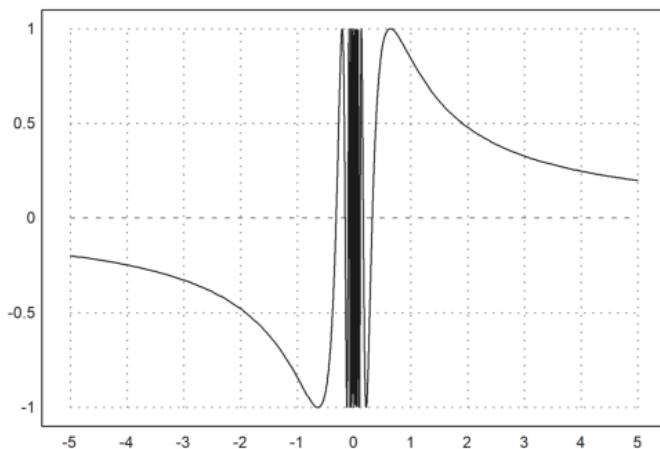
```
>$showev('limit(sin(1/x),x,0))
```

$$\lim_{x \rightarrow 0} \sin \left(\frac{1}{x} \right) = \text{ind}$$

```
>$showev('limit(sin(1/x),x,inf))
```

$$\lim_{x \rightarrow \infty} \sin \left(\frac{1}{x} \right) = 0$$

```
>plot2d("sin(1/x)",-5,5):
```



Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, hitung nilai limit fungsi tersebut di beberapa nilai dan di tak hingga. Gambar grafik fungsi tersebut untuk mengkonfirmasi nilai-nilai limit tersebut.

Jawab:

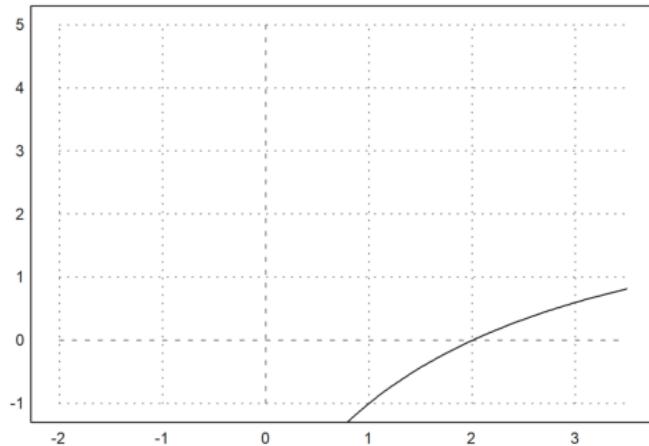
1. Fungsi 1

$$f(x) = \frac{3x-6}{x+2}$$

```
>$showev('limit((3*x-6)/(x+2),x,2))
```

$$\lim_{x \rightarrow 2} \frac{3x - 6}{x + 2} = 0$$

```
>plot2d("(3*x-6)/(x+2)", -2, 3.5, -1, 5):
```



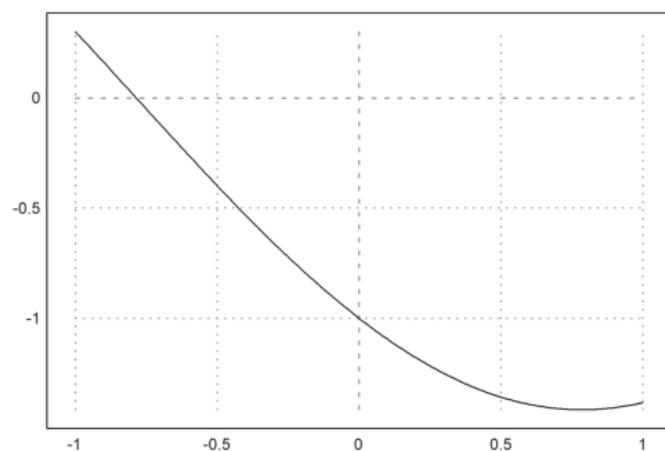
2. Fungsi 2

$$f(x) = \frac{\cos 2x}{\sin x - \cos x}$$

```
>showev('limit(cos(2*x)/(sin(x) - cos(x)), x, 0))
```

$$\lim_{x \rightarrow 0} \frac{\cos(2x)}{\sin x - \cos x} = -1$$

```
>plot2d("cos(2*x)/(sin(x) - cos(x))", -1, 1):
```



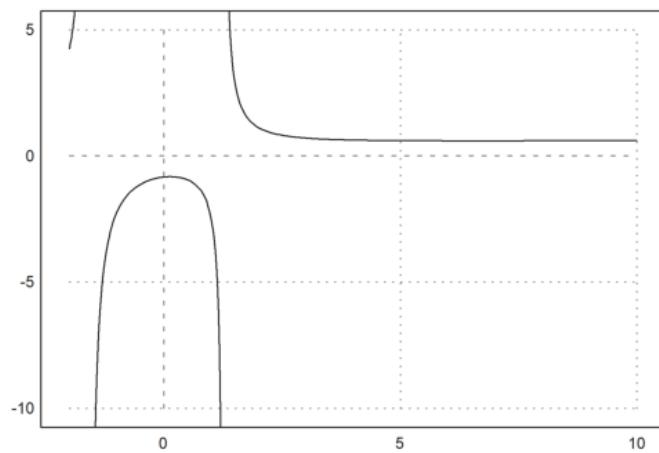
3. Fungsi 3

$$f(x) = \frac{2x^2 - 2x + 5}{3x^2 + x - 6}$$

```
>$showev('limit(((2*x^2-2*x+5)/(3*x^2+x-6)),x,3))
```

$$\lim_{x \rightarrow 3} \frac{2x^2 - 2x + 5}{3x^2 + x - 6} = \frac{17}{24}$$

```
>plot2d("(2*x^2-2*x+5)/(3*x^2+x-6)",-2,10,-10,5):
```



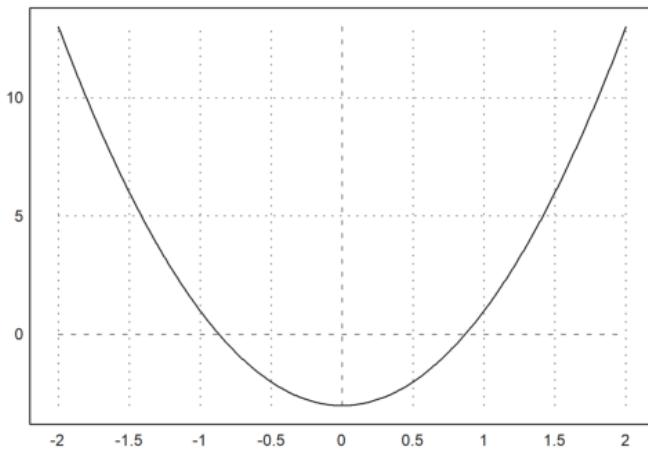
4. Fungsi 4

$$f(x) = 4x^2 - 3$$

```
>$showev('limit((4*x^2-3),x,0))
```

$$\lim_{x \rightarrow 0} 4x^2 - 3 = -3$$

```
>plot2d("(4*x^2-3)":
```



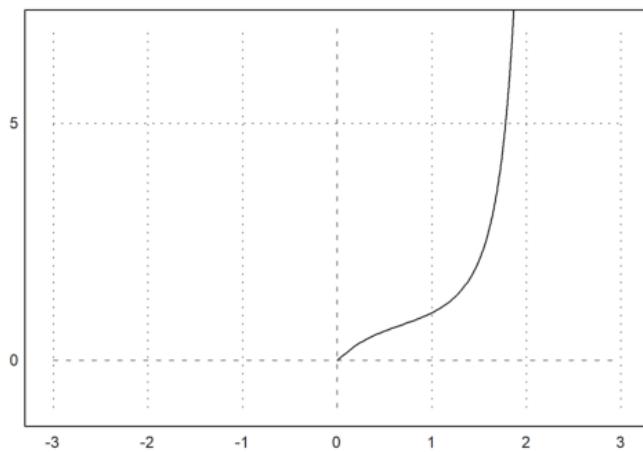
5. Fungsi 5

$$f(x) = x^{x^x}$$

```
>showev('limit((x^(x^x)), x, 0, plus))
```

$$\lim_{x \downarrow 0} x^{x^x} = 0$$

```
>plot2d(" (x^(x^x)) ", -3, 3, -1, 7);
```



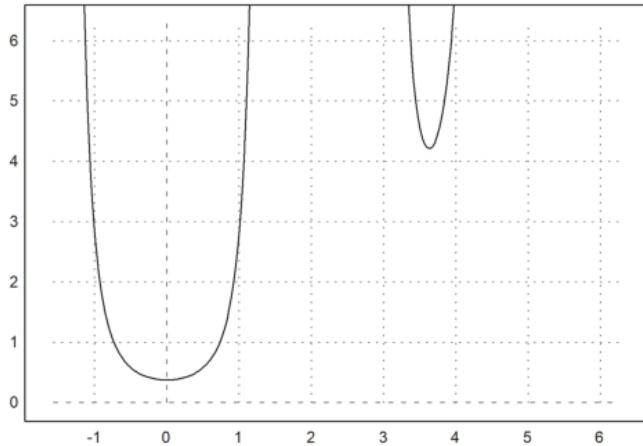
6. Fungsi 6

$$f(x) = \frac{3xtanx}{1 - \cos 4x}$$

```
>showev('limit((3*x*tan(x)) / (1 - cos(4*x)), x, 0))
```

$$3 \left(\lim_{x \rightarrow 0} \frac{x \tan x}{1 - \cos(4x)} \right) = \frac{3}{8}$$

```
>plot2d("(3*x*tan(x)) / (1-cos(4*x))", -pi/2, 2pi, 0, 2pi):
```



Turunan Fungsi

Definisi turunan:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Berikut adalah contoh-contoh menentukan turunan fungsi dengan menggunakan definisi turunan (limit).

```
>$showev('limit(((x+h)^n-x^n)/h,h,0)) // turunan x^n
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} = n x^{n-1}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini.

Sebagai petunjuk, ekspansikan $(x+h)^n$ dengan menggunakan teorema binomial.

Jawab:

$$\text{Akan ditunjukkan bahwa } f'(x) = \lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} = n x^{n-1}$$

Pertama, ekspansikan $(x+h)^n$, yakni:

$$\begin{aligned} (x+h)^n &= \sum_{k=0}^n \binom{n}{k} x^{n-k} h^k \\ \Leftrightarrow (x+h)^n &= \binom{n}{0} x^n + \binom{n}{1} x^{n-1} h + \binom{n}{2} x^{n-2} h^2 + \dots + \binom{n}{n} h^n \\ \Leftrightarrow (x+h)^n &= x^n + n x^{n-1} h + \binom{n}{2} x^{n-2} h^2 + \binom{n}{3} x^{n-3} h^3 + \dots + h^n \end{aligned}$$

Sehingga, $f'(x)$ menjadi: $f'(x) = \lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h}$

$$\Leftrightarrow f'(x) = \lim_{h \rightarrow 0} \frac{x^n + nx^{n-1}h + \binom{n}{2}x^{n-2}h^2 + \binom{n}{3}x^{n-3}h^3 + \dots + h^n - x^n}{h}$$

$$\Leftrightarrow f'(x) = \lim_{h \rightarrow 0} nx^{n-1} + \binom{n}{2}x^{n-2}h + \binom{n}{3}x^{n-3}h^2 + \dots + h^{n-1}$$

$$\Leftrightarrow f'(x) = nx^{n-1}. \text{ Terbukti.}$$

```
>$showev('limit((sin(x+h)-sin(x))/h,h,0)) // turunan sin(x)
```

$$\lim_{h \rightarrow 0} \frac{\sin(x+h) - \sin x}{h} = \cos x$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini. Sebagai petunjuk, ekspansikan $\sin(x+h)$ dengan menggunakan rumus jumlah dua sudut.
Jawab:

Akan ditunjukkan bahwa $\lim_{h \rightarrow 0} \frac{\sin(x+h) - \sin x}{h} = \cos x$

Diketahui bahwa:

$$1). \sin(x+h) = \sin x \cos h + \cos x \sin h$$

$$2). \lim_{h \rightarrow 0} \frac{1 - \cos h}{h} = 0$$

$$3). \lim_{h \rightarrow 0} \frac{\sin h}{h} = 1$$

$$\begin{aligned} & \lim_{h \rightarrow 0} \frac{\sin(x+h) - \sin x}{h} \\ &= \lim_{h \rightarrow 0} \frac{\sin x \cos h + \cos x \sin h - \sin x}{h} \\ &= \lim_{h \rightarrow 0} \left[-\sin x \cdot \frac{1 - \cos h}{h} + \cos x \cdot \frac{\sin h}{h} \right] \\ &= (-\sin x) \left[\lim_{h \rightarrow 0} \frac{1 - \cos h}{h} + (\cos x) \lim_{h \rightarrow 0} \frac{\sin h}{h} \right] \\ &= (-\sin x)(0) + (\cos x)(1) = \cos x. \text{ Terbukti.} \end{aligned}$$

```
>$showev('limit((log(x+h)-log(x))/h,h,0)) // turunan log(x)
```

$$\lim_{h \rightarrow 0} \frac{\log(x+h) - \log x}{h} = \frac{1}{x}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini. Sebagai petunjuk, gunakan sifat-sifat logaritma dan hasil limit pada bagian sebelumnya di atas.

Jawab:

Bukti:

Ambil $f(x) =^a \log x$.

$$\begin{aligned}
& \lim_{h \rightarrow 0} \frac{^a \log(x+h) - ^a \log x}{h} \\
&= \lim_{h \rightarrow 0} \frac{^a \log \frac{(x+h)}{x}}{h} \\
&= \lim_{h \rightarrow 0} \frac{^a \log(1 + \frac{h}{x})}{h} \\
&= \lim_{h \rightarrow 0} \frac{^a \log(1 + \frac{h}{x})}{\frac{h}{x} x} \\
&= \lim_{h \rightarrow 0} \frac{\frac{x}{h} \cdot ^a \log(1 + \frac{h}{x})}{x} \\
&= \lim_{h \rightarrow 0} \frac{^a \log(1 + \frac{h}{x})^{\frac{x}{h}}}{x} \\
&= \frac{\lim_{h \rightarrow 0} ^a \log(1 + \frac{h}{x})^{\frac{x}{h}}}{\lim_{h \rightarrow 0} x} \\
&= \frac{1}{x \cdot {}^e \log a} \\
&= \frac{1}{x \cdot \ln a}
\end{aligned}$$

Menggunakan hasil di atas, maka:

$$\frac{d \ln x}{dx} = \frac{d {}^e \log x}{dx} = \frac{1}{x \cdot \ln e} = \frac{1}{x}. \text{Terbukti.}$$

```
>$showev('limit((1/(x+h)-1/x)/h,h,0)) // turunan 1/x
```

$$\lim_{h \rightarrow 0} \frac{\frac{1}{x+h} - \frac{1}{x}}{h} = -\frac{1}{x^2}$$

```
>$showev('limit((E^(x+h)-E^x)/h,h,0)) // turunan f(x)=e^x
```

```

Answering "Is x an integer?" with "integer"
Maxima is asking
Acceptable answers are: yes, y, Y, no, n, N, unknown, uk

```

Is x an integer?

Use assume!

Error in:

```
$showev('limit((E^(x+h)-E^x)/h,h,0)) // turunan f(x)=e^x ...  
^
```

Maxima bermasalah dengan limit:

$$\lim_{h \rightarrow 0} \frac{e^{x+h} - e^x}{h}.$$

Oleh karena itu diperlukan trik khusus agar hasilnya benar.

```
>$showev('limit((E^h-1)/h,h,0))
```

$$\lim_{h \rightarrow 0} \frac{e^h - 1}{h} = 1$$

```
>$factor(E^(x+h)-E^x)
```

$$(e^h - 1) e^x$$

```
>$showev('limit(factor((E^(x+h)-E^x)/h),h,0)) // turunan f(x)=e^x
```

$$\left(\lim_{h \rightarrow 0} \frac{e^h - 1}{h} \right) e^x = e^x$$

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

$$\begin{matrix} x \\ x \end{matrix}$$

```
>$showev('limit((f(x+h)-f(x))/h,h,0)) // turunan f(x)=x^x
```

$$\lim_{h \rightarrow 0} \frac{(x + h)^{x+h} - x^x}{h} = infinity$$

Di sini Maxima juga bermasalah terkait limit:

$$\lim_{h \rightarrow 0} \frac{(x + h)^{x+h} - x^x}{h}.$$

Dalam hal ini diperlukan asumsi nilai x.

```
>&assume(x>0); $showev('limit((f(x+h)-f(x))/h,h,0)) // turunan f(x)=x^x
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^{x+h} - x^x}{h} = x^x (\log x + 1)$$

```
>&forget(x>0) // jangan lupa, lupakan asumsi untuk kembali ke semula
```

[x > 0]

```
>&forget(x<0)
```

[x < 0]

```
>&facts()
```

[]

```
>$showev('limit((asin(x+h)-asin(x))/h,h,0)) // turunan arcsin(x)
```

$$\lim_{h \rightarrow 0} \frac{\arcsin(x+h) - \arcsin x}{h} = \frac{1}{\sqrt{1-x^2}}$$

```
>$showev('limit((tan(x+h)-tan(x))/h,h,0)) // turunan tan(x)
```

$$\lim_{h \rightarrow 0} \frac{\tan(x+h) - \tan x}{h} = \frac{1}{\cos^2 x}$$

```
>function f(x) &= sinh(x) // definisikan f(x)=sinh(x)
```

sinh(x)

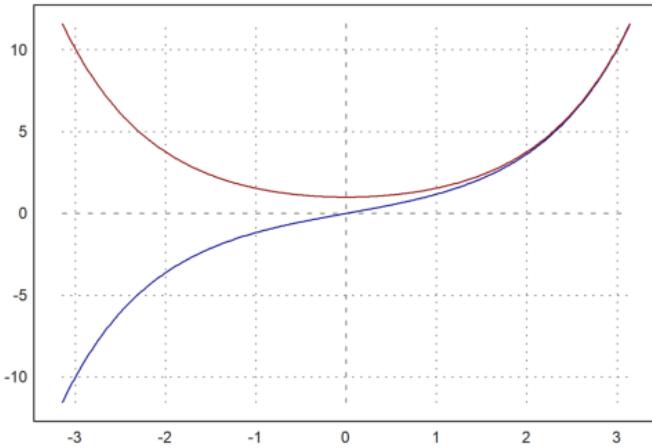
```
>function df(x) &= limit((f(x+h)-f(x))/h,h,0); $df(x) // df(x) = f'(x)
```

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

Hasilnya adalah cosh(x), karena

$$\frac{e^x + e^{-x}}{2} = \cosh(x).$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]):
```



Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, tentukan turunannya dengan menggunakan definisi turunan (limit), seperti contoh-contoh tersebut. Gambar grafik fungsi asli dan fungsi turunannya pada sumbu koordinat yang sama.

Jawab:

1. Fungsi 1

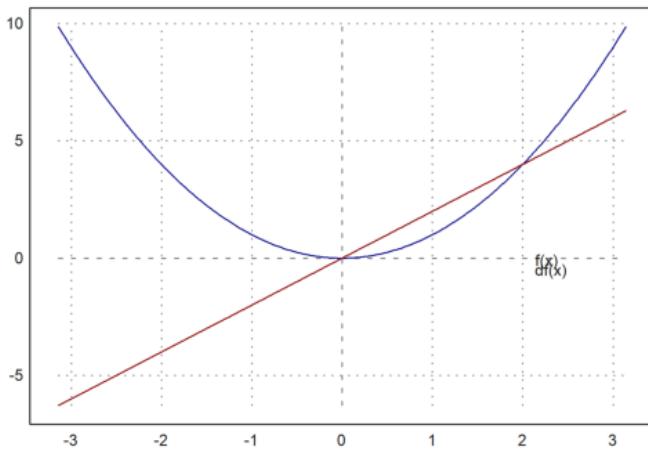
```
>function f(x) := x^2
>$showev('limit(((x+h)^2-x^2)/h,h,0)) // turunan x^2
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h} = 2x$$

```
>function df(x) &= limit(((x+h)^2-x^2)/h,h,0); $df(x) // df(x) = f'(x)
```

$$2x$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]), label("f(x)", 2, 0.6), label("df(x)", 2, 0.6)
```



2. Fungsi 2

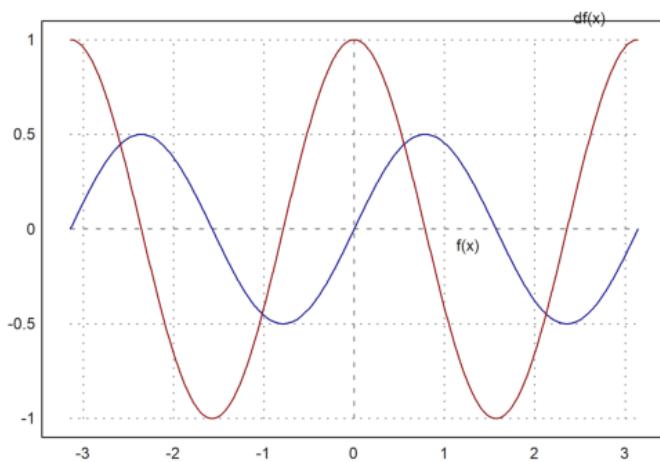
```
>function f(x) := sin(x)*cos(x)
>$showev('limit(((sin(x+h)*cos(x+h))-sin(x)*cos(x))/h,h,0)) // turunan sin(x)*cos(x)
```

$$\lim_{h \rightarrow 0} \frac{\cos(x+h)\sin(x+h) - \cos x \sin x}{h} = \cos^2 x - \sin^2 x$$

```
>function df(x) &= limit(((sin(x+h)*cos(x+h))-sin(x)*cos(x))/h,h,0); $df(x) // df(x) = f'(x)
```

$$\cos^2 x - \sin^2 x$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]), label("f(x)", 1, 0), label("df(x)", 2.3, 1.1)
```



3. Fungsi 3

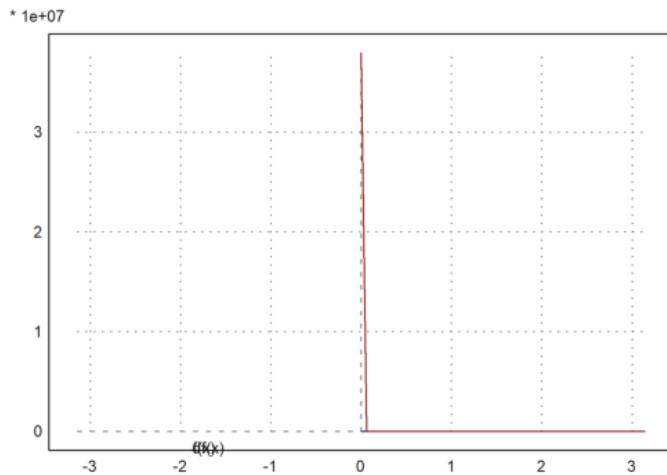
```
>function f(x) := sqrt(x)*4
>$showev('limit((sqrt(x+h)*4-sqrt(x)*4)/h,h,0)) // turunan sqrt(x)*4
```

$$\lim_{h \rightarrow 0} \frac{4\sqrt{x+h} - 4\sqrt{x}}{h} = \frac{2}{\sqrt{x}}$$

```
>function df(x) &= limit((sqrt(x+h)*4-sqrt(x)*4)/h,h,0); $df(x) // df(x) = f'(x)
```

$$\frac{2}{\sqrt{x}}$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]), label("f(x)", -2, 11), label("df(x)", -2, -1)
```



4. Fungsi 4

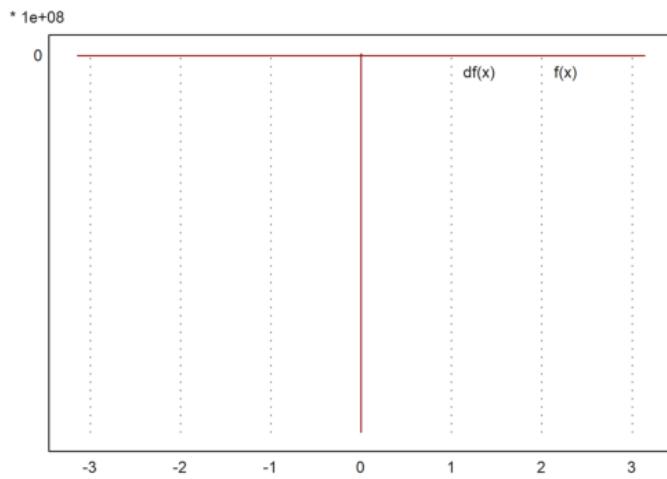
```
>function f(x) := cos(1/x)
>$showev('limit((cos(1/(x+h))-cos(1/x))/h,h,0)) // turunan cos(1/x)
```

$$\lim_{h \rightarrow 0} \frac{\cos\left(\frac{1}{x+h}\right) - \cos\left(\frac{1}{x}\right)}{h} = \frac{\sin\left(\frac{1}{x}\right)}{x^2}$$

```
>function df(x) &= limit((cos(1/(x+h))-cos(1/x))/h,h,0); $df(x) // df(x) = f'(x)
```

$$\frac{\sin\left(\frac{1}{x}\right)}{x^2}$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]), label("f(x)", 2, 0.4), label("df(x)", 1, -0.4)
```



5. Fungsi 5

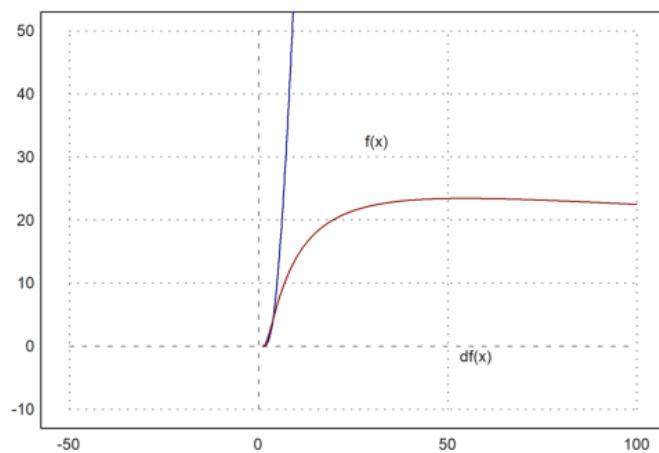
```
>function f(x) := (log(x))^5
>$showev('limit(((log(x+h))^5-(log(x))^5)/h,h,0)) // turunan (log(x))^5
```

$$\lim_{h \rightarrow 0} \frac{\log^5(x+h) - \log^5 x}{h} = \frac{5 \log^4 x}{x}$$

```
>function df(x) &= limit(((log(x+h))^5-(log(x))^5)/h,h,0); $df(x) // df(x) = f'(x)
```

$$\frac{5 \log^4 x}{x}$$

```
>plot2d(["f(x)", "df(x)"], -50, 100, -10, 50, color=[blue, red]), label("f(x)", 25, 35), label("df(x)", 50, -5)
```



6. Fungsi 6

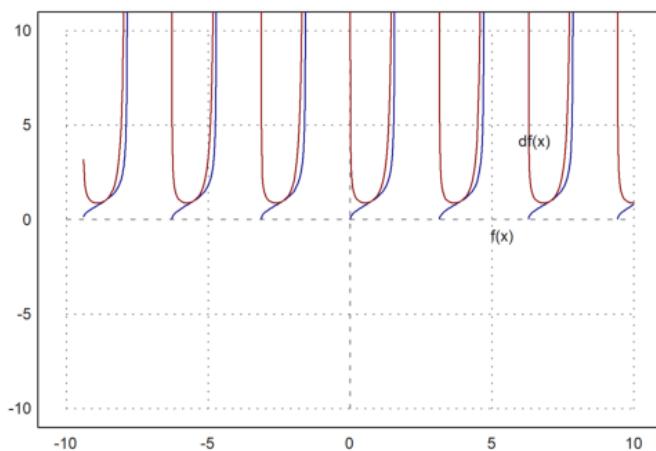
```
>function f(x) := sqrt(tan(x))
>$showev('limit((sqrt(tan(x+h))-sqrt(tan(x)))/h,h,0)) // turunan exp(x)*cos(x)
```

$$\lim_{h \rightarrow 0} \frac{\sqrt{\tan(x+h)} - \sqrt{\tan x}}{h} = \frac{1}{2 \cos^2 x \sqrt{\tan x}}$$

```
>function df(x) &= limit((sqrt(tan(x+h))-sqrt(tan(x)))/h,h,0); $df(x)// df(x) = f'(x)
```

$$\frac{1}{2 \cos^2 x \sqrt{\tan x}}$$

```
>plot2d(["f(x)", "df(x)"], -10, 10, -10, 10, color=[blue, red]), label("f(x)", 4.5, 0), label("df(x)", 5.5, 4)
```



Integral

EMT dapat digunakan untuk menghitung integral, baik integral tak tentu maupun integral tentu. Untuk integral tak tentu (simbolik) sudah tentu EMT menggunakan Maxima, sedangkan untuk perhitungan integral tentu EMT sudah menyediakan beberapa fungsi yang mengimplementasikan algoritma kuadratur (perhitungan integral tentu menggunakan metode numerik).

Pada notebook ini akan ditunjukkan perhitungan integral tentu dengan menggunakan Teorema Dasar Kalkulus:

$$\int_a^b f(x) dx = F(b) - F(a), \quad \text{dengan } F'(x) = f(x).$$

Fungsi untuk menentukan integral adalah `integrate`. Fungsi ini dapat digunakan untuk menentukan, baik integral tentu maupun tak tentu (jika fungsinya memiliki antiderivatif). Untuk perhitungan integral tentu fungsi `integrate` menggunakan metode numerik (kecuali fungsinya tidak integrabel, kita tidak akan menggunakan metode ini).

```
>$showev('integrate(x^n,x))
```

Answering "Is n equal to -1?" with "no"

$$\int x^n \, dx = \frac{x^{n+1}}{n+1}$$

```
>$showev('integrate(1/(1+x),x))
```

$$\int \frac{1}{x+1} \, dx = \log(x+1)$$

```
>$showev('integrate(1/(1+x^2),x))
```

$$\int \frac{1}{x^2+1} \, dx = \arctan x$$

```
>$showev('integrate(1/sqrt(1-x^2),x))
```

$$\int \frac{1}{\sqrt{1-x^2}} \, dx = \arcsin x$$

```
>$showev('integrate(sin(x),x,0,pi))
```

$$\int_0^\pi \sin x \, dx = 2$$

```
>$showev('integrate(sin(x),x,a,b))
```

$$\int_a^b \sin x \, dx = \cos a - \cos b$$

```
>$showev('integrate(x^n,x,a,b))
```

Answering "Is n positive, negative or zero?" with "positive"

$$\int_a^b x^n \, dx = \frac{b^{n+1}}{n+1} - \frac{a^{n+1}}{n+1}$$

```
>$showev('integrate(x^2*sqrt(2*x+1),x))
```

$$\int x^2 \sqrt{2x+1} \, dx = \frac{(2x+1)^{\frac{7}{2}}}{28} - \frac{(2x+1)^{\frac{5}{2}}}{10} + \frac{(2x+1)^{\frac{3}{2}}}{12}$$

```
>$showev('integrate(x^2*sqrt(2*x+1),x,0,2))
```

$$\int_0^2 x^2 \sqrt{2x+1} dx = \frac{25^{\frac{5}{2}}}{21} - \frac{2}{105}$$

```
>${ratsimp(%)
```

$$\int_0^2 x^2 \sqrt{2x+1} dx = \frac{25^{\frac{7}{2}} - 2}{105}$$

```
>${showev('integrate((sin(sqrt(x)+a)*E^sqrt(x))/sqrt(x),x,0,pi^2))
```

$$\int_0^{\pi^2} \frac{\sin(\sqrt{x} + a) e^{\sqrt{x}}}{\sqrt{x}} dx = (-e^\pi - 1) \sin a + (e^\pi + 1) \cos a$$

```
>${factor(%)
```

$$\int_0^{\pi^2} \frac{\sin(\sqrt{x} + a) e^{\sqrt{x}}}{\sqrt{x}} dx = (-e^\pi - 1) (\sin a - \cos a)$$

```
>function map f(x) &= E^(-x^2); $f(x)
```

$$e^{-x^2}$$

```
>${showev('integrate(f(x),x))
```

$$\int e^{-x^2} dx = \frac{\sqrt{\pi} \operatorname{erf}(x)}{2}$$

Fungsi f tidak memiliki antiturunan, integralnya masih memuat integral lain.

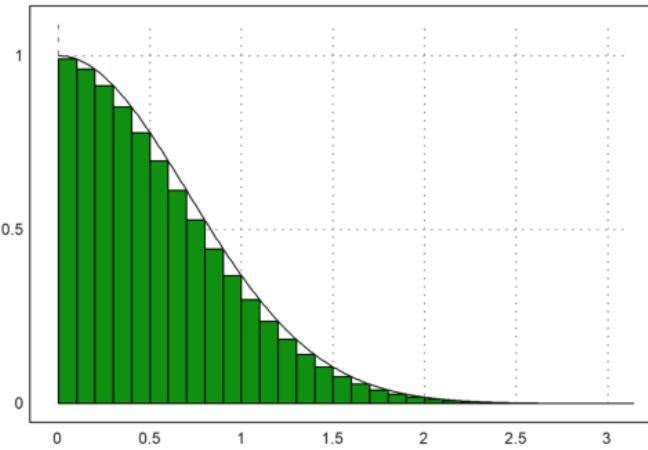
$$\operatorname{erf}(x) = \int \frac{e^{-x^2}}{\sqrt{\pi}} dx.$$

Kita tidak dapat menggunakan teorema Dasar kalkulus untuk menghitung integral tentu fungsi tersebut jika semua batasnya berhingga. Dalam hal ini dapat digunakan metode numerik (rumus kuadratur).

Misalkan kita akan menghitung:

```
maxima: 'integrate(f(x),x,0,pi)
```

```
>x=0:0.1:pi-0.1; plot2d(x,f(x+0.1),>bar); plot2d("f(x)",0,pi,>add):
```



Integral tentu

maxima: 'integrate(f(x),x,0,pi)

dapat dihampiri dengan jumlah luas persegi-persegi panjang di bawah kurva $y=f(x)$ tersebut. Langkah-langkahnya adalah sebagai berikut.

```
>t &= makelist(a,a,0,pi-0.1,0.1); // t sebagai list untuk menyimpan nilai-nilai x
>fx &= makelist(f(t[i]+0.1),i,1,length(t)); // simpan nilai-nilai f(x)
>/> jangan menggunakan x sebagai list, kecuali Anda pakar Maxima!
```

Hasilnya adalah:

maxima: 'integrate(f(x),x,0,pi) = 0.1*sum(fx[i],i,1,length(fx))

Jumlah tersebut diperoleh dari hasil kali lebar sub-subinterval (=0.1) dan jumlah nilai-nilai $f(x)$ untuk $x = 0.1, 0.2, 0.3, \dots, 3.2$.

```
>0.1*sum(f(x+0.1)) // cek langsung dengan perhitungan numerik EMT
```

0.836219610253

Untuk mendapatkan nilai integral tentu yang mendekati nilai sebenarnya, lebar sub-intervalnya dapat diperkecil lagi, sehingga daerah di bawah kurva tertutup semuanya, misalnya dapat digunakan lebar subinterval 0.001. (Silakan dicoba!)

Meskipun Maxima tidak dapat menghitung integral tentu fungsi tersebut untuk batas-batas yang berhingga, namun integral tersebut dapat dihitung secara eksak jika batas-batasnya tak hingga. Ini adalah salah satu keajaiban di dalam matematika, yang terbatas tidak dapat dihitung secara eksak, namun yang tak hingga malah dapat dihitung secara eksak.

```
>$showev('integrate(f(x),x,0,inf))
```

$$\int_0^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

Berikut adalah contoh lain fungsi yang tidak memiliki antiderivatif, sehingga integral tentunya hanya dapat dihitung dengan metode numerik.

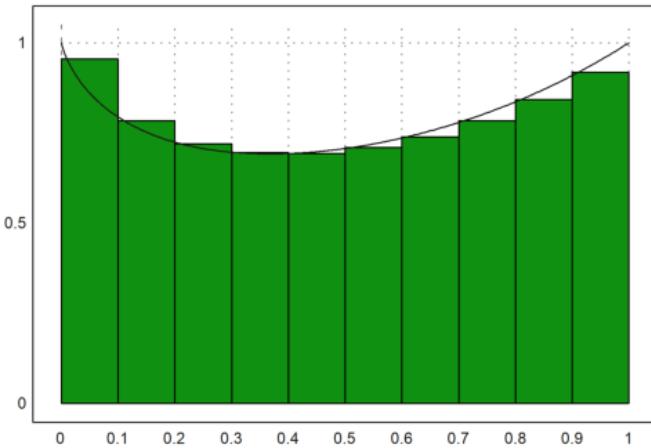
```
>function f(x) &= x^x; $f(x)
```

x^x

```
>$showev('integrate(f(x),x,0,1))
```

$$\int_0^1 x^x \, dx = \int_0^1 x^x \, dx$$

```
>x=0:0.1:1-0.01; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,1,>add):
```



Maxima gagal menghitung integral tentu tersebut secara langsung menggunakan perintah integrate. Berikut kita lakukan seperti contoh sebelumnya untuk mendapat hasil atau pendekatan nilai integral tentu tersebut.

```
>t &= makelist(a,a,0,1-0.01,0.01);
>fx &= makelist(f(t[i]+0.01),i,1,length(t));
```

Latihan

- Bukalah buku Kalkulus.
- Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi).
- Untuk setiap fungsi, tentukan anti turunannya (jika ada), hitunglah integral tentu dengan batas-batas yang menarik (Anda tentukan sendiri), seperti contoh-contoh tersebut.
- Lakukan hal yang sama untuk fungsi-fungsi yang tidak dapat diintegralkan (cari sedikitnya 3 fungsi).
- Gambar grafik fungsi dan daerah integrasinya pada sumbu koordinat yang sama.
- Gunakan integral tentu untuk mencari luas daerah yang dibatasi oleh dua kurva yang berpotongan di dua titik. (Cari dan gambar kedua kurva dan arsir (warnai) daerah yang dibatasi oleh keduanya.)
- Gunakan integral tentu untuk menghitung volume benda putar kurva $y = f(x)$ yang diputar mengelilingi sumbu x dari $x=a$ sampai $x=b$, yakni

$$V = \int_a^b \pi(f(x))^2 \, dx.$$

(Pilih fungsinya dan gambar kurva dan benda putar yang dihasilkan. Anda dapat mencari contoh-contoh bagaimana cara menggambar benda hasil perputaran suatu kurva.)

- Gunakan integral tentu untuk menghitung panjang kurva $y=f(x)$ dari $x=a$ sampai $x=b$ dengan menggunakan rumus:

$$S = \int_a^b \sqrt{1 + (f'(x))^2} dx.$$

(Pilih fungsi dan gambar kurvanya.)

Jawab:

1. Fungsi 1

```
>function f(x) &= 5*x^2; $f(x)
```

$$5x^2$$

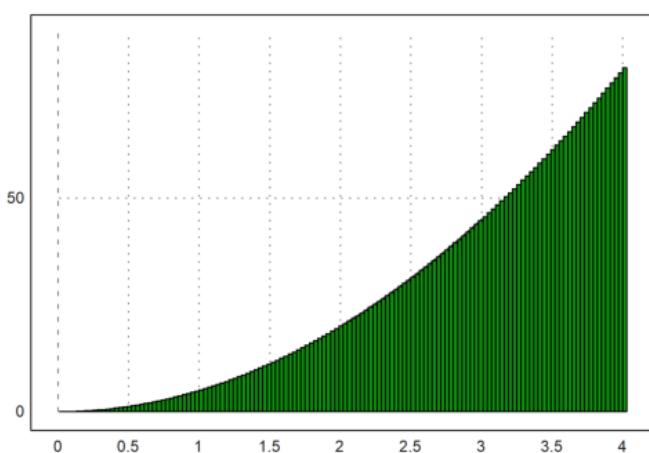
```
>$showev('integrate(f(x),x))
```

$$5 \int x^2 dx = \frac{5x^3}{3}$$

```
>$showev('integrate(f(x),x,2,3))
```

$$5 \int_2^3 x^2 dx = \frac{95}{3}$$

```
>x=0.01:0.03:4; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",2,3,>add):
```



2. Fungsi 2

```
>function f(x) &= cos(2*x+5); $f(x)
```

$$\cos(2x + 5)$$

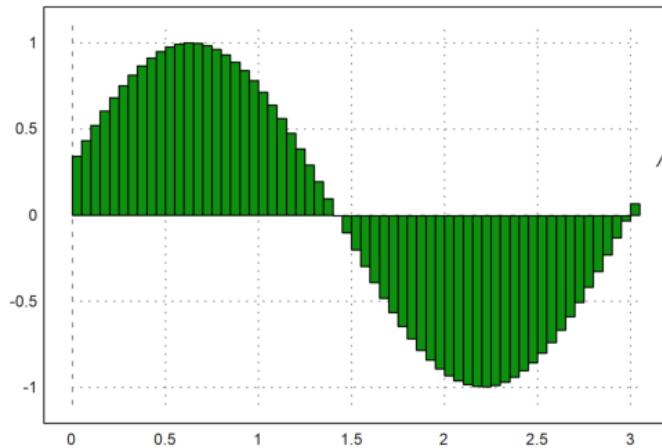
```
>showev('integrate(f(x),x))
```

$$\int \cos(2x + 5) dx = \frac{\sin(2x + 5)}{2}$$

```
>showev('integrate(f(x),x,pi,2*pi))
```

$$\int_{\pi}^{2\pi} \cos(2x + 5) dx = 0$$

```
>x=0:0.05:pi-0.1; plot2d(x,f(x+0.03),>bar); plot2d("f(x)",pi,2*pi,>add):
```



3. Fungsi 3

```
>function f(x) &= (sin(x)) * (cos((x))^2); $f(x)
```

$$\cos^2 x \sin x$$

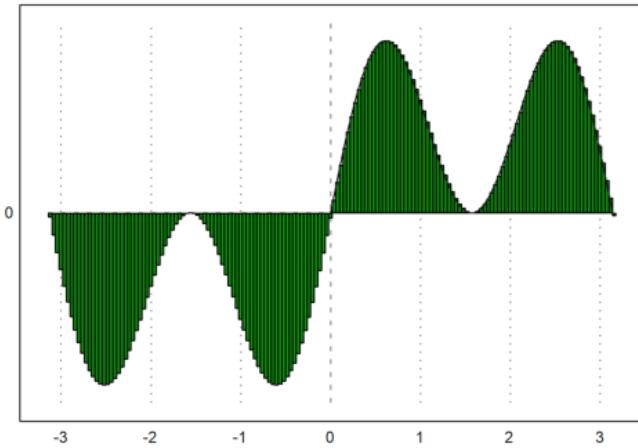
```
>showev('integrate(f(x),x))
```

$$\int \cos^2 x \sin x dx = -\frac{\cos^3 x}{3}$$

```
>showev('integrate(f(x),x,0,pi))
```

$$\int_0^\pi \cos^2 x \sin x dx = \frac{2}{3}$$

```
>x=-pi:0.04:pi; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,pi,>add):
```



4. Fungsi 4

```
>function f(x) &= (x^2*(2-x^3)^(1/2)); $f(x)
```

$$x^2 \sqrt{2 - x^3}$$

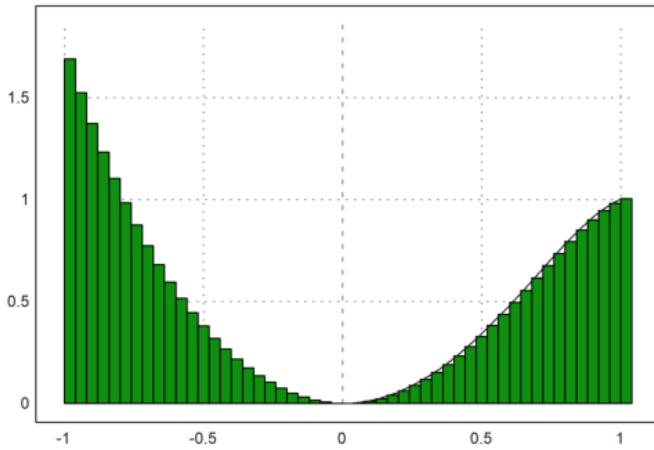
```
>$showev('integrate(f(x),x))
```

$$\int x^2 \sqrt{2 - x^3} dx = -\frac{2 (2 - x^3)^{\frac{3}{2}}}{9}$$

```
>$showev('integrate(f(x),x,0,1))
```

$$\int_0^1 x^2 \sqrt{2 - x^3} dx = \frac{2^{\frac{5}{2}}}{9} - \frac{2}{9}$$

```
>x=-1:0.04:1; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,1,>add):
```



5. Fungsi 5

```
>function f(x) &= sqrt(24-x^2); $f(x)
```

$$\sqrt{24 - x^2}$$

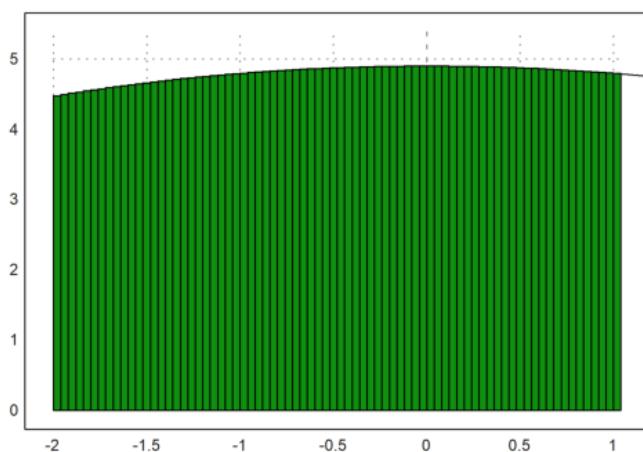
```
>$showev('integrate(f(x),x))
```

$$\int \sqrt{24 - x^2} dx = 12 \arcsin\left(\frac{x}{2\sqrt{6}}\right) + \frac{x\sqrt{24 - x^2}}{2}$$

```
>$showev('integrate(f(x),x,1,2))
```

$$\int_1^2 \sqrt{24 - x^2} dx = 12 \arcsin\left(\frac{1}{\sqrt{6}}\right) - \frac{24 \arcsin\left(\frac{1}{2\sqrt{6}}\right) + \sqrt{23}}{2} + 2\sqrt{5}$$

```
>x=-2:0.04:1; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",1,2,>add):
```

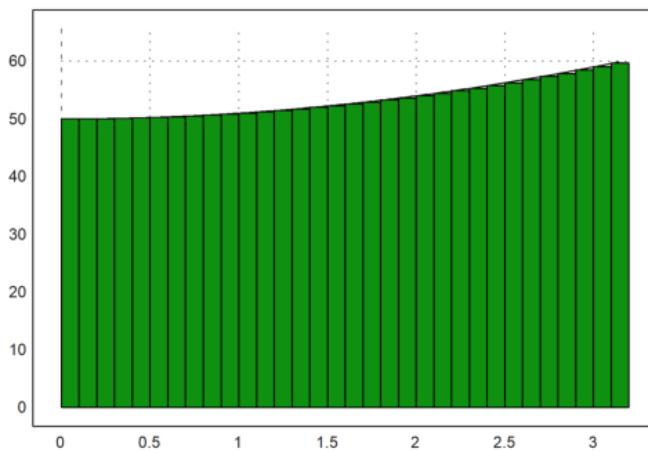


6. Fungsi 6

```
>t &= makelist(a,a,0,1-0.01,0.01);
>fx &= makelist(f(t[i]+0.01),i,1,length(t));
>function f(x) &= x^2+50; $f(x)
```

$$x^2 + 50$$

```
>x=0:0.1:pi-0.01; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,pi,>add):
```



```
>0.01*sum(f(x+0.01))
```

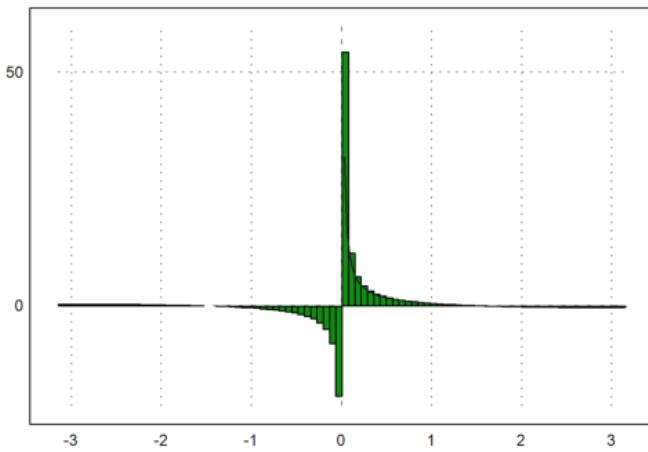
17.051552

7. Fungsi 7

```
>t &= makelist(a,a,0,1-0.01,0.01);
>fx &= makelist(f(t[i]+0.01),i,1,length(t));
>function f(x) &= cos(x)/x; $f(x)
```

$$\frac{\cos x}{x}$$

```
>x=-pi:0.07:pi-0.01; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,pi,>add):
```



```
>0.01*sum(f(x+0.01))
```

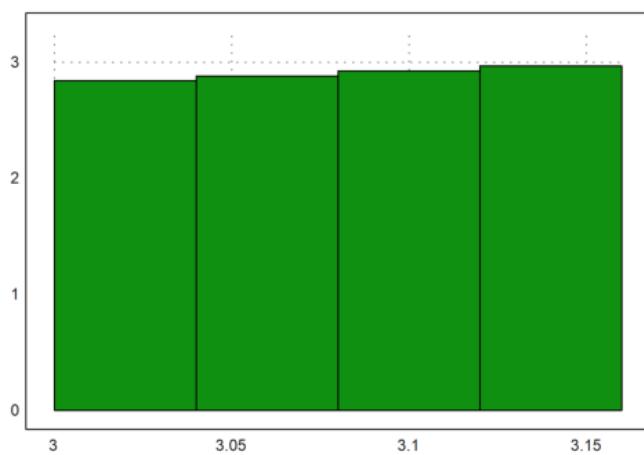
0.415163991256

8. Fungsi 8

```
>t &= makelist(a,a,0,1-0.01,0.01);
>fx &= makelist(f(t[i]+0.01),i,1,length(t));
>function f(x) &= sqrt(x^2-1); $f(x)
```

$$\sqrt{x^2 - 1}$$

```
>x=3:0.04:pi-0.01; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,2,>add):
```



```
>0.01*sum(f(x+0.01))
```

0.11610107668

Luas daerah dibatasi 2 kurva

1). Fungsi 1

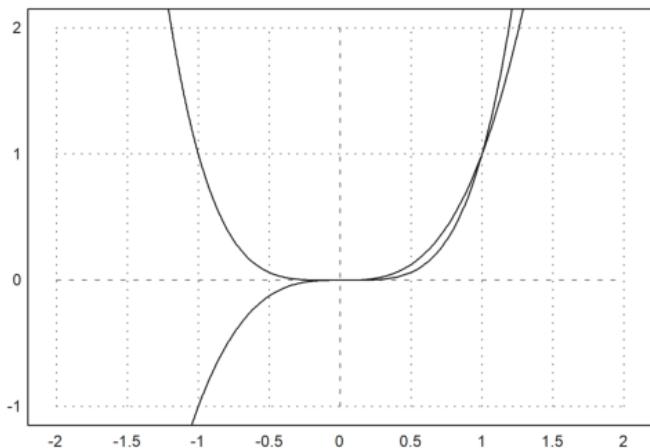
```
>function f(x) &= x^3; $f(x)
```

$$x^3$$

```
>function g(x) &= x; $g(x)
```

$$x$$

```
>plot2d(["x^4", "x^3"], -2, 2, -1, 2);
```



```
>function h(x) &= f(x)-g(x); $h(x)
```

$$x^3 - x$$

```
>$showev('integrate(h(x),x))
```

$$\int x^3 - x \, dx = \frac{x^4}{4} - \frac{x^2}{2}$$

```
>$&solve(f(x)=g(x))
```

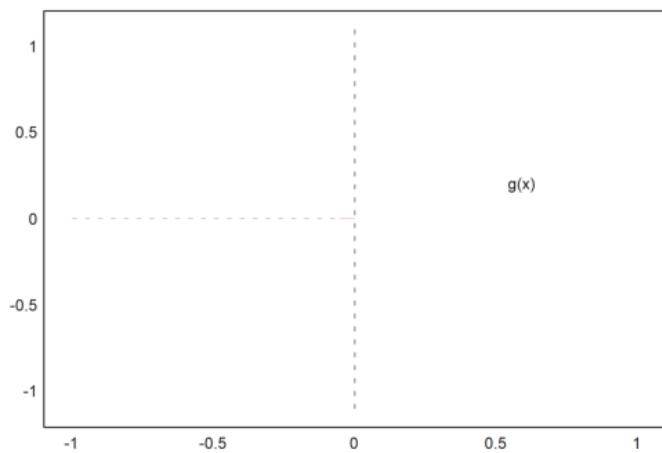
$$[x = -1, x = 1, x = 0]$$

```
>$showev('integrate(h(x),x,0,1)) // menghitung luas daerah yang dibatasi 2 kurva
```

$$\int_0^1 x^3 - x \, dx = -\frac{1}{4}$$

Arsiran daerah yang dibatasi kurva $f(x)$ dan $g(x)$ sebagai berikut:

```
>x=-1:0.01:1; plot2d(x,f(x),>bar,>filled,style="--",fillcolor=orange,>grid); plot2d(x,g(x),
```



2). Fungsi 2

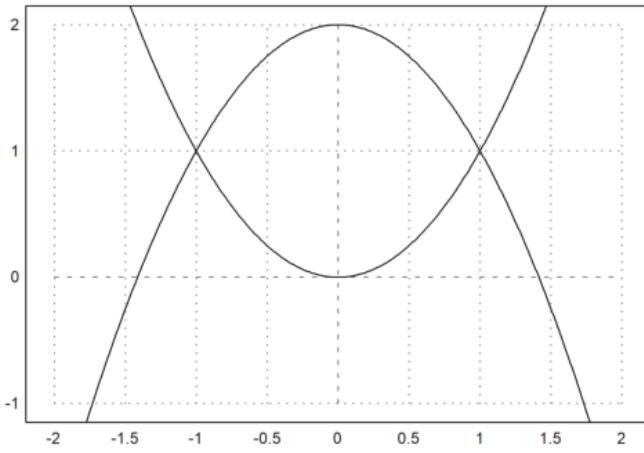
```
>function f(x) &= x^3+1; $f(x)
```

$$x^3 + 1$$

```
>function g(x) &= x^2; $g(x)
```

$$x^2$$

```
>plot2d([-x^2+2, "x^2"], -2, 2, -1, 2):
```



```
>function h(x) &= f(x)-g(x); $h(x)
```

$$x^3 - x^2 + 1$$

```
>$&solve(f(x)=g(x))
```

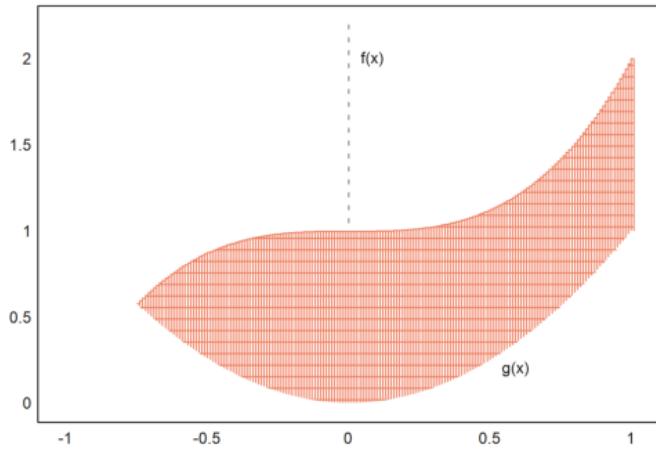
$$\left[x = \frac{\frac{\sqrt{3}i}{2} - \frac{1}{2}}{9\left(\frac{\sqrt{23}}{23^{\frac{3}{2}}} - \frac{25}{54}\right)^{\frac{1}{3}}} + \left(\frac{\sqrt{23}}{23^{\frac{3}{2}}} - \frac{25}{54}\right)^{\frac{1}{3}} \left(-\frac{\sqrt{3}i}{2} - \frac{1}{2}\right) + \frac{1}{3}, x = \left(\frac{\sqrt{23}}{23^{\frac{3}{2}}} - \frac{25}{54}\right)^{\frac{1}{3}} \left(\frac{\sqrt{3}i}{2} - \frac{1}{2}\right) + \frac{-\frac{\sqrt{3}i}{2} - \frac{1}{2}}{9\left(\frac{\sqrt{23}}{23^{\frac{3}{2}}} - \frac{25}{54}\right)^{\frac{1}{3}}} + \frac{1}{3}, x = \right]$$

```
>$showev('integrate(h(x),x,-1,1)) // menghitung luas daerah yang dibatasi 2 kurva
```

$$\int_{-1}^1 x^3 - x^2 + 1 \, dx = \frac{4}{3}$$

Arsiran daerah yang dibatasi kurva $f(x)$ dan $g(x)$ sebagai berikut:

```
>x=-1:0.01:1; plot2d(x,f(x),>bar,>filled,style="--",fillcolor=orange,>grid); plot2d(x,g(x),
```



Volume benda putar

Menghitung volume hasil perputaran kurva

$$m(x) = x^4 + 3$$

dari $x=-1$ sampai $x=0$. Diputar terhadap sumbu-x.

Jawab:

```
>function m(x) &= x^4+3; $m(x)
```

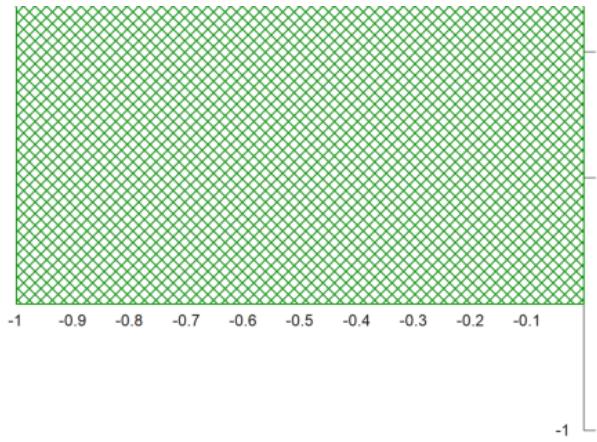
$$x^4 + 3$$

```
>$showev('integrate(pi*(m(x))^2,x,-1,0)) // Menghitung volume hasil perputaran m(x)
```

$$\pi \int_{-1}^0 (x^4 + 3)^2 dx = \frac{464\pi}{45}$$

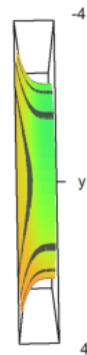
Daerah di bawah kurva yang akan dirotasi terhadap sumbu x sebagai berikut:

```
>plot2d("m(x)",-1,0,-1,2,grid=7,>filled, style="/\"):
```



Hasil perputaran m(x) terhadap sumbu x sebagai berikut:

```
>plot3d("m(x)",-1,0,-1,1,>rotate,angle=6.3,>hue,>contour,color=redgreen,height=11):
```



Menghitung panjang kurva

Menghitung panjang kurva

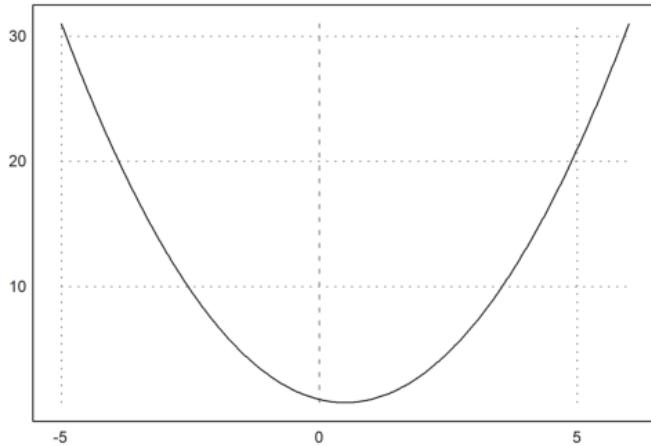
$$y = x^2 - x + 1$$

dari $x=1$ sampai $x=3$.

```
>function d(x) &= x^2-x+1; $d(x)
```

$$x^2 - x + 1$$

```
>plot2d("d(x)", -5, 6); // gambar kurva d(x)
```



```
>$showev('limit((d(x+h)-d(x))/h,h,0))
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2 - h}{h} = 2x - 1$$

```
>function dd(x) &= limit((d(x+h)-d(x))/h,h,0); $dd(x)
```

$$2x - 1$$

```
>function q(x) &= ((dd(x))^2); $q(x)
```

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

```
>$showev('integrate(sqrt(1+q(x)),x,1,3)) // menghitung panjang kurva
```

$$\int_1^3 \sqrt{(2x-1)^2 + 1} dx = \frac{\operatorname{asinh} 5 + 5\sqrt{26}}{4} - \frac{\operatorname{asinh} 1 + \sqrt{2}}{4}$$

Jadi, panjang kurva

$$y = x^2 - x + 1$$

dari x=0 sampai x=4 adalah

$$S = \frac{\operatorname{asinh} 5 + 5\sqrt{26}}{4} - \frac{\operatorname{asinh} 1 + \sqrt{2}}{4}.$$

Barisan dan Deret

(Catatan: bagian ini belum lengkap. Anda dapat membaca contoh-contoh penggunaan EMT dan Maxima untuk menghitung limit barisan, rumus jumlah parsial suatu deret, jumlah tak hingga suatu deret konvergen, dan sebagainya. Anda dapat mengeksplor contoh-contoh di EMT atau perbagai panduan penggunaan Maxima di software Maxima atau dari Internet.)

Barisan dapat didefinisikan dengan beberapa cara di dalam EMT, di antaranya:

- dengan cara yang sama seperti mendefinisikan vektor dengan elemen-elemen beraturan (menggunakan titik dua ":");
- menggunakan perintah "sequence" dan rumus barisan (suku ke -n);
- menggunakan perintah "iterate" atau "niterate";
- menggunakan fungsi Maxima "create_list" atau "makelist" untuk menghasilkan barisan simbolik;
- menggunakan fungsi biasa yang inputnya vektor atau barisan;
- menggunakan fungsi rekursif.

EMT menyediakan beberapa perintah (fungsi) terkait barisan, yakni:

- sum: menghitung jumlah semua elemen suatu barisan
- cumsum: jumlah kumulatif suatu barisan
- differences: selisih antar elemen-elemen berturutan

EMT juga dapat digunakan untuk menghitung jumlah deret berhingga maupun deret tak hingga, dengan menggunakan perintah (fungsi) "sum". Perhitungan dapat dilakukan secara numerik maupun simbolik dan eksak.

Berikut adalah beberapa contoh perhitungan barisan dan deret menggunakan EMT.

```
>1:10 // barisan sederhana
```

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

```
>1:2:30
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
```

```
>sum(1:2:30), sum(1/(1:2:30))
```

```
225
```

```
2.33587263431
```

```
>$' sum(k, k, 1, n) = factor(ev(sum(k, k, 1, n), simpsum=true)) // simpsum:menghitung deret
```

$$\sum_{k=1}^n k = \frac{n(n+1)}{2}$$

```
>$' sum(1/(3^k+k), k, 0, inf) = factor(ev(sum(1/(3^k+k), k, 0, inf), simpsum=true))
```

$$\sum_{k=0}^{\infty} \frac{1}{3^k + k} = \sum_{k=0}^{\infty} \frac{1}{3^k + k}$$

Di sini masih gagal, hasilnya tidak dihitung.

```
>$' sum(1/x^2, x, 1, inf) = ev(sum(1/x^2, x, 1, inf), simpsum=true) // ev: menghitung nilai e
```

$$\sum_{x=1}^{\infty} \frac{1}{x^2} = \frac{\pi^2}{6}$$

```
>$' sum((-1)^(k-1)/k, k, 1, inf) = factor(ev(sum((-1)^(x-1)/x, x, 1, inf), simpsum=true))
```

$$\sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} = - \sum_{x=1}^{\infty} \frac{(-1)^x}{x}$$

Di sini masih gagal, hasilnya tidak dihitung.

```
>$' sum((-1)^k/(2*k-1), k, 1, inf) = factor(ev(sum((-1)^k/(2*k-1), k, 1, inf), simpsum=true))
```

$$\sum_{k=1}^{\infty} \frac{(-1)^k}{2k-1} = \sum_{k=1}^{\infty} \frac{(-1)^k}{2k-1}$$

```
>$ev(sum(1/n!, n, 0, inf), simpsum=true)
```

$$\sum_{n=0}^{\infty} \frac{1}{n!}$$

Di sini masih gagal, hasilnya tidak dihitung, harusnya hasilnya e.

```
>&assume(abs(x)<1); $' sum(a*x^k, k, 0, inf)=ev(sum(a*x^k, k, 0, inf), simpsum=true), &forget
```

$$a \sum_{k=0}^{\infty} x^k = \frac{a}{1-x}$$

Deret geometri tak hingga, dengan asumsi rasional antara -1 dan 1. **Deret Taylor**

Deret Taylor suatu fungsi f yang diferensiabel sampai tak hingga di sekitar $x=a$ adalah:

$$f(x) = \sum_{k=0}^{\infty} \frac{(x-a)^k f^{(k)}(a)}{k!}.$$

```
>$' e^x=taylor(exp(x),x,0,10) // deret Taylor e^x di sekitar x=0, sampai suku ke-11
```

$$e^x = \frac{x^{10}}{3628800} + \frac{x^9}{362880} + \frac{x^8}{40320} + \frac{x^7}{5040} + \frac{x^6}{720} + \frac{x^5}{120} + \frac{x^4}{24} + \frac{x^3}{6} + \frac{x^2}{2} + x + 1$$

```
>$' log(x)=taylor(log(x),x,1,10)// deret log(x) di sekitar x=1
```

$$\log x = x - \frac{(x-1)^{10}}{10} + \frac{(x-1)^9}{9} - \frac{(x-1)^8}{8} + \frac{(x-1)^7}{7} - \frac{(x-1)^6}{6} + \frac{(x-1)^5}{5} - \frac{(x-1)^4}{4} + \frac{(x-1)^3}{3} - \frac{(x-1)^2}{2} - 1$$

BAB 5

KB PEKAN 8: MENGGUNAKAN EMT UNTUK GEOMETRI

[a4paper,10pt]article eumat

Visualisasi dan Perhitungan Geometri dengan EMT

Euler menyediakan beberapa fungsi untuk melakukan visualisasi dan perhitungan geometri, baik secara numerik maupun analitik (seperti biasanya tentunya, menggunakan Maxima). Fungsi-fungsi untuk visualisasi dan perhitungan geometri tersebut disimpan di dalam file program "geometry.e", sehingga file tersebut harus dipanggil sebelum menggunakan fungsi-fungsi atau perintah-perintah untuk geometri.

```
>load geometry
```

Numerical and symbolic geometry.

Fungsi-fungsi Geometri

Fungsi-fungsi untuk Menggambar Objek Geometri:

```
defaultd:=textheight()*1.5: nilai asli untuk parameter d
setPlotrange(x1,x2,y1,y2): menentukan rentang x dan y pada bidang koordinat
setPlotRange(r): pusat bidang koordinat (0,0) dan batas-batas sumbu-x dan y adalah -r sd
plotPoint (P, "P"): menggambar titik P dan diberi label "P"
plotSegment (A,B, "AB", d): menggambar ruas garis AB, diberi label "AB" sejauh d
plotLine (g, "g", d): menggambar garis g diberi label "g" sejauh d
plotCircle (c,"c",v,d): Menggambar lingkaran c dan diberi label "c"
plotLabel (label, P, V, d): menuliskan label pada posisi P
```

Fungsi-fungsi Geometri Analitik (numerik maupun simbolik):

```
turn(v, phi): memutar vektor v sejauh phi
turnLeft (v): memutar vektor v ke kiri
turnRight (v): memutar vektor v ke kanan
normalize(v): normal vektor v
crossProduct(v, w): hasil kali silang vektorv dan w.
lineThrough(A, B): garis melalui A dan B, hasilnya [a,b,c] sdh. ax+by=c.
```

```

lineWithDirection(A, v): garis melalui A searah vektor v
getLineDirection(g): vektor arah (gradien) garis g
getNormal(g): vektor normal (tegak lurus) garis g
getPointOnLine(g): titik pada garis g
perpendicular(A, g): garis melalui A tegak lurus garis g
parallel (A, g): garis melalui A sejajar garis g
lineIntersection(g, h): titik potong garis g dan h
projectToLine(A, g): proyeksi titik A pada garis g
distance(A, B): jarak titik A dan B
distanceSquared(A, B): kuadrat jarak A dan B
quadrance(A, B): kuadrat jarak A dan B
areaTriangle(A, B, C): luas segitiga ABC
computeAngle(A, B, C): besar sudut <ABC
angleBisector(A, B, C): garis bagi sudut <ABC
circleWithCenter (A, r): lingkaran dengan pusat A dan jari-jari r
getCircleCenter(c): pusat lingkaran c
getCircleRadius(c): jari-jari lingkaran c
circleThrough(A,B,C): lingkaran melalui A, B, C
middlePerpendicular(A, B): titik tengah AB
lineCircleIntersections(g, c): titik potong garis g dan lingkaran c
circleCircleIntersections (c1, c2): titik potong lingkaran c1 dan c2
planeThrough(A, B, C): bidang melalui titik A, B, C

```

Fungsi-fungsi Khusus Untuk Geometri Simbolik:

```

getLineEquation (g,x,y): persamaan garis g dinyatakan dalam x dan y
getHesseForm (g,x,y,A): bentuk Hesse garis g dinyatakan dalam x dan y dengan titik A pada

```

sisi positif (kanan/atasi) garis

```

quad(A,B): kuadrat jarak AB
spread(a,b,c): Spread segitiga dengan panjang sisi-sisi a,b,c, yakni sin(alpha)^2 dengan

```

alpha sudut yang menghadap sisi a.

```

crosslaw(a,b,c,sa): persamaan 3 quads dan 1 spread pada segitiga dengan panjang sisi a,
c.

```

```

triplespread(sa,sb,sc): persamaan 3 spread sa,sb,sc yang memebntuk suatu segitiga
doublespread(sa): Spread sudut rangkap Spread 2*phi, dengan sa=sin(phi)^2 spread a.

```

Contoh 1: Luas, Lingkaran Luar, Lingkaran Dalam Segitiga

Untuk menggambar objek-objek geometri, langkah pertama adalah menentukan rentang sumbu-sumbu koordinat. Semua objek geometri akan digambar pada satu bidang koordinat, sampai didefinisikan bidang koordinat yang baru.

```
>setPlotRange(-0.5,2.5,-0.5,2.5); // mendefinisikan bidang koordinat baru
```

Sekarang atur tiga poin dan plot.

```
>A=[1,0]; plotPoint(A, "A"); // definisi dan gambar tiga titik  
>B=[0,1]; plotPoint(B, "B");  
>C=[2,2]; plotPoint(C, "C");
```

Lalu tiga segmen.

```
>plotSegment(A,B, "c"); // c=AB  
>plotSegment(B,C, "a"); // a=BC  
>plotSegment(A,C, "b"); // b=AC
```

Fungsi geometri meliputi fungsi untuk membuat garis dan lingkaran.

Format untuk garis adalah $[a, b, c]$, yang merepresentasikan garis dengan persamaan $ax + by = c$

```
>lineThrough(B,C) // garis yang melalui B dan C
```

$[-1, 2, 2]$

Hitung garis tegak lurus melalui A pada BC.

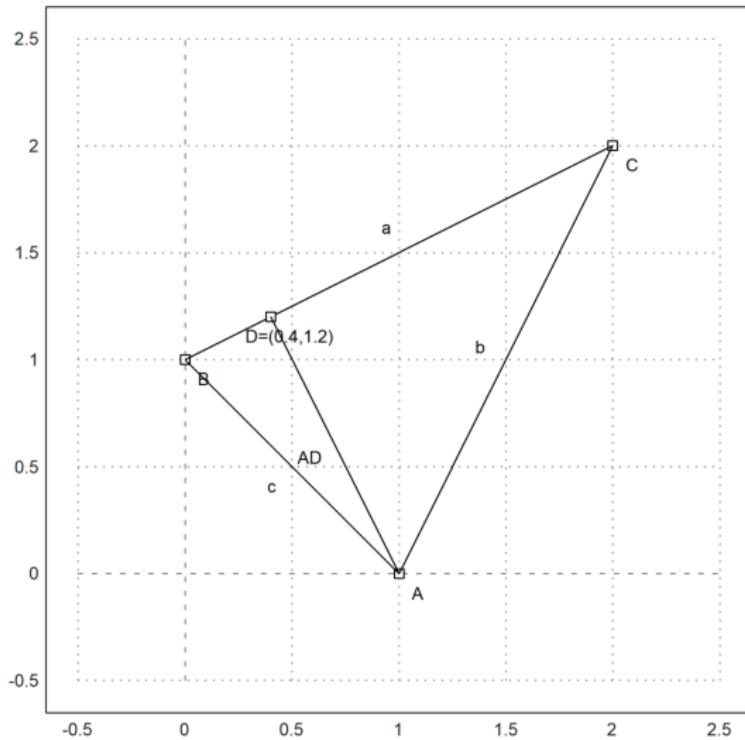
```
>h=perpendicular(A,lineThrough(B,C)); // garis h tegak lurus BC melalui A
```

Dan perpotongannya dengan BC.

```
>D=lineIntersection(h,lineThrough(B,C)); // D adalah titik potong h dan BC
```

Plot itu.

```
>plotPoint(D,value=1); // koordinat D ditampilkan  
>aspect(1); plotSegment(A,D); // tampilkan semua gambar hasil plot...()
```



Hitung luas ABC:

$$L_{\triangle ABC} = \frac{1}{2} AD \cdot BC.$$

```
>norm(A-D)*norm(B-C)/2 // AD=norm(A-D), BC=norm(B-C)
```

1.5

Bandingkan dengan rumus determinan.

```
>areaTriangle(A,B,C) // hitung luas segitiga langsung dengan fungsi
```

1.5

Cara lain menghitung luas segitiga ABC:

```
>distance(A,D)*distance(B,C)/2
```

1.5

sudut di C.

```
>degsprint(computeAngle(B,C,A))
```

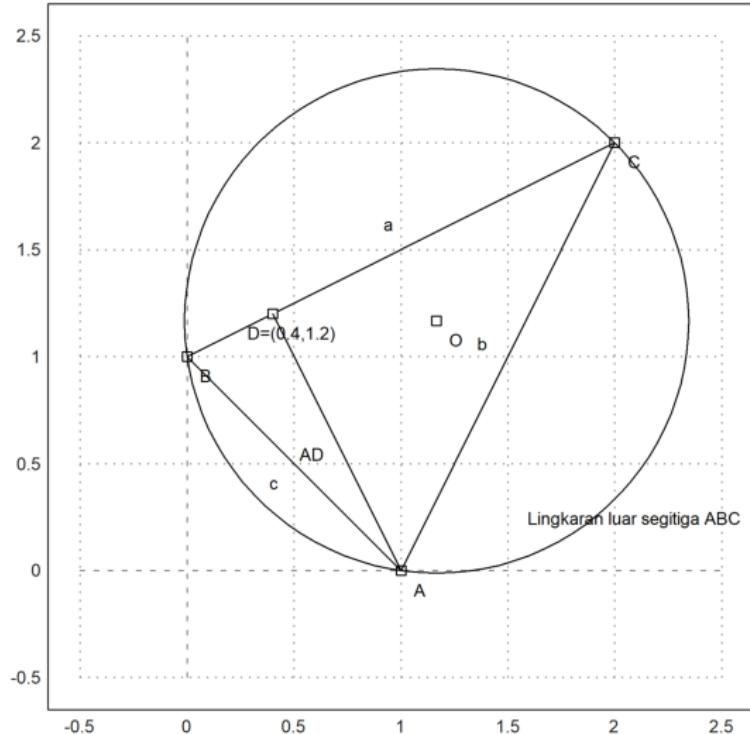
$36^\circ 52' 11.63''$

Sekarang lingkaran sirkuit segitiga.

```

>c=circleThrough(A,B,C); // lingkaran luar segitiga ABC
>R=getCircleRadius(c); // jari2 lingkaran luar
>O=getCircleCenter(c); // titik pusat lingkaran c
>plotPoint(O,"O"); // gambar titik "O"
>plotCircle(c,"Lingkaran luar segitiga ABC");

```



Tampilkan koordinat titik pusat dan jari-jari lingkaran luar.

```
>O, R
```

```
[1.16667, 1.16667]
1.17851130198
```

Sekarang akan digambar lingkaran dalam segitiga ABC. Titik pusat lingkaran dalam adalah titik potong garis-garis bagi sudut.

```

>l=angleBisector(A,C,B); // garis bagi <ACB
>g=angleBisector(C,A,B); // garis bagi <CAB
>P=lineIntersection(l,g) // titik potong kedua garis bagi sudut

```

```
[0.86038, 0.86038]
```

Tambahkan semuanya ke plot.

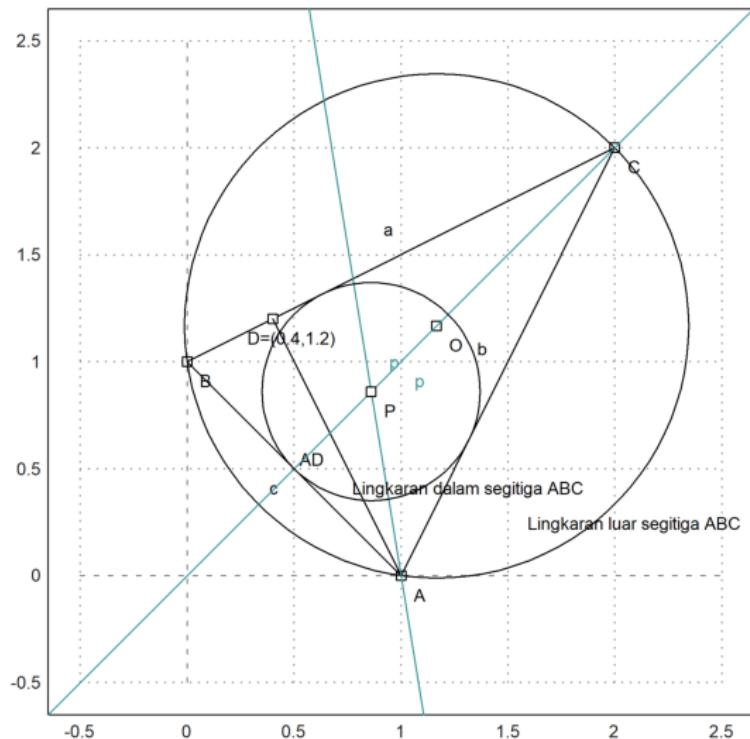
```

>color(5); plotLine(l); plotLine(g); color(1); // gambar kedua garis bagi sudut
>plotPoint(P,"P"); // gambar titik potongnya
>r=norm(P-projectToLine(P,lineThrough(A,B))) // jari-jari lingkaran dalam

```

0.509653732104

```
>plotCircle(circleWithCenter(P,r), "Lingkaran dalam segitiga ABC"); // gambar lingkaran dal
```



Latihan

1. Tentukan ketiga titik singgung lingkaran dalam dengan sisi-sisi segitiga ABC.
2. Gambar segitiga dengan titik-titik sudut ketiga titik singgung tersebut.
3. Tunjukkan bahwa garis bagi sudut yang ke tiga juga melalui titik pusat lingkaran dalam.
4. Gambar jari-jari lingkaran dalam.

Jawab :

- 1.) Titik singgung BC dengan lingkaran dalam

```
>s=lineThrough(B,C)
```

[-1, 2, 2]

```
>m=circleWithCenter(P,r)
```

[0.86038, 0.86038, 0.509654]

```
>S=lineCircleIntersections(s,m)
```

[0.632456, 1.31623]

Titik singgung garis AC dengan lingkaran dalam

```
>p=lineThrough (A, C)
```

```
[ -2, 1, -2]
```

```
>Q=lineCircleIntersections (p, m)
```

```
[ 1.31623, 0.632456]
```

```
>q=lineThrough (A, B)
```

```
[ -1, -1, -1]
```

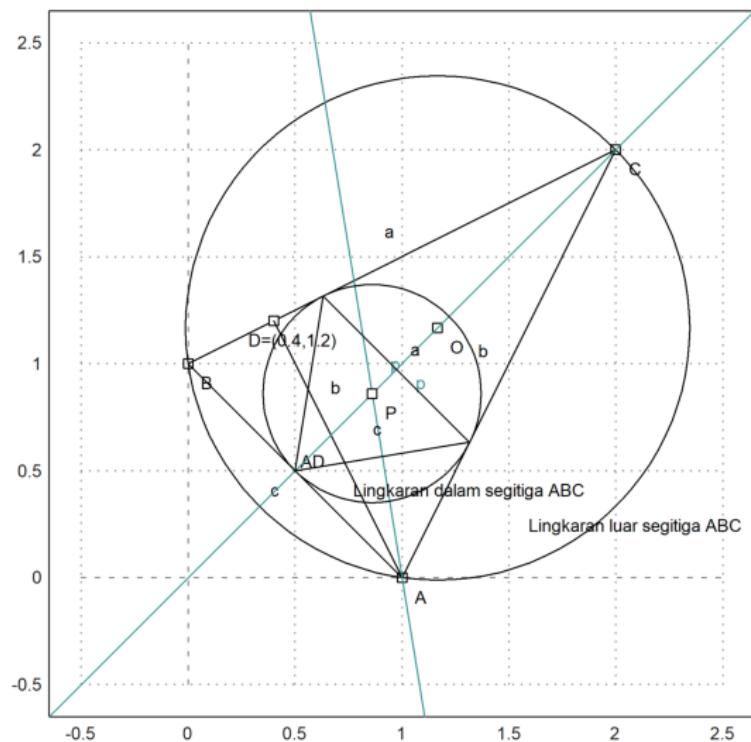
```
>L=lineCircleIntersections (q, m)
```

```
[ 0.5, 0.5]
```

jadi titik singgung lingkaran dalam dengan sisi-sisi segitiga adalah
 $(0.632456, 1.31623)$, $(1.31623, 0.632456)$, dan $(0.5, 0.5)$

2.)

```
>plotSegment (S, Q, "a");
>plotSegment (S, L, "b");
>plotSegment (L, Q, "c"):
```



3.)

```
>P, r
```

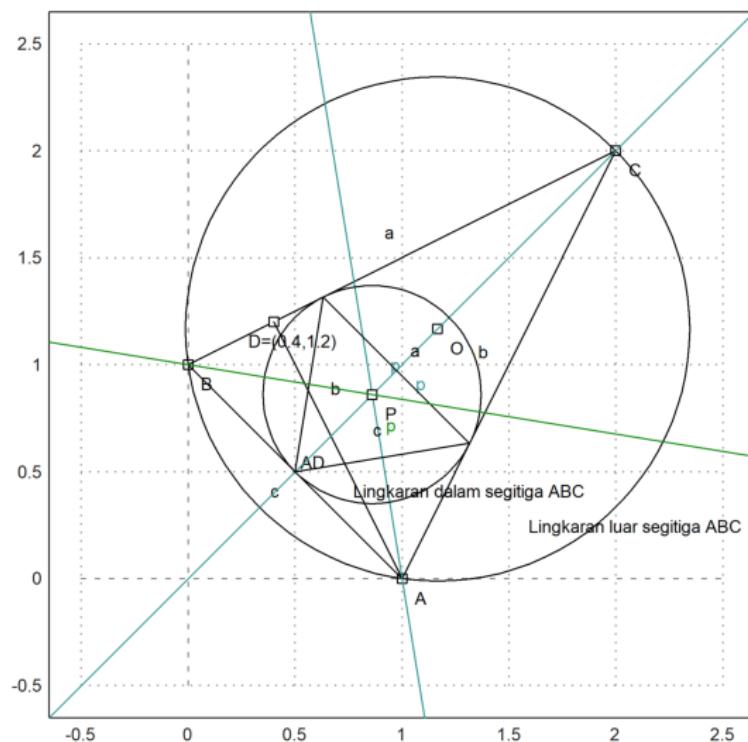
[0.86038, 0.86038]

0.509653732104

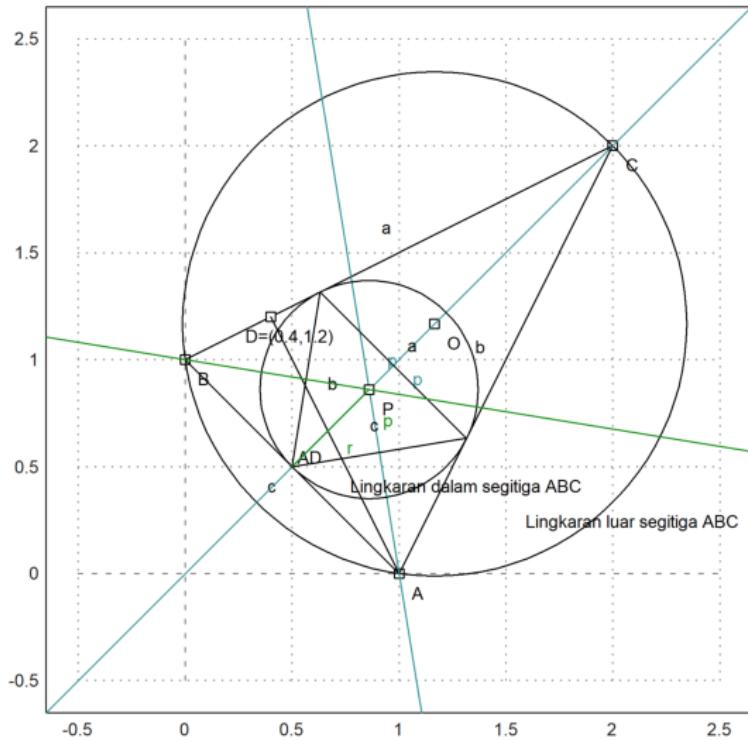
```
>k=angleBisector(A,B,C)
```

[-0.264911, -1.63246, -1.63246]

```
>color(3); plotLine(k):
```



```
>plotSegment(P,L,"r"):
```



Contoh 2: Geometri Simbolik

Kita dapat menghitung geometri tepat dan simbolis menggunakan Maxima.

Geometri file.e menyediakan fungsi yang sama (dan lebih banyak lagi) di Maxima. Namun, sekarang kita dapat menggunakan perhitungan simbolik.

```
>A &= [1,0]; B &= [0,1]; C &= [2,2]; // menentukan tiga titik A, B, C
```

Fungsi garis dan lingkaran bekerja seperti fungsi Euler, tetapi menyediakan penghitungan simbolik.

```
>c &= lineThrough(B,C) // c=BC
```

$[-1, 2, 2]$

Kita bisa mendapatkan persamaan untuk sebuah garis dengan mudah.

```
>$getLineEquation(c,x,y), $solve(%,y) | expand // persamaan garis c
```

$$\left[y = \frac{x}{2} + 1 \right]$$

$$\left[y = \frac{x}{2} + 1 \right]$$

```
>$getLineEquation(lineThrough(A, [x1,y1]),x,y) // persamaan garis melalui A dan (x1, y1)
```

$$(x_1 - 1) y - x y_1 = -y_1$$

```
>h &= perpendicular(A,lineThrough(B,C)) // h melalui A tegak lurus BC
```

$$[2, 1, 2]$$

```
>Q &= lineIntersection(c,h) // Q titik potong garis c=BC dan h
```

$$\begin{bmatrix} 2 & 6 \\ - & - \\ 5 & 5 \end{bmatrix}$$

```
>$projectToLine(A,lineThrough(B,C)) // proyeksi A pada BC
```

$$\left[\frac{2}{5}, \frac{6}{5} \right]$$

```
>$distance(A,Q) // jarak AQ
```

$$\frac{3}{\sqrt{5}}$$

```
>cc &= circleThrough(A,B,C); $cc // (titik pusat dan jari-jari) lingkaran melalui A, B, C
```

$$\left[\frac{7}{6}, \frac{7}{6}, \frac{5}{3\sqrt{2}} \right]$$

```
>r&=getCircleRadius(cc); $r , $float(r) // tampilkan nilai jari-jari
```

$$1.178511301977579$$

```
>$computeAngle(A,C,B) // nilai <ACB
```

$$\arccos\left(\frac{4}{5}\right)$$

```
>$solve(getLineEquation(angleBisector(A,C,B),x,y),y) [1] // persamaan garis bagi <ACB
```

$$y = x$$

```
>P &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A)); $P // titik potong 2 ga
```

$$\left[\frac{\sqrt{2}\sqrt{5} + 2}{6}, \frac{\sqrt{2}\sqrt{5} + 2}{6} \right]$$

```
>P() // hasilnya sama dengan perhitungan sebelumnya
```

$$[0.86038, 0.86038]$$

Garis dan Lingkaran yang Berpotongan

Tentu saja, kita juga bisa memotong garis dengan lingkaran, dan lingkaran dengan lingkaran.

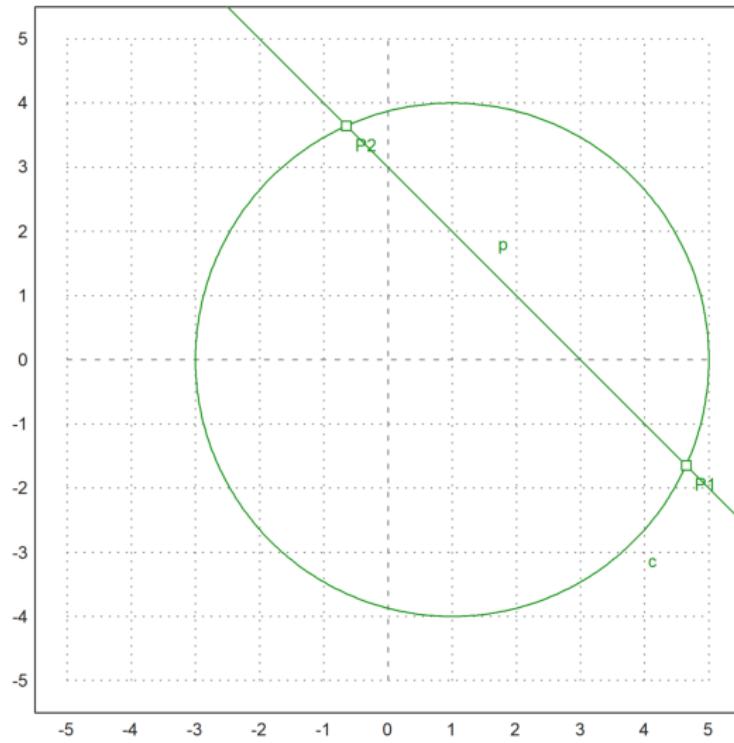
```
>A &:= [1,0]; c=circleWithCenter(A,4);
>B &:= [1,2]; C &:= [2,1]; l=lineThrough(B,C);
>setPlotRange(5); plotCircle(c); plotLine(l);
```

Perpotongan garis dengan lingkaran mengembalikan dua titik dan jumlah titik perpotongan.

```
>{P1,P2,f}=lineCircleIntersections(l,c);
>P1, P2,
```

$$[4.64575, -1.64575]
[-0.645751, 3.64575]$$

```
>plotPoint(P1); plotPoint(P2):
```



Hal yang sama di Maxima.

```
>c &= circleWithCenter(A, 4) // lingkaran dengan pusat A jari-jari 4
```

```
[1, 0, 4]
```

```
>l &= lineThrough(B,C) // garis l melalui B dan C
```

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

```
>$lineCircleIntersections(l,c) | radcan, // titik potong lingkaran c dan garis l
```

$$\left[\left[\sqrt{7} + 2, 1 - \sqrt{7} \right], \left[2 - \sqrt{7}, \sqrt{7} + 1 \right] \right]$$

Akan ditunjukkan bahwa sudut-sudut yang menghadap bsuusr yang sama adalah sama besar.

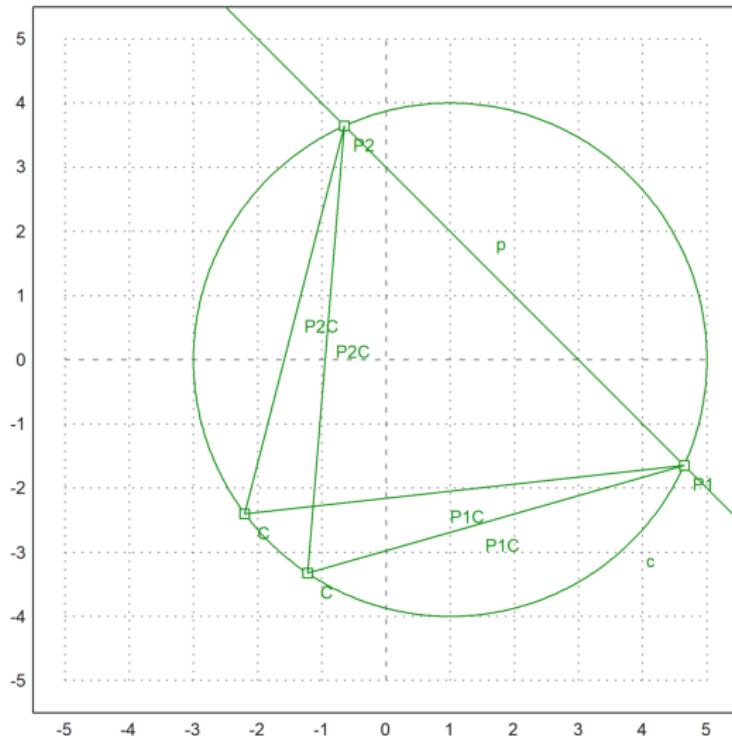
```
>C=A+normalize([-2,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>degrprint(computeAngle(P1,C,P2))
```

$69^\circ 17' 42.68''$

```
>C=A+normalize([-4,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>deprint(computeAngle(P1,C,P2))
```

$69^\circ 17' 42.68''$

```
>insimg;
```

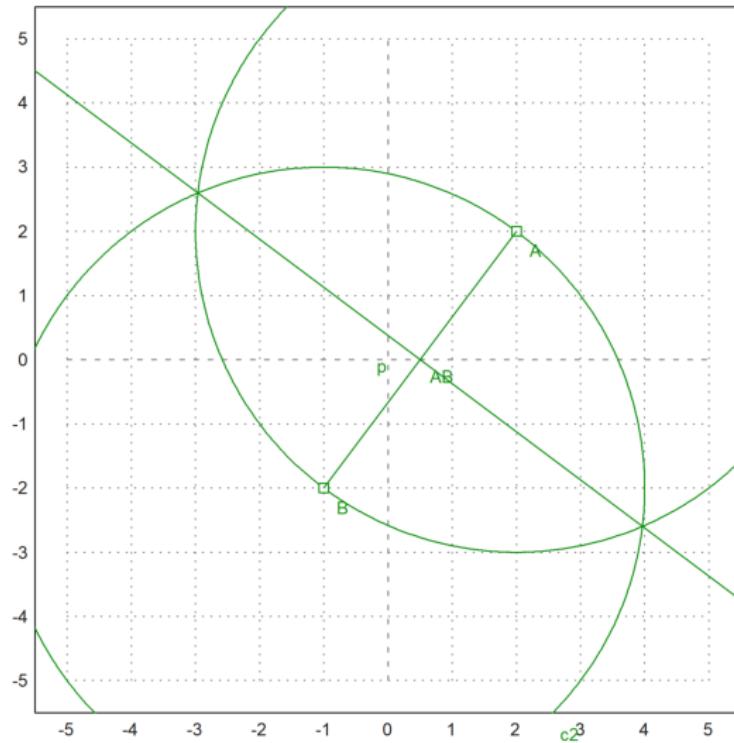


Garis Sumbu

Berikut adalah langkah-langkah menggambar garis sumbu ruas garis AB:

1. Gambar lingkaran dengan pusat A melalui B.
2. Gambar lingkaran dengan pusat B melalui A.
3. Tarik garis melalui kedua titik potong kedua lingkaran tersebut. Garis ini merupakan garis sumbu (melalui titik tengah dan tegak lurus) AB.

```
>A=[2,2]; B=[-1,-2];
>c1=circleWithCenter(A,distance(A,B));
>c2=circleWithCenter(B,distance(A,B));
>{P1,P2,f}=circleCircleIntersections(c1,c2);
>l=lineThrough(P1,P2);
>setPlotRange(5); plotCircle(c1); plotCircle(c2);
>plotPoint(A); plotPoint(B); plotSegment(A,B); plotLine(l);
```



Selanjutnya, kami melakukan hal yang sama di Maxima dengan koordinat umum.

```
>A &= [a1,a2]; B &= [b1,b2];
>c1 &= circleWithCenter(A,distance(A,B));
>c2 &= circleWithCenter(B,distance(A,B));
>P &= circleCircleIntersections(c1,c2); P1 &= P[1]; P2 &= P[2];
```

Persamaan untuk persimpangan cukup terlibat. Tapi kita bisa menyederhanakan, jika kita menyelesaikan y.

```
>g &= getLineEquation(lineThrough(P1,P2),x,y);
>$solve(g,y)
```

$$\left[y = \frac{-(2b_1 - 2a_1)x + b_2^2 + b_1^2 - a_2^2 - a_1^2}{2b_2 - 2a_2} \right]$$

Ini memang sama dengan tengah tegak lurus, yang dihitung dengan cara yang sama sekali berbeda.

```
>$solve(getLineEquation(middlePerpendicular(A,B),x,y),y)
```

$$\left[y = \frac{-(2b_1 - 2a_1)x + b_2^2 + b_1^2 - a_2^2 - a_1^2}{2b_2 - 2a_2} \right]$$

```
>h &= getLineEquation(lineThrough(A,B),x,y);
>$solve(h,y)
```

$$\left[y = \frac{(b_2 - a_2)x - a_1b_2 + a_2b_1}{b_1 - a_1} \right]$$

Perhatikan hasil kali gradien garis g dan h adalah:

$$\frac{-(b_1 - a_1)}{(b_2 - a_2)} \times \frac{(b_2 - a_2)}{(b_1 - a_1)} = -1.$$

Artinya kedua garis tegak lurus. **Contoh 3: Rumus Heron**

Rumus Heron menyatakan bahwa luas segitiga dengan panjang sisi-sisi a, b dan c adalah:

$$L = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{dengan } s = (a+b+c)/2.$$

Untuk membuktikan hal ini kita misalkan C(0,0), B(a,0) dan A(x,y), b=AC, c=AB. Luas segitiga ABC adalah

$$L_{\triangle ABC} = \frac{1}{2}a \times y.$$

Nilai y didapat dengan menyelesaikan sistem persamaan:

$$x^2 + y^2 = b^2, \quad (x-a)^2 + y^2 = c^2.$$

```
>sol &= solve([x^2+y^2=b^2, (x-a)^2+y^2=c^2], [x, y])
```

```
[ ]
```

Ekstrak larutan y.

```
>ysol &= y with sol[2][2]: $ysol
```

```
ysol
```

Kami mendapatkan formula Heron.

```
>function H(a,b,c) &= sqrt(factor((ysol*a/2)^2)); \$'H(a,b,c)=H(a,b,c)
```

$$H(a, b, [1, 0, 4]) = \frac{|a| |ysol|}{2}$$

Tentu saja, setiap segitiga persegi panjang adalah kasus yang terkenal.

```
>H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5
```

Wrong argument!

Cannot combine a symbolic expression here.
Did you want to create a symbolic expression?
Then start with &.

This function can only use real or complex values!
Error in abs
Try "trace errors" to inspect local variables after errors.

```

H:
    useglobal; return abs(a)*abs(ysol)/2
Error in:
H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5 ...
^

```

Dan jelas juga, bahwa ini adalah segitiga dengan luas maksimal dan kedua sisinya 3 dan 4.

```
>aspect (1.5); plot2d(&H(3,4,x),1,7); // Kurva luas segitiga sengan panjang sisi 3, 4, x (
```

Wrong argument!

```

Cannot combine a symbolic expression here.
Did you want to create a symbolic expression?
Then start with &.

```

```

This function can only use real or complex values!
Error in abs
Error in expression: 3*abs(ysol)/2
%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());

```

Kasus umum juga berfungsi.

```
>$solve(diff(H(a,b,c)^2,c)=0,c)
```

```

Maxima said:
diff: second argument must be a variable; found [1,0,4]
-- an error. To debug this try: debugmode(true);

Error in:
$solve(diff(H(a,b,c)^2,c)=0,c) ...
^

```

Sekarang mari kita cari himpunan semua titik di mana $b + c = d$ untuk beberapa konstanta d. Diketahui bahwa ini adalah elips.

```
>s1 &= subst(d-c,b,sol[2]); $s1
```

```

Maxima said:
part: invalid index of list or matrix.
-- an error. To debug this try: debugmode(true);

Error in:
s1 &= subst(d-c,b,sol[2]); $s1 ...
^

```

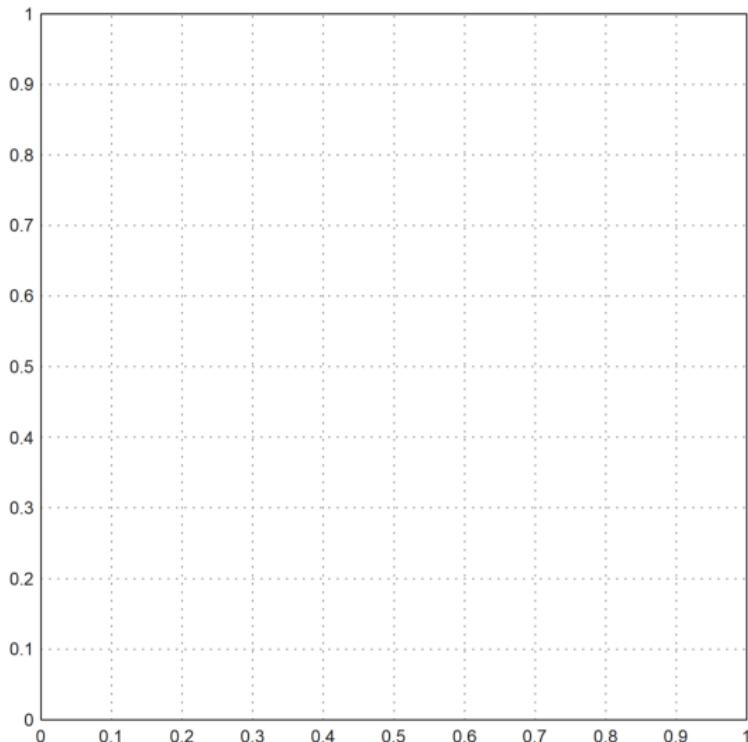
Dan manfaatkan ini.

```
>function fx(a,c,d) &= rhs(s1[1]); $fx(a,c,d), function fy(a,c,d) &= rhs(s1[2]); $fy(a,c,d)
```

0

Sekarang kita bisa menggambar setnya. Sisi b bervariasi dari 1 hingga 4. Diketahui bahwa kita mendapatkan elips.

```
>aspect(1); plot2d(&fx(3,x,5),&fy(3,x,5),xmin=1,xmax=4,square=1):
```



Kita dapat memeriksa persamaan umum elips ini, yaitu.

$$\frac{(x - x_m)^2}{u^2} + \frac{(y - y_m)^2}{v^2} = 1,$$

di mana (x_m, y_m) adalah pusat, dan u dan v adalah setengah sumbu.

```
>$ratsimp((fx(a,c,d)-a/2)^2/u^2+fy(a,c,d)^2/v^2 with [u=d/2,v=sqrt(d^2-a^2)/2])
```

$$\frac{a^2}{d^2}$$

Kita melihat bahwa tinggi dan luas segitiga adalah maksimal untuk $x = 0$. Jadi luas segitiga dengan $a + b + c = d$ adalah maksimal, jika sama sisi. Kami ingin mendapatkan ini secara analitis.

```
>eqns &= [diff(H(a,b,d-(a+b))^2,a)=0, diff(H(a,b,d-(a+b))^2,b)=0]; $eqns
```

$$\left[\frac{ay sol^2}{2} = 0, 0 = 0 \right]$$

Kami mendapatkan beberapa minima, yang termasuk dalam segitiga dengan satu sisi 0, dan solusi $a = b = c = d / 3$.

```
>$solve(eqns, [a,b])
```

$$[[a = 0, b = \%r_1]]$$

Ada juga metode Lagrange, memaksimalkan $H(a, b, c)^2$ terhadap $a + b + d = d$.

```
>&solve([diff(H(a,b,c)^2,a)=la, diff(H(a,b,c)^2,b)=la, ...
>      diff(H(a,b,c)^2,c)=la, a+b+c=d], [a,b,c,la])
```

Maxima said:

```
diff: second argument must be a variable; found [1,0,4]
-- an error. To debug this try: debugmode(true);
```

Error in:

```
... la,      diff(H(a,b,c)^2,c)=la, a+b+c=d], [a,b,c,la]) ...  
          ^
```

Kita bisa membuat plot situasinya

Pertama, atur poin di Maxima.

```
>A &= at([x,y],sol[2]); $A
```

Maxima said:

```
part: invalid index of list or matrix.
-- an error. To debug this try: debugmode(true);
```

Error in:

```
A &= at([x,y],sol[2]); $A ...  
          ^
```

```
>B &= [0,0]; $B, C &= [a,0]; $C
```

$$[a, 0]$$

Kemudian atur rentang plot, dan plot poinnya.

```
>setPlotRange(0,5,-2,3); ...
>a=4; b=3; c=2; ...
>plotPoint(mxmeval("B"), "B"); plotPoint(mxmeval("C"), "C"); ...
>plotPoint(mxmeval("A"), "A");
```

```
Variable a1 not found!
Use global variables or parameters for string evaluation.
Error in Evaluate, superfluous characters found.
Try "trace errors" to inspect local variables after errors.
mxmeval:
    return evaluate(mxm(s));
Error in:
... otPoint(mxmeval("C"), "C"); plotPoint(mxmeval("A"), "A"): ...  
^
```

Plot segmennya.

```
>plotSegment(mxmeval("A"),mxmeval("C")); ...
>plotSegment(mxmeval("B"),mxmeval("C")); ...
>plotSegment(mxmeval("B"),mxmeval("A")):
```

```
Variable a1 not found!
Use global variables or parameters for string evaluation.
Error in Evaluate, superfluous characters found.
Try "trace errors" to inspect local variables after errors.
mxmeval:
    return evaluate(mxm(s));
Error in:
plotSegment(mxmeval("A"), mxmeval("C")); plotSegment(mxmeval("B" ...)
```

Hitung tengah tegak lurus di Maxima.

```
>h &= middlePerpendicular(A,B); g &= middlePerpendicular(B,C);
```

Dan bagian tengah dari keliling.

```
>U &= lineIntersection(h,g);
```

Kami mendapatkan rumus untuk jari-jari lingkaran sunat.

```
>&assume(a>0,b>0,c>0); $distance(U,B) | radcan
```

$$\frac{\sqrt{{a_2}^2 + {a_1}^2} \sqrt{{a_2}^2 + {a_1}^2 - 2 a a_1 + a^2}}{2 |a_2|}$$

Mari kita tambahkan ini ke plot.

```
>plotPoint(U()); ...
>plotCircle(circleWithCenter(mxmeval("U"), mxmeval("distance(U,C)"))):
```

```
Variable a2 not found!
Use global variables or parameters for string evaluation.
Error in ^
Error in expression: [a/2, (a2^2+a1^2-a*a1)/(2*a2)]
Error in:
plotPoint(U()); plotCircle(circleWithCenter(mxmeval("U"), mxmev ...
^
```

Menggunakan geometri, kami mendapatkan rumus sederhana

$$\frac{a}{\sin(\alpha)} = 2r$$

untuk radius. Kami dapat memeriksa, apakah ini benar dengan Maxima. Maxima akan memfaktorkannya hanya jika kita mengkuadratkannya.

```
>$c^2/sin(computeAngle(A,B,C))^2 | factor
```

$$\left[\frac{a_2^2 + a_1^2}{a_2^2}, 0, \frac{16 (a_2^2 + a_1^2)}{a_2^2} \right]$$

Contoh 4: Garis Euler dan Parabola

Garis euler adalah garis yang ditentukan dari segitiga yang tidak sama sisi. Ini adalah garis tengah segitiga, dan melewati beberapa titik penting yang ditentukan dari segitiga, termasuk pusat ortosentrum, sirkumenter, pusat massa, titik Exeter, dan pusat lingkaran sembilan titik segitiga.

Untuk demonstrasi, kami menghitung dan memplot garis Euler dalam segitiga.

Pertama, kami menentukan sudut segitiga di Euler. Kami menggunakan definisi, yang terlihat dalam ekspresi simbolik.

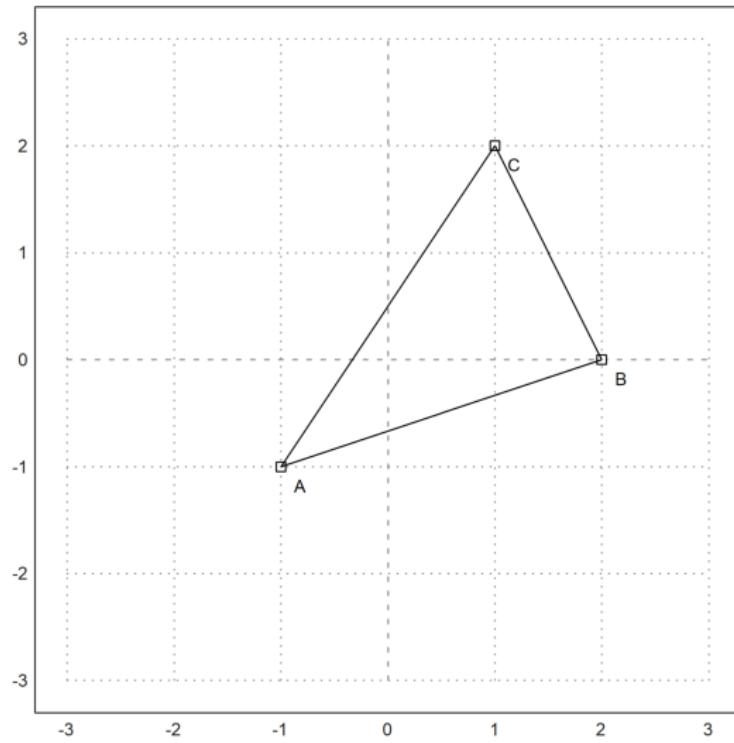
```
>A:=[-1,-1]; B:=[2,0]; C:=[1,2];
```

Untuk memplot objek geometris, kami menyiapkan area plot, dan menambahkan poin ke dalamnya. Semua plot objek geometris ditambahkan ke plot saat ini.

```
>setPlotRange(3); plotPoint(A, "A"); plotPoint(B, "B"); plotPoint(C, "C");
```

Kita juga bisa menambahkan sisi segitiga.

```
>plotSegment(A,B,""); plotSegment(B,C,""); plotSegment(C,A,"");
```



Berikut adalah luas segitiga menggunakan rumus determinan. Tentu saja kita harus mengambil nilai absolut dari hasil ini.

```
>$areaTriangle(A,B,C)
```

$$-\frac{7}{2}$$

Kita dapat menghitung koefisien dari sisi c.

```
>c &= lineThrough(A,B)
```

$$[-1, 3, -2]$$

Dan juga dapatkan rumus untuk baris ini.

```
>$getLineEquation(c,x,y)
```

$$3y - x = -2$$

Untuk bentuk Hesse, kita perlu menentukan titik, sehingga titik tersebut berada di sisi positif dari bentuk Hesse. Memasukkan titik menghasilkan jarak positif ke garis.

```
>$getHesseForm(c,x,y,C), $at(%,[x=C[1],y=C[2]])
```

$$\frac{7}{\sqrt{10}}$$

$$\frac{7}{\sqrt{10}}$$

Sekarang kami menghitung sirkit ABC.

```
>LL &= circleThrough(A,B,C); $getCircleEquation(LL,x,y)
```

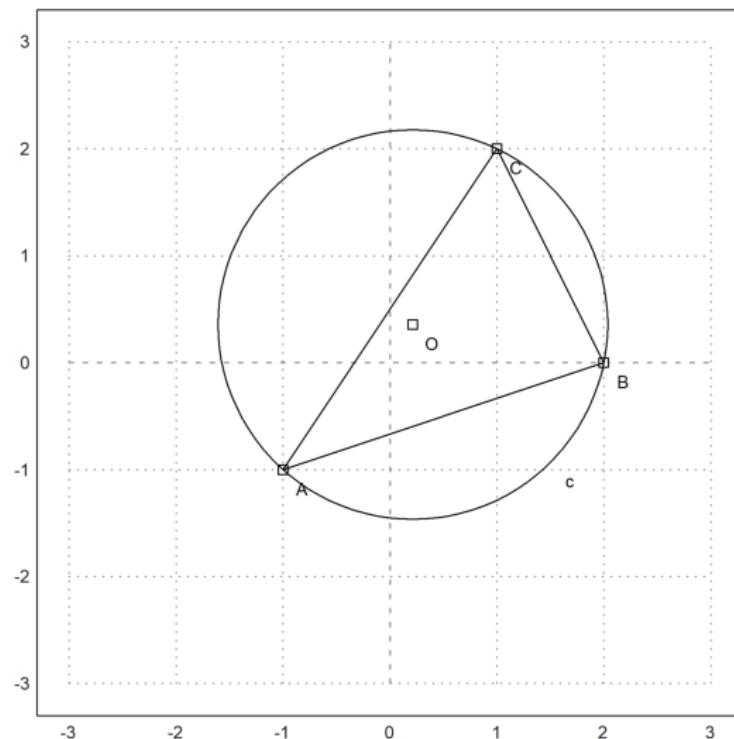
$$\left(y - \frac{5}{14}\right)^2 + \left(x - \frac{3}{14}\right)^2 = \frac{325}{98}$$

```
>O &= getCircleCenter(LL); $O
```

$$\left[\frac{3}{14}, \frac{5}{14}\right]$$

Plot lingkaran dan pusatnya. Cu dan U adalah simbolik. Kami mengevaluasi ekspresi ini untuk Euler.

```
>plotCircle(LL()); plotPoint(O(), "O"):
```



Kita dapat menghitung perpotongan ketinggian di ABC (orthocenter) secara numerik dengan perintah berikut.

```
>H &= lineIntersection(perpendicular(A,lineThrough(C,B)), ...
>    perpendicular(B,lineThrough(A,C))); $H
```

$$\left[\frac{11}{7}, \frac{2}{7} \right]$$

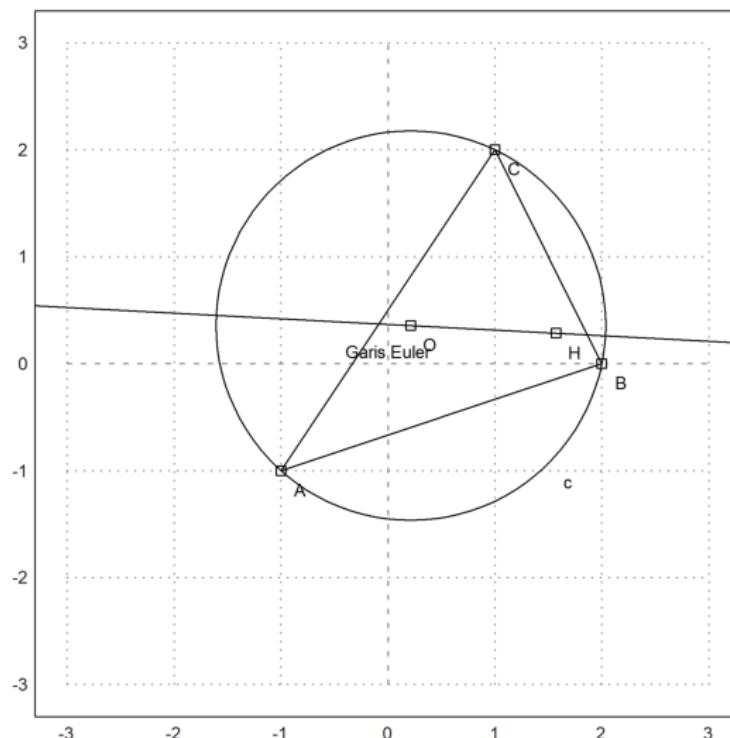
Sekarang kita dapat menghitung garis Euler dari segitiga tersebut.

```
>el &= lineThrough(H,O); $getLineEquation(el,x,y)
```

$$-\frac{19y}{14} - \frac{x}{14} = -\frac{1}{2}$$

Tambahkan ke plot kita.

```
>plotPoint(H(),"H"); plotLine(el(),"Garis Euler"):
```

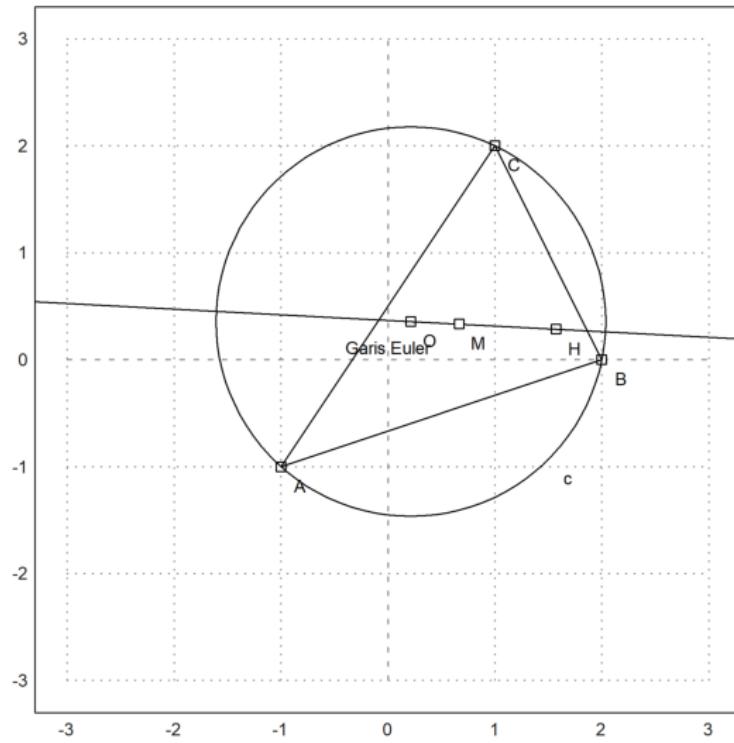


Pusat gravitasi harus berada di garis ini.

```
>M &= (A+B+C)/3; $getLineEquation(el,x,y) with [x=M[1],y=M[2]]
```

$$-\frac{1}{2} = -\frac{1}{2}$$

```
>plotPoint(M(),"M"); // titik berat
```



Teorinya mengatakan bahwa $MH = 2 * MO$. Kita perlu menyederhanakan dengan radcan untuk mencapai ini.

```
> $distance(M, H) / $distance(M, O) | radcan
```

$$2$$

Fungsinya termasuk fungsi untuk sudut juga.

```
> $computeAngle(A, C, B), degprint(%())
```

$$\arccos\left(\frac{4}{\sqrt{5}\sqrt{13}}\right)$$

$60^\circ 15' 18.43''$

Persamaan untuk pusat lingkaran tidak terlalu bagus.

```
> Q &= lineIntersection(angleBisector(A, C, B), angleBisector(C, B, A)) | radcan; $Q
```

$$\left[\frac{\left(2^{\frac{3}{2}} + 1\right)\sqrt{5}\sqrt{13} - 15\sqrt{2} + 3}{14}, \frac{(\sqrt{2} - 3)\sqrt{5}\sqrt{13} + 52^{\frac{3}{2}} + 5}{14} \right]$$

Mari kita hitung juga ekspresi jari-jari lingkaran yang tertulis.

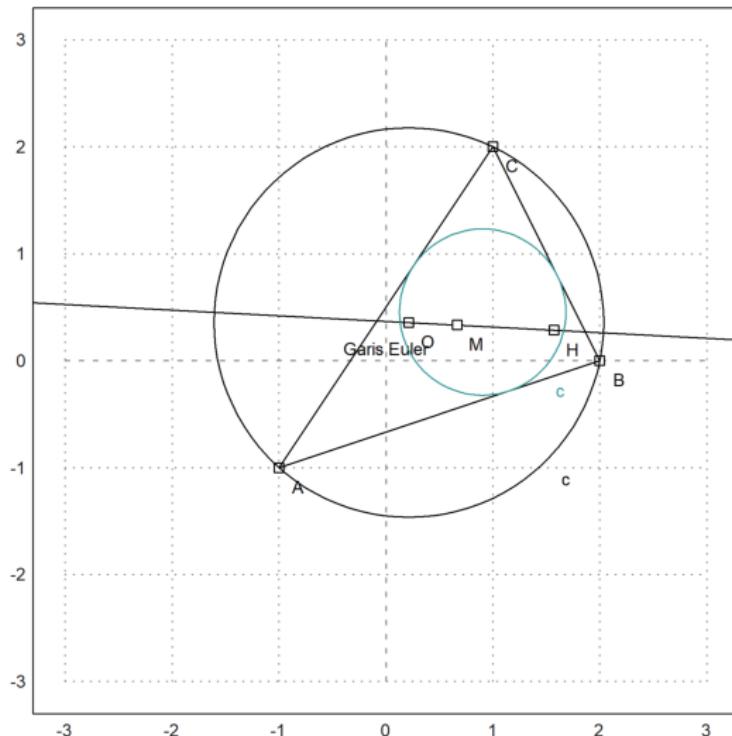
```
> r &= distance(Q, projectToLine(Q, lineThrough(A, B))) | ratsimp; $r
```

$$\frac{\sqrt{(-41\sqrt{2} - 31) \sqrt{5}\sqrt{13} + 115\sqrt{2} + 614}}{7\sqrt{2}}$$

```
>LD &= circleWithCenter(Q,r); // Lingkaran dalam
```

Mari kita tambahkan ini ke plot.

```
>color(5); plotCircle(LD()):
```



Parabola

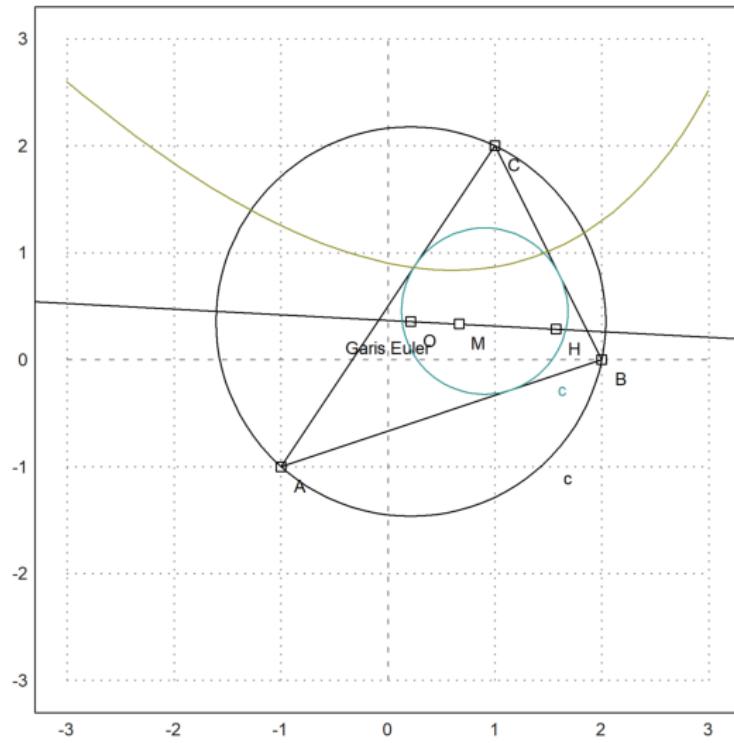
Selanjutnya akan dicari persamaan tempat kedudukan titik-titik yang berjarak sama ke titik C dan ke garis AB.

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

$$\frac{3y - x + 2}{\sqrt{10}} - \sqrt{(2-y)^2 + (1-x)^2} = 0$$

Persamaan tersebut dapat digambar menjadi satu dengan gambar sebelumnya.

```
>plot2d(p,level=0,add=1,contourcolor=6):
```



This should be some function, but the default solver of Maxima can find the solution only, if we square the equation. Consequently, we get a fake solution.

```
>akar &= solve(getHesseForm(lineThrough(A,B),x,y,C)^2-distance([x,y],C)^2,y)
```

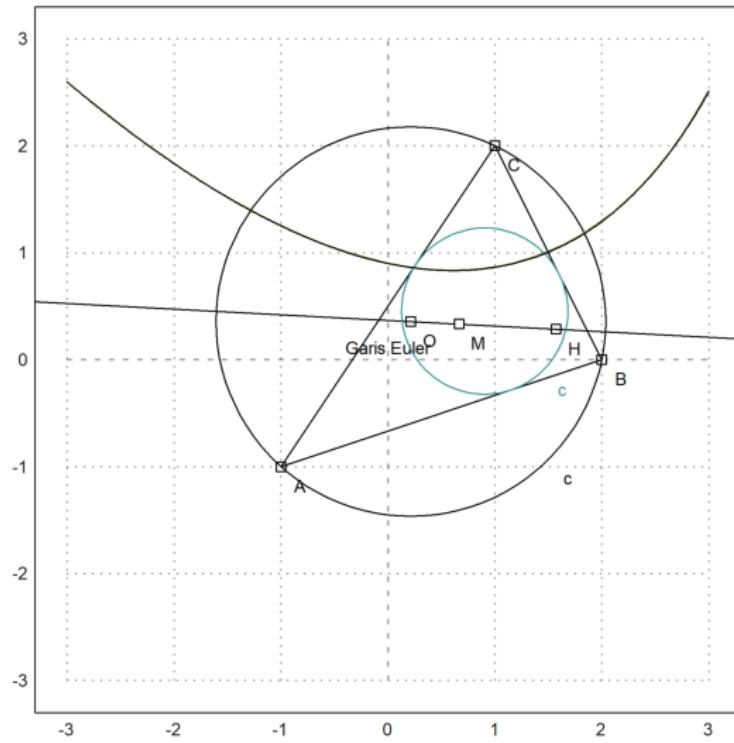
$$\begin{aligned}y &= -3x - \sqrt{70} \sqrt{9 - 2x} + 26, \\y &= -3x + \sqrt{70} \sqrt{9 - 2x} + 26\end{aligned}$$

The first solution is

maxima: akar[1]

Adding the first solution to the plot show, that it is indeed the path we are looking for. The theory tells us that it is a rotated parabola.

```
>plot2d(&rhs(akar[1]),add=1):
```



```
>function g(x) &= rhs(akar[1]); $'g(x)= g(x)// fungsi yang mendefinisikan kurva di atas
```

$$g(x) = -3x - \sqrt{70} \sqrt{9 - 2x} + 26$$

```
>T &=[-1, g(-1)]; // ambil sebarang titik pada kurva tersebut
>dTC &= distance(T,C); $fullratsimp(dTC), $float(%); // jarak T ke C
```

$$2.135605779339061$$

```
>U &= projectToLine(T, lineThrough(A, B)); $U // proyeksi T pada garis AB
```

$$\left[\frac{80 - 3\sqrt{11}\sqrt{70}}{10}, \frac{20 - \sqrt{11}\sqrt{70}}{10} \right]$$

```
>dU2AB &= distance(T, U); $fullratsimp(dU2AB), $float(%); // jarak T ke AB
```

$$2.135605779339061$$

Ternyata jarak T ke C sama dengan jarak T ke AB. Coba Anda pilih titik T yang lain dan ulangi perhitungan-perhitungan di atas untuk menunjukkan bahwa hasilnya juga sama.

Contoh 5: Trigonometri Rasional

Ini terinspirasi oleh ceramah N.J.Wildberger. Dalam bukunya "Proporsi Agung", Wildberger mengusulkan untuk menggantikan pengertian klasik tentang jarak dan sudut dengan kuadransi dan penyebaran. Dengan menggunakan ini, memang mungkin untuk menghindari fungsi trigonometri dalam banyak contoh, dan tetap "rasional".

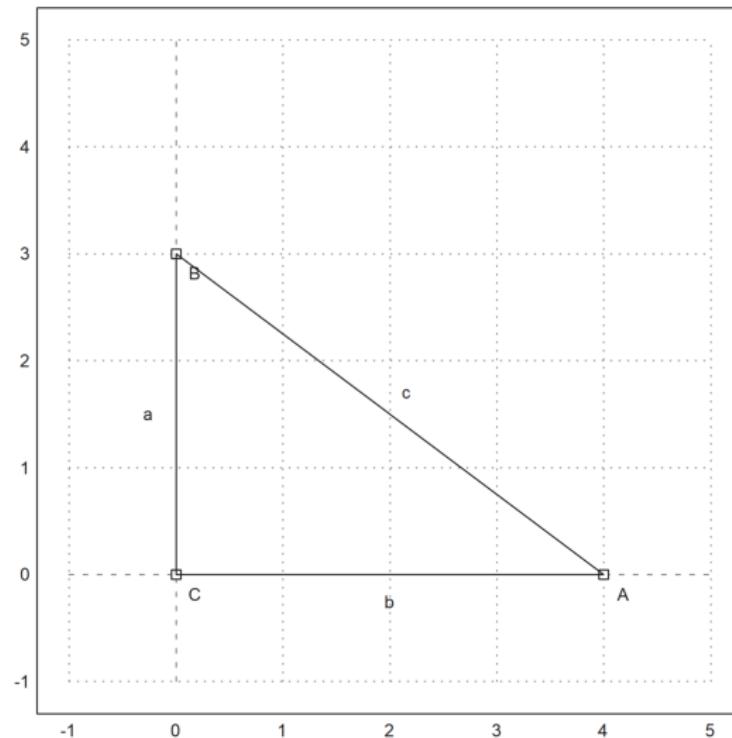
Berikut ini, saya memperkenalkan konsep, dan memecahkan beberapa masalah. Saya menggunakan perhitungan simbolik Maxima di sini, yang menyembunyikan keuntungan utama dari trigonometri rasional bahwa perhitungan dapat dilakukan dengan kertas dan pensil saja. Anda diundang untuk memeriksa hasil tanpa komputer.

Intinya adalah bahwa perhitungan rasional simbolis sering kali menghasilkan hasil yang sederhana. Sebaliknya, trigonometri klasik menghasilkan hasil trigonometri yang rumit, yang mengevaluasi ke pendekatan numerik saja.

```
>load geometry;
```

Untuk pendahuluan pertama, kami menggunakan segitiga persegi panjang dengan proporsi Mesir terkenal 3, 4 dan 5. Perintah berikut adalah perintah Euler untuk memplot geometri bidang yang terdapat dalam file Euler "geometry.e".

```
>C:=[0,0]; A:=[4,0]; B:=[0,3]; ...
>setPlotRange(-1,5,-1,5); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg(30);
```



Tentu saja,

$$\sin(w_a) = \frac{a}{c},$$

di mana wa adalah sudut di A. Cara biasa untuk menghitung sudut ini, adalah dengan melakukan invers dari fungsi sinus. Hasilnya adalah sudut yang tidak dapat dicerna, yang hanya dapat dicetak secara perkiraan.

```
>wa := arcsin(3/5); degprint(wa)
```

36°52'11.63''

Trigonometri rasional mencoba menghindari hal ini.

Pengertian pertama dari trigonometri rasional adalah kuadran, yang menggantikan jarak. Faktanya, itu hanya kuadrat jarak. Berikut ini, a, b, dan c menunjukkan kuadran sisi-sisinya.

Teorema Pythagoras menjadi $a + b = c$ lalu.

```
>a &= 3^2; b &= 4^2; c &= 5^2; &a+b=c
```

$$25 = 25$$

Gagasan kedua dari trigonometri rasional adalah penyebarannya. Spread mengukur bukaan antar baris. Ini adalah 0, jika garis sejajar, dan 1, jika garis persegi panjang. Ini adalah kuadran dari sinus sudut antara dua garis.

Penyebaran garis AB dan AC pada gambar di atas didefinisikan sebagai

$$s_a = \sin(\alpha)^2 = \frac{a}{c},$$

di mana a dan c adalah kuadrat dari segitiga persegi panjang mana pun dengan satu sudut di A.

```
>sa &= a/c; $sa
```

$$\frac{9}{25}$$

Ini lebih mudah dihitung daripada sudut, tentu saja. Tetapi Anda kehilangan properti yang sudut dapat ditambahkan dengan mudah.

Tentu saja, kita dapat mengubah nilai perkiraan sudut wa menjadi sprad, dan mencetaknya sebagai pecahan.

```
>fracprint(sin(wa)^2)
```

9/25

Hukum kosinus dari trigonometri klasik diterjemahkan menjadi "hukum silang" berikut.

$$(c + b - a)^2 = 4bc(1 - s_a)$$

Di sini a, b, dan c adalah kuadran dari sisi-sisi segitiga, dan sa adalah sebaran di sudut A. Sisi a, seperti biasa, berlawanan dengan sudut A.

Hukum ini diimplementasikan dalam file geometry.e yang kami muat ke Euler.

```
>$crosslaw(aa,bb,cc,saa)
```

$$\left[\left(bb - aa + \frac{7}{6} \right)^2, \left(bb - aa + \frac{7}{6} \right)^2, \left(bb - aa + \frac{5}{3\sqrt{2}} \right)^2 \right] = \left[\frac{14 bb (1 - saa)}{3}, \frac{14 bb (1 - saa)}{3}, \frac{5 2^{\frac{3}{2}} bb (1 - saa)}{3} \right]$$

Dalam kasus kami, kami mendapatkan

```
>$crosslaw(a,b,c,sa)
```

$$1024 = 1024$$

Mari kita gunakan crosslaw ini untuk mencari sebaran di A. Untuk melakukan ini, kita menghasilkan crosslaw untuk kuadran a, b, dan c, dan menyelesaiannya untuk sebaran yang tidak diketahui sa.

Anda dapat melakukan ini dengan tangan dengan mudah, tetapi saya menggunakan Maxima. Tentu saja, kami mendapatkan hasilnya, kami sudah mendapatkannya.

```
>$crosslaw(a,b,c,x), $solve(%,x)
```

$$\left[x = \frac{9}{25} \right]$$

$$\left[x = \frac{9}{25} \right]$$

Kami sudah tahu ini. Definisi penyebaran adalah kasus khusus dari hukum lintas hukum.

Kita juga bisa menyelesaikan ini untuk umum a, b, c. Hasilnya adalah rumus yang menghitung sebaran sudut segitiga berdasarkan kuadran ketiga sisinya.

```
>$solve(crosslaw(aa,bb,cc,x),x)
```

$$\left[\left[\frac{168 bb x + 36 bb^2 + (-72 aa - 84) bb + 36 aa^2 - 84 aa + 49}{36}, \frac{168 bb x + 36 bb^2 + (-72 aa - 84) bb + 36 aa^2 - 84 aa + 49}{36} \right], \dots \right]$$

Kita bisa membuat fungsi dari hasilnya. Fungsi seperti itu sudah ditentukan dalam file geometry.e Euler.

```
>$spread(a,b,c)
```

$$\frac{9}{25}$$

Sebagai contoh, kita bisa menggunakannya untuk menghitung sudut segitiga bersisi

$$a, \quad a, \quad \frac{4a}{7}$$

Hasilnya rasional, yang tidak mudah didapat jika kita menggunakan trigonometri klasik.

```
>$spread(a,a,4*a/7)
```

$$\frac{6}{7}$$

Ini adalah sudut dalam derajat.

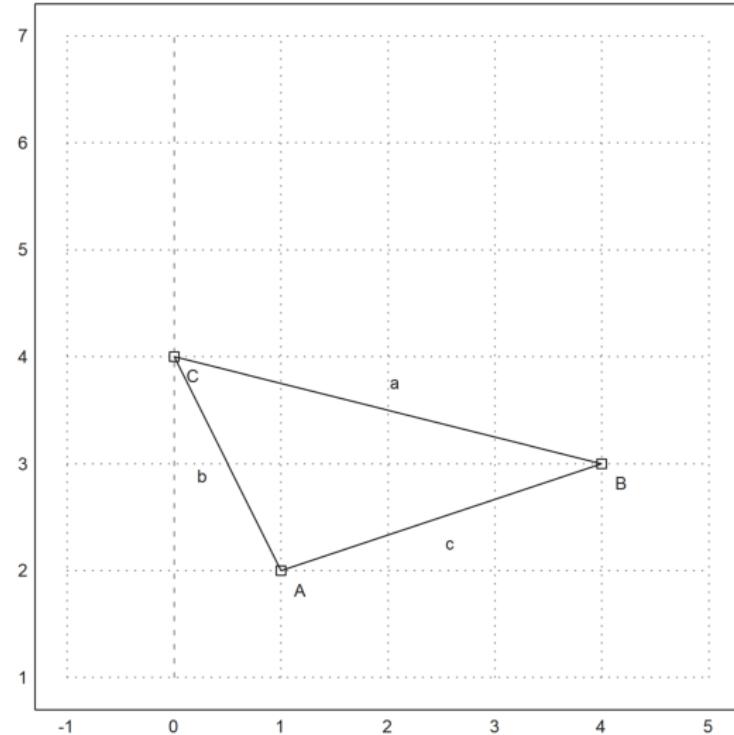
```
>degsprint(arcsin(sqrt(6/7)))
```

$67^\circ 47' 32.44''$

Contoh lain

Sekarang, mari kita coba contoh yang lebih canggih.
Kami mengatur tiga sudut segitiga sebagai berikut.

```
>A&:=[1,2]; B&:=[4,3]; C&:=[0,4]; ...
>setPlotRange(-1,5,1,7); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg;
```



Menggunakan Pythagoras, mudah untuk menghitung jarak antara dua titik. Saya pertama kali menggunakan jarak fungsi file Euler untuk geometri. Jarak fungsi menggunakan geometri klasik.

```
>$distance(A,B)
```

$$\sqrt{10}$$

Euler juga memiliki fungsi kuadrans antara dua titik.

Dalam contoh berikut, karena $c + b$ bukan a , segitiga tidak persegi panjang.

```
>c &= quad(A,B); $c, b &= quad(A,C); $b, a &= quad(B,C); $a,
```

17

Pertama, mari kita hitung sudut tradisional. Fungsi computeAngle menggunakan metode biasa berdasarkan perkalian titik dari dua vektor. Hasilnya adalah beberapa pendekatan floating point.

```
>wb &= computeAngle(A,B,C); $wb, $(wb/pi*180)()
```

$$\arccos\left(\frac{11}{\sqrt{10}\sqrt{17}}\right)$$

32.4711922908

Menggunakan pensil dan kertas, kita bisa melakukan hal yang sama dengan hukum silang. Kami memasukkan kuadran a , b , dan c ke dalam hukum silang dan menyelesaikan untuk x .

```
>$crosslaw(a,b,c,x), $solve(% ,x),
```

$$\left[x = \frac{49}{50}\right]$$

$$\left[x = \frac{49}{50}\right]$$

Artinya, fungsi penyebaran yang didefinisikan dalam "geometry.e" tidak.

```
>sb &= spread(b,a,c); $sb
```

$$\frac{49}{170}$$

Maxima mendapatkan hasil yang sama dengan menggunakan trigonometri biasa, jika kita memaksakannya. Itu menyelesaikan istilah $\sin(\arccos(...))$ menjadi hasil pecahan. Kebanyakan siswa tidak dapat melakukan ini.

```
>$sin(computeAngle(A,B,C))^2
```

$$\frac{49}{170}$$

Setelah kita mendapatkan sebaran di B , kita bisa menghitung tinggi ha di sisi a . Ingat bahwa

$$s_b = \frac{h_a}{c}$$

Menurut definisi.

```
>ha &= c*sb; $ha
```

$$\frac{49}{17}$$

Gambar berikut telah diproduksi dengan program geometri C.a.R., yang dapat menggambar kuadran dan menyebar.

gambar : (20) Rational_Geometry_CaR.png

Menurut definisi, panjang ha adalah akar kuadrat dari kuadrannya.

```
>$sqrt (ha)
```

$$\frac{7}{\sqrt{17}}$$

Sekarang kita bisa menghitung luas segitiga. Jangan lupa, bahwa kita berurusan dengan kuadran!

```
>$sqrt (ha) *sqrt (a) /2
```

$$\frac{7}{2}$$

Rumus determinan yang biasa menghasilkan hasil yang sama.

```
>$areaTriangle (B,A,C)
```

$$\frac{7}{2}$$

The Heron Formula

Sekarang, mari kita selesaikan masalah ini secara umum!

```
>&remvalue (a,b,c,sb,ha);
```

Pertama-tama kita menghitung spread di B untuk segitiga dengan sisi a, b, dan c. Kemudian kami menghitung luas area yang dikuadratkan ("kuadrea"?), Memfaktorkannya dengan Maxima, dan kami mendapatkan rumus Heron yang terkenal.

Memang, ini sulit dilakukan dengan pensil dan kertas.

```
>$spread(b^2,c^2,a^2), $factor(%*c^2*a^2/4)
```

$$\frac{(-c + b + a)(c - b + a)(c + b - a)(c + b + a)}{16}$$

$$\frac{(-c + b + a)(c - b + a)(c + b - a)(c + b + a)}{16}$$

The Triple Spread Rule

Kerugian dari spread adalah bahwa mereka tidak lagi hanya menambahkan sudut serupa. Namun, tiga sebaran segitiga memenuhi aturan "penyebaran rangkap tiga" berikut.

```
>&remvalue(sa,sb,sc); $triplespread(sa,sb,sc)
```

$$(sc + sb + sa)^2 = 2 (sc^2 + sb^2 + sa^2) + 4 sa sb sc$$

Aturan ini berlaku untuk tiga sudut yang bertambah menjadi 180° .

$$\alpha + \beta + \gamma = \pi$$

Sejak penyebaran

$$\alpha, \pi - \alpha$$

sama, aturan penyebaran tiga kali lipat juga benar, jika

$$\alpha + \beta = \gamma$$

Karena penyebaran sudut negatif adalah sama, aturan penyebaran tiga kali lipat juga berlaku, jika

$$\alpha + \beta + \gamma = 0$$

Misalnya, kita dapat menghitung sebaran sudut 60° . Ini $3/4$. Persamaan memiliki solusi kedua, di mana semua spread adalah 0.

```
>$solve(triplespread(x,x,x),x)
```

$$\left[x = \frac{3}{4}, x = 0 \right]$$

Sebaran 90° jelaslah 1. Jika dua sudut dijumlahkan menjadi 90° , penyebarannya menyelesaikan persamaan penyebaran rangkap tiga dengan $a, b, 1$. Dengan perhitungan berikut kita mendapatkan $a + b = 1$.

```
>$triplespread(x,y,1), $solve(%,x)
```

$$[x = 1 - y]$$

Karena penyebaran 180° -t sama dengan penyebaran t, rumus penyebaran rangkap tiga juga berlaku, jika satu sudut adalah jumlah atau perbedaan dari dua sudut lainnya.

Jadi kita bisa menemukan sebaran sudut berlipat ganda. Perhatikan bahwa ada dua solusi lagi. Kami menjelaskan ini sebuah fungsi.

```
>$solve(triplespread(a,a,x),x), function doublespread(a) &= factor(rhs(%[1]))
```

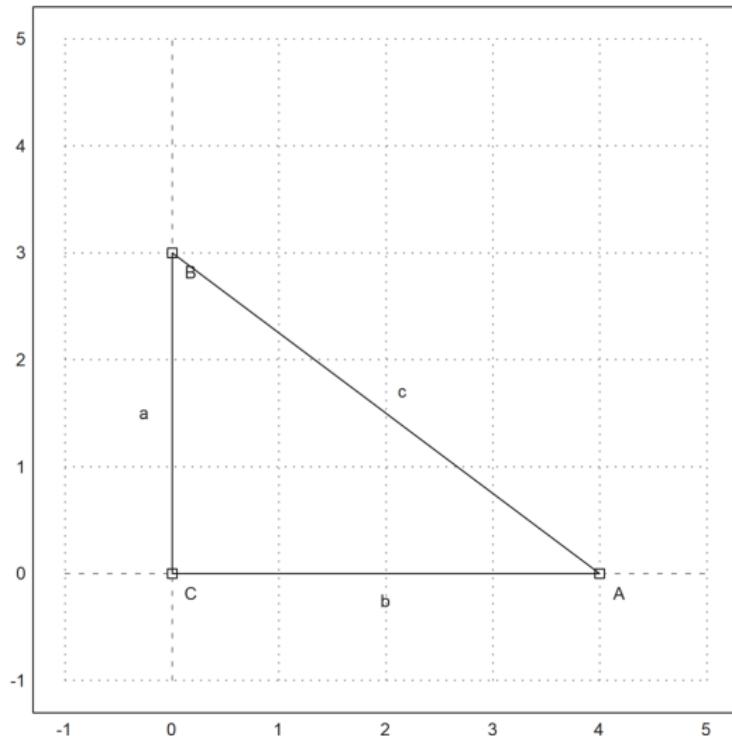
$$[x = 4a - 4a^2, x = 0]$$

$$- 4 (a - 1) a$$

Angle Bisectors

Ini situasinya, kita sudah tahu.

```
>C:=[0,0]; A:=[4,0]; B:=[0,3]; ...
>setPlotRange(-1,5,-1,5); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg;
```



Mari kita hitung panjang bisektor sudut pada A. Tapi kita ingin menyelesaiakannya untuk umum a, b, c.

```
>&remvalue(a,b,c);
```

Jadi pertama-tama kita menghitung sebaran sudut terbagi di A, menggunakan rumus sebaran rangkap tiga. Masalah dengan rumus ini muncul lagi. Ini memiliki dua solusi. Kami harus memilih yang benar. Solusi lainnya mengacu pada sudut terbagi 180° -wa.

```
>$triplespread(x,x,a/(a+b)), $solve(% ,x), sa2 &= rhs(%[1]); $sa2
```

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

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

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

Mari kita periksa persegi panjang Mesir.

```
>$sa2 with [a=3^2,b=4^2]
```

$$\frac{1}{10}$$

Kami dapat mencetak sudut di Euler, setelah mentransfer penyebaran ke radian.

```
>wa2 := arccsin(sqrt(1/10)); degprint(wa2)
```

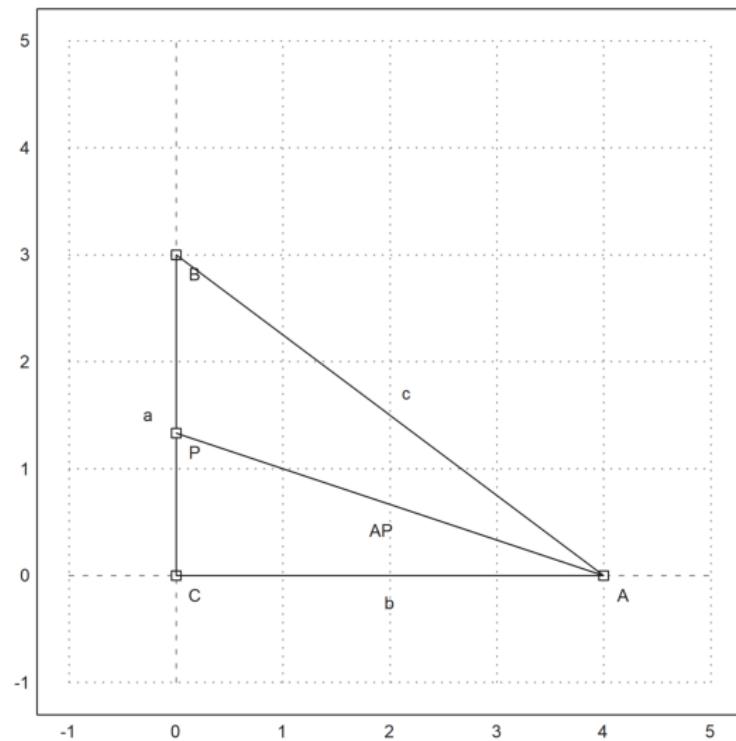
$$18^\circ 26' 5.82''$$

Titik P adalah perpotongan dari garis bagi sudut dengan sumbu y.

```
>P := [0,tan(wa2)*4]
```

$$[0, 1.33333]$$

```
>plotPoint(P,"P"); plotSegment(A,P):
```



Mari kita periksa sudut dalam contoh spesifik kita.

```
>computeAngle(C,A,P), computeAngle(P,A,B)
```

$$0.321750554397$$

$$0.321750554397$$

Sekarang kita menghitung panjang bisektor AP.

Kami menggunakan teorema sinus di segitiga APC. Teorema ini menyatakan bahwa

$$\frac{BC}{\sin(w_a)} = \frac{AC}{\sin(w_b)} = \frac{AB}{\sin(w_c)}$$

memegang di segitiga apa pun. Persegi itu, itu diterjemahkan ke dalam apa yang disebut "hukum penyebaran"

$$\frac{a}{s_a} = \frac{b}{s_b} = \frac{c}{s_b}$$

dimana a, b, c menunjukkan qudrance.

Karena BPA sebaran adalah 1-sa2, kita dapatkan darinya bisa / 1 = b / (1-sa2) dan dapat menghitung bisa (kuadran garis-garis).

```
>&factor(ratsimp(b/(1-sa2))); bisa &= %; $bisa
```

$$\frac{2b(b+a)}{\sqrt{b}\sqrt{b+a}+b+a}$$

Mari kita periksa rumus ini untuk nilai Mesir kita.

```
>sqrt(mxmeval("at(bisa,[a=3^2,b=4^2])")), distance(A,P)
```

```
4.21637021356  
4.21637021356
```

Kami juga dapat menghitung P menggunakan rumus spread.

```
>py&=factor(ratsimp(sa2*bisa)); $py
```

$$-\frac{b(\sqrt{b}\sqrt{b+a}-b-a)}{\sqrt{b}\sqrt{b+a}+b+a}$$

Nilainya sama dengan yang kita dapatkan dengan rumus trigonometri.

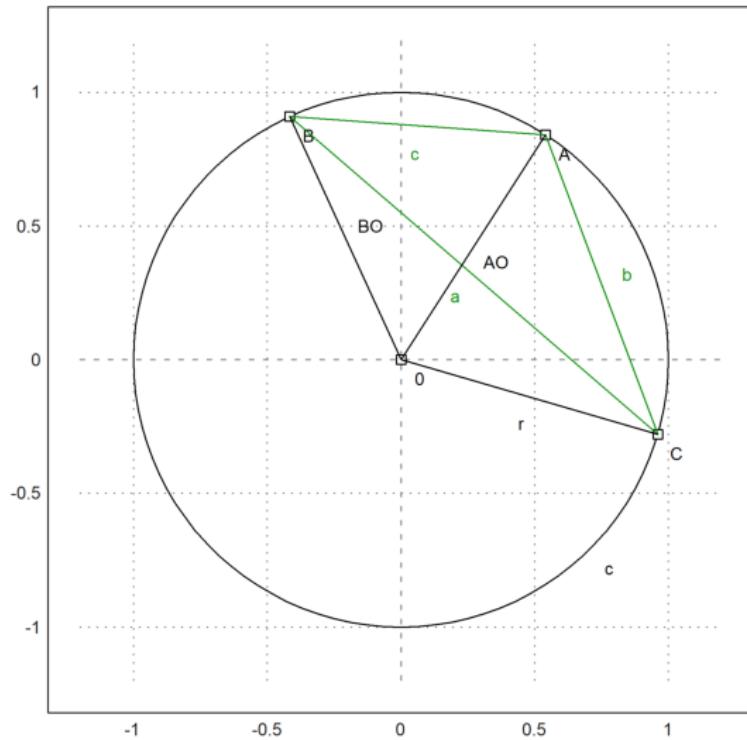
```
>sqrt(mxmeval("at(py,[a=3^2,b=4^2])"))
```

```
1.33333333333
```

The Chord Angle

Perhatikan situasi berikut.

```
>setPlotRange(1.2); ...  
>color(1); plotCircle(circleWithCenter([0,0],1)); ...  
>A:=[cos(1),sin(1)]; B:=[cos(2),sin(2)]; C:=[cos(6),sin(6)]; ...  
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...  
>color(3); plotSegment(A,B,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...  
>color(1); O:=[0,0]; plotPoint(O,"O"); ...  
>plotSegment(A,O); plotSegment(B,O); plotSegment(C,O,"r"); ...  
>insimg;
```



Kita bisa menggunakan Maxima untuk menyelesaikan rumus sebaran rangkap tiga untuk sudut di pusat O untuk r . Jadi kita mendapatkan rumus untuk jari-jari kuadrat dari keliling dalam hal kuadrat sisi. Kali ini, Maxima menghasilkan beberapa angka nol yang kompleks, yang kita abaikan.

```
>&remvalue(a,b,c,r); // hapus nilai-nilai sebelumnya untuk perhitungan baru
>rabc &= rhs(solve(triplespread(spread(b,r,r),spread(a,r,r),spread(c,r,r)),r)[4]); $rabc
```

$$-\frac{abc}{c^2 - 2bc + a(-2c - 2b) + b^2 + a^2}$$

Kita bisa menjadikannya sebagai fungsi Euler.

```
>function periradius(a,b,c) &= rabc;
```

Mari kita periksa hasilnya untuk poin A, B, C kita.

```
>a:=quadrance(B,C); b:=quadrance(A,C); c:=quadrance(A,B);
```

Radiusnya memang 1.

```
>periradius(a,b,c)
```

1

Faktanya, penyebaran CBA hanya bergantung pada b dan c . Ini adalah teorema sudut akord.

```
>$spread(b,a,c)*rabc | ratsimp
```

$$\frac{b}{4}$$

Sebenarnya sebarannya adalah $b / (4r)$, dan kita melihat bahwa sudut akor b adalah setengah dari sudut tengah.

```
>$doublespread(b/(4*r))-spread(b,r,r) | ratsimp
```

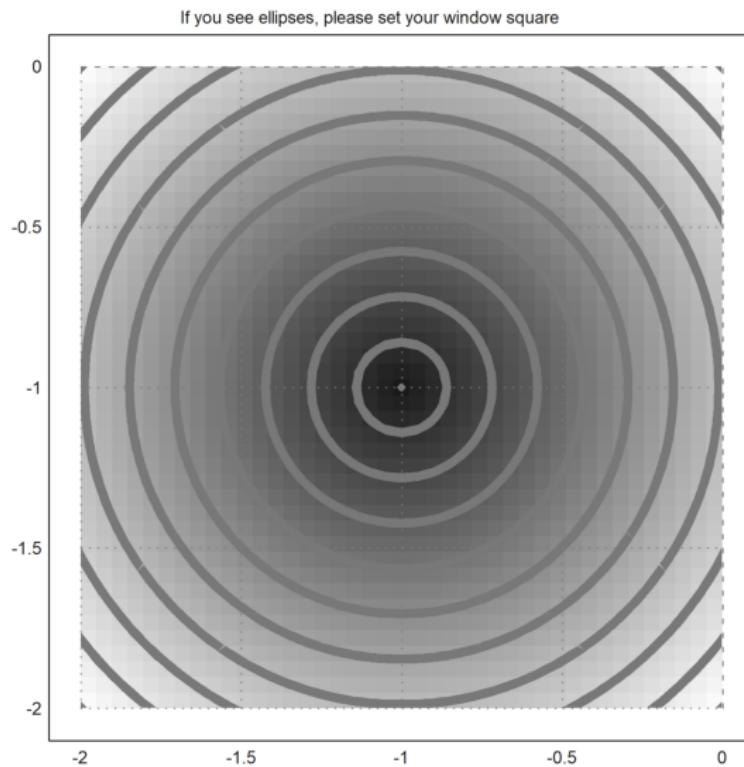
$$0$$

Contoh 6: Jarak Minimal pada Bidang

Ucapan awal

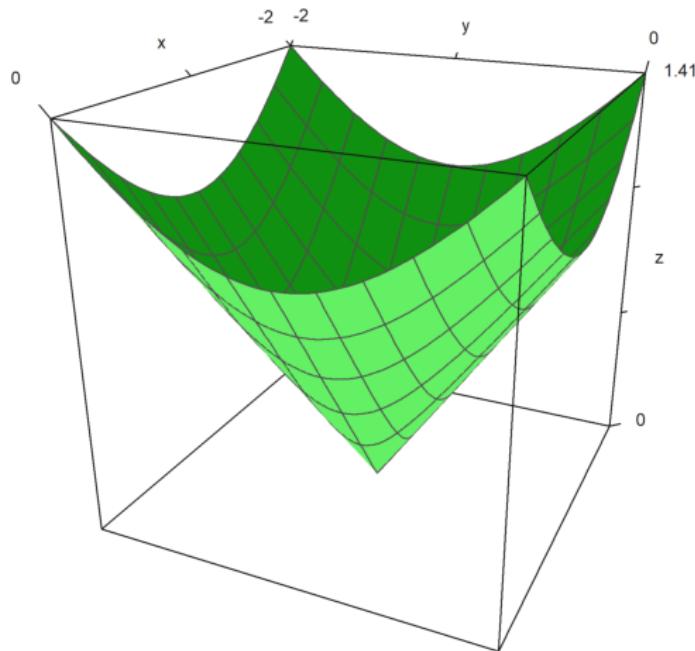
Fungsi yang, ke titik M di bidang, menetapkan jarak AM antara titik tetap A dan M, memiliki garis level yang agak sederhana: lingkaran berpusat di A.

```
>&remvalue();  
>A=[-1,-1];  
>function d1(x,y):=sqrt((x-A[1])^2+(y-A[2])^2)  
>fcontour("d1",xmin=-2,xmax=0,ymin=-2,ymax=0,hue=1, ...  
>title="If you see ellipses, please set your window square":
```



dan grafiknya juga agak sederhana: bagian atas kerucut:

```
>plot3d("d1",xmin=-2,xmax=0,ymin=-2,ymax=0):
```

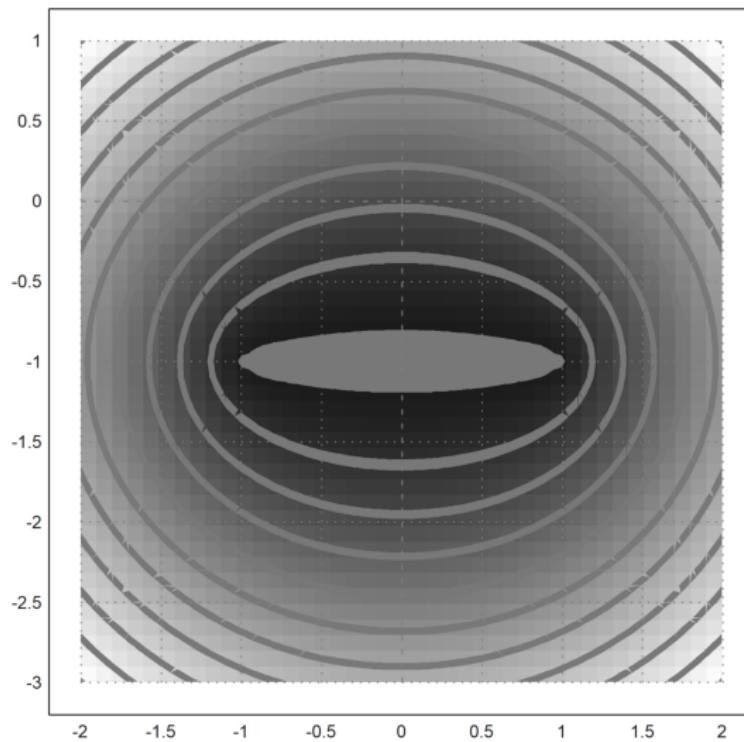


Tentu saja minimal 0 dicapai di A.

Two points

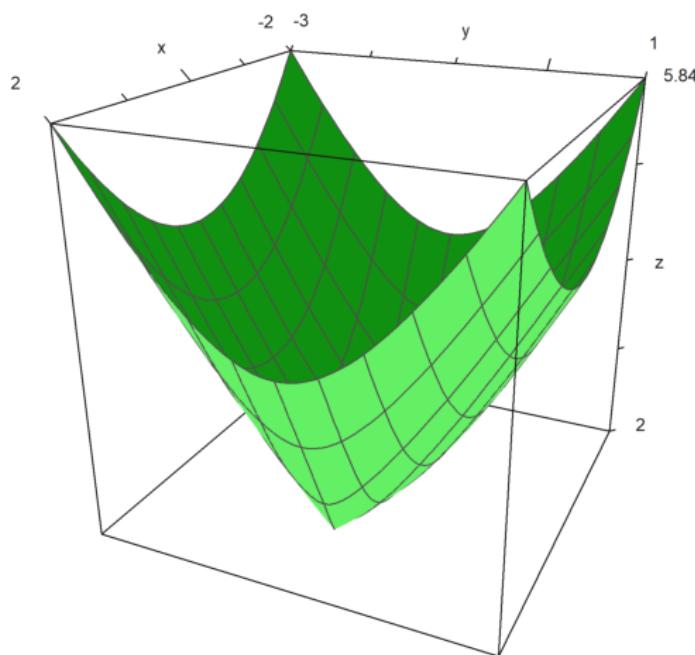
Sekarang kita melihat fungsi $MA + MB$ dimana A dan B adalah dua titik (tetap). Ini adalah "fakta yang terkenal" bahwa kurva level adalah elips, titik fokusnya adalah A dan B; kecuali untuk minimum AB yang konstan pada segmen [AB]:

```
>B=[1,-1];
>function d2(x,y):=d1(x,y)+sqrt((x-B[1])^2+(y-B[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
```



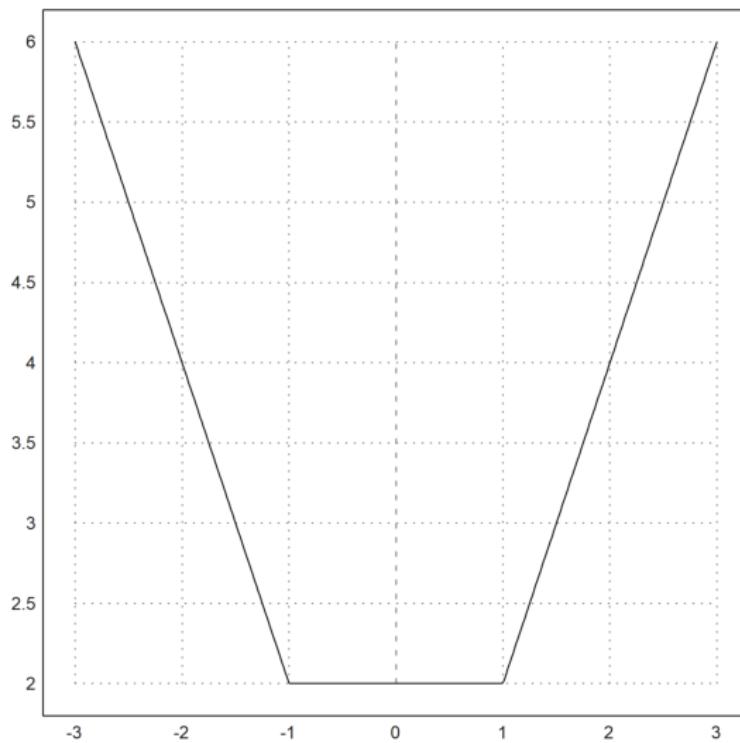
Grafiknya lebih menarik:

```
>plot3d("d2", xmin=-2, xmax=2, ymin=-3, ymax=1):
```



Batasan ke baris (AB) lebih terkenal:

```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```



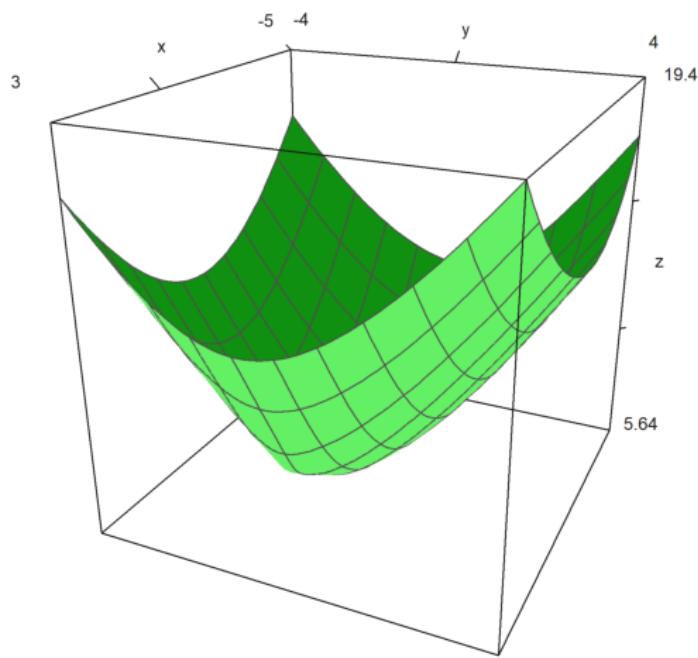
Three points

Sekarang hal-hal menjadi kurang sederhana: Sedikit kurang diketahui bahwa $MA + MB + MC$ mencapai minimumnya pada satu titik bidang tetapi untuk menentukannya kurang sederhana:

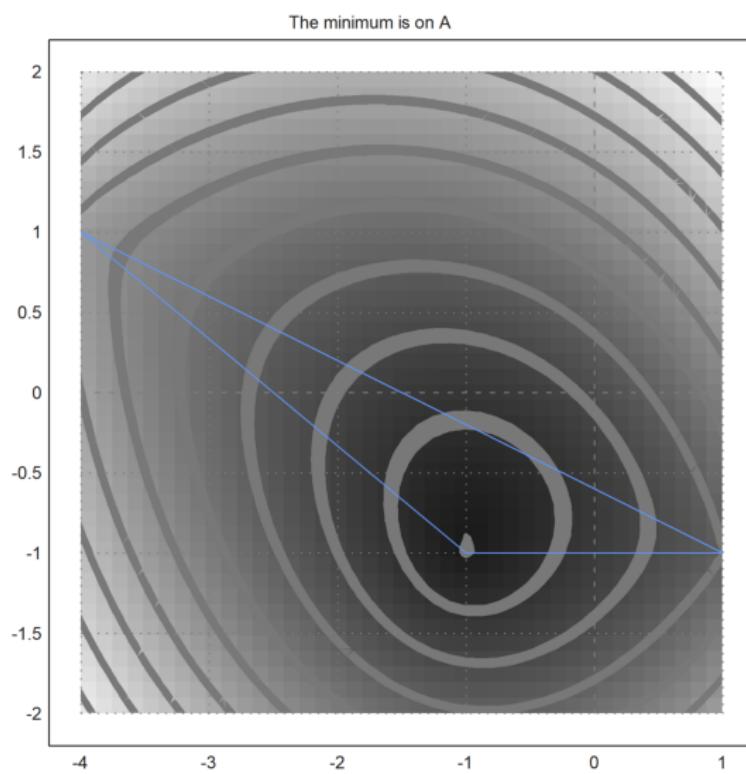
- 1) Jika salah satu sudut segitiga ABC lebih dari 120° (katakanlah dalam A), maka minimum tercapai pada titik ini (katakanlah AB + AC).

Contoh:

```
>C=[-4,1];
>function d3(x,y):=d2(x,y)+sqrt((x-C[1])^2+(y-C[2])^2)
>plot3d("d3",xmin=-5,xmax=3,ymin=-4,ymax=4);
>insimg;
```

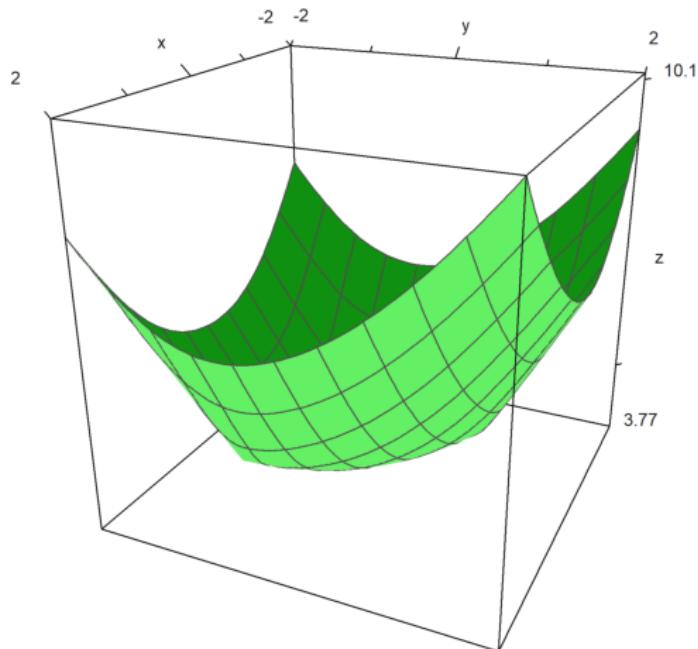


```
>fcontour("d3",xmin=-4,xmax=1,ymin=-2,ymax=2,hue=1,title="The minimum is on A");
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);
>insimg;
```

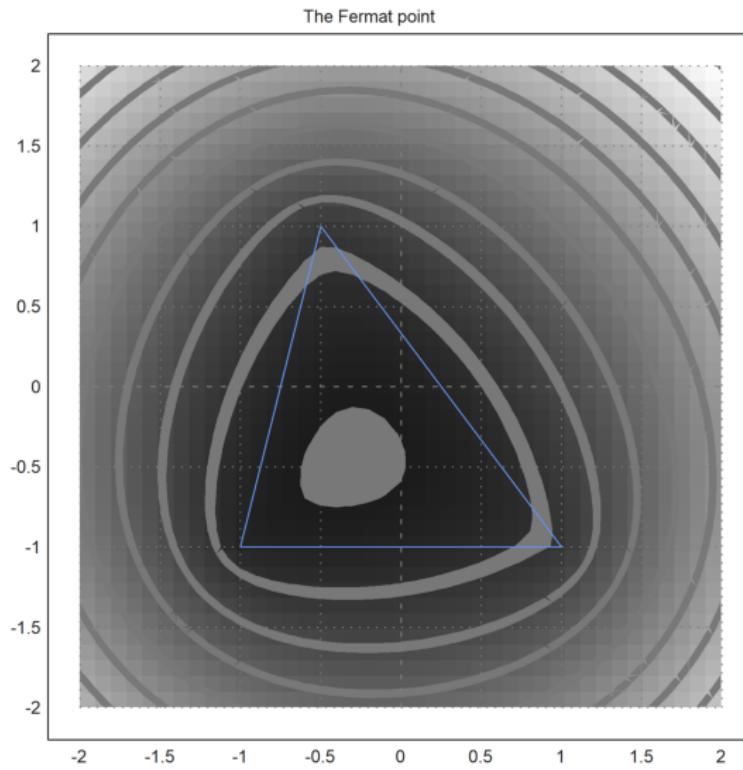


2) Tetapi jika semua sudut segitiga ABC kurang dari 120° , minimum berada pada titik F di bagian dalam segitiga, yang merupakan satu-satunya titik yang melihat sisi ABC dengan sudut yang sama (lalu masing-masing 120°) :

```
>C=[-0.5,1];  
>plot3d("d3",xmin=-2,xmax=2,ymin=-2,ymax=2);
```



```
>fcontour("d3",xmin=-2,xmax=2,ymin=-2,ymax=2,hue=1,title="The Fermat point");  
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);  
>insimg;
```



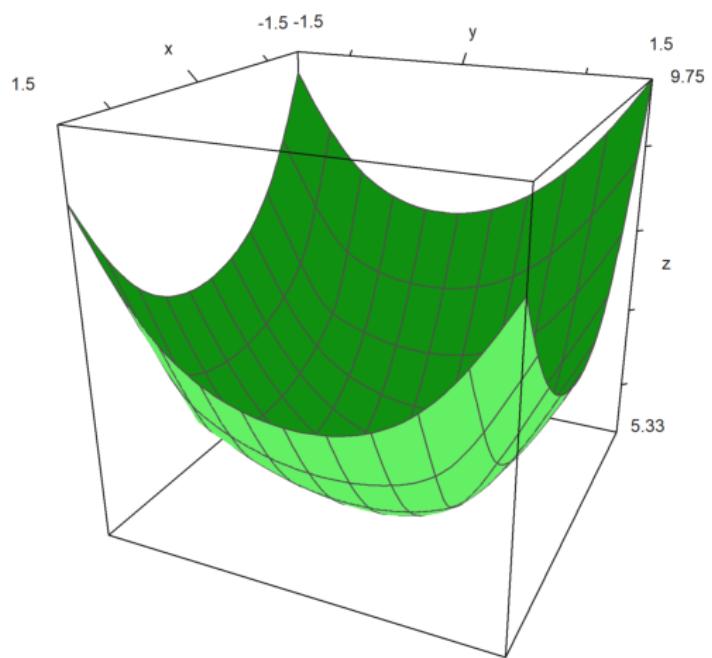
Merupakan kegiatan yang menarik untuk mewujudkan gambar di atas dengan perangkat lunak geometri; sebagai contoh, saya tahu soft tertulis di Java yang memiliki instruksi "garis kontur" ...

Semua ini di atas telah ditemukan oleh seorang hakim Prancis bernama Pierre de Fermat; dia menulis surat kepada para penggila lainnya seperti pendeta Marin Mersenne dan Blaise Pascal yang bekerja di bagian pajak penghasilan. Jadi titik unik F sehingga $FA + FB + FC$ minimal disebut titik Fermat segitiga. Tetapi tampaknya beberapa tahun sebelumnya, Torricelli Italia telah menemukan titik ini sebelum Fermat melakukannya! Pokoknya tradisinya adalah mencatat poin ini ...

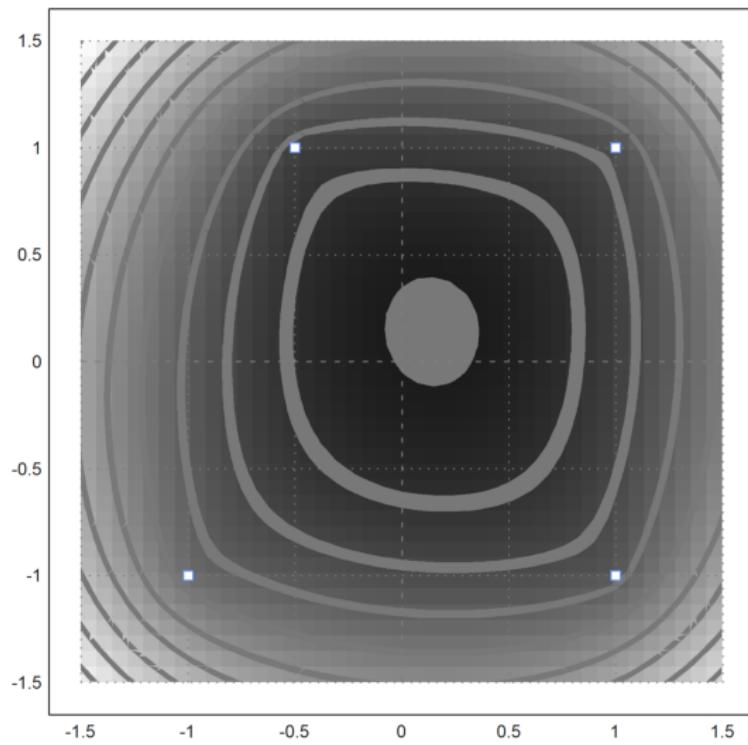
Four points

Langkah selanjutnya adalah menambahkan titik D ke-4 dan mencoba meminimalkan $MA + MB + MC + MD$; katakanlah bahwa Anda adalah operator TV kabel dan ingin mencari di bidang mana Anda harus meletakkan antena sehingga Anda dapat memberi makan empat desa dan menggunakan kabel sesedikit mungkin!

```
>D=[1,1];
>function d4(x,y):=d3(x,y)+sqrt((x-D[1])^2+(y-D[2])^2)
>plot3d("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5):
```



```
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)'; plot2d(P[1],P[2],points=1,add=1,color=12);
>insimg;
```



Masih ada minimum dan tidak ada yang dicapai pada simpul A, B, C atau D:

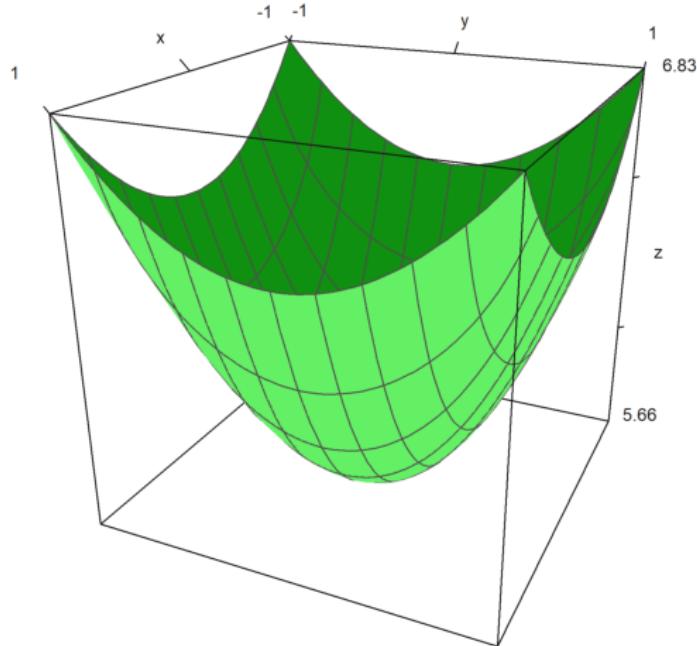
```
>function f(x):=d4(x[1],x[2])
>neldermin("f", [0.2,0.2])
```

```
[0.142858, 0.142857]
```

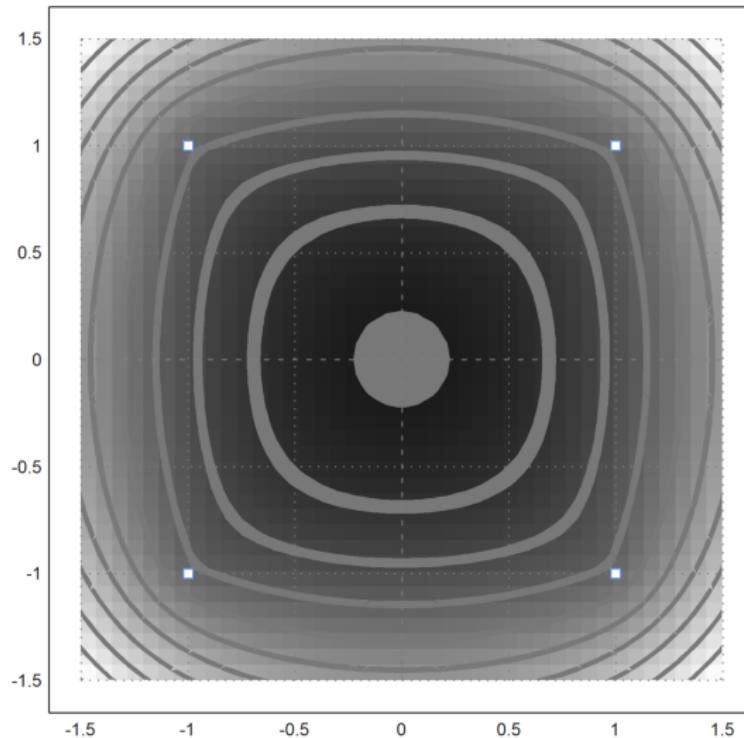
Tampaknya dalam kasus ini, koordinat titik optimal rasional atau mendekati rasional ...

Sekarang ABCD adalah bujur sangkar, kami berharap bahwa titik optimal adalah pusat ABCD:

```
>C=[-1,1];
>plot3d("d4",xmin=-1,xmax=1,ymin=-1,ymax=1):
```



```
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);
>P=(A_B_C_D)'; plot2d(P[1],P[2],add=1,color=12,points=1);
>insimg;
```



Contoh 7: Bola Dandelin dengan Povray

Anda dapat menjalankan demonstrasi ini, jika Anda memiliki Povray diinstal, dan pvcgine.exe di jalur program.

Pertama kami menghitung jari-jari bola.

Jika Anda melihat gambar di bawah, Anda melihat bahwa kita membutuhkan dua lingkaran yang menyentuh dua garis yang membentuk kerucut, dan satu garis yang membentuk bidang yang memotong kerucut.

Kami menggunakan file geometry.e dari Euler untuk ini.

```
>load geometry;
```

Pertama, dua garis yang membentuk kerucut.

```
>g1 &= lineThrough([0,0], [1,a])
```

```
[ - a, 1, 0 ]
```

```
>g2 &= lineThrough([0,0], [-1,a])
```

```
[ - a, - 1, 0 ]
```

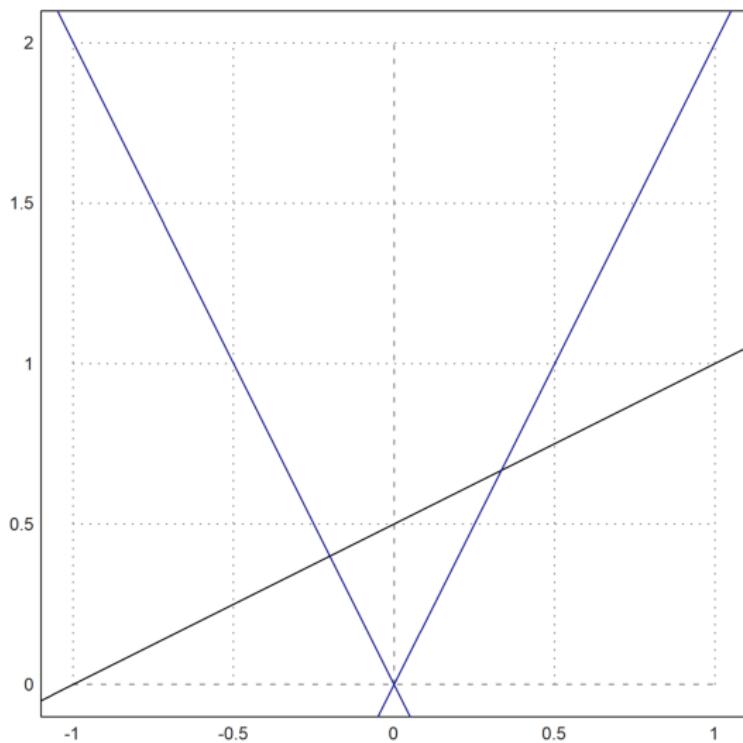
Lalu baris ketiga.

```
>g &= lineThrough([-1,0],[1,1])
```

```
[- 1, 2, 1]
```

Kami merencanakan semuanya sejauh ini.

```
>setPlotRange(-1,1,0,2);  
>color(black); plotLine(g(), "")  
>a:=2; color(blue); plotLine(g1(), ""), plotLine(g2(), ""):
```



Sekarang kita ambil titik umum pada sumbu y.

```
>P &= [0,u]
```

```
[0, u]
```

Hitung jarak ke g1.

```
>d1 &= distance(P,projectToLine(P,g1)); $d1
```

$$\sqrt{\left(\frac{a^2 u}{a^2 + 1} - u\right)^2 + \frac{a^2 u^2}{(a^2 + 1)^2}}$$

Hitung jarak ke g.

```
>d &= distance(P,projectToLine(P,g)); $d
```

$$\sqrt{\left(\frac{u+2}{5} - u\right)^2 + \frac{(2u-1)^2}{25}}$$

Dan temukan pusat kedua lingkaran, di mana jaraknya sama.

```
>sol &= solve(d1^2=d^2,u); $sol
```

$$\left[u = \frac{-\sqrt{5}\sqrt{a^2+1} + 2a^2 + 2}{4a^2 - 1}, u = \frac{\sqrt{5}\sqrt{a^2+1} + 2a^2 + 2}{4a^2 - 1} \right]$$

Ada dua solusi.

Kami mengevaluasi solusi simbolis, dan menemukan kedua pusat, dan kedua jarak.

```
>u := sol()
```

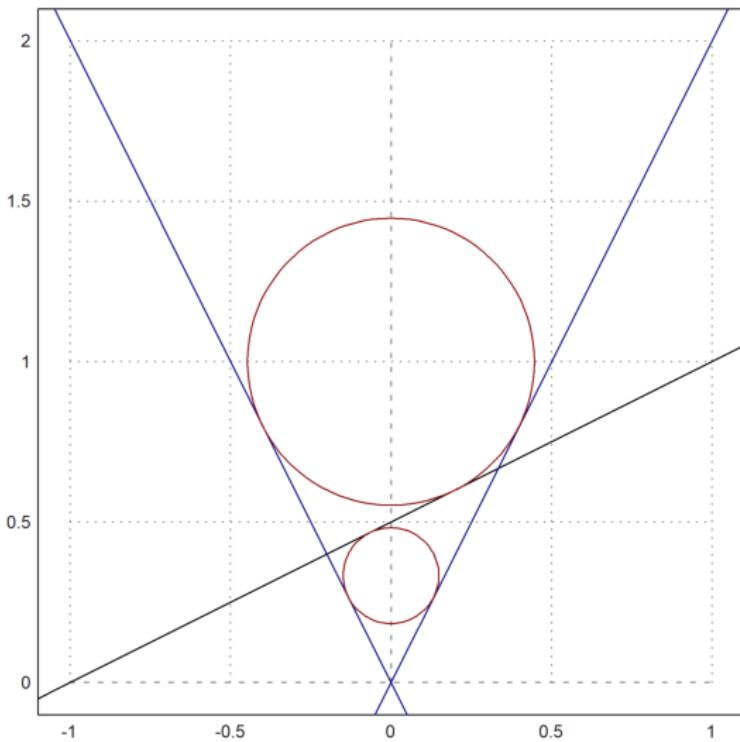
```
[0.333333, 1]
```

```
>dd := d()
```

```
[0.149071, 0.447214]
```

Plot lingkaran ke dalam gambar.

```
>color(red);
>plotCircle(circleWithCenter([0,u[1]],dd[1]), "");
>plotCircle(circleWithCenter([0,u[2]],dd[2]), "");
>insimg;
```



Plot dengan Povray

Selanjutnya kami merencanakan semuanya dengan Povray. Perhatikan bahwa Anda mengubah perintah apa pun dalam urutan perintah Povray berikut, dan menjalankan kembali semua perintah dengan Shift-Return. Pertama kita memuat fungsi povray.

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

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

Kami mengatur adegan dengan tepat.

```
>povstart(zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```

Selanjutnya kami menulis dua bidang ke file Povray.

```
>writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
>writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
```

Dan kerucutnya, transparan.

```
>writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
```

Kami menghasilkan pesawat terbatas pada kerucut.

```
>gp=g();  
>pc=povcone([0,0,0],0,[0,0,a],1,"");  
>vp=[gp[1],0,qp[2]]; dp=gp[3];  
>writeln(povplane(vp,dp,povlook(blue,0.5),pc));
```

Sekarang kami menghasilkan dua titik pada lingkaran, di mana bola menyentuh kerucut.

```
>function turnz(v) := return [-v[2],v[1],v[3]]  
>P1=projectToLine([0,u[1]],g()); P1=turnz([P1[1],0,P1[2]]);  
>writeln(povpoint(P1,povlook(yellow)));  
>P2=projectToLine([0,u[2]],g()); P2=turnz([P2[1],0,P2[2]]);  
>writeln(povpoint(P2,povlook(yellow)));
```

Kemudian kami menghasilkan dua titik di mana bola menyentuh bidang. Ini adalah fokus ellips.

```
>P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];  
>writeln(povpoint(P3,povlook(yellow)));  
>P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];  
>writeln(povpoint(P4,povlook(yellow)));
```

Selanjutnya kita menghitung perpotongan P1P2 dengan bidang.

```
>t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);  
>writeln(povpoint(P5,povlook(yellow)));
```

Kami menghubungkan titik dengan segmen garis.

```
>writeln(povsegment(P1,P2,povlook(yellow)));  
>writeln(povsegment(P5,P3,povlook(yellow)));  
>writeln(povsegment(P5,P4,povlook(yellow)));
```

Sekarang kami membuat pita abu-abu, di mana bola menyentuh kerucut.

```
>pcw=povcone([0,0,0],0,[0,0,a],1.01);  
>pc1=povcylinder([0,0,P1[3]-defaultpointsiz/2],[0,0,P1[3]+defaultpointsiz/2],1);  
>writeln(povintersection([pcw,pc1],povlook(gray)));  
>pc2=povcylinder([0,0,P2[3]-defaultpointsiz/2],[0,0,P2[3]+defaultpointsiz/2],1);  
>writeln(povintersection([pcw,pc2],povlook(gray)));
```

Mulai program Povray.

```
>povend();
```

Untuk mendapatkan Anaglyph ini, kita perlu memasukkan semuanya ke dalam fungsi scene. Fungsi ini akan digunakan dua kali nanti.

```
>function scene () ...  
  
    global a,u,dd,g,g1,defaultpointsize;  
    writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));  
    writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));  
    writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));  
    gp=g();  
    pc=povcone([0,0,0],0,[0,0,a],1,"");  
    vp=[gp[1],0,gp[2]]; dp=gp[3];  
    writeln(povplane(vp,dp,povlook(blue,0.5),pc));  
    P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);  
    writeln(povpoint(P1,povlook(yellow)));  
    P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);  
    writeln(povpoint(P2,povlook(yellow)));  
    P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];  
    writeln(povpoint(P3,povlook(yellow)));  
    P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];  
    writeln(povpoint(P4,povlook(yellow)));  
    t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);  
    writeln(povpoint(P5,povlook(yellow)));  
    writeln(povsegment(P1,P2,povlook(yellow)));  
    writeln(povsegment(P5,P3,povlook(yellow)));  
    writeln(povsegment(P5,P4,povlook(yellow)));  
    pcw=povcone([0,0,0],0,[0,0,a],1.01);  
    pc1=povcylinder([0,0,P1[3]-defaultpointsize/2],[0,0,P1[3]+defaultpointsize/2],1);  
    writeln(povintersection([pcw,pc1],povlook(gray)));  
    pc2=povcylinder([0,0,P2[3]-defaultpointsize/2],[0,0,P2[3]+defaultpointsize/2],1);  
    writeln(povintersection([pcw,pc2],povlook(gray)));  
endfunction
```

Anda membutuhkan kacamata merah / cyan untuk mengapresiasi efek berikut.

```
>povanaglyph("scene", zoom=11, center=[0,0,0.5], height=10°, angle=140°);
```

Contoh 8: Geometri Bumi

Di notebook ini, kami ingin melakukan beberapa komputasi bola. Fungsi-fungsi tersebut terdapat dalam file "spherical.e" di folder contoh. Kita perlu memuat file itu dulu.

```
>load "spherical.e";
```

Untuk memasukkan posisi geografis, kami menggunakan vektor dengan dua koordinat dalam radian (utara dan timur, nilai negatif untuk selatan dan barat). Berikut koordinat Kampus FMIPA UNY.

```
>FMIPA=[rad(-7,-46.467),rad(110,23.05)]
```

[-0.13569, 1.92657]

Anda dapat mencetak posisi ini dengan sposprint (cetak posisi bola).

```
>sposprint(FMIPA) // posisi garis lintang dan garis bujur FMIPA UNY
```

S $7^{\circ}46.467'$ E $110^{\circ}23.050'$

Mari kita tambahkan dua kota lagi, Solo dan Semarang.

```
>Solo=[rad(-7,-34.333),rad(110,49.683)]; Semarang=[rad(-6,-59.05),rad(110,24.533)];  
>sposprint(Solo), sposprint(Semarang),
```

S $7^{\circ}34.333'$ E $110^{\circ}49.683'$

S $6^{\circ}59.050'$ E $110^{\circ}24.533'$

Pertama kita menghitung vektor dari satu bola ke bola lainnya pada bola ideal. Vektor ini adalah [heading, distance] dalam radian. Untuk menghitung jarak di bumi, kita mengalikan dengan jari-jari bumi pada garis lintang 7° .

```
>br=svector(FMIPA,Solo); degprint(br[1]), br[2]*rearth( $7^{\circ}$ )->km // perkiraan jarak FMIPA-Solo
```

$65^{\circ}20'26.60''$

53.8945384608

Ini adalah perkiraan yang bagus. Rutinitas berikut menggunakan perkiraan yang lebih baik. Pada jarak yang begitu dekat hasilnya hampir sama.

```
>esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semarang
```

Commands must be separated by semicolon or comma!

Found: // perkiraan jarak FMIPA-Semarang (character 32)

You can disable this in the Options menu.

Error in:

```
esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semarang ...  
^
```

Ada fungsi untuk heading, dengan mempertimbangkan bentuk bumi yang elips. Sekali lagi, kami mencetak dengan cara yang canggih.

```
>sdegprint(esdir(FMIPA,Solo))
```

65.34°

1 Sudut segitiga melebihi 80° pada bola.

```
>asum=sangle(Solo,FMIPA,Semarang)+sangle(FMIPA,Solo,Semarang)+sangle(FMIPA,Semarang,Solo);
```

$180^{\circ}0'10.77''$

Ini dapat digunakan untuk menghitung luas segitiga. Catatan: Untuk segitiga kecil, ini tidak akurat karena kesalahan pengurangan dalam asum- π .

```
>(asum-pi)*rearth(48°)^2->" km^2"; // perkiraan luas segitiga FMIPA-Solo-Semarang
```

Ada fungsi untuk ini, yang menggunakan garis lintang rata-rata segitiga untuk menghitung jari-jari bumi, dan menangani kesalahan pembulatan untuk segitiga yang sangat kecil.

```
>esarea(Solo,FMIPA,Semarang)->" km^2", //perkiraan yang sama dengan fungsi esarea()
```

2123.64310526 km²

Kami juga dapat menambahkan vektor ke posisi. Vektor berisi heading dan jarak, keduanya dalam radian. Untuk mendapatkan vektor, kami menggunakan svector. Untuk menambahkan vektor ke posisi, kami menggunakan saddvector.

```
>v=svector(FMIPA,Solo); sposprint(saddvector(FMIPA,v)), sposprint(Solo),
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

Fungsi-fungsi ini mengasumsikan bola yang ideal. Hal yang sama di bumi.

```
>sposprint(esadd(FMIPA,esdir(FMIPA,Solo),esdist(FMIPA,Solo))), sposprint(Solo),
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

Mari kita beralih ke contoh yang lebih besar, Tugu Jogja dan Monas Jakarta (menggunakan Google Earth untuk mencari koordinatnya).

```
>Tugu=[-7.7833°,110.3661°]; Monas=[-6.175°,106.811944°];  
>sposprint(Tugu), sposprint(Monas)
```

S 7°46.998' E 110°21.966'
S 6°10.500' E 106°48.717'

Menurut Google Earth, jaraknya 429,66 km. Kami mendapatkan perkiraan yang bagus.

```
>esdist(Tugu,Monas)->" km" // perkiraan jarak Tugu Jogja - Monas Jakarta
```

Commands must be separated by semicolon or comma!
Found: // perkiraan jarak Tugu Jogja - Monas Jakarta (character 32)
You can disable this in the Options menu.
Error in:
esdist(Tugu,Monas)->" km" // perkiraan jarak Tugu Jogja - Mona ...
^

Judulnya sama dengan yang dihitung di Google Earth.

```
>degrprint(esdir(Tugu,Monas))
```

294°17'2.85''

However, we do no longer get the exact target position, if we add the heading and distance to the original position. This is so, since we do not compute the inverse function exactly, but take an approximation of the earth radius along the path.

```
>sposprint(esadd(Tugu, esdir(Tugu, Monas), esdist(Tugu, Monas)))
```

S 6°10.500' E 106°48.717'

The error is not large, however.

```
>sposprint(Monas),
```

S 6°10.500' E 106°48.717'

Tentunya kita tidak bisa berlayar dengan tujuan yang sama dari satu tujuan ke tujuan lainnya, jika kita ingin mengambil jalur terpendek. Bayangkan, Anda terbang NE mulai dari titik mana pun di bumi. Kemudian Anda akan berputar ke kutub utara. Lingkaran besar tidak mengikuti arah yang konstan!

Perhitungan berikut menunjukkan bahwa kami jauh dari tujuan yang benar, jika kami menggunakan tajuk yang sama selama perjalanan kami.

```
>dist=esdist(Tugu,Monas); hd=esdir(Tugu,Monas);
```

Sekarang kita tambahkan 10 kali sepersepuluh jaraknya, menggunakan heading ke Monas, kita sampai di Tugu.

```
>p=Tugu; loop 1 to 10; p=esadd(p,hd,dist/10); end;
```

Hasilnya masih jauh.

```
>sposprint(p), skmpprint(esdist(p,Monas))
```

S 6°11.250' E 106°48.372'
1.529km

Sebagai contoh lain, mari kita ambil dua titik di bumi pada ketinggian yang sama.

```
>P1=[30°,10°]; P2=[30°,50°];
```

Jalur terpendek dari P1 ke P2 bukanlah lingkaran dengan garis lintang 30 ?, tetapi jalur yang lebih pendek mulai 10 ? lebih jauh ke utara di P1.

```
>sdegprint(esdir(P1,P2))
```

79.69°

Tapi, jika kita mengikuti pembacaan kompas ini, kita akan berputar ke kutub utara! Jadi kita harus menyesuaikan arah tujuan kita di sepanjang jalan. Untuk tujuan kasar, kami menyesuaikannya pada 1/10 dari jarak total.

```
>p=P1; dist=esdist(P1,P2); ...
> loop 1 to 10; dir=esdir(p,P2); sdegprint(dir), p=esadd(p,dir,dist/10); end;
```

79.69°
81.67°
83.71°
85.78°
87.89°
90.00°
92.12°
94.22°
96.29°
98.33°

Jaraknya tidak tepat, karena kita akan menambahkan sedikit kesalahan, jika kita mengikuti tajuk yang sama terlalu lama.

```
>skmpprint(esdist(p,P2))
```

0.203km

Kami mendapatkan perkiraan yang baik, jika kami menyesuaikan heading setelah setiap 1/100 dari total jarak dari Tugu ke Monas.

```
>p=Tugu; dist=esdist(Tugu,Monas); ...
> loop 1 to 100; p=esadd(p,esdir(p,Monas),dist/100); end;
>skmpprint(esdist(p,Monas))
```

0.000km

Untuk keperluan navigasi, kita bisa mendapatkan urutan posisi GPS di sepanjang lingkaran besar menuju Monas dengan fungsi navigasi.

```
>load spherical; v=navigate(Tugu,Monas,10); ...
> loop 1 to rows(v); sposprint(v[#]), end;
```

S 7°46.998' E 110°21.966'
S 7°37.422' E 110°0.573'
S 7°27.829' E 109°39.196'
S 7°18.219' E 109°17.834'
S 7°8.592' E 108°56.488'
S 6°58.948' E 108°35.157'
S 6°49.289' E 108°13.841'
S 6°39.614' E 107°52.539'
S 6°29.924' E 107°31.251'
S 6°20.219' E 107°9.977'
S 6°10.500' E 106°48.717'

Kami menulis sebuah fungsi, yang menggambarkan bumi, dua posisi, dan posisi di antaranya.

```
>function testplot ...
```

```

useglobal;
plotearth;
plotpos(Tugu, "Tugu Jogja"); plotpos(Monas, "Tugu Monas");
plotposline(v);
endfunction

```

Now plot everything.

```
>plot3d("testplot", angle=25, height=6, >own, >user, zoom=4) :
```

Atau gunakan plot3d untuk mendapatkan tampilan anaglyphnya. Ini terlihat sangat bagus dengan kacamata merah / cyan.

```
>plot3d("testplot", angle=25, height=6, distance=5, own=1, anaglyph=1, zoom=4) :
```

Latihan

1. Gambarlah segi-n beraturan jika diketahui titik pusat O, n, dan jarak titik pusat ke titik-titik sudut segi-n tersebut (jari-jari lingkaran luar segi-n), r.

Petunjuk:

- Besar sudut pusat yang menghadap masing-masing sisi segi-n adalah $(360/n)$.
- Titik-titik sudut segi-n merupakan perpotongan lingkaran luar segi-n dan garis-garis yang melalui pusat dan saling membentuk sudut sebesar kelipatan $(360/n)$.
- Untuk n ganjil, pilih salah satu titik sudut adalah di atas.
- Untuk n genap, pilih 2 titik di kanan dan kiri lurus dengan titik pusat.
- Anda dapat menggambar segi-3, 4, 5, 6, 7, dst beraturan.

2. Gambarlah suatu parabola yang melalui 3 titik yang diketahui.

Petunjuk:

- Misalkan persamaan parabolanya $y = ax^2 + bx + c$.
- Subsitusikan koordinat titik-titik yang diketahui ke persamaan tersebut.
- Selesaikan SPL yang terbentuk untuk mendapatkan nilai-nilai a, b, c.

3. Gambarlah suatu segi-4 yang diketahui keempat titik sudutnya, misalnya A, B, C, D.

– Tentukan apakah segi-4 tersebut merupakan segi-4 garis singgung

(sisinya-sisinya merupakan garis singgung lingkaran yang sama yakni lingkaran dalam segi-4 tersebut).

– Suatu segi-4 merupakan segi-4 garis singgung apabila keempat

garis bagi sudutnya bertemu di satu titik.

– Jika segi-4 tersebut merupakan segi-4 garis singgung, gambar

lingkaran dalamnya.

– Tunjukkan bahwa syarat suatu segi-4 merupakan segi-4 garis

singgung apabila hasil kali panjang sisi-sisi yang berhadapan sama.

4. Gambarlah suatu ellips jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P

dan Q adalah tempat kedudukan titik-titik yang jumlah jarak ke P dan ke Q selalu sama (konstan).

5. Gambarlah suatu hiperbola jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P

dan Q adalah tempat kedudukan titik-titik yang selisih jarak ke P dan ke Q selalu sama (konstan).

JAWAB:

1.

```
>o &:= [0,0]; c=circleWithCenter(o,5);
>color(1); setPlotRange(5); plotPoint(o); plotCircle(c);
>A=[-5,0]; plotPoint(A,"A");
>B=[5,0]; plotPoint(B,"B");
>plotSegment(A,B,"");
>c1=circleWithCenter(A,distance(A,o));
>c2=circleWithCenter(B,distance(B,o));
>k=circleCircleIntersections(c1,c);
>l=circleCircleIntersections(c,c2);
>m=circleCircleIntersections(c2,c);
>n=circleCircleIntersections(c,c1);
>r=lineThrough(k,m); s=lineThrough(l,n);
>setPlotRange(8); plotPoint(o); plotCircle(c); plotPoint(A,"A"); plotPoint(B,"B"); plotSeg
>color(4); plotCircle(c1); plotCircle(c2); plotPoint(k); plotPoint(l); plotPoint(m); plotP
>color(5); plotLine(r); plotLine(s);
>color(6); plotSegment(A,k,""); plotSegment(A,n,""); plotSegment(k,l,""); plotSegment(l,B,
```

No. 2

```
>load geometry;
>setPlotRange(5); P=[1,0]; Q=[4,0]; R=[0,-4];
>plotPoint(P,"P"); plotPoint(Q,"Q"); plotPoint(R,"R");
>sol &= solve([a+b=-c,16*a+4*b=-c,c=-4],[a,b,c])
```

[[a = - 1, b = 5, c = - 4]]

```
>function y&=-x^2+5*x-4
```

$$-\frac{2}{x^2 + 5x - 4}$$

```
>plot2d("-x^2+5*x-4",-5,5,-5,5):
```

No.3

```

>setPlotRange(-5,5,-5,5);
>A=[-2,2]; plotPoint(A,"A");
>B=[2,2]; plotPoint(B,"B");
>C=[2,-2]; plotPoint(C,"C");
>D=[-2,-2]; plotPoint(D,"D");
>plotSegment(A,B);
>plotSegment(B,C);
>plotSegment(C,D);
>plotSegment(D,A);
>plotSegment(A,C,"q1");
>plotSegment(B,D,"q2");
>q1=lineThrough(A,C);
>q2=lineThrough(B,D);
>p=lineIntersection(q1,q2);
>plotLine(q1); plotLine(q2);
>plotPoint(p, "P");
>r=norm(p-projectToLine(p,lineThrough(A,B)))

```

2

```
>plotCircle(circleWithCenter(p,r),"lingkaran dalam segi-4 ABCD"):
```

no.4

```

>P=[-1,-1]; Q=[1,-1];
>function d1(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)
>Q=[1,-1]; function d2(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)+sqrt((x-Q[1])^2+(y-Q[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):

```

Grafik yang lebih menarik

```
>plot3d("d2",xmin=-2,xmax=2,ymin=-3,ymax=1):
```

Batasan ke garis PQ

```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```

no.5

```

>P=[-1,-1]; Q=[1,-1];
>function d1(x,y):=sqrt((x-p[1])^2+(y-p[2])^2)
>Q=[1,-1]; function d2(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)+sqrt((x+Q[1])^2+(y+Q[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):

```

Grafik yang lebih menarik

```
>plot3d("d2",xmin=-2,xmax=2,ymin=-3,ymax=1):  
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```

BAB 6

KB PEKAN 10; MENGGUNAKAN EMT UNTUK STATISTIKA

[a4paper,10pt]article eumat

EMT untuk Statistika

Dalam buku catatan ini, kami mendemonstrasikan plot statistik utama, pengujian, dan distribusi di Euler. Mari kita mulai dengan beberapa statistik deskriptif. Ini bukan pengantar statistik. Jadi, Anda mungkin memerlukan beberapa latar belakang untuk memahami detailnya.

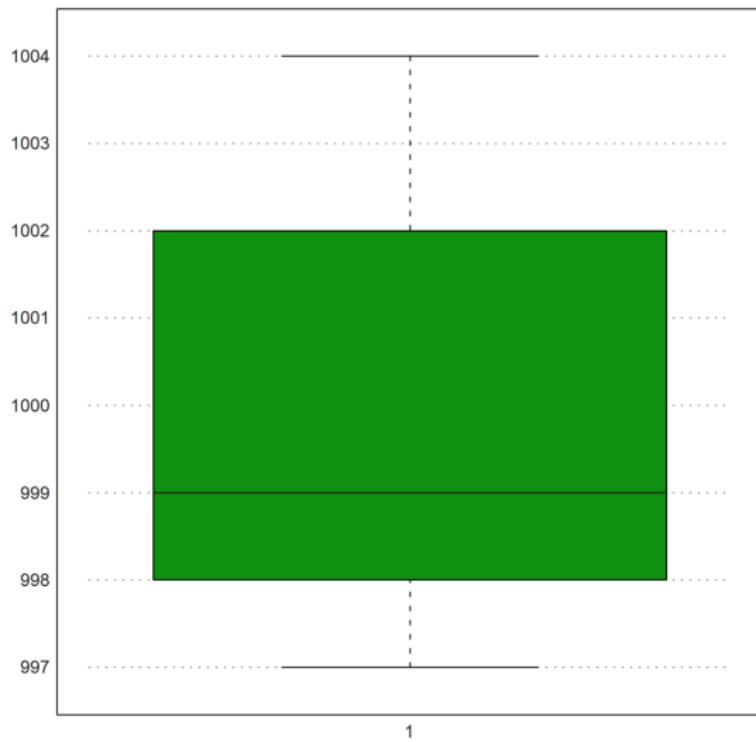
Asumsikan pengukuran berikut. Kami ingin menghitung nilai rata-rata dan standar deviasi yang diukur.

```
>M=[1000,1004,998,997,1002,1001,998,1004,998,997]; ...
>mean(M), dev(M),
```

999.9
2.7264

We can plot the box-and-whiskers plot for the data. In our case there are no outliers.

```
>boxplot(M) :
```



We compute the probability that a value is bigger than 1005, assuming the measured values and a normal distribution.

All functions for distributions in Euler end with ...dis and compute the cumulative probability distribution (CPF).

$$\text{normaldis}(x, m, d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

We print the result in % with 2 digits accuracy using the print function.

```
>print((1-normaldis(1005,mean(M),dev(M)))*100,2,unit=" %")
```

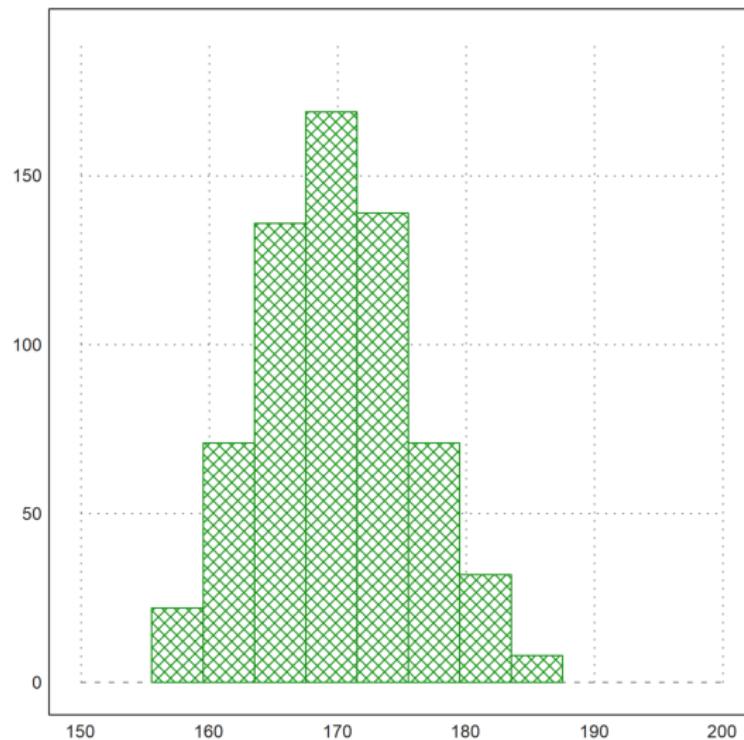
3.07 %

For the next example, we assume the following numbers of men in given a size ranges.

```
>r=155.5:4:187.5; v=[22,71,136,169,139,71,32,8];
```

Here is a plot of the distribution.

```
>plot2d(r,v,a=150,b=200,c=0,d=190,bar=1,style="/"):
```



We can put such raw data into a table.

Tables are a method to store statistical data. Our table should contain three columns: Start of range, end of range, number of men in the range.

Tables can be printed with headers. We use a vector of strings to set the headers.

```
>T:=r[1:8]' | r[2:9]' | v'; writetable(T,labc=["from","to","count"])
```

from	to	count
155.5	159.5	22
159.5	163.5	71
163.5	167.5	136
167.5	171.5	169
171.5	175.5	139
175.5	179.5	71
179.5	183.5	32
183.5	187.5	8

If we need the mean value and other statistics of the sizes, we need to compute the midpoint of the ranges. We can use the first two columns of our table for this.

The symbol "`|`" is used to separate column, the function "writetable" is used to write the table, with options "labc" is to specify column headers.

```
>(T[,1]+T[,2])/2 // the midpoint of each interval
```

```
157.5
161.5
165.5
169.5
173.5
```

177.5
181.5
185.5

But it is easier, to fold the ranges with the vector [1/2,1/2].

```
>M=fold(r, [0.5, 0.5])
```

[157.5, 161.5, 165.5, 169.5, 173.5, 177.5, 181.5, 185.5]

Now we can compute the mean and deviation of the sample with the given frequencies.

```
>{m, d}=meandev(M, v); m, d,
```

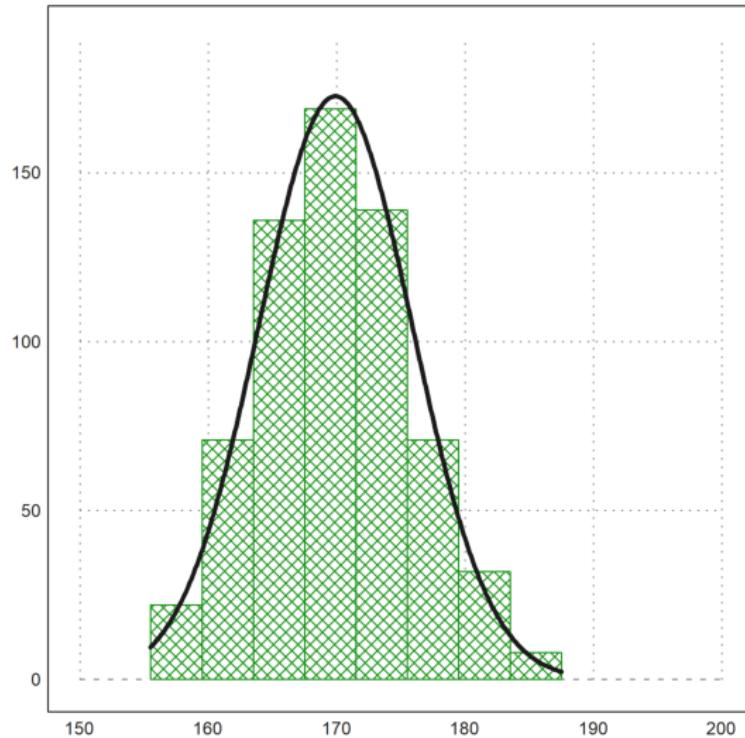
169.9
5.9891

Let us add the normal distribution of the values to the above bar plot. The formula for normal distribution with mean m and standard deviation d is:

$$y = \frac{1}{d\sqrt{2\pi}} e^{\frac{-(x-m)^2}{2d^2}}.$$

Because its values are between 0 and 1, to plot it on the bar plot it must be multiplied by 4 times the total number of data.

```
>plot2d("qnormal(x,m,d)*sum(v)*4", ...
> xmin=min(r), xmax=max(r), thickness=3, add=1):
```



Tables

In the directory of this notebook you find a file with a table. The data represent the results of a survey. Here are the first four lines of the file. The data are from an German online book "Einführung in die Statistik mit R" by A. Handl.

```
>printfile("table.dat", 4);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
printfile:
  open(filename, "r");
```

The table contains 7 columns of numbers or tokens (strings). We want read the table from the file. First, we use our own translation for the tokens.

To this, we define sets of tokens. The function strtokens() gets a string vector of tokens from a given string.

```
>mf:=[ "m", "f" ]; yn:=[ "y", "n" ]; ev:=strtokens("g vg m b vb");
```

Now we read the table with these translations.

The arguments tok2, tok4 etc. are the translations of the columns of the table. These arguments are not in the parameter list of readtable(), so you need to provide them with "=:".

```
>{MT,hd}=readtable("table.dat",tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
  if filename!=none then open(filename, "r"); endif;
```

```
>load over statistics;
```

For printing, we need to specify the same token sets. We print the first four lines only.

```
>writetable(MT[1:4],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
MT is not a variable!
Error in:
writetable(MT[1:4],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok ...
^
```

The dots "..." represent values, which are not available.

If we do not want to specify the tokens for the translation in advance, we only need to specify, which columns contain tokens and not numbers.

```
>ctok=[2,4,5,7]; {MT,hd,tok}=readtable("table.dat",ctok=ctok);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
    if filename!=none then open(filename,"r"); endif;
```

The function `readtable()` now returns a set of tokens.

```
>tok
```

```
f
m
```

The table contains the entries from the file with tokens translated to numbers.

The special string `NA="."` is interpreted as "Not Available", and gets `NAN` (not a number) in the table. This translation can be changed with the parameters `NA`, and `NAval`.

```
>MT [1]
```

```
MT is not a variable!
Error in:
MT[1] ...
^
```

Here is the content of the table with untranslated numbers.

```
>writetable(MT,wc=5)
```

```
Variable or function MT not found.
Error in:
writetable(MT,wc=5) ...
^
```

For convenience, you can put the output of `readtable()` into a list.

```
>Table={{readtable("table.dat",ctok=ctok)}};
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
    if filename!=none then open(filename,"r"); endif;
```

Using the same token columns and the tokens read from the file, we can print the table. We can either specify `ctok`, `tok`, etc. or use the list `Table`.

```
>writetable(Table,ctok=ctok,wc=5);
```

```
Variable or function Table not found.
Error in:
writetable(Table,ctok=ctok,wc=5); ...
^
```

The function `tablecol()` returns the values of columns of the table, skipping any rows with NAN values("." in the file), and the indices of the columns, which contain these values.

```
>{c,i}=tablecol(MT,[5,6]);
```

```
Variable or function MT not found.  
Error in:  
{c,i}=tablecol(MT,[5,6]); ...  
^
```

We can use this to extract columns from the table for a new table.

```
>j=[1,5,6]; writetable(MT[i,j],labc=hd[j],ctok=[2],tok=tok)
```

```
MT is not a variable!  
Error in:  
j=[1,5,6]; writetable(MT[i,j],labc=hd[j],ctok=[2],tok=tok) ...  
^
```

Of course, we need to extract the table itself from the list `Table` in this case.

```
>MT=Table[1];
```

```
Table is not a variable!  
Error in:  
MT=Table[1]; ...  
^
```

Of course, we can also use it to determine the mean value of a column or any other statistical value.

```
>mean(tablecol(MT,6))
```

```
Variable or function MT not found.  
Error in:  
mean(tablecol(MT,6)) ...  
^
```

The `getstatistics()` function returns the elements in a vector, and their counts. We apply it to the "m" and "f" values in the second column of our table.

```
>{xu,count}=getstatistics(tablecol(MT,2)); xu, count,
```

```
Variable or function MT not found.  
Error in:  
{xu,count}=getstatistics(tablecol(MT,2)); xu, count, ...  
^
```

We can print the result in a new table.

```
>writetable(count',labr=tok[xu])
```

```
Variable count not found!  
Error in:  
writetable(count',labr=tok[xu]) ...  
^
```

The function selectable() returns a new table with the values in one column selected from a vector of indices. First we look up the indices of two of our values in the token table.

```
>v:=indexof(tok, ["g", "vg"])
```

```
[0, 0]
```

Now we can select the rows of the table, which have any of the values in v in their 5-th row.

```
>MT1:=MT[selectrows(MT, 5, v)]; i:=sortedrows(MT1, 5);
```

```
Variable or function MT not found.  
Error in:  
MT1:=MT[selectrows(MT, 5, v)]; i:=sortedrows(MT1, 5); ...  
^
```

Now we can print the table, with extracted and sorted values in the 5-th column.

```
>writetable(MT1[i], labc=hd, ctok=ctok, tok=tok, wc=7);
```

```
MT1 is not a variable!  
Error in:  
writetable(MT1[i], labc=hd, ctok=ctok, tok=tok, wc=7); ...  
^
```

For the next statistic, we want to relate two columns of the table. So we extract column 2 and 4 and sort the table.

```
>i=sortedrows(MT, [2, 4]); ...  
> writetable(tablecol(MT[i], [2, 4])', ctok=[1, 2], tok=tok)
```

```
Variable or function MT not found.  
Error in:  
i=sortedrows(MT, [2, 4]);      writetable(tablecol(MT[i], [2, 4])', c ...  
^
```

With getstatistics(), we can also relate the counts in two columns of the table to each other.

```
>MT24=tablecol(MT, [2, 4]); ...  
>{xu1, xu2, count}=getstatistics(MT24[1], MT24[2]); ...  
>writetable(count, labr=tok[xu1], labc=tok[xu2])
```

```
Variable or function MT not found.  
Error in:  
MT24=tablecol(MT, [2, 4]); {xu1, xu2, count}=getstatistics(MT24[1] ...  
^
```

A table can be written to a file.

```
>filename="test.dat"; ...  
>writetable(count, labr=tok[xu1], labc=tok[xu2], file=filename);
```

```

Variable or function count not found.
Error in:
filename="test.dat"; writetable(count,labr=tok[xul],labc=tok[x ...
^

```

Then we can read the table from the file.

```

>{MT2,hd,tok2,hdr}=readtable(filename,>clabs,>rlabs); ...
>writetable(MT2,labr=hdr,labc=hd)

```

```

Could not open the file
test.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
  if filename!=none then open(filename,"r"); endif;

```

And delete the file.

```
>fileremove(filename);
```

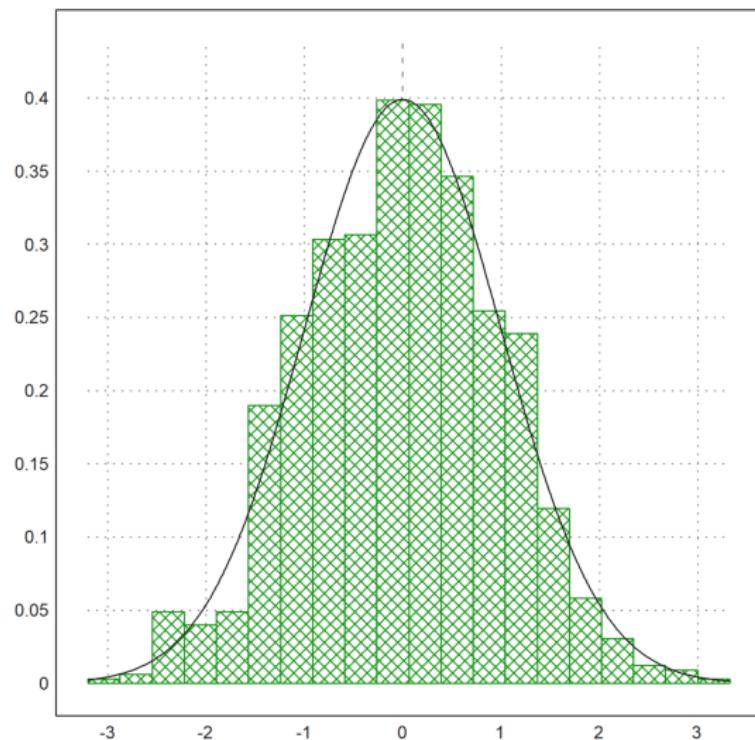
Distributions

With plot2d, there is a very easy method to plot a distribution of experimental data.

```

>p=normal(1,1000); //1000 random normal-distributed sample p
>plot2d(p,distribution=20,style="/"); // plot the random sample p
>plot2d("qnormal(x,0,1)",add=1); // add the standard normal distribution plot

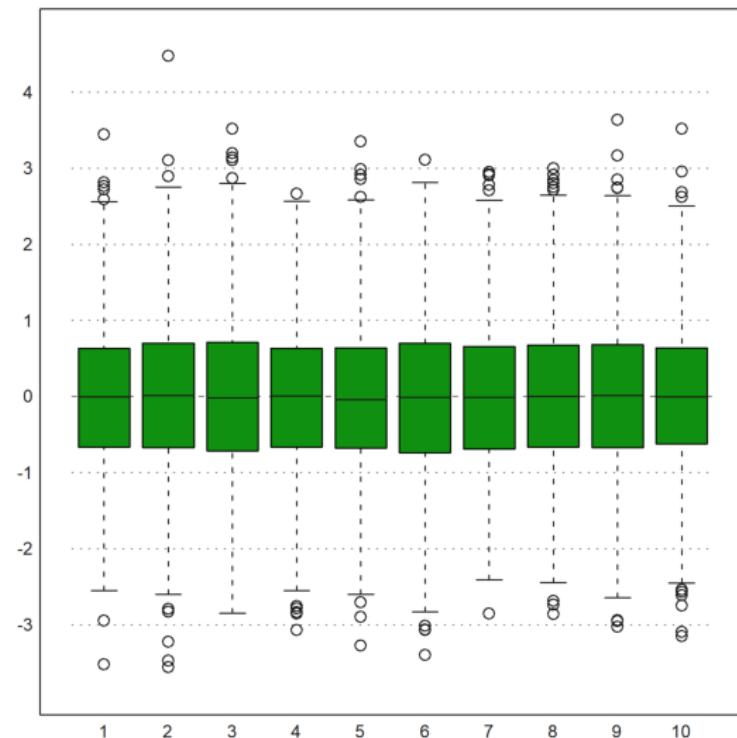
```



Please note the different between the bar plot (sample) and the normal curve (the real distribution) . Reenter the three commands to see another sampling result.

Here is a comparison of 10 simulations of 1000 normal distributed values using a so-called box plot. This plot shows the median, the 25% and 75% quartiles, the minimal and maximal values, and the outliers.

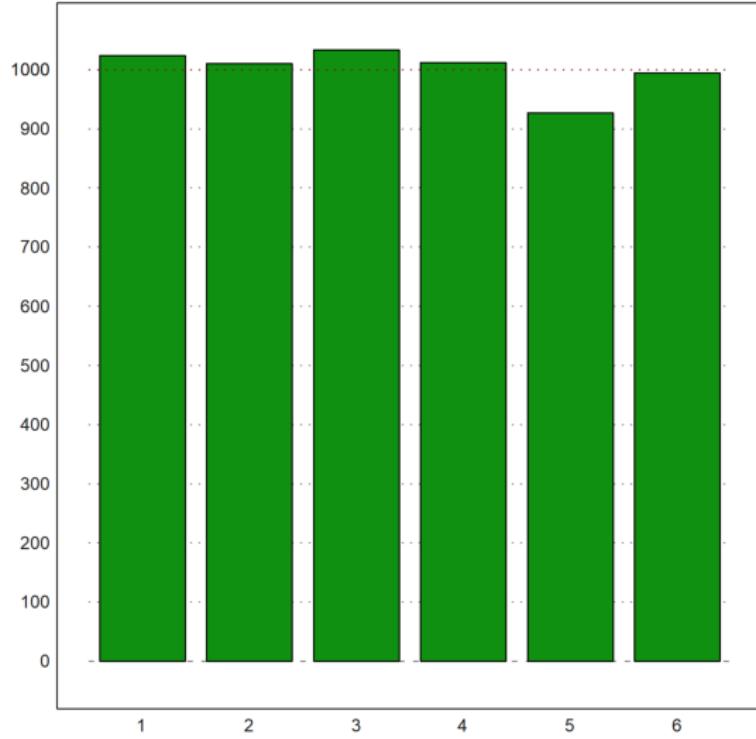
```
>p=normal(10,1000); boxplot(p):
```



To generate random integers, Euler has `intrandom`. Let us simulate dice throws and plot the distribution.

We use the `getmultiplicities(v,x)` function, which counts how often the elements of `v` appear in `x`. Then we plot the the result using `columnsplot()`.

```
>k=intrandom(1,6000,6); ...
>columnsplot(getmultiplicities(1:6,k)); ...
>ygrid(1000,color=red):
```



While `intrandom(n,m,k)` returns uniformly distributed integers from 1 to k, it is possible to use any other given distribution of integers with `randpint()`.

In the following example, the probabilities for 1,2,3 are 0.4,0.1,0.5 respectively.

```
>randpint(1,1000,[0.4,0.1,0.5]); getmultiplicities(1:3,%)
```

[409, 89, 502]

Euler can produce random values from more distributions. Have a look into the reference.

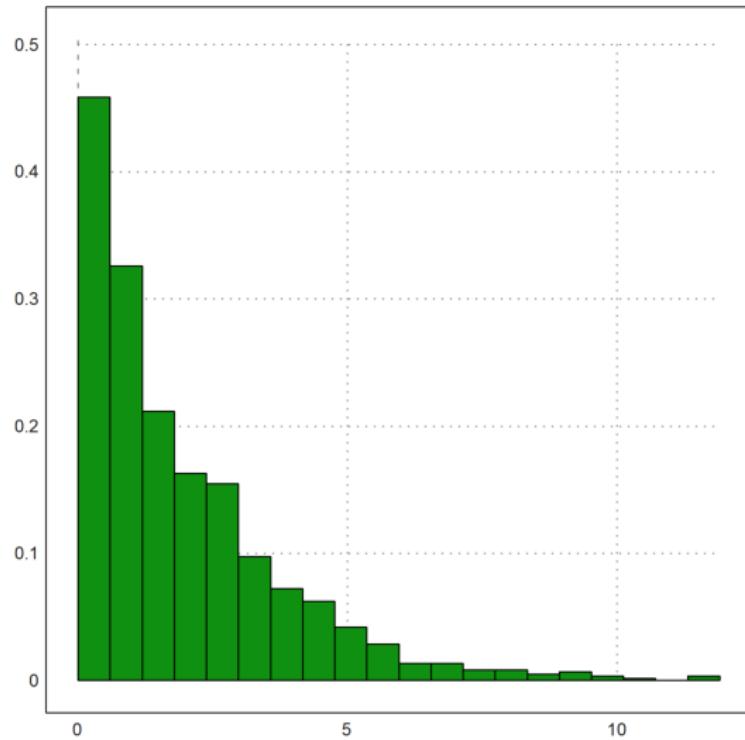
E.g., we try the exponential distribution. A continuous random variable X is said to have an exponential distribution, if its PDF is given by

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0,$$

with parameter

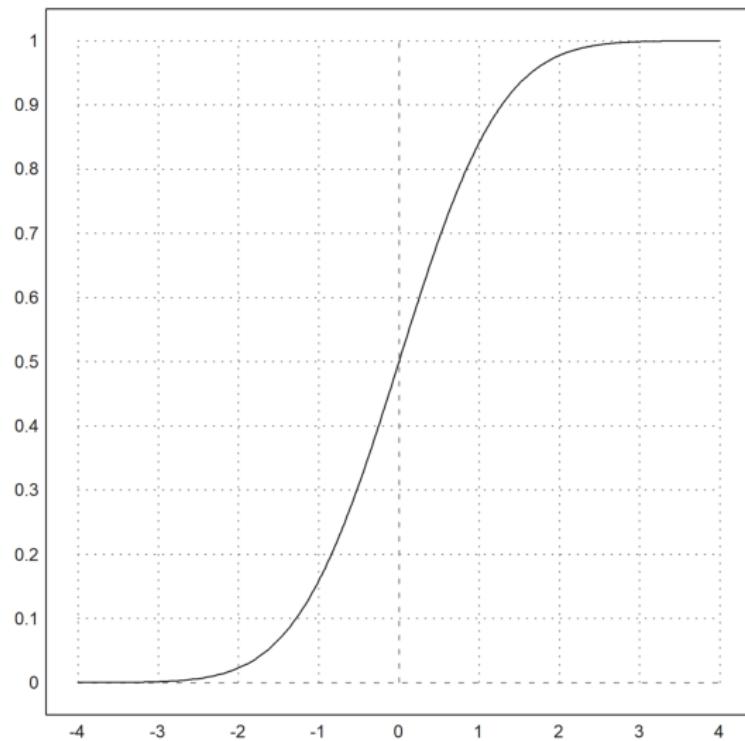
$$\lambda = \frac{1}{\mu}, \quad \mu \text{ is the mean, and denoted by } X \sim \text{Exponential}(\lambda).$$

```
>plot2d(randexponential(1,1000,2),>distribution):
```



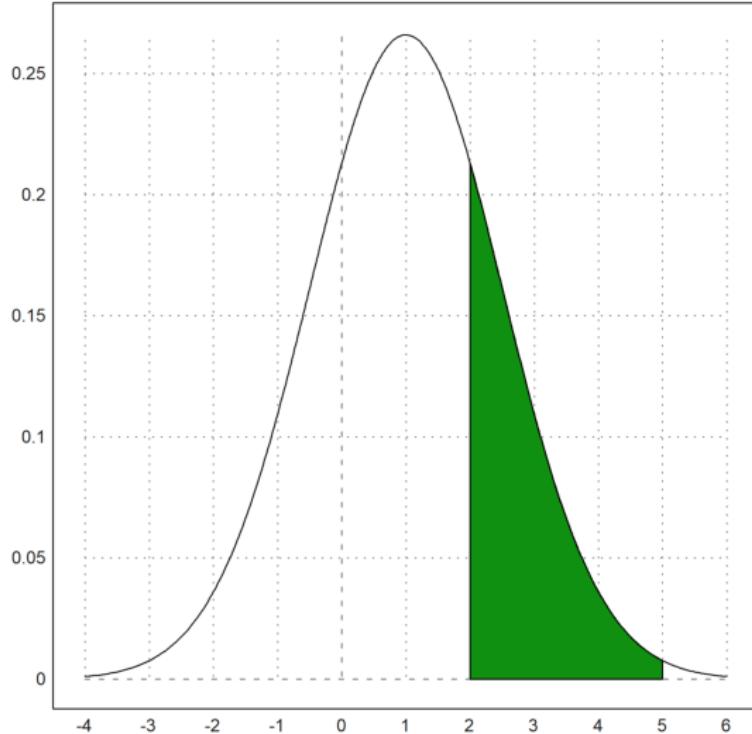
For many distributions, Euler can compute the distribution function and the inverse.

```
>plot2d("normaldis", -4, 4) :
```



The following is one way to plot a quantile.

```
>plot2d("qnormal(x,1,1.5)",-4,6); ...
>plot2d("qnormal(x,1,1.5)",a=2,b=5,>add,>filled):
```



$$\text{normaldis}(x,m,d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

The probability to be in the green area is the following.

```
>normaldis(5,1,1.5)-normaldis(2,1,1.5)
```

0.24866

This can be computed numerically with the following integral.

$$\int_2^5 \frac{1}{1.5\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-1}{1.5})^2} dx.$$

```
>gauss("qnormal(x,1,1.5)",2,5)
```

0.24866

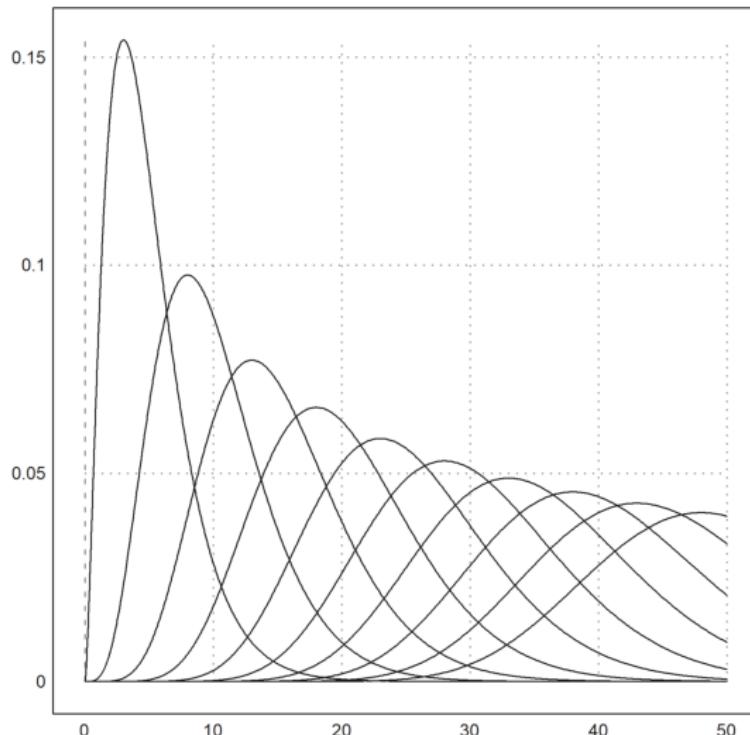
Let us compare the binomial distribution with the normal distribution of same mean and deviation. The function `invbindis()` solves a linear interpolation between integer values.

```
>invbindis(0.95,1000,0.5), invnormaldis(0.95,500,0.5*sqrt(1000))
```

```
525.52  
526.01
```

The function qdis() is the density of the chi-square distribution. As usual, Euler maps vectors to this function. Thus we get a plot of all chi-square distributions with degrees 5 to 30 easily in the following way.

```
>plot2d("qchidis(x,(5:5:50)')",0,50):
```



Euler has accurate functions to evaluate distributions. Let us check chidis() with an integral.

The naming tries to be consistent. E.g.,

- the chi-square distribution is chidis(),
- the inverse function is invchidis(),
- the density is qchidis().

The complements of the distribution (upper tail) is chicdis().

```
>chidis(1.5,2), integrate("qchidis(x,2)",0,1.5)
```

```
0.52763  
0.52763
```

Discrete Distributions

To define your own discrete distribution, you can use the following method.

First we set the distribution function.

```
>wd = 0 | ((1:6)+[-0.01,0.01,0,0,0,0])/6
```

```
[0, 0.165, 0.335, 0.5, 0.66667, 0.83333, 1]
```

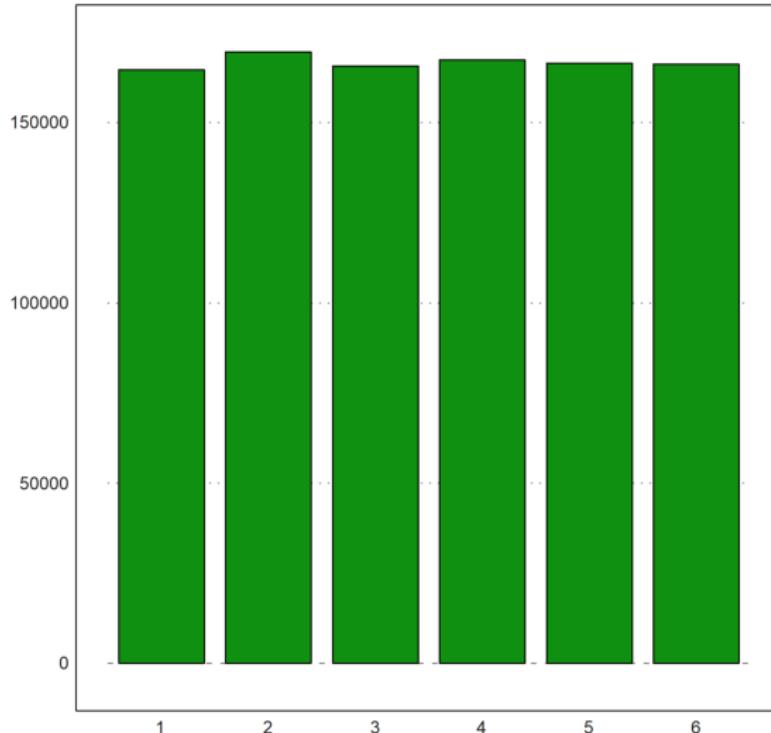
The meaning is that with probability $wd[i+1]-wd[i]$ we produce the random value i .

This is almost a uniform distribution. Let us define a random number generator for this. The `find(v,x)` function finds the value x in the vector v . It works for vectors x too.

```
>function wrongdice (n,m) := find(wd,random(n,m))
```

The error is so subtle that we see it only with very many iterations.

```
>columnsplot (getmultiplicities(1:6,wrongdice(1,1000000))):
```



Here is a simple function to check for uniform distribution of the values $1 \dots K$ in v . We accept the result, if for all frequencies

$$\left| f_i - \frac{1}{K} \right| < \frac{\delta}{\sqrt{n}}.$$

```
>function checkrandom (v, delta=1) ...
```

```
K=max(v); n=cols(v);
fr=getfrequencies(v,1:K);
return max(fr/n-1/K)<delta/sqrt(n);
endfunction
```

Indeed the function rejects the uniform distribution.

```
>checkrandom(wrongdice(1,1000000))
```

0

And it accepts the built-in random generator.

```
>checkrandom(intrandom(1,1000000,6))
```

1

We can compute the binomial distribution. First there is binomialsim(), which returns the probability of i or less hits out of n trials.

```
>bindis(410,1000,0.4)
```

0.7514

The inverse Beta function is used to compute a Clopper-Pearson confidence interval for the parameter p. The default level is alpha.

The meaning of this interval is that if p is outside the interval, the observed result of 410 in 1000 is rare.

```
>clopperpearson(410,1000)
```

[0.37932, 0.44121]

The following commands are the direct way to get the above result. But for large n, the direct summation is not accurate and slow.

```
>p=0.4; i=0:410; n=1000; sum(bin(n,i)*p^i*(1-p)^(n-i))
```

0.7514

By the way, invbinsum() computes the inverse of binomialsim().

```
>invbindis(0.75,1000,0.4)
```

409.93

In Bridge, we assume 5 outstanding cards (out of 52) in two hands (26 cards). Let us compute the probability of a distribution worse than 3:2 (e.g. 0:5, 1:4, 4:1 or 5:0).

```
>2*hypergeomsum(1,5,13,26)
```

0.32174

There is also a simulation of multinomial distributions.

```
>randmultinomial(10,1000,[0.4,0.1,0.5])
```

376	109	515
374	108	518
418	101	481
372	91	537
405	103	492
386	106	508
382	99	519
427	118	455
417	108	475
361	103	536

Plotting Data

To plot data, we try the results of the German elections since 1990, measured in seats.

```
>BW := [ ...
>1990,662,319,239,79,8,17; ...
>1994,672,294,252,47,49,30; ...
>1998,669,245,298,43,47,36; ...
>2002,603,248,251,47,55,2; ...
>2005,614,226,222,61,51,54; ...
>2009,622,239,146,93,68,76; ...
>2013,631,311,193,0,63,64];
```

For the parties, we use a string of names.

```
>P:=[ "CDU/CSU", "SPD", "FDP", "Gr", "Li"];
```

Let us print the percentages nicely.

First we extract the necessary columns. Columns 3 to 7 are the seats of each party, and column 2 is the total number of seats. column 1 is the year of the election.

```
>BT:=BW[,3:7]; BT:=BT/sum(BT); YT:=BW[,1]';
```

Then we print the statistics in table form. We use the names as column headers, and the years as headers for the rows. The default width for the columns is `wc=10`, but we prefer a denser output. The columns will be expanded for the labels of the columns, if necessary.

```
>writetable(BT*100,wc=6,dc=0,>fixed,labc=P,labr=YT)
```

	CDU/CSU	SPD	FDP	Gr	Li
1990	48	36	12	1	3
1994	44	38	7	7	4
1998	37	45	6	7	5
2002	41	42	8	9	0
2005	37	36	10	8	9
2009	38	23	15	11	12
2013	49	31	0	10	10

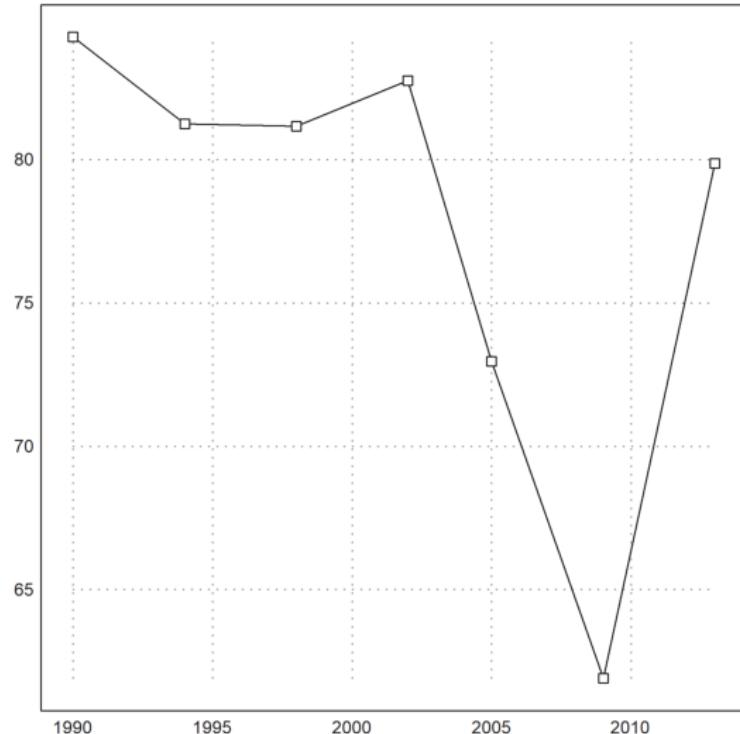
The following matrix multiplication extracts the sum of the percentages of the two big parties showing that the small parties have gained footage in the parliament until 2009.

```
>BT1:=(BT.[1;1;0;0;0])'*100
```

```
[84.29, 81.25, 81.166, 82.753, 72.964, 61.897, 79.873]
```

There is also a simple statistical plot. We use it to display lines and points simultaneously. The alternative is to call `plot2d` twice with `>add`.

```
>statplot(YT,BT1,"b"):
```

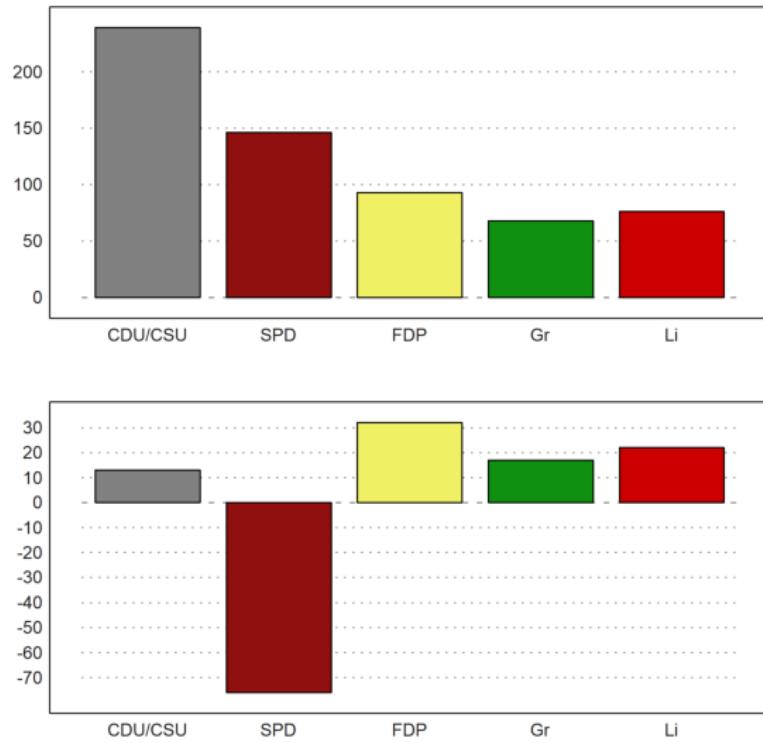


Define some colors for each party.

```
>CP:=[rgb(0.5,0.5,0.5),red,yellow,green,rgb(0.8,0,0)];
```

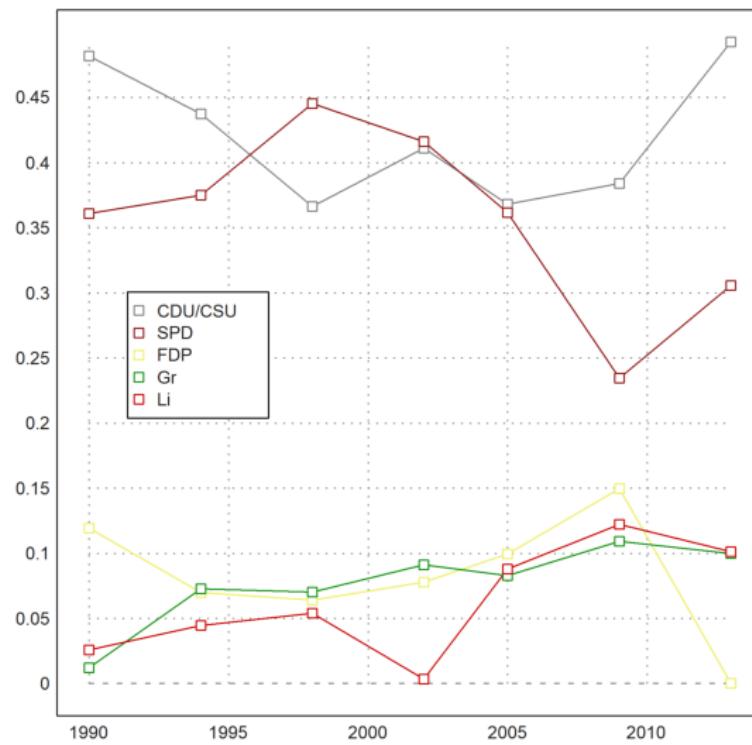
Now we can plot the results of the 2009 election and the changes into one plot using `figure`. We can add a vector of columns to each plot.

```
>figure(2,1); ...
>figure(1); columnsplot(BW[6,3:7],P,color=CP); ...
>figure(2); columnsplot(BW[6,3:7]-BW[5,3:7],P,color=CP); ...
>figure(0):
```



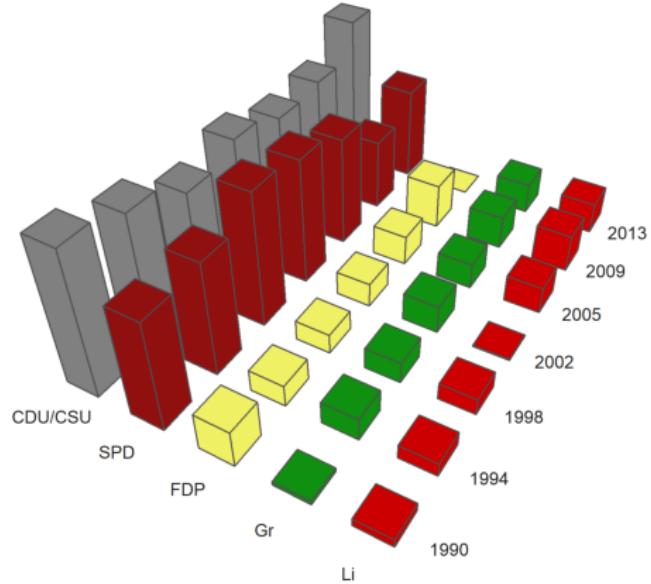
Data plots combine rows of statistical data in one plot.

```
>J:=BW[,1]'; DP:=BW[,3:7]'; ...
>dataplot(YT,BT',color=CP); ...
>labelbox(P,colors=CP,styles="[]",>points,w=0.2,x=0.3,y=0.4):
```



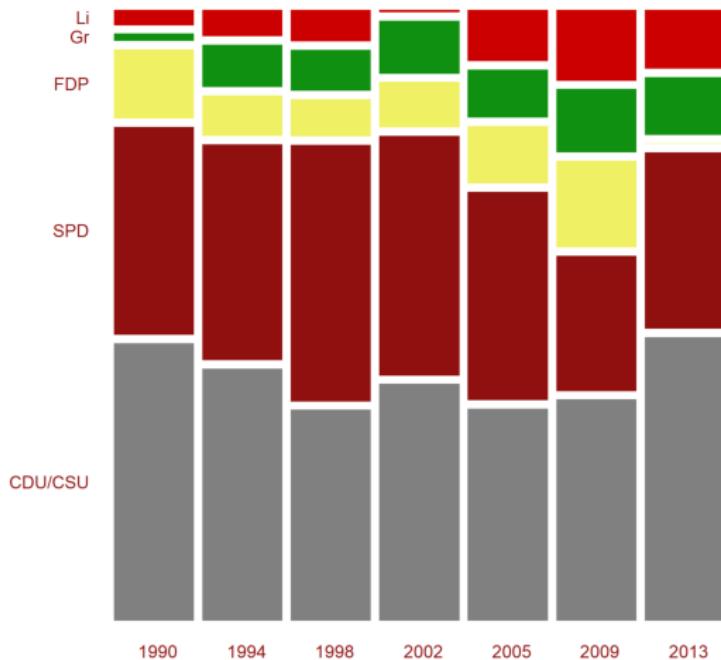
A 3D columns plot shows rows of statistical data in form of columns. We provide labels for the rows and the columns. angle is the viewing angle.

```
>columnsplot3d(BT,scols=P,srows=YT, ...
>  angle=30°,ccols=CP):
```



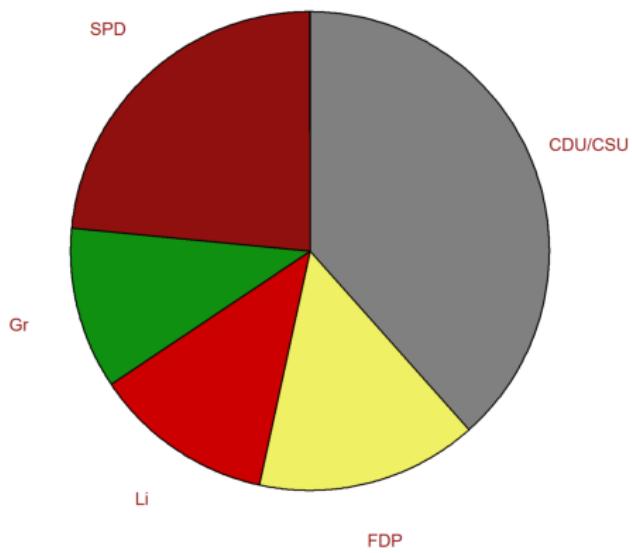
Another representation is the mosaic plot. Note that the columns of the plot represent the columns of the matrix here. Because of the length of the label CDU/CSU, we take a smaller window than usual.

```
>shrinkwindow(>smaller); ...
>mosaicplot(BT',srows=YT,scols=P,color=CP,style="#" ); ...
>shrinkwindow():
```



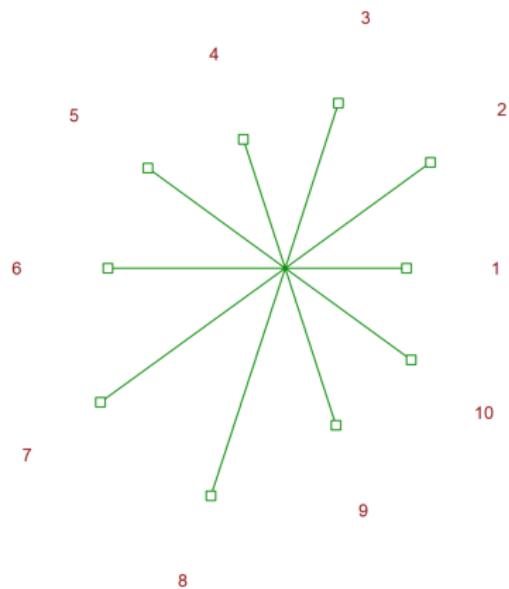
We can also do a pie chart. Since black and yellow form a coalition, we reorder the elements.

```
>i=[1,3,5,4,2]; piechart(BW[6,3:7][i],color=CP[i],lab=P[i]):
```



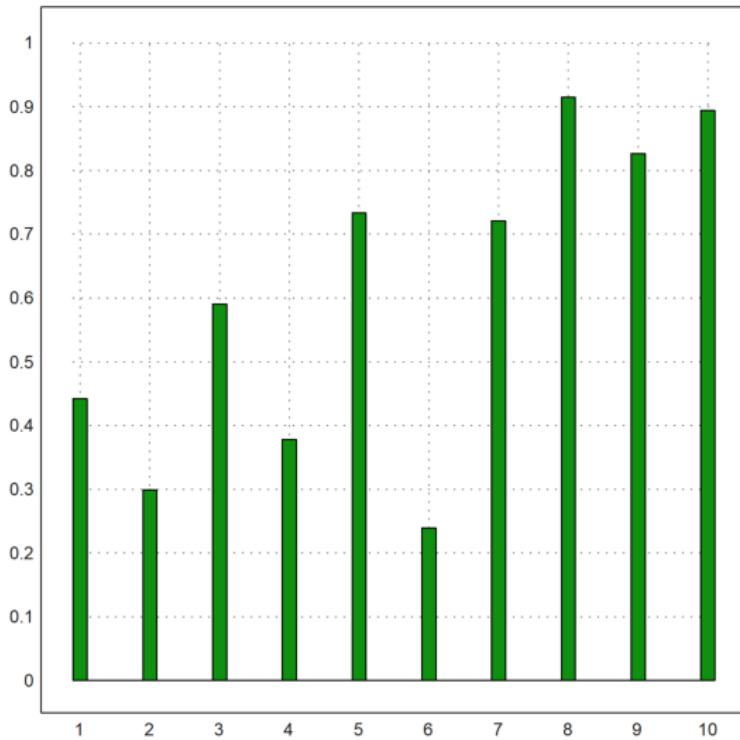
Here is another kind of plot.

```
>starplot(normal(1,10)+4,lab=1:10,>rays):
```



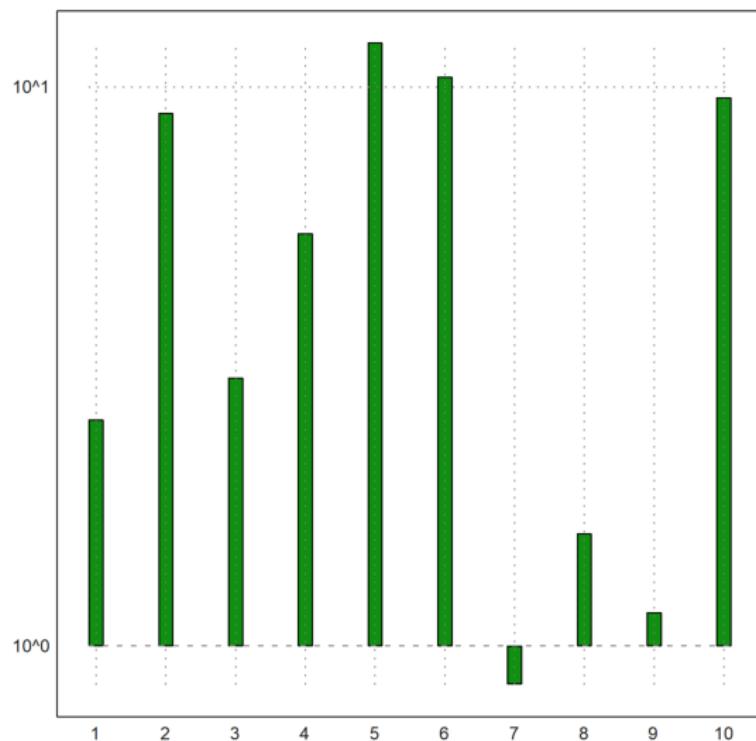
Some plots in plot2d are good for statics. Here is an impulse plot of random data, uniformly distributed in [0,1].

```
>plot2d(makeimpulse(1:10,random(1,10)),>bar):
```



But for exponentially distributed data, we may need a logarithmic plot.

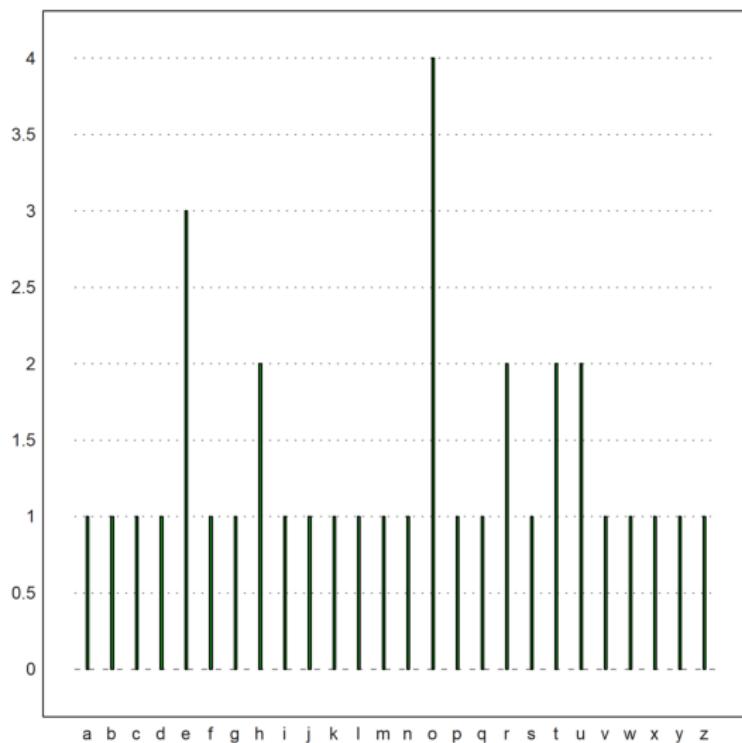
```
>logimpulseplot(1:10,-log(random(1,10))*10):
```



The function `columnsplot()` is easier to use, since it needs just a vector of values. Moreover, it can set its labels to anything we want, we demonstrated this already in this tutorial.

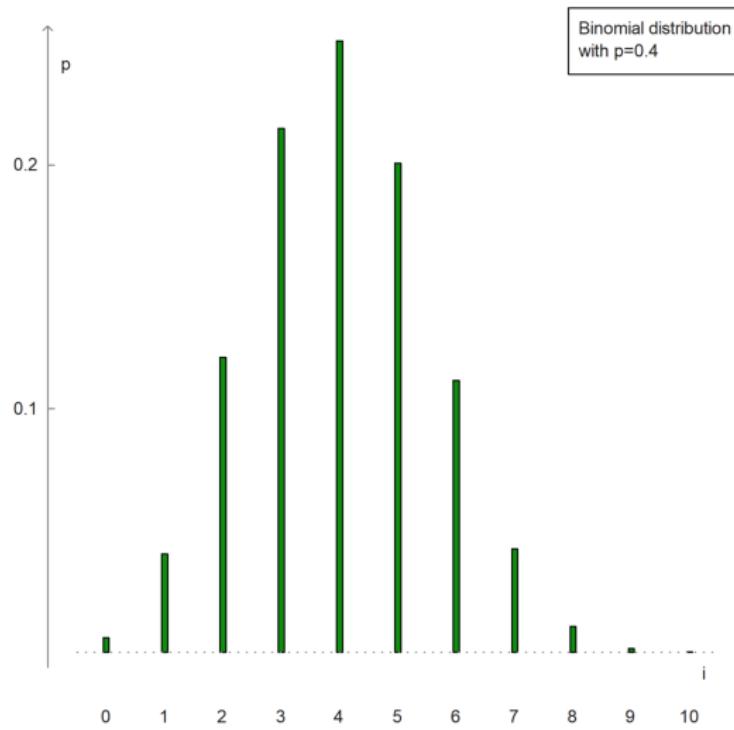
Here is another application, where we count characters in a sentence and plot a statistics.

```
>v=char("the quick brown fox jumps over the lazy dog"); ...
>w=ascii("a"):ascii("z"); x=getmultiplicities(w,v); ...
>cw=[]; for k=w; cw=cw|char(k); end; ...
>columnsplot(x,lab=cw,width=0.05):
```



It is also possible to manually set axes.

```
>n=10; p=0.4; i=0:n; x=bin(n,i)*p^i*(1-p)^(n-i); ...
>columnsplot(x,lab=i,width=0.05,<frame,<grid); ...
>yaxis(0,0:0.1:1,style="->",>left); xaxis(0,style="."); ...
>label("p",0,0.25), label("i",11,0); ...
>textbox(["Binomial distribution","with p=0.4"]):
```



The following is a way to plot the frequencies of numbers in a vector.
We create a vector of integer random numbers 1 to 6.

```
>v:=inrandom(1,10,10)
```

```
[4, 7, 5, 9, 2, 8, 5, 2, 7, 9]
```

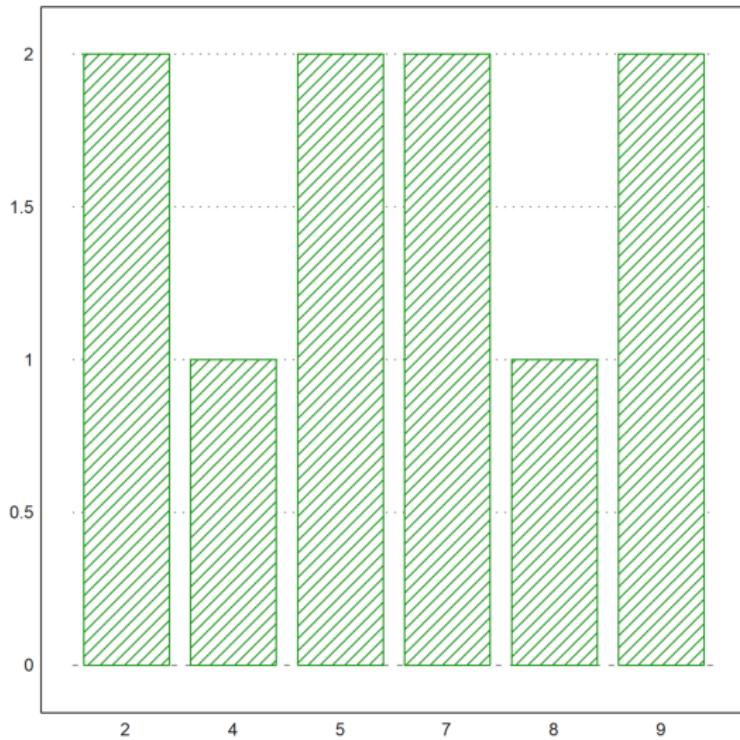
Then extract the unique numbers in v.

```
>vu:=unique(v)
```

```
[2, 4, 5, 7, 8, 9]
```

And plot the frequencies in a columns plot.

```
>columnsplot(getmultiplicities(vu,v),lab=vu,style="/"):
```



We want to demonstrate functions for the empirical distribution of values.

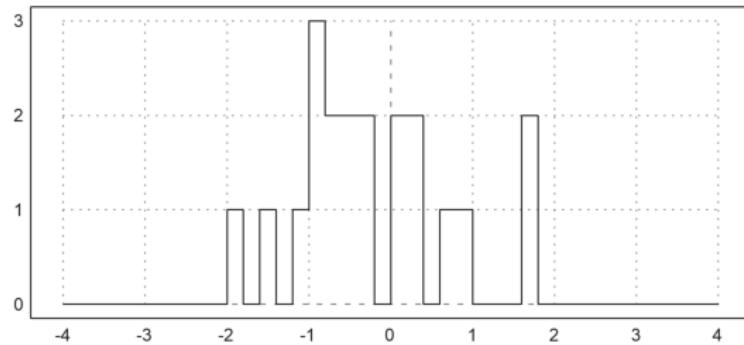
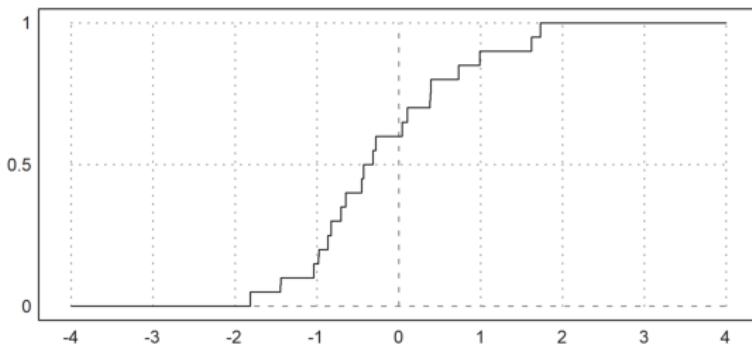
```
>x=normal(1,20);
```

The function `empdist(x,vs)` needs a sorted array of values. So we have to sort `x` before we can use it.

```
>xs=sort(x);
```

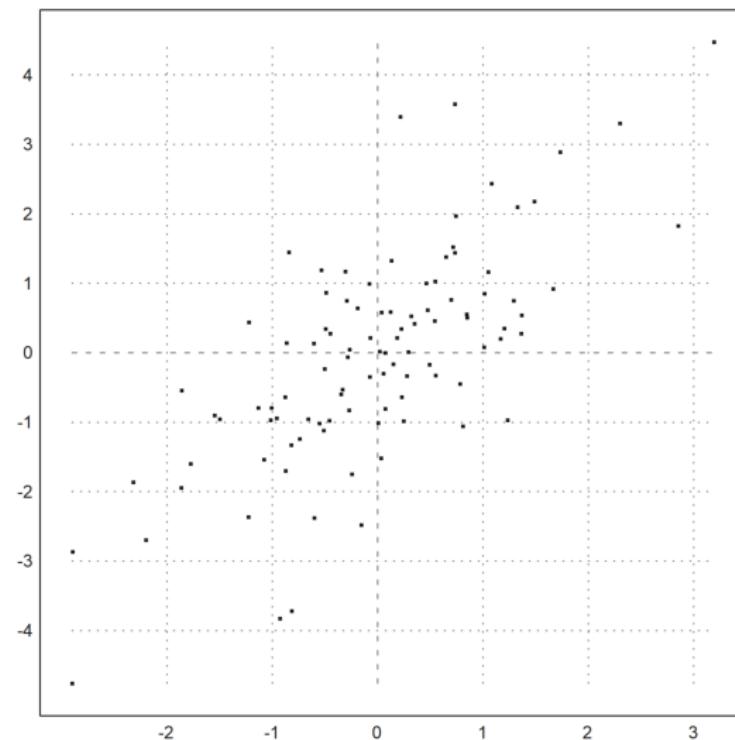
Then we plot the empirical distribution and some density bars into one plot. Instead of a bar plot for the distribution we use a sawtooth plot this time.

```
>figure(2,1); ...
>figure(1); plot2d("empdist",-4,4;xs); ...
>figure(2); plot2d(histo(x,v=-4:0.2:4,<bar)); ...
>figure(0):
```



A scatter plot is easy to do in Euler with the usual point plot. The following graph shows that the X and $X+Y$ are clearly positively correlated.

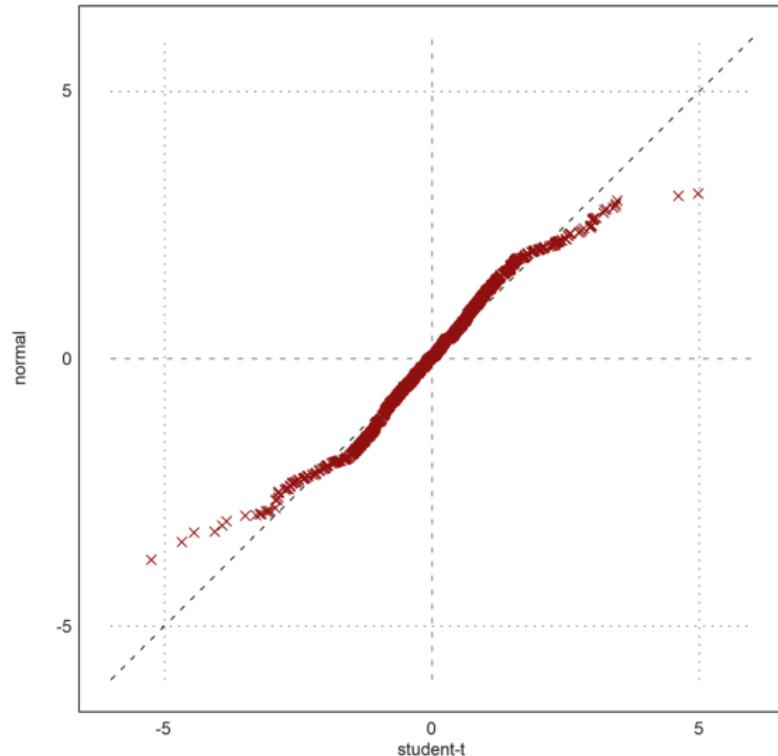
```
>x=normal(1,100); plot2d(x,x+rotright(x),>points,style="."):
```



Often, we wish to compare two samples of different distributions. This can be done with a quantile-quantile-plot.

For a test, we try the student-t distribution and exponential distribution.

```
>x=randt(1,1000,5); y=randnormal(1,1000,mean(x),dev(x)); ...
>plot2d("x",r=6,style="--",yl="normal",xl="student-t",>vertical); ...
>plot2d(sort(x),sort(y),>points,color=red,style="x",>add):
```



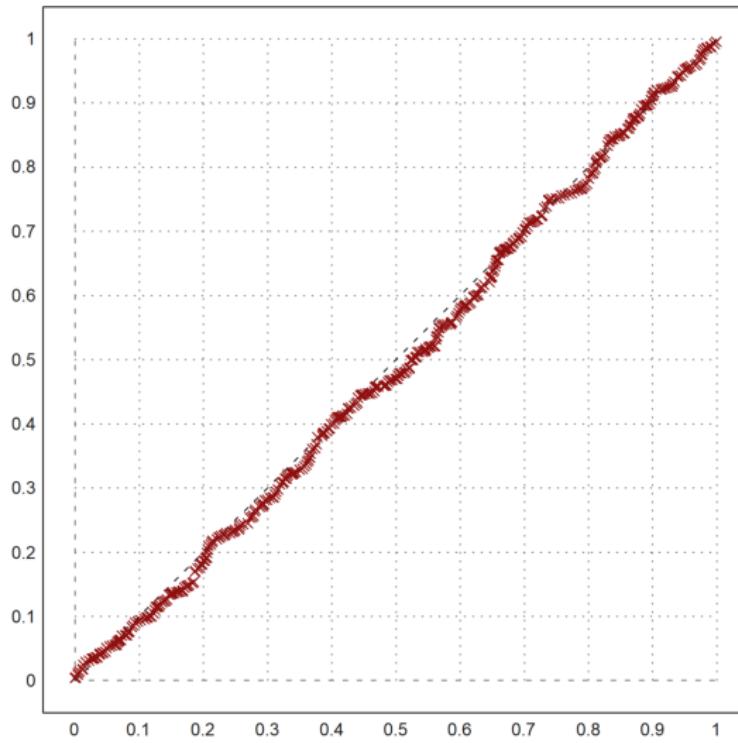
The plot clearly shows that the normal distributed values tend to be smaller at the extreme ends.

If we have two distributions of different size, we can expand the smaller one or shrink the larger one. The following function is good for both. It takes the median values with percentages between 0 and 1.

```
>function medianexpand (x,n) := median(x,p=linspace(0,1,n-1));
```

Let us compare two equal distributions.

```
>x=random(1000); y=random(400); ...
>plot2d("x",0,1,style="--"); ...
>plot2d(sort(medianexpand(x,400)),sort(y),>points,color=red,style="x",>add):
```



Regression and Correlation

Linear regression can be done with the functions `polyfit()` or various fit functions.
For a start we find the regression line for univariate data with `polyfit(x,y,1)`.

```
>x=1:10; y=[2,3,1,5,6,3,7,8,9,8]; writetable(x' | y', labc=["x", "y"])
```

x	y
1	2
2	3
3	1
4	5
5	6
6	3
7	7
8	8
9	9
10	8

We want to compare non-weighted and weighted fits. First the coefficients of the linear fit.

```
>p=polyfit(x,y,1)
```

```
[0.73333, 0.81212]
```

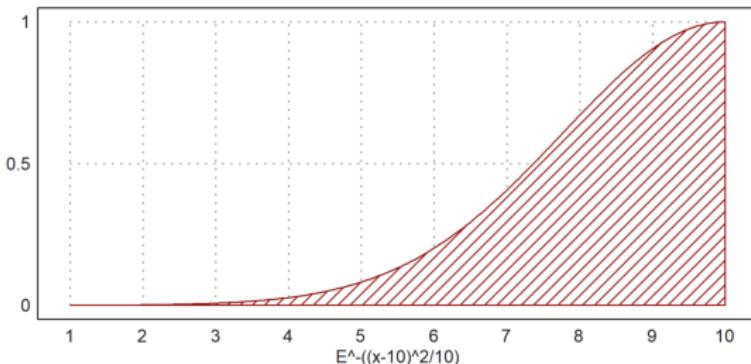
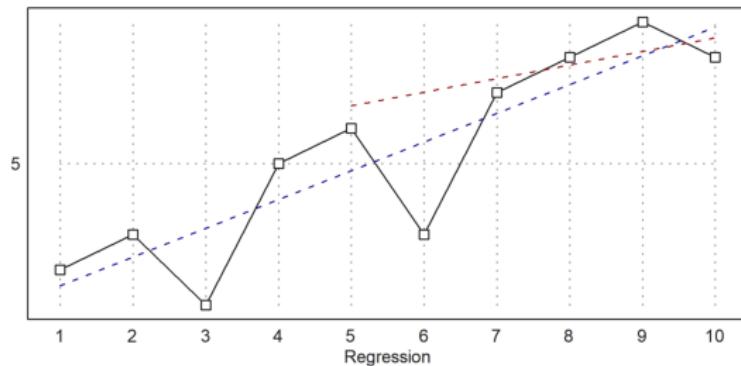
Now the coefficients with a weight that emphasizes the last values.

```
>w &= "exp(-(x-10)^2/10)"; pw=polyfit(x,y,1,w=w(x))
```

```
[4.7157, 0.38319]
```

We put everything into one plot for the points and the regression lines, and for the weights used.

```
>figure(2,1); ...
>figure(1); statplot(x,y,"b",xl="Regression"); ...
> plot2d("evalpoly(x,p)",>add,color=blue,style="--"); ...
> plot2d("evalpoly(x,pw)",5,10,>add,color=red,style="--"); ...
>figure(2); plot2d(w,1,10,>filled,style="/",fillcolor=red,xl=w); ...
>figure(0):
```



For another example we read a survey of students, their ages, the ages of their parents and the number of siblings from a file.

This table contains "m" and "f" in the second column. We use the variable tok2 to set the proper translations instead of letting readtable() collect the translations.

```
>{MS,hd}:=readtable("table1.dat",tok2:=[ "m", "f" ]); ...
>writetable(MS,labc=hd,tok2:=[ "m", "f" ]);
```

```
Could not open the file
table1.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
if filename!=none then open(filename,"r"); endif;
```

How do the ages depend on each other? A first impression comes from a pairwise scatterplot.

```
>scatterplots(tablecol(MS, 3:5), hd[3:5]):
```

```
Variable or function MS not found.  
Error in:  
scatterplots(tablecol(MS, 3:5), hd[3:5]): ...  
^
```

It is clear that the age of the father and mother depend on each other. Let us determine and plot the regression line.

```
>cs:=MS[, 4:5]'; ps:=polyfit(cs[1], cs[2], 1)
```

```
MS is not a variable!  
Error in:  
cs:=MS[, 4:5]'; ps:=polyfit(cs[1], cs[2], 1) ...  
^
```

This is obviously the wrong model. The regression line would be $s=17+0.74t$, where t is the age of the mother and s the age of the father. The age difference may depend a little bit on the age, but not that much. Rather, we suspect a function like $s=a+t$. Then a is the mean of the $s-t$. It is the average age difference between fathers and mothers.

```
>da:=mean(cs[2]-cs[1])
```

```
cs is not a variable!  
Error in:  
da:=mean(cs[2]-cs[1]) ...  
^
```

Let us plot this into one scatter plot.

```
>plot2d(cs[1], cs[2], >points); ...  
>plot2d("evalpoly(x,ps)", color=red, style=". ", >add); ...  
>plot2d("x+da", color=blue, >add):
```

```
cs is not a variable!  
Error in:  
plot2d(cs[1], cs[2], >points); plot2d("evalpoly(x,ps)", color=re ...  
^
```

Here is a box plot of the two ages. This only shows, that the ages are different.

```
>boxplot(cs, ["mothers", "fathers"]):
```

```
Variable or function cs not found.  
Error in:  
boxplot(cs, ["mothers", "fathers"]): ...  
^
```

It is interesting that the difference in medians is not as large as the difference in means.

```
>median(cs[2])-median(cs[1])
```

```
cs is not a variable!
Error in:
median(cs[2])-median(cs[1]) ...
^
```

The correlation coefficient suggests a positive correlation.

```
>correl(cs[1],cs[2])
```

```
cs is not a variable!
Error in:
correl(cs[1],cs[2]) ...
^
```

The correlation of the ranks is a measure for the same order in both vectors. It is also quite positive.

```
>rankcorrel(cs[1],cs[2])
```

```
cs is not a variable!
Error in:
rankcorrel(cs[1],cs[2]) ...
^
```

Creating new Functions

Of course, the EMT language can be used to program new functions. E.g., we define the skewness function.

$$\text{sk}(x) = \frac{\sqrt{n} \sum_i (x_i - m)^3}{(\sum_i (x_i - m)^2)^{3/2}}$$

where m is the mean of x .

```
>function skew (x:vector) ...
```

```
m=mean(x);
return sqrt(cols(x))*sum((x-m)^3)/(sum((x-m)^2))^(3/2);
endfunction
```

As you see, we can easily use the matrix language to get a very short and efficient implementation. Let us try this function.

```
>data=normal(20); skew(normal(10))
```

-0.2356

Here is another function, called the Pearson skewness coefficient.

```
>function skew1 (x) := 3*(mean(x)-median(x))/dev(x)
>skew1(data)
```

0.20287

Monte Carlo Simulation

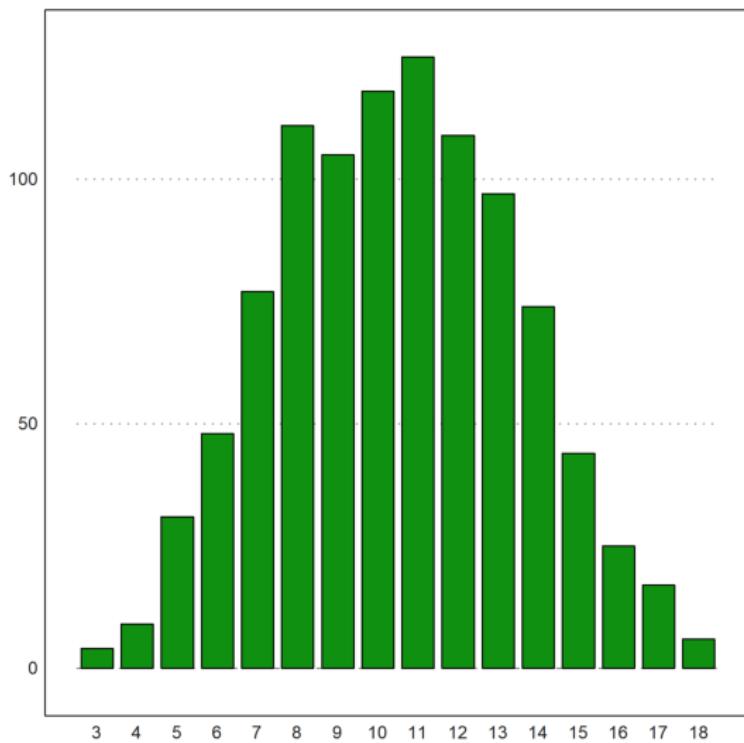
Euler can be used to simulate random events. We have already seen simple examples above. Here is another one, which simulates 1000 times 3 dice throws, and asks for the distribution of the sums.

```
>ds:=sum(intrandom(1000,3,6))'; fs=getmultiplicities(3:18,ds)
```

```
[4, 9, 31, 48, 77, 111, 105, 118, 125, 109, 97, 74, 44,
25, 17, 6]
```

We can plot this now.

```
>columnsplot(fs,lab=3:18):
```



To determine the expected distribution is not so easy. We use an advanced recursion for this.

The following function counts the number of ways the number k can be represented as the sum of n numbers in the range of 1 to m. It works recursively in an obvious way.

```
>function map countways (k; n, m) ...
```

```

if n==1 then return k>=1 && k<=m
else
  sum=0;
  loop 1 to m; sum=sum+countways(k-#,n-1,m); end;
  return sum;
end;
endfunction

```

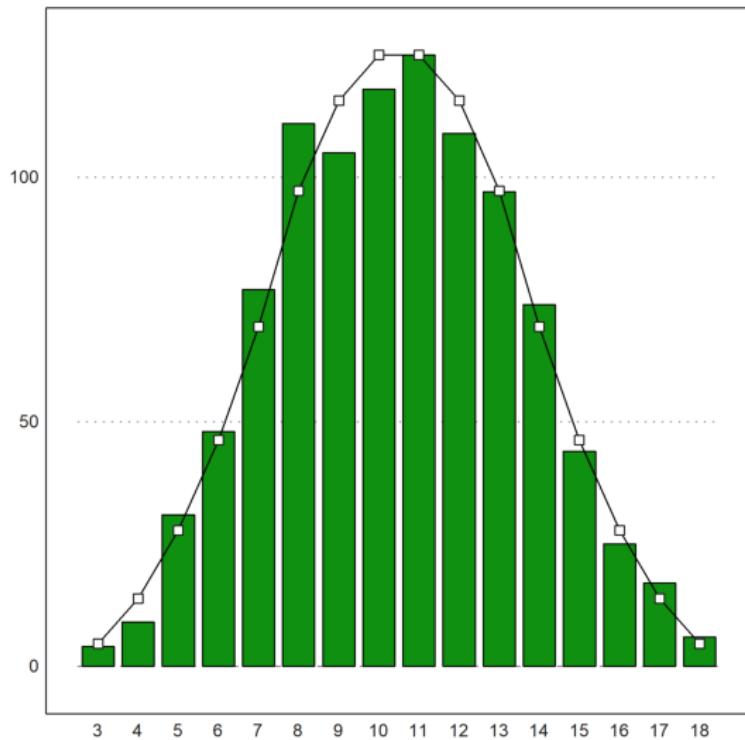
Here is the result for three throws of dices.

```
>cw=countways(3:18,3,6)
```

```
[1, 3, 6, 10, 15, 21, 25, 27, 27, 25, 21, 15, 10, 6, 3,
1]
```

We add the expected values to the plot.

```
>plot2d(cw/6^3*1000,>add); plot2d(cw/6^3*1000,>points,>add):
```



For another simulation, the deviation of the mean value of n 0-1-normal distributed random variables is $1/\sqrt{n}$.

```
>longformat; 1/sqrt(10)
```

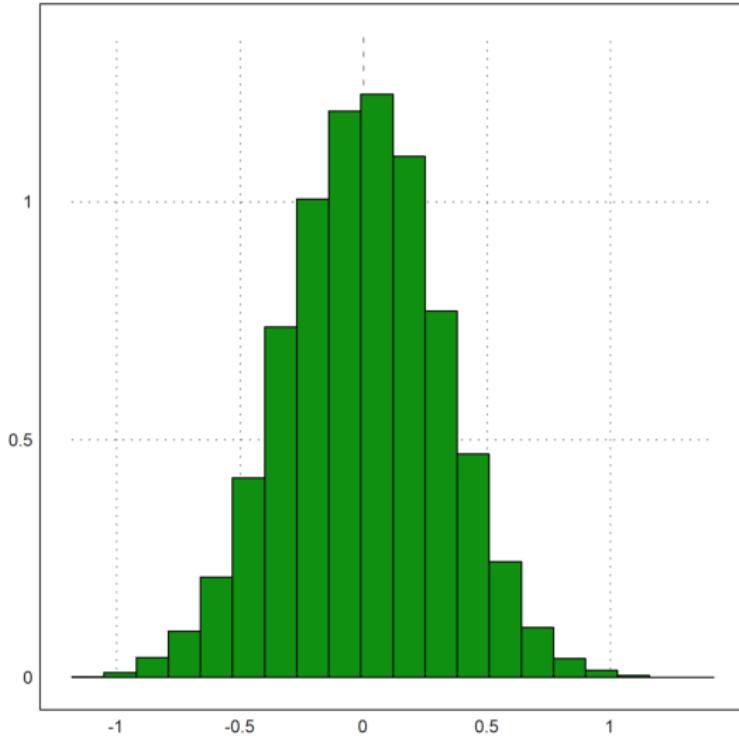
```
0.316227766017
```

Let us check this with a simulation. We produce 10000 times 10 random vectors.

```
>M=normal(10000,10); dev(mean(M)')
```

0.317437977836

```
>plot2d(mean(M)',>distribution):
```



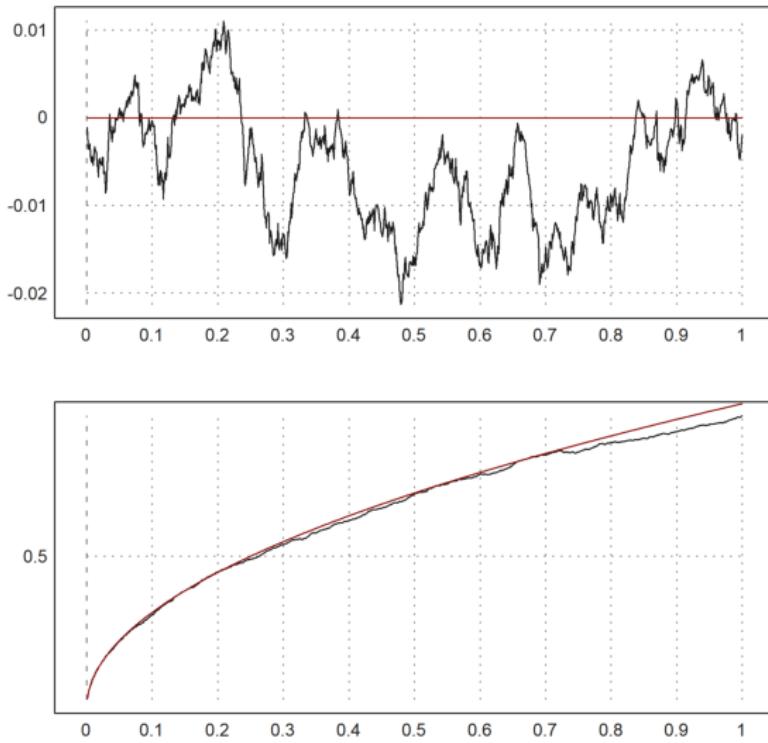
The median of 10 0-1-normal distributed random numbers has a larger deviation.

```
>dev(median(M)')
```

0.376067877174

Since we can easily generate random walks, we can simulate the Wiener process. We take 1000 steps of 1000 processes. We then plot the standard deviation and the mean of the n-th step of these processes together with the expected values in red.

```
>n=1000; m=1000; M=cumsum(normal(n,m)/sqrt(m)); ...
>t=(1:n)/n; figure(2,1); ...
>figure(1); plot2d(t,mean(M)'), plot2d(t,0,color=red,>add); ...
>figure(2); plot2d(t,dev(M)'), plot2d(t,sqrt(t),color=red,>add); ...
>figure(0):
```



Tests

Tests are an important tool in statistics. In Euler, many tests are implemented. All of these tests returns the error that we accept if we reject the zero hypothesis.

For an example, we test dice throws for uniform distribution. At 600 throws, we got the following values, which we plug into the chi-square test.

```
>chitest([90,103,114,101,103,89],dup(100,6)')
```

0.498830517952

The chi-square test also has a mode, which uses a Monte Carlo simulation to test the statistics. The result should be almost the same. The parameter `>p` interprets the `y`-vector as a vector of probabilities.

```
>chitest([90,103,114,101,103,89],dup(1/6,6)',>p,>montecarlo)
```

0.493

This error is much too large. So we cannot reject uniform distribution. This does not prove that our dice was fair. But we cannot reject our hypothesis.

Next we generate 1000 dice throws using the random number generator, and do the same test.

```
>n=1000; t=random([1,n*6]); chitest(count(t*6,6),dup(n,6)')
```

0.593661619845

Let us test for the mean value 100 with the t-test.

```
>s=200+normal([1,100])*10; ...
>ttest(mean(s),dev(s),100,200)
```

0.13110793575

The function `ttest()` needs the mean value, the deviation, the number of data, and the mean value to test for. Now let us check two measurements for the same mean. We reject the hypothesis that they have the same mean, if the result is <0.05.

```
>tcomparedata(normal(1,10),normal(1,10))
```

0.0565894669501

If we add a bias to one distribution, we get more rejections. Repeat this simulation several times to see the effect.

```
>tcomparedata(normal(1,10),normal(1,10)+2)
```

0.000992104768909

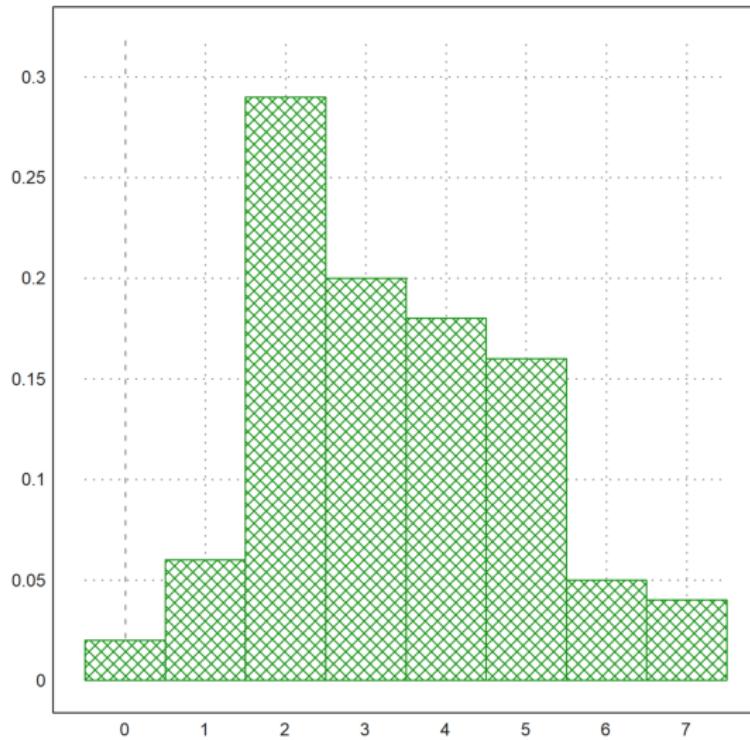
In the next example, we generate 20 random dice throws 100 times and count the ones in it. There must be $20/6=3.3$ ones on average.

```
>R=random(100,20); R=sum(R*6<=1); mean(R)
```

3.34

We now compare the number of ones with the binomial distribution. First we plot the distribution of ones.

```
>plot2d(R,distribution=max(R)+1,even=1,style="\\"/"):
```



```
>t=count(R,21);
```

Then we compute the expected values.

```
>n=0:20; b=bin(20,n)*(1/6)^n*(5/6)^(20-n)*100;
```

We have to collect several numbers to get categories, which are big enough.

```
>t1=sum(t[1:2])|t[3:7]|sum(t[8:21]); ...
>b1=sum(b[1:2])|b[3:7]|sum(b[8:21]);
```

The chi-square test rejects the hypothesis that our distribution is a binomial distribution, if its result is <0.05.

```
>chitest(t1,b1)
```

0.229210433366

The following example contains results of two groups of persons (male and female, say) voting for one out of six parties.

```
>A=[23,37,43,52,64,74;27,39,41,49,63,76]; ...
> writetable(A,wc=6,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	23	37	43	52	64	74
f	27	39	41	49	63	76

We wish to test for independence of the votes from the sex. The chi² table test does this. The result is way to large to reject independence. So we cannot say, if the voting depends on the sex from these data.

```
>tabletest(A)
```

0.990701632326

The following is the expected table, if we assume the observed frequencies of voting.

```
>writetable(expectedtable(A), wc=6, dc=1, labr=c("m", "f"), labc=1:6)
```

	1	2	3	4	5	6
m	24.9	37.9	41.9	50.3	63.3	74.7
f	25.1	38.1	42.1	50.7	63.7	75.3

We can compute the corrected contingency coefficient. Since it very close to 0, we conclude that the voting does not depend on the sex.

```
>contingency(A)
```

0.0427225484717

Some More Tests

Next we use a variance analysis (F-test) to test three samples of normally distributed data for same mean value. The method is called ANOVA (analysis of variance). In Euler, the function varanalysis() is used.

```
>x1=[109,111,98,119,91,118,109,99,115,109,94]; mean(x1),
```

106.545454545

```
>x2=[120,124,115,139,114,110,113,120,117]; mean(x2),
```

119.111111111

```
>x3=[120,112,115,110,105,134,105,130,121,111]; mean(x3)
```

116.3

```
>varanalysis(x1,x2,x3)
```

0.0138048221371

This means, we reject the hypothesis of same mean value. We do this with an error probability of 1.3%. There is also the median test, which rejects data samples with different mean distribution testing the median of the united sample.

```
>a=[56, 66, 68, 49, 61, 53, 45, 58, 54];  
>b=[72, 81, 51, 73, 69, 78, 59, 67, 65, 71, 68, 71];  
>mediantest(a,b)
```

0.0241724220052

Another test on equality is the rank test. It is much sharper than the median test.

```
>ranktest(a,b)
```

0.00199969612469

In the following example, both distributions have the same mean.

```
>ranktest(random(1,100),random(1,50)*3-1)
```

0.236498298684

Let us now try to simulate two treatments a and b applied to different persons.

```
>a=[8.0, 7.4, 5.9, 9.4, 8.6, 8.2, 7.6, 8.1, 6.2, 8.9];  
>b=[6.8, 7.1, 6.8, 8.3, 7.9, 7.2, 7.4, 6.8, 6.8, 8.1];
```

The signum test decides, if a is better than b.

```
>signtest(a,b)
```

0.0546875

This is too much of an error. We cannot reject that a is as good as b.

The Wilcoxon test is sharper than this test, but relies on the quantitative value of the differences.

```
>wilcoxon(a,b)
```

0.0296680599405

Let us try two more tests using generated series.

```
>wilcoxon(normal(1,20),normal(1,20)-1)
```

0.00085650881911

```
>wilcoxon(normal(1,20),normal(1,20))
```

0.921998212017

Random Numbers

The following is a test for the random number generator. Euler uses a very good generator, so we need not expect any problems.

First we generate ten millions of random numbers in [0,1].

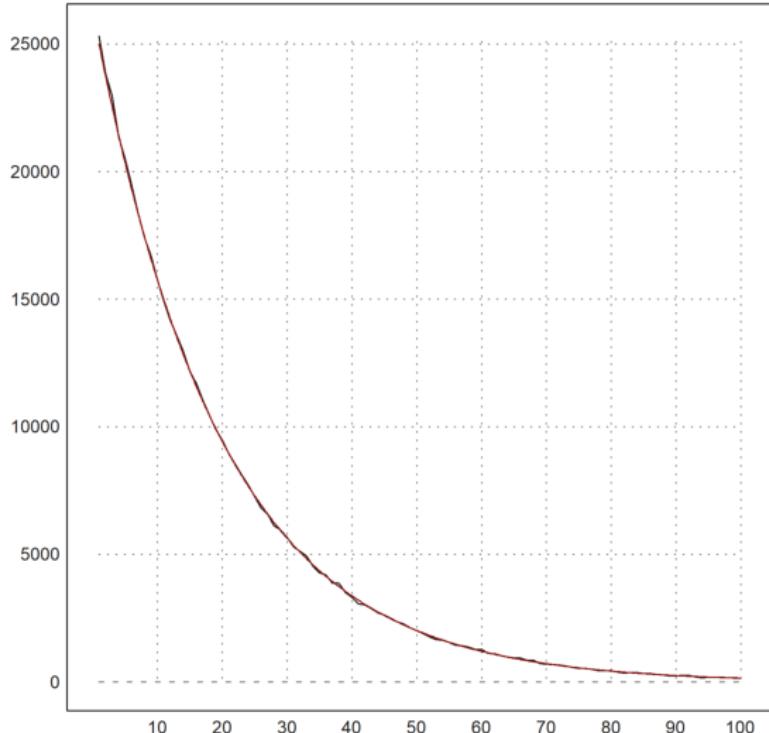
```
>n:=10000000; r:=random(1,n);
```

Next we count the distances between two numbers less than 0.05.

```
>a:=0.05; d:=differences(nonzeros(r<a));
```

Finally, we plot the number of times, each distance occurred, and compare with the expected value.

```
>m=getmultiplicities(1:100,d); plot2d(m); ...
> plot2d("n*(1-a)^(x-1)*a^2",color=red,>add):
```



Clear the data.

```
>remvalue n;
```

Introduction for Users of the R Project

Clearly, EMT is not competing with R as a statistical package. However, there are many statistical procedures and functions available in EMT too. So EMT may satisfy the basic needs. After all, EMT comes with numerical packages and a computer algebra system.

This notebook is for you if you are familiar with R, but need to know the differences of the syntax of EMT and R. We try to give an overview of obvious and less obvious things you need to know.

Moreover, we look at ways to exchange data between the two systems.

Note that this is a work in progress. **Basic Syntax**

The first thing you learn in R is to make a vector. In EMT, the main difference is that the `:` operator can take a step size. Moreover it has a low binding power.

```
>n=10; 0:n/20:n-1
```

```
[0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5,  
7, 7.5, 8, 8.5, 9]
```

The `c()` function does not exist. It is possible to use vectors to concatenate things.

The following example is, like many others, from the "Introduction to R" that comes with the R project. If you read this PDF, you will find that I follow its path in this tutorial.

```
>x=[10.4, 5.6, 3.1, 6.4, 21.7]; [x,0,x]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 0, 10.4, 5.6, 3.1, 6.4, 21.7]
```

The colon operator with step size of EMT is replaced by the function `seq()` in R. We can write this function in EMT.

```
>function seq(a,b,c) := a:b:c; ...  
>seq(0,-0.1,-1)
```

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

The function `rep()` of R is not present in EMT. For vector input, it could be written as follows.

```
>function rep(x:vector,n:index) := flatten(dup(x,n)); ...  
>rep(x,2)
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Note that `"="` or `"":=` is used for assignments. The `"->"` operator is used for units in EMT.

```
>125km -> " miles"
```

```
77.6713990297 miles
```

The `"<"` operator for assignment is misleading anyway, and not a good idea of R. The following will compare `a` and `-4` in EMT.

```
>a=2; a<-4
```

```
0
```

In R, `"a<-4<3"` works, but `"a<-4<-3"` does not. I had similar ambiguities in EMT too, but tried to eliminate them by and by.

EMT and R have vectors of boolean type. But in EMT, the numbers 0 and 1 are used to represent false and true. In R, the values true and false can nevertheless used in ordinary arithmetic just like in EMT.

```
>x<5, %*x
```

```
[0, 0, 1, 0, 0]
[0, 0, 3.1, 0, 0]
```

EMT throws errors or yields NAN depending on the flag "errors".

```
>errors off; 0/0, isNaN(sqrt(-1)), errors on;
```

```
NAN
1
```

Strings are the same in R and EMT. Both are in the current locale, not in Unicode.

In R there are packages for Unicode. In EMT, a string can be Unicode string. A unicode string can be translated to the local encoding and vice versa. Moreover, u"..." can contain HTML entities.

```
>u"     ; Ren   Grothmann"
```

  Ren  Grothmann

The following may or may not display correctly on your system as A with dot and dash above it. It depends on the font you are using.

```
>chartoutf([480])
```

The string concatenation is done with "+" or "|". It can include numbers, which will print in the current format.

```
>"pi = "+pi
```

```
pi = 3.14159265359
```

Indexing

Most of the time, this will work as in R.

But EMT will interpret negative indices from the back of the vector, while R interprets x[n] as x without the n-th elements.

```
>x, x[1:3], x[-2]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]
[10.4, 5.6, 3.1]
6.4
```

The behavior of R can be achieved in EMT with drop().

```
>drop(x, 2)
```

```
[10.4, 3.1, 6.4, 21.7]
```

Logical vectors are not treated differently as index in EMT, in contrast to R. You need to extract the non-zero elements first in EMT.

```
>x, x>5, x[nonzeros(x>5)]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]  
[1, 1, 0, 1, 1]  
[10.4, 5.6, 6.4, 21.7]
```

Just as in R, the index vector can contain repetitions.

```
>x[[1,2,2,1]]
```

```
[10.4, 5.6, 5.6, 10.4]
```

But names for indices are not possible in EMT. For a statistical package, this may often be necessary to ease access to elements of vectors.

To mimic this behavior, we can define a function as the following.

```
>function sel (v,i,s) := v[indexof(s,i)]; ...  
>s=["first","second","third","fourth"]; sel(x,[ "first","third"],s)
```

```
Trying to overwrite protected function sel!  
Error in:  
function sel (v,i,s) := v[indexof(s,i)]; ... ...  
^
```

```
Trying to overwrite protected function sel!  
Error in:  
function sel (v,i,s) := v[indexof(s,i)]; ... ...  
^  
[10.4, 3.1]
```

Data Types

EMT has more fixed data types than R. Obviously, in R there exist growing vectors. You can set an empty numerical vector v and assign a value to the element v[17]. This is not possible in EMT.

The following is a bit inefficient.

```
>v=[]; for i=1 to 10000; v=v|i; end;
```

EMT will now construct a vector with v and i appended on the stack and copy that vector back to the global variable v.

The more efficient pre-defines the vector.

```
>v=zeros(10000); for i=1 to 10000; v[i]=i; end;
```

To change date types in EMT, you can use functions like complex().

```
>complex(1:4)
```

```
[ 1+0i ,  2+0i ,  3+0i ,  4+0i ]
```

Conversions to strings is possible for elementary data types only. The current format is used for simple string concatenation. But there are functions like print() or frac().

For vectors, you can easily write your own function.

```
>function tostr (v) ...
```

```
s="[";  
loop 1 to length(v);  
  s=s+print(v[#,2,0);  
  if #<length(v) then s=s+","; endif;  
end;  
return s+"]";  
endfunction
```

```
>tostr(linspace(0,1,10))
```

```
[0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90,1.00]
```

For communication with Maxima, there exists a function convertm xm(), which can also be used to format a vector for output.

```
>convertm xm(1:10)
```

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

For Latex the tex command can be used to obtain the Latex command.

```
>tex(&[1,2,3])
```

```
\left[ 1 , 2 , 3 \right]
```

Factors and Tables

In the introduction to R there is an example with so called factors.

The following is a list of the territories of 30 states.

```
>austates = ["tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", ...  
>"qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas", ...  
>"sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa", ...  
>"sa", "act", "nsw", "vic", "vic", "act"];
```

Assume, we have corresponding incomes in each state.

```
>incomes = [60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, ...
>61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46, ...
>59, 46, 58, 43];
```

Now, we want to compute the mean of incomes in the territories. Being a statistical program, R has factor() and tapply() for this.

EMT can make this by finding the index of territories in the unique list of territories.

```
>auterr=sort(unique(austates)); f=indexofsor...)
```

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

At that point, we can write our own loop function to do things for one factor only.

Or we can mimic the tapply() function in the following way.

```
>function map_tappl (i; f$call, cat, x) ...
```

```
u=sort(unique(cat));
f=indexof(u,cat);
return f$(x[nonzeros(f==indexof(u,i))]);
endfunction
```

It is a bit inefficient, since it computes The unique territories for each i, but it works.

```
>tappl(auterr,"mean",austates,incomes)
```

```
[44.5, 57.3333333333, 55.5, 53.6, 55, 60.5, 56, 52.25]
```

Note that it works for each vector of territories.

```
>tappl(["act","nsw"],"mean",austates,incomes)
```

```
[44.5, 57.3333333333]
```

Now, the statistical package of EMT defines tables just as in R. The functions readtable() and writetable() can be used for input and output.

So we can print the average state income in the territories in a friendly way.

```
>writetable(tappl(auterr,"mean",austates,incomes),labc=auterr,wc=7)
```

act	nsw	nt	qld	sa	tas	vic	wa
44.5	57.33	55.5	53.6	55	60.5	56	52.25

We can also try to mimic the behavior of R completely.

The factors should clearly be kept in a collection with the types and the categories (states and territories in our example). For EMT, we add the pre-computed indices.

```
>function makef (t) ...
## Factor data
## Returns a collection with data t, unique data, indices.
## See: tapply
u=sort(unique(t));
return {{t,u,indexofsorted(u,t)}};
endfunction
```

```
>statef=makef(austates);
```

Now the third element of the collection will contain the indices.

```
>statef[3]
```

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

Now we can mimic tapply() in the following way. It will return a table as a collection of table data and column headings.

```
>function tapply (t:vector,tf,f$:call) ...
```

```
## Makes a table of data and factors
## tf : output of makef()
## See: makef
uf=tf[2]; f=tf[3]; x=zeros(length(uf));
for i=1 to length(uf);
    ind=nonzeros(f==i);
    if length(ind)==0 then x[i]=NAN;
    else x[i]=f$(t[ind]);
    endif;
end;
return {{x,uf}};
endfunction
```

We did not add much type checking here. The only precaution concerns categories (factors) with no data. But one should check for the correct length of t and for the correctness of the collection tf.

This table can be printed as a table with writetable().

```
>writetable(tapply(incomes,statef,"mean"),wc=7)
```

	act	nsw	nt	qld	sa	tas	vic	wa
	44.5	57.33	55.5	53.6	55	60.5	56	52.25

Arrays

EMT has only two dimensions for arrays. The data type is called a matrix. It would be easy to write functions for higher dimensions or a C library for this, however.

R has more than two dimensions. In R the array is a vector with a dimension field.

In EMT, a vector is a matrix with one row. It can be made into a matrix with `redim()`.

```
>shortformat; X=redim(1:20, 4, 5)
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Extraction of rows and columns, or sub-matrices, is much like in R.

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

2	3
7	8
12	13
17	18

However, in R it is possible to set a list of specific indices of the vector to a value. The same is possible in EMT only with a loop.

```
>function setmatrixvalue (M, i, j, v) ...  
  
loop 1 to max(length(i),length(j),length(v))  
    M[i#{},j#{}] = v#;  
end;  
endfunction
```

We demonstrate this to show that matrices are passed by reference in EMT. If you do not want to change the original matrix M, you need to copy it in the function.

```
>setmatrixvalue(X, 1:3, 3:-1:1, 0); X,
```

1	2	0	4	5
6	0	8	9	10
0	12	13	14	15
16	17	18	19	20

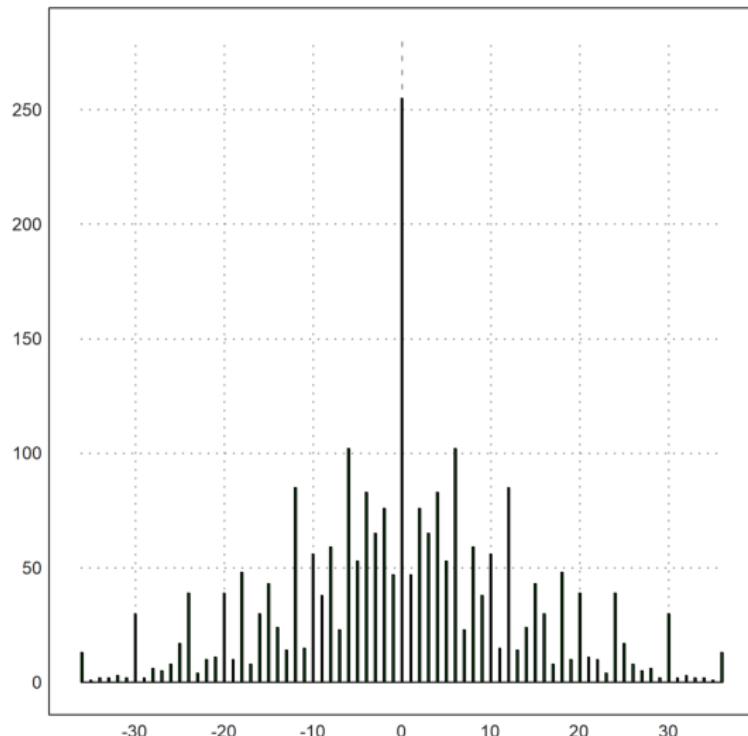
The outer product in EMT can only be done between vectors. It is automatic due to the matrix language. One vector needs to be a column vector and the other a row vector.

```
>(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

In the introduction PDF for R there is an example, which computes the distribution of $ab \cdot cd$ for a,b,c,d chosen from 0 to n randomly. The solution in R is form a 4-dimensional matrix and run table() over it. Of course, this can be achieved with a loop. But loops are not effective in EMT or R. In EMT, we could write the loop in C and that would be the quickest solution. But we want to mimic the behavior of R. For this, we need to flatten the multiplications ab and make a matrix of $ab \cdot cd$.

```
>a=0:6; b=a'; p=flatten(a*b); q=flatten(p-p'); ...
>u=sort(unique(q)); f=getmultiplicities(u,q); ...
>statplot(u,f,"h"):
```



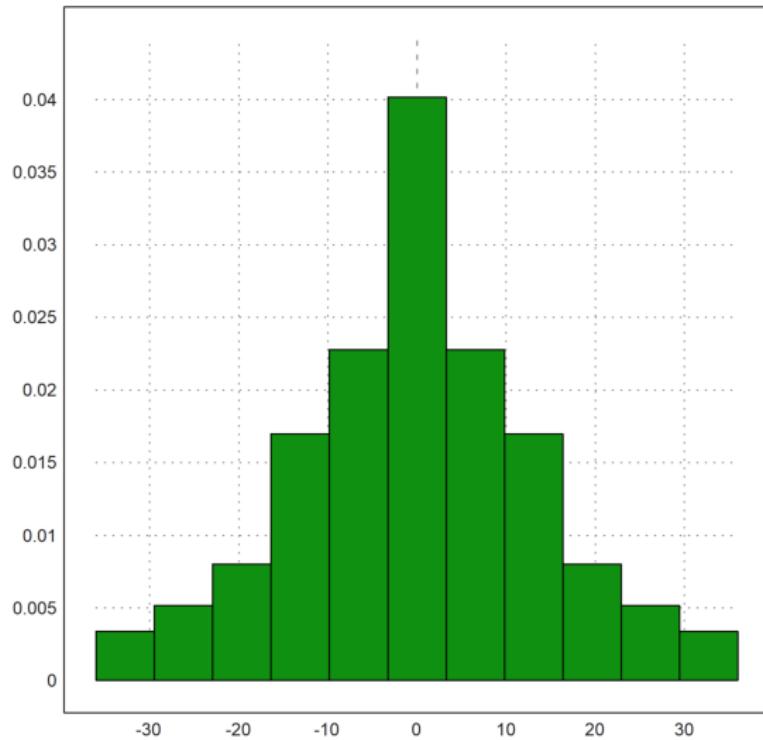
Besides the exact multiplicities, EMT can compute frequencies in vectors.

```
>getfrequencies(q,-50:10:50)
```

```
[0, 23, 132, 316, 602, 801, 333, 141, 53, 0]
```

The most easy way to plot this as a distribution is the following.

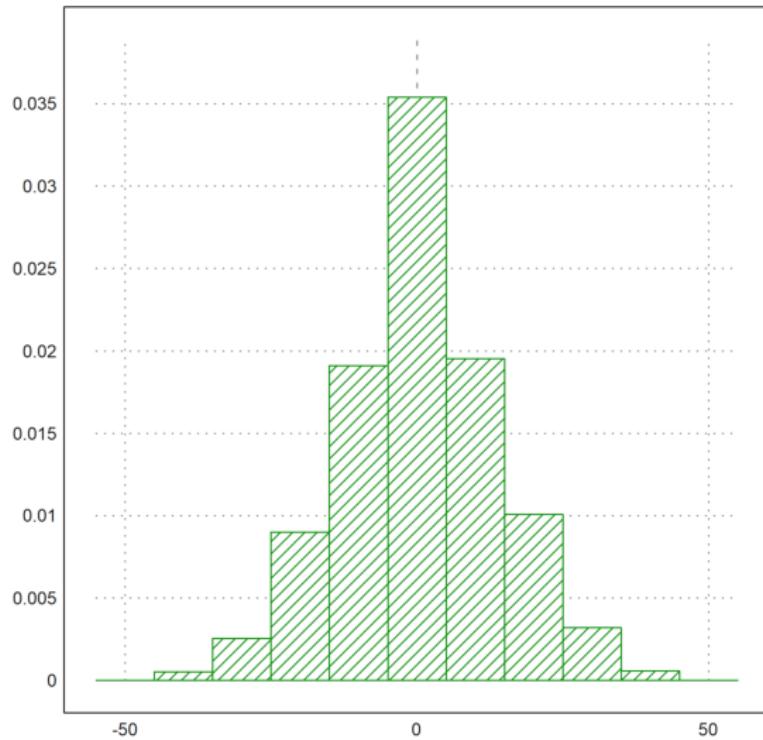
```
>plot2d(q,distribution=11):
```



But it is also possible to pre-compute the count in chosen intervals beforehand. Of course, the following uses getfrequencies() internally.

Since the histo() function returns frequencies, we need to scale these so that the integral under the bar graph is 1.

```
>{x,y}=histo(q,v=-55:10:55); y=y/sum(y)/differences(x); ...
>plot2d(x,y,>bar,style="/"):
```



Lists

EMT has two sorts of lists. One is a global list which is mutable, and the other is a list type which is immutable. We do not care about global lists here.

The immutable list type is called a collection in EMT. It behaves like a structure in C, but the elements are just numbered and not named.

```
>L={ {"Fred","Flintstone",40,[1990,1992]} }
```

```
Fred
Flintstone
40
[1990, 1992]
```

Currently the elements do not have names, though names can be set for special purposes. They are accessed by numbers.

```
> (L[4])[2]
```

```
1992
```

File Input and Output (Reading and Writing Data)

You will often want to import a matrix of data from other sources to EMT. This tutorial tells you about the many ways to achieve this. Simple functions are `writematrix()` and `readmatrix()`.

Let us demonstrate how to read and write a vector of reals to a file.

```
>a=random(1,100); mean(a), dev(a),
```

```
0.50599  
0.28401
```

To write the data to a file, we use the function `writematrix()`.

Since this introduction is most likely in a directory, where the user has no write access, we write the data to the user home directory. For own notebooks, this is not necessary, since the data file will be written into the same directory.

```
>filename="test.dat";
```

Now we write the column vector `a'` to the file. This yields one number in each line of the file.

```
>writematrix(a',filename);
```

To read the data, we use `readmatrix()`.

```
>a=readmatrix(filename)';
```

And remove the file.

```
>fileremove(filename);  
>mean(a), dev(a),
```

```
0.50599  
0.28401
```

The functions `writematrix()` or `writetable()` can be configured for other languages.

E.g., if you have an Indonesian system (decimal point with comma), your Excel needs values with decimal commas separated by semicolons in a csv (the default is comma separated values) file. The following file "test.csv" should appear on your current folder.

```
>filename="test.csv"; ...  
>writematrix(random(5,3),file=filename,separator=",");
```

You can now open this file with an Indonesian Excel directly.

```
>fileremove(filename);
```

Sometimes we have strings with tokens like the following.

```
>s1:="f m m f m m m f f f m m f"; ...  
>s2:="f f f m m f f";
```

To tokenize this, we define a vector of tokens.

```
>tok:=[ "f", "m" ]
```

```
f  
m
```

Then we can count the number of times each token appears in the string, and put the result into a table.

```
>M:=getmultiplicities(tok,strtokens(s1))_ ...  
> getmultiplicities(tok,strtokens(s2));
```

Write the table with the token headers.

```
>writetable(M,labc=tok,labr=1:2,wc=8)
```

	f	m
1	6	7
2	5	2

For statics, EMT can read and write tables.

```
>file="test.dat"; open(file,"w"); ...  
>writeln("A,B,C"); writematrix(random(3,3)); ...  
>close();
```

The file looks like this.

```
>printfile(file)
```

```
A,B,C  
0.5256028071400889,0.2243779125027929,0.3198022531267216  
0.4113190914327851,0.8794498442454435,0.2560562760405166  
0.3833353247416684,0.3362650687724961,0.7708751435970828
```

The function `readtable()` in its simplest form can read this and return a collection of values and heading lines.

```
>L=readtable(file,>list);
```

This collection can be printed with `writetable()` to the notebook, or to a file.

```
>writetable(L,wc=10,dc=5)
```

A	B	C
0.5256	0.22438	0.3198
0.41132	0.87945	0.25606
0.38334	0.33627	0.77088

The matrix of values is the first element of L. Note that `mean()` in EMT computes the mean values of the rows of a matrix.

```
>mean(L[1])
```

```
0.35659  
0.51561  
0.49683
```

CSV Files

First, let us write a matrix into a file. For the output, we generate a file in the current working directory.

```
>file="test.csv"; ...  
>M=random(3,3); writematrix(M,file);
```

Here is the content of this file.

```
>printfile(file)
```

```
0.07166984867860429,0.5675975444008555,0.7301219585632749  
0.8550456332410121,0.3255039219129054,0.4493189687803911  
0.1545075453506509,0.9822732980797209,0.7767471829538748
```

This CVS can be opened on English systems into Excel by a double click. If you get such a file on a German system, you need to import the data into Excel taking care of the decimal dot.

But the decimal dot is the default format for EMT too. You can read a matrix from a file with readmatrix().

```
>readmatrix(file)
```

```
0.07167      0      0.5676      0      0.73012  
0.85505      0      0.3255      0      0.44932  
0.15451      0      0.98227     0      0.77675
```

It is possible to write several matrices to one file. The open() command can open a file for writing with the "w" parameter. The default is "r" for reading.

```
>open(file,"w"); writematrix(M); writematrix(M'); close();
```

The matrices are separated by a blank line. To read the matrices, open the file and call readmatrix() several times.

```
>open(file); A=readmatrix(); B=readmatrix(); A==B, close();
```

```
1      1      0      1      0  
0      1      1      1      0  
0      1      0      1      1
```

In Excel or similar spreadsheets, you can export a matrix as CSV (comma separated values). In Excel 2007, use "save as" and "other formats", then select "CSV". Make sure, the current table contains only data you wish to export.

Here is an example.

```
>printfile("excel-data.csv")
```

```
Could not open the file  
excel-data.csv  
for reading!  
Try "trace errors" to inspect local variables after errors.  
printfile:  
    open(filename, "r");
```

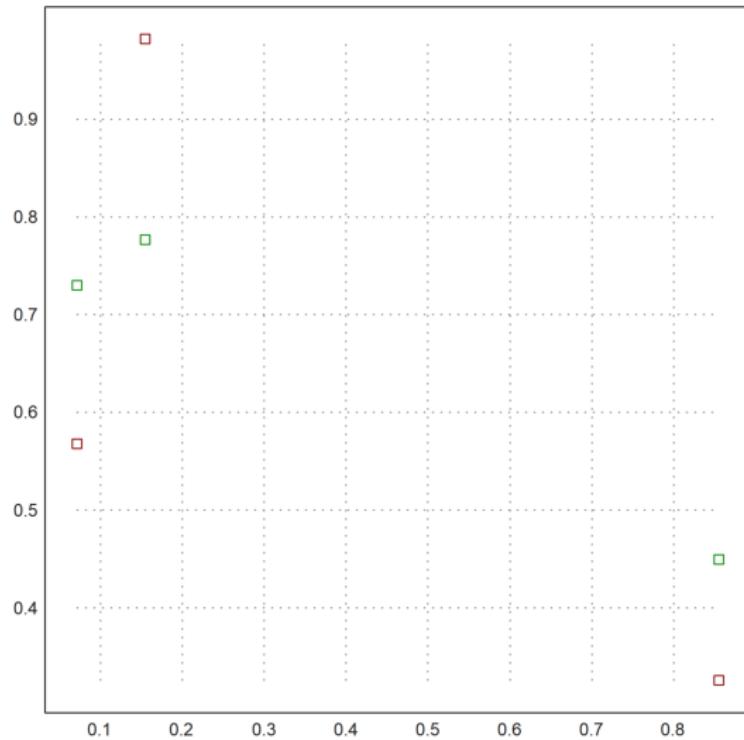
As you can see, my German system has used a semicolon as separator and a decimal comma. You can change this in the system settings or in Excel, but it is not necessary for reading the matrix into EMT. The easiest way to read this into Euler is readmatrix(). All commas are replaced by dots with the parameter >comma. For English CSV, simply omit this parameter.

```
>M=readmatrix("excel-data.csv",>comma)
```

```
Could not open the file  
excel-data.csv  
for reading!  
Try "trace errors" to inspect local variables after errors.  
readmatrix:  
    if filename<>"" then open(filename, "r"); endif;
```

Let us plot this.

```
>plot2d(M' [1],M' [2:3],>points,color=[red,green]'):
```



There are more elementary ways to read data from a file. You can open the file and read the numbers line by line. The function `getvectorline()` will read numbers from a line of data. By default, it expects a decimal dot. But it can also use a decimal comma, if you call `setdecimaldot(",")` before you use this function.

The following function is an example for this. It will stop at the end of the file or an empty line.

```
>function myload (file) ...
```

```
open(file);
M=[];
repeat
  until eof();
  v=getvectorline(3);
  if length(v)>0 then M=M_v; else break; endif;
end;
return M;
close(file);
endfunction
```

```
>myload(file)
```

0.07167	0	0.5676	0	0.73012
0.85505	0	0.3255	0	0.44932
0.15451	0	0.98227	0	0.77675

It would also be possible to read all numbers in that file with `getvector()`.

```
>open(file); v=getvector(10000); close(); redim(v[1:9],3,3)
```

```

0.07167      0      0.5676
      0   0.73012  0.85505
      0   0.3255      0

```

Thus it is very easy to save a vector of values, one value in each line and read back this vector.

```
>v=random(1000); mean(v)
```

```
0.50337
```

```
>writematrix(v',file); mean(readmatrix(file)')
```

```
0.50337
```

Using Tables

Tables can be used to read or write numerical data. For an example, we write a table with row and column headers to a file.

```

>file="test.tab"; M=random(3,3); ...
>open(file,"w"); ...
>writetable(M,separator=",",labc=[ "one","two","three"]); ...
>close(); ...
>printfile(file)

```

```

one,two,three
 0.97,      0.24,      0.44
 0.12,      0.84,      0.35
 0.58,      0.58,      0.05

```

This can be imported into Excel.

To read the file in EMT, we use `readtable()`.

```

>{M,headings}=readtable(file,>clabs); ...
>writetable(M,labc=headings)

```

one	two	three
0.97	0.24	0.44
0.12	0.84	0.35
0.58	0.58	0.05

Analyzing a Line

You could even evaluate each line by hand. Suppose, we have a line of the following format.

```
>line="2020-11-03,Tue,1'114.05"
```

```
2020-11-03,Tue,1'114.05
```

First we can tokenize the line.

```
>vt=strtoks(line)
```

2020-11-03

Tue

1'114.05

Then we can evaluate each element of the line using appropriate evaluations.

```
>day(vt[1]), ...
>indexof(["mon","tue","wed","thu","fri","sat","sun"],tolower(vt[2])), ...
>strrepl(vt[3], "'", "")()
```

7.3816e+05
2
1114

Using regular expressions, it is possible to extract almost any information from a line of data.
Assume we have the following line an HTML document.

```
>line="<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>"
```

<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>

To extract this, we use a regular expression, which searches for

- a closing bracket >,
- any string not containing brackets with a sub-match "(...)" ,
- an opening and a closing bracket using the shortest solution,
- again any string not containing brackets,
- and an opening bracket <.

Regular expressions are somewhat difficult to learn but very powerful.

```
>{pos,s,vt}=strxfind(line,>([>[^<>]+)<.+?>([>[^<>]+)<" );
```

The result is the position of the match, the matched string, and a vector of strings for sub-matches.

```
>for k=1:length(vt); vt[k](), end;
```

1145.5
5.6

Here is a function, which reads all numerical items between <td> and </td>.

```
>function readtd (line) ...
v=[]; cp=0;
repeat
{pos,s,vt}=strxfind(line,"<td.*?>(.+?)</td>",cp);
until pos==0;
```

```

    if length(vt)>0 then v=v|vt[1]; endif;
    cp=pos+strlen(s);
end;
return v;
endfunction

```

```
>readtd(line+"<td>non-numerical</td>")
```

```

1145.45
5.6
-4.5
non-numerical

```

Reading from the Web

A web site or a file with an URL can be opened in EMT and can be read line by line.

In the example, we read the current version from the EMT site. We use regular expression to scan for "Version ..." in a heading.

```
>function readversion () ...
```

```

urlopen("http://www.euler-math-toolbox.de/Programs/Changes.html");
repeat
  until urleof();
  s=urlgetline();
  k=strfind(s,"Version ",1);
  if k>0 then substring(s,k,strfind(s,<,k)-1), break; endif;
end;
urlclose();
endfunction

```

```
>readversion
```

```
Version 2022-05-18
```

Input and Output of Variables

You can write a variable in the form of an Euler definition to a file or to the command line.

```
>writevar(pi,"mypi");
```

```
mypi = 3.141592653589793;
```

For a test, we generate an Euler file in the work directory of EMT.

```

>file="test.e"; ...
>writevar(random(2,2),"M",file); ...
>printfile(file,3)

```

```
M = [ ..
0.5403599552487005, 0.07421314404724037;
0.7263059512414065, 0.5694313810214714];
```

We can now load the file. It will define the matrix M.

```
>load(file); show M,
```

```
M =  
0.54036  0.074213  
0.72631  0.56943
```

By the way, if writevar() is used on a variable, it will print the variable definition with the name of this variable.

```
>writevar(M); writevar(inch$)
```

```
M = [ ..  
0.5403599552487005, 0.07421314404724037;  
0.7263059512414065, 0.5694313810214714];  
inch$ = 0.0254;
```

We can also open a new file or append to an existing file. In the example we append to the previously generated file.

```
>open(file,"a"); ...  
>writevar(random(2,2),"M1"); ...  
>writevar(random(3,1),"M2"); ...  
>close();  
>load(file); show M1; show M2;
```

```
M1 =  
0.57886  0.17807  
0.68275  0.033488  
M2 =  
0.029606  
0.68824  
0.89453
```

To remove any files use fileremove().

```
>fileremove(file);
```

A row vector in a file does not need commas, if each number is in a new line. Let us generate such a file, writing every line one by one with writeln().

```
>open(file,"w"); writeln("M = ["); ...  
>for i=1 to 5; writeln(""+random()); end; ...  
>writeln("]"); close(); ...  
>printfile(file)
```

```
M = [  
0.567415056392  
0.718507842195  
0.582608560809
```

```
0.0592582649723  
0.596425711501  
];
```

```
>load(file); M
```

```
[0.56742, 0.71851, 0.58261, 0.059258, 0.59643]
```