Rövid bevezetés a Maple használatába – I

Összeállította: Kovács Zoltán

2004. szeptember 8.

1. Numerikus számítások: a ♣, mint számológép

Ebben a segédanyagban a ♣ jel a „Maple” szót rövidíti.

A feladatlapokon példákat, feladatokat és szorgalmi feladatokat talál. A **példáknál** a feladatlapon a > jel után levő szöveget be kell gépelnünk a megnyitott munkalapra. A begépelt szöveg pirossal jelenik meg a képernyőn. Az ↩ jel azt jelzi, hogy a klaviatúrán meg kell nyomni az „enter” gombot. Az enter gomb megnyomására úgy kerül sor, hogy a kurzor a begépelt sor végén, vagy bárhol a begépelt sorban áll. (Az ↩ jelet néha elhagyom.) A feladatlapon általában megadom azt az eredményt is, amit a ♣ kiad. A **feladatokat** önállóan kell megoldani, s legkésőbb a feladatsor befejezésekor be kell mutatni. A **szorgalmi feladatokkal** akkor foglalkozzon, ha a teljes munkalappal készen van, s az órából még van idő. (A szorgalmi feladatok megoldása jó gyakorlás a zárthelyi dolgozatra.)

Jelen fejezetben a ♣-lel numerikus számításokat végzünk. Figyeljük meg, hogy hogyan használjuk a ♣-t

- *egzakt* eredmények illetve
- numerikus *közelítések* kiszámítására!

Emellett a ♣ kezelésére vonatkozóan is megtanulunk néhány fontos fogást:

- a beépített „Help” használatát,
- a munkalap „takarítását”,
- a teljes munkalap behelyettesítését és
- a rendszermag újraindítását.

1.1. Egzakt aritmetika ♣-lel

A ♣ direkt módon használható numerikus számolásokra: egyszerűen gépeljük be a kifejezést, zárjuk a sort *pontosvesszővel*. Az ↩ után a ♣ végrehajtja a számítást és a képernyő közepén kékkel kiírja az eredményt. ;

1. Példa:

```
> 2+4;↩
```

$$6$$

```
> 12*34567890;↩
```

$$414814680$$

Vegyük észre, hogy a szorzás jele $*$. *

Minden pirossal gépelt input sor „élő”, azaz bármikor módosíthatjuk azt. Az előbbi sorban cseréljük ki a négyest nyolcasra, majd ↩. Figyeljük meg, hogy a kék output sor az ↩ után már az új eredményt adja.

2. Példa: Számítsuk ki: 134^{39} értékét. ^

```
> 134^39;↩
```

$$90591434403147370552516385662067771291402350911187037423856474074\backslash 097423209059057664$$

```
> length(134^39);↩
```

$$83$$

Az előbbi eredmény egzakt, és 83 jegyet tartalmaz. `length()`

3. Példa: Számolás közönséges törtekkel. `/`

```
> 3/5 + 5/9 + 7/12;↩
```

$$\frac{313}{180}$$

4. Példa: A négyzetgyök használata: `sqrt()`. (Square root.) `sqrt()`

```
> sqrt(24);↩
```

$$2\sqrt{6}$$

A ♣ ugyan egyszerűbb alakra hozta a $\sqrt{24}$ számot, de az eredményt egzakt alakban hagyta.

5. Példa: Matematikai konstansok. Ha π-t akarjuk használni gépeljünk `Pi`-t. `Pi`

```
> 4*(3+Pi);↩
```

$$12 + 4\,\pi$$

A ♣ elvégezte a szorzást, de az eredmény egzakt maradt.

6. Példa: A zsebszámológépekkel ellentétben az alábbi példákban a trigonometrikus függvények értéke is egzakt marad. `sin`

```
> sin(5*Pi/3);↩
```

$$-\frac{1}{2}\sqrt{3}$$

`arcsin`

```
> arcsin(-1);↩
```

$$-\frac{1}{2}\,\pi$$

Ha nem definiált értéket próbálunk kiszámítani, akkor a ♣ hibaüzenetet ad: `tan`

```
> tan(Pi/2);↩
```

```
Error, (in tan) numeric exception: division by zero
```

1. Feladat. *Keressünk rá a Helpben a* trigonometric *szóra, s nyissuk meg a trigonometrikus függvények lapját. Tanulmányozzuk a trigonometrikus függvények bevitelét. Nyissunk új munkalapot, próbáljuk meg kiszámítani* $\sin(\pi/8)$ *pontos értékét. A Help alapján kényszerítsük ki, hogy a ♣ ezt négyzetgyökjelekkel felírja. (A margón segítséget is talál.) Most próbálkozzunk* $\sin(\pi/9)$*-el. (A sikertelenség okát majd csak algebrai* `convert`

tanulmányaink során értjük meg.)

7. Példa: A természetes alapú exponenciális függvény használata. e^x gépelése: `exp(x)`. `exp`

```
> exp(x);↩
```

$$e^x$$

Az e számot kissé bonyolultan kell meghívni: `exp(1)`

```
> exp(1);↩
```

$$e$$

8. Példa: Az abszolút érték: $|x|$ gépelése: `abs(x)`. `abs`

```
> abs(-3);↩
```

$$3$$

9. Példa: A ♣ sok számelméleti parancsot ismer. Az `ifactor()` parancs megadja egy egész szám prímtényezős szorzatra bontását. A következő példa behelyettesítése után kedvünk szerint változtassuk meg a faktorizálni kívánt egész számot. `ifactor()`

```
> ifactor(31722722304);↩
```

$$(2)^{10}\ (3)\ (7)^2\ (13)^2\ (29)\ (43)$$

2. Feladat. *Keressük ki a Helpben az* `ifactor` *parancs leírását. Nyissunk egy új munkalapot és a példákat másoljuk oda. (A Help lapon* `Edit` *menüpont* `Copy Examples`*, a megnyitott új lapon* `Edit, Paste`*). Töröljük ki az összes outputot (*`Edit, Remove Output`*), majd helyettesítsük be a teljes munkalapot! (*`Edit, Execute, Worksheet`*).*

10. Példa: Több ♣ parancsot is bevihetünk egy sorban, csak arra kell vigyáznunk, hogy mindegyik parancsot *pontosvesszővel* zárjuk.

```
> sin(Pi/3);   cos(Pi/3);   tan(Pi/3);↩
```

11. Példa: Egy számsorozat kiszámítására a `seq ()` parancsot használhatjuk. Az alábbiakban az első 100 természetes szám négyzetét számoljuk ki. (Az eredmény sort most nem gépeltem rá a feladatsorra.) `seq()`

```
> seq(k^2,k=1..100);↩
```

3. Feladat. *Írassuk ki* $\sin\left(k \cdot \frac{\pi}{8}\right)$ *értékeit gyökjelekkel* $k = 0$*-tól 8-ig, a* `seq()` *parancsot használva!*

1.2. Numerikus közelítések az `evalf( )` parancs használatával

Vessünk egy pillantást az előző szakaszban a 3. példára, majd hasonlítsuk össze az alábbiakkal: `evalf()`

12. Példa:

```
> evalf(3/5+5/9+7/12);↩
```

$$1.738888889$$

4. Feladat. *Keressük ki a helben az* `evalf()` *parancs leírását, és írassuk ki* $\sqrt{\pi}$ *értékét 500 értékes jegyre. Olvassuk el a* `Digits` *parancs helpjét is. Nyissunk új munkalapot, másoljuk át a* `Digits` *helpjének utolsó példáját, s változtassuk meg az értékes jegyek számát. > restart;↩ kiadásával indítsuk újra a rendszermagot.*

`Digits`

`restart;`

13. Példa: Ha egy eredményhez egy nevet rendelünk, akkor a későbbi használatban ez megkönnyíti az előbbi eredmény használatát. Az értékadás jele: `:=`. Ügyeljünk arra, hogy a ♣ különbséget tesz kis betű és nagy betű között!

`:=`

```
> k:=3/5+5/9+7/12;↩
```

$$k := \frac{313}{180}$$

```
> evalf(k);↩
```

$$1.738888889$$

```
> k;↩
```

$$\frac{313}{180}$$

```
> K;↩
```

$$K$$

Változónévként szavakat is használhatunk:

```
> joe:=2^5;↩
```

$$joe := 32$$

```
> sqrt(joe);↩
```

$$4\sqrt{2}$$

14. Példa:

Ha a számokat tizedes ponttal visszük be, akkor a ♣ automatikusan tizedes törtben adja az eredményt (numerikus közelítéssel).

```
> sqrt(34);↩
```

$$\sqrt{34}$$

```
> sqrt(34.0);↩
```

$$5.830951895$$

15. Példa: Az `evalf()` parancsot számsorozatra is alkalmazhatjuk.

```
> result:=seq(sqrt(k),k=1..10);↩
```

$$result := 1,\ \sqrt{2},\ \sqrt{3},\ 2,\ \sqrt{5},\ \sqrt{6},\ \sqrt{7},\ 2\sqrt{2},\ 3,\ \sqrt{10}$$

```
> evalf(result);↩
```

1., 1.414213562, 1.732050808, 2., 2.236067978, 2.449489743, 2.645751311, 2.828427124, 3., 3.162277660

1.3. A változók tisztítása

Ha létrehoztunk egy változót, akkor annak értékét a ♣ a teljes munkafolyamat során megjegyzi. Ha ugyanannak a változónak új értéket akarunk adni, akkor egyszerűen használjuk újra az értékadás `:=` parancsát.

16. Példa:

```
> restart;↩
> h;↩
```

$$h$$

```
> h:=56;↩
```

$$h := 56$$

```
> h;↩
```

$$56$$

```
> h:=sqrt(Pi);↩
```

$$h := \sqrt{\pi}$$

```
> h;↩
```

$$\sqrt{\pi}$$

17. Példa: Szükség lehet egy változó tisztítására, ami azt jelenti, hogy újra szeretnénk a szóban forgó betűt (szót) használni, de korábbi értéke nélkül. Az alábbiakban először az x változónak értéket adunk.

```
> x:=65;↩
```

$$x := 65$$

Szeretnénk létrehozni egy w-vel jelölt algebrai kifejezést:

```
> w:=x^2-4*x+7;↩
```

$$w := 3972$$

Az x nem általános változóként szerepel! Ahhoz, hogy általános változóként használjuk, „meg kell tisztítani”.

```
> x:='x';↩
```

$$x := x$$

```
> w:=x^2-4*x+7;↩
```

$$w := x^2 - 4\,x + 7$$

A `restart` parancs *minden* addig használt változót megtisztít. (Úgy működik, mintha egy teljesen új munkalapot indítottunk volna.)

1. Szorgalmi feladat. *A Help alapján ismerjünk meg néhány további számelmélettel kapcsolatos parancsot:* `igcd`, `ilcm`, `iquo`, `irem`.

2. Szorgalmi feladat. *Keressünk megoldást* ♣*-lel az alábbi kérdésre! Hány nullára végződik* $1000!$*? (A faktoriális kiszámítása egyszerűen a felkiáltójel gépelésével történik.)* !

3. Szorgalmi feladat. *Hogyan használhatjuk a* `convert()` *parancsot egy tizedes tört közönséges törtté való átalakítására? Végezzen néhány próbát. Nyisson egy új munkalapot, s végezze el az alábbiakat!*

```
> restart;↩
> convert(0.333333333,fraction);↩
> convert(0.3333333333,fraction);↩
```

Magyarázza meg a különbséget! A rövid magyarázatot szövegként írja be a munkalapba! (A szöveg begépelése előtt kattintson a `T` *ikonra a menüsorban.)*

2. Algebrai kifejezések (formulakezelés)

Ebben a fejezetben megtanulja, hogyan lehet algebrai kifejezéseket bevinni, helyettesítési értéküket kiszámítani, továbbá azt, hogyan lehet azokat manipulálni: felbontani a zárójeleket, faktorizálni, egyszerűsíteni.

2.1. A `subs()` parancs

1. Példa: Vigyük be a $3x^2 + 8$ algebrai kifejezést, és jelöljük el W-vel:

> `W:=3*x^2+8:`↩

Vegyük észre, hogy a sort most kettősponttal zártuk. Ilyenkor a ♣ végrehajtja a sort, de az eredményt nem írja ki a képernyőre. (Zárhattuk volna a sort pontosvesszővel is.) `:`

Ki akarjuk számítani az előző kifejezés értékét az $x = 4$ helyen. Erre több lehetőség is van:

> `subs(x=4,3*x^2+8);`↩

> `subs(x=4,W);`↩

(`subs`, mint *substitute*.) Figyeljünk az egyenlőségjel (`=`) és az értékadás (`:=`) közötti különbségre! `subs()`

A `subs()` parancs kifejezésekkel is működik: Helyettesítsünk be x helyére $5+2u$-t, s az eredményt jelöljük M-el:

> `M:=subs(x=5+2*u,W);`↩

$$M := 3\,(5 + 2\,u)^2 + 8$$

Az `expand` paranccsal felbonthatjuk a zárójeleket: `expand()`

> `expand(M);`↩

$$83 + 60\,u + 12\,u^2$$

2. Példa: A `subs` parancsot több változó behelyettesítésére is használhatjuk:

> `U:=(2/5)*x^2+3*y;`↩

$$U := \frac{2}{5}\,x^2 + 3\,y$$

> `subs(x=7,y=12,U);`↩

$$\frac{278}{5}$$

3. Példa: A `subs` paranccsal egyenletekbe is behelyettesíthetünk számokat. Döntsük el behelyettesítéssel, hogy $x = 3$, illetve $x = 4$ gyöke-e az $x^3-5x^2+7x-12 = 0$ egyenletnek.

> `eqn:=x^3-5*x^2+7*x-12=0;`↩

$$eqn := x^3 - 5\,x^2 + 7\,x - 12 = 0$$

Magát az egyenletet jelöltük el `eqn`-el. Figyeljünk arra, hogy az egyenlet megadásánál ismét a „szimpla” egyenlőségjelet (`=`) használtuk.

> `subs(x=3,eqn);`↩

$$-9 = 0$$

```
> subs(x=4,eqn);↩
```

$$0 = 0$$

1. Feladat. *Helyettesítsük be* -100*-tól* 100*-ig az egész számokat az* $x^3 - 5x^2 + 7x - 12$ *algebrai kifejezésbe!*

4. Példa: A `subs()` parancs „szintaktikai" helyettesítést végez: egy kifejezésben a helyettesíteni kívánt karakterek pontos egyezését keresi. Ezért a következő helyettesítés nem vezet eredményre:

```
> prod:=a^3*b^2;
```

$$prod := a^3\, b^2$$

```
> subs(a*b=1,prod);
```

$$a^3\, b^2$$

Eredményt ilyenkor algebrai helyettesítéssel érhetünk el az `algsubs()` parancs segítségével: `algsubs()`

```
> algsubs(a*b=1,prod);
```

$$a$$

A későbbiekben leírt `simplify()` parancs is célravezető.

2.2. Az `expand( )` parancs

Zárójelek felbontására az `expand` parancsot használhatjuk. Algebrai, trigonometikus és egészen általános kifejezések felbontására is használható.

5. Példa: Végezzük el a szorzásokat a $(x+2)^2\,(3\,x-3)\,(x+5)$ kifejezésben.

```
> k:=(x+2)^2*(3*x-3)*(x+5);
```

$$k := (x+2)^2\,(3\,x-3)\,(x+5)$$

```
> expand(k);
```

$$3\,x^4 + 24\,x^3 + 45\,x^2 - 12\,x - 60$$

6. Példa: A jól ismert trigonometrikus azonosságok: $\sin(2\,x)$ és $\cos(2\,x)$.

```
> expand(sin(2*x));
```

$$2\sin(x)\cos(x)$$

```
> expand(cos(2*x));
```

$$2\cos(x)^2 - 1$$

2. Feladat. *Végezzük el a beszorzást a következő kifejezésben:* $x^{(\frac{1}{2})}\,(x^{(\frac{3}{2})} + x^{(-\frac{1}{2})})$

(Akkor dolgozott helyesen, ha az eredménye $x^2 + 1$.)

3. Feladat. *Határozzuk meg az $a^{32}+b^{32}$ kifejezés minimumát, ha $a+b=1$. (Útmutatás: helyettesítsünk a helyébe $(1/2-u)$-t, b helyébe $(1/2+u)$-t, majd végezzük el a hatványozást.)*

2.3. A `factor( )` parancs

7. Példa: Alakítsuk szorzattá (azaz bontsuk elsőfokú tényezők szorzatára) a következő kifejezést: $3\,x^2-10\,x-8$ `factor`

```
> w:=3*x^2-10*x-8;
```

$$w := 3\,x^2-10\,x-8$$

```
> factor(w);
```

$$(3\,x+2)\,(x-4)$$

8. Példa: A `factor` parancsot többváltozós algebrai kifejezésekre és trigonometrikus kifejezésekre is alkalmazhatjuk.

Alakítsuk szorzattá a $x^2\,y+2\,xy+y$ kifejezést!

```
> h:=x^2*y+2*x*y+y;
```

$$h := x^2\,y+2\,x\,y+y$$

```
> factor(h);
```

$$y\,(1+x)^2$$

9. Példa: Alakítsuk szorzattá: $\sin^2 x-\cos^2 x$.

```
> factor((sin(x))^2-(cos(x)^2));
```

$$(\sin(x)-\cos(x))\,(\sin(x)+\cos(x))$$

10. Példa: A polinomok alacsonyabb fokú tényezőkre bontásánál fontos kérdés, hogy a tényezők együtthatói mely számtestben vannak. Alapértelmezésben ez a racionális számtest, így a következő polinonomot a ♣ alapértelmezésben nem tudja szorzattá alakítani, bár a polinomnak vannak valós gyökei.

```
> poly1:=x^2-2;
```

$$poly1 := x^2-2$$

```
> factor(poly1);
```

$$x^2-2$$

Opcionális argumentumként megadhatjuk, hogy a használt számtest a valós számok teste legyen. Ekkor a ♣ a faktorizációt numerikus közelítéssel hajtja végre:

```
> factor(poly1,real);
```

$$(x+1.414213562)\,(x-1.414213562)$$

★[1] Ha pontosan megadjuk, hogy a faktorizálásnál használt test $\mathbb{Q}(\sqrt{2})$ – $\mathbb{Q}$ $\sqrt{2}$-vel való bővítése – legyen, akkor egzakt eredményt kapunk.

```
> factor(poly1,sqrt(2));
```

[1] ★ arra figyelmeztet, hogy olyan elméleti hivatkozás történik, amely még az elsőéves hallgató számára nem biztos, hogy világos.

$$(x-\sqrt{2})\,(x+\sqrt{2})$$

11. Példa: Ha a `factor` parancsot törtkifejezésre alkalmazzuk, akkor a számláló és a nevező szorzattá alakítása után a ♣ még a lehetséges egyszerűsítéseket is elvégzi:

```
> A:=(x^3-7*x^2+15*x-9)/(x^2+4*x+4);
```

$$A := \frac{x^3-7\,x^2+15\,x-9}{x^2+4\,x+4}$$

```
> factor(A);
```

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

```
> B:=(x^3-7*x^2+15*x-9)/(x^2-4*x+3);
```

$$B := \frac{x^3-7\,x^2+15\,x-9}{x^2-4\,x+3}$$

```
> factor(B);
```

$$x-3$$

12. Példa: Ha nem akarjuk az egyszerűsítést elvégezni, akkor a számlálót (`numer()`) és a nevezőt (`denom()`) külön kell szorzattá alakítanunk: `numer()` `denom()`

```
> B:=(x^3-7*x^2+15*x-9)/(x^2-4*x+3);
```

$$B := \frac{x^3-7\,x^2+15\,x-9}{x^2-4\,x+3}$$

```
> factor(numer(B));  factor(denom(B));
```

$$(x-1)\,(x-3)^2$$

$$(x-1)\,(x-3)$$

2.4. A `simplify( )` parancs

13. Példa: A ♣ `simplify()` parancsával nagyon rugalmasan tudunk kifejezéseket „egyszerűbb” alakra hozni. Figyeljük meg a $\sin(3t)-\sin(7t)$ trigonometrikus kifejezés következő két átalakítását! A második esetben megadtuk az alkalmazni kívánt egyszerűsítési szabályt is. `simplify()`

```
> simplify(sin(3*t)-sin(7*t));
```

$$-20\sin(t)\cos(t)^2-64\sin(t)\cos(t)^6+80\sin(t)\cos(t)^4$$

```
> simplify(sin(3*t)-sin(7*t),{cos(t)^2=1-sin(t)^2});
```

$$64\sin(t)^7-112\sin(t)^5+52\sin(t)^3-4\sin(t)$$

4. Feladat. *Feltéve, hogy $a\cdot b=1$, írjuk fel a és b hatványainak összegeként az $(a+b)^{10}$ kifejezést!* ★*Magyarázzuk meg az eredményben a 252 számot.*

3. Oldjuk meg!

Ebben a fejezetben megtanuljuk, hogy hogyan lehet a ♣-lel egyenletek *egzakt* és *közelítő* megoldásait megkeresni, valamint egyenleteket grafikusan megoldani.

```
> restart:
> with(plots):
Warning, the name changecoords has been redefined
```

A `with(plots):` paranccsal a ♣ alkalmazható parancsait bővítettük egy *csomaggal*, nevezetesen a `plots` csomaggal. Ha a sort nem kettősponttal, hanem pontosvesszővel zárjuk le, akkor a képernyőn megkapjuk az új parancsok listáját. Erre a csomagra csak a grafikus megoldásnál lesz szükségünk. `with()`

3.1. Az egyenletek bevitele, az `lhs()` és `rhs()` parancsok

1. Példa: Egyenlet bevitelére már láttunk pédát a `subs()` parancs megismerésekor.

A következőekben bevisszük az $x^3 - 5\,x^2 + 23 = 2\,x^2 + 4\,x - 8$ egyenletet, s ennek az `eqn1` nevet adjuk:

```
> eqn1:=x^3-5*x^2+23=2*x^2+4*x-8;↩
```

Az `lhs()` és az `rhs()` parancsokkal különválaszthatjuk az egyenlet oldalait (*right hand side, left hand side*). `lhs()` `rhs()`

```
> lhs(eqn1);↩
> rhs(eqn1);↩
```

Az előbbi parancsokkal az egyenletet egy oldalra rendezhetjük:

```
> eqn2:=lhs(eqn1)-rhs(eqn1)=0;↩
```

$$eqn2 := x^3 - 7\,x^2 + 31 - 4\,x = 0$$

3.2. Egzakt megoldás keresése: a `solve( )` parancs

★Legfeljebb negyedfokú algebrai egyenletek megoldására van általános megoldóképlet, melyet a ♣ ismer. Ez a beépített algoritmus a `solve()` paranccsal hívható meg.

2. Példa: Keressük meg a következő algebrai egyenlet megoldásait:

$$3x^3 - 4x^2 - 43x + 84 = 0.$$

A `solve()` parancs második argumentuma mondja meg, hogy mely ismeretlenre kell az egyenletet megoldani.

```
> solve(3*x^3-4*x^2-43*x+84=0,x);↩
```

$$3,\ -4,\ \frac{7}{3}$$

3. Példa: Szükség lehet arra, hogy a megoldások listájából valamelyik számot kiválasszuk egy esetleges későbbi használatra. Először elnevezzük a megoldás-listát, majd ennek egy tagjára az alábbiak szerint hivatkozunk:

```
> N:=solve(x^2-5*x+3=0,x);
```

$$N := \frac{5}{2} + \frac{1}{2}\sqrt{13},\ \frac{5}{2} - \frac{1}{2}\sqrt{13}$$

```
> N[1];
```

$$\frac{5}{2} + \frac{1}{2}\sqrt{13}$$

4. Példa: Előfordulhat, hogy az egzakt megoldások nehezen olvashatók:

```
> eqn1:=x^3-34*x^2+4=0;↩
> H:=solve(eqn1,x);↩
```

★A megoldásban olvasható *I* szám a képzetes egység ($(I)^2 = -1$). Kiírathatjuk a megoldások közelítő értékét is:

```
> evalf(H);↩
```

A `solve()` parancs néha nem algebrai egyenleteknél is működik, de nem mindig adja meg a teljes megoldáshalmazt.

1. Feladat. *Oldjuk meg az* $x^3 - 11x^2 + 7x + 147 = 0$ *egyenletet, magyarázzuk meg az eredménylistát a bal oldal szorzattá alakításával.*

2. Feladat. *Írassuk ki az* $ax^2 + bx + c = 0$ *egyenlet gyökeit!*

3. Feladat. *Oldja meg az alábbi egyenletrendszert:* $x + 2y = 3, \quad y + 1/x = 1.$ *Az előbbi egyenletrendszert nevezze el* `eqns`*-nek, megoldásait* `soln`*-nak. Írassa ki külön-külön a megoldásokat, majd ellnőriztesse azokat.*

3.3. Közelítő megoldás keresése, az `fsolve()` parancs

`fsolve()`

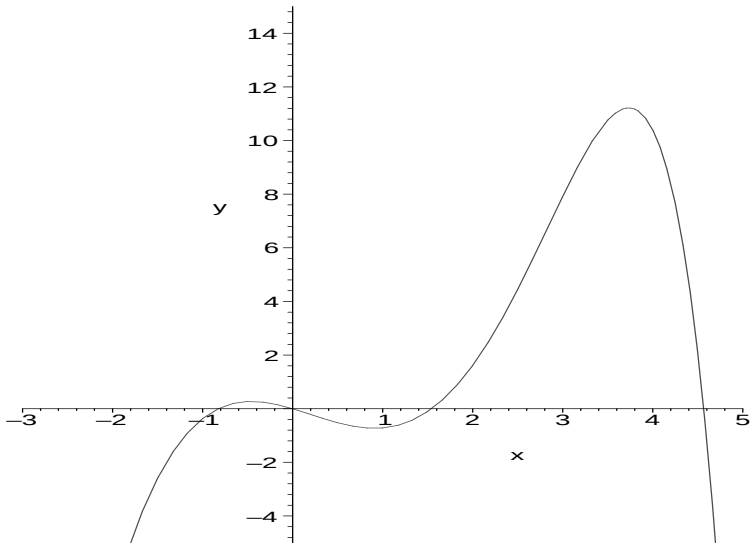
Az `fsolve()` paranccsal az algebrai egyenletek közelítő megoldásait egy lépésben megkaphatjuk, illetve kereshetjük más egyenletek közelítő megoldásait is. Olvassuk el a parancs helpjét, tanulmányozzuk az opcionális argumentumokat! Ügyeljünk arra, hogy az `fsolve()` parancs sikere a kiválasztott intervallumtól nagymértékben függ.

4. Feladat. *Keressük meg az* `fsolve` *paranccsal az* $x^3 + 1 - \exp(x) = 0$ *egyenlet összes megoldását a* $[-3, 5]$ *intervallumban!*

Az `fsolve()` paranccsal közvetlenül csak a $0.$ számot kapjuk meg, még ha az intervallumot meg is adjuk. (Hajtsuk végre!)

Ábrázoljuk a bal oldalt:

```
> plot(x^3+1-exp(x),x=-3..5,y=-5..15);
```



A grafikon szerint 4 zérushely is van. Az `fsolve()` parancs opcionális argumentumában megadható, hogy mely valós intervallumban keresse a ♣ a gyököt.

```
> fsolve(x^3+1-exp(x)=0,x=-1..-.2);
```

$$-.8251554697$$

Keressük meg a többi megoldást is! A `plot()` parancs helpje alapján készítsünk olyan ábrát is, amelyen egyszerre látható az $x \mapsto x^3+1$ és az $x \mapsto \exp(x)$ függvények grafikonja.

5. Feladat. *Kattintsunk az előbbi grafikonra bal egérgombbal! Figyeljük meg a menüsor változását, próbáljuk ki az új ikonokat! Figyeljük meg, hogy ha ismét a grafikon területére kattintunk az egérrel, akkor a kattintási pont koordinátái megjelennek a menüsorban. Most kattintsunk a grafikonra jobb egérgombbal! Tanulmányozzzuk és próbáljuk ki a legördülő menü pontjait.*

6. Feladat. *Keressük meg a következő egyenlet közelítő megoldásait:*

$$x^2/20 - 10x = 15\cos(x+15).$$

(Készítsünk ábrát, mely tartalmazza a jobb oldal és bal oldal grafikonját is, illetve olyan ábrát is, mely az egy oldalra rendezett függvény grafikonját tartalmazza – az ésszerű intervallumot határozzuk meg.)

4. Adattípusok

Ez a fejezet a legfontosabb Maple adattípusokkal foglalkozik: a *kifejezéssorozatokkal*, *listákkal* és *halmazokkal*

4.1. Kifejezéssorozatok

1. Példa: Röviden sorozatok. A sorozat vesszővel elválasztott Maple kifejezések egymásutánja:

```
> 2,1,x,sin(Pi/2);↩
```

$$2, 1, x, 1$$

A kifejezéssorozatokat összefűzhetjük:

```
> X:=a,b:Y:=1,2:X,Y;↩
```

$$a, b, 1, 2$$

1. Feladat. *Készítsünk kifejezéssorozatot az első 100 prímszámból! (Segítség a margón.) A kifejezéssorozatot nevezzük el* `primes`*-nak.* `ithprime()`

4.2. Listák

2. Példa: Ha a kifejezéssorozatot szögletes zárójelbe tesszük, akkor listát kapunk: `[]`

```
>  S:= [1,2,2,4];↩
```

S := [1, 2, 2, 4]

A lista elemeire az elem sorszámával hivatkozhatunk: `S[3]` az előbbi lista harmadik eleme.

3. Példa: A lista elemeit az `add()` paranccsal adhatjuk össze. `add()`

```
> add(i,i=S);↩
```

9

A `nops()` parancs megadja a lista elemszámát. `nops()`

2. Feladat. *Az előbbi feladatban szereplő* `primes` *kifejezéssorozatból készítsünk listát, majd határozzuk meg az első 100 prím számtani közepét.*

4. Példa: Az `op()` parancs a listát kifejezéssorozattá konvertálja: `op()`

```
>  op(S); ↩
```

$$1, 2, 2, 4$$

3. Feladat. *Legyenek az* `X` *lista elemei a négyzetszámok 1-től 100-ig, az* `Y` *lista elemei pedig a köbszámok 1-től 100-ig. Ha az* `X` *és* `Y` *listákat össze akarjuk fűzni egyetlen listává, akkor* `X,Y` *nem ad megfelelő eredményt. (Próbáljuk ki!) Hogyan járjunk el?*

4. Feladat. *Készítsen olyan listát, amelyben a* $2k+1$*-edik elem a* k*, a* $2k$*-adik elem pedig a* k*-adik prímszám,* $k = 1 \ldots 100$*.*

4.3. Halmazok

5. Példa: A Maple a halmazokat matematikai értelemben kezeli:

```
> data_set:={1,2,-1,a,1,a};↩
```

$$\text{data\_set} := \{-1, 1, 2, a\}$$

6. Példa: A Maple több halmazműveletet is támogat, beleértve az uniót és a metszetet. Például: `intersect` `union`

```
> {a,b,c} intersect {a,x,y};↩
```

$$\{a\}$$

```
> {a,b,c} union {a,x,y};↩
```

$$\{a, x, y, b, c\}$$

7. Példa: A `nops()` parancs megadja a halmaz elemszámát: `nops()`

```
> nops(data_set);↩
```

$$4$$

8. Példa: Az `op()` parancs kifejezéssorozattá konvertálja a halmazt: `op`

```
> op(data_set);↩
```

$$-1, 1, 2, a$$

9. Példa: A `map()` parancs egy függvénybe a halmaz vagy lista minden elemét behelyettesíti: `map()`

```
> numbers:={0,Pi/2,Pi,3*Pi/2,2*Pi};↩
```

$$numbers := \{0, \pi, \tfrac{1}{2}\pi, \tfrac{3}{2}\pi, 2\pi\}$$

```
> map(sin,numbers);↩
```

$$\{-1, 0, 1\}$$

Figyeljük meg, hogy az eredmény halmaz lesz! Ha a `map` parancsot listára alkalmazzuk, akkor az eredmény lista lesz:

```
> numbers:=[0,Pi/2,Pi,3*Pi/2,2*Pi]:↩
> map(sin,numbers);↩
```

$$[0, 1, 0, -1, 0]$$

5. Feladat. *Tanulmányozzuk a Helpben a* `select()` *és* `remove()` *parancsot. Az első 100 prímszám listájából válasszuk ki a 100-nál nagyobb elemeket, s hozzunk létre ebből új listát! (Útmutatás. Szüksége lehet az alábbi függvényre:* `large:=x-> is(x>100);` *Mit csinál ez a függvény? Az* `is()` *parancs helpjét is nézze meg!)* `is()`

6. Feladat. *1. Keresse ki a helpben a* `rand()` *parancsot. Hozzon létre egy X és egy Y listát, mely 150–150 véletlen számból áll, s a számok 1 és 50 között vannak.*

2. *A listák egyesítésére lehetőség a* `zip()` *parancs. Ezt használva hozzunk létre listát, amely számpárokból áll, az* i*-edik számpár pedig* $[X[i], Y[i]]$. `zip()`

3. *Az előbb egy pontsorozat koordinátapárjait kaptunk. Ábrázoljuk a pontsorozatot. (Szüksége lehet a* `plots` *csomagra.)*

4. Szorgalmi feladat. *Keressük meg a Helpben és tanulmányozzuk az összefűzés (concatenation) operátor használatát.*

5. Függvények

Ebben a fejezetben megtanulja, hogyan definiálunk függvényeket a ♣-ben, hogyan lehet függvényértéket kiszámolni, illetve a függvény gráfját ábrázolni.

```
> restart;
```

5.1. A függvény definíciója

1. Példa: A ♣ különbséget tesz a függvény és az algebrai kifejezés között. Például az

$$f\colon \mathbf{R} \to \mathbf{R}, \quad x \mapsto f(x) = \cos(\pi \cdot x) + 3$$

függvényt a következőképpen visszük be[2].

```
> f:=x->cos(Pi*x)+3;↩
```

$$f := x \to \cos(\pi\, x) + 3$$

A nyilat a „mínusz” jel és a „nagyobb” jel egymás után való gépelésével hoztuk létre. A nyíl a függvény definíciójakor kötelező. `->`

A függvény definícióját a ♣ az egész munkalap során megőrzi:

```
> f(x);↩
```

$$\cos(\pi\, x) + 3$$

Ha azt szeretnénk, hogy a ♣ felejtse el az f előbbi definícióját, akkor a változók tisztításánál megismert módon járunk el:

```
> f:='f';↩
```

$$f := f$$

5.2. Behelyettesítés függvénybe

2. Példa: Definiáljuk a következő függvényt:

```
> f:=x->3*x+x^2;
```

$$f := x \to 3\, x + x^2$$

Keressük meg a következő függvényértékeket:

```
> f(-1);
```

$$-2$$

```
> f(2+sqrt(5));
```

$$6 + 3\,\sqrt{5} + (2 + \sqrt{5})^2$$

```
> evalf(f(2+sqrt(5)));
```

$$30.65247584$$

```
> f(x+4);
```

[2] A továbbiakban a függvény értelmezési tartományát nem írom ki, az mindig a valós számok lehető legbővebb részhalmazát jelenti.

$$3\,x + 12 + (x+4)^2$$

```
> simplify(%);
```

$$11\,x + 28 + x^2$$

```
> (f(x+h)-f(x))/h;
```

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

```
> simplify(%);
```

$$3 + 2\,x + h$$

Figyeljük meg, hogy a függvényértékek kiszámításánál nincs szükségünk a `subs` parancsra.

3. Példa: Függvények kompozíciója:

```
> g:=x->cos(x)+1;
```

$$g := x \to \cos(x) + 1$$

```
> f(g(Pi/3));
```

$$\frac{27}{4}$$

```
> j:=x->g(f(x));
```

$$j := x \to \mathrm{g}(\mathrm{f}(x))$$

```
> j(x);
```

$$\cos(3\,x + x^2) + 1$$

1. Feladat. *Definiáljuk az*

$$s := t \mapsto s(t) = \frac{3 + t^2}{\sqrt{3t+1}}$$

függvényt. Számítsuk ki a következő értékeket: $s(2)$, $s(t-3)$, $s(t) - s(3)$. *Egyszerűsítsünk is!*

5.3. Függvények gráfja

4. Példa: A már megismer `plot` parancsot függvényekre is használhatjuk:

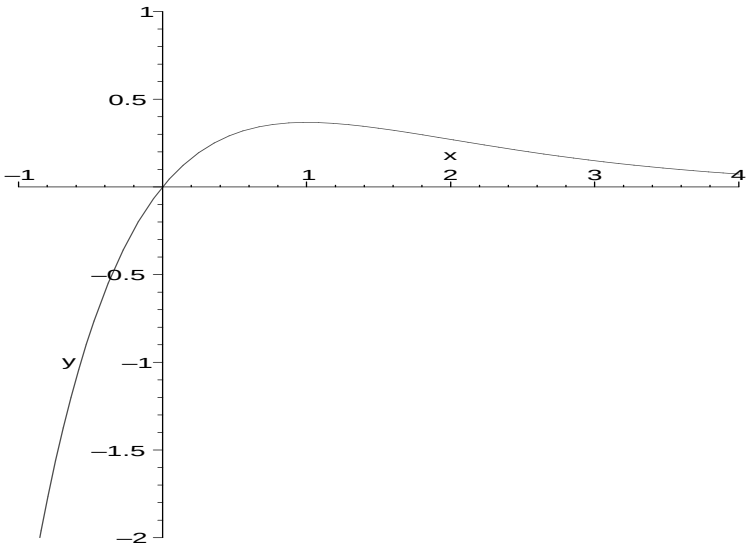
```
> h:='h';   y:='y';  x:='x';
```

$$h := h$$
$$y := y$$
$$x := x$$

```
> h:=x->x*exp(-x);
```

$$h := x \to x\,e^{(-x)}$$

```
> plot(h(x),x=-1..4,y=-2..1);
```



2. Feladat. *Tekintsük az*

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

függvényt! Ábrázoljuk közös koordinátarendszerben:

$$g(x) = f(x+1), \quad h(x) = f(x-1), \quad j(x) = f(x) + 1.$$

(Azonosítsuk be a függvényeket!)

3. Feladat. *Definiáljuk az*

$$f(x) = 2x - |x^2 - 5|$$

függvényt!

1. *Hozzuk egyszerűbb alakra az $f(z-4)$ kifejezést!*
2. *Rajzoljuk ki f gráfját!*
3. *Oldjuk meg az $f(x) = 0$ egyenletet! (Közelítő megoldás.)*

6. Eljárások szervezése

A ♣ eljárás lényegében ♣ parancsok olyan csoportja, melyet együtt hajtunk végre.

Egy eljárás alapelemei

1. Példa: Figyeljük meg hogyan szervezzük meg az eljárást: `proc()`

```
>half := proc(x)
>      evalf(x/2);
>end proc;↩
          half := proc (x) evalf(1/2*x) end proc
```

Ez az egyszerű program egyetlen Maple parancsot hajt végre: `evalf(x/2)`, azaz x szám felének közelítő értékét adja meg.

- Az eljárás neve `half`.
- Az eljárás definíciója a `proc()` utasítással kezdődik. A zárójelbe kerül, hogy milyen input adatot vár az eljárás. Ha több input adat van, akkor azokat vesszővel választjuk el.
- ; választja el az eljárásba szervezett parancsok sorozatát.
- `end proc;` zárja az eljárást.

Ha egyszer definiáltunk egy eljárást, akkor a továbbiakban Maple parancsként használhatjuk:

```
> half(2/3);↩
                          .3333333333
```

Lekérdezhetjük az egyszer már definiált eljárás definícióját:

```
showstat(half);↩
```

1. Feladat. *Írjunk eljárást, mely kiszámítja az input szám négyzetét!*

2. Feladat. *Írjunk eljárást, mely kiírja az (a, b) vektor eukildészi és taxis normáját. (*$\|(a,b)\|_E = \sqrt{a^2+b^2}$*,* $\|(a,b)\|_T = |a| + |b|$*) Írjunk olyan eljárást, amely kiszámítja egy tetszőleges számlista euklidészi normáját, azaz az elemek négyzetösszegéből vont négyzetgyököt.*

Lokális és globális változók

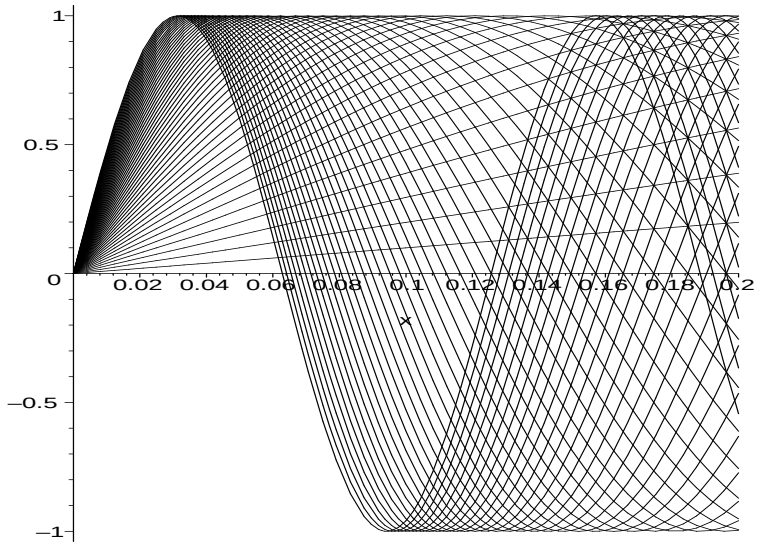
Az interaktív szinten használt változók a *globális* változók, az eljárásokban használt változók a *lokális* változók.
2. Példa: Tekintsük az alábbi eljárást:

```
> sqrs:=proc(n)
> seq(n^i,i=1..4)
> end;↩
Warning, ‘i‘ in call to ‘seq‘ is not local
        sqrs := proc(n) seq(n^i, i = 1 .. 4) end
```

Itt az i lokális változó, s a hibaüzenet figyelmeztet, hogy ezt deklarálni kell: `local`

```
> sqrs:=proc(n)
> local i;
> seq(n^i,i=1..4)
> end;
```

3. Feladat. *Írjunk eljárást, mely ugyanabban a koordinátarendszerben 1-től n-ig kirajzolja az $x \mapsto \sin(n \cdot x)$ függvényeket! ($n \in \mathbf{N}$.) Valami ilyesmit fog kapni:*



A változó típusa

3. Példa: A következő eljárás kiírja az n első m hatványát:

```
> sqrs:=proc(n,m)
> local i;
```

```
> seq(n^i,i=1..m)
> end;
```

Próbáljuk ki input adatként a $(2, 4.78)$ számpárt is...

A `seq` parancsban i helyén nyilván egész számnak kell szerepelnie: `::`

```
> sqrs:=proc(n,m::integer)
```

4. Feladat. *Nézzük meg a Help-ben a* `type/integer` *címszót, tanulmányozzuk a változók további lehetséges típusait!*

5. Feladat. *Írjunk eljárást, mely n-től m-ig prímhatványok szorzatára bontja az egész számokat. (Az n és m változók típusát deklaráljuk!)*

Ciklusok

A ♣ más programozási nyelvekhez hasonlóan biztosítja a ciklusok szervezését, azaz bizonyos utasítások ismételt végrehajtásának lehetőségét.

4. Példa: Tanulmányozzuk a `for` utasítással szervezett ♣ ciklus jellemzőit: `for`

```
> for i from 1 to 10 do i*i; od;↩
```

– `i` a *ciklusváltozó*.

– A ciklus elején áll a ciklus *leállási feltétel*e; jelen esetben a ciklus leáll, ha a ciklusváltozó elérte a 10-et. (10-re még végrehajtódik a ciklus.)

– `do` és `od` között van a ciklus *mag*ja, ezeket az utasításokat hajtja végre a ♣ a ciklusváltozó minden értékére. (Az `od` helyett lehetséges az `end do` is.)

5. Példa: A következő példában egy másik leállási feltételt tanulmányozhatunk. A `while feltétel` használata esetén a `feltétel` minden ciklus *előtt* kiértékelődik, és ha a feltétel *igaz*, akkor a ♣ végrehajtja a ciklusmag utasításait: `while`

```
> for i from 1 to 10 while i*i<50 do i*i; od;
```

6. Feladat. *Tanulmányozzuk a Help-ben a* `for` *ciklust! (Hajtsuk végre és magyarázzuk meg a példákat!)*

Most lássunk egy – nem túl összetett – példát eljárás szervezésére.

6. Példa: Írjunk olyan eljárást, mely felsorolja az n jegyű páratlan számokat.

A feladatra több megoldás is adható, egy lehetséges a következő. Először a feladatot interaktívan oldom meg n egy konkrét értékére, mondjuk 2-re. A számokat egy `lista` elnevezésű kifejezéssorozatba rendezem. A `lista` kezdetben üres, s ciklikusan bővítem úgy, hogy mindig hozzáveszem a következő, a feltételeknek eleget tevő páratlan számot, először a 11-t.

```
> restart;
> lista:=op({});
```

$$lista :=$$

```
> for i from 11 by 2 while i<100 do
> lista:=lista,i od:↩
> lista;↩
```

A programozásban járatlanok figyelmét felhívom a `lista:=lista,i` utasításra, amely azt jelenti, hogy a lista változó új értékét úgy kapom, hogy a régi lista mellé még odaírom `i`-t. (Értékadás.)

Miután az interaktív megoldás sikeres volt, megszervezem az eljárást. Bemeneti adat a számjegyek száma, mely egy pozitív egész.

```
> ptlanok:=proc(n::posint)
> local lista,i;
> lista:=op({});
> for i from 10^(n-1)+1 by 2 while i<10^n do
> lista:=lista,i od:
> lista;
> end proc;↩
```

Nézzük eljárásunkat $n = 2$-re:

```
> ptlanok(2);↩
```

Az eljárás megszervezésekor nem voltunk elég körültekintőek, $n = 1$-re nem azt adja, amit kérünk:

```
> ptlanok(1);↩
```

$$2,\ 4,\ 6,\ 8$$

A javítás:

```
> ptlanok:=proc(n::posint)
> local lista,i;
> lista:=op({});
> if n=1 then lista:=1,3,5,7,9:  else
> for i from 10^(n-1)+1 by 2 while i<10^n do
> lista:=lista,i od:fi:
> lista;
> end proc;↩
> ptlanok(1);↩
```

$$1,\ 3,\ 5,\ 7,\ 9$$

7. Feladat. *Írassuk ki a háromjegyű ikerprím párokat! Írjunk olyan eljárást, mely kiíratja az n-jegyű ikerprímeket! Írjunk olyan eljárást, mely meghatározza az n-jegyű ikerprímek számát/számtani közepét.*

8. Feladat. *Írjunk olyan eljárást a* `for` *ciklus segítségével, mely kiszámítja az első n természetes szám négyzetösszegét.*

9. Feladat. *Írjunk eljárást a* `while` *leállási feltétel segítségével, mely egy egész számot addig oszt kettővel, ameddig csak egész szám a hányados, s kiírja az elvégzett osztások számát.*

Feltételvizsgálat

10. Feladat. *Tanulmányozzuk a Help-ben az* `if then else` *feltételvizsgálatot.*

11. Feladat. *Írjunk olyan eljárást, mely az x abszolút értékét adja meg, ha x valamilyen szám ellenkező estben kiírja hogy* `ABS(x)`. *(Működik-e az eljárása komplex számokra?)*

12. Feladat. *Írjunk eljárást a következő valós függvény értékének kiszámítására: $f(x) = 0$, ha $x \leq 0$; $f(x) = 1$ ha $x > 0$.*

Rekurzió

Olyan eljárást is írhatunk mely önmagára hivatkozik. (Ezt nevezzük *rekurzió*nak.)
7. Példa: Gépeljük be az alábbi eljárást, mely az n. Fibonacci számot írja ki:

```
> Fibonacci:=proc(n::nonnegint)
> if n<2 then
> n;
> else
> Fibonacci(n-1)+Fibonacci(n-2);
> fi;
> end;
```

A voltaképpeni rekurzió a program ötödik sora, ahol az n-edik Fibonacci számot visszavezettük az $(n-1)$-edik és az $(n-2)$-edik összegére.

Nézzük meg, mennyi idő alatt számol ki az eljárás egy Fibonacci számot:

```
> time(Fibonacci(25));↩
```

A viszonylag hosszú számolás oka, hogy az eljárás újra és újra „elölről” kezdi a számolást, egy értéket többször is kiszámol. Az eljárásban elhelyezett `remember` opció utasít arra, hogy az egyszer már kiszámolt értékeket a program raktározza. Az eljárás második sorába illesszük be az alábbi utasítást:

```
> option remember;
```

Próbáljuk ki a futást a módosított eljárással. (Az időadatot nyugodtan lekérdezhetjük akár a 2000. Fibonacci számra is.)

13. Feladat. *Írjunk rekurzív eljárást a binomiális együtthatók kiszámítására.*

További feladatok

14. Feladat. *Írjunk eljárást, mely két pozitív egészre kiírja az euklidészi algoritmus maradékait.*

15. Feladat. *Írjunk eljárást két pozitív egész legnagyobb közös osztójának meghatározására. (Útmutatás: az euklidészi algoritmus utolsó nullától különböző maradékáról van szó.)*

16. Feladat. *Írjunk eljárást, mely egy elemről eldönti, hogy hanyadik elem egy listában.*

17. Feladat. *Írjunk eljárást, mely egy számlistából kiválasztja (*`select`*) az n-nél nagyobb elemeket.*

Útmutatás. Szüksége lehet az alábbi függvényre: `large:=x-> is(x>n);`

18. Feladat. *Írjon eljárást, mely kiszámítja egy polinom euklidészi normáját.*

5. Szorgalmi feladat. *A Fibonacci számok kielégítik az alábbi relációt:* $F(2n) = 2F(n-1)F(n) + F(n)^2 \quad (n > 1)$ *és* $F(2n+1) = F(n+1)^2 + F(n)^2 \quad (n > 1)$. *Írjunk rekurzív eljárást a Fibonacci számok kiszámítására ezekkel a relációkkal.*

6. Szorgalmi feladat. *Írjunk ciklus verziót az n. Fibonacci szám kiszámítására. Hasonlítsuk össze a rekurzív verzió és a ciklus verzió futásidejét!*

7. Szorgalmi feladat. *Írjunk eljárást, mely egy polinom komplex gyökeit ábrázolja a komplex számsíkon.*

7. Mátrixok, mátrix aritmetika

```
> restart;
> with(linalg):
```

7.1. Mátrixok megadása

Egy mátrix megadására több lehetőség is kínálkozik:
1. Példa: A típus megadása (sor, oszlop sorrendben), majd az elemek soronkénti felsorolása egyetlen listában. `matrix`

```
> A:=matrix(2,2,[1,-2,0,3]);
```

$$A := \begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix}$$

Hivatkozás a mátrix egy elemére:

```
> d:=A[1,2];
```

$$d := -2$$

Megadhatjuk a mátrixot a sorvektorok listájával:

```
> A:=matrix([[1,-2],[0,3]]);
```

$$A := \begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix}$$

Vagy a sorvektorok listáját előbb külön megadva egy kettős listában:

```
> ll:=[[1,-2],[0,3]];
```

$$ll := [[1, -2], [0, 3]]$$

Majd mátrixszá konvertálva:

```
> convert(ll,matrix);
```
`convert`

$$\begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix}$$

Megadhatjuk, hogy az i-edik sor j-edik elemét mely függvény szerint képezzük:

```
> matrix(3,3,(i,j)->i-j);
```

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

1. Feladat. *Adjuk meg mindegyik tanult módon a 4×4-típusú egységmátrixot!*

2. Feladat. *Tanulmányozzuk a* `diag` *parancs helpjét! Ennek alapján is adjuk meg a 4×4 típusú egységmátrixot, majd írjunk eljárást, mely bemeneti adata az $n \in \mathbf{N}$ természetes szám és kimenete az $n \times n$-típusú egységmátrix.* `diag`

3. Feladat. *Állítsuk elő $\mathbf{R}^2$ origó körüli α szögű elforgatásának mátrixát! ★A beépített teszttel ellenőrizzük, hogy ortogonális mátrixot kaptunk-e. (Segítség a margón.) Mi a geometriai jelentése annak a mátrixnak, amelyet úgy kapunk, hogy az előző mátrix második oszlopát -1-el megszorozzuk?* `orthog`

4. Feladat. *Legyen $n = 4$. Adjuk meg a következő 4×4-típusú (a_{ij}) mátrixot:*

$$a_{ij} = \begin{cases} 1, & \text{ha } j \equiv i+1 \pmod{n} \\ 0, & \text{egyébként} \end{cases}$$

Segít a `mod` *parancs.* `mod`

8. Szorgalmi feladat. *Írjunk eljárást, mely az előző feladat definícióival $n \times n$ típusú mátrixot generál. (Tehát n az input adat.)*

2. Példa: Legyen az A mátrix ugyanaz, mint a fejezet első példájában. (Egy `restart` is jól jöhet.) A elemeit soroljuk fel egyetlen kifejezéssorozatban!

Először meghatározzuk a sorvektorok listáját:

```
> ll:=convert(A,listlist);
```

`listlist`

$$ll := [[1, -2], [0, 3]]$$

Megállapítjuk a sorvektorok számát, majd generáljuk a mátrixelemek kifejezéssorozatát.

```
> n:=nops(ll);
```

$$n := 2$$

```
> l:=seq(op(ll[i]),i=1..n);
```

$$l := 1, -2, 0, 3$$

Állapítsuk meg a mátrix legnagyobb, illetve legnagyobb abszolút értékű elemét:

```
> max(l);
```

$$3$$

```
> absl:=map(abs,[l]);
```

$$absl := [1, 2, 0, 3]$$

```
> max(op(absl));
```

$$3$$

5. Feladat. *Írjunk eljárást, mely megadja egy tetszőleges mátrix legnagyobb abszolút értékű elemét!*

9. Szorgalmi feladat. *Az $A = (a_{ij})$ $m \times n$ típusú mátrix* normáját *értelmezzük a következőképpen:*

$$\|A\| = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|.$$

(Azaz az összes mátrixelem abszolút érékeinek összege. Más mátrixnormák is használatosak, a `norm` *helpjében megtaláljuk a beépített mátrixnormák listáját.) Írjunk eljárást, mely tetszőleges mátrixra kiszámítja az előbb értelmezett mátrixnormát.*

10. Szorgalmi feladat. *Írjunk olyan eljárást, mely megállapítja egy mátrix típusát!*

11. Szorgalmi feladat. *Írjunk olyan eljárást, mely egy mátrixot a sorvektorok listájává, egy kettős listát pedig mátrixszá konvertál. Ha a bemenet nem mátrix és nem kettős lista, akkor a bemenetet változtatás nélkül adjuk vissza. Kettős lista esetén ellenőrizzük, hogy a konverzió lehetséges-e, s ha nem, akkor a bemenetet szintén változtatás nélkül adjuk vissza.*

12. Szorgalmi feladat. *Írjunk eljárást, mely egy listát minden lehetséges módon mátrixszá konvertál.* – **Ezen feladat megoldásáért jutalom(pont) jár.**

7.2. Mátrix aritmetika

3. Példa: Az A mátrix ismét legyen az, ami az első példában, a B mátrix pedig a következő:

```
> B:=matrix([[-3,4],[2,1]]);
```

$$B := \begin{bmatrix} -3 & 4 \\ 2 & 1 \end{bmatrix}$$

Próbáljuk meg A-t és B-t összeadni:

```
> A+B;
```

$$A + B$$

Ehelyett az `evalm(A+B);` parancsot kell alkalmazni. `evalm`

```
> evalm(A+B);
```

$$\begin{bmatrix} -2 & 2 \\ 2 & 4 \end{bmatrix}$$

A skalárral való szorzás:

```
> evalm(2*B);
```

$$\begin{bmatrix} -6 & 8 \\ 4 & 2 \end{bmatrix}$$

Mátrixegyenlet megoldása:

```
> evalm(solve(3*X+A=B,X));
```

$$\begin{bmatrix} \frac{-4}{3} & 2 \\ \frac{2}{3} & \frac{-2}{3} \end{bmatrix}$$

Mátrixok szorzása:

```
> evalm(A&*B);
```

$$\begin{bmatrix} -7 & 2 \\ 6 & 3 \end{bmatrix}$$

Transzponált:

```
> transpose(A);
```

transpose

$$\begin{bmatrix} 1 & 0 \\ -2 & 3 \end{bmatrix}$$

Rang:

```
> rank(A);
```

rank

$$2$$

Inverz:

```
> inverse(A);
```

inverse

$$\begin{bmatrix} 1 & \frac{2}{3} \\ 0 & \frac{1}{3} \end{bmatrix}$$

Determináns:

det

```
> det(A);
```

$$3$$

Oszlopvektor megadása:

```
> s:=vector(2,[1,2]);
```

vector

$$s := [1,\ 2]$$

Az A mátrixot kibővítjük az s oszloppal:

```
> augment(A,s);
```

augment

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 3 & 2 \end{bmatrix}$$

Tanulmányozzuk a Help-ben az augment parancsot!

6. Feladat. *Mennyi az alábbi polinomban x^3 együtthatója?*

$$\begin{vmatrix} 3x & 5 & 7 & 1 \\ 2x^2 & 5x & 6 & 2 \\ 1 & x & 0 & 3 \\ 2 & 1 & 4 & 7 \end{vmatrix}$$

7. Feladat. *Az $n \times n$-es D determináns i-edik sorának j-edik eleme 2^{ij}. 2-nek hanyadik hatványával osztható D. Oldjuk meg a feladatot ♣-lel $n = 8$-ra. (Meg tudnánk-e általánosan is oldani a feladatot?)*

8. Feladat. ★*Legyen A $n \times n$ típusú négyzetes mátrix. Az*

$$L_A\colon \mathbf{R}^n \to \mathbf{R}^n, \quad X \to A \cdot X \text{ (mátrix szorzás)}$$

leképezés lineáris transzformáció. Legyen

$$A = \begin{pmatrix} 2 & 1 & 2 \\ 3 & -4 & 2 \\ 1 & 5 & 2 \end{pmatrix}.$$

Határozzuk meg az

$$\frac{x-1}{2} = \frac{y+2}{4} = \frac{z}{3}$$

egyenletrendszerű egyenes; illetve az

$$x + 2y - 4z + 7 = 0$$

egyenletű sík képét az L_A transzformációnál.

9. Feladat. ★*Generáljunk „nem túlságosan triviális”, 1 rangú, V-vel jelölt 4×4 típusú mátrixot. Próbáljuk ki, hogy V^2 mindig a V skalárszorosa. (Be tudnánk látni általánosan is?)* `randmatrix`

10. Feladat. *Döngöljük földbe a ♣-t! Számítsuk ki a már korábban felírt α szöghöz tartozó forgatási mátrix nyolcadik hatványát ♣-lel! Elégedettek vagyunk az eredménnyel? Most csak a saját fejünket használva számítsuk ki a forgatási mátrix n-edik hatványát! És mi a helyzet a tükrözési mátrixszal?*

13. Szorgalmi feladat. *Könyörüljünk meg a ♣-ön! Írjunk olyan eljárást, mely a mátrixelemekben elvégzi az addíciós képletek alapján lehetséges trigonometrikus összevonásokat. Segítség a margón. Teszteljük eljárásunkat az előző feladaton, valamint két forgatási mátrix szorzatán is. (Persze tetszőleges n-re azért ne várjunk eredményt!)* `combine`

14. Szorgalmi feladat. *Vegyünk fel két lineárisan független vektort $\mathbf{R}^2$-ben. Képezzük 500 véletlenszerű lineáris kombinációjukat, ahol a skalárok a $[-1, 1]$ intervallumba esnek. (ld. a* `rand` *parancsot). Az így kapott listát ábrázoljuk! (*`pointplot` *parancs).*

7.3. A Gauss kiküszöbölés

A Gauss kiküszöbölés (Gauss elimináció) módszere a lineáris algebra legfontosabb gyakorlati technikái közé tartozik. A fő állítás az, hogy minden mátrix sorekvivalens egy lépcsős mátrixszal. A ♣ a *Gauss-Jordan alak*ú mátrixnak az olyan lépcsős mátrixot nevezi, amelyben minden sor vezető eleme 1, s a sorok vezető elemei fölött zéró elemek állanak.

Először az elemi sorátalakításokat „kézzel” hajtjuk végre, azaz ugyanúgy fogunk a ♣-lel dolgozni, mint a papírral ceruzával, csak az elemi sorátalakítások végrehajtását bízzuk a ♣-re. Az elemi sorátalakítások ♣ parancsai:

Az M mátrix r_1-edik sorának λ szorosát hozzáadjuk az r_2-edik sorhoz ($r1 \neq r2$): `addrow`

`addrow(M,r1,r2,lambda`.

Az M mátrix r_1-edik sorát szorozzuk a λ nemzéró skalárral: `mulrow`

`mulrow(M,r1,lambda)`.

Az r_1-edik és az r_2-edik sor cseréje: `swaprow`

`swaprow(M,r1,r2)`.

1. Példa: Elemi sorátalakításokkal hozzuk lépcsős alakra az $\begin{pmatrix} 1 & 1 & 2 & 1 \\ 3 & -4 & 1 & 1 \\ 4 & -3 & 3 & 2 \end{pmatrix}$ mátrixot!

Elindítjuk a `linalg` csomagot és bevisszük a mátrixot:

```
> with(linalg):
> M:=matrix(3,4,[1,1,2,1,3,-4,1,1,4,-3,3,2]);
```

$$M := \begin{bmatrix} 1 & 1 & 2 & 1 \\ 3 & -4 & 1 & 1 \\ 4 & -3 & 3 & 2 \end{bmatrix}$$

(A kiindulásnál sorcserére nincs szükség, mert az első sor vezető eleme az első elem.) Az első sor -3-szorosát hozzáadjuk a második sorhoz:

```
> M1:=addrow(M,1,2,-3);
```

$$M1 := \begin{bmatrix} 1 & 1 & 2 & 1 \\ 0 & -7 & -5 & -2 \\ 4 & -3 & 3 & 2 \end{bmatrix}$$

Az első sor -4-szeresét hozzáadjuk a harmadik sorhoz:

```
> M2:=addrow(M1,1,3,-4);
```

$$M2 := \begin{bmatrix} 1 & 1 & 2 & 1 \\ 0 & -7 & -5 & -2 \\ 0 & -7 & -5 & -2 \end{bmatrix}$$

A második sor -1-szeresét hozzáadjuk a harmadik sorhoz:

```
> M3:=addrow(M2,2,3,-1);
```

$$M3 := \begin{bmatrix} 1 & 1 & 2 & 1 \\ 0 & -7 & -5 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ez a kívánt lépcsős alak. Ezt az alakot egyetlen ♣ paranccsal is megkaphatjuk:

> `gausselim(M);`↩

A Gauss-Jordan alak eléréséhez még szükség van két lépésre. A második sort szorozzuk $-1/7$-del:

> `M4:=mulrow(M3,2,-1/7);`

$$M4 := \begin{bmatrix} 1 & 1 & 2 & 1 \\ 0 & 1 & \frac{5}{7} & \frac{2}{7} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Végül a második sor -1-szeresét hozzáadjuk az első sorhoz.

> `m5:=addrow(M4,2,1,-1);`

$$m5 := \begin{bmatrix} 1 & 0 & \frac{9}{7} & \frac{5}{7} \\ 0 & 1 & \frac{5}{7} & \frac{2}{7} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Az Gauss-Jordan alak szintén elérhető egyetlen paranccsal is:

> `gaussjord(M);`↩

A kapott alakból természetesen azonnal leolvashatjuk, hogy az M mátrix rangja 2.

1. Feladat. *Elemi sorátalakításokkal hozzuk lépcsős alakra az alábbi M mátrixot, majd keressük meg a Gauss-Jordan alakját is. Eredményünket ellenőrizzük direkt ♣ parancs segítségével is.* $\begin{pmatrix} 0 & 0 & 2 & 4 & 1 \\ 3 & 1 & 2 & 6 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & -1 & 1 \end{pmatrix}$

2. Feladat. *Határozzuk meg az alábbi A mátrix inverzét szimultán Gauss-eliminációval! Eredményünket ellenőrizzük direkt ♣ paranccsal és a szorzás elvégzésével is!* $A = \begin{pmatrix} 2 & 3 & 4 \\ 2 & 1 & 1 \\ -1 & 1 & 2 \end{pmatrix}$

Útmutatás: A mátrix bevitele után először generáljunk az E 3×3-as egységmátrixot, az `augment` paranccsal hozzuk létre az AE mátrixot, ezt elemi sorátalakításokkal hozzuk EB alakra – ha lehet, azaz, ha a mátrix rangja 3, ami az eljárás során kiderül. Ekkor $B = A^{-1}$.

7.4. Lineáris egyenletrendszerek megoldása

A lineáris egyenletrendszer bevitelére két, egymással egyenértékű módszer van: az egyenleteket bevihetjük soronként, illetve bevihetjük az alapmátrixot és a konstansokat külön.

2. Példa: A lineáris egyenletrendszer alapmátrixának, bővített alapmátrixának meghatározása. Tekinsük a következő lineáris egyenletrendszert:

$$\begin{aligned} x+\ \ y+2z+\ \ w &= 1\\ 3x-4y+\ \ z+\ \ w &= 2\\ 4x-3y+3z+2w &= 3 \end{aligned}$$

Először bevisszük az egyenleteket soronként.

```
> restart;↩
> with(linalg):↩
> eq1:=x+y+2*z+w=1:↩
> eq2:=3*x-4*y+z+w=2:↩
> eq3:=4*x-3*y+3*z+2*w=3:↩
```

Az egyenletek listája:

```
> eqs:=[eq1,eq2,eq3]:↩
```

Az ismeretlenek listája:

```
> vars:=[x,y,z,w]:
```

Az egyenletrendszer alapmátrixa: genmatrix

```
> M1:=genmatrix(eqs,vars);
```

$$M1 := \begin{bmatrix} 1 & 1 & 2 & 1\\ 3 & -4 & 1 & 1\\ 4 & -3 & 3 & 2 \end{bmatrix}$$

Az egyenletrendszer bővített alapmátrixa:

```
> M2:=genmatrix(eqs,vars,flag);
```

$$M2 := \begin{bmatrix} 1 & 1 & 2 & 1 & 1\\ 3 & -4 & 1 & 1 & 2\\ 4 & -3 & 3 & 2 & 3 \end{bmatrix}$$

A konstansok oszlopvektorát törölve ismét az alapmátrixot kapjuk:

```
> M2a:=delcols(M2,5..5);
```

$$M2a := \begin{bmatrix} 1 & 1 & 2 & 1\\ 3 & -4 & 1 & 1\\ 4 & -3 & 3 & 2 \end{bmatrix}$$

A konstansok vektora:

```
> M2b:=col(M2,5);
```

$$M2b := [1,\ 2,\ 3]$$

Az M_1 alapmátrixból képezett homogén lineáris egyenletrendszer:

```
> homsys:=geneqns(M1,[x1,x2,x3,x4]);↩
```

Az M_1 alapmátrixszal rendelkező inhomogén lineáris egyenletrendszer, ahol a jobb oldali konstansok az $M_2 b$ elemei:

```
> nonhomsys:=geneqns(M1,[x1,x2,x3,x4],M2b);↩
```

Megoldhatjuk az egyenletrendszert a már tanult `solve` parancs segítségével, ekkor az argumentumban maguk az egyenletek vannak:

```
> solve(homsys);
```

$$\{x2 = \frac{5}{9}\,x1 + \frac{1}{9}\,x4,\ x4 = x4,\ x1 = x1,\ x3 = -\frac{7}{9}\,x1 - \frac{5}{9}\,x4\}$$

```
> solve(nonhomsys);
```

$$\{x2 = \frac{5}{9}\,x1 - \frac{1}{3} + \frac{1}{9}\,x4,\ x4 = x4,\ x1 = x1,\ x3 = -\frac{7}{9}\,x1 + \frac{2}{3} - \frac{5}{9}\,x4\}$$

Megoldhatjuk az egyenletrendszert a `linsolve` parancs segítségével, ekkor az argumentumban az egyenletrendszer alapmátrixa, s a jobb oldali konstansok vektora szerepel. `linsolve`

```
> solution_of_homsys:=linsolve(M1,[0,0,0]);
```

$$solution\_of\_homsys := [\_t_1,\ \_t_2,\ 2\,\_t_1 - 5\,\_t_2,\ -5\,\_t_1 + 9\,\_t_2]$$

Azaz

$$\begin{aligned} x_1 &= t_1 \\ x_2 &= t_2 \\ x_3 &= 2t_1 - 5t_2 \\ x_4 &= -5t_1 + 9t_2 \end{aligned}$$

```
> solution_of_nonhomsys:=linsolve(M1,M2b);
```

$$solution\_of\_nonhomsys := [\_t_1,\ \_t_2,\ 2\,\_t_1 - 5\,\_t_2 - 1,\ -5\,\_t_1 + 9\,\_t_2 + 3]$$

(A megoldás kiolvasása remélem nem jelent gondot, az előzőektől x_3 és x_4 esetében egy-egy konstanssal tér el.) A megoldásban szereplő szabad paramétereket a ♣ $\_t_1$-el és $\_t_2$-vel jelöli, amire `_t[1]`-el és `_t[2]`-vel hivatkozhatunk. (Lásd később!)

Az inhomogén rendszer megoldáshalmaza lineáris sokaság. Keressük meg irányterét (illetve annak bázisát) és eltolóvektorát! (Másképpen fogalmazva: keressük meg a megoldás szerkezetét!) Az eltolóvektort úgy kapjuk meg, hogy a szabad paraméterek mindegyikének zéró értéket adunk.

```
> tvec:=subs(_t[1]=0,_t[2]=0, op(solution_of_nonhomsys));
```

$$tvec := [0,\ 0,\ -1,\ 3]$$

A továbbiakban bázist keresünk az irányterben. Ehhez a szabad paraméterek egyikének 1, a többinek pedig 0 értéket adunk, majd a kapott vektorból az eltolóvektort kivonjuk.

```
> bvec1:=evalm(subs(_t[1]=1,_t[2]=0,
> op(solution_of_nonhomsys))-tvec);
```

$$bvec1 := [1,\ 0,\ 2,\ -5]$$

```
> bvec2:=evalm(subs(_t[1]=0,_t[2]=1,
> op(solution_of_nonhomsys))-tvec);
```

$$bvec2 := [0,\ 1,\ -5,\ 9]$$

3. Feladat. *Tekintsük a következő lineáris egyenletrendszert:*

$$\begin{aligned} x - \ \ y + 2z - 2w &= 1 \\ 2x + \ \ y \ \ \ \ \ + 3w &= 4 \\ 2x + 3y + 2z \ \ \ \ \ &= 6 \end{aligned}$$

1. *Vigyük be az egyenleteket, írassuk ki az alapmátrixát (*`genmatrix`*, bővített alapmátrixát, a jobb oldali konstansokat.*
2. *Hasonlítsuk össze az alapmátrix és a kibővített alapmátrix rangját! Mit tudunk ez alapján állítani a megoldhatóságról és a megoldás szerkezetéről? Hány szabad paramétert várunk a megoldásban?*
3. *A* `linsolve` *parancs segítségével oldjuk meg az egyenletrendszert. Adjuk meg a megoldás szerkezetét!*

4. Feladat. *Az x, y, z, w ismeretlenekre felírt lineáris egyenletrendszer kibővített alapmátrixa:* $\begin{pmatrix} 0 & 0 & 2 & 4 & 1 \\ 3 & 1 & 2 & 6 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & -1 & 1 \end{pmatrix}$

1. *Vigyük be a mátrixot, írjuk fel az egyenletrendszert (*`geneqns`*)!*
2. *Vizsgáljuk a rendszer megoldhatóságát a Kronecker – Capelli tétellel! Hány szabad paramétert várunk?*
3. *Oldjuk meg az egyenletrendszert!*
4. *Mi lenne az ugyanilyen alapmátrixú, homogén lineáris egyenletrendszer megoldása? (Először a ♣ nélkül válaszoljunk a kérdésre.)*

5. Feladat. *Legyen:* $A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$ $B = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 5 & 7 \\ 2 & 3 & 4 & 5 \end{pmatrix}$ *Határozzuk meg az összes olyan X, Y 4×4-típusú mátrixot, amelyre $AX = 0$, illetve $YB = 0$ teljesül. A szorzás elvégzésével ellenőrizzük, hogy az általunk megadott mátrixokra az állítás teljesül.*

7.5. Grafikus megoldás

Az alábbiakban a két illetve három változós lineáris egyenletrendszerek megoldásának vizualizációjára látunk néhány példát.

```
> restart;
```

```
> with(linalg):with(plots):with(plottools):↩
```

3. Példa: Oldjuk meg grafikusan az `eq1`, `eq2` egyenletekből álló lineáris egyenletrendszert:

```
> eq1:=3*x-4*y=3:
> eq2:=x+y=1:
```

A megoldás:

```
> solve({eq1,eq2});
```

$$\{y = 0,\ x = 1\}$$

A grafikus megoldás (szemléltetés) lényege, hogy az egyenletekből y-t kifejezzük:

```
> plot({solve(eq1,y),solve(eq2,y)},x=-5..5,y=-5..5);
```

6. Feladat. *Bővítsük az egyenletrendszert a két egyenlet lineáris kombinációival (legalább még kettővel), s szemléltessük a rendszert grafikusan!* ★*V.ö. sugársor egyenlete!*

4. Példa: Három változóra az eljárás hasonlóan egyszerű:

```
> eq3 := 2*x + 3*y + z = 2:eq4 := 2*x + 4*y + 7*z =6:
> eq5 := 3*x + 10*y + 5*z = 7:
> solve({eq3,eq4,eq5});
```

$$\{z = \frac{36}{59},\ y = \frac{20}{59},\ x = \frac{11}{59}\}$$

```
> plot3d({solve(eq3,z), solve(eq4,z),solve(eq5,z)},
> x=-5..5, y=-5..5, view=-5..5, axes=boxed);↩
```

A színezést kicsit megváltoztatjuk:

```
> graf1:=plot3d(solve(eq3,z),
> x=-5..5, y=-5..5, view=-5..5, axes=boxed, color=red):
> graf2:=plot3d(solve(eq4,z),
> x=-5..5, y=-5..5, view=-5..5, axes=boxed, color=green):
> graf3:=plot3d(solve(eq5,z),
> x=-5..5, y=-5..5, view=-5..5, axes=boxed, color=yellow):
> display(graf1,graf2,graf3);↩
```

7. Feladat. *Bővítsük a rendszert az egyenletek egy lineáris kombinációjával, végezzük el a szemléltetést (valamely új színt használva.)*

8. Feladat. *Oldjuk meg és szemléltessük az alábbi lineáris egynletrendszereket:*

```
> eq1 := x - 3*z = -3;
> eq2 := 2*x + 2*y - z = -2;
> eq3 := x + 2*y + 2*z = 1;
```

```
> eq1 := x - 3*z = -3;
> eq2 := 2*x - 5*y - z = -2;
> eq3 := x + 2*y - 5*z = 1;
```

8. Sajátérték, sajátvektor

```
> restart:with(plots):with(linalg):
```

1. Példa: Definiáljuk az A mátrixot, s az azonos típusú egységmátrixot.

```
> A := matrix(3,3,[[-1,0,5],[0,4,0],[5,0,-1]]);
> I3 := evalm(array(identity,1..3,1..3));
```

Definiáljuk a karakterisztikus polinomot:

```
> chpol := det(A - x*I3);
```

Megoldjuk a karakterisztikus egyenletet:

```
> AEvals := solve(chpol = 0,x);
> AEvals[1];AEvals[2];AEvals[3];
```

Megkeressük az első sajátértékhez tartozó invariáns alteret (sajátalteret).

```
> AEspace1 := linsolve(A-AEvals[1]*I3,[0,0,0]);
```

Bázis az invariáns altérben:

```
> AEbasis1 := [subs(_t[1]=1, op(AEspace1))];
```

A második sajátértékhez tartozó invariáns altér:

```
> AEspace2 := linsolve(A-AEvals[2]*I3,[0,0,0]);
```

Bázis az invariáns altérben (most 2 szabad paraméter van, a geometriai dimenzió 2):

```
> AEbasis2 := [subs(_t[1]=1,_t[2]=0, op(AEspace2)),
> subs(_t[1]=0,_t[2]=1, op(AEspace2))];
```

Ellenőrizzük, hogy az előbbi bázisvektorok valóban sajátvektorok:

```
> evalm(A &* AEbasis1[1]);
> evalm(A &* AEbasis2[1]);
> evalm(A &* AEbasis2[2]);
```

Megjegyezzük, hogy az A mátrix diagonalizálható (a valós számok teste fölött), mert 3 valós sajátértéke van, s mindegyik sajátérték geometriai és algebrai multiplicitása megegyezik.

Keressük meg azt az S mátrixot, melyre $S^{(-1)}\,A\,S$ diagonális:

```
> S := augment(AEbasis1[1],AEbasis2[1],AEbasis2[2]);
```

Valóban:

```
> B:=evalm(inverse(S)&*A&*S);
```

1. Feladat. *Mit tudunk az alábbi mátrixok diagonalizálhatóságáról állítani?*

```
> C := matrix(3,3,[[0,1,1],[0,2,0],[-2,1,3]]);
> E := matrix(3,3,[[4,-3,1],[4,-1,0],[1,7,-4]]);
```

2. Példa: A Maple saját beépített paranccsal rendelkezik a sajátértékek és sajátvektorok kiszámítására:

```
> eigenvectors(A);
```

Az előbbi eredmény a következőt jelenti: az A első sajátértéke -6, egyszeres algebrai multiplicitással, a hozzá tartozó sajátaltérben bázis a {[-1,0,1]} vektorrendszer. Az A második sajátértéke 4, kétszeres algebrai multiplicitással, s a kiírt két vektor alkot bázist a hozzá tartozó sajátaltérben.

Helyezzük el az előbbi hármasokat egyetlen sorozatban:

```
> evA := [eigenvectors(A)];
```

A sajátértékek:

```
> evA[1][1]; evA[2][1];
```

A sajátvektorok:

```
> evA[1][3][1]; evA[1][3][2]; evA[2][3][1];
```

Ha itt hibát talált, akkor ez azért van mert a Maple más sorrendben listázta a sajátértékeket, mint amikor a munkalapot írtam. Ez esetben szerkessze át a parancssort.

2. Feladat. *Számítsuk ki az $A^{25}X$ vektort, ahol $X = [2, 3, -1]^t$. (Az 1. példa jelöléseivel.)*

Megoldás: $A^{25} = S\,B^{25}\,S^{(-1)}$ (miért?), ahol B^{25} könnyen számítható, mert B diagonális mátrix.

```
> X := vector(3,[2,3,-1]);
> evalm(S&*B^(25)&*inverse(S)&*X);
```

A Maple persze direkt nódon is kiszámítja az eredményt:.

```
> evalm(A^25 &* X);
```

3. Példa: Generáljunk 3×3 típusú diagonalizálható mátrixot!

```
> restart:with(linalg):
> randomize():
```

Először generálunk egy véletlen 3×3 típusú diagonális mátrixot:

```
> DIAG:=
> matrix(3,3,[[rand(-3..3)(),0,0],[0,rand(-3..3)(),0],
> [0,0,rand(-3..3)()]]);
```

Most generálunk egy 3×3 típusú nem elfajuló mátrixot:

```
> S1 := randvector(3,entries=rand(-3..3));
> S2 := randvector(3,entries=rand(-3..3));
> S3 := randvector(3,entries=rand(-3..3));
> S := augment(S1,S2,S3);
> rank(S);
```

Ha S rangja nem 3, akkor azt újra kell generálni.

```
> A := evalm(S&* DIAG &* inverse(S));
```

Ellenőrzés:

```
> eigenvectors(A);
```

3. Feladat. *Generáljunk olyan egészekből álló diagonalizálható mátrixot, melynek sajátértékei általunk választott számok!*

4. Feladat. *Írjunk eljárást, mely előállít egy egészekből álló $n \times n$ típusú nem elfajuló véletlen mátrixot!*