

EMT untuk Perhitungan Aljabar

Pada notebook ini Anda belajar menggunakan EMT untuk melakukan berbagai perhitungan terkait dengan materi atau topik dalam Aljabar. Kegiatan yang harus Anda lakukan adalah sebagai berikut:

- Membaca secara cermat dan teliti notebook ini;
- Menerjemahkan teks bahasa Inggris ke bahasa Indonesia;
- Mencoba contoh-contoh perhitungan (perintah EMT) dengan cara meng-ENTER setiap perintah EMT yang ada (pindahkan kursor ke baris perintah)
- Jika perlu Anda dapat memodifikasi perintah yang ada dan memberikan keterangan/penjelasan tambahan terkait hasilnya.
- Menyisipkan baris-baris perintah baru untuk mengerjakan soal-soal Aljabar dari file PDF yang saya berikan;
- Memberi catatan hasilnya.
- Jika perlu tuliskan soalnya pada teks notebook (menggunakan format LaTeX).
- Gunakan tampilan hasil semua perhitungan yang eksak atau simbolik dengan format LaTeX. (Seperti contoh-contoh pada notebook ini.)

Contoh pertama

Menyederhanakan bentuk aljabar:

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

Menjabarkan:

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

Baris Perintah

Baris perintah Euler terdiri dari satu atau beberapa perintah Euler 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 spasi (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 baris dihubungkan dengan "..." kedua baris akan selalu dijalankan 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 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; ...
// This line will not be visible once the cursor is off the line
```

Baris yang dimulai dengan %+ tidak akan terlihat sama sekali.

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 ...
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 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 buka dan tutup 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. Memasuki

```
\lateks: e^2 \cdot \left( \frac{3+4 \log(0.6)}{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

```
\lateks: \left(\frac{17}{18} + \frac{2}{\frac{13}{12}}\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 operator plus
- unary atau operator minus
- *, /
- . produk matriks
- a^b pangkat untuk a positif atau bilangan bulat b ($a^{**}b$ juga

berfungsi)

$N!$ 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,lantai,langit-langit,bulat,abs,tanda tangan  
conj,re,im,arg,conj,nyata,kompleks  
beta,betai,gamma,gamma kompleks,ellrf,elf,ellrd,elle  
bitand,bitor,bitxor,bitnot
```

Beberapa perintah memiliki alias, mis. `ln` untuk `log`.

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

2
0.5

```
>sin(30°)
```

0.5

Make sure to use parentheses (round brackets), whenever there is doubt about the order of execution! The following is not the same as $(2^3)^4$, which is the default for 2^3^4 in EMT (some numerical systems do it the other way).

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

2.41785163923e+24
4096
2.41785163923e+24

Bilangan Nyata

Tipe data primer pada Euler adalah bilangan real. Real direpresentasikan dalam format IEEE dengan akurasi sekitar 16 digit desimal.

```
>longest 1/3
```

0.3333333333333333

Representasi ganda internal membutuhkan 8 byte.

```
>printdual(1/3)

1.01010101010101010101010101010101010101010101010101010101010101*2^-2
```

```
>printhex(1/3)

5.555555555554*16^-1
```

Senar

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, yang dalam hal ini diubah 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 cm².

Fungsi print juga mengubah angka menjadi string. Ini dapat memerlukan sejumlah digit dan sejumlah 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 menjadi masalah. (Ini dikembalikan secara otomatis, jika fungsi tidak memiliki pernyataan return.)

```
>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
```

$\alpha = 45^\circ$

I

Di komentar, entitas yang sama seperti alpha;, beta; 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

Fungsi utf() dapat menerjemahkan string dengan entitas dalam variabel menjadi string Unicode.

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

We have $\alpha=\beta$.

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

Boolean operators obey the rules of the matrix language.

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

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

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 secara lengkap 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 "longest" 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 menggunakan di atas.

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

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+0.1-1
```

```
-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+0.1-1
```

```
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); ...
f("at*x^2",3,5) // computes 4*3^2 not 5*3^2
```

```
36.00000
```

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

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

```
45.00000
```

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); ...
f({{"at*x^2",at=5}},3)
```

```
45.00000
```

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)
```

12.00000

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

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

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

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 "@(variables) ...".

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

```
@(a,b) a^2+b^2  
41.00000
```

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; ...  
a=1.2; fx(0.5)
```

-0.47500

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

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

-0.42500

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 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!
```

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 "with".

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

120

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

Ekspresi simbolik dicetak oleh Maxima dalam bentuk 2D. Alasannya adalah adanya tanda simbolis khusus pada string tersebut.

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)
```

$$\frac{x+3}{x^2+1}$$

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

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

25

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
```

$$x^4 + 4x^3 + 6x^2 + 4x + 1$$

$$4(x+1)^3$$

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
```

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

Ekspresi simbolik dapat digunakan dalam ekspresi simbolik lainnya.

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

$$\frac{-3x^4 - 4x^3 + 1}{(x^4 + 1)^2}$$

Input langsung dari perintah Maxima juga tersedia. Mulai baris perintah dengan "::". Sintaks Maxima disesuaikan dengan sintaks EMT (disebut "mode kompatibilitas").

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

2432902008176640000

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

$$\begin{smallmatrix} 8 & 4 & 2 \\ 2 & 3 & 5 & 7 \end{smallmatrix}$$

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

$$\begin{smallmatrix} 18 & 8 & 4 & 2 \\ 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 \end{smallmatrix}$$

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

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

$$\begin{smallmatrix} 2 \\ g \end{smallmatrix}$$

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

$$\begin{smallmatrix} 3 & x \\ x & E \end{smallmatrix}$$

$$x^3 e^x$$

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

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

5
125 E

$$125 e^5$$

18551.64488782208

>fx(5)

18551.64489

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

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

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

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

2.20079141499189e+7

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

$$x (x^2 + 6x + 6) e^x$$

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)

0.20609

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

>\$&fx with x=1/2

$$\frac{\sqrt{e}}{8}$$

The assignment can also be symbolic.

>\$&fx with x=t

$$(t + 1)^3 e^{t+1}$$

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

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

$$\left[x = \frac{-\sqrt{17} - 1}{2}, x = \frac{\sqrt{17} - 1}{2} \right]$$

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

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

1.56155

Nilai numerik dari solusi simbolik dapat dihitung dengan evaluasi hasil simbolik. Euler akan membacaikan tugas $x = \text{dst}$. 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)
```

$$\left[x = -\sqrt{5} - 1, x = \sqrt{5} - 1 \right]$$

-3.23607 1.23607

$$[x = -3.23606797749979, x = 1.23606797749979]$$

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

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

$$\left[x = \frac{-\sqrt{5} - 1}{2}, x = \frac{\sqrt{5} - 1}{2} \right]$$

$$\frac{\sqrt{5} - 1}{2}$$

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

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

$$[[x = 2, y = 1], [x = 1, y = 2]]$$

2

Symbolic expressions can have flags, which indicate a special treatment in Maxima. Some flags can be used as commands too, others can't. Flags are appended with "|" (a nicer form of "ev(...,flags)")

```
>$& diff((x^3-1)/(x+1),x) //turunan bentuk pecahan
```

$$\frac{3x^2}{x + 1} - \frac{x^3 - 1}{(x + 1)^2}$$

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

$$\frac{2x^3 + 3x^2 + 1}{x^2 + 2x + 1}$$

```
>$&factor(%)
```

$$\frac{2x^3 + 3x^2 + 1}{(x + 1)^2}$$

Fungsi

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)
```

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

```
>f(2)
```

4.47214

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

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

Real 1 x 11 matrix

0.00000 0.10050 0.20396 0.31321 0.43081 0.55902 ...

Fungsi dapat diplot. Daripada ekspresi, kita hanya perlu memberikan nama fungsinya.

Berbeda dengan ekspresi simbolik atau numerik, nama fungsi harus diberikan dalam string.

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

0.78615

By default, if you need to overwrite a built-in function, you must add the keyword "overwrite". Overwriting built-in functions is dangerous and can cause problems for other functions depending on them.

You can still call the built-in function as "`_...`", if it is function in the Euler core.

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

0.70711

We better remove this redefinition of sin.

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

0.70711

Parameter Bawaan

Fungsi numerik dapat memiliki parameter default.

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

Omitting this parameter uses the default value.

```
>f(4)
```

16.00000

Setting it overwrites the default value.

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

80.00000

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

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

16.00000

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

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

24.00000

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)
```

20.00000

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)
```

$$x^3 - x e^{-x}$$

Symbolic functions can be used in symbolic expressions.

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

$$x e^{-x} - e^{-x} + 3 x^2$$

$$\frac{e^{-\frac{4}{3}}}{3} + \frac{16}{3}$$

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))
```

178.63510

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

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

$$\frac{e^{-c} (c^4 e^c + 4 c + 4)}{4}$$

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

0.70347

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

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

0.70347

```
>function P(x,n) &= (2*x-1)^n; $&P(x,n)
```

$$(2x - 1)^n$$

```
>function Q(x,n) &= (x+2)^n; $&Q(x,n)
```

$$(x + 2)^n$$

```
>$&P(x,4), $&expand(%)
```

$$(2x - 1)^4$$

$$16x^4 - 32x^3 + 24x^2 - 8x + 1$$

```
>P(3,4)
```

625.00000

```
>$&P(x,4)+ Q(x,3), $&expand(%)
```

$$(2x - 1)^4 + (x + 2)^3$$

$$16x^4 - 31x^3 + 30x^2 + 4x + 9$$

```
>$&P(x,4)-Q(x,3), $&expand(%), $&factor(%)
```

$$(2x - 1)^4 - (x + 2)^3$$

$$16x^4 - 33x^3 + 18x^2 - 20x - 7$$

$$16x^4 - 33x^3 + 18x^2 - 20x - 7$$

```
>$&P(x,4)*Q(x,3), $&expand(%), $&factor(%)
```

$$(x + 2)^3 (2x - 1)^4$$

$$16x^7 + 64x^6 + 24x^5 - 120x^4 - 15x^3 + 102x^2 - 52x + 8$$

$$(x + 2)^3 (2x - 1)^4$$

```
>$&P(x,4)/Q(x,1), $&expand(%), $&factor(%)
```

$$\frac{(2x - 1)^4}{x + 2}$$

$$\frac{16x^4}{x + 2} - \frac{32x^3}{x + 2} + \frac{24x^2}{x + 2} - \frac{8x}{x + 2} + \frac{1}{x + 2}$$

$$\frac{(2x - 1)^4}{x + 2}$$

```
>function f(x) &= x^3-x; $&f(x)
```

$$x^3 - x$$

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

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

$$\frac{x^4}{4} - \frac{x^2}{2}$$

With := the function is numerical. A good example is a definite integral like which can not be evaluated symbolically.

If we redefine the function with the keyword "map" it can be used for vectors x. Internally, the function is called for all values of x once, and the results are stored in a vector.

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

```
-0.78343 -0.41082 0.00000 0.67686 2.05045
```

Functions can have default values for parameters.

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

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

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

```
2.00000
6.70000
```

Moreover, it is possible to use assigned parameters.

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

```
2.00000
```

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

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

$$\begin{aligned} b^2 - a b + b + a^2 \\ y^2 - x y + y + x^2 \end{aligned}$$

Such a symbolic function can be used for symbolic variables.

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

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

```
17.00000
```

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

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

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

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

$$y^4 - 6x^2y^2 + x^4$$

0

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

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

$$10(y^2 + x)^3(9y^2 + x + 2)$$

Untuk meringkas

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

Memecahkan Ekspresi

Ekspresi dapat diselesaikan secara numerik dan simbolis.

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

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

1.41421

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

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

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

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

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

```
>$&solve(a*x^2+b*x+c=0,x)
```

$$\left[x = \frac{-\sqrt{b^2 - 4ac} - b}{2a}, x = \frac{\sqrt{b^2 - 4ac} - b}{2a} \right]$$

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

$$\left[\left[x = -\frac{ce}{b(d-5) - ae}, y = \frac{c(d-5)}{b(d-5) - ae} \right] \right]$$

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

$$4x^8 + x^7 - x^4 - x$$

Now we search the point, where the polynomial is 2. In `solve()`, the default target value `y=0` can be changed with an assigned variable.

We use `y=2` and check by evaluating the polynomial at the previous result.

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

0.96672
2.00000

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

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

$$\left[x = \frac{1 - \sqrt{5}}{2}, x = \frac{\sqrt{5} + 1}{2} \right]$$

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

```
>longest sol()
```

```
-0.6180339887498949 1.618033988749895
```

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

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

$$\frac{(\sqrt{5} - 1)^2}{4}$$

0

Solving systems of equations symbolically can be done with vectors of equations and the symbolic solver `solve()`. The answer is a list of lists of equations.

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

```
[[x = -1, y = 3], [x = 1, y = 1], [x = 0, y = 2]]
```

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

with `a=3`.

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

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

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

```
2.54116
```

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

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

```
2.54116
```

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\fourier_elim/fourier_elim.lisp
```

```

>$&fourier_elim([x^2 - 1 > 0], [x]) // x^2-1 > 0
[1 < x] ∨ [x < -1]

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

>$&fourier_elim([x^2 - 1 ≠ 0], [x]) // x^-1 <> 0
[-1 < x, x < 1] ∨ [1 < x] ∨ [x < -1]

>$&fourier_elim([x ≠ 6], [x])
[x < 6] ∨ [6 < x]

>$&fourier_elim([x < 1, x > 1], [x]) // tidak memiliki penyelesaian
emptyset

>$&fourier_elim([minf < x, x < inf], [x]) // solusinya R
universalset

>$&fourier_elim([x^3 - 1 > 0], [x])
[1 < x, x^2 + x + 1 > 0] ∨ [x < 1, -x^2 - x - 1 > 0]

>$&fourier_elim([cos(x) < 1/2], [x]) // ??? gagal
[1 - 2 cos x > 0]

>$&fourier_elim([y-x < 5, x - y < 7, 10 < y], [x,y]) // sistem pertidaksamaan
[y - 5 < x, x < y + 7, 10 < y]

>$&fourier_elim([y-x < 5, x - y < 7, 10 < y], [y,x])
[max(10, x - 7) < y, y < x + 5, 5 < x]

>$&fourier_elim((x + y < 5) and (x - y > 8), [x,y])
[y + 8 < x, x < 5 - y, y < -3/2]

>$&fourier_elim(((x + y < 5) and x < 1) or (x - y > 8), [x,y])
[y + 8 < x] ∨ [x < min(1, 5 - y)]

>&fourier_elim([max(x,y) > 6, x ≠ 8, abs(y-1) > 12], [x,y])
[6 < x, x < 8, y < -11] ∨ [8 < x, y < -11]
or [x < 8, 13 < y] ∨ [x = y, 13 < y] ∨ [8 < x, x < y, 13 < y]
or [y < x, 13 < y]

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

```

$$[x = 12] \vee [12 < x] \vee [x < 9]$$

Bahasa Matriks

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]
```

```
1.00000 2.00000  
3.00000 4.00000
```

The matrix product is denoted by a dot.

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

```
3.00000  
4.00000
```

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

```
3.00000 4.00000
```

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

```
-2.00000 1.00000  
1.50000 -0.50000
```

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

```
11.00000  
25.00000
```

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

```
1.00000 0.00000  
0.00000 1.00000
```

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

```
>A.A
```

```
7.00000 10.00000  
15.00000 22.00000
```

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

```
1.00000 4.00000  
9.00000 16.00000
```

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

```
37.00000 54.00000  
81.00000 118.00000
```

```

>power(A, 3) //perpangkatan matriks

37.00000 54.00000
81.00000 118.00000

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

1.00000 1.00000
1.00000 1.00000

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

0.33333 0.66667
0.75000 1.00000

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

-2.00000
2.50000

>inv(A).b

-2.00000
2.50000

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

1.00000 0.00000
0.00000 1.00000

>inv(A).A

1.00000 0.00000
0.00000 1.00000

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

1.00000 4.00000
9.00000 16.00000

```

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

```

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

9.00000
16.00000

```

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

```

>2*A

2.00000 4.00000
6.00000 8.00000

```

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

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

```
1.00000 4.00000  
3.00000 8.00000
```

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

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

```
2.00000 6.00000  
6.00000 12.00000
```

One can imagine this multiplication as if the row vector v had been duplicated to form a matrix of the same size as A.

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

```
1.00000 2.00000  
1.00000 2.00000
```

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

```
1.00000 4.00000  
3.00000 8.00000
```

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

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

```
1.00000 2.00000 3.00000 4.00000 5.00000  
2.00000 4.00000 6.00000 8.00000 10.00000  
3.00000 6.00000 9.00000 12.00000 15.00000  
4.00000 8.00000 12.00000 16.00000 20.00000  
5.00000 10.00000 15.00000 20.00000 25.00000
```

Again, remember that this is not the matrix product!

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

```
55.00000
```

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

```
55.00000
```

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

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

```
Real 1 x 10 matrix
```

```
1.00000 1.00000 1.00000 1.00000 1.00000 0.00000 ...
```

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

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

```
5.00000
```

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

We get a vector of 0 and 1, where 1 stands for true.

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

```
Real 1 x 10 matrix
```

```
0.00000 0.00000 0.00000 0.00000 1.00000 0.00000 ...
```

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

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

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

```
8.00000 9.00000 10.00000
```

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

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

```
64.00000 81.00000 100.00000
```

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

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

```
Real 1 x 28 matrix
```

```
4.00000 48.00000 95.00000 139.00000 147.00000 191.00000 ...
```

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

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

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

```
112.00000
```

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

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

```
0.76576 0.40119 0.40635 0.26783  
0.13673 0.39057 0.49598 0.95281  
0.54814 0.00608 0.44425 0.53925
```

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

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

```
1.00000 4.00000  
2.00000 1.00000  
2.00000 2.00000  
3.00000 2.00000
```

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

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

```
0.76576  0.40119  0.40635  0.00000  
0.00000  0.00000  0.49598  0.95281  
0.54814  0.00000  0.44425  0.53925
```

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

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

```
0.76576  0.40119  0.40635  -0.12692  
-0.12240 -0.69167  0.49598  0.95281  
0.54814  -0.48390  0.44425  0.53925
```

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

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

```
0.26783  0.13673  0.39057  0.00608
```

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

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

```
0.26783  4.00000  0.76576  1.00000  
0.13673  1.00000  0.95281  4.00000  
0.00608  2.00000  0.54814  1.00000
```

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

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

```
0.76576  0.95281  0.54814
```

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

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

```
0.76576  0.95281  0.54814
```

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

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

```
1.00000  1.00000  
2.00000  4.00000  
3.00000  1.00000  
-0.76576 -0.95281 -0.54814
```

Fungsi Matriks Lainnya (Matriks Bangunan)

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

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

```
1.00000 2.00000 3.00000  
1.00000 2.00000 3.00000
```

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

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

```
0.03244 0.05342 0.59571 0.56445 1.00000  
0.83916 0.17555 0.39699 0.83514 2.00000  
0.02576 0.65859 0.62983 0.77090 3.00000
```

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

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

```
>A|1
```

```
0.03244 0.05342 0.59571 0.56445 1.00000  
0.83916 0.17555 0.39699 0.83514 1.00000  
0.02576 0.65859 0.62983 0.77090 1.00000
```

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

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

```
1.00000 2.00000 3.00000  
1.00000 2.00000 3.00000
```

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

```
1.00000 1.00000  
2.00000 2.00000  
3.00000 3.00000
```

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

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

```
1.00000 1.00000  
2.00000 4.00000  
3.00000 9.00000
```

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

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

```
2.00000  
4.00000  
2.00000 4.00000  
4.00000
```

For vectors, there is length().

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

```
9.00000
```

There are many other functions, which generate matrices.

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

```
1.00000 1.00000  
1.00000 1.00000
```

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

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

```
6.00000 6.00000 6.00000 6.00000 6.00000
```

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

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

```
0.66566 0.83184  
0.97700 0.54426
```

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

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

```
1.00000 2.00000 3.00000  
4.00000 5.00000 6.00000  
7.00000 8.00000 9.00000
```

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

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

Let us test.

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

```
Real 1 x 15 matrix  
1.00000 2.00000 3.00000 1.00000 2.00000 3.00000 ...
```

The function multdup() duplicates elements of a vector.

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

```
Real 1 x 15 matrix  
1.00000 1.00000 1.00000 1.00000 1.00000 2.00000 ...  
1.00000 1.00000 2.00000 2.00000 2.00000 3.00000 3.00000
```

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

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

```
5.00000 4.00000 3.00000 2.00000 1.00000
```

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

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

```
2.00000 3.00000 4.00000 5.00000 1.00000
```

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

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

Real 1 x 10 matrix

```
10.00000 11.00000 13.00000 14.00000 15.00000 16.00000 ...
```

Note that the vector `i` in `drop(v,i)` refers to indices of elements in `v`, not the values of the elements. If you want to remove elements, you need to find the elements first. The function `indexof(v,x)` can be used to find elements `x` in a sorted vector `v`.

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

Real 1 x 15 matrix

```
2.00000 3.00000 5.00000 7.00000 11.00000 13.00000 ...  
Real 1 x 11 matrix
```

```
0.00000 5.00000 0.00000 6.00000 0.00000 0.00000 ...  
Real 1 x 11 matrix
```

```
2.00000 3.00000 5.00000 7.00000 23.00000 29.00000 ...
```

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

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

Real 1 x 8 matrix

```
1.00000 2.00000 3.00000 4.00000 6.00000 8.00000 ...
```

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

We start with the identity matrix.

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

```
1.00000 0.00000 0.00000 0.00000 0.00000  
0.00000 1.00000 0.00000 0.00000 0.00000  
0.00000 0.00000 1.00000 0.00000 0.00000  
0.00000 0.00000 0.00000 1.00000 0.00000  
0.00000 0.00000 0.00000 0.00000 1.00000
```

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

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

```
1.00000 0.00000 0.00000 0.00000 0.00000  
1.00000 1.00000 0.00000 0.00000 0.00000  
0.00000 2.00000 1.00000 0.00000 0.00000  
0.00000 0.00000 3.00000 1.00000 0.00000  
0.00000 0.00000 0.00000 4.00000 1.00000
```

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

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

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

```

2.00000 3.00000 0.00000 0.00000 0.00000
1.00000 2.00000 3.00000 0.00000 0.00000
0.00000 1.00000 2.00000 3.00000 0.00000
0.00000 0.00000 1.00000 2.00000 3.00000
0.00000 0.00000 0.00000 1.00000 2.00000

```

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

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

```

1.00000 2.00000 3.00000
4.00000 5.00000 6.00000
7.00000 8.00000 9.00000

```

Now we can extract the diagonal.

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

```
1.00000 5.00000 9.00000
```

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

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

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

Vektorisasi

Hampir semua fungsi di Euler juga berfungsi untuk input matriks dan vektor, jika hal ini masuk akal.

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

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

```
1.00000 1.41421 1.73205
```

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

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

With this and the colon operator a:delta:b, vectors of values of functions can be generated easily.

In the following example, we generate a vector of values t[i] with spacing 0.1 from -1 to 1. Then we generate a vector of values of the function

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

Real 1 x 21 matrix

```
0.00000 0.17100 0.28800 0.35700 0.38400 0.37500 ...
```

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

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

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

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

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

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

55.00000

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

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

1.00000 2.00000 3.00000 4.00000

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

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

5.00000
25.00000

To transpose a matrix we use the apostrophe.

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

1.00000
2.00000
3.00000
4.00000

So we can compute matrix A times vector b.

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

30.00000
70.00000

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

```
>v'.v
```

1.00000 2.00000 3.00000 4.00000
2.00000 4.00000 6.00000 8.00000
3.00000 6.00000 9.00000 12.00000
4.00000 8.00000 12.00000 16.00000

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

```
>v.v'
```

30.00000

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

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

30.00000

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 elemen-elemen matriks.
- Jika kedua matriks mempunyai dimensi yang berbeda, keduanya diekspansi secara wajar sehingga mempunyai ukuran yang sama.

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

Berikut ini adalah kasus sederhana dengan operator ^.

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

1.00000 4.00000 9.00000

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

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

1.00000 2.00000 3.00000
2.00000 4.00000 6.00000
3.00000 6.00000 9.00000

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

```
>v.v'
```

14.00000

There are numerous functions for matrices. We give a short list. You should to consult the documentation for more information on these commands.

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

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

1.00000 4.00000 9.00000
14.00000
1.00000 5.00000 14.00000

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

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

```
1.00000 2.00000 3.00000 4.00000  
1.00000 3.00000 5.00000 7.00000 9.00000
```

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

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

```
1.00000 2.00000 3.00000 4.00000 5.00000  
1.00000 2.00000 3.00000  
1.00000 1.00000 1.00000
```

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

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

```
6.00000
```

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

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

```
6.00000  
7.00000 8.00000 9.00000
```

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

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

```
2.00000 4.00000  
2.00000  
5.00000  
8.00000
```

A short form for : is omitting the index completely.

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

```
2.00000 3.00000  
5.00000 6.00000  
8.00000 9.00000
```

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

```
>A{4}
```

```
4.00000
```

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

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

```
Real 1 x 9 matrix
```

```
1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 ...  
Real 1 x 9 matrix  
1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 ...
```

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

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

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

Now we append columns to a matrix.

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

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

Using the matrix language, we can generate several tables of several functions at once.

In the following example, we compute $t[j]^i$ for i from 1 to n . We get a matrix, where each row is a table of t^i for one j . I.e., the matrix has the elements

A function which does not work for vector input should be "vectorized". This can be achieved by the "map" keyword in the function definition. Then the function will be evaluated for each element of a vector parameter.

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

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

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

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

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

Sub-Matrices and Matrix-Elements

To access a matrix element, use the bracket notation.

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

1	2	3
4	5	6
7	8	9
5		

We can access a complete line of a matrix.

```
>A[2]
```

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

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

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

```
2
```

To make sure, you get the first row for a $1 \times n$ and a $m \times n$ matrix, specify all columns using an empty second index.

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

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

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

Here we want the first and second row of A.

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

1	2	3
4	5	6

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

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

7	8	9
4	5	6
1	2	3

The index trick works with columns too.

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

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

2	3
5	6
8	9

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

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

3
6
9

Alternatively, leave the first index empty.

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

2	3
5	6
8	9

We can also get the last line of A.

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

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

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

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

4	2	3
4	5	6
7	8	9

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

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

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

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

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

5	6	-1
7	8	6
7	8	9

Moreover, some shortcuts are allowed.

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

0	0	-1
0	0	6
7	8	9

Peringatan: Indeks di luar batas mengembalikan matriks kosong, atau pesan kesalahan, bergantung pada pengaturan sistem. Standarnya adalah pesan kesalahan. Namun perlu diingat bahwa indeks negatif dapat digunakan untuk mengakses elemen matriks yang dihitung dari akhir.

```
>A[4]
```

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

Sorting and Shuffling

The function `sort()` sorts a row vector.

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

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

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

Let us shuffle a vector.

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

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

The indices contain the proper order of v.

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

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

This works for string vectors too.

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

a
d
e
a
aa
e

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

a
a
aa
d
e
e
```

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

```
>ind

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

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

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

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

This works for string vectors too.

```
>unique(s)

a
aa
d
e
```

Aljabar Linier

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

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

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

$$\begin{array}{r} -4 \\ 4.5 \end{array}$$

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

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

$$1.177724584522366e-12$$

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

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

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}$$

The determinant of this matrix is 0.

```
>det(A)
```

$$0$$

Matriks Simbolik

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

```
>A &= [a,1,1;1,a,1;1,1,a]; $A
```

$$\begin{pmatrix} a & 1 & 1 \\ 1 & a & 1 \\ 1 & 1 & a \end{pmatrix}$$

```
>$&det(A), &$amp;factor(%)
```

$$a (a^2 - 1) - 2 a + 2$$

$$(a - 1)^2 (a + 2)$$

```
>$&invert(A) with a=0
```

$$\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

```
>A &= [1,a;b,2]; $A
```

$$\begin{pmatrix} 1 & a \\ b & 2 \end{pmatrix}$$

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

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

$$(1-x)(2-x)-ab$$
$$\left[x = \frac{3 - \sqrt{4ab + 1}}{2}, x = \frac{\sqrt{4ab + 1} + 3}{2} \right]$$

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

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

$$[[a-1, a+1], [1, 1]]$$

To extract a specific eigenvector needs careful indexing.

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

$$[[[a-1, a+1], [1, 1]], [[[1, -1]], [[1, 1]]]]$$
$$[1, -1]$$

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

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

$$\begin{matrix} 1 & 4 \\ 5 & 2 \end{matrix}$$

In symbolic expressions, use with.

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

$$\begin{pmatrix} 1 & 4 \\ 5 & 2 \end{pmatrix}$$

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

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

$$[1, a]$$

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

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

$$\begin{pmatrix} t+1 & a \\ b & 2 \end{pmatrix}$$

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

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

$$\left[\frac{1}{j+1}, \frac{1}{j+2}, \frac{1}{j+3} \right]$$

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

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$\begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

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

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

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

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)
```

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

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

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

$$[16.1168, -1.11684, 0]$$

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

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

$$\left[\left[\frac{15 - 3\sqrt{33}}{2}, \frac{3\sqrt{33} + 15}{2}, 0 \right], [1, 1, 1] \right]$$

Nilai Numerik dalam Ekspresi simbolik

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

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

$$\begin{pmatrix} 1 & 3.14159 \\ 4 & 5 \end{pmatrix}$$

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

```
>$&A
```

$$\begin{pmatrix} 1 & \frac{1146408}{364913} \\ 4 & 5 \end{pmatrix}$$

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

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

$$\begin{pmatrix} 1 & 3.141592653589793 \\ 4 & 5 \end{pmatrix}$$

Maxima can also compute with floating point numbers, and even with big floating numbers with 32 digits. The evaluation is much slower, however.

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

1.4142135623730950488016887242097 $B \times 10^0$

1.414213562373095

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

```
>&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)}
```

-5.424777960769379

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

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

```
>K*1.03
```

5150

Euler would understand the following syntax too.

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

5150

But it is easier to use the factor

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

1.03
5150

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

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

6719.58189672

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

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

6719.58

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

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

Starting from 5000\$ you get 6719.58\$.

What if we want to know the intermediate results from year 1 to year 9? For this, Euler's matrix language is a big help. You do not have to write a loop, but simply enter

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

Real 1 x 11 matrix

5000.00 5150.00 5304.50 5463.64 ...

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

```
>short 0:10
```

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

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

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

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

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

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

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

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

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

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

1271.61
1271.6071

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

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

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

Real 1 x 11 matrix

5000.00 5150.00 5304.50 5463.64 ...

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

```
>VKr'
```

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

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

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

```
5150.00  
5000.00      5150.00      5304.50
```

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

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

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

```
6719.60  
6719.58
```

Perbedaannya sangat kecil.

Memecahkan 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 menjalankan perintah, kita harus mendefinisikan nilai-nilai ini. Kami memilih R=200.

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

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

What if we remove the same amount each year?

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

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

We see that the money decreases. Obviously, if we get only 150 of interest in the first year, but remove 200, we lose money each year.

How can we determine the number of years the money will last? We would have to write a loop for this. The easiest way is to iterate long enough.

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

Real 1 x 51 matrix

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

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

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

```
48.00
```

Alasannya adalah 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("onepay",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]
```

The iteration is just as above

But we do longer use the global value of R in our expression. Functions like iterate() have a special trick in Euler. You can pass the values of variables in the expression as semicolon parameters. In this case P and R.

Moreover, we are only interested in the last value. So we take the index [-1].

Let us try a test.

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

```
-19.83
```

Now we can solve our problem.

```
>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)
```

$$R + qK$$

We can now iterate this.

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

$$\begin{aligned} &q(q(q(R + qK) + R) + R) + R \\ &q^3R + q^2R + qR + R + q^4K \end{aligned}$$

We see a pattern. After n periods we have

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

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

$$\sum_{k=0}^{n-1} q^k = \frac{q^n - 1}{q - 1}$$

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

Let us make a function for this.

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

$$\frac{100 \left(\left(\frac{P}{100} + 1 \right)^n - 1 \right) R}{P} + K \left(\frac{P}{100} + 1 \right)^n$$

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

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

-19.82504734650985
-19.82504734652684

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

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

20.51

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 Kn (pada saat pembayaran terakhir). Rumusnya jelas

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

$$\frac{100 \left(\left(\frac{P}{100} + 1 \right)^n - 1 \right) R}{P} + K \left(\frac{P}{100} + 1 \right)^n = Kn$$

Usually this formula is given in terms of

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

$$\frac{((i+1)^n - 1) R}{i} + (i+1)^n K = Kn$$

We can solve for the rate R symbolically.

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

$$\left[R = \frac{i Kn - i (i+1)^n K}{(i+1)^n - 1} \right]$$

As you can see from the formula, this function returns a floating point error for i=0. Euler plots it nevertheless.

Of course, we have the following limit.

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

$$\lim_{x \rightarrow 0} R(5000, 0, x, 10)$$

Clearly, without interest we have to pay back 10 rates of 500.

The equation can also be solved for n. It looks nicer, if we apply some simplification to it.

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

$$\left[n = \frac{\log \left(\frac{R+i Kn}{R+i K} \right)}{\log(i+1)} \right]$$

Latihan Soal R.2

soal no. 49

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

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

$$\frac{b^{25}}{32 a^{20} c^{10}}$$

soal no. 50

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

$$\frac{q^{80}}{625 p^{16} r^{148}}$$

soal no. 90

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

soal no. 91

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

5

soal no. 92

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

-5

Latihan Soal R.3

soal no. 7

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

$$-2z + 2y + 3x - 3$$

soal no. 13

```
>$ (3*a^2)*(-7*a^4)
```

$$-21a^6$$

soal no. 15

```
>$ (6*x*y^3)*(9*(x^4)*y^2)
```

$$54x^5y^5$$

soal no. 21

```
>$ (x+6)*(x+3)
```

$$(x+3)(x+6)$$

```
>$ expand( (x+6)*(x+3))
```

$$x^2 + 9x + 18$$

```
>$ solve(%)
```

$$[x = -3, x = -6]$$

soal no. 29

```
>$ (y-5)^2
```

$$(y-5)^2$$

```
>$ expand( (y-5)^2)
```

$$y^2 - 10y + 25$$

```
>$ solve(%)
```

$$[y = 5]$$

```
>$ (y-5)^2
```

$$(y-5)^2$$

```
>$expand( (y-5)^2)
```

$$y^2 - 10y + 25$$

```
>$solve(%)
```

$$[y = 5]$$

```
>$expand( (x+1) * (x-1) * (x^2+1))
```

$$x^4 - 1$$

Latihan Soal R.4

soal no. 77

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

$$-3ab(5b - 6a)$$

soal no. 78

```
>$factor(4*x^2*y - 12*x*y^2)
```

$$-4xy(3y - x)$$

soal no. 79

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

$$(x - 4)(x^2 + 5)$$

soal no. 81

```
>$factor(8*x^2 - 32)
```

$$8(x - 2)(x + 2)$$

soal no. 101

```
>$factor(4*a*x^2 + 20*a*x - 56*a)
```

$$4a(x - 2)(x + 7)$$

Latihan Soal R.5

soal no. 31

```
>$solve(7*(3*x+6)=11-(x+2))
```

$$\left[x = -\frac{3}{2} \right]$$

soal no. 32

```
>$solve(9*(2*x+8)=20-(x+5))
```

$$[x = -3]$$

soal no. 33

```
>$solve(4*(3*y-1)-6=5*(y+2))
```

$$\left[y = \frac{20}{7} \right]$$

soal no. 34

```
>$solve(3*(2*n-5)-7=4*(n-9))
```

$$[n = -7]$$

soal no. 35

```
>$solve(x^2+3*x-28=0)
```

$$[x = 4, x = -7]$$

Latihan Soal R.6

soal no. 9

```
>$ratsimp((x^2-4)/(x^2-4*x+4))
```

$$\frac{x + 2}{x - 2}$$

soal no. 11

```
>$ratsimp((x^3-6*x^2+9*x)/(x^3-3*x^2))
```

$$\frac{x - 3}{x}$$

soal no. 10

```
>$ratsimp((x^2+2*x-3)/(x^2-9))
```

$$\frac{x - 1}{x - 3}$$

soal no. 15

```
>$ratsimp((4-x)/(x^2+4*x-32))
```

$$-\frac{1}{x + 8}$$

soal no. 12

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

$$\frac{y^3 - y^2}{y - 2}$$