# Komputeralgebra rendszerek elméleti anyag

MAPLE

## 1) Különböző egyenletek megoldásával kapcsolatos téma (fsolve, isolve, rsolve, dsolve, solve)

SOLVE:

`solve(egyenlet, melyik változóra)` (egy változónál nem kell két paraméter)
Használat:
Egyenlet, egyenletrendszer megoldására használják. A megoldásokat egymától vesszővel elválasztva kapjuk meg. Paraméteres egyenlet esetén meg kell adnunk, hogy melyik paraméterre oldja meg.
Később a paraméter behelyettesítésére is van lehetőség:
eval(egyenlet, paraméter1 =..,)
Egyenletrendszer esetén az egyenleteket és ismereteket halmazként adjuk mg, kapcsos zárójelekkel, vesszővel elválasztva:

```
a := x-1 = y
b := 3 * y = z
c := x + y = x
solve ({a, b, c}, {x, y, z});
```

FSOLVE:

`fsolve(egyenlet, melyik változóra)`
Használat:
Egyenletek közelítésére használjuk, valós számok körében. Racionális törtfüggvények esetén megadhatunk 3. paramétert, egy intervallumot, hogy hol számoljon gyököt.
A 'complex' paraméter megadása esetén komplex gyököket számol.

ISOLVE:

`isolve(egyenletek, változók)`
Használat:
Diofantoszi egyenletek megoldására alkalmas, a kimenete egész szám.
Ha nem adunk meg paramétert, akkor ő felruház egy paramétert, és megmondja, hogy az milyen lehet (egész, negatív, stb..)

RSOLVE:

`rsolve(rekurzív fv., hol szeretnénk zárt alakot létrehozni)`
Használat:
Rekurzív függvények megoldására alkalmas, rekurzív függvényt hoz zárt alakra.

DSOLVE:

`dsolve(normál diff. egyenlet)`
Használat:
Differenciálegyenleteket oldhatunk meg vele. Második paraméterben megadhatjuk az alappontot.

$$ode := \frac{d^2}{dx^2}\, y(x) = 2\,y(x) + 1$$

$$ics := y(0) = 1, D(y)(0) = 0$$

$$dsolve(\{ode, ics\})$$

$$y(x) = \frac{3}{4}e^{\sqrt{2}\,x} + \frac{3}{4}e^{-\sqrt{2}\,x} - \frac{1}{2}$$

## 2) Gráfok (létrehozása, ábrázolása, speciális gráfok, gráfok izomorfiája és gráf algoritmusok)

GraphTheory csomag ( `with(GraphTheory)` )

### Létrehozás

Az alkalmazott adatstruktúra dönti el, irányított vagy irányítatlan gráfot akarunk-e:

G := Graph(5, {{1, 2}, {1, 4}, {2, 3}, {3, 4}, {3, 5}, {4, 5}}); //irányítatlan
H := Graph(5, {[1, 2], [2, 3], [3, 4], [3, 5], [4, 1], [4, 3], [4, 5]}); //irányított
G := Graph(4, {[{1, 2}, 2], [{1, 4}, 1], [{2, 3}, 1], [{2, 4}, 1], [{3, 4}, 2]}); //irányított súllyal

Ekvivalens megadások :

```
Graph({{a, b}, {a, d}, {b, c}, {c, d}})
Graph(4, [a, b, c, d], Array(1 .. 4, [{2, 4}, {1, 3}, {2, 4}, {1, 3}]))
A := Matrix([[0, 1, 0, 1], [1, 0, 1, 0], [0, 1, 0, 1], [1, 0, 1, 0]]) ;
Graph([a, b, c, d], A)
Graph(Trail(a, b, c, d, a));
```

Trail == megadási sorrendben húzzuk be az éleket a-b-c-d-a

### Ábrázolás

`DrawGraph(gráfváltozónév, style = x),`
x = circle, tree, bipartite, spring, planar; // gráf kirajzolása
`AddEdge(G, {a, b})` G gráfba ‘a’ és ‘b’ csúcsok közé él

### Speciális gráfok:

```
PetersonGraph()
SoccerBallGraph()
```

> $P := SpecialGraphs[PetersenGraph]()$

> $DrawGraph(P)$
> $IsPlanar(P)$
> $IsVertexColorable(P, 3,'C')$
> $C$
> $HighlightVertex(P, [1, 3, 8, 10], red);\ HighlightVertex(P, [2, 4, 6], green)$
> $DrawGraph(P)$
> $CP := ChromaticPolynomial(P, \lambda)$
> $eval(CP, \lambda = 3)$
> $eval(CP, \lambda = 2)$

### Gráfok izomorfiája:

`NonIsomorphicGraphs(X, Y, restrictto, output = graphs, outputform = graph)`
//IsomorphGráfokatKreál
X = hány csúcs

Y = hány él
Restrictto = korlátozások e.g. connected, regular, regular[n]

### ALGORITMUSOK:

`MinimalSpanningTree(gráf,gráfnév(opt),animate(opt) )` // minimális feszítőfa, Kruskal-t használ

```
KruskalsAlgorithm(gráf, gráfnév(opt),animate(opt))
PrimsAlgorithm(gráf,gráfnév(opt),animate(opt))
BellmanFordAlgorithm(H<-súlyozott gráf, x, y) //Az x, y pontok
között keres az algoritmus
DijkstrasAlgorithm(todo)
ShortestPath(todo)
TopologicSort(todo)
TravelingSalesman(todo)
```

## 3) ADATSZERKEZETEK MINDEN CSOMAGBÓL (LIST, SETS, VECTOR, MÁTRIX, ARRAY, RTABLE)

### LIST

A listákat [kifejezéssorozat] alakban lehet megadni.

```
lista := [3, 4, 5, 6];
```

A lista elemeit megváltoztathatjuk:

```
lista[2] := 100;
lista;
```

Egy L Lista elemszámát a `nops(L)`, n-edik elemét `L[n]`, elemeinek sorozatát L[] vagy op(L) adja meg.

```
nops(lista);
lista[2];
lista[];
op(L);
```

Listák összefûzése az op függvénnyel történhet (vagy az L[] kifejezéssel).

```
L1 := ["a", "b", "c"];
L2 := ["c", "d", "e"];
[op(L1), op(L2)];

///edit
L := [[1, 2], [3, 4]]; Flatten(L) eredménye:  [1, 2, 3, 4]
```

### SET:

Egy halmazt kapcsos zárójelek közé tett kifejezéssorozattal adhatunk meg.

```
a := {1, 2, 3, 4};
b := {seq(x^2, x=1..5)};
```

Ha a halmaz elemeit sorozatként szeretnénk látni, használjunk szögletes zárójeleket

```
a[];
```

Halmazok unióját, metszetét, különbségét az union, intersect, minus függvények számolják.
A member függvénnyel kérdezhetjük meg, hogy valami eleme-e a halmaznak.

```
a union b;
a intersect b;
member(7, b);
```

Egy halmaz elemszámát a nops függvény adja meg.

```
nops(a);
```
Egy halmaz elemei közül bizonyos tulajdonságúakat a select ill. remove függvényekkel választhatunk ki ill. hagyhatunk el. Ez az eredeti halmazt nem változtatja meg, csak visszaad egy újat.
```
select(isprime, a);
remove(isprime, a);
```

## ARRAY

A listához hasonló, de tetszõleges egész intervallummal indexelhetõ összetett típus a tömb (array). További különbség a listához képest, hogy létrehozáskor a tömb lefoglal magának egy fix memóriaterületet, és így a tömb egy elemének változtatásakor csak az adott elemnek megfelelõ tárterület íródik felül (míg listánál egy teljesen új lista jön létre). Ez a gyakorlatban azt jelenti, hogy ha elõre tudjuk a tömb hosszát, az hatékonyabb a listánál. Tömbök létrehozására való az array parancs. Az alapváltozatban elsõ paramétere az indextartomány, második az elemek listája.
```
t1 := Array(1..5,[1,3,4,5,2]);
t2 := Array(-3..2,["a","b","c","d","e","f"]);
t1[3]; t2[-2];
```
Többdimenziós tömböket is létrehozhatunk. Ekkor az intervallumokat vesszõvel elválasztva, az elemeket pedig egymásba ágyazott listákban lehet megadni.
```
t3 := Array(1..3,1..2,[[1,2],[3,4],[5,6]]);
```
Arra is van mód, hogy a tömb létrehozásakor az elemek közül egyet sem (vagy csak néhányat) inicializálunk.
```
t4 := Array(1..2, 1..3);
t4[2,1] := 7;
eval(t4);
t5 := Array(1..2, 1..3, [(2,2)=4, (2,3)=5]);
```

## VECTOR

A vektorok lényegében egydimenziós tömbök. A vector paranccsal hozhatók létre, melynek használata hasonlít az array-hez. A vektor elemeinek indexelése mindig 1-nél kezdõdik. Az intervallum helyett tehát elég hosszt megadni, sőt, ha minden elemet megadunk egy listában, ez sem szükséges:
```
v0 := vector(4, [1,2,3,4]);
v1 := Vector([5,6,7,8]);
v2 := vector(4);
v2[3] := 333;
eval(v2);
Vector(5, symbol=v);
oszlop_vector_igy_is_megadhato := <1,2,3>;
```
Létrehozhatunk olyan vektorokat, melyeknek az i-edik eleme az i valamely függvénye (speciálisan konstans függvény). Egy i-től függő képlet megadásához használjuk a -> jelölést:
```
v3 := vector(7, 0);
v4 := vector(10, isprime);
v5 := vector(8, (i)->i^2);
```

## MÁTRIX:

A mátrixok kétdimenziós tömbök, melyekben az indexelés 1-tõl kezdõdik. Megadásuk a matrix paranccsal vagy a [<...>...] karakterekkel történik, melynek használata a vector-ral analóg.
```
m1 := Matrix(2, 3, [[1,2,3], [4,5,6]]);
```

```
m2 := Matrix([[1,2,3], [4,5,6]]);
m3 := matrix([[1,2,3], [4,5,6]]);
m4 := <1,2,3; 4,5,6>;
m5 := Matrix(4, 4, 0);
m6 := Matrix(3, 3, (i,j)->i+j-1);
m7 := Matrix(3, 3, (i,j)->i+j-1, shape=triangular);
m8 := Matrix(5, 5, m7, fill=8);
m7 := Matrix(3, 7, shape=identity);
```

Bizonyos mátrixtípusoknak beépített neve van. Például olyan mátrixot, melynek csak a fõátlójában van elem, a diag paranccsal hozhatunk létre (ehhez már kell a linalg csomag).

```
m1 := DiagonalMatrix([1, 2, 3, 4]);
```

### RTABLE(Maple helpből)

The rtable(..) function is the low level routine used by Maple to build an Array, a Matrix or a Vector. The user level commands for constructing these objects are Array(..), Matrix(..), and Vector(..), respectively. See their help pages for more information.
 Each of the parameters in the calling sequence is optional. If no parameters are provided, an empty 0-dimensional Array is returned.
If dims and init are not specified, or if only a scalar value is specified for init, a 0-dimensional Array containing a single element is constructed.

## 4) Egyváltozós kalkulus eszközeinek az ismerete (határérték, differenciálás, integrálás)

### HATÁRÉRTÉK

Kifejezések határértékét a limit paranccsal számíthatjuk ki. Első argumentuma a kifejezés, második a hely, harmadiknak megadhatjuk, hogy jobb vagy bal oldali határértéket keresünk. A végtelenre infinity néven hivatkozhatunk.

```
limit(1/x, x=-infinity);
limit(1/x, x=0);
limit(1/x, x=0, left);
```

Végtelen összegeket a sum, végtelen szorzatokat a product parancsok számolnak ki.

```
sum(1/x^2, x=1..infinity);
product((4*i^2)/(4*i^2-1), i=1..infinity);
```

Van kis és nagybetűs limit is: limit kiértékeli (megbízhatóbb), míg a Limit nem, ezutóbbi nem is ellenőrzi, hogy létezik-e a határéték.

### DIFFERENCIÁLSZÁMÍTÁS

Kifejezések deriváltját a diff, függvényekkel a D paranccsal számíthatjuk ki. A diff esetén meg kell adni, hogy mi szerint deriválunk.

```
diff(x^7, x);
diff(x^2 * y^2, x);
diff(x^2 * y^2, y);
D(x->x^7);
D(sin);
f := 'f';
g := 'g';
D(f @ g);
```

Ha a D-vel többváltozós függvényt deriválunk, szögletes zárójelben adhatjuk meg, hogy hányadik argumentum szerint deriválunk:

```
D[1]((x,y)->x^2 * y^2);
D[2]((x,y)->x^2 * y^2);
```

Többszörös deriváltaknál a változókat listában adhatjuk meg.

```
diff(x^2*y^2, [x, x, y]);
D[1,1,2]((x,y)->x^2*y^2);
```

```
diff(x^10, [x$8]);
D[1$8](x->x^10); //1$8 == 1. változó szerint 8x deriválunk
(D@@8)(x->x^10);
```

Egy kifejezés Taylor-sorának kezdetét a taylor paranccsal számíthatjuk ki. A közelítõ polinom fokát opcionális harmadik argumentumként adhatjuk meg.

taylor(sin(x), x);

```
taylor(sin(x), x, 10);
```

**INTEGRÁLÁS**

Integrálni az int paranccsal tudunk:

```
int(x^2,x);
int(sqrt(ln(x-1))/x,x);
int(x^2, x=1..10);
int(exp(-x^2), x=-infinity..infinity);
```

## 5) Animációk (Egyszerű függvény animációja, Kétváltozós függvény animációja)

Az animate és animate3d használata a plothoz hasonló, egy új koordináta, az idõ is megjelenik.

```
animate(t*cos(x),x=0..Pi,t=1..3,coords=polar,scaling=constrain
ed);
animate3d(
     [cos(v)*cos(u+t),cos(v)*sin(u+t),sin(v)],
     u=-Pi..Pi,
     v=-Pi/2..Pi/2,
     t=2*Pi/24/16..2*Pi/24,
     scaling=constrained,
     grid=[25,13]
);
```

Az animate pillanatfelvételeket készít (ezek számát a frames = x opcióval adhatjuk meg), majd mozgóképpé állítja össze. Bármilyen plot-szerû parancsot használhatunk az egyes állóképekhez, pl. az alábbi két módon:

```
animate(implicitplot, [x^2 + y^2 = t, x=-1..1, y=-1..1],
t=0..1);
kepek := [seq(implicitplot(x^2 + y^2 = t, x=-1..1, y=-1..1),
t=0..1, .1)]:
display(kepek, insequence = true);
display(kepek); # nem mozgókép
```

EGYVÁLTOZÓS FÜGGVÉNY ANIMÁCIÓJA

> $restart : with(plots) :$

> $animate(t \cdot \sin(x), x = 0 ..4 \cdot \pi, t = 1 ..4)$

> $animate([r \cdot \cos(\theta), r \cdot \sin(\theta), \theta = 0 ..2 \cdot \pi], r = 1 ..4, scaling = constrained)$ # *paraméteresen*

> $\mathrm{animatecurve}(\ \sin(x), x = 0 ..2 * \pi, \mathrm{color} = \mathrm{green}\ );$

TÖBBVÁLTOZÓS FÜGGVÉNYEK ANIMÁCIÓJA

> $restart : with(plots) :$

> $\mathrm{animate3d}(\ \sin(x-t) \cdot \cos(y-t), x = 0 ..2 \cdot \pi, y = 0 ..2 \cdot \pi, t = 0 ..\pi\ );$

> $animate3d(\ x^2 + t \cdot x \cdot y + y^2, x = -4 ..4, y = -4 ..4, t = -4 ..4, frames = 20, numpoints = 900, shading$
$= zhue, view = -4 ..4, axes = normal, orientation = [10, 70]\ );$

> $animate3d(\ [t,y,z], y=-4\,..4, z=-4\,..4, t=-3\,..3, axes=normal\ );$

## 6) Különböző ábrázolási módok (Egyváltozós függvény ábrázolása, kétváltozós függvény ábrázolása, paraméteres alakba adott függvény ábrázolása)

### Egyváltozós, többváltozós függvény ábrázolása

Függvényábrázoláshoz használhatjuk a plot parancsot. Elsõ argumentuma a kifejezés (vagy kifejezések listája, halmaza), második az intervallum. Harmadik argumentumként opcionális beállításokat adhatunk meg:

```
plot(x^2+5*x-7,x=-11..9);
plot({sin(x),cos(x)},x=-2*Pi..3*Pi);
plot(arctan,1..infinity);
plot({sin, cos},-2*Pi..2*Pi);
f := x -> x^2+x+1+4*sin(x)^2; plot(f,-2..2);
tortresz := x -> x-floor(x);
plot(tortresz, -3..3);
```

A szakadási pontokban nem húz vonalat, ha a discont=true opciót használjuk.

```
plot(tortresz, -3..3, discont=true);
plot(tan, -10..10);
```

A függõleges tengely intervallumát is megadhatjuk:

```
plot(tan, view=[-10..10, -5..5], discont=true);
plot(tan, -10..10, -5..5, discont=true);
```

További hasznos opció a scaling=constrained. Hasonlítsuk össze az alábbi két rajzot:

```
plot(sqrt(1-x^2), x=-2..2);
plot(sqrt(1-x^2), x=-2..2, scaling=constrained);
```

A feliratokat, tengelyek osztását is beállíthatjuk:

```
plot(tan, -Pi..Pi, -10..10, discont = true, xtickmarks=[-
3.14="-Pi",-1.57="-Pi/2",1.57="Pi/2",3.14="Pi"],
ytickmarks=4,
title="Tangens függvény", labels=["x","tan(x)"]);
```

Színek, vonalak stílusa.

```
pontok:= [[1,2], [1,4], [2,3], [2,-1]];
plot([sin,pontok], style=[line,point], color=[brown,green]);
```

A kiszámított pontok számát a numpoints opcióval állíthatjuk be.

```
plot( (x-25)^2/10+cos(2*Pi*x) , x=0..49 );
plot( (x-25)^2/10+cos(2*Pi*x) , x=0..49 , numpoints=2000 );
```

A plots csomag betöltése után további rajzoló függvényeket érhetünk el, pl. az implicitplot nevût.
with(plots):

```
implicitplot(x^2 + y^2 = 1,x=-1..1,y=-1..1);
```

### Egyváltozós

> $plot(x\cdot sin(3\cdot x), x=0\,..2\cdot\pi);$

> $plot(\ \{\sin(x), \sin(2\cdot x)\}, x=0\,..2\cdot\pi\ );$

### Többváltozós

> $plot3d(x\cdot\sin(y), x=-8\,..8, y=-\pi\,..\pi)$

> $plot3d(\{4-x^2-2\cdot y^2, 6-4\cdot y\}, x=-4..4, y=-3..3);$

## 7) Nem Descartes-féle koordinátarendszerek (Polár koordináták, Hengerkoordináták, Gömbkoordináták)

### Polárkoordináták

> $plot(\sin(4\cdot\theta), \theta=0..2\cdot\pi, coords=\text{polar}, scaling=constrained);$
> $plot([\cos(t), 3\cdot t, t=0..\pi], coords=\text{polar}, scaling=constrained);$
> $with(plots):$
> $polarplot\left(\cos(2\cdot\theta)\cdot\sec(\theta), \theta=-\frac{3\cdot\pi}{8}..\frac{3\cdot\pi}{8}, scaling=constrained\right)$
> $polarplot\left(\left[\cot(\theta), 3\cdot\sin(2\cdot\theta), \theta=\frac{\pi}{8}..\frac{7\cdot\pi}{8}\right], scaling=constrained\right);$

Polárkoordinátákat a coords=polar opcióval használhatunk:

```
plot( 1, 0..2*Pi, coords=polar );
plot( phi, phi=0..50, coords=polar, scaling=constrained );
plot( sin(11*x), x=0..2*Pi, coords=polar, scaling=constrained
);
S:=t->100/(100+(t-Pi/2)^8):
R:=t->S(t)*(2-sin(7*t)-cos(30*t)/2):
plot([R,t->t,-Pi/2..3/2*Pi],coords=polar,color=green,
numpoints=1000,axes=none,scaling=constrained);
```

*További opciók*

A feliratokat, tengelyek osztását is beállíthatjuk:

```
plot(
     tan, -Pi..Pi, -10..10, discont = true, xtickmarks=[-
     3.14="-Pi",-1.57="-Pi/2",1.57="Pi/2",3.14="Pi"],
     ytickmarks=4,
     title="Tangens függvény", labels=["x","tan(x)"]
);
```

Színek, vonalak stílusa.

```
pontok:= [[1,2], [1,4], [2,3], [2,-1]];
plot([sin,pontok], style=[line,point], color=[brown,green]);
```

A kiszámított pontok számát a numpoints opcióval állíthatjuk be.

```
plot( (x-25)^2/10+cos(2*Pi*x) , x=0..49 );
plot( (x-25)^2/10+cos(2*Pi*x) , x=0..49 , numpoints=2000 );
```

A plots csomag betöltése után további rajzoló függvényeket érhetünk el, pl. az implicitplot nevût.

```
with(plots):
implicitplot(x^2 + y^2 = 1,x=-1..1,y=-1..1);

implicitplot(
     //retardáltan hosszú fv innentől
 ((x/7)^2*sqrt(abs(abs(x)-3)/(abs(x)-3)) +
(y/3)^2*sqrt(abs(y+3/7*sqrt (33))/(y+3/7*sqrt(33)))-1) *
 (abs(x/2) - ((3*sqrt(33)-7)/112)*x^2 - 3 + sqrt(1-
(abs(abs(x)-2)-1)^2) - y) *
```

```
 (9*sqrt(abs((abs(x)-1)*(abs(x)-.75))/((1-abs(x))*(abs(x)-.75))) - 8*abs(x) - y) *
 (3*abs(x) + .75*sqrt(abs((abs(x)-.75)*(abs(x)-.5))/((.75-abs(x))*(abs(x)-.5))) - y) *
 (2.25*sqrt(abs((x-.5)*(x+.5))/((.5-x)*(.5+x))) -y) *
 (6/7*sqrt(10) + (1.5-0.5*abs(x))*sqrt(abs(abs(x)-1)/(abs(x)-1)) -
    6/14*sqrt(10)*sqrt(4-(abs(x)-1)^2) -y ) = 0,
     //ideáig
x=-7..7, y=-3..3, factor=true, scaling=constrained,
grid=[100,100], gridrefine=5, axes=none, color=black,
thickness=5);
```

## GÖMBI, HENGER

A 3 DIMENZIÓS RAJZOLÁS PARANCSA PLOT3D. HASZNÁLATA ÉS OPCIÓI A PLOT-HOZ HASONLÓAK.

```
plot3d((x/2)**2+(y/2)**2,x=-5..5,y=-5..5,scaling=constrained);
f := (x,y) -> x^3+x^2+1+4*sin(y)^3; plot3d(f,-2..2,-9..9);
plot3d([sin(u)*sin(v),sin(u)*cos(v),cos(u)],u=-Pi..Pi,v=-Pi..Pi,
style=wireframe, scaling=constrained);
plot3d([u*sin(v),u*cos(v),u*v],u=1..2,v=0..4*Pi,grid=[6,60]);
plot3d(sin(x*y) ,x=-2*Pi..2*Pi , y=-2*Pi..2*Pi,axes=boxed,
labels=["x változó","y változó;","érték"]);
```

Dolgozhatunk gömbi és henger koordinátarendszerben.

```
plot3d( [1,s,t], s=0..Pi, t=0..Pi, coords=spherical,
scaling=constrained, axes=normal );
spacecurve( [1,u,u], u=-10*Pi..10*Pi, coords=cylindrical,
numpoints=1000 );
```

## HENGER

> $plot3d(\theta \cdot sqrt(1-z), \theta = 0..2\cdot\pi, z = 0..1,\ coords = cylindrical, axes = normal);$

> $plot3d(2, \theta = 0..2\cdot\pi, z = -1..1, coords = cylindrical, axes = normal);$

> $plot3d([s, t, s^2 + t^2], s = -1..1, t = 0..2\cdot\pi,\ coords = cylindrical, axes = normal);$

> $plot3d([r, \theta, 12 - r^2\cdot(\cos(\theta)^2 + \sin(\theta)^2)],\ r = 1..3, \theta = 0..2*\pi, coords = cylindrical, view = 0..12,\ axes = normal);$
>
> ;

> $cylinderplot(z^2\cdot\theta, \theta = 0..5\cdot\pi, z = -1..1, grid = [80, 40]);$

> $cylinderplot([\sin(s-t), s\cdot t, s + t], s = 0..\pi, t = 0..\pi, numpoints = 1200);$