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:

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

```
>$&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))))
```

The Command Line

Sebuah "Command Line of Euler" mengandung satu atau beberapa perintah Euler yang diikuti dengan seabuah tanda semicolon ";" atau tanda koma ",". Semicolon dalam hal ini menunjukkan untuk memberikan hasil. Tanda koma setelah perintah terakhir untuk dihilangkan atau ditiadakan.
Beberapa "command line" berikut hanya akan ditunjukkan hasil dari ekspresi matematika, bukan untuk penugasan atau suatu format beberapa perintah.

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

```
  16.7551608191
```

Perintah harus dipisahkan dengan blank. Baris perintah selanjutnya akan memunculkan hasilnya menjadi dua.

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

```
  50.2654824574
  100.530964915
```

Baris Perintah akan dieksekusi dengan urutan perintah penggunanya. Jadi anda dapat sebuah value baru setiap kali anda mengeksekusi baris yang kedua.

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

```
  0.540302305868
```

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

```
  0.857553215846
```

Ketika dua baris disambungkan dengan "..." kedua baris akan selalu dieksekusi 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
```

Hal ini juga dapat menjadi langkah yang baik untuk menyenatkan sebuah perintah yang panjang yang mengandung dua atau lebih baris. Anda bisa menekan Ctrl+Returm untuk membagi sebuah baris menjadi sua dalam beberapa posisi kursor, atau Ctrl+Back untuk menggabungkan baris tersebut.
Untuk menggabungkan semua multi-lines tekan Ctrl+L. Selanjutnya hanya beberapa baris yang akan terlihat, jika satu dari baris tersebut telah fokus. Untuk menggabungkan satu multy-line mulai dengan baris pertama dengan"%+".

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

Sebuah baris dimulai dengan tanda %% akan menjadi tidak terlihat sepenuhnya.

```
  81
```

Euler support pengulangan dalam perintah baris, selama pengulangan tersebut muat dijadikan satu baris perintah atau muat dijadikan sebuah multy-line. Dalam program, pembatasan ini tidak dapat dikendalika, untuk informasi lebih lanjut terdapat dalam perkenalan lanjutan.

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

```
  1.5
  1.41666666667
  1.41421568627
  1.41421356237
  1.41421356237
```

Tidak apa-apa untuk menggunakan multy-line. Pastikan baris diakhiri dengan "..."

```
>x := 1.5; // komentar disini sebelum ...
>repeat xnew:=(x+2/x)/2; until xnew~=x; ...
>   x := xnew; ...
>end; ...
>x,
```

```
  1.41421356237
```

Struktur khusus juga dapat bekerja disini.

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

```
  Thought so!
```

Saat anda mengeksekusi sebuah perintah, kursor dapat berada dimanapun di baris perintah. Anda dapat kembali ke perintah sebelumnya atau melanjutkan ke perintah selanjutnya menggunakan tombol arah. Atau Anda juga bisa meng-klik sesi komentar diatas menggunakan perintah untuk melakukan perintah.
Ketika Anda memindahkan kursor saat baris pembukaan dan baris penutupan disambungkan dari baris perintah akan digarisbawahi. Juga, lihat baris status. Setelah baris pembuka dari fungsi sqrt(), baris status akan menampilkan sebuah kata bantuan untuk fungsi tersebut. Eksekusi perintah dengan tombol return.

```
>sqrt(sin(10°)/cos(20°))
```

```
  0.429875017772
```

Untuk menengok bantuan dalam perintah yang sering digunakan, buka jendela bantuan dengan menekan tombol F1. Disana, Anda dapat memasukkan kaata untuk dicari. Dalam sebuah baris yang kosong, bantuan untuk jendela bantuan akan ditampilkan. Anda dapat menekan escape untuk meniadakan baris tersebut, atau tutup jendela bantuan.
Anda dapat meng-klik dua kali pada setiap perintah untuk membuka bantuan dalam perintah ini. Cobalah untuk melakukan double click perintah exp dibawah dalam baris perintah.

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

```
  2.5
```

Basic Syntax

Euler tau fungsi matematika yang sering digunakan. Selama Anda pernah melihatnya, fungsi trigonometri dapat bekerja dalam radian atau derajat. Untuk mengubah menjadi derajat, tambahkan simbol derajat (dengan tombol F7) untuk isinya, atau gunakan fungsi rad(x). Fungsi kuadrat dikenal dengan sqrt di Euler. Dan tentu saja, juga memungkinkan x^(1/2).
Untuk menentukan variabel, gunakan "=" atau ":=". Untuk menentukan maksudnya, perkenalan ini menggunakan latter form. Spasi tidak masalah, namun spasi diantara perintah juga harus diperhatikan.
Beberapa perintah dalam satu barus akan dipisahkan dengan "," atau ";". Simbol semicolon menandakan output dari perintah. Dalam akhir sebuah baris sebuah "," juga dinilai, jika ";" hilang.

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

```
  30.65625
```

EMT menggunakan sebuat sintax pemograman untuk banyak ekspresi. Untuk memasukkan

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

Anda mempunyai himpunan dari kata yang benar dan menggunakan "/" untuk pecahan. Lihat dengan saksama kata yang digarisbawahi untuk bantuan. Untuk catatan bahwa Euler konstan e bernama E di EMT.

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

```
  8.77908249441
```

Untuk menghitung ekspresi yang rumit seperti ini

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

kamu harus menginput di baris

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

```
  23.2671801626
```

Hati-hati dalam memasukkan kata disekitar sub-ekspresi yang seharusnya dihitung pertama. EMT membantu Anda dengan menggarisbawahi ekspresi yang dekat dengan kata terakhir. Anda juga harus memasukkan nama "pi" untuk huruf romawi pi.
Hasil dari komputasi ini adalah floating point number atau dalam bahasa indonesia disebut aritmetika titik kambang. Dalam setelan pabrik dituliskan dengan sekitar akurasi 12 digit. Dalam baris perintah yang selanjutnya, kita akan belajar tentang bagaimana cara untuk mrnggunakan hasil sebelumnya dalam satu baris.

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

```
  0.47619047619
  10/21
```

Sebuah perintah Euler daoar menjadi ekspresi ataupun sebuah perintah lama. Sebuah ekspresi dibuat dengan adanya operator dan fungsi. Jika diperlukan itu juga harus mengandung katauntuk menekan urutan yang tepat saat proses eksekusi. Jika terdapat keraguan, ubah setelan kata juga merupakan ide yang bagus. Catatan bahwa EMT menggolongkan opening dan closing saat mengedit perintah baris.

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

```
  14.4978445072
```

Numerical operator di Euler meliputi
+ untuk penjumlahan plus
- untuk pengurangan minus
*, / untuk simbol perkalian / untuk simbol pembagian
. merupakan produk matriks
a^b pangkat untuk bilangan positif atau integer b
n! operasi faktorial
dan masih banyak lagi
Beberapa fungsi yang mungkin dibutuhkan, dan maish 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 sinonim contohnya aliases, e.g. ln for log.

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

```
  2
  0.5
```

```
>sin(30°)
```

```
  0.5
```

Paastikan untuk menggunakan parentheses, dimanapun ternyata terdapat keraguan atas urutan dari eksekusi! Urutan tidak akan sama seperti (2^3)^4, dimana setelan pabriknya yaitu 2^3^4 di EMT(beberapa sistem numeric menggunakan cara yang berbeda)

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

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

Real Numbers (Bilangan Riil)

Data tipe primeer dalam Euler adalah Bilangan Riil. Rill merepresentasikan di format IEEE dengan akurasi sekitar 16 digit desimal.

```
>longest 1/3
```

```
       0.3333333333333333
```

representasi dari dual internal menggunakan 8 bytes

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

```
  1.0101010101010101010101010101010101010101010101010101*2^-2
```

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

```
  5.5555555555554*16^-1
```

Strings

String di Euler didefinisikan dengan "...".

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

```
  A string can contain anything.
```

String dapat digabungkan dega | atau dengan +. Ini juga bekerja dengan angka, yang mana dapat diubah menjadi string dalam kendala ini.

```
>"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.
```

Hasil dari fungsi juga dapat mngubah angka menjadi string. Itu juga dapat mengambil digit dari angka dan tempat dari angka (0 untuk dense output), dan mengoptimalisasi unit.

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

```
  Golden Ratio : 1.61803
```

Ada beberapa string none yang spesial, yang mana tidak dituliskam. Itu berubah menjadi beberapa fungsi, saat hasilnya tidak terlalu berpengaruh. (Itu adlaah pengembalian otomatis, jika fungsinya tidak memiliki pernytaan pengembalian atau pernyataan ulang.)

```
>none
```

Untuk mengubah string menjadi angka dengan mudah mengevaluasi itu. Ini bekerja untuk ekspresi ini juga (lihat bawah).

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

```
  1234.5
```

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

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

```
  affe
  charlie
  bravo
```

Vektor String kosong dinotasikan dengan [none]. String vektor juga bisa digabungkan.

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

```
  affe
  charlie
  bravo
  affe
  charlie
  bravo
```

Stirng dapat mengandung karakter Unicode. Dalam hal ini, string mengandung kode UTF-8. Untuk mengumumkan seperti string, gunakan u"..." dan satu dari entitas HTML.

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

```
   = 45°
```

I

Dalam komentar, entitas yang sama seperti \alpha , \beta dan sebagainya dapat digunakan. Cara ini mungkin sebuah cara cepat dalam latex. (Detail lebih banyak ada di komen dibawah)

Ini beberapa fungsi untuk membuat atau menganalisa string unicode. Fungsi strtochar() akan direkognisi string Unicode, dan menerjemahkan dengan baik.

```
>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 vector unicide. Fungsi pembalik adalah chartoutf().

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

```
  Ü is a German letter
```

Fungsi utf() dapat diterjemahkan sebagai string dengan entitas dalam sebuah variabel dalam string unicode.

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

```
  We have =.
```

Ini juga memungkinkan untuk menggunakan entitas numerik.

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

```
  Ähnliches
```

Boolean Values

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

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

```
  0
  1
```

"and" adalah operasi "&&" atau "or" adalah operasi "||", sebagai bahasa dari C program. (Kata "and" dan "or" digunakan hanya dalam kondisi "if")

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

```
  1
```

Operasi Boolean juag memperbolehkan aturan dari bahasa matriks.

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

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

Anda juga dapat menggunakan fungsi nonzerol() untuk mengekstrak bentuk elemen khusus sebuah vektor. Seperti contoh, kita dapat kondisi isprime(n).

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

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

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

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

Output Formats

Format keluaran default ini mencetak 12 digit. Untuk menyesuaikan bahwa kita dapat melihat setelan deafultnya, kita dapat mereset format.

```
>defformat; pi
```

```
  3.14159265359
```

Dalam masalah ini, EMT menggunakan standar IEE untuk doubled numbers dengan sekitar 16 decimal digit. Untuk melihat angka penuh dengan digit, gunakan perintah "longestformat", atau gunakan operasi "longest" untuk menampilkan hasil dalam format terpanjang.

```
>longest pi
```

```
        3.141592653589793
```

Disini adalah masalah hexadecimal yang merepresentasikan dari sebuah angka double.

```
>printhex(pi)
```

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

Keluaran foramt dapat diubah permanen dengan format perintahnya.

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

```
      0.33333
      3.14159
      0.84147
```

format defaultnya adalah (12)

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

```
  0.333333333333
```

Fungsi seperti "shortestformat", "shortformat", "longformat" bekerja untuk vektor dalam cara yang berkelanjutan.

```
>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
```

Default format untuk scalar adalah format (12). Namun ini dapat dirubah.

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

```
  3.1416
```

Fungsi "longestformat" mengatur format skalar juga.

```
>longestformat; pi
```

```
  3.141592653589793
```

Untuk referensi, disini adalah daftar dari format output yang penting.

```
shortestformat shortformat longformat, longestformat
format(length,digits) goodformat(length)
fracformat(length)
defformat
```

akurasi internal dari EMTadalah sekitar 16 desimal, yang mana IEEE standar.

```
>longestformat; pi,
```

```
  3.141592653589793
```

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

```
    3.14159
```

defaultnya yaitu defformat().

```
>defformat; // default
```

ada operasi pendek yang mana mencetak hanya satu value. Operasi "longest" akan deictak semua digit yang valid adalah angka.

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

```
        4.934802200544679
```

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

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

```
  25/12
```

Karena format internal menggunakan cara biner untuk menyimpan angka, maka nilai 0,1 tidak akan terwakili dengan tepat. Kesalahan bertambah sedikit, seperti yang Anda lihat dalam perhitungan berikut ini.

```
>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 "longformat" default, Anda tidak akan melihat hal ini. Untuk kenyamanan, output angka yang sangat kecil adalah 0.

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

```
  0
```

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 tersebut. Parameter tambahan dapat ditambahkan dengan menggunakan parameter yang ditetapkan.

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

```
  -0.919535764538
```

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

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

```
  36
```

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

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

```
  45
```

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

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

```
  45
```

Ekspresi dalam x sering digunakan seperti halnya fungsi.
Perhatikan bahwa mendefinisikan fungsi dengan nama yang sama seperti ekspresi simbolik global akan menghapus variabel ini untuk menghindari kebingungan antara ekspresi simbolik dan fungsi.

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

```
  12
```

Sesuai dengan 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 sebuah ekspresi memungkinkan variabel apa pun sebagai parameter tanpa nama untuk evaluasi ekspresi, bukan hanya "x", "y", dll. Untuk ini, mulailah ekspresi dengan "@(variabel)...".

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

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

Hal 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 simbolik atau numerik. Jika variabel utamanya adalah x, ekspresi tersebut dapat dievaluasi seperti halnya fungsi.
Seperti yang Anda lihat pada 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 berbentuk simbolik. Hal ini diperlukan, jika ekspresi mengandung fungsi-fungsi, yang hanya dikenal di kernel numerik, bukan di Maxima. **Symbolic Mathematics**

EMT melakukan matematika simbolik dengan bantuan Maxima. Untuk detailnya, mulailah dengan tutorial berikut ini, atau telusuri referensi untuk Maxima. Para ahli dalam Maxima harus mencatat bahwa ada perbedaan dalam sintaks antara sintaks asli Maxima dan sintaks default ekspresi simbolik dalam EMT.
Matematika simbolik diintegrasikan dengan mulus ke dalam Euler dengan &. Setiap ekspresi yang dimulai dengan & adalah ekspresi simbolik. Ekspresi ini dievaluasi dan dicetak oleh Maxima.
Pertama-tama, Maxima memiliki aritmatika "tak terbatas" yang dapat menangani angka yang sangat besar.

```
>$&44!
```

$$26582715747884487680436258110146158903196385280000000000$$

Dengan cara ini, Anda dapat menghitung hasil yang besar secara 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 hal ini (seperti halnya bagian numerik EMT).

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

Untuk mempelajari lebih lanjut tentang fungsi tertentu, klik dua kali pada fungsi tersebut. Sebagai contoh, coba klik dua kali pada "&binomial" di baris perintah sebelumnya. Ini akan membuka dokumentasi Maxima yang disediakan oleh pembuat program tersebut.
Anda akan mengetahui bahwa hal berikut ini juga dapat digunakan.

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

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

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

Dengan begitu, Anda dapat menggunakan solusi dari sebuah persamaan dalam persamaan lain.
Ekspresi simbolik dicetak oleh Maxima dalam bentuk 2D. Alasan untuk ini adalah bendera simbolis khusus dalam string.
Seperti yang telah Anda lihat pada contoh sebelumnya dan contoh berikut, jika Anda memiliki LaTeX yang terinstal, Anda dapat mencetak ekspresi simbolik dengan Latex. Jika tidak, perintah berikut ini akan mengeluarkan pesan kesalahan.
Untuk mencetak ekspresi simbolik dengan LaTeX, gunakan $ di depan & (atau Anda dapat menghilangkan &) sebelum perintah. Jangan jalankan perintah Maxima dengan $, jika Anda tidak memiliki LaTeX yang terinstal.

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

Ekspresi simbolik diuraikan oleh Euler. Jika Anda membutuhkan sintaks yang kompleks dalam satu ekspresi, Anda dapat mengapit ekspresi dalam "...". Menggunakan lebih dari satu ekspresi sederhana dimungkinkan, tetapi sangat tidak disarankan.

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


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

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

Sekali lagi, % mengacu pada hasil sebelumnya.
Untuk mempermudah, kita menyimpan solusi ke dalam sebuah variabel simbolik. Variabel simbolik didefinisikan dengan "&=".

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

Ekspresi simbolik dapat digunakan dalam ekspresi simbolik lainnya.

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

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!)
```

$$8 \quad 4 \quad 2$$
$$2 \quad 3 \quad 5 \quad 7$$

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

$$18 \quad 8 \quad 4 \quad 2$$
$$2 \quad 3 \quad 5 \quad 7 \quad 11 \quad 13 \quad 17 \quad 19$$

Jika Anda adalah seorang ahli dalam Maxima, Anda mungkin ingin menggunakan sintaks asli Maxima. Anda dapat melakukan ini dengan ":::".

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

$$2$$
$$g$$

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

$$3 \quad x$$
$$x \quad E$$

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

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

$$125\,E^{5}$$

$$18551.64488782208$$

```
>fx(5)
```

```
  18551.6448878
```

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

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

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

$$2.20079141499189e+7$$

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

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

```
>tex(fx)
```

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

Ekspresi simbolik dapat dievaluasi seperti halnya ekspresi numerik.

```
>fx(0.5)
```

```
  0.206090158838
```

Dalam ekspresi simbolik, hal ini tidak dapat dilakukan, karena Maxima tidak mendukungnya. Sebagai gantinya, gunakan sintaks "with" (bentuk yang lebih baik dari perintah at(...) pada Maxima).

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

Penugasan ini juga bisa bersifat simbolis.

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

Perintah solve menyelesaikan ekspresi simbolik untuk sebuah variabel di Maxima. Hasilnya adalah sebuah vektor solusi.

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

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

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

```
  1.56155281281
```

Nilai numerik dari solusi simbolik dapat dihitung dengan evaluasi hasil simbolik. Euler akan membaca penugasan x= dst. Jika Anda tidak membutuhkan hasil numerik untuk perhitungan lebih lanjut, Anda juga bisa membiarkan Maxima menemukan nilai numeriknya.

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

```
  [-3.23607,  1.23607]
```

Untuk mendapatkan solusi simbolik yang spesifik, seseorang dapat menggunakan "dengan" dan indeks.

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

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]
```

Ekspresi simbolik dapat memiliki bendera, yang menunjukkan perlakuan khusus di Maxima. Beberapa flag dapat digunakan sebagai perintah juga, namun ada juga yang tidak. Bendera ditambahkan dengan "|" (bentuk yang lebih baik 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

Dalam EMT, fungsi adalah program yang didefinisikan dengan perintah "function". Fungsi ini dapat berupa fungsi satu baris atau fungsi multiline.
Fungsi satu baris dapat berupa numerik atau simbolik. Fungsi satu baris numerik didefinisikan dengan ":=".

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

Sebagai gambaran umum, kami menunjukkan semua definisi yang mungkin untuk fungsi satu baris. Sebuah fungsi dapat dievaluasi seperti halnya fungsi Euler bawaan.

```
>f(2)
```

```
  4.472135955
```

Fungsi ini juga dapat digunakan untuk vektor, mengikuti bahasa matriks Euler, karena ekspresi yang digunakan dalam fungsi ini adalah vektor.

```
>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]
```

Fungsi dapat diplot. Alih-alih ekspresi, kita hanya perlu memberikan nama fungsi.
Berbeda dengan ekspresi simbolik atau numerik, nama fungsi harus disediakan dalam bentuk string.

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

```
  0.786151377757
```

Secara default, jika Anda perlu menimpa fungsi built-in, Anda harus menambahkan kata kunci "overwrite". Menimpa fungsi bawaan berbahaya dan dapat menyebabkan masalah pada fungsi lain yang bergantung pada fungsi tersebut.
Anda masih dapat memanggil fungsi bawaan sebagai "_...", jika itu adalah fungsi dalam inti Euler.

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

```
  0.707106781187
```

Sebaiknya kita hilangkan definisi ulang tentang sin.

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

```
  0.707106781187
```

Default Parameters

Fungsi numerik dapat memiliki parameter default.

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

Menghilangkan parameter ini menggunakan nilai default.

```
>f(4)
```

```
  16
```

Menetapkannya akan menimpa nilai default.

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

```
  80
```

Parameter yang ditetapkan juga menimpanya. Ini digunakan oleh banyak fungsi Euler seperti plot2d, plot3d.

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

```
  16
```

Jika sebuah variabel bukan parameter, maka variabel tersebut harus bersifat global. Fungsi satu baris dapat melihat variabel global.

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

```
  24
```

Tetapi, parameter yang ditetapkan akan mengesampingkan nilai global.
Jika argumen tidak terdapat dalam daftar parameter yang telah ditentukan sebelumnya, argumen tersebut harus dideklarasikan dengan ":="!

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

```
  20
```

Fungsi simbolik didefinisikan dengan "&=". Fungsi-fungsi ini didefinisikan dalam Euler dan Maxima, dan dapat digunakan di kedua bahasa tersebut. Ekspresi pendefinisian dijalankan melalui Maxima sebelum definisi.

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

Fungsi simbolis dapat digunakan dalam ekspresi simbolis.

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

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

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

```
  178.635099908
```

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
```

Hal berikut ini juga dapat digunakan, karena Euler menggunakan ekspresi simbolik dalam fungsi g, jika tidak menemukan variabel simbolik g, dan jika ada fungsi simbolik 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)
```

Dengan &=, fungsi ini bersifat simbolis, dan dapat digunakan dalam ekspresi simbolis lainnya.

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

Dengan := fungsi tersebut berupa angka. Contoh yang baik adalah integral pasti seperti

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

dimana tidak dapat dievaluasikan sebagai simbol.
Jika kita mendefinisikan ulang fungsinya dengan keyword "map" itu bisa digunakan untuk vektor x. Dimana fungsinya dapat dinamakan untuk semua value dalam sekali, dan hasilnya akan dikembalikan menjadi vektor.

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

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

Fungsi dapat memiliki nilai default untuk parameter.

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

Sekarang, fungsi ini dapat dipanggil dengan atau tanpa parameter "base".

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

```
  2
  6.7
```

Selain itu, dimungkinkan untuk menggunakan parameter yang ditetapkan.

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

```
  2
```

Sering kali, kita ingin menggunakan fungsi untuk vektor di satu tempat, dan untuk masing-masing elemen di tempat lain. Hal ini dimungkinkan dengan parameter vektor.

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

Fungsi simbolik seperti itu dapat digunakan untuk variabel simbolik.
Tetapi fungsi ini juga dapat digunakan untuk vektor numerik.

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

```
  17
```

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

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

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

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

Tetapi tentu saja, semua itu bisa digunakan dalam ekspresi simbolis atau dalam definisi fungsi simbolis.

```
>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 yang murni simbolis.

Solving Expressions

Ekspresi dapat diselesaikan secara numerik dan simbolis.
Untuk menyelesaikan ekspresi sederhana dari satu variabel, kita dapat menggunakan fungsi solve(). Fungsi ini membutuhkan sebuah nilai awal untuk memulai pencarian. Secara internal, solve() menggunakan metode secant.

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

```
  1.41421356237
```

Hal ini juga bisa digunakan untuk ekspresi simbolis. Perhatikan fungsi berikut ini.

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

Sekarang kita mencari titik, di mana polinomialnya adalah 2. Dalam solve(), nilai target default y=0 dapat diubah dengan variabel yang ditetapkan.
Kami menggunakan y = 2 dan memeriksa dengan mengevaluasi polinomial pada hasil sebelumnya.

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

```
  0.966715594851
  2
```

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

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

Cara termudah untuk mendapatkan nilai numerik adalah dengan mengevaluasi solusi secara numerik seperti sebuah 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])
```

Menyelesaikan sistem persamaan secara simbolik dapat dilakukan dengan vektor persamaan dan pemecah simbolik solve(). Jawabannya adalah sebuah daftar daftar persamaan.

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

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

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

with a=3.

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

Salah satu cara untuk mengoper parameter tambahan ke f() adalah dengan menggunakan sebuah daftar yang berisi nama fungsi dan parameternya (cara lainnya adalah dengan menggunakan parameter titik koma).

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

```
  2.54116291558
```

Hal ini juga dapat dilakukan dengan ekspresi. Namun, elemen daftar bernama harus digunakan. (Lebih lanjut tentang daftar dalam 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])
```

Dokumentasi inti EMT berisi diskusi terperinci tentang bahasa matriks Euler.
Vektor dan matriks dimasukkan dengan tanda kurung siku, elemen dipisahkan dengan koma, baris dipisahkan dengan titik koma.

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

```
            1             2
            3             4
```

Hasil kali matriks dilambangkan dengan sebuah 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 demi elemen.

```
>A.A
```

```
                7               10
               15               22
```

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

```
                1                4
                9               16
```

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

```
               37               54
               81              118
```

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

```
               37               54
               81              118
```

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

```
                1                1
                1                1
```

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

```
         0.333333         0.666667
             0.75                1
```

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

```
               -2
              2.5
```

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

```
              -2
             2.5
```

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

```
               1              0
               0              1
```

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

```
               1              0
               0              1
```

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

```
               1              4
               9             16
```

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

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

```
               9
              16
```

Jika salah satu operan adalah vektor atau skalar, maka operan tersebut akan diperluas dengan cara alami. Misalnya, jika operan adalah vektor kolom, elemen-elemennya diterapkan ke semua baris A.

```
>2*A
```

```
               2              4
               6              8
```

E.g., if the operand is a column vector its elements are applied to all rows of A.

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

```
               1              4
               3              8
```

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

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

```
               2              6
               6             12
```

Kita dapat membayangkan perkalian ini seolah-olah vektor baris v telah diduplikasi untuk membentuk matriks 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
```

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

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

```
                1               2               3               4               5
                2               4               6               8              10
                3               6               9              12              15
                4               8              12              16              20
                5              10              15              20              25
```

Sekali lagi, ingatlah bahwa ini bukan produk matriks!

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

```
  55
```

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

```
  55
```

Bahkan operator seperti < or== bekerja dengan cara yang sama.

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

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

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

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

```
  5
```

Euler memiliki operator perbandingan, seperti "==", yang memeriksa kesetaraan. Kita mendapatkan vektor 0 dan 1, 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,  0]
```

Dari vektor semacam itu, "non-zeros" memilih elemen bukan nol.
Dalam hal ini, kita mendapatkan indeks semua elemen yang lebih besar dari 50.

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

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

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

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

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

Sebagai contoh, mari kita cari semua kuadrat dari angka 1 sampai 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 komputasi bilangan bulat. EMT menggunakan floating point presisi ganda secara internal. Akan tetapi, hal ini sering kali sangat berguna.
Kita dapat memeriksa bilangan prima. Mari kita cari tahu, berapa banyak kuadrat ditambah 1 yang merupakan bilangan prima.

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

```
  112
```

Fungsi nonzeros() hanya bekerja 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 menetapkan elemen ke suatu 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-elemen pada indeks ke entri-entri dari beberapa matriks lain.

```
>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-elemen dalam vektor.

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

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

Fungsi lain yang berguna adalah extrema, 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 bisa menggunakan ini untuk mengekstrak nilai maksimal dalam setiap baris.

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

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

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

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

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

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

```
>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 membuat sebuah matriks, kita dapat menumpuk satu matriks di atas matriks lainnya. 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 matriks 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 keduanya 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 hal ini adalah untuk menginterpretasikan vektor ekspresi untuk vektor kolom.

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

```
                1               1
                2               4
                3               9
```

Untuk mendapatkan ukuran A, kita dapat menggunakan fungsi berikut ini.

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

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

Untuk vektor, ada length().

```
>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]
```

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

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

```
        0.66566       0.831835
          0.977       0.544258
```

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

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

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

Dengan fungsi berikut, kita dapat menggunakan fungsi ini dan fungsi dup untuk menulis fungsi rep(), yang mengulang sebuah vektor sebanyak 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-elemen vektor.

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

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

Fungsi flipx() dan flipy() membalik urutan baris atau kolom dari sebuah matriks. Misalnya, 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]
```

Fungsi khusus adalah drop(v,i), yang menghapus 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 dalam drop(v,i) merujuk pada indeks elemen dalam v, bukan nilai elemen. Jika Anda ingin menghapus elemen, Anda harus menemukan elemen-elemen tersebut terlebih dahulu. Fungsi indexof(v,x) dapat digunakan untuk menemukan 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,  0,  7,  0,  8,  0]
  [2,  3,  5,  7,  23,  29,  31,  37,  41,  43,  47]
```

Seperti yang Anda lihat, tidak ada salahnya menyertakan indeks di luar jangkauan (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 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, kami menetapkan diagonal bawah (-1) ke 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 kita tidak mengubah matriks A. Kita mendapatkan sebuah matriks baru sebagai hasil dari setdiag().
Berikut adalah sebuah fungsi yang mengembalikan sebuah matriks tri-diagonal.

```
>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 sebuah matriks juga dapat diekstrak dari matriks. Untuk mendemonstrasikan hal

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 bisa mengekstrak diagonal.

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

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

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

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

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

Vectorization

Almost all functions in Euler work for matrix and vector input too, whenever this makes sense.
E.g., the sqrt() function computes the square root of all elements of the vector or matrix.

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

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

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

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

With this and the colon operator a:delta:b, vectors of values of functions can be generated easily.
In the following example, we generate a vector of values t[i] with spacing 0.1 from -1 to 1. Then we generate a vector of values of the function

$$s = t^3 - t$$

```
>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 akan diperluas menjadi matriks, jika sebuah 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 dengan hasil kali matriks. Hasil kali matriks dilambangkan dengan sebuah titik "." dalam EMT.

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

```
  55
```

Secara default, vektor baris dicetak dalam format ringkas.

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

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

Untuk matriks, operator khusus . menyatakan perkalian matriks, dan A' menyatakan transposisi. Matriks 1x1 dapat digunakan seperti halnya bilangan real.

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

```
  5
  25
```

Untuk mentransposisi matriks, kita menggunakan apostrof.

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

```
                1
                2
                3
                4
```

Jadi kita dapat menghitung matriks A dikali vektor b.

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

```
               30
               70
```

Perhatikan bahwa v masih merupakan vektor baris. Jadi v'.v berbeda dengan v.v'.

```
>v'.v
```

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

v.v' menghitung norma v kuadrat untuk vektor baris v. Hasilnya adalah vektor 1x1, yang berfungsi seperti bilangan real.

```
>v.v'
```

```
  30
```

Ada juga norma fungsi (bersama dengan banyak fungsi Aljabar Linear lainnya).

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

```
  30
```

Operator dan fungsi mematuhi bahasa matriks Euler. Berikut ini adalah ringkasan dari aturan-aturan tersebut.
Fungsi yang diterapkan ke vektor atau matriks diterapkan ke setiap elemen.
Operator yang beroperasi pada dua matriks dengan ukuran yang sama diterapkan secara berpasangan pada elemen-elemen matriks.
Jika kedua matriks memiliki dimensi yang berbeda, keduanya diperluas dengan cara yang masuk akal, sehingga keduanya memiliki ukuran yang sama.
Misalnya, nilai skalar dikalikan vektor mengalikan nilai dengan setiap elemen vektor. Atau matriks dikali vektor (dengan *, bukan .) memperluas vektor ke ukuran matriks dengan menduplikasinya.
Berikut ini adalah kasus sederhana dengan operator ^.

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

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

Ini adalah kasus yang lebih rumit. Vektor baris dikalikan vektor kolom memperluas 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 produk *!

```
>v.v'
```

```
  14
```

Ada banyak fungsi untuk matriks. Kami memberikan daftar singkat. Anda harus membaca dokumentasi untuk informasi lebih lanjut mengenai perintah-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 dengan 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, terdapat operator "|" dan "_".

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

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

Elemen-elemen dari sebuah 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 tersebut. Untuk matriks, ini mengembalikan baris ke-i dari matriks.

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

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

Indeks juga dapat berupa 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-elemen matriks dapat diakses seolah-olah mereka adalah vektor.

```
>A{4}
```

```
  4
```

Sebuah matriks juga dapat diratakan, dengan menggunakan fungsi redim(). Hal 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 atur ulang 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
```

Using the matrix language, we can generate several tables of several functions at once.
In the following example, we compute t[j]^i for i from 1 to n. We get a matrix, where each row is a table of t^i for one i. I.e., the matrix has the elements latex: a_{i,j} = t_j^i, \quad 1 \le j \le Dengan menggunakan bahasa matriks, kita dapat menghasilkan beberapa tabel dari beberapa fungsi sekaligus.
Pada contoh berikut, kita menghitung t[j]^i untuk i dari 1 hingga n. Kita mendapatkan sebuah matriks, di mana setiap baris adalah tabel dari t^i untuk satu i. Dengan kata lain, matriks tersebut memiliki elemen-elemen
Fungsi yang tidak berfungsi untuk input vektor harus "divektorkan". Hal ini dapat dicapai dengan kata kunci "map" dalam definisi fungsi. Kemudian fungsi akan dievaluasi untuk setiap elemen parameter vektor.
Integrasi numerik integrate() hanya bekerja untuk batas interval skalar. Jadi, kita perlu membuat vektornya.
101, \quad 1 \le i \le n
A function which does not work for vector input should be "vectorized". This can be achieved by the "map" keyword in the function definition. Then the function will be evaluated for each element of a vector parameter.
The numerical integration integrate() works only for scalar interval bounds. So we need to vectorize it.

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

Kata kunci "map" membuat vektor fungsi. Fungsi ini sekarang akan bekerja untuk vektor angka.

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

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

Sub-Matrices and Matrix-Elements

Untuk mengakses elemen matriks, gunakan notasi kurung.

```
>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 baris lengkap dari sebuah matriks.

```
>A[2]
```

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

Untuk 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 yang kosong.

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

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

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

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, kita tidak mengubah A di sini, tetapi menghitung versi susunan ulang dari A.

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

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

Trik indeks juga dapat digunakan pada kolom.
Contoh ini memilih semua baris A dan kolom kedua dan ketiga.

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

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

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

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

```
                3
                6
                9
```

Sebagai alternatif, biarkan indeks pertama kosong.

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

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

Kita juga bisa mendapatkan baris terakhir A.

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

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

Sekarang mari kita ubah elemen-elemen dari A dengan memberikan sebuah submatriks dari A ke suatu nilai.
Hal ini sebenarnya mengubah matriks A yang tersimpan.

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

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

Kami juga dapat
menetapkan sebuah nilai ke A deretan

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

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

Kami bahkan dapat menetapkan ke sub-matriks jika itumemilik i 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 akan mengembalikan matriks kosong, atau pesan kesalahan, tergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Namun, ingatlah bahwa indeks negatif dapat digunakan untuk mengakses elemen-elemen matriks yang dihitung dari akhir.

```
>A[4]
```

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

Sorting and Shuffling

Fungsi mengurutkan() mengurutkan vektor baris.

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

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

Sering kali diperlukan untuk mengetahui indeks vektor yang diurutkan dalam vektor aslinya. Hal ini dapat digunakan untuk menyusun ulang vektor lain dengan cara yang sama.
Mari kita kocok sebuah vektor.

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

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

Indeks berisi urutan v yang tepat.

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

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

Hal ini juga berlaku untuk vektor string.

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

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

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

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

Seperti yang Anda lihat, posisi entri ganda agak acak.

```
>ind
```

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

Fungsi unik mengembalikan daftar terurut dari elemen unik vektor.

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

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

Hal ini juga berlaku untuk vektor string.

```
>unique(s)
```

```
  a
  aa
  d
  e
```

Linear Algebra

EMT memiliki banyak fungsi untuk menyelesaikan sistem linier, sistem jarang, 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
```

Sebagai contoh lain, kita membuat matriks 200x200 dan jumlah barisnya. Kemudian kita selesaikan Ax = b dengan menggunakan matriks kebalikannya. Kita mengukur kesalahan sebagai deviasi maksimal dari 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, pencocokan 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 dari matriks ini adalah 0.

```
>det(A)
```

```
  0
```

Symbolic Matrices

Maxima memiliki matriks simbolik. Tentu saja, Maxima dapat digunakan untuk masalah aljabar linier sederhana. Kita bisa mendefinisikan matriks untuk Euler dan Maxima dengan &:=, lalu menggunakannya dalam ekspresi simbolik. Bentuk [...] yang biasa untuk mendefinisikan matriks dapat digunakan dalam 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 sebuah vektor dengan dua vektor nilai eigen dan kelipatannya.

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

Untuk mengekstrak vektor eigen tertentu, diperlukan pengindeksan yang cermat.

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

```
[1, - 1]
```

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

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

```
                1                4
                5                2
```

Dalam ekspresi simbolis, gunakan dengan.

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

Akses ke baris matriks simbolik bekerja seperti halnya matriks numerik.

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

Ekspresi simbolik dapat berisi sebuah penugasan. Dan itu mengubah matriks A.

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

Terdapat fungsi-fungsi simbolik dalam Maxima untuk membuat vektor dan matriks. Untuk hal ini, lihat dokumentasi Maxima atau tutorial tentang Maxima di 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)()
```

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

Euler juga memiliki fungsi yang kuat xinv(), yang membuat usaha yang 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 dapat menggunakannya di sini.

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

```
                        1                        0
                        0                        1
```

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 mengetahui detailnya.

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

Numerical Values in symbolic Expressions

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

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

```
              1        3.14159
              4              5
```

Masih ada perbedaan antara bentuk numerik dan bentuk simbolik. Ketika mentransfer matriks ke bentuk simbolik, perkiraan pecahan untuk bilangan real akan digunakan.

```
>$&A
```

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

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

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

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

Ketepatan angka floating point yang besar dapat diubah.

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

```
          3.14159265358979323846264338327950288419716939937510582097494\
  459230781640628620899862803482534211706 8b0
```

Variabel numerik dapat digunakan dalam ekspresi simbolik apa pun dengan menggunakan "@var".
Perhatikan bahwa hal 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 menghitung suku bunga. Kami melakukannya secara numerik dan simbolik untuk menunjukkan kepada Anda bagaimana Euler dapat digunakan untuk memecahkan masalah kehidupan nyata.
Asumsikan Anda memiliki modal awal sebesar 5000 (katakanlah dalam dolar).

```
>K=5000
```

```
  5000
```

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

```
>K*1.03
```

```
  5150
```

Euler juga akan memahami sintaks berikut ini.

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

```
  5150
```

Tetapi akan lebih mudah untuk menggunakan faktor

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

```
  1.03
  5150
```

Untuk 10 tahun, kita cukup mengalikan faktor-faktor tersebut dan mendapatkan nilai akhir dengan suku bunga majemuk.

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

```
  6719.58189672
```

Untuk tujuan kita, kita bisa menetapkan format ke 2 digit setelah titik desimal.

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

```
      6719.58
```

Mari kita cetak angka 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 ke-1 hingga tahun ke-9? Untuk hal ini, bahasa matriks Euler sangat membantu. Anda tidak perlu menulis perulangan, 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 sebuah 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 vektor elemen demi elemen. Jadi

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

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

adalah vektor faktor q^0 hingga q^10. Ini dikalikan dengan K, dan kita mendapatkan vektor nilai.

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

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

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

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

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

```
                  1271.61
                1271.6071
```

Sekarang tidak ada rumus sederhana untuk tahun ke-n, dan kita harus mengulang selama bertahun-tahun. Euler menyediakan banyak solusi untuk ini.
Cara termudah adalah iterasi fungsi, yang mengulang fungsi tertentu beberapa kali.

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

```
  Real 1 x 11 matrix

      5000.00      5150.00      5304.50      5463.64      ...
```

Kami dapat mencetaknya dengan cara yang ramah, menggunakan format kami dengan angka

desimal yang 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, kita menggunakan indeks dalam tanda kurung siku.

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

```
      5150.00
      5000.00      5150.00      5304.50
```

Yang mengejutkan, kita juga dapat menggunakan vektor indeks. Ingatlah bahwa 1:3 menghasilkan vektor [1,2,3].
Mari kita bandingkan elemen terakhir dari nilai yang dibulatkan dengan nilai penuh.

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

```
      6719.60
      6719.58
```

Perbedaannya sangat kecil..

Solving Equations

Sekarang kita mengambil fungsi yang lebih canggih, yang menambahkan sejumlah uang setiap tahun.

```
>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 uangnya berkurang. Jelas, jika kita hanya mendapatkan 150 bunga di tahun pertama, tetapi menghapus 200 bunga, kita akan kehilangan uang setiap tahun.
Bagaimana kita dapat menentukan jumlah tahun uang tersebut akan bertahan? Kita harus menulis perulangan untuk ini. Cara termudah adalah dengan melakukan perulangan yang 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
```

Alasannya adalah karena nonzeros(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. Fungsi ini dapat menerima kondisi akhir sebagai argumen. Kemudian ia 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. Anggaplah kita tahu bahwa nilainya adalah 0 setelah 50 tahun. Berapakah tingkat suku bunganya?
Ini adalah pertanyaan yang hanya bisa dijawab secara numerik. Di bawah ini, kami akan menurunkan rumus yang diperlukan. Kemudian Anda akan melihat bahwa tidak ada rumus yang mudah untuk suku bunga. Namun untuk saat ini, kita akan mencari solusi numerik.
Langkah pertama adalah mendefinisikan sebuah 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

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

Tetapi kita tidak lagi menggunakan nilai global R dalam ekspresi kita. Fungsi-fungsi seperti iterate() memiliki trik khusus dalam Euler. Anda bisa mengoper nilai dari variabel-variabel dalam ekspresi sebagai parameter titik koma. Dalam kasus ini P dan R.
Selain itu, kita hanya tertarik pada nilai terakhir. Jadi kita ambil indeks [- 1]. Mari kita coba sebuah tes.

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

```
       -19.83
```

Sekarang kita bisa menyelesaikan masalah kita.

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

```
         3.15
```

Rutin penyelesaian menyelesaikan ekspresi = 0 untuk variabel x. Jawabannya adalah 3,15% per tahun. Kita mengambil nilai awal 3% untuk algoritma ini. Fungsi solve() selalu membutuhkan nilai awal.
Kita dapat menggunakan fungsi yang sama untuk menyelesaikan pertanyaan berikut: Berapa banyak yang dapat kita keluarkan per tahun agar modal awal habis setelah 20 tahun dengan asumsi suku bunga 3% per tahun.

```
>solve("f(5000,x,3,20)",-200)
```

```
      -336.08
```

Perhatikan bahwa Anda tidak dapat menyelesaikan jumlah tahun, karena fungsi kami mengasumsikan n sebagai nilai bilangan bulat.

Symbolic Solutions to the Interest Rate Problem

Kita bisa menggunakan bagian simbolik dari Euler untuk mempelajari masalah ini. Pertama, kita mendefinisikan fungsi onepay() secara simbolik.

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

Sekarang kita bisa mengulanginya.

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

Kita melihat sebuah pola. Setelah n periode, kita memiliki
Rumus tersebut adalah rumus untuk jumlah geometris, yang diketahui oleh Maxima.

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

Ini sedikit rumit. Jumlahnya dievaluasi dengan flag "simpsum" untuk menguranginya menjadi hasil bagi.
Mari kita buat 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)
```

Biasanya rumus ini diberikan dalam bentuk

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

```
       -19.82504734650985
       -19.82504734652684
```

Sekarang kita dapat menggunakannya untuk menanyakan waktu n. Kapan modal kita habis? Perkiraan awal kita 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 sisi simbolis dari Euler untuk menghitung rumus pembayaran.
Asumsikan kita mendapatkan pinjaman sebesar K, dan membayar n kali pembayaran sebesar R (dimulai setelah tahun pertama) sehingga menyisakan sisa utang sebesar Kn (pada saat pembayaran terakhir). Rumus untuk ini adalah sebagai berikut

```
>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 menyelesaikan laju R secara simbolis.

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

Seperti yang dapat Anda lihat dari rumusnya, fungsi ini mengembalikan kesalahan titik mengambang untuk i = 0. Euler tetap memplotnya.

Tentu saja, kami memiliki batasan sebagai berikut.

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

Jelas, tanpa bunga kita harus membayar kembali 10 suku bunga 500.
Persamaan ini juga dapat diselesaikan untuk n. Akan terlihat lebih baik jika kita menerapkan beberapa penyederhanaan.

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