EMT_PKN3_CHOI

CHOIRUL ULY NUCHA
24030130023
PENDIDIKAN MATEMATIKA C

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)
```

$$-\frac{42}{x\,y^4}$$

Menjabarkan:

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

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

Baris Perintah

Satu baris perintah di Euler terdiri dari satu atau beberapa perintah Euler yang diakhiri dengan titik koma ";" atau koma ",". Tanda titik koma mencegah hasil ditampilkan hasilnya. Koma setelah perintah terakhir boleh dihilangkan.

Baris perintah berikut hanya akan mencetak hasil dari 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 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 saat pengguna menekan tombol kembali. Jadi Anda 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 "..." maka keduanya 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 atau lebih baris. Anda dapat menekan Ctrl+Return untuk membagi satu baris menjadi dua pada posisi kursor saat ini, atau Ctrl+Back untuk menyatukan kembali baris-baris tersebut.

Untuk melipat semua multi-baris tekan Ctrl+L. Maka baris-baris berikutnya hanya akan terlihat jika salah satunya memiliki fokus. Untuk melipat satu multi-baris, awali baris pertamanya dengan "%+ ".

```
>%+ x=4+5; ...
  // Ini baris akan not terlihat once cursor adalah off baris
```

Baris yang diawali dengan %% akan sepenuhnya tidak terlihat.

```
  81
```

Euler mendukung perulangan (loop) di baris perintah, selama masih muat dalam satu baris tunggal atau multi-baris. Dalam program, batasan ini tentu tidak berlaku, tentu saja. Untuk informasi lebih lanjutperhatikan 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
```

Boleh menggunakan multi-baris. Pastikan baris diakhiri dengan " ...".

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

```
      1.41421356237
```

Struktur bersyarat juga berfungsi.

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

```
      Thought so!
```

Saat Anda mengeksekusi sebuah perintah, kursor bisa berada di posisi mana pun pada 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 perintah tersebut.

Saat Anda menggerakkan kursor 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. Eksekusi perintah dengan tombol return.

```
>sqrt(sin(10°)/cos(20°))
```

```
      0.429875017772
```

Untuk melihat bantuan perintah terbaru, buka jendela bantuan dengan F1. Di sana, Anda bisa mengetik teks untuk mencari. Pada baris kosong, bantuan untuk jendela bantuan akan ditampilkan. Anda dapat menekan escape untuk menghapus baris, atau untuk menutup jendela bantuan.

Anda dapat klik ganda pada perintah apa saja untuk membuka bantuan perintah tersebut. Coba klik ganda perintah exp di bawah ini pada baris perintah.

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

```
      2.5
```

Anda dapat menyalin dan menempelkan teks di dalam Euler juga. Gunakan Ctrl-C dan Ctrl-V untuk melakukan ini. Tandai teks, seret dengan mouse, atau gunakan tombol Shift bersama dengan tombol kursor. Selain itu, Anda juga dapat menyalin tanda kurung yang disorot.

Sintaks Dasar

Euler memiliki fungsi-fungsi matematika umum. Seperti yang Anda lihat di atas, fungsi trigonometri bekerja dalam radian atau derajat. Untuk mengonversi ke derajat, tambahkan simbol derajat (dengan tombol F7) pada nilainya, atau gunakan fungsi rad(x). Fungsi akar kuadrat disebut sqrt di Euler. Tentu saja, x^(1/2) juga memungkinkan.

Untuk menetapkan variabel, gunakan tanda "=" atau ":=". Demi kejelasan, pengantar ini menggunakan bentuk kedua. Spasi tidak berpengaruh. Namun spasi antar perintah diharapkan ada.

Beberapa perintah dalam satu baris dipisahkan dengan "," atau ";". Tanda titik koma menekan penampilan keluaran perintah. Di akhir baris perintah, "," diasumsikan jika ";" tidak ada.

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

```
  30.65625
```

EMT menggunakan programming syntax untuk expressions. enter

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

Anda telah membuat tanda kurung dengan benar dan gunakan simbol / untuk pecahan. Perhatikan tanda kurung yang disorot sebagai bantuan. Perlu dicatat bahwa konstanta Euler e dinamai dengan huruf E dalam EMT.

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

```
  8.77908249441
```

Hitung ekspresi yang rumit seperti

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

Anda perlu menekan enter dalam baris form.

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

```
  23.2671801626
```

Letakkan tanda kurung dengan hati-hati di sekitar sub-ekspresi yang perlu dihitung terlebih dahulu. EMT membantu Anda dengan menyoroti ekspresi hingga tanda kurung penutupnya selesai. Anda juga harus memasukkan nama pi untuk huruf Yunani p.

Hasil perhitungan ini adalah bilangan floating point. Secara bawaan, hasil tersebut ditampilkan dengan ketelitian sekitar 12 digit. Pada baris perintah berikut, kita juga akan mempelajari bagaimana cara merujuk hasil sebelumnya dalam baris yang sama

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

```
  0.47619047619
  10/21
```

Perintah Euler dapat berupa ekspresi atau perintah primitif. Ekspresi dibuat dari operator dan fungsi. Jika diperlukan, ekspresi harus diberi tanda kurung untuk memaksa urutan eksekusi yang benar. Jika ragu, memberikan tanda kurung adalah ide yang baik. Perlu dicatat bahwa EMT menampilkan tanda kurung pembuka dan penutup saat Anda mengedit baris perintah.

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

```
14.4978445072
```

Operator numerik dalam Euler meliputi:

+ operator plus (unary atau biner)

- operator minus (unary atau biner)

*, / perkalian dan pembagian

. perkalian matriks

^b pemangkatan untuk b positif atau bilangan bulat (juga bisa dengan **b)

n! operator faktorial
dan masih banyak lagi.

Berikut beberapa fungsi yang mungkin Anda perlukan (masih 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, 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
```

Untuk memastikan, gunakan tanda kurung (parentheses) ketika ada keraguan tentang urutan eksekusi! Ekspresi berikut tidak sama dengan 2 karena dalam EMT bentuk bawaan untuk 2^3^4 adalah(beberapa sistem numerik lain melakukannya dengan cara berbeda

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

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

Bilangan Real

Tipe data utama dalam Euler adalah bilangan real. Bilangan real direpresentasikan dalam format IEEE dengan ketelitian 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
```

Strings

string dalam Euler adalah defined dengan "...".

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

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

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

```
>"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.
```

cetak fungsi does juga konversi number string. It dapat take number digits dan number places (0 untuk dense keluaran), dan optimally unit.

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

```
Golden Ratio : 1.61803
```

Di sana adalah special string none, which does not cetak. It adalah returned by some fungsi, when hasil does not matter. (It adalah returned automatically, jika fungsi does not have return statement.)

```
>none
```

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

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

```
1234.5
```

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

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

```
affe
charlie
bravo
```

Vektor string kosong dinyatakan dengan [none]. Vektor string dapat digabungkan.

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

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

Strings dapat contain Unicode characters. Internally, ini strings contain UTF-8 code. generate such string, gunakan u"..." dan one HTML entities.

Unicode strings dapat concatenated like other strings.

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

```
α = 45°
```

I

Dalam comments, same entities like α, β etc. dapat digunakan. Ini membuat alternatic cepat untuk Latex. (detail lebih di bawah).

Ada beberapa fungsi untuk membuat atau menganalisis string Unicode. Fungsi strtochar() akan mengenali string Unicode dan menerjemahkannya dengan benar.

```
>v=strtochar(u"&Auml; is a German letter")
```

```
[196,  32,  105,  115,  32,  97,  32,  71,  101,  114,  109,  97,  110,
32,  108,  101,  116,  116,  101,  114]
```

hasil adalah vector Unicode numbers. converse fungsi adalah chartoutf().

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

```
Ü is a German letter
```

fungsi utf() dapat translate string dengan entities dalam variabel into Unicode string.

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

```
We have α=β.
```

It adalah juga possible gunakan numerical entities.

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

```
Ähnliches
```

Boolean Values

Boolean values adalah represented dengan 1=true atau 0=false dalam Euler. Strings dapat compared, just like numbers.

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

```
0
1
```

"dan" adalah operator "&&" dan "atau" adalah operator "||", sebagai dalam C language. (words "dan" dan "atau" dapat digunakan dalam kondisi untuk "jika".)

```
>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 gunakan fungsi nonzeros() ekstrak spesifik elemen vector. Dalam conoth, kita gunakan 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]
```

Keluaran Formats

default keluaran format EMT prints 12 digits. buat itu kita lihat default, kita reset format.

```
>defformat; pi
```

```
     3.14159265359
```

Internally, EMT menggunakan IEEE standard untuk ganda numbers dengan about 16 decimal digits. lihat semua digits, gunakan perintah "longestformat", atau kita gunakan operator "longest" display hasil dalam longest format.

```
>longest pi
```

```
           3.141592653589793
```

Here adalah internal hexadecimal representation ganda angka.

```
>printhex(pi)
```

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

keluaran format dapat changed permanently dengan format perintah.

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

```
         0.33333
         3.14159
         0.84147
```

default adalah format(12).

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

```
     0.333333333333
```

Fungsi like "shortestformat", "shortformat", "longformat" work untuk vectors dalam 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
```

default format untuk scalars adalah format(12). But ini dapat changed.

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

```
     3.1416
```

fungsi "longestformat" set scalar format too.

```
>longestformat; pi
```

```
     3.141592653589793
```

Untuk reference, here adalah list most important keluaran formats.

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

internal accuracy EMT adalah about 16 decimal places, which adalah IEEE standard. Numbers adalah stored dalam ini internal format.

But keluaran format EMT dapat set dalam flexible way.

```
>longestformat; pi,
```

```
     3.141592653589793
```

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

```
       3.14159
```

default adalah defformat().

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

Di sana adalah short operators which cetak only one value. operator "longest" akan cetak all valid digits number.

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

```
           4.934802200544679
```

Di sana adalah juga short operator untuk menampilkan hasil dalam fractional format. We have already digunakan it above.

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

```
     25/12
```

Since internal format menggunakan binary way store numbers, value 0.1 akan not represented exactly. error adds up bit, sebagai Anda see dalam 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 dengan default "longformat" Anda akan not notice ini. Untuk convenience, keluaran very small numbers adalah 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 itu, gunakan tanda kurung setelah ekspresi. Jika Anda bermaksud menggunakan string sebagai ekspresi, gunakan konvensi penamaan seperti "fx" atau "fxy", dan sebagainya. Ekspresi memiliki prioritas lebih tinggi dibandingkan fungsi.

Variabel global dapat digunakan dalam evaluasi.

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

```
           12.56637061435917
```

Parameters adalah assigned x, y, dan z dalam itu order. Additional parameters dapat added using assigned parameters.

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

```
     -0.919535764538
```

Note itu expression akan always gunakan global variabel, even jika di sana adalah variabel dalam fungsi dengan same name. (Otherwise evaluation expressions dalam fungsi could have very confusing hasil untuk user itu called fungsi.)

```
>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 want gunakan another value untuk "pada" than global value Anda need add "pada=value".

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

```
     45
```

Untuk reference, we remark itu call collections (discussed elsewhere) dapat contain expressions. So we dapat buat above example sebagai follows.

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

```
     45
```

Expressions dalam x adalah often digunakan just like fungsi.
Note itu defining fungsi dengan same name like global symbolic expression deletes ini variabel avoid confusion antara symbolic expressions dan fungsi.

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

```
  12
```

By way convention, symbolic atau numerical expressions sebaiknya named fx, fxy etc. Ini naming scheme sebaiknya not digunakan untuk fungsi.

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

$$x^x\ (\log x + 1)$$

special form expression allows any variabel sebagai unnamed parameter evaluation expression, not just "x", "y" etc. Untuk ini, mulai expression dengan "@(variabel) ...".

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

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

Ini allows manipulate expressions dalam other variabel untuk fungsi EMT which need expression dalam "x".

most elementary way define simple fungsi adalah store its formula dalam symbolic atau numerical expression. Jika main variabel adalah x, expression dapat evaluated just like fungsi.

Sebagai Anda see dalam following example, global variabel adalah terlihat during evaluation.

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

```
  -0.475
```

All other variabel dalam expression dapat specified dalam evaluation using assigned parameter.

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

```
  -0.425
```

Ekspresi tidak harus simbolik. Hal ini diperlukan jika ekspresi memuat fungsi yang hanya dikenal dalam kernel numerik, bukan dalam Maxima.

Matematika Simbolik

EMT melakukan perhitungan simbolik dengan bantuan Maxima. Untuk detail lebih lanjut, mulailah dengan tutorial berikut, atau jelajahi referensi untuk Maxima. Para ahli Maxima sebaiknya mencatat bahwa ada perbedaan dalam sintaks antara sintaks asli Maxima dan sintaks bawaan ekspresi simbolik dalam EMT.

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

Pertama-tama, Maxima memiliki aritmetika “tak hingga” yang dapat menangani bilangan yang sangat besar.

```
>$&44!
```

Ini way, Anda dapat compute large hasil exactly. Let us compute

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

course, Maxima has more efficient fungsi untuk ini (sebagai does numerical part EMT).

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

learn more about specific fungsi ganda klik on it. E.g., try ganda clicking on "&binomial" dalam sebelumnya perintah baris. Ini opens documentation Maxima sebagai provided by authors itu program.

Anda akan learn itu following works too.

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

Jika Anda want replace x dengan any specific value gunakan "dengan".

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

Itu way Anda dapat gunakan solution equation dalam another equation.

Symbolic expressions adalah printed by Maxima dalam 2D form. reason untuk ini adalah special symbolic flag dalam string.

Sebagai Anda akan have seen dalam sebelumnya dan following examples, jika Anda have LaTeX installed, Anda dapat cetak symbolic expression dengan Latex. Jika not, following perintah akan issue error message.

cetak symbolic expression dengan LaTeX, gunakan $ infront & (atau Anda may ommit &) before perintah. Do not run Maxima perintah dengan $, jika Anda don't have LaTeX installed.

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

Symbolic expressions adalah parsed by Euler. Jika Anda need complex syntax dalam one expression, Anda dapat enclose expression dalam "...". gunakan more than simple expression adalah possible, but strongly discouraged.

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


Untuk completeness, we remark itu symbolic expressions dapat digunakan dalam programs, but need enclosed dalam quotes. Moreover, it adalah much more effective call Maxima pada compile time jika possible.

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

Again, % refers sebelumnya hasil.

buat things easier we save solution symbolic variabel. Symbolic variabel adalah defined dengan "&=".

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

Symbolic expressions dapat digunakan dalam other symbolic expressions.

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

direct masukan Maxima perintah adalah available too. Mulai perintah baris dengan "::". syntax Maxima adalah adapted syntax EMT (called "compatibility mode").

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

$$2432902008176640000$$

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

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

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

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

Jika Anda adalah expert dalam Maxima, Anda may wish gunakan original syntax Maxima. Anda dapat do ini dengan ":::".

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

$$g^2$$

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

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

Such variabel dapat digunakan dalam other symbolic expressions. Note, itu dalam following perintah right hand side &= adalah evaluated before penugasan Fx.

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

```
                                        5
                                   125 E

                              18551.64488782208
```

```
>fx(5)
```

```
18551.6448878
```

Untuk evaluation expression dengan specific values variabel, Anda dapat gunakan "dengan" operator.

following perintah baris juga demonstrates itu Maxima dapat evaluate expression numerically dengan float().

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

```
                                   10        5
                              1000 E   - 125 E

                             2.20079141499189e+7
```

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

get Latex code untuk expression, Anda dapat gunakan tex perintah.

```
>tex(fx)
```

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

Symbolic expressions dapat evaluated just like numerical expressions.

```
>fx(0.5)
```

```
0.206090158838
```

Dalam symbolic expressions, ini does not work, since Maxima does not support it. Instead, gunakan "dengan" syntax (nicer form pada(...) perintah Maxima).

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

penugasan dapat juga symbolic.

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

perintah solve solves symbolic expressions untuk variabel dalam Maxima. hasil adalah vector solutions.

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

Compare dengan numerical "solve" perintah dalam Euler, which needs mulai value, dan optionally target value.

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

```
  1.56155281281
```

numerical values symbolic solution dapat computed by evaluation symbolic hasil. Euler akan read over penugasan x= etc. Jika Anda do not need numerical hasil untuk further computations Anda dapat juga let Maxima find numerical values.

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

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

get specific symbolic solution, one dapat gunakan "dengan" dan index.

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

solve system equations, gunakan vector equations. hasil adalah vector solutions.

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

Symbolic expressions dapat have flags, which indicate special treatment dalam Maxima. Some flags dapat digunakan sebagai perintah too, others dapat't. Flags adalah appended dengan "|" (nicer form "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 programs defined dengan perintah "fungsi". It dapat one-baris fungsi atau multiline fungsi.
one-baris fungsi dapat numerical atau symbolic. numerical one-baris fungsi adalah defined by ":=".

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

Untuk overview, we show all possible definitions untuk one-baris fungsi. fungsi dapat evaluated just like any built-dalam Euler fungsi.

```
>f(2)
```

```
  4.472135955
```

Ini fungsi akan work untuk vectors too, obeying matrix language Euler, since expressions digunakan dalam fungsi adalah 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]
```

Fungsi dapat plotted. Instead expressions, we need only provide fungsi name.

Dalam contrast symbolic atau numerical expressions, fungsi name harus provided dalam string.

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

```
0.786151377757
```

By default, jika Anda need overwrite built-dalam fungsi, Anda harus add keyword "overwrite". Overwriting built-dalam fungsi adalah dangerous dan dapat cause problems untuk other fungsi depending on them.

Anda dapat still call built-dalam fungsi sebagai "_...", jika it adalah fungsi dalam Euler core.

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

```
0.707106781187
```

We better remove ini redefinition sin.

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

```
0.707106781187
```

parameter umum

fungsi numerik memiliki parameter umum

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

Omitting ini parameter menggunakan default value.

```
>f(4)
```

```
16
```

Setting it overwrites default value.

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

```
80
```

assigned parameter overwrite it too. Ini adalah digunakan by many Euler fungsi like plot2d, plot3d.

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

```
  16
```

Jika variabel adalah not parameter, it harus global. One-baris fungsi dapat see global variabel.

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

```
  24
```

But assigned parameter overrides global value.

Jika argument adalah not dalam list pre-defined parameters, it harus declared dengan ":="!

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

```
  20
```

Symbolic fungsi adalah defined dengan "&=". They adalah defined dalam Euler dan Maxima, dan work dalam both worlds. defining expression adalah run through Maxima before definition.

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

Symbolic fungsi dapat digunakan dalam symbolic expressions.

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

They dapat juga digunakan dalam numerical expressions. course, ini akan only work jika EMT dapat interpret everything inside fungsi.

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

```
  178.635099908
```

They dapat digunakan define other symbolic fungsi atau expressions.

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

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

```
  0.703467422498
```

following works too, since Euler menggunakan symbolic expression dalam fungsi g, jika it does not find symbolic variabel g, dan jika di sana adalah symbolic fungsi g.

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

```
0.703467422498
```

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

```
625
```

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

Dengan &= fungsi adalah symbolic, dan dapat digunakan dalam other symbolic expressions.

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

Dengan := fungsi adalah numerical. good example adalah definite integral like

which dapat not evaluated symbolically.

Jika we redefine fungsi dengan keyword "map" it dapat digunakan untuk vectors x. Internally, fungsi adalah called untuk all values x once, dan hasil adalah stored dalam vector.

```
>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 have default values untuk parameters.

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

Now fungsi dapat called dengan atau without parameter "base".

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

```
2
6.7
```

Moreover, it adalah possible gunakan assigned parameters.

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

```
2
```

Often, we want gunakan fungsi untuk vectors pada one place, dan untuk individual elements pada other places. Ini adalah possible dengan vector parameters.

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

Such symbolic fungsi dapat digunakan untuk symbolic variabel.

But fungsi dapat juga digunakan untuk numerical vector.

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

```
  17
```

Di sana adalah juga purely symbolic fungsi, which cannot digunakan 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 course, they dapat digunakan dalam symbolic expressions atau dalam definition symbolic fungsi.

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

Ringkasan

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

Menyelesaikan Ekspresi

Ekspresi dapat diselesaikan baik secara numerik maupun 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 secant method.

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

```
  1.41421356237
```

Ini works untuk symbolic expression too. Take following fungsi.

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

```
>px &= 4*x^8+x^7-x^4-x; $&px
```

Now we search point, where polynomial adalah 2. Dalam solve(), default target value y=0 dapat changed dengan assigned variabel.
We gunakan y=2 dan check by evaluating polynomial pada sebelumnya hasil.

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

```
  0.966715594851
  2
```

Solving symbolic expression dalam symbolic form returns list solutions. We gunakan symbolic solver solve() provided by Maxima.

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

easiest way get numerical values adalah evaluate solution numerically just like expression.

```
>longest sol()
```

```
      -0.6180339887498949       1.618033988749895
```

gunakan solutions symbolically dalam other expressions, easiest way adalah "dengan".

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

Solving systems equations symbolically dapat done dengan vectors equations dan symbolic solver solve(). answer adalah list lists equations.

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

fungsi f() dapat see global variabel. But often we want gunakan local parameters.

dengan =3.

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

One way pass additional parameter f() adalah gunakan list dengan fungsi name dan parameters (other way adalah titik koma parameters).

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

```
  2.54116291558
```

Ini does juga work dengan expressions. But maka, named list element has digunakan. (More on lists dalam tutorial about syntax 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 matrix

documentation EMT core contains detailed discussion on matrix language Euler.

Vectors dan matrices adalah entered dengan kuadrat brackets, elements dipisahkan by commas, rows dipisahkan by semicolons.

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

```
                1              2
                3              4
```

matrix product adalah denoted by dot.

```
>b=[3;4]
```

```
                3
                4
```

```
>b' // transpose b

     [3,  4]

>inv(A) //inverse

               -2             1
              1.5          -0.5

>A.b //perkalian matriks

               11
               25

>A.inv(A)

                1             0
                0             1
```

main point matrix language adalah itu all fungsi dan operators work element untuk element.

```
>A.A

                7            10
               15            22

>A^2 //perpangkatan elemen2

                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 dan b, ^(-1)b
```

```
               -2
              2.5
```

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

```
               -2
              2.5
```

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

```
                 1             0
                 0             1
```

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

```
                 1             0
                 0             1
```

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

```
                 1             4
                 9            16
```

Ini adalah not matrix product, but multiplication element by element. same works untuk vectors.

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

```
               9
              16
```

Jika one operands adalah vector atau scalar it adalah expanded dalam natural way.

```
>2*A
```

```
                 2              4
                 6              8
```

E.g., jika operand adalah column vector its elements adalah applied all rows .

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

```
                 1              4
                 3              8
```

Jika it adalah row vector it adalah applied all columns .

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

```
                 2              6
                 6             12
```

One dapat imagine ini multiplication sebagai jika row vector v had been duplicated form matrix same size sebagai .

```
>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 does juga apply untuk two vectors where one adalah row vector dan other adalah column vector. We compute i*j untuk i,j from 1 5. trick adalah multiply 1:5 dengan its transpose. matrix language Euler automatically generates table values.

```
>(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 itu ini adalah not matrix product!

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

```
     55
```

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

```
     55
```

Even operators like < atau == work dalam 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 dapat count number elements satisfying certain condition dengan fungsi sum().

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

```
     5
```

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

We get vector 0 dan 1, where 1 stands untuk 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 vector, "nonzeros" selects non-zero elements.

Dalam ini case, we get indices all elements greater than 50.

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

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

course, we dapat gunakan ini vector indices get corresponding values dalam t.

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

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

Sebagai example, let us find all squares numbers 1 1000, which adalah 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 adalah not completely effective untuk integer computations. It menggunakan ganda precision floating point internally. However, it adalah often very useful.

We dapat check untuk primality. Let us find out, how many squares plus 1 adalah primes.

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

```
  112
```

fungsi nonzeros() works only untuk vectors. Untuk matrices, di sana adalah 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 indices elements, which adalah not zeros.

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

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

Ini indices dapat digunakan set elements 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
```

fungsi mset() dapat juga set elements pada indices entries 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
```

Dan it adalah possible get elements dalam vector.

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

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

Another useful fungsi adalah extrema, which returns minimal dan maximal values dalam each row matrix dan their positions.

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

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

We dapat gunakan ini extract maximal values dalam each row.

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

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

Ini, course, adalah same sebagai fungsi max().

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

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

But dengan mget(), we dapat extract indices dan gunakan ini information extract elements pada 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 matrix lain

membangun matrix, kita dapat satu stack matrix diatas matrix laainnya. Jika tidak ada matrik columns yang sama, maka akan gagal dengan 0.

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

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

Likewise, we dapat attach matrix another side by side, jika both have same number 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
```

Jika they do not have same number rows shorter matrix adalah filled dengan 0.

Di sana adalah exception ini rule. real number attached matrix akan digunakan sebagai column filled dengan itu 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 adalah possible buat matrix row dan column vectors.

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

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

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

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

main purpose ini adalah interpret vector expressions untuk column vectors.

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

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

get size , we dapat gunakan following fungsi.

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

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

Untuk vectors, di sana adalah length().

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

```
     9
```

Di sana adalah many other fungsi, which generate matrices.

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

```
                 1              1
                 1              1
```

Ini dapat juga digunakan dengan one parameter. get vector dengan another number than 1, gunakan following.

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

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

Juga matrix random numbers dapat generated dengan random (uniform distribution) atau normal (Gauß distribution).

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

```
          0.66566      0.831835
            0.977      0.544258
```

Here adalah another useful fungsi, which restructures elements 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
```

Dengan following fungsi, we dapat gunakan ini dan dup fungsi write rep() fungsi, which repeats 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]
```

fungsi multdup() duplicates elements 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]
```

fungsi flipx() dan flipy() revert order rows atau columns matrix. I.e., fungsi flipx() flips horizontally.

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

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

Untuk rotations, Euler has rotleft() dan rotright().

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

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

special fungsi adalah drop(v,i), which removes elements dengan indices dalam i from vector v.

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

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

Note itu vector i dalam drop(v,i) refers indices elements dalam v, not values elements. Jika Anda want remove elements, Anda need find elements first. fungsi indexof(v,x) dapat digunakan find elements x dalam sorted vector 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]
```

Sebagai Anda see, it does not harm include indices out range (like 0), ganda indices, atau unsorted indices.

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

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

Di sana adalah some special fungsi set diagonals atau generate diagonal matrix.

We mulai dengan 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
```

Maka we set lower diagonal (-1) 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 itu we did not change matrix . We get new matrix sebagai hasil setdiag().

Here adalah fungsi, which returns 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
```

diagonal matrix dapat juga extracted from matrix. demonstrate ini, we restructure vector 1:9 3x3 matrix.

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

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

Now we dapat extract diagonal.

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

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

E.g. We dapat divide matrix by its diagonal. matrix language takes care itu column vector d adalah applied matrix row by row.

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

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

Vector

untuk smua fungsi dalam Euler work untuk matrix dan vector masukan juga, ketika ini masuk akal.

E.g., sqrt() fungsi computes kuadrat akar semua elemen vector atau matrix.

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

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

So Anda dapat easily create table values. Ini adalah one way plot fungsi (alternative menggunakan expression).

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

Dengan ini dan colon operator :delta:b, vectors values fungsi dapat generated easily.

Dalam following example, we generate vector values t[i] dengan spacing 0.1 from -1 1. Maka we generate vector values fungsi

```
>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 untuk scalars, vectors, dan matrices dalam obvious way.

E.g., column vector times row vector expands matrix, jika operator adalah applied. Dalam following, v' adalah transposed vector (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, itu ini adalah quite different from matrix product. matrix product adalah denoted dengan dot "." dalam EMT.

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

```
55
```

By default, row vectors adalah printed dalam compact format.

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

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

Untuk matrices special operator . denotes matrix multiplication, dan ' denotes transposing. 1x1 matrix dapat digunakan just like real number.

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

```
5
25
```

transpose matrix we gunakan apostrophe.

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

```
1
2
3
4
```

So we dapat compute matrix times vector b.

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

```
                30
                70
```

Note itu v adalah still row vector. So v'.v adalah 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 norm v squared untuk row vectors v. hasil adalah 1x1 vector, which works just like real number.

```
>v.v'
```

```
    30
```

Di sana adalah juga fungsi norm (along dengan many other fungsi Linear Algebra).

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

```
    30
```

Operators dan fungsi obey matrix language Euler.

Here adalah summary rules.

- fungsi applied vector atau matrix adalah applied each element.
- operator operating on two matrices same size adalah applied pairwise elements matrices.
- Jika two matrices have different dimensions, both adalah expanded dalam sensible way, so itu they have same size.

E.g., scalar value times vector multiplies value dengan each element vector. Atau matrix times vector (dengan *, not .) expands vector size matrix by duplicating it.

following adalah simple case dengan operator ^.

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

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

Here adalah more complicated case. row vector times column vector expands both by duplicating.

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

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

Note itu scalar product menggunakan matrix product, not *!

```
>v.v'


     14
```

Di sana adalah numerous fungsi untuk matrices. We give short list. Anda sebaiknya consult documentation untuk more information on ini perintah.

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

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


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

: operator generates equally spasi row vector, optionally dengan step size.

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


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

concatenate matrices dan vectors di sana adalah operators "|" dan "_".

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


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

elements matrix adalah referred dengan "[i,j]".

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


     6
```

Untuk row atau column vectors, v[i] adalah i-th element vector. Untuk matrices, ini returns complete i-th row matrix.

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

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

indices dapat juga row vectors indices. : denotes all indices.

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

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

short form untuk : adalah omitting index completely.

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

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

Untuk purposes vectorization, elements matrix dapat accessed sebagai jika they adalah vectors.

```
>A{4}
```

```
  4
```

matrix dapat juga flattened, using redim() fungsi. Ini adalah implemented 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]
```

gunakan matrices untuk tables, let us reset default format, dan compute table sine dan cosine values. Note itu angles adalah dalam 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 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
```

Using matrix language, we dapat generate several tables several fungsi pada once.

Dalam following example, we compute t[j]^i untuk i from 1 n. We get matrix, where each row adalah table t^i untuk one i. I.e., matrix has elements

fungsi which does not work untuk vector masukan sebaiknya "vectorized". Ini dapat achieved by "map" keyword dalam fungsi definition. Maka fungsi akan evaluated untuk each element vector parameter.

numerical integration integrate() works only untuk scalar interval bounds. So we need vectorize it.

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

"map" keyword vectorizes fungsi. fungsi akan now work untuk vectors numbers.

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

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

Sub-Matrices dan Matrix-Elements

access matrix element, gunakan 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 dapat access complete baris matrix.

```
>A[2]
```

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

Dalam case row atau column vectors, ini returns element vector.

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

```
 2
```

buat sure, Anda get first row untuk 1xn dan mxn matrix, specify all columns using empty second index.

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

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

Jika index adalah vector indices, Euler akan return corresponding rows matrix.

Here we want first dan second row .

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

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

We dapat even reorder using vectors indices. precise, we do not change here, but compute reordered version .

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

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

index trick works dengan columns too.

Ini example selects all rows dan second dan third column.

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

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

Untuk abbreviation ":" denotes all row atau column indices.

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

```
                3
                6
                9
```

Alternatively, leave first index empty.

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

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

We dapat juga get last baris .

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

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

Now let us change elements by assigning submatrix some value. Ini does dalam fact change stored matrix .

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

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

We dapat juga assign value row .

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

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

We dapat even assign sub-matrix jika it has proper size.

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

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

Moreover, some shortcuts adalah allowed.

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

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

warning: Indices out bounds return empty matrices, atau error message, depending on system setting. default adalah error message. Remember, however, itu negative indices may digunakan access elements matrix counting from akhir.

```
>A[4]
```

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

Pengurutan dan Pengacakan

Fungsi sort() mengurutkan vektor baris.

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

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

It adalah often necessary know indices sorted vector dalam original vector. Ini dapat digunakan reorder another vector dalam same way.

Let us shuffle vector.

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

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

indices contain proper order v.

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

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

Ini works untuk 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
```

Sebagai Anda see, position ganda entries adalah somewhat random.

```
>ind
```

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

fungsi unique returns sorted list unique elements vector.

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

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

Ini works untuk string vectors too.

```
>unique(s)
```

```
     a
     aa
     d
     e
```

aljabar linear

EMT has lots fungsi solve linear systems, sparse systems, atau regression problems.

Untuk linear systems Ax=b, Anda dapat gunakan Gauss algorithm, inverse matrix atau linear fit. operator \b menggunakan version Gauss algorithm.

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

```
                -4
               4.5
```

Untuk another example, we generate 200x200 matrix dan sum its rows. Maka we solve Ax=b using inverse matrix. We measure error sebagai maximal deviation all elements from 1, which course adalah correct solution.

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

```
       8.790745908981989e-13
```

Jika system does not have solution, linear fit minimizes norm error Ax-b.

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

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

determinant ini matrix adalah 0.

```
>det(A)
```

```
     0
```

Symbolic Matrices

Maxima has symbolic matrices. course, Maxima dapat digunakan untuk such simple linear algebra problems. We dapat define matrix untuk Euler dan Maxima dengan &:=, dan maka gunakan it dalam symbolic expressions. usual [...] form define matrices dapat digunakan dalam Euler 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 variabel, ini matrices dapat digunakan dalam other symbolic expressions.

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

eigenvalues dapat juga computed automatically. hasil adalah vector dengan two vectors eigenvalues dan multiplicities.

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

extract specific eigenvector needs careful indexing.

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

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

Symbolic matrices dapat evaluated dalam Euler numerically just like other symbolic expressions.

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

```
                 1              4
                 5              2
```

Dalam symbolic expressions, gunakan dengan.

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

Access rows symbolic matrices work just like dengan numerical matrices.

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

symbolic expression dapat contain penugasan. Dan itu changes matrix .

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

Di sana adalah symbolic fungsi dalam Maxima create vectors dan matrices. Untuk ini, refer documentation Maxima atau tutorial about Maxima dalam EMT.

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

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

hasil dapat evaluated numerically dalam Euler. Untuk more information about Maxima, see introduction Maxima.

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

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

Euler has juga powerful fungsi xinv(), which makes bigger effort dan gets more exact hasil.

Note, itu dengan &:= matrix B has been defined sebagai symbolic dalam symbolic expressions dan sebagai numerical dalam numerical expressions. So we dapat gunakan it here.

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

```
                              1                              0
                              0                              1
```

E.g. eigenvalues dapat computed numerically.

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

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

Atau symbolically. See tutorial about Maxima untuk details on ini.

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

Nilai Numerik dalam Ekspresi Simbolik

Ekspresi simbolik hanyalah sebuah 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
```

Di sana adalah still difference antara numerical dan symbolic form. When transferring matrix symbolic form, fractional approximations untuk reals akan digunakan.

```
>$&A
```

avoid ini, di sana adalah fungsi "mxmset(variabel)".

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

Maxima dapat juga compute dengan floating point numbers, dan even dengan big floating numbers dengan 32 digits. evaluation adalah much slower, however.

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

precision big floating point numbers dapat changed.

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

```
            3.14159265358979323846264338327950288419716939937510582097494\
  459230781640628620899862803482534211706 8b0
```

numerical variabel dapat digunakan dalam any symbolic expressions using "@var".

Note itu ini adalah only necessary, jika variabel has been defined dengan ":=" atau "=" sebagai numerical variabel.

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

Demo – Suku Bunga

Below, we gunakan Euler Math Toolbox (EMT) untuk calculation interest rates. We do itu numerically dan symbolically show Anda how Euler dapat digunakan solve real life problems.

Assume Anda have seed capital 5000 (say dalam dollars).

```
>K=5000
```

```
  5000
```

Now we assume interest rate 3% per year. Let us add one simple rate dan compute hasil.

```
>K*1.03
```

```
  5150
```

Euler would understand following syntax too.

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

```
  5150
```

But it adalah easier gunakan factor

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

```
1.03
5150
```

Untuk 10 years, we dapat simply multiply factors dan get final value dengan compound interest rates.

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

```
6719.58189672
```

Untuk our purposes, we dapat set format 2 digits after decimal dot.

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

```
6719.58
```

Let us cetak itu rounded 2 digits dalam complete sentence.

```
>"Starting from " + K + "$ you get " + round(K*q^10,2) + "$."
```

```
Starting from 5000$ you get 6719.58$.
```

What jika we want know intermediate hasil from year 1 year 9? Untuk ini, Euler's matrix language adalah big bantuan. Anda do not have write loop, but simply enter

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

```
Real 1 x 11 matrix
5000.00     5150.00     5304.50     5463.64     ...
```

How does ini miracle work? First expression 0:10 returns vector integers.

```
>short 0:10
```

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

Maka all operators dan fungsi dalam Euler dapat applied vectors element untuk 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]
```

adalah vector factors q^0 q^10. Ini adalah multiplied by K, dan we get vector values.

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

course, realistic way compute ini interest rates would round nearest cent after each year. Let us add fungsi untuk ini.

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

Let us compare two hasil, dengan dan without rounding.

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

```
                  1271.61
                1271.6071
```

Now di sana adalah no simple formula untuk n-th year, dan we harus loop over years. Euler provides many solutions untuk ini.

easiest way adalah fungsi iterate, which iterates given fungsi number times.

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

```
    Real 1 x 11 matrix
        5000.00     5150.00     5304.50     5463.64      ...
```

We dapat cetak itu dalam friendly way, using our format dengan 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
```

get specific element vector, we gunakan indices dalam kuadrat brackets.

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

```
        5150.00
        5000.00     5150.00     5304.50
```

Surprisingly, we dapat juga gunakan vector indices. Remember itu 1:3 produced vector [1,2,3].

Let us compare last element rounded values dengan full values.

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

```
        6719.60
        6719.58
```

difference adalah very small.

Menyelesaikan Persamaan

sekarang kita bisa gunakan fungsi lanjutan , dengan menggunakan uang tiap tahun.

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

We do not have specify q atau R untuk definition fungsi. Only jika we run perintah, we have define ini 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 jika we remove same amount each year?

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

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

We see itu money decreases. Obviously, jika we get only 150 interest dalam first year, but remove 200, we lose money each year.

How dapat we determine number years money akan last? We would have write loop untuk ini. easiest way adalah iterate long enough.

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

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

Using matrix language, we dapat determine first negative value dalam following way.

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

```
           48.00
```

Alasannya adalah karena nonzeros(VKR<0) mengembalikan indeks vektor
I di mana VKR[i]<0, dan min menghitung indeks terkecil.

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

Fungsi iterate() memiliki satu trik lagi. Fungsi ini dapat menerima kondisi akhir sebagai argumen. Maka, fungsi tersebut akan mengembalikan nilai sekaligus jumlah iterasi.

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

```
          -19.83
           47.00
```

Mari kita coba menjawab pertanyaan yang lebih ambigu. Misalkan kita tahu nilainya menjadi 0 setelah 50 tahun. Berapa suku bunganya?

Ini adalah pertanyaan yang hanya bisa dijawab secara numerik. Di bawah, kita akan menurunkan rumus-rumus yang diperlukan. Nanti Anda akan melihat bahwa tidak ada rumus sederhana untuk suku bunga. Tetapi untuk sekarang, kita bertujuan pada solusi numerik.

Langkah pertama adalah mendefinisikan fungsi yang melakukan iterasi sebanyak N kali. Kita tambahkan semua parameter ini ke fungsi tersebut.

```
>function f(K,R,P,n) := iterate("x*(1+P/100)+R",K,n;P,R)[-1]
```

Iterasi sama saja seperti di atas

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

Tetapi kita tidak lagi menggunakan nilai global R dalam ekspresi kita. Fungsi seperti iterate() memiliki trik khusus dalam Euler: Anda dapat meneruskan nilai variabel ke dalam ekspresi sebagai parameter dengan titik koma. Dalam hal ini, variabelnya adalah P dan R.

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

Mari kita coba uji

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

```
          -19.83
```

Now we dapat solve our problem.

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

```
            3.15
```

solve routine solves expression=0 untuk variabel x. answer adalah 3.15% per year. We take mulai value 3% untuk algorithm. solve() fungsi always needs mulai value.

We dapat gunakan same fungsi solve following question: How much dapat we remove per year so itu seed capital adalah exhausted after 20 years assuming interest rate 3% per year.

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

```
         -336.08
```

Catatan: Anda tidak dapat menyelesaikan untuk jumlah tahun (number of years), karena fungsi kita mengasumsikan nilai
?
adalah bilangan bulat.

Solusi Simbolik — Masalah Suku Bunga

Kita dapat menggunakan bagian simbolik Euler untuk mempelajari masalah ini. Pertama, kita definisikan fungsi onepay() secara simbolik.

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

We dapat now iterate ini.

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

We see pattern. After n periods we have

formula adalah formula untuk geometric sum, which adalah known Maxima.

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

Ini adalah bit tricky. sum adalah evaluated dengan flag "simpsum" reduce it quotient.

Let us 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)
```

fungsi does same sebagai our fungsi f before. But it adalah more effective.

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

```
          -19.82504734650985
          -19.82504734652684
```

Kuta dapat gunakan ini untuk bertanyaa 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 nilainya akan menjadi negatif setelah 21 tahun.

Kita juga dapat menggunakan sisi simbolik Euler untuk menghitung rumus-rumus untuk pembayaran.

Misalkan kita mendapat pinjaman K, dan membayar K kali pembayaran K(dimulai setelah tahun pertama) sehingga tersisa utang residu K pada saat pembayaran terakhir. Rumus untuk ini jelas sebagai berikut

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

Biasanya rumus ini diberikan dalam bentuk

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

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

Kita dapat menyelesaikan untuk suku bunga R secara simbolik

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

Seperti yang dapat Anda lihat dari rumus, fungsi ini menghasilkan galat bilangan floating point untuk.
Namun demikian, Euler tetap menampilkannya dalam plot.

Tentu saja, kita memiliki limit berikut.

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

Jelas, tanpa bunga kita harus membayar kembali 10 kali cicilan sebesar 500.
Persamaan tersebut juga dapat diselesaikan untuk n. Hasilnya akan terlihat lebih rapi jika kita menerapkan beberapa penyederhanaan.

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

---

* RANGKUMAN MATERI PEKAN 3-4

1) Operasi bentuk aljabar: menjabarkan (expand), menyederhanakan (simplify), memfaktorkan (factor)
Definisikan variabel simbolik (opsional untuk beberapa operasi simbolik)
x := symbol("x");
y := symbol("y");

Contoh expand, factor, simplify
expand((x+1)^3);
factor(x^3-1);
menyederhanakan pecahan
simplify((x^2-1)/(x-1));

collect (mengelompokkan menurut x)
collect(x^3 + 2*x^2 + x, x);

2) Perhitungan numerik dengan operasi dan fungsi matematika
perasi dasar
a := 12.5;
b := 3.2;
a + b;
a * b;
a / b;
fungsi trigonometri dan log
sin(pi/6); cos(45°); tan(30°);
log(10); exp(1);

3) Perhitungan dengan bilangan kompleks
z := 1 + 2i; i adalah unit imajiner
w := 3 - 4i;
z * w;
abs(z); modulus
arg(z); argumen / sudut (radian)
conj(z); konjugat
re(z); im(z); bagian real & imajiner
pangkat dan eksponensial kompleks
exp(i * pi); simplify(exp(i*pi));

4) Fungsi buatan sendiri (numerik dan simbolik)
fungsi numerik sederhana
function f(x) := x^2 + 2*x + 1; definisi numerik
f(3);

fungsi simbolik / turunan simbolik
function g(x) &= diff(x^x, x); definisi simbolik (gunakan & untuk symbolic)
g(2); evaluasi numerik dari fungsi simbolikturunan, integral simbolik
h := x^3 - 3*x^2 + 2*x;
diff(h, x);
integrate(h, x);

5) Menyelesaikan persamaan dan sistem persamaan
Persamaan tunggal (metode numerik/simbolik)
solve("x^2-2", 1); mencari akar dekat 1 (hasil ~ sqrt(2))
solve("cos(x)-x", 1); contoh nonlinier, mulai tebakan 1

Persamaan simbolik (bisa menghasilkan beberapa solusi)
sol := &solve(x^2 - x - 2, x); simbolik, simpan solusi
sol; tampilkan solusi
Menyisipkan solusi ke dalam ekspresi: contoh cek
expand((x^2 - x - 2) with sol[2]);

```
Sistem persamaan (simbolik)
contoh: x + y = 3, x^2 + y^2 = 5
```

solsys := &solve([x + y = 3, x^2 + y^2 = 5], [x, y]);
solsys;
% Ambil solusi pertama dan substitusi
xval := x with solsys[1];
yval := y with solsys[1];
xval, yval;

6) Menyelesaikan pertidaksamaan (cara praktis menggunakan akar batas)
% Untuk pertidaksamaan sederhana, cari akar polinomial pembatas, lalu tentukan interval
% contoh: x^2 - 4 > 0 -> akar di x= -2, 2 -> solusi: (-inf, -2) U (2, inf)
roots := solve("x^2-4", 3); % dapatkan akar (numerik/simbolik)
roots;
% setelah mendapat akar, Anda bisa cek tanda pada interval untuk menentukan solusi pertidaksamaan

7) Manipulasi dan perhitungan matriks & vektor
% definisi vektor dan matriks
v := [1, 2, 3]; % vektor baris
w := [1; 2; 3]; % vektor kolom
A := [1, 2; 3, 4]; % matriks 2x2
B := [2, 0; 1, 3];

% operasi dasar
A + B;
A * B;
A * v'; % A kali v kolom (v' membuat v kolom)
det(A);
inv(A); % invers matriks
rank(A);
% solusi sistem linear numerik: A * x = b
bvec := [5; 6];
xsol := A \ bvec; % operator backslash (jika tersedia) untuk menyelesaikan
xsol;
% alternatif: gunakan inv(A)*bvec jika backslash tidak tersedia
inv(A) * bvec;

% nilai eigen, vektor eigen
eig(A); % menghasilkan nilai eigen (numeric) dan/atau pasangan

% dekomposisi dan norma
svd(A);
norm(A);

8) Menggunakan aljabar untuk menyelesaikan masalah nyata (contoh)

```
% Contoh A: Luas segitiga dari koordinat tiga titik (x1,y1), (x2,y2), (x3,y3)
x1 := 0; y1 := 0;
x2 := 4; y2 := 0;
x3 := 1; y3 := 3;
% Luas = 0.5 * | det([x2-x1, y2-y1; x3-x1, y3-y1]) |
area_triangle := 0.5 * abs(det([x2-x1, y2-y1; x3-x1, y3-y1]));
area_triangle;

% Contoh B: Pecahkan masalah aljabar nyata
% Misal soal: dua bilangan x,y memenuhi x+y=10 dan x*y=21. Cari x dan y.
sol_b := &solve([x + y = 10, x * y = 21], [x, y]);
sol_b;

% Contoh C: Interpolasi polinomial sederhana (fit polinomial)
xs := [0, 1, 2, 3];
ys := [1, 3, 7, 13];
p := polyfit(xs, ys, 2); % polinomial orde 2 aproksimasi
p; % koefisien polinom
% evaluasi polinom di x=1.5
polyval(p, 1.5);
```

soal soal untuk EMT

TENTUKAN HASIL DARI

3^-7

JAWABAN :

```
>expr1 := 3^(-7)
```

```
0.000457247370828
```

```
>1/3^7
```

```
0.000457247370828
```

MENYEDERHANAKAN BENTUK ALJABAR

3x2y2·(-10x2y-9)

HASILNYA :

```
>$&3*x^(2)*y^2*-10*x^2*y^(-9)
```

$$-\frac{30\,x^4}{y^7}$$

Menjabarkan Ekspresi (Expand)

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

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

Definisi Variabel dan Operato

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

```
    16.7551608191
```

Fungsi dan Evaluasi Fungsi

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

```
    4.472135955
```

Loop dan Iterasi

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

```
    1.5
    1.41666666667
    1.41421568627
    1.41421356237
    1.41421356237
```

Pemecahan Persamaan

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

```
    1.41421356237
```

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

$$\left[x=-\sqrt{2},x=\sqrt{2}\right]$$

Matriks dan Operasi Linear

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

```
                1              2
                3              4
```

```
>b = [5; 6]
```

```
                5
                6
```

```
>A \ b
```

```
                -4
               4.5
```

String dan Unicode

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

```
    Golden Ratio: 1.61803
```

```
>u"&alpha; = " + 45 + u"&deg;"
```

```
    α = 45°
```

Penyelesaian Pertidaksamaan Simbolik

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

```
          C:/Program Files/Euler x64/maxima/share/maxima/5.35.1/share/fo\
    urier_elim/fourier_elim.lisp
```

```
>$&fourier_elim([x^2 - 1 > 0], [x])
```

$$[1 < x] \vee [x < -1]$$

Fungsi dengan Parameter Default

```
>function mylog (x, base=10) := ln(x) / ln(base)
>mylog(100)
```

```
    2
```

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

```
    6.7
```

Fungsi Iterasi dan Simulasi Kas Bunga Majemuk

```
> K = 5000
```

```
    5000
```

```
>q = 1 + 3%

    1.03

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

    [5000,  5150,  5304.5,  5463.64,  5627.54,  5796.37,  5970.26,
    6149.37,  6333.85,  6523.87,  6719.58]

>format(12,2); VK

    Real 1 x 11 matrix

        5000.00     5150.00     5304.50     5463.64     ...
```