EMT untuk Perhitungan Aljabar

KELOMPOK 1 PENDIDIKAN MATEMATIKA D

- Nathanael Paulus Lumban Tobing (23030130085)

Materi: Bilangan riil, String, Nilai Boolean

- Nanda Shelvy Pratiwi (23030130093)

Materi: Baris Perintah, Sintaks Dasar

- Amelia Fadilah (23030130098)

Materi: Matematika simbolik

- Ajeng Puspa Nabillah (23030130102)

Materi: Format keluaran, ekspresi

- Anantasya Asmarani Batudawati (23030130105)

Materi: Matematika simbolik

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:

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

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

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

Baris Perintah

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

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

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

```
16.7551608191
```

Perintah harus dipisahkan dengan yang kosong. Baris perintah

berikut mencetak dua hasilnya.

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

```
50.2654824574
```

```
100.530964915
```

Baris perintah dijalankan sesuai urutan yang ditekan pengguna kembali. Jadi, Anda mendapatkan nilai baru setiap kali Anda menjalankan baris kedua.

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

0.540302305868

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

0.857553215846

Jika dua jalur dihubungkan dengan "..." kedua jalur akan selalu dijalankan secara bersamaan.

```
>x := 1.5; ...
```

1.41666666667
1.41421568627
1.41421356237

Ini juga merupakan cara yang baik untuk menyebarkan perintah panjang ke 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.

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

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

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

```
>% x^2 // This line will be invisible.
```

81

Euler mendukung loop di baris perintah, asalkan cocok ke dalam satu baris atau multi-baris. Tentu saja, pembatasan ini tidak berlaku dalam program.

Untuk informasi lebih lanjut 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 apa-apa menggunakan multi-baris. Pastikan baris diakhiri dengan "...".

```
>x := 1.5; // comments go here before the ...
```

```
1.41421356237
```

Struktur bersyarat juga berfungsi.

```
>if E^pi>pi^E; then "Ayo sregep sinau!", endif;
```

```
Ayo sregep sinau!
```

Saat Anda menjalankan perintah, kursor dapat berada di posisi mana pun di baris perintah. Anda dapat kembali ke perintah sebelumnya atau melompat ke perintah berikutnya dengan tombol panah. Atau Anda dapat mengklik bagian komentar di atas perintah untuk membuka perintah.

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

```
>sqrt(sin(10°)/cos(20°))
```

```
0.429875017772
```

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

Anda dapat mengklik dua kali pada perintah apa pun untuk membuka bantuan untuk perintah ini. Coba klik dua kali perintah `exp` di bawah ini 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 mouse atau gunakan shift bersamaan dengan tombol kursor apa pun. Selain itu, Anda dapat menyalin tanda kurung yang disorot.

Sintaks Dasar

Euler mengetahui fungsi matematika biasa. Seperti yang Anda lihat di atas, fungsi trigonometri bekerja dalam radian atau derajat. Untuk mengonversi ke derajat, tambahkan simbol derajat (dengan tombol F7) ke nilainya, atau gunakan fungsi rad(x). Fungsi akar kuadrat disebut sqrt di Euler. Tentu saja, $x^{(1/2)}$ juga dimungkinkan.

Untuk menyetel variabel, gunakan "=" atau ":=". Demi kejelasan, pendahuluan ini menggunakan bentuk yang terakhir. Spasi tidak penting. Tapi jarak antar perintah diharapkan.

Beberapa perintah dalam satu baris dipisahkan dengan ";" atau ";". Titik koma menekan keluaran perintah. Di akhir baris perintah, ";" diasumsikan, jika ";" hilang.

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

30.65625

EMT menggunakan sintaks pemrograman untuk ekspresi. Untuk enter

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

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

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

8.77908249441

Untuk menghitung ekspresi rumit seperti

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

Anda harus memasukkannya dalam formulir baris.

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

```
23.2671801626
```

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

Hasil perhitungan ini berupa bilangan floating point. Ini secara default dicetak dengan akurasi sekitar 12 digit. Di baris perintah berikut, kita juga mempelajari bagaimana kita bisa merujuk ke hasil sebelumnya dalam baris yang sama.

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

```
0.47619047619  
10/21
```

Perintah Euler dapat berupa ekspresi atau perintah primitif. Ekspresi terbuat dari operator dan fungsi. Jika perlu, harus berisi tanda kurung untuk memaksakan urutan eksekusi yang benar. Jika ragu, memasang braket 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 operasi tambah  
- unary atau operasi kurang  
a^b a pangkat b, atau integer b (selain itu dapat juga memakai a**b)  
n! operator faktorial
```

dan masih banyak lagi.

Berikut 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 mempunyai alias, misalnya `ln` untuk `log`.

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

```
2  
0.5
```

```
>sin(30°)
```

```
0.5
```

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

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

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


Sebuah string di Euler didefinisikan dengan "...".

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

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

String dapat digabungkan dengan `|` atau dengan `+`. Ini juga berfungsi dengan angka-angka, yaitu dikonversi menjadi string dalam hal ini.

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

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

Fungsi `print` juga mengubah angka menjadi string. Ini bisa memakan waktu beberapa saat digit dan jumlah tempat (0 untuk keluaran padat), dan optimalnya satuan.

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

```
Golden Ratio : 1.61803
```

Ada string khusus `none` yang tidak dicetak. Itu dikembalikan oleh beberapa fungsi, ketika hasilnya tidak penting. (Ini dikembalikan secara otomatis, jika fungsinya demikian tidak memiliki pernyataan pengembalian.)

```
>none
```

Untuk mengonversi string menjadi angka, cukup evaluasi saja. Ini juga berfungsi untuk ekspresi (lihat di bawah).

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

```
1234.5
```

Untuk mendefinisikan vektor string, gunakan notasi vektor

```
[...].
```

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

```
affe  
charlie  
bravo
```

Vektor string kosong dilambangkan dengan [tidak ada]. Vektor string dapat digabungkan.

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

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

String dapat berisi karakter Unicode. Secara internal, string ini berisi kode UTF-8. Untuk menghasilkan string seperti itu, gunakan u"..." dan salah satu entitas HTML.

String Unicode dapat digabungkan seperti string lainnya.

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

```
= 45°
```

Dalam komentar, entitas yang sama seperti $\ddot{y}$, $\ddot{y}$ dll. dapat digunakan. Ini mungkin merupakan alternatif cepat untuk Lateks. (Detail lebih lanjut di komentar di bawah).

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

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

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

Hasilnya adalah vektor angka Unicode. Fungsi kebalikannya adalah `chartoutf()`.

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

```
Ü is a German letter
```

Dimungkinkan juga untuk menggunakan entitas numerik.

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

```
We have =.
```

Dimungkinkan juga untuk menggunakan entitas numerik.

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

Ähnliches

Nilai Boolean

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

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

0
1

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

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

1

Operator Boolean mematuhi aturan bahasa matriks.

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

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

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

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

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

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

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

Format Keluaran

Format keluaran default EMT mencetak 12 digit. Untuk memastikan bahwa kami melihat defaultnya, kami mengatur ulang formatnya

```
>defformat; pi
```

```
3.14159265359
```

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

```
>longest pi
```

```
3.141592653589793
```

Berikut adalah representasi heksadesimal internal dari bilangan ganda.

```
>printhex(pi)
```

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

Format keluaran dapat diubah secara permanen dengan perintah format.

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

```
0.33333  
3.14159  
0.84147
```

Standarnya adalah format(12).

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

```
0.333333333333
```

Fungsi seperti "shortestformat", "shortformat", "longformat" berfungsi untuk vektor dengan cara berikut.

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

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

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

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

```
3.1416
```

Fungsi "longestformat" juga mengatur format skalar.

```
>longestformat; pi
```

```
3.141592653589793
```

Sebagai referensi, berikut adalah daftar format keluaran terpenting.

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

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

Namun format keluaran EMT dapat diatur dengan cara yang fleksibel.

```
>longestformat; pi,
```

```
3.141592653589793
```

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

```
3.14159
```

Standarnya adalah `defformat()`.

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

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

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

```
4.934802200544679
```

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

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

```
25/12
```

Karena format internal menggunakan cara biner untuk menyimpan angka, nilai 0,1 tidak akan direpresentasikan secara tepat. Kesalahannya bertambah sedikit, seperti yang Anda lihat pada perhitungan berikut.

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

```
-1.110223024625157e-16
```

Tetapi dengan "longformat" default Anda tidak akan

menyadarinya. Untuk kenyamanan, keluaran angka yang sangat kecil adalah 0.

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

0

Ekspresi

String atau nama dapat digunakan untuk menyimpan ekspresi matematika, yang dapat dievaluasi dengan EMT. Untuk ini, gunakan tanda kurung setelah ekspresi. Jika Anda ingin menggunakan string sebagai ekspresi, gunakan konvensi untuk menamainya "fx" atau "fxy" dll. Ekspresi lebih diutamakan daripada fungsi.

Variabel global dapat digunakan dalam evaluasi.

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

12.56637061435917

Parameter ditetapkan ke x, y, dan z dalam urutan itu. Parameter tambahan dapat ditambahkan menggunakan parameter yang ditetapkan.

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

-0.919535764538

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

```
>at:=4; function f(expr,x,at) := expr(x); ...
```

36

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

```
>at:=4; function f(expr,x,a) := expr(x,at=a); ...
```

45

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

```
>at:=4; function f(expr,x) := expr(x); ...
```

45

Ekspresi dalam x sering digunakan seperti fungsi.

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

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

12

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

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

Bentuk ekspresi khusus memungkinkan variabel apa pun sebagai parameter yang tidak disebutkan namanya untuk mengevaluasi ekspresi, bukan hanya " x ", " y ", dll. Untuk ini, mulailah ekspresi dengan "@(variabel) ...".

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

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

41

Hal ini memungkinkan untuk memanipulasi ekspresi dalam variabel lain untuk fungsi EMT yang memerlukan ekspresi dalam "x".

Cara paling dasar untuk mendefinisikan suatu fungsi sederhana adalah dengan menyimpan rumusnya dalam ekspresi simbolik atau numerik.

Jika variabel utamanya adalah x, ekspresi dapat dievaluasi seperti halnya fungsi.

Seperti yang Anda lihat pada contoh berikut, variabel global terlihat selama evaluasi.

```
>fx &= x^3-a*x; ...
```

```
-0.475
```

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

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

```
-0.425
```

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

Matematika Simbolik

EMT melakukan matematika simbolis dengan bantuan Maxima. Untuk detailnya, mulailah dengan tutorial berikut, atau telusuri referensi untuk Maxima.

Para ahli di Maxima harus memperhatikan bahwa ada perbedaan sintaksis antara sintaksis asli Maxima dan sintaksis default ekspresi simbolik di EMT.

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

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

```
>&44!
```

```
2658271574788448768043625811014615890319638528000000000
```

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

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

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

```
2481256778
```

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

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

2481256778

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

Anda akan mengetahui bahwa cara berikut juga bisa dilakukan.

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

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

$$\frac{(x-2)(x-1)x}{6}$$

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

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

Dengan begitu Anda bisa menggunakan solusi suatu persamaan di persamaan lain.

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

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

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

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

Ekspresi simbolik diurai oleh Euler. Jika Anda memerlukan sintaksis kompleks dalam satu ekspresi, Anda dapat mengait ekspresi tersebut di "...". Menggunakan lebih dari sekedar ekspresi sederhana mungkin saja dilakukan, namun sangat tidak disarankan.

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

Untuk kelengkapannya, kami mencatat bahwa ekspresi simbolik dapat digunakan dalam program, namun perlu diapit dalam 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
```

Sekali lagi, % mengacu pada hasil sebelumnya.

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

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

Ekspresi simbolik dapat digunakan dalam ekspresi simbolik lainnya.

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

Input langsung dari perintah Maxima juga tersedia. Mulai baris perintah dengan ":".

Sintaks Maxima disesuaikan dengan sintaks EMT (disebut

"compatibility mode").

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

```
2432902008176640000
```

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

```
8 4 2  
2 3 5 7
```

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

```
18 8 4 2  
2 3 5 7 11 13 17 19
```

Jika Anda ahli dalam Maxima, Anda mungkin ingin menggunakan

sintaks asli Maxima. Anda dapat melakukan ini dengan ">:::".

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

$$g^2$$

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

$$x^3 E$$

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

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

5
125 E

18551.64488782208

```
>fx(5)
```

18551.6448878

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

Baris perintah berikut juga menunjukkan bahwa Maxima dapat mengevaluasi ekspresi secara numerik 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))
```

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

```
>tex(fx)
```

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

Ekspresi simbolik dapat dievaluasi seperti halnya ekspresi numerik.

```
>fx(0.5)
```

```
0.206090158838
```

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

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

Penugasannya juga bisa bersifat simbolis.

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

Perintah `solve` menyelesaikan ekspresi simbolik untuk variabel

di Maxima. Hasilnya adalah vektor solusi.

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

Bandungkan dengan perintah numerik "solve" di Euler, yang memerlukan nilai awal, dan opsional nilai target.

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

```
1.56155281281
```

Nilai numerik dari solusi simbolik dapat dihitung dengan evaluasi hasil simbolik. Euler akan membacakan tugas `x=dst`. Jika Anda tidak memerlukan hasil numerik untuk perhitungan lebih lanjut, Anda juga dapat membiarkan Maxima menemukan nilai numeriknya.

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

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

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

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

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

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

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

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

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

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

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

```
>f(2)
```

4.472135955

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

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

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

Fungsi dapat diplot. Daripada ekspresi, kita hanya perlu menyediakan fungsinya nama. Berbeda dengan ekspresi simbolik atau numerik, nama fungsi harus diberikan dalam string.

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

```
0.786151377757
```

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

Anda masih dapat memanggil fungsi bawaan sebagai "...", jika fungsi tersebut ada di Euler inti.

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

```
0.707106781187
```

Sebaiknya kita menghilangkan redefinisi sin ini

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

```
0.707106781187
```

Fungsi numerik dapat memiliki parameter default.

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

Menghilangkan parameter ini akan menggunakan nilai default.

```
>f(4)
```

16

Menyetelnya akan menimpa nilai default.

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

80

Parameter yang ditetapkan akan menyimpannya juga. Ini digunakan oleh banyak fungsi Euler seperti `plot2d`, `plot3d`.

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

16

Jika suatu variabel bukan parameter, maka harus bersifat

global. Fungsi satu baris dapat melihat variabel global.

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

24

Namun parameter yang ditetapkan mengesampingkan nilai global.

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

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

20

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

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

Fungsi simbolik dapat digunakan dalam ekspresi simbolik.

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

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

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

178.635099908

Mereka dapat digunakan untuk mendefinisikan fungsi atau ekspresi simbolik lainnya.

```
>function G(x) = factor(integrate(g(x),x)); %G(c) // integrate: mengintegalkan  
>solve(%g(x),0.5)
```

0.703467422498

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

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

0.703467422498

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

625

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

Dengan &= fungsinya bersifat simbolis, dan dapat digunakan dalam ekspresi simbolik lainnya.

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

yang tidak dapat dievaluasi secara simbolis.

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

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

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

Fungsi dapat memiliki nilai default untuk parameter.

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

Sekarang fungsinya bisa dipanggil dengan atau tanpa parameter ”base”.

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

```
2
6.7
```

Selain itu, dimungkinkan untuk menggunakan parameter yang ditetapkan.

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

2

Seringkali, kita ingin menggunakan fungsi untuk vektor di satu tempat, dan untuk elemen individual di tempat lain. Hal ini dimungkinkan dengan parameter vektor.

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

Fungsi simbolik seperti ini dapat digunakan untuk variabel simbolik.

Namun fungsinya juga dapat digunakan untuk vektor numerik.

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

17

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

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

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

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

Namun tentu saja, mereka dapat digunakan dalam ekspresi simbolik atau dalam definisi fungsi simbolik.

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

Untuk meringkas

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

Ekspresi dapat diselesaikan secara numerik dan simbolis.

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

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

1.41421356237

Ini juga berfungsi untuk ekspresi simbolik. Ambil fungsi berikut.

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

Sekarang kita mencari titik yang polinomialnya adalah 2. Dalam solve(), nilai target default $y=0$ dapat diubah dengan variabel yang ditetapkan.

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

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

```
0.966715594851  
2
```

Memecahkan ekspresi simbolik dalam bentuk simbolik

mengembalikan daftar solusi. Kami menggunakan pemecah simbolis solve() yang disediakan oleh Maxima.

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

Cara termudah untuk mendapatkan nilai numerik adalah dengan mengevaluasi solusi secara numerik seperti halnya ekspresi.

```
>longest sol()
```

```
-0.6180339887498949
```

```
1.618033988749895
```

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

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

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

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

Fungsi f() dapat melihat variabel global. Namun seringkali kita ingin menggunakan parameter lokal. dengan a=3

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

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

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

2.54116291558

Ini juga berfungsi dengan ekspresi. Namun kemudian, elemen daftar bernama harus digunakan.(Lebih lanjut tentang daftar di tutorial tentang sintaks EMT).

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

2.54116291558

Menyelesaikan Pertidaksamaan

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

```
>load(fourier_elim)
```

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

```
>fourier_elim([x^2 - 1>0],[x]) // x^2-1 > 0
>fourier_elim([x^2 - 1<0],[x]) // x^2-1 < 0
>fourier_elim([x^2 - 1 # 0],[x]) // x^2-1 <> 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])
```

```
fourier_elim([max(x, y) > 6, x # 8, mabs(y - 1) > 12], [x, y])
```

```
>$fourier_elim([(x+6)/(x-9) <= 6], [x])
```

Dokumentasi inti EMT berisi pembahasan rinci tentang bahasa matriks Euler.

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

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

```
1      2
3      4
```

Hasil kali matriks dilambangkan dengan titik.

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

```
3
4
```

```
>b' // transpose b
```

```
[3, 4]
```

```
>inv(A) //inverse A
```

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

```
>A.b //perkalian matriks
```

```
    11  
    25
```

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

```
    1     0  
    0     1
```

Poin utama dari bahasa matriks adalah semua fungsi dan operator berfungsi untuk elemen

```
>A.A
```

```
      7      10  
     15     22
```

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

```
      1      4  
      9     16
```

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

```
     37     54  
     81    118
```

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

```
     37     54  
     81    118
```

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

```
1      1
1      1
```

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

```
0.333333  0.666667
0.75      1
```

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

```
-2
2.5
```

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

```
-2
2.5
```

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

```
1      0
0      1
```

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

```
    1    0  
    0    1
```

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

```
    1    4  
    9   16
```

Ini bukan hasil kali matriks, melainkan perkalian elemen demi elemen. Hal yang sama berhasil untuk vektor

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

```
    9  
   16
```

Jika salah satu operan adalah vektor atau skalar, maka operan tersebut diperluas secara alami

$>2*A$

2	4
6	8

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

$>[1,2]*A$

1	4
3	8

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

$>A*[2,3]$

2	6
6	12

Kita dapat membayangkan perkalian ini seolah-olah vektor baris v telah diduplikasi menjadi bentuk matriks yang berukuran sama dengan A .

```
>dup([1,2],2) // dup: menduplikasi/menggandakan vektor [1,2] sebanyak 2 kali (baris)
```

1	2
1	2

```
>A*dup([1,2],2)
```

1	4
3	8

Hal ini juga berlaku untuk dua vektor dimana yang satu adalah vektor baris dan yang lainnya adalah a vektor kolom. Kita menghitung $i*j$ untuk i,j dari 1 sampai 5. Caranya adalah dengan mengalikan 1:5 dengan transposnya. Bahasa matriks Euler secara otomatis menghasilkan tabel nilai.

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

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

Sekali lagi, ingatlah bahwa ini bukan produk matriks!

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

55

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

55

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

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

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

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

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

5

Euler memiliki operator perbandingan, seperti "==", yang memeriksa kesetaraan. Kita mendapatkan vektor 0 dan 1, dimana 1 berarti benar.

```
>t=(1:10)^2; t==25 //menguji elemen2 t yang sama dengan 25 (hanya ada 1)
```

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

Dari vektor tersebut, "bukan nol" memilih elemen bukan nol. Dalam hal ini, kita mendapatkan indeks semua elemen lebih besar dari 50.

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

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

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

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

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

Sebagai contoh, mari kita cari semua kuadrat bilangan 1 sampai 1000, yaitu 5 modulo 11 dan 3 modulo 13.

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

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

EMT tidak sepenuhnya efektif untuk perhitungan bilangan bulat. Ia menggunakan floating point presisi ganda secara internal. Namun, seringkali hal ini sangat berguna.

Kita dapat memeriksa primalitasnya. Mari kita cari tahu, berapa banyak persegi ditambah 1 yang merupakan bilangan prima.

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

Fungsi `nonzeros()` hanya berfungsi untuk vektor. Untuk matriks, ada `mnonzeros()`.

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

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

Ini mengembalikan indeks elemen, yang bukan nol.

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

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

Indeks ini dapat digunakan untuk mengatur elemen ke nilai tertentu.

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

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

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

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

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

Dan dimungkinkan untuk mendapatkan elemen dalam vektor.

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

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

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

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

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

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

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

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

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

```
>max(A)
```

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

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

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

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

Fungsi Matriks Lainnya (Matriks Pembangun)

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

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

1	2	3
1	2	3

Demikian pula, kita dapat melampirkan matriks ke matriks lain secara berdampingan, jika keduanya mempunyai bilangan yang sama

```
>A=random(3,4); A|v'
```

0.032444	0.0534171	0.595713	0.564454	1
0.83916	0.175552	0.396988	0.83514	2
0.0257573	0.658585	0.629832	0.770895	3

Jika jumlah barisnya tidak sama, matriks yang lebih pendek diisi dengan

Ada pengecualian untuk aturan ini. Bilangan real yang melekat pada suatu matriks akan digunakan sebagai kolom yang diisi dengan bilangan real itu.

```
>A|1
```

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

Dimungkinkan untuk membuat matriks vektor baris dan kolom

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

1	2	3
1	2	3

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

1	1
2	2
3	3

Tujuan utamanya adalah untuk menafsirkan ekspresi vektor untuk vektor kolom.

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

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

Untuk mendapatkan ukuran A, kita bisa menggunakan fungsi berikut

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

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

Untuk vektor, ada panjang()

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

Masih banyak fungsi lain yang menghasilkan matriks.

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

```
     1     1
     1     1
```

Ini juga dapat digunakan dengan satu parameter. Untuk mendapatkan vektor dengan bilangan lain selain 1, gunakan seperti berikut.

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

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

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

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

```
0.66566    0.831835
0.977      0.544258
```

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

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

1	2	3
4	5	6
7	8	9

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

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

Mari kita coba.

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

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

fungsi multdup() menduplikasi elemen vektor.

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

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

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

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

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

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

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

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

Fungsi khusus adalah `drop(v,i)`, yang menghapus elemen dengan vektor di `i` dari vektor `v`.

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

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

Perhatikan bahwa vektor `i` di `drop(v,i)` mengacu pada indeks elemen di `v`, bukan nilainya dari elemen. Jika Anda ingin menghapus elemen, Anda perlu mencari elemennya terlebih dahulu. Fungsi `indexof(v,x)` dapat digunakan untuk mencari elemen `x` dalam vektor yang diurutkan `v`

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

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

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

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

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

da beberapa fungsi khusus untuk mengatur diagonal atau menghasilkan matriks diagonal. kita mulai dari matriks identitas

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

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Kemudian kita atur diagonal bawah (-1) menjadi 1:4.

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

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

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

Berikut adalah fungsi yang mengembalikan matriks diagonal tiga

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

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

Diagonal suatu matriks juga dapat diekstraksi dari matriks tersebut. Untuk menunjukkan hal ini, kami menyusun ulang vektor 1:9 menjadi matriks 3x3

```
>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]
```

Misalnya kita dapat membagi matriks dengan diagonalnya. Bahasa matriks memperhatikan bahwa vektor kolom d diterapkan pada matriks baris demi baris

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

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

Hampir semua fungsi di Euler juga berfungsi untuk input matriks dan vektor, kapan pun hal ini terjadi nalar

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

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

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

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

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

Dengan ini dan operator titik dua a:delta:b, vektor nilai fungsi dapat dibuat dihasilkan dengan mudah. Dalam contoh berikut, kita menghasilkan vektor nilai t[i] dengan jarak 0,1 dari -1 menjadi 1. Kemudian kita menghasilkan vektor nilai fungsi.

$$s = t^3 - t$$

```
>t=-1:0.1:1; s=t^3-t
```

```
[0, 0.171, 0.288, 0.357, 0.384, 0.375, 0.336, 0.273, 0.192,  
0.099, 0, -0.099, -0.192, -0.273, -0.336, -0.375, -0.384,  
-0.357, -0.288, -0.171, 0]
```

EMT memperluas operator untuk skalar, vektor, dan matriks dengan cara yang jelas

Msalnya, vektor kolom dikalikan vektor baris diekspansi ke matriks, jika operatornya adalah terapan. Berikut ini, v' adalah vektor yang dialihkan (vektor kolom).

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

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

Perhatikan, ini sangat berbeda dengan perkalian matriks. Produk matriksnya adalah dilambangkan dengan titik "." di EMT.

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

55

Secara default, vektor baris dicetak dalam format ringkas.

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

[1, 2, 3, 4]

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

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

5
25

Untuk mengubah urutan matriks kita menggunakan apostrof.

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

```
1  
2  
3  
4
```

Jadi kita dapat menghitung matriks A dikalikan vektor b.

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

```
30  
70
```

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

```
>v'.v
```

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

$v v'$ menghitung norma v kuadrat untuk vektor baris v . Hasilnya adalah vektor 1×1 yang berfungsi sebagai bilangan real.

```
>v.v'
```

30

Ada juga fungsi norma (bersama dengan banyak fungsi Aljabar Linier lainnya)

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

30

Operator dan fungsi mematuhi bahasa matriks Euler.

Berikut ringkasan peraturannya

- Suatu fungsi yang diterapkan pada vektor atau matriks diterapkan pada setiap elemen.
- Operator yang mengoperasikan dua matriks dengan ukuran yang sama diterapkan secara berpasangan pada matriks tersebut elemen matriks.
- Jika kedua matriks mempunyai dimensi yang berbeda, maka keduanya diperluas secara sensibel caranya, agar ukurannya sama.

Misalnya, nilai skalar dikalikan vektor dengan mengalikan nilai setiap elemennya vektor. Atau matriks dikalikan dengan vektor (dengan $*$, bukan $.$) akan memperluas vektor tersebut ke ukuran element matriks dengan menduplikasinya.

Berikut ini adalah kasus sederhana dengan operator $\hat{\cdot}$.

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

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

Ini kasus yang lebih rumit. Vektor baris dikalikan vektor kolom memperluas keduanya sebesar menggan-dakan.

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

1	2	3
2	4	6
3	6	9

Perhatikan bahwa perkalian skalar menggunakan perkalian matriks, bukan *!

```
>v.v'
```

14

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

sum,prod menghitung jumlah dan produk dari baris cumsum,cumprod

melakukan hal yang sama cumulatively

computes the extremal values of each row
extrema returns a vector with the extremal information
diag(A,i) returns the i-th diagonal
setdiag(A,i,v) sets the i-th diagonal
id(n) the identity matrix
det(A) the determinant
charpoly(A) the characteristic polynomial
eigenvalues(A) the eigenvalues

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

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

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

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

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

Untuk menggabungkan matriks dan vektor terdapat operator "|" Dan "_"

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

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

Elemen-elemen matriks disebut dengan "A[i,j]".

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

6

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

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

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

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

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

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

Bentuk kependekan dari : menghilangkan indeks sepenuhnya.

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

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

Untuk tujuan vektorisasi, elemen matriks dapat diakses seolah-olah ada vektor.

```
>A{4}
```

Matriks juga dapat diratakan menggunakan fungsi `redim()`. Hal ini diterapkan difungsi `meratakan()`

```
>redim(A,1,prod(size(A))), flatten(A)
```

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

Untuk menggunakan matriks pada tabel, mari kita atur ulang ke format default, dan hitung tabelnya nilai sinus dan cosinus. Perhatikan bahwa sudut dinyatakan dalam radian secara default

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

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

Sekarang kita menambahkan kolom ke matrik

```
>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, yang setiap barisnya merupakan tabel t^i untuk satu i . Artinya, matriks memiliki elemen.

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

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

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

Untuk memastikan, Anda mendapatkan baris pertama untuk matriks $1 \times n$ dan $m \times n$, tentukan semua kolom menggunakan indeks kedua yang kosong.

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

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

*Sub-Matriks dan Elemen Matrik

Untuk mengakses elemen matriks, gunakan notasi braket.

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

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

Kita dapat mengakses baris matriks secara lengkap.

```
>A[2]
```

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

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

```
>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]

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

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

1	2	3
4	5	6

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

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

7	8	9
4	5	6
1	2	3

Trik indeks juga berfungsi dengan kolom.

Contoh ini memilih semua baris A dan kolom kedua dan ketiga.

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

2	3
5	6
8	9

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

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

3
6
9

Alternatifnya, biarkan indeks pertama kosong.

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

2	3
5	6
8	9

Kita juga bisa mendapatkan baris terakhir A.

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

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

Sekarang mari kita ubah elemen A dengan menetapkan submatriks A ke suatu nilai. Ini benar sebenarnya mengubah matriks yang disimpan A.

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

4	2	3
4	5	6
7	8	9

Kita juga dapat memberikan nilai pada baris A.

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

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

Kita bahkan dapat menetapkan sub-matriks jika ukurannya sesuai.

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

5	6	-1
7	8	6
7	8	9

Selain itu, beberapa jalan pintas diperbolehkan.

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

0	0	-1
0	0	6
7	8	9

Peringatan: Indeks di luar batas mengembalikan matriks kosong, atau pesan kesalahan, tergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Namun ingat, hal itu bersifat negatif indeks dapat digunakan untuk mengakses elemen matriks yang dihitung dari akhir.

```
>A[4]
```

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

Menyortir dan Mengacak

Fungsi `sort()` mengurutkan vektor baris.

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

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

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

Mari kita mengacak sebuah vektor.

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

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

Indeks berisi urutan `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
```

Seperti yang Anda lihat, posisi entri ganda agak acak.

```
>ind
```

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

Ini juga berfungsi untuk vektor string.

```
>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
```

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

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

```
-4  
4.5
```

Contoh lain, kita membuat matriks berukuran 200x200 dan jumlah baris-barisnya. Lalu kita selesaikan $Ax=b$ menggunakan matriks invers. Kami mengukur kesalahan sebagai deviasi maksimum semua elemen dari 1, yang tentunya merupakan solusi yang tepat.

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

```
8.790745908981989e-13
```

Jika sistem tidak mempunyai solusi, kesesuaian linier meminimalkan norma kesalahan kapak-b

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

1	2	3
4	5	6
7	8	9

penentu matriks ini adalah 0

```
>det(A)
```

0

Matriks Simbol

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

Euler untuk mendefinisikan matriks simbolik.

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

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

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

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

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

Untuk mengekstrak vektor eigen tertentu memerlukan pengindeksan yang cermat.

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

[1, - 1]

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

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

1	4
5	2

Dalam ekspresi simbolik, gunakan dengan

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

Akses ke deretan matriks simbolik berfungsi sama seperti matriks numerik.

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

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

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

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

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

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

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

```
>invert(B)()
```

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

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

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

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

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

Misalnya nilai eigen dari A dapat dihitung secara numerik

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

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

Atau secara simbolis. Lihat tutorial tentang Maxima untuk detailnya

```
>eigenvalues(A)
```

Nilai Numerik dalam Ekspresi simbolik

Ekspresi simbolik hanyalah string yang berisi ekspresi. Jika kita ingin mendefinisikan nilai 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 simbolik. Saat mentransfer matriks ke bentuk simbolik, pendekatan pecahan untuk real akan digunakan.

```
>$&A
```

untuk menghindari hal ini, ada fungsi "mxmset(variabel)"

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

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

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

Ketepatan angka floating point besar dapat diubah

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

```
3.14159265358979323846264338327950288419716939937510582097494\  
4592307816406286208998628034825342117068b0
```

variabel numerik dapat digunakan dalam ekspresi simbolik apa pun menggunakan "@var". Perhatikan bahwa ini hanya diperlukan, jika variabel telah didefinisikan dengan ":=" atau "=" sebagai variabel numerik.

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

Demo Suku Bunga

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

Asumsikan Anda memiliki modal awal sebesar 5.000 (katakanlah dalam dolar)

```
>K=5000
```

```
5000
```

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

```
>K*1.03
```

```
5150
```

Euler would understand the following syntax too.

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

```
5150
```

Euler juga akan memahami sintaks berikut

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

```
1.03  
5150
```

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

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

```
6719.58189672
```

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

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

```
6719.58
```

Bagaimana keajaiban ini terjadi? Pertama, ekspresi 0:10 mengembalikan vektor bilangan bulat

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

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

Sekarang tidak ada rumus sederhana untuk tahun ke-n, dan kita harus mengulanginya selama bertahun-tahun. Euler memberikan banyak solusi untuk ini.

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

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

```
Real 1 x 11 matrix
```

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

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

```
>short 0:10
```

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

Kemudian semua operator dan fungsi di Euler dapat diterapkan pada elemen vektor elemen. Jadi

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

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

adalah vektor faktor q^0 sampai q^{10} . Ini dikalikan dengan K , dan kita mendapatkan vektornya nilai-nilai

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

Tentu saja, cara realistis untuk menghitung tingkat suku bunga adalah dengan membulatkan kesen terdekat setiap tahunnya. Mari kita tambahkan fungsi untuk ini.

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

Mari kita bandingkan kedua hasil tersebut, dengan dan tanpa pembulatan

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

```
1271.61  
1271.6071
```

Sekarang tidak ada rumus sederhana untuk tahun ke-n, dan kita harus mengulanginya selama bertahun-tahun. Euler memberikan banyak solusi untuk ini.

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

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

```
Real 1 x 11 matrix
```

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

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

```
>VKr'
```

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

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

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

```
5150.00  
5000.00      5150.00      5304.50
```

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

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

```
6719.60  
6719.58
```

Perbedaannya sangat kecil.

Menyelesaikan Persamaan

Sekarang kita mengambil fungsi yang lebih maju, yang menambahkan tingkat uang tertentu setiap tahunnya.

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

Kita tidak perlu menentukan q atau R untuk definisi fungsi. Hanya jika kita menjalankannya perintah, kita harus mendefinisikan nilai-nilai ini. Kami memilih R=200

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

Real 1 x 11 matrix

```
5000.00    5350.00    5710.50    6081.82    ...
```

Bagaimana jika kita menghapus jumlah yang sama setiap tahun?

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

Real 1 x 11 matrix

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

Kami melihat uangnya berkurang. Jelasnya, jika kita hanya mendapat bunga sebesar 150 pada tahun pertama, namun menghapus 200, kita kehilangan uang setiap tahunnya.

Bagaimana kita dapat menentukan berapa tahun uang tersebut akan bertahan? Kita harus menulis satu lingkaran untuk ini. Cara termudah adalah dengan melakukan iterasi cukup lama.

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

```
Real 1 x 51 matrix
```

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

Dengan menggunakan bahasa matriks, kita dapat menentukan nilai negatif pertama dengan cara berikut.

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

```
48.00
```

Alasannya adalah bukan nol ($VKR < 0$) mengembalikan vektor indeks i , dengan $VKR[i] < 0$, dan $\min$ menghitung indeks minimal.

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

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

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

```
-19.83  
47.00
```

Mari kita coba menjawab pertanyaan yang lebih ambigu. Asumsikan kita mengetahui bahwa nilainya adalah 0 setelah 50 tahun. Berapa tingkat bunganya?

Ini adalah pertanyaan yang hanya bisa dijawab secara numerik. Di bawah ini, kita akan mendapatkan rumus yang diperlukan. Kemudian Anda akan melihat bahwa tidak ada rumus yang mudah untuk menentukan tingkat suku bunga. Namun untuk saat ini, kami menargetkan solusi numerik.

Langkah pertama adalah mendefinisikan fungsi yang melakukan iterasi sebanyak n kali. Kami menambahkan semua parameter ke fungsi ini.

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

Iterasinya sama seperti di atas

Namun kami tidak lagi menggunakan nilai global R dalam ekspresi kami. Fungsi seperti `iterate()` memiliki trik khusus di Euler. Anda dapat meneruskan nilai variabel dalam ekspresi sebagai parameter titik koma. Dalam hal ini P dan R.

Apalagi kami hanya tertarik pada nilai terakhir. Jadi kita ambil indeks [-1].

Mari kita coba tes.

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

-19.83

Sekarang kita bisa menyelesaikan masalah kita

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

3.15

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

Kita dapat menggunakan fungsi yang sama untuk menyelesaikan pertanyaan berikut: Berapa banyak yang dapat kita keluarkan per tahun sehingga modal awal habis setelah 20 tahun dengan asumsi tingkat bunga 3% per tahun.

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

-336.08

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

*Solusi Simbolis Masalah Suku Bunga

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

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

Sekarang kita dapat mengulanginya.

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

Kami melihat sebuah pola. Setelah n periode yang kita milik

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

Ini agak rumit. Jumlahnya dievaluasi dengan tanda "simpsum" untuk menguranginya menjadi hasil bagi.

Mari kita membuat fungsi untuk ini.

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

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

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

```
-19.82504734650985  
-19.82504734652684
```

Sekarang kita dapat menggunakannya untuk menanyakan waktu n. Kapan modal kita habis? Perkiraan awal kami adalah 30 tahun

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

20.51

jawaban ini mengatakan akan menjadi negatif setelah 21 tahun.

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

Asumsikan kita mendapatkan pinjaman sebesar K , dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama)

meninggalkan sisa hutang sebesar K_n (pada saat pembayaran terakhir). Rumusnya jelas.

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

Biasanya rumus ini diberikan dalam bentuk

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

kita dapat menyelesaikan nilai R secara simbolis

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

Seperti yang Anda lihat dari rumusnya, fungsi ini mengembalikan kesalahan floating point untuk $i=0$. Euler tetap merencanakannya.

Tentu saja, kami memiliki batasan berikut.

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

Yang jelas tanpa bunga kita harus membayar kembali 10 tarif 500.

Persamaan tersebut juga dapat diselesaikan untuk n . Akan terlihat lebih bagus jika kita menerapkan beberapa penyederhanaan padanya.

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