Aplikasi Komputer

NIM: 24030130053

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:

```
>$&6*x^(-3)*y^5*-7*x^2*y^(-9) //hasilnya -42/xy^4
```

Menjabarkan:

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

Baris Perintah

Baris perintah Euler terdiri dari satu atau beberapa perintah Euler yang diikuti titik koma ";" atau koma ",". Titik koma mencegah pencetakan hasil. Koma setelah perintah terakhir dapat dihilangkan.

Baris perintah berikut hanya akan mencetak hasil ekspresi, bukan penugasan atau perintah format.

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

```
  16.7551608191
```

Perintah harus dipisahkan dengan spasi. Baris perintah berikut akan mencetak dua hasilnya.

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

```
  50.2654824574
  100.530964915
```

Baris perintah dieksekusi sesuai urutan pengguna menekan tombol kembali. Jadi, Anda akan mendapatkan nilai baru setiap kali Anda mengeksekusi baris kedua.

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

```
  0.540302305868
```

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

```
  0.857553215846
```

Jika dua baris dihubungkan dengan "..." kedua baris akan selalu dieksekusi secara bersamaan.

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

```
  1.41666666667
  1.41421568627
  1.41421356237
```

Ini juga cara yang baik untuk membagi perintah panjang menjadi dua baris atau lebih. Anda dapat menekan Ctrl+Return untuk membagi baris menjadi dua pada posisi kursor saat ini, atau Ctrl+Back untuk menggabungkan baris-baris tersebut.

Untuk melipat semua baris yang memiliki banyak baris, tekan Ctrl+L. Baris-baris berikutnya hanya akan terlihat jika salah satunya menjadi fokus. Untuk melipat satu baris yang memiliki banyak baris, awali baris pertama dengan "%+".

```
>%+ x=4+5; ...
  // This line will not be visible once the cursor is off the line
```

Baris yang dimulai dengan %% akan sepenuhnya tidak terlihat.

```
>Tidak masalah menggunakan beberapa baris. Pastikan baris diakhiri dengan "...".%% x^2 // This line wil
```

```
  81
```

Euler mendukung perulangan dalam baris perintah, asalkan dapat digunakan dalam satu baris atau beberapa baris. Dalam program, batasan ini tentu saja tidak berlaku. Untuk informasi lebih lanjut, silakan lihat pendahuluan berikut.

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

```
  1.5
  1.41666666667
  1.41421568627
  1.41421356237
  1.41421356237
```

Tidak masalah menggunakan beberapa 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
```

Conditional structures do also work.

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

```
  Thought so!
```

Saat Anda menjalankan perintah, kursor dapat berada di posisi mana pun di baris perintah. Anda dapat kembali ke perintah sebelumnya atau melompat ke perintah berikutnya dengan tombol panah. Atau, Anda dapat mengeklik bagian komentar di atas perintah untuk membuka 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 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 untuk perintah terbaru, buka jendela bantuan dengan F1. Di sana, Anda dapat memasukkan teks yang ingin dicari. Pada baris kosong, bantuan untuk jendela bantuan akan ditampilkan. Anda dapat menekan tombol escape untuk menghapus baris tersebut, atau untuk menutup jendela bantuan.

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

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

```
  2.5
```

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

Sintaks Dasar

Euler menguasai fungsi-fungsi matematika umum. Seperti yang telah Anda lihat di atas, fungsi trigonometri bekerja dalam radian atau derajat. Untuk mengonversi ke derajat, tambahkan simbol derajat (dengan

tombol F7) ke nilai tersebut, atau gunakan fungsi rad(x). Fungsi akar kuadrat disebut sqrt dalam Euler. Tentu saja, x^(1/2) juga dimungkinkan.

Untuk mengatur variabel, gunakan "=" atau ":=". Demi kejelasan, pengantar ini menggunakan bentuk yang terakhir. Spasi tidak masalah. Namun, spasi antar perintah tetap diperlukan.

Beberapa perintah dalam satu baris dipisahkan dengan "," atau ";". Titik koma menghilangkan keluaran perintah. Di akhir baris perintah, tanda "," diasumsikan, jika ";" tidak ada.

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

```
  30.65625
```

EMT menggunakan sintaksis pemrograman untuk ekspresi. Untuk memasukkan

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

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

```
  8.77908249441
```

Untuk menghitung ekspresi rumit seperti

Anda perlu memasukkannya dalam bentuk baris.

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

```
  23.2671801626
```

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

Hasil perhitungan ini adalah angka floating point. Secara default, angka ini dicetak dengan akurasi sekitar 12 digit. Pada baris perintah berikut, kita juga akan mempelajari cara merujuk ke hasil sebelumnya dalam baris yang sama.

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

```
  0.47619047619
  10/21
```

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

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

```
  14.4978445072
```

Operator numerik Euler meliputi:

+ unary atau operator tambah
- unary atau operator kurang
*, /

perkalian matriks

pangkat a^b untuk a positif atau bilangan bulat b (a**b juga berfungsi)

n! operator faktorial

dan masih banyak lagi.

Berikut adalah beberapa fungsi yang mungkin Anda perlukan. Masih banyak lagi.

sin,cos,tan,atan,asin,acos,rad,deg

log,exp,log10,sqrt,logbase

bin,logbin,logfac,mod,floor,ceil,round,abs,sign

conj,re,im,arg,conj,real,complex

beta,betai,gamma,complexgamma,ellrf,ellf,ellrd,elle

bitand,bitor,bitxor,bitnot

Beberapa perintah memiliki alias, misalnya ln untuk log.

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

```
    2
    0.5
```

```
>sin(30°)
```

```
    0.5
```

Pastikan untuk menggunakan tanda kurung (kurung bulat) jika ada keraguan tentang urutan eksekusi! Berikut ini tidak sama dengan (2^3)^4, yang merupakan nilai default untuk 2^3^4 dalam EMT (beberapa sistem numerik melakukannya dengan cara lain).

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

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

Bilangan Riil

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 membutuhkan 8 byte.

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

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

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

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

String

Sebuah string dalam Euler didefinisikan dengan "...".

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

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

Strings can be concatenated with | or with +. This also works with numbers, which are converted to strings in that case.

```
>"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.
```

The print function does also convert a number to a string. It can take a number of digits and a number of places (0 for dense output), and optimally a unit.

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

```
    Golden Ratio : 1.61803
```

There is a special string none, which does not print. It is returned by some functions, when the result does not matter. (It is returned automatically, if the function does not have a return statement.)

```
>none
```

To convert a string to a number simply evaluate it. This works for expressions too (see below).

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

```
1234.5
```

To define a string vector, use the vector [...] notation.

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

```
affe
charlie
bravo
```

The empty string vector is denoted by [none]. String vectors can be concatenated.

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

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

Strings can contain Unicode characters. Internally, these strings contain UTF-8 code. To generate such a string, use u"..." and one of the HTML entities.

Unicode strings can be concatenated like other strings.

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

```
α = 45°
```

I

In comments, the same entities like α, β etc. can be used. This may be a quick alternative to Latex. (More details on comments below).

There are some functions to create or analyze unicode strings. The function strtochar() will recognize Unicode strings, and translate them correctly.

```
>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]
```

The result is a vector of Unicode numbers. The converse function is chartoutf().

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

```
Ü is a German letter
```

The function utf() can translate a string with entities in a variable into a Unicode string.

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

```
We have α=β.
```

It is also possible to use numerical entities.

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

```
Ähnliches
```

Boolean Values

Boolean values are represented with 1=true or 0=false in Euler. Strings can be compared, just like numbers.

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

```
0
```

```
1
```

"and" is the operator "&&" and "or" is the operator "||", as in the C language. (The words "and" and "or" can only be used in conditions for "if".)

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

```
1
```

Boolean operators obey the rules of the matrix language.

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

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

You can use the function nonzeros() to extract specific elements form a vector. In the example, we use the conditional 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

The default output format of EMT prints 12 digits. To make sure that we see the default, we reset the format.

```
>defformat; pi
```

```
3.14159265359
```

Internally, EMT uses the IEEE standard for double numbers with about 16 decimal digits. To see the full number of digits, use the command "longestformat", or we use the operator "longest" to display the result in the longest format.

```
>longest pi
```

```
3.141592653589793
```

Here is the internal hexadecimal representation of a double number.

```
>printhex(pi)
```

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

The output format can be changed permanently with a format command.

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

```
0.33333
3.14159
0.84147
```

The default is format(12).

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

```
0.333333333333
```

Functions like "shortestformat", "shortformat", "longformat" work for vectors in the following way.

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

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

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

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

```
    3.1416
```

The function "longestformat" set the scalar format too.

```
>longestformat; pi
```

```
    3.141592653589793
```

For reference, here is a list of the most important output formats.

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

The internal accuracy of EMT is about 16 decimal places, which is the IEEE standard. Numbers are stored in this internal format.

But the output format of EMT can be set in a flexible way.

```
>longestformat; pi,
```

```
    3.141592653589793
```

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

```
      3.14159
```

The default is defformat().

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

There are short operators which print only one value. The operator "longest" will print all valid digits of a number.

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

```
          4.934802200544679
```

There is also a short operator for printing a result in fractional format. We have already used it above.

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

```
    25/12
```

Since the internal format uses a binary way to store numbers, the value 0.1 will not be represented exactly. The error adds up a bit, as you see in the following computation.

```
>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
```

But with the default "longformat" you will not notice this. For convenience, the output of very small numbers is 0.

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

```
    0
```

Expressions

Strings or names can be used to store mathematical expressions, which can be evaluated by EMT. For this, use parentheses after the expression. If you intend to use a string as an expression, use the convention to name it "fx" or "fxy" etc. Expressions take precedence over functions.

Global variables can be used in the evaluation.

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

```
            12.56637061435917
```

Parameters are assigned to x, y, and z in that order. Additional parameters can be added using assigned parameters.

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

```
     -0.919535764538
```

Note that expression will always use global variables, even if there is a variable in a function with the same name. (Otherwise the evaluation of expressions in functions could have very confusing results for the user that called the function.)

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

```
     36
```

If you want to use another value for "at" than the global value you need to add "at=value".

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

```
     45
```

For reference, we remark that call collections (discussed elsewhere) can contain expressions. So we can make the above example as follows.

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

```
     45
```

Expressions in x are often used just like functions.
Note that defining a function with the same name like a global symbolic expression deletes this variable to avoid confusion between symbolic expressions and functions.

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

```
     12
```

By way of convention, symbolic or numerical expressions should be named fx, fxy etc. This naming scheme should not be used for functions.

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

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

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

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

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

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

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

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

```
    -0.475
```

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

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

```
    -0.425
```

Suatu ekspresi tidak harus simbolis. Hal ini diperlukan jika ekspresi tersebut mengandung fungsi yang hanya diketahui dalam kernel numerik, bukan dalam Maxima.

Matematika Simbolik

EMT melakukan matematika simbolik dengan bantuan Maxima. Untuk detailnya, mulailah dengan tutorial berikut, atau telusuri referensi untuk Maxima. Para ahli Maxima perlu memperhatikan bahwa terdapat perbedaan sintaksis antara sintaksis asli Maxima dan sintaksis standar ekspresi simbolik dalam EMT.

Matematika simbolik terintegrasi 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 terhingga" yang dapat menangani bilangan yang sangat besar.

```
>$&44!
```

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

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

Of course, Maxima has a more efficient function for this (as does the numerical part of EMT).

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

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

Anda akan mempelajari bahwa perintah berikut juga berfungsi.

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

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

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

Dengan cara ini, Anda dapat menggunakan solusi suatu persamaan dalam persamaan lain.

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

Seperti yang telah Anda lihat pada contoh sebelumnya dan selanjutnya, jika Anda telah menginstal LaTeX, Anda dapat mencetak ekspresi simbolik dengan Latex. Jika tidak, perintah berikut akan menampilkan pesan kesalahan.

Untuk mencetak ekspresi simbolik dengan LaTeX, gunakan $ di depan & (atau Anda dapat menghilangkan &) sebelum perintah. Jangan jalankan perintah Maxima dengan $ jika Anda belum menginstal LaTeX.

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

Ekspresi simbolik diurai oleh Euler. Jika Anda membutuhkan sintaksis yang kompleks dalam satu ekspresi, Anda dapat melampirkan ekspresi tersebut dalam "...". Penggunaan lebih dari satu ekspresi sederhana dimungkinkan, tetapi sangat tidak disarankan.

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


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

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

Again, % refers to the previous result.

To make things easier we save the solution to a symbolic variable. Symbolic variables are defined with "&=".

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

Symbolic expressions can be used in other symbolic expressions.

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

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

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

$$2432902008176640000$$

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

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

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

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

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

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

$$g^2$$

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

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

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

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

$$125\,E^5$$

$$18551.64488782208$$

```
>fx(5)
```

18551.6448878

Untuk mengevaluasi ekspresi dengan nilai variabel tertentu, Anda dapat menggunakan operator "with".

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

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

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

```
2.20079141499189e+7
```

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

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

```
>tex(fx)
```

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

Symbolic expressions can be evaluated just like numerical expressions.

```
>fx(0.5)
```

```
1.41176470588
```

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

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

The assignment can also be symbolic.

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

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

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

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

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

```
1.56155281281
```

Nilai numerik dari solusi simbolik dapat dihitung dengan mengevaluasi hasil simbolik. Euler akan membaca nilai x= dst. Jika Anda tidak memerlukan hasil numerik untuk perhitungan lebih lanjut, Anda juga dapat membiarkan Maxima mencari nilai numeriknya.

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

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

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

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

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

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

Ekspresi simbolik dapat memiliki tanda (flag), yang menunjukkan perlakuan khusus di Maxima. Beberapa tanda juga dapat digunakan sebagai perintah, sementara yang lain tidak. Tanda 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(%)
```

Fungsi

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

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

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

```
>f(2)
```

```
    4.472135955
```

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

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

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

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

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

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

```
    0.786151377757
```

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

Anda masih dapat memanggil fungsi bawaan sebagai "_...", jika fungsinya berada di inti Euler.

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

```
    0.707106781187
```

We better remove this redefinition of sin.

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

```
    0.707106781187
```

Parameter Default

Fungsi numerik dapat memiliki parameter default.

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

Omitting this parameter uses the default value.

```
>f(4)
```

```
    16
```

Setting it overwrites the default value.

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

```
    80
```

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

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

```
    16
```

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

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

```
    24
```

But an assigned parameter overrides the global value.

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

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

```
  20
```

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

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

Symbolic functions can be used in symbolic expressions.

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

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

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

```
  178.635099908
```

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

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

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

```
  0.703467422498
```

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

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

```
  0.703467422498
```

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

```
  625
```

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

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

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

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

yang tidak dapat dievaluasi secara simbolis.

Jika kita mendefinisikan ulang fungsi tersebut dengan kata kunci "map", fungsi tersebut dapat digunakan untuk vektor x. Secara internal, fungsi tersebut dipanggil untuk semua nilai x sekali, 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]
```

Functions can have default values for parameters.

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

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

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

```
     2
     6.7
```

Moreover, it is possible to use assigned parameters.

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

```
     2
```

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

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

Such a symbolic function can be used for symbolic variables.

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

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

```
     17
```

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

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

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

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

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

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

Singkatnya,

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

Menyelesaikan Ekspresi

Ekspresi dapat diselesaikan secara numerik dan simbolik.

Untuk menyelesaikan ekspresi sederhana dengan satu variabel, kita dapat menggunakan fungsi solve(). Fungsi ini membutuhkan nilai awal untuk memulai pencarian. Secara internal, solve() menggunakan metode sekan.

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

```
     1.41421356237
```

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

```
>$&solve(x^2=2,x)
>$&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 bernilai 2. Dalam solve(), nilai target default y=0 dapat diubah dengan variabel yang ditetapkan.
Kita menggunakan y=2 dan memeriksa dengan mengevaluasi polinomial pada hasil sebelumnya.

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

```
     0.966715594851
     2
```

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

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

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

```
>longest sol()
```

```
        -0.6180339887498949       1.618033988749895
```

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

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

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

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

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

with a=3.

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

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

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

```
     2.54116291558
```

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

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

```
     2.54116291558
```

Menyelesaikan Pertidaksamaan

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

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

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

```
>$&fourier_elim([x^2 - 1>0],[x]) // x^2-1 > 0
>$&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 pembahasan mendetail tentang bahasa matriks Euler.

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

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

```
                1              2
                3              4
```

The matrix product is denoted by a dot.

```
>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
```

The main point of a matrix language is that all functions and operators work element for element.

```
>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
```

This is not the matrix product, but a multiplication element by element. The same works for vectors.

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

                9
               16
```

If one of the operands is a vector or a scalar it is expanded in the natural way.

```
>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
```

If it is a row vector it is applied to all columns of A.

```
>A*[2,3]

                2             6
                6            12
```

One can imagine this multiplication as if the row vector v had been duplicated to form a matrix of the same size as 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, yang satu merupakan vektor baris dan yang lainnya merupakan vektor kolom. Kita menghitung i*j untuk i,j dari 1 hingga 5. Triknya adalah 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
```

Again, remember that this is not the matrix product!

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

```
     55
```

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

```
     55
```

Even operators like < or == work in the same way.

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

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

E.g., we can count the number of elements satisfying a certain condition with the function sum().

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

```
     5
```

Euler has comparison operators, like "==", which checks for equality.

We get a vector of 0 and 1, where 1 stands for 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]
```

From such a vector, "nonzeros" selects the non-zero elements.

In this case, we get the indices of all elements greater than 50.

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

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

Of course, we can use this vector of indices to get the corresponding values in t.

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

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

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

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

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

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

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

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

```
  112
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fungsi Matriks Lainnya (Membangun Matriks)

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

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

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

Likewise, we can attach a matrix to another side by side, if both have the same number of rows.

```
>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
```

If they do not have the same number of rows the shorter matrix is filled with 0.

There is an exception to this rule. A real number attached to a matrix will be used as a column filled with that real number.

```
>A|1
```

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

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

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

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

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

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

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

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

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

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

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

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

For vectors, there is length().

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

```
    9
```

There are many other functions, which generate matrices.

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

```
                1              1
                1              1
```

This can also be used with one parameter. To get a vector with another number than 1, use the following.

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

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

Also a matrix of random numbers can be generated with random (uniform distribution) or normal (Gauß distribution).

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

```
           0.66566       0.831835
             0.977       0.544258
```

Here is another useful function, which restructures the elements of a matrix into another matrix.

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

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

With the following function, we can use this and the dup function to write a rep() function, which repeats a vector n times.

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

Let us test.

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

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

The function multdup() duplicates elements of a vector.

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

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

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

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

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

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

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

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

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

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

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

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

```
>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]
```

As you see, it does not harm to include indices out of range (like 0), double indices, or unsorted indices.

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

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

There are some special functions to set diagonals or to generate a diagonal matrix.

We start with the identity matrix.

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

```
1  0  0  0  0
0  1  0  0  0
0  0  1  0  0
0  0  0  1  0
0  0  0  0  1
```

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

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

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

Note that we did not change the matrix A. We get a new matrix as result of setdiag().

Here is a function, which returns a tri-diagonal matrix.

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

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

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

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

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

Now we can extract the diagonal.

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

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

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

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

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

Vektorisasi

Hampir semua fungsi di Euler juga berfungsi untuk input matriks dan vektor, jika diperlukan.

Misalnya, fungsi sqrt() menghitung akar kuadrat dari semua elemen vektor atau matriks.

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

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

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

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

```
>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 expands operators for scalars, vectors, and matrices in the obvious way.

E.g., a column vector times a row vector expands to matrix, if an an operator is applied. In the following, v' is the transposed vector (a column vector).

```
>shortest (1:5)*(1:5)'
```

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

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

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

```
55
```

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

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

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

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

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

```
5
25
```

To transpose a matrix we use the apostrophe.

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

```
1
2
3
4
```

So we can compute matrix A times vector b.

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

```
30
70
```

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

```
>v'.v
```

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

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

```
>v.v'
```

```
    30
```

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

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

```
    30
```

Operator dan fungsi mematuhi bahasa matriks Euler.

Berikut ringkasan aturannya.

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

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

Berikut adalah kasus sederhana dengan operator ^.

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

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

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

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

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

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

```
>v.v'
```

```
    14
```

Ada banyak fungsi untuk matriks. Kami memberikan daftar singkatnya. Anda sebaiknya merujuk ke dokumentasi untuk informasi lebih lanjut tentang perintah-perintah ini.

sum,prod menghitung jumlah dan hasil kali baris-baris
cumsum,cumprod melakukan hal yang sama secara kumulatif

menghitung nilai ekstrem setiap baris
extrema mengembalikan vektor dengan informasi ekstrem
diag(A,i) mengembalikan diagonal ke-i
setdiag(A,i,v) menetapkan diagonal ke-i
id(n) matriks identitas
det(A) determinan
charpoly(A) polinomial karakteristik
eigenvalues(A) nilai eigen

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

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

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

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

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

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

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

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

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

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

```
  6
```

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

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

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

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

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

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

A short form for : is omitting the index completely.

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

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

For purposes of vectorization, the elements of a matrix can be accessed as if they were vectors.

```
>A{4}
```

```
  4
```

A matrix can also be flattened, using the redim() function. This is implemented in the function 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]
```

To use matrices for tables, let us reset to the default format, and compute a table of sine and cosine values. Note that angles are in radians by default.

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

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

```
  315
  360
```

Now we append columns to a matrix.

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

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

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

Dalam contoh berikut, kita menghitung t[j]^i untuk i dari 1 hingga n. Kita mendapatkan sebuah matriks, di mana setiap barisnya merupakan tabel t^i untuk satu i. Dengan kata lain, matriks tersebut memiliki elemen-elemen lateks: 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 "divektorkan". Hal ini dapat dicapai dengan kata kunci "map" 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 memvektorkannya.

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

The "map" keyword vectorizes the function. The function will now work for vectors of numbers.

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

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

Sub-Matrices and Matrix-Elements

To access a matrix element, use the bracket notation.

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

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

We can access a complete line of a matrix.

```
>A[2]
```

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

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

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

```
  2
```

To make sure, you get the first row for a 1xn and a mxn matrix, specify all columns using an empty second index.

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

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

If the index is a vector of indices, Euler will return the corresponding rows of the matrix.

Here we want the first and second row of A.

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

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

We can even reorder A using vectors of indices. To be precise, we do not change A here, but compute a reordered version of A.

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

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

The index trick works with columns too.

This example selects all rows of A and the second and third column.

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

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

For abbreviation ":" denotes all row or column indices.

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

```
                3
                6
                9
```

Alternatively, leave the first index empty.

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

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

We can also get the last line of A.

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

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

Now let us change elements of A by assigning a submatrix of A to some value. This does in fact change the stored matrix A.

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

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

We can also assign a value to a row of A.

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

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

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

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

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

Moreover, some shortcuts are allowed.

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

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

Peringatan: Indeks yang melebihi batas akan menghasilkan matriks kosong, atau pesan kesalahan, tergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Namun, perlu diingat bahwa indeks negatif dapat digunakan untuk mengakses elemen matriks yang dihitung dari akhir.

```
>A[4]
```

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

Sorting and Shuffling

The function sort() sorts a row vector.

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

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

It is often necessary to know the indices of the sorted vector in the original vector. This can be used to reorder another vector in the same way.

Let us shuffle a vector.

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

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

The indices contain the proper order of v.

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

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

This works for string vectors too.

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

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

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

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

As you see, the position of double entries is somewhat random.

```
>ind
```

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

The function unique returns a sorted list of unique elements of a vector.

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

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

This works for string vectors too.

```
>unique(s)
```

```
     a
     aa
     d
     e
```

Aljabar Linear

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 pencocokan linear. Operator A\b menggunakan versi algoritma Gauss.

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

```
               -4
              4.5
```

Untuk contoh lain, kita buat matriks 200x200 dan jumlah barisnya. Kemudian, kita selesaikan Ax=b menggunakan matriks invers. Kita ukur galat sebagai deviasi maksimum semua elemen dari 1, yang tentu saja merupakan solusi yang tepat.

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

```
       8.790745908981989e-13
```

If the system does not have a solution, a linear fit minimizes the norm of the error Ax-b.

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

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

The determinant of this matrix is 0.

```
>det(A)
```

```
     0
```

Symbolic Matrices

Maxima has symbolic matrices. Of course, Maxima can be used for such simple linear algebra problems. We can define the matrix for Euler and Maxima with &:=, and then use it in symbolic expressions. The usual [...] form to define matrices can be used in Euler to define symbolic matrices.

```
>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
```

Like all symbolic variables, these matrices can be used in other symbolic expressions.

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

The eigenvalues can also be computed automatically. The result is a vector with two vectors of eigenvalues and multiplicities.

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

To extract a specific eigenvector needs careful indexing.

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

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

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

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

```
                1             4
                5             2
```

In symbolic expressions, use with.

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

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

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

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

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

There are symbolic functions in Maxima to create vectors and matrices. For this, refer to the documentation of Maxima or to the tutorial about Maxima in EMT.

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

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

The result can be evaluated numerically in Euler. For more information about Maxima, see the introduction to Maxima.

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

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

Euler juga memiliki fungsi xinv() yang canggih, yang membutuhkan upaya lebih besar dan menghasilkan 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
```

E.g. the eigenvalues of A can be computed numerically.

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

```
    [16.1168,  -1.11684,  0]
```

Or symbolically. See the tutorial about Maxima for details on this.

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

Nilai Numerik dalam Ekspresi Simbolik

Ekspresi simbolik hanyalah 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 terdapat perbedaan antara bentuk numerik dan bentuk simbolik. Saat mengubah matriks ke bentuk simbolik, pendekatan pecahan untuk bilangan riil akan digunakan.

```
>$&A
```

To avoid this, there is the function "mxmset(variable)".

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

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

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

The precision of the big floating point numbers can be changed.

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

```
            3.14159265358979323846264338327950288419716939937510582097494\
    459230781640628620899862803482534211706 8b0
```

A numerical variable can be used in any symbolic expressions using "@var".

Note that this is only necessary, if the variable has been defined with ":=" or "=" as a numerical variable.

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

Demo - Suku Bunga

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

Asumsikan Anda memiliki modal awal sebesar 5000 (misalnya dalam dolar).

```
>K=5000
```

```
    5000
```

Now we assume an interest rate of 3% per year. Let us add one simple rate and compute the result.

```
>K*1.03
```

```
    5150
```

Euler would understand the following syntax too.

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

```
    5150
```

But it is easier to use the factor

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

```
    1.03
    5150
```

For 10 years, we can simply multiply the factors and get the final value with compound interest rates.

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

```
    6719.58189672
```

For our purposes, we can set the format to 2 digits after the decimal dot.

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

```
        6719.58
```

Let us print that rounded to 2 digits in a complete sentence.

```
>"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 ke-9? Untuk itu, bahasa matriks Euler sangat membantu. Anda tidak perlu menulis perulangan, cukup masukkan

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

```
Real 1 x 11 matrix

    5000.00     5150.00     5304.50     5463.64     ...
```

How does this miracle work? First the expression 0:10 returns a vector of integers.

```
>short 0:10


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

Then all operators and functions in Euler can be applied to vectors element for element. So

```
>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]
```

is a vector of factors q^0 to q^10. This is multiplied by K, and we get the vector of values.

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


    Variable q not found!
    Error in ^
    Error in:
    VK=K*q^(0:10); ...
                ^
```

Of course, the realistic way to compute these interest rates would be to round to the nearest cent after each year. Let us add a function for this.

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

Let us compare the two results, with and without rounding.

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


    Variable q not found!
    Use global or local variables defined in function oneyear.
    Try "trace errors" to inspect local variables after errors.
    oneyear:
        useglobal; return round(K*q,2)
    Error in:
    longest oneyear(1234.57), longest 1234.57*q ...
                              ^
```

Now there is no simple formula for the n-th year, and we must loop over the years. Euler provides many solutions for this.

The easiest way is the function iterate, which iterates a given function a number of times.

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


    Variable q not found!
    Use global or local variables defined in function oneyear.
    oneyear:
        useglobal; return round(K*q,2)
    niterate:
        x=f$(x,args());
    Try "trace errors" to inspect local variables after errors.
    iterate:
        return niterate(f$,x,n,till;args());
```

We can print that in a friendly way, using our format with fixed decimal places.

```
>VKr'


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

To get a specific element of the vector, we use indices in square brackets.

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

```
         5150.00
         5000.00     5150.00     5304.50
```

Surprisingly, we can also use a vector of indices. Remember that 1:3 produced the vector [1,2,3].

Let us compare the last element of the rounded values with the full values.

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

```
         6719.60
         6719.58
```

Selisihnya sangat kecil.

Menyelesaikan Persamaan

Sekarang kita ambil fungsi yang lebih maju, yang menambahkan tingkat uang tertentu setiap tahun.

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

We do not have to specify q or R for the definition of the function. Only if we run the command, we have to define these values. We select R=200.

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

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

What if we remove the same amount each year?

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

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

Kita melihat uangnya berkurang. Jelas, jika kita hanya mendapatkan bunga 150 di tahun pertama, tetapi mengurangi 200, kita kehilangan uang setiap tahun.

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

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

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

Using the matrix language, we can determine the first negative value in the following way.

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

```
           48.00
```

Alasannya adalah nonzeros(VKR<0) mengembalikan vektor indeks i, dengan 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 mengambil kondisi akhir sebagai argumen. Kemudian, fungsi ini 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 0 setelah 50 tahun. Berapa tingkat bunganya?

Ini adalah pertanyaan yang hanya dapat dijawab secara numerik. Di bawah ini, kita akan mendapatkan rumus-rumus yang diperlukan. Kemudian Anda akan melihat bahwa tidak ada rumus yang mudah untuk tingkat bunga. Namun untuk saat ini, kita akan mencari solusi numerik.

Langkah pertama adalah mendefinisikan fungsi yang melakukan iterasi sebanyak n kali. Kita tambahkan 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.

Namun, 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, kita hanya tertarik pada nilai terakhir. Jadi, kita mengambil indeks [-1].

Mari kita coba uji coba.

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

```
          -19.83
```

Now we can solve our problem.

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

```
            3.15
```

Rutin solve menyelesaikan ekspresi = 0 untuk variabel x. Jawabannya adalah 3,15% per tahun. Kita mengambil nilai awal 3% untuk algoritma tersebut. Fungsi solve() selalu membutuhkan nilai awal.

Kita dapat menggunakan fungsi yang sama untuk menyelesaikan pertanyaan berikut: Berapa banyak yang dapat kita hapus 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 masalah jumlah tahun, karena fungsi kita mengasumsikan n sebagai nilai integer.

Solusi Simbolis untuk Masalah Suku Bunga

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

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

We can now iterate this.

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

We see a pattern. After n periods we have

The formula is the formula for the geometric sum, which is known to Maxima.

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

This is a bit tricky. The sum is evaluated with the flag "simpsum" to reduce it to the quotient.

Let us make a function for this.

```
>function fs(K,R,P,n) &= (1+P/100)^n*K + ((1+P/100)^n-1)/(P/100)*R; $&fs(K,R,P,n)
```

The function does the same as our function f before. But it is more effective.

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

```
Function f needs only 1 arguments (got 4)!
Use: map f (x)
Error in map.
Error in:
longest f(5000,-200,3,47), longest fs(5000,-200,3,47) ...
                           ^
```

We can now use it to ask for the time n. When is our capital exhausted? Our initial guess is 30 years.

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

```
     20.5061016552
```

Jawaban ini menyatakan bahwa hasilnya akan negatif setelah 21 tahun.

Kita juga dapat menggunakan sisi simbolis Euler untuk menghitung rumus pembayaran.

Asumsikan kita mendapatkan pinjaman sebesar K, dan membayar n kali cicilan sebesar R (dimulai setelah tahun pertama) sehingga menyisakan utang sebesar Kn (pada saat pembayaran terakhir). Rumus untuk hal ini jelas

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

Biasanya rumus ini diberikan dalam bentuk

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

We can solve for the rate R symbolically.

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

Seperti yang Anda lihat dari rumus, fungsi ini mengembalikan kesalahan floating point untuk i=0. Namun, Euler memplotnya.

Tentu saja, kita memiliki limit berikut.

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

Jelas, tanpa bunga, kita harus membayar kembali 10 kali lipat dari 500.

Persamaan ini juga dapat diselesaikan untuk n. Akan terlihat lebih baik jika kita menyederhanakannya.

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

Penyelesaian Soal Aljabar

1. Sederhanakan bentuk eksponen berikut
//$&((24*a^10*b^(-8)*c^7)/(12*a^6*b^(-3)*c^5))^(-5)$

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

2. Sederhanakan bentuk eksponen berikut

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

Penyelesaian:

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

3. Hitunglah

//$&(4*(8-6)^2-4*3+2*8)/(3^1 + 19^0)

Penyelesaian:

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

4. Hitunglah

//$&((4*(8 - 6)^2 + 4)*(3 - 2 *8))/(2^2*(2^3 + 5))

Penyelesaian :

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

5. Hitunglah

//$&(3*x^2 - 2*x - x^3 + 2) - (5*x^2 - 8*x - x^3 + 4)

Penyelesaian :

```
>$&(3*x^2 - 2*x - x^3 + 2) - (5*x^2 - 8*x - x^3 + 4)
```

6. Menyelesaikan pemfaktoran

//$&factor(t^2 + 8*t + 15)

Penyelesaian :

```
>$&factor(t^2 + 8*t + 15)
```

7. Menyelesaikan operasi yang di tunjukkan

//$&(2*x + 3*y + z - 7) + (4*x - 2*y - z + 8) + (-3*x + y - 2*z - 4)

Penyelesaian :

```
>$&(2*x + 3*y + z - 7) + (4*x - 2*y - z + 8) + (-3*x + y - 2*z - 4)
```

8. Hitunglah :

//$&factor(x^3 - 4*x^2 + 5*x - 20)

Penyelesaian :
(4*x - 2*y - z + 8) + (-3*x + y - 2*z - 4)

Penyelesaian :

```
>$&factor(x^3 - 4*x^2 + 5*x - 20)
```

9. Faktorkan

//$&factor(250*z^4 - 2*z)

Penyelesaian :
lesaian :
(4*x - 2*y - z + 8) + (-3*x + y - 2*z - 4)

Penyelesaian :

```
>$&factor(250*z^4 - 2*z)
```

10. Faktorkan :

//$&(18*a^2*b - 15*a*b^2)

Penyelesaian :

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

11. Asumsikan bahwa semua eksponen adalah bilangan asli. Kalikan

//$&(a^n + b^n)(a^n - b^n)

Penyelesaian :

```
>$&expand(a^n + b^n)*(a^n - b^n)
```

12. Asumsikan bahwa semua eksponen adalah bilangan asli. Kalikan

//$&(x^(3*m) - t^(5*n))^2)

Penyelesaian :

```
>$&expand((x^(3*m)-t^(5*n))^2)
```