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)
```

```
incorrect syntax: 6 is not an infix operator
&6*
 ^
```

Menjabarkan:

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

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

Baris perintah

Baris perintah Euler terdiri dari satu atau beberapa perintah Euler diikuti dengan 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 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 dijalankan sesuai urutan saat pengguna menekan tombol Enter. Jadi, anda akan mendapatkan nilai baru setiap kali menjalankan baris kedua.

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

```
0.540302305868
```

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

```
0.857553215846
```

Jika dua baris dihubungkan dengan "..." keduanya akan selalu dijalankan 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 membagi perintah panjang ke dalam dua atau lebih baris. Anda bisa menekan Ctrl+Return untuk memisahkan satu baris menjadi dua di posisi kursor saat ini, atau Ctrl+Back untuk menggabungkan kembali baris-baris tersebut.

Untuk melipat semua baris multi-line, tekan Ctrl+L. Maka baris-baris berikutnya hanya akan terlihat jika salah satunya sedang aktif (fokus). Untuk melipat satu baris multi-line secara khusus, awali baris pertama dengan '%+.

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

Baris yang dimulai dengan %% akan sepenuhnya tidak terlihat.

```
  81
```

Euler mendukung penggunaan loop (perulangan) dalam baris perintah, selama loop tersebut muat dalam satu baris atau dalam format multi-baris. Namun, dalam program, batasan ini tentu tidak berlaku. Untuk informasi lebih lanjut, silakan lihat pengantar berikut.

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

```
  1.5
  1.41666666667
  1.41421568627
  1.41421356237
  1.41421356237
```

Tidak apa apa untuk menggunakan multi-baris. 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 kondisional juga berfungsi.

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

```
  Thought so!
```

Saat anda menjalankan sebuah perintah, kursor bisa berada di posisi mana pun dalam baris perintah. Anda bisa kembali ke perintah sebelumnya atau melompat ke perintah berikutnya dengan tombol panah. Atau anda bisa mengklik bagian komentar di atas perintah untuk langsung menuju ke perintah tersebut.

Saat anda menggerakkan kursor di sepanjang baris, pasangan tanda kurung buka dan tutup akan disorot. Perhatikan juga baris status. Setelah tanda kurung buka pada fungsi sqrt(), baris status akan menampilkan teks bantuan untuk fungsi tersebut. Jalankan perintah dengan tombol Enter.

```
>sqrt(sin(10°)/cos(20°))
```

```
  0.429875017772
```

Untuk melihat bantuan dari perintah terakhir yang Anda jalankan, buka jendela bantuan dengan menekan tombol F1. Di sana, Anda bisa mengetik teks untuk mencari informasi. Jika barisnya kosong, maka yang muncul adalah bantuan tentang cara menggunakan jendela bantuan itu sendiri.
Anda bisa menekan tombol Escape untuk menghapus teks di baris tersebut, atau untuk menutup jendela bantuan.

Anda juga bisa klik dua kali pada perintah mana pun untuk membuka bantuan khusus tentang perintah itu. Coba klik dua kali pada perintah exp di baris perintah di bawah ini.

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

```
  2.5
```

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

Sintax Dasar

Euler mengenal fungsi-fungsi matematika yang umum. Seperti yang telah Anda lihat sebelumnya, fungsi trigonometri bekerja dalam satuan radian maupun derajat. Untuk mengubah ke derajat, tambahkan simbol derajat (dengan tombol F7) pada nilai tersebut, atau gunakan fungsi rad(x).
Fungsi akar kuadrat disebut sqrt dalam Euler. Tentu saja, bentuk x^(1/2) juga dapat digunakan.

Untuk menetapkan variabel, Anda dapat menggunakan tanda = atau :=. Demi kejelasan, pengantar ini menggunakan bentuk yang terakhir. Spasi tidak berpengaruh, namun spasi antar perintah tetap diperlukan.

Beberapa perintah dalam satu baris dipisahkan dengan tanda , atau ;. Tanda titik koma (;) akan menyembunyikan keluaran dari perintah tersebut. Di akhir baris perintah, tanda koma (,) akan dianggap ada jika tanda titik koma tidak digunakan.

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

```
30.65625
```

EMT menggunakan sintaks pemrograman untuk ekspresi. Untuk memasukkan

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

Anda harus menggunakan tanda kurung yang benar dan memakai simbol '/' untuk menyatakan pecahan. Perhatikan tanda kurung yang disorot sebagai bantuan. Perlu dicatat bahwa konstanta Euler 'e' disebut 'E' dalam EMT.

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

```
8.77908249441
```

Untuk menghitung ekspresi yang rumit seperti

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

Anda perlu memasukannya dalam bentuk satu 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 akan membantu Anda dengan menyorot ekspresi yang ditutup oleh tanda kurung akhir. Anda juga harus memasukkan nama 'pi' untuk mewakili huruf Yunani pi.

Hasil dari perhitungan ini adalah angka pecahan (floating point). Secara default, hasil tersebut ditampilkan dengan akurasi sekitar 12 digit.
Pada baris perintah berikutnya, Anda juga akan mempelajari cara merujuk ke hasil sebelumnya dalam baris yang sama.

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

```
  0.47619047619
  10/21
```

Perintah dalam Euler dapat berupa ekspresi atau perintah dasar (primitive command). Ekspresi terdiri dari operator dan fungsi. Jika diperlukan, ekspresi tersebut harus menggunakan tanda kurung untuk memastikan urutan eksekusi yang benar. Jika Anda ragu, menambahkan tanda kurung adalah pilihan yang bijak. Perlu diketahui bahwa EMT akan menampilkan pasangan tanda kurung buka dan tutup saat Anda mengedit baris perintah.

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

```
  14.4978445072
```

Operator numerikal Euler termasuk

```
+ 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 lain sebagainya.

Ini beberapa fungsi yang mungkin Anda butuhkan. Masih ada banyak lainnya.

```
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, misalnya ln sebagai pengganti 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 mengenai urutan eksekusi!

Ekspresi berikut tidak sama dengan (2^3)^4, yang merupakan bentuk default dari 2^3^4 dalam EMT (beberapa sistem numerik menerapkannya dengan cara yang berbeda).

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

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

Bilangan Asli

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

```
>longest 1/3
```

```
      0.3333333333333333
```

Representasi ganda internal menggunakan 8 byte.

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

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

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

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

Strings

Sebuah string dalam Euler didefinisikan dengan tanda kutip ganda (”...”).

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

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

String dapat digabungkan menggunakan simbol | atau tanda +. Hal ini juga berlaku untuk angka, yang akan dikonversi menjadi string dalam kasus tersebut.

```
>"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. Fungsi ini dapat menerima jumlah digit dan jumlah tempat desimal (0 untuk hasil kompak), serta secara opsional satuan.

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

```
  Golden Ratio : 1.61803
```

Terdapat string khusus bernama none, yang tidak akan dicetak. String ini dikembalikan oleh beberapa fungsi ketika hasilnya tidak penting. (String ini juga akan dikembalikan secara otomatis jika fungsi tersebut tidak memiliki pernyataan return.)

```
>none
```

Untuk mengonversi string menjadi angka, cukup evaluasi string tersebut. Cara ini juga berlaku untuk ekspresi (lihat di bawah).

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

```
  1234.5
```

Untuk mendefinisikan vektor string, gunakan notasi vektor dengan tanda kurung siku [...].

```
>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 menggunakan kode UTF-8. Untuk menghasilkan string semacam itu, gunakan format 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 &, a, dan sebagainya dapat digunakan. Ini bisa menjadi alternatif cepat untuk LaTeX. (Penjelasan lebih lanjut tentang komentar akan dibahas di bawah).

Terdapat 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 sebuah vektor berisi angka Unicode. Fungsi kebalikannya adalah chartoutf().

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

```
  Ü is a German letter
```

Fungsi utf() dapat menerjemahkan string yang berisi entitas dalam sebuah variabel menjadi string Unicode.

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

```
  We have =.
```

Anda juga dapat menggunakan entitas numerik.

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

```
  Ähnliches
```

Nilai Boolean

Nilai boolean dalam Euler direpresentasikan dengan 1 untuk benar (true) dan 0 untuk salah (false). String dapat dibandingkan seperti halnya angka.

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

```
  0
  1
```

Operator and ditulis sebagai "&&", dan operator or ditulis sebagai "||", seperti dalam bahasa C. (Kata "and" dan "or" hanya dapat digunakan dalam kondisi untuk perintah "if".)

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

```
  1
```

Operator boolean mengikuti aturan bahasa matriks.

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

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

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

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

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

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

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

Format Keluaran

Format keluaran default EMT mencetak 12 digit. Untuk memastikan bahwa Anda melihat format default tersebut, format perlu diatur ulang.

```
>defformat; pi
```

```
  3.14159265359
```

Secara internal, EMT menggunakan standar IEEE untuk bilangan double dengan akurasi sekitar 16 digit desimal. Untuk melihat seluruh digit secara lengkap, gunakan perintah longestformat, atau gunakan operator longest untuk menampilkan hasil dalam format terpanjang.

```
>longest pi
```

```
        3.141592653589793
```

Berikut adalah representasi heksadesimal internal dari sebuah bilangan double.

```
>printhex(pi)
```

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

Format keluaran dapat diubah secara permanen dengan perintah format.

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

```
      0.33333
      3.14159
      0.84147
```

Format default adalah format(12).

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

```
  0.333333333333
```

Fungsi seperti shortestformat, shortformat, dan longformat bekerja pada 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). Namun, pengaturan ini dapat diubah.

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

```
  3.1416
```

Fungsi longestformat juga menetapkan format tampilan untuk skalar.

```
>longestformat; pi
```

```
  3.141592653589793
```

Sebagai referensi, berikut adalah daftar format keluaran yang paling penting.

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

Akurasi internal EMT sekitar 16 angka di belakang koma, sesuai dengan standar IEEE. Angka-angka disimpan dalam format internal ini.

Namun, format keluaran EMT dapat diatur secara fleksibel.

```
>longestformat; pi,
```

```
  3.141592653589793
```

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

```
    3.14159
```

Format default-nya adalah defformat().

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

Ada operator pendek yang hanya mencetak satu nilai. Operator 'longest' akan mencetak semua digit valid dari sebuah angka.

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

```
        4.934802200544679
```

Ada juga operator pendek untuk mencetak hasil dalam format pecahan. Kita sudah menggunakannya di atas.

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

```
  25/12
```

Karena format internal menggunakan cara biner untuk menyimpan angka, nilai 0,1 tidak akan direpresentasikan secara tepat. Kesalahan ini akan bertambah sedikit demi sedikit, seperti yang terlihat 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
```

Namun dengan format default 'longformat', kamu tidak akan menyadarinya. Demi kenyamanan, angka yang sangat kecil akan ditampilkan sebagai 0.

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

```
  0
```

Ekspresi

String atau nama dapat digunakan untuk menyimpan ekspresi matematika, yang kemudian dapat dievaluasi oleh EMT. Untuk melakukannya, gunakan tanda kurung setelah ekspresi tersebut. Jika Anda bermaksud menggunakan string sebagai ekspresi, gunakan konvensi penamaan seperti 'fx' atau 'fxy', dan sebagainya. Dalam evaluasi, ekspresi memiliki prioritas lebih tinggi dibandingkan fungsi.

Variabel global juga dapat digunakan dalam proses evaluasi.

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

```
        12.56637061435917
```

Parameter akan ditetapkan ke x, y, dan z secara berurutan. Parameter tambahan dapat ditambahkan menggunakan parameter yang telah ditetapkan.

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

```
  -0.919535764538
```

Perlu dicatat bahwa ekspresi akan selalu menggunakan variabel global, bahkan jika terdapat variabel dengan nama yang sama di dalam suatu fungsi. (Jika tidak, evaluasi ekspresi dalam fungsi dapat menghasilkan 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=nilai".

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

```
  45
```

Sebagai referensi, perlu kami sampaikan bahwa koleksi pemanggilan (dibahas di bagian lain) dapat memuat ekspresi. Maka, contoh di atas dapat disusun ulang 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. Perlu dicatat bahwa jika Anda mendefinisikan sebuah fungsi dengan nama yang sama seperti ekspresi simbolik global, maka variabel tersebut akan dihapus untuk menghindari kebingungan antara ekspresi simbolik dan fungsi.

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

```
  12
```

Sebagai konvensi, ekspresi simbolik atau numerik sebaiknya diberi nama seperti fx, fxy, dan sebagainya. Skema penamaan ini tidak seharusnya digunakan untuk fungsi.

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

Bentuk khusus dari ekspresi memungkinkan penggunaan variabel apa pun sebagai parameter tanpa nama dalam proses evaluasi ekspresi, tidak hanya terbatas pada ”x”, ”y”, dan sebagainya. Untuk itu, awali ekspresi dengan ”@(variabel) ... ”.

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

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

Hal ini memungkinkan manipulasi ekspresi dengan variabel lain untuk fungsi-fungsi EMT yang membutuhkan ekspresi dalam "x".
Cara paling dasar untuk mendefinisikan fungsi sederhana adalah dengan menyimpan rumusnya dalam bentuk ekspresi simbolik atau numerik.
Jika variabel utama adalah x, maka ekspresi tersebut dapat dievaluasi seperti halnya fungsi. Seperti yang terlihat pada contoh berikut, variabel global tetap terlihat selama proses evaluasi.

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

```
  -0.475
```

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

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

```
  -0.425
```

Sebuah ekspresi tidak harus bersifat simbolik. Hal ini diperlukan apabila ekspresi tersebut memuat fungsi-fungsi yang hanya dikenal oleh kernel numerik, bukan oleh Maxima.

Matematika Simbolik

EMT melakukan matematika simbolik dengan bantuan Maxima. Untuk informasi lebih lanjut, mulailah dengan tutorial berikut atau telusuri referensi Maxima. Para ahli Maxima perlu mencatat bahwa terdapat perbedaan sintaks antara sintaks asli Maxima dan sintaks default ekspresi simbolik di EMT.

Matematika simbolik terintegrasi secara mulus ke dalam Euler melalui tanda &. Setiap ekspresi yang dimulai dengan & adalah ekspresi simbolik. Ekspresi tersebut akan dievaluasi dan ditampilkan oleh Maxima.

Pertama-tama, Maxima memiliki aritmetika "tak hingga" yang mampu menangani angka-angka yang sangat besar

```
>$&44!
```

Dengan cara ini, Anda dapat menghitung hasil yang sangat besar secara tepat. Mari kita lakukan perhitungannya.

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

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

Tentu saja, Maxima memiliki fungsi yang lebih efisien untuk hal ini (begitu pula dengan bagian numerik dari EMT).

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

Untuk mempelajari lebih lanjut tentang suatu fungsi tertentu, klik ganda pada fungsi tersebut. Misalnya, coba klik ganda pada '&binomial' di baris perintah sebelumnya. Ini akan membuka dokumentasi Maxima yang disediakan oleh para pengembang program tersebut.

Anda akan mengetahui bahwa perintah berikut juga dapat digunakan.

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

```
>$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 cara ini, Anda dapat menggunakan solusi dari suatu persamaan dalam persamaan lainnya.

Ekspresi simbolik dicetak oleh Maxima dalam bentuk 2D. Hal ini disebabkan oleh adanya flag simbolik khusus dalam string.

Seperti yang telah Anda lihat dalam contoh sebelumnya dan sesudahnya, jika Anda memiliki LaTeX yang terpasang, Anda dapat mencetak ekspresi simbolik menggunakan LaTeX. Jika tidak, perintah berikut akan menghasilkan pesan kesalahan.

Untuk mencetak ekspresi simbolik dengan LaTeX, gunakan tanda di depan &, atau Anda dapat menghilangkan & sebelum perintah. Jangan jalankan perintah Maxima dengan tanda, jika Anda belum memasang LaTeX.

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

Ekspresi simbolik diproses oleh Euler. Jika Anda membutuhkan sintaks kompleks dalam satu ekspresi, Anda dapat menyelubungi ekspresi tersebut dengan tanda kutip ganda (”...”). Penggunaan lebih dari satu ekspresi memang dimungkinkan, namun sangat tidak disarankan.

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

Sebagai pelengkap, perlu kami sampaikan bahwa ekspresi simbolik dapat digunakan dalam program, namun harus diselubungi dengan tanda kutip. Selain itu, akan jauh lebih efisien jika Maxima dipanggil saat waktu kompilasi, apabila memungkinkan.

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

Seperti sebelumnya, simbol % merujuk pada hasil sebelumnya.

Untuk mempermudah, kita menyimpan solusi tersebut ke dalam variabel simbolik. Variabel simbolik didefinisikan dengan "&=".

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

Ekspresi simbolik dapat digunakan di dalam ekspresi simbolik lainnya.

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

Masukan langsung perintah Maxima juga tersedia. Mulailah baris perintah dengan ":: ". Sintaks Maxima telah disesuaikan dengan sintaks EMT (disebut sebagai "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 dari Maxima. Anda dapat melakukannya dengan menggunakan ":::: ".

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

$$g^2$$

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

$$x^3\,E^x$$

Variabel semacam ini dapat digunakan dalam ekspresi simbolik lainnya. Perlu dicatat bahwa pada perintah berikut, sisi kanan dari &= akan dievaluasi terlebih dahulu sebelum penugasan ke Fx dilakukan.

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

```
                5
125 E
```

```
18551.64488782208
```

```
>fx(5)
```

```
  18551.6448878
```

Untuk mengevaluasi suatu ekspresi dengan nilai tertentu dari variabel-variabelnya, Anda dapat menggunakan operator "with".

Baris perintah berikut juga menunjukkan bahwa Maxima dapat mengevaluasi ekspresi secara numerik menggunakan float().

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

```
       10        5
1000 E   - 125 E
```

```
2.20079141499189e+7
```

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

Untuk mendapatkan kode LaTeX dari suatu 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(...) milik Maxima).

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

Penugasan juga dapat bersifat simbolik.

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

Perintah solve digunakan untuk menyelesaikan ekspresi simbolik terhadap suatu variabel dalam Maxima. Hasilnya berupa vektor solusi.

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

Bandingkan dengan perintah ”solve” numerik dalam 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 mengevaluasi hasil simbolik tersebut. Euler akan membaca penugasan seperti x = ... secara langsung. Jika Anda tidak memerlukan hasil numerik untuk perhitungan lanjutan, Anda juga dapat membiarkan Maxima menghitung nilai numeriknya.

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

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

Untuk mendapatkan solusi simbolik tertentu, Anda dapat menggunakan "with" dan indeks.

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

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

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

Ekspresi simbolik dapat memiliki flag, yang menunjukkan perlakuan khusus dalam Maxima. Beberapa flag dapat digunakan sebagai perintah, sementara yang lain tidak. Flag ditambahkan dengan tanda "|", yang merupakan bentuk yang lebih ringkas dari "ev(..., flags) ".

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

Fungsi

Dalam EMT, fungsi adalah program yang didefinisikan dengan perintah "function ". Fungsi dapat berupa satu baris atau terdiri dari beberapa baris.

Fungsi satu baris dapat bersifat numerik maupun simbolik. Fungsi satu baris numerik didefinisikan dengan ":= ".

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

Sebagai gambaran umum, kami tampilkan semua kemungkinan definisi untuk fungsi satu baris. Fungsi dapat dievaluasi seperti halnya fungsi bawaan Euler.

```
>f(2)
```

```
  4.472135955
```

Fungsi ini juga akan bekerja untuk vektor, mengikuti bahasa matriks dari Euler, karena ekspresi yang digunakan dalam fungsi telah tervektorisasi.

```
>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 sebuah string.

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

```
  0.786151377757
```

Secara default, jika Anda perlu menimpa fungsi bawaan, Anda harus menambahkan kata kunci "timpa". Menimpa fungsi bawaan berbahaya dan dapat menyebabkan masalah pada fungsi lain yang bergantung padanya.

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

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

```
  0.707106781187
```

Lebih baik kita menghapus definisi ulang sin.

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

```
  0.707106781187
```

Parameter Default

Fungsi numerik dapat memiliki parameter default.

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

Menghilangkan parameter ini menggunakan nilai default.

```
>f(4)
```

```
  16
```

Menyetelnya menimpa nilai default.

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

```
  80
```

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

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

```
  16
```

Jika sebuah variabel bukan merupakan 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 menimpa nilai global.

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

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

```
  20
```

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

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

Fungsi simbolik dapat digunakan dalam ekspresi simbolik.

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

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

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

```
  178.635099908
```

Mereka dapat digunakan untuk mendefinisikan fungsi atau ekspresi simbolik lainnya.

```
>function G(x) &= factor(integrate(g(x),x)); $&G(c) // integrate: mengintegralkan
>solve(&g(x),0.5)
```

```
  0.703467422498
```

Berikut ini juga berfungsi, 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)
```

```
incorrect syntax: P is not an infix operator
&P(
 ^
```

```
>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 &= fungsinya simbolis, dan dapat digunakan dalam ekspresi simbolik lainnya.

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

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

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

yang tidak dapat dievaluasi secara simbolis.

Jika kita mendefinisikan ulang fungsi dengan kata kunci ”peta”, fungsi tersebut dapat digunakan untuk vektor x. Secara internal, fungsi tersebut dipanggil untuk semua nilai x satu kali, dan hasilnya disimpan dalam sebuah 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 standar untuk parameter.

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

Sekarang fungsi tersebut 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
```

Seringkali, kita ingin menggunakan fungsi untuk vektor di satu tempat, dan untuk elemen individual 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)
```

Fungsi simbolik seperti itu dapat digunakan untuk variabel simbolik.

Tetapi fungsi tersebut juga dapat digunakan untuk vektor numerik.

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

```
  17
```

Ada juga fungsi simbolik 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)
```

Tapi 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.

Memecahkan Ekspresi

Ekspresi dapat diselesaikan secara numerik dan simbolis.

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

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

```
  1.41421356237
```

Ini juga berfungsi untuk ekspresi simbolik. 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
```

Sekarang kita mencari titik, di mana polinomial adalah 2. Dalam solve (), nilai sasaran default y=0 boleh ditukar dengan pembolehubah yang ditetapkan.

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

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

```
  0.966715594851
  2
```

Memecahkan ekspresi simbolik dalam bentuk simbolik mengembalikan 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 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])
```

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

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

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;
```

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

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

```
  2.54116291558
```

Ini juga berfungsi 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])
```

Bahasa Matriks

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
```

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 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 perkalian 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, operan tersebut diperluas secara alami.

```
>2*A
```

```
                2             4
                6             8
```

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

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

```
                1             4
                3             8
```

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

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

```
                2             6
                6            12
```

Perkalian ini dapat dibayangkan 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
```

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 transposenya. Bahasa matriks Euler secara otomatis menghasilkan tabel nilai.

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

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

Sekali lagi, ingatlah bahwa ini bukan produk matrix!

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

```
  55
```

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

```
  55
```

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

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

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

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

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

```
  5
```

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

Kita mendapatkan vektor 0 dan 1, dimana 1 adalah true.

```
>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 seperti itu, "nonzeros" 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 di t.

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

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

Sebagai contoh, mari kita cari semua kuadrat dari bilangan 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 perhitungan bilangan bulat. Ini menggunakan floating point presisi ganda secara internal. Namun, seringkali sangat berguna.

Kita bisa memeriksa keutamaan. Mari kita cari tahu, berapa kuadrat ditambah 1 bilangan prima.

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

```
 112
```

Fungsi nonzeros () 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 sebuah 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 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(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.

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

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

Fungsi Matriks Lainnya (Matriks Bangunan)

Untuk membangun sebuah matriks, kita dapat menumpuk satu matriks di atas matriks lainnya. Jika keduanya tidak memiliki jumlah kolom yang sama, 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 sisi 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 menginterpretasikan vektor ekspresi untuk vektor kolom.

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

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

Untuk mendapatkan ukuran A, kita dapat menggunakan fungsi-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 bilangan lain 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 (gau distribution distribusi).

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

```
        0.66566      0.831835
          0.977      0.544258
```

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

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

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

Dengan fungsi berikut, kita dapat menggunakan 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,  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]
```

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 dalam drop (v,i) mengacu pada indeks elemen dalam 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 yang diurutkan v.

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

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

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

```
>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.

```
>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 bisa mengekstrak diagonalnya.

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

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

Misalnya 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
```

Vektorisasi

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 ditampikan
```

Dengan ini dan operator titik dua a: delta: b, vektor nilai fungsi dapat dihasilkan dengan mudah.

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

$$s = t^3 - t$$

```
>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 diperluas 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 dengan produk matriks. Produk matriks dilambangkan dengan titik”.”dalam 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 transposisi. Matriks 1x1 dapat digunakan seperti bilangan real.

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

```
  5
  25
```

Untuk mengubah urutan matriks, kita menggunakan apostrof.

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

```
1
2
3
4
```

Jadi kita dapat menghitung matriks A dikalikan vektor b.

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

```
30
70
```

Perhatikan bahwa v masih berupa 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 Linier lainnya).

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

```
  30
```

Operator dan fungsi mematuhi bahasa matriks Euler.

Berikut adalah ringkasan aturannya.

-Fungsi yang diterapkan pada vektor atau matriks diterapkan pada setiap elemen.

-Operator yang beroperasi pada dua matriks dengan ukuran yang sama diterapkan berpasangan pada elemen matriks.

```
-Jika kedua matriks memiliki dimensi yang berbeda, keduanya diperluas
```

dengan cara yang masuk akal, sehingga memiliki ukuran yang sama.

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

Berikut ini adalah kasus sederhana dengan operator ^.

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

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

Ini kasus yang lebih rumit. Vektor baris dikalikan vektor kolom mengembang keduanya dengan menggandakan.

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

```
                1               2               3
                2               4               6
                3               6               9
```

Perhatikan bahwa perkalian skalar menggunakan perkalian matriks, bukan *!

```
>v.v'
```

```
  14
```

Ada banyak fungsi untuk matriks. Kami memberikan daftar singkat. Anda harus membaca 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 terdapat 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 tersebut. Untuk matriks, ini mengembalikan baris ke-i lengkap dari matriks.

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

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

Indeks juga dapat berupa vektor baris indeks. : menunjukkan semua indeks.

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

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

Bentuk singkat untuk : 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, 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 atur ulang ke format default, dan hitung tabel nilai sinus dan kosinus. Perhatikan bahwa sudut berada 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 membuat beberapa tabel dari beberapa fungsi sekaligus.

Dalam contoh berikut, kita menghitung t[j]^i untuk i dari 1 hingga n. Kita mendapatkan sebuah matriks, di mana setiap baris adalah tabel t^i untuk satu i. Yaitu, matriks tersebut memiliki unsur-unsurnya

$$a_{i,j} = t_j^i, \quad 1 \le j \le 101, \quad 1 \le i \le n$$

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

Integrasi numerik integrate () hanya berfungsi untuk batas interval skalar. Jadi kita perlu memvektorasinya.

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

Kata kunci ”peta” mengubah vektor fungsi. Fungsi sekarang akan berfungsi untuk vektor bilangan.

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

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

Sub-Matriks dan Elemen Matriks

Untuk mengakses elemen matriks, gunakan notasi tanda 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 seluruh baris dari sebuah matriks.

```
>A[2]
```

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

Dalam kasus vektor baris atau kolom, ini akan mengembalikan elemen dari vektor tersebut.

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

```
  2
```

Untuk memastikan Anda mendapatkan baris pertama dari matriks berukuran 1×n maupun m×n, tentukan semua kolom dengan menggunakan indeks kedua yang kosong.

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

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

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

Di sini kita menginginkan baris pertama dan kedua dari matriks A.

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

```
                1               2               3
                4               5               6
```

Kita bahkan dapat menyusun ulang A menggunakan vektor indeks. Secara tepat, kita tidak mengubah A, melainkan menghitung versi A yang telah disusun ulang.

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

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

Trik indeks juga berlaku untuk kolom.

Contoh berikut memilih semua baris dari A dan kolom kedua serta ketiga.

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

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

Sebagai bentuk singkatan, tanda ':' digunakan untuk mewakili 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 dapat mengambil baris terakhir dari matriks A.

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

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

Sekarang mari kita ubah elemen-elemen dari A dengan menetapkan submatriks A ke suatu nilai. Ini benar-benar mengubah matriks A yang tersimpan.

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

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

Kita juga dapat menetapkan nilai ke suatu baris dari matriks A.

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

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

Kita bahkan dapat menetapkan nilai ke suatu submatriks, asalkan ukurannya sesuai.

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

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

Selain itu, beberapa bentuk singkatan diperbolehkan.

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

```
0    0   -1
0    0    6
7    8    9
```

Peringatan: Indeks yang berada di luar batas akan menghasilkan matriks kosong atau pesan kesalahan, tergantung pada pengaturan sistem. Secara bawaan, sistem akan menampilkan pesan kesalahan. Namun, perlu diingat bahwa indeks negatif dapat digunakan untuk mengakses elemen matriks dari arah belakang.

```
>A[4]
```

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

Menyortir dan Mengacak

Fungsi sort() mengurutkan vektor baris.

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

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

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

Mari kita acak sebuah vektor.

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

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

Indeks-indeks tersebut memuat urutan yang benar dari vektor v.

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

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

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 bersifat agak acak.

```
>ind
```

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

Fungsi unique mengembalikan daftar elemen unik dari sebuah vektor yang telah diurutkan.

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

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

Ini juga berlaku untuk vektor string.

```
>unique(s)
```

```
a
aa
d
e
```

Aljabar Linier

EMT memiliki banyak fungsi untuk menyelesaikan sistem linear, sistem sparse, atau masalah regresi.

Untuk sistem linear Ax = b, Anda dapat menggunakan algoritma Gauss, matriks invers, atau pendekatan linear fit. Operator A\b menggunakan versi dari algoritma Gauss.

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

```
      -4
     4.5
```

Sebagai contoh lain, kita menghasilkan matriks berukuran 200×200 dan menghitung jumlah setiap barisnya. Kemudian kita menyelesaikan Ax = b menggunakan matriks invers. Kita mengukur galat sebagai deviasi maksimum dari semua elemen terhadap nilai 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, pendekatan linear fit akan meminimalkan norma dari galat 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
```

Matriks Simbolik

Maxima memiliki matriks simbolik. Tentu saja, Maxima dapat digunakan untuk masalah aljabar linier sederhana seperti itu. Kita dapat mendefinisikan matriks untuk Euler dan Maxima dengan&:=, lalu menggunakannya dalam ekspresi simbolik. Yang biasa [...] bentuk 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 vektor dengan dua vektor nilai eigen dan perkalian.

```
>$&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 with.

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

Akses ke baris matriks simbolik berfungsi seperti halnya dengan matriks numerik.

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

Ekspresi simbolik dapat berisi penugasan. 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.

```
>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 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 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 detailnya.

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

Nilai Numerik dalam Ekspresi simbolik

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

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

```
                1       3.14159
                4             5
```

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

```
>$&A
```

Untuk menghindarinya, ada fungsi "mxmset (variable)".

```
>mxmset(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 yang besar dapat diubah.

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

```
        3.14159265358979323846264338327950288419716939937510582097494\
 459230781640628620899862803482534211706 8b0
```

Variabel numerik dapat digunakan dalam ekspresi simbolik apa pun menggunakan ”@ var”.

Perhatikan bahwa ini hanya diperlukan, jika variabel telah didefinisikan dengan”: = ”atau” = ” sebagai variabel numerik.

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

Demo-Suku Bunga

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 sebesar 5.000 (katakanlah dalam dolar).

```
>K=5000
```

```
  5000
```

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

```
>K*1.03
```

```
  5150
```

Euler juga akan memahami sintaks berikut.

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

```
  5150
```

Tetapi lebih mudah menggunakan faktor tersebut

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

```
  1.03
  5150
```

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

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

```
  6719.58189672
```

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

```
>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 perlu 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 ke 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 kita mendapatkan vektor nilai.

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

Tentu saja, cara 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 mengulanginya selama bertahun-tahun. Euler memberikan banyak solusi untuk ini.

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

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

```
  Real 1 x 11 matrix

      5000.00      5150.00      5304.50      5463.64      ...
```

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

```
>VKr'
```

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

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

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

```
      5150.00
      5000.00     5150.00     5304.50
```

Ternyata, 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 penuhnya.

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

```
      6719.60
      6719.58
```

Perbedaannya sangat kecil.

Memecahkan Persamaan

Sekarang kita gunakan fungsi yang lebih lanjut yang menambahkan sejumlah tingkat pertumbuhan uang setiap tahun.

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

Kita tidak harus menentukan nilai q dan R saat mendefinisikan fungsi. Nilai-nilai tersebut hanya perlu ditentukan saat fungsi dijalankan. Di sini, kita 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      ...
```

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

Bagaimana kita bisa menentukan berapa tahun uang itu akan bertahan? Kita harus menulis satu lingkaran untuk ini. Cara termudah adalah dengan melakukan 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
```

Alasannya adalah 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. Ini dapat 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 secara numerik. Di bawah ini, kita akan mendapatkan rumus yang diperlukan. Maka Anda akan melihat bahwa tidak ada rumus yang mudah untuk suku bunga. Tetapi 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 kita tidak lagi menggunakan nilai global R dalam ekspresi kita. 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 indeksnya [-1].

Mari kita coba tes.

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

```
       -19.83
```

Sekarang kita bisa menyelesaikan masalah kita.

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

```
         3.15
```

Rutinitas penyelesaian memecahkan ekspresi=0 untuk variabel x. Jawabannya adalah 3,15% per tahun. Kita mengambil nilai awal 3% untuk algoritme. 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 untuk jumlah tahun, karena fungsi kita mengasumsikan n sebagai nilai bilangan bulat.

Solusi Simbolis untuk Masalah Suku Bunga

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

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

Kita sekarang dapat mengulangi ini.

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

Kita melihat sebuah pola. Setelah n periode kita memiliki

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

Rumusnya adalah rumus penjumlahan geometrik yang diketahui Maxima.

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

Ini agak rumit. Jumlah tersebut dievaluasi dengan tanda "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)
```

Fungsinya sama dengan fungsi kita f sebelumnya. Tapi itu lebih efektif.

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

```
-19.82504734650985
-19.82504734652684
```

Kita sekarang dapat menggunakannya untuk menanyakan waktu n. Kapan modal kita habis? Tebakan 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 simbolik Euler untuk menghitung rumus pembayaran.

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

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

Biasanya formula ini dinyatakan

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

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

Kita dapat menyelesaikan nilai R secara simbolik.

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

Seperti yang dapat Anda lihat dari rumusnya, fungsi ini menghasilkan galat bilangan pecahan (floating point error) saat i = 0. Namun, Euler tetap memplotnya. Tentu saja, kita memiliki batas berikut.

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

Jelas, tanpa bunga kita harus membayar kembali 10 kali cicilan sebesar 500.

Persamaan ini juga dapat diselesaikan terhadap variabel n. Hasilnya akan tampak lebih rapi jika kita menerapkan beberapa penyederhanaan.

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

Latihan Soal

Sederhanakan!

$$\left(\frac{24a^{10}b^{-8}c^{7}}{12a^{6}b^{-3}c^{5}}\right)^{-5}$$

```
>$factor((24*a^10*b^-8*c^7)/(12*a^6*b^-3*c^5))
```

$$\frac{2\,a^{4}\,c^{2}}{b^{5}}$$

Sederhanakan!

$$\left(\frac{125p^{12}q^{-14}r^{22}}{25p^{8}q^{6}r^{-15}}\right)^{-4}$$

```
>$factor((125*p^12*q^(-14)*r^22)/(25*p^8*q^6*r^(-15)))^(-4)
```

$$\frac{q^{80}}{16\;148}$$

```
625 p   r
```

Hitunglah!

$$2^6 \cdot 2^{-3} \div 2^{10} \div 2^{-8}$$

```
>2^6*2^(-3)/2^10/2^(-8)
```

```
2
```

Hitunglah!

$$\frac{4(8-6)^2 - 4 \cdot 3 + 2 \cdot 8}{3^1 + 19^0}$$

```
>(4*(8-6)^2 - 4*3 + 2*8) / (3^1 + 19^0)
```

```
5
```

Hitunglah!

$$\frac{[4(8-6)^2+4](3-2\cdot 8)}{2^2(2^3+5)}$$

```
>((4*(8-6)^2+4)(3-2*8))/2^2*(2^3+5)
```

```
apply: found 20 where a function was expected.
 -- an error. To debug this try: debugmode(true);
```

Operasikan!

$$(5x-3)^2$$

```
>expand((5*x-3)^2)
```

```
                2
25 x  - 30 x + 9
```

Operasikan!

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

```
>expand((3*x - 2*y)*(3*x + 2*y))
```

```
   2      2
9 x  - 4 y
```

Faktorkan!

$$16x^2 - 9$$

```
>$factor(16*x^2-9)
```

```
(4 x - 3) (4 x + 3)
```

Faktorkan!

$$x^2 + 12x + 36$$

```
>$factor(x^2+12*x+36)
```

```
          2
(x + 6)
```

Faktorkan!

$$x^3 + 64$$

```
>$factor(x^3+64)
```

```
                2
(x + 4) (x  - 4 x + 16)
```

Faktorkan!

$$18a^2b - 15ab^2$$

```
>$factor(18*a^2*b-15*a*a^2)
```

```
   2
3 a  (6 b - 5 a)
```

Faktorkan!

$$y^4 + 84 + 5y^2$$

```
>$factor(y^4+84+5*y^2)
```

```
  4      2
 y  + 5 y  + 84
```

$$f(x) = 3x + 1$$

$$g(x) = x^2 - 2x - 6$$

$$h(x) = x^3$$

```
>f(x):= 3*x + 1
```

```
f(x) := 3 x + 1
```

```
>g(x):=x^2-2*x-6
```

```
                2
g(x) := x  - 2 x - 6
```

```
>h(x):=x^3
```

```
                3
h(x) := x
```

$$f(g(x))$$

```
>expand(f(g(x)))
```

$$3\,x^2 - 6\,x - 17$$

$f(g(-1))$

```
>expand(f(g(-1)))
```

$$-8$$

$g(f(-2))$

```
>expand(g(f(-2)))
```

$$h(f(1))$$

```
>expand(h(f(1)))
```

```
64
```

$$g(h(1/2))$$

```
>expand(g(h(1/2)))
```

```
  399
- ---
  64
```

$$(-5+3i)+(7+8i)$$

```
>(-5+3*i) + (7+8*i)
```

```
11 i + 2
```

```
>(2+3*i) / (1-2*i)
```

```
3 i + 2
-------
1 - 2 i
```

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

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

```
[C:/Program Files/Euler x64/maxima/maxout.gnuplot,
                                    C:/Users/HP/Euler/gnuout.png]
```

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

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

```
[C:/Program Files/Euler x64/maxima/maxout.gnuplot,
```

C:/Users/HP/Euler/gnuout.png]

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

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

$$h(x) = 3x + 1$$

```
>plot2d([x^2, x^3, 3*x+1], [x, -3, 3])
```

[C:/Program Files/Euler x64/maxima/maxout.gnuplot,
C:/Users/HP/Euler/gnuout.png]

$$y = x^2 - 2x - 3$$

```
>plot2d(x^2 - 2*x - 3, [x, -5, 5])
```

[C:/Program Files/Euler x64/maxima/maxout.gnuplot,
C:/Users/HP/Euler/gnuout.png]

$$y = x^2$$

$$y = 2x + 3$$

```
>plot2d([x^2, 2*x+3], [x, -5, 5])
```

```
[C:/Program Files/Euler x64/maxima/maxout.gnuplot,
                                    C:/Users/HP/Euler/gnuout.png]
```