MatLab összefoglaló

Alapfogalmak:

Konstansok: i, j, pi, e, inf, -inf, NaN, eps

Aritmetika: sin, cos, tan, atan, sind, cosd, tand, atand, sqrt, exp, pow, pow2, log, log10, exp, factorial, factor, primes, round, floor, ceil, abs, ~

Vektorok: *transzponált, összefűzés, indexelés*, length, size, ones, zeros, rand, diag, eye, linspace, (logspace), find, *beépített fv.-ek vektorokon (sin, abs, exp, round, stb.), logikai műveletek és indexelés (pl. a<1.2, b~=3)*, sum, prod, mean, min, max, *elemenkénti műveletvégzés (a.*b, a.^2, stb.)*

2D ábrázolás: figure, plot(x), plot(x,y), stem(x), stem(x,y), stairs, hold on/off, subplot, grid on/off, legend, *linespec*, xlim, ylim, box, text, xlabel, ylabel, title, ('FontSize', 14, 'FontWeight', 'bold'), close all, set(0,'DefaultAxesFontSize',20), clf

Kiíratás, szövegformázás: disp, sprintf, fprintf, *formatSpec*, rats, format, num2str, mat2str

Mátrixok, alapműveletek: *mátrix létrehozása, transzponált, összefűzés, indexelés(skalárral, egész vagy logikai vektorral, lineárisan)*, length, size, ones, zeros, rand, diag, eye, squeeze, reshape, numel**,** find, *beépített fv.-ek mátrixokon (sin, abs, exp, round, stb.)*, *logikai műveletek és indexelés (pl. A<1.2, B~=3)*, sum, prod, mean, min, max, trace, rank, det, inv, *elemenkénti műveletvégzés (A.*B, A.^2, stb.)*

Egyenletrendszerek: *megoldása osztással*

Polinomok: roots, polyval, poly, polyfit

Deriválás, integrálás: diff, *differenciahányados,* trapz, cumsum, cumtrapz, integral

Képek betöltése, kirajzolása: colormap, imagesc

Egyéb: pause, error, clc, clear all, load, save, isequal

3D ábrázolás: meshgrid, surf, mesh, surfc, meshc, quiver, quiver3, contour, contour3, contourf, clabel, colorbar, view

Cellatömbök: cell, *elemeinek indexelése*, *létrehozása ismert elemekkel*, cellplot

Struktúrák: *használatuk*

Fájlkezelés: filesep, fopen, fclose, fprintf, fscanf, textscan, fwrite, fread

Szótár:

Konstansok:

- i és j : képzetes egység (mindkettő ugyanaz)
- pi, e, inf, -inf
- NaN : nem elvégezhető műveletek eredménye (pl: nullával osztás, végtelenek összeadása…)
- eps : számábrázolás hibája, konkrétan az 1 és a legnagyobb 0 és 1 közt lebegőpontosan számolható szám különbsége (=2^{-52}). Jelentősége: két elvileg egyenlő szám ennyivel eltérhet egymástől.

Aritmetika:

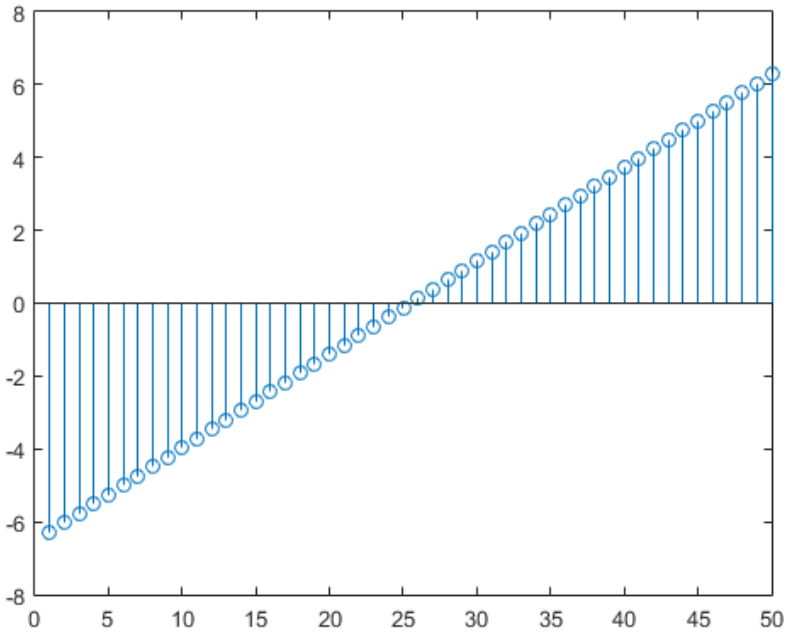
- radiános trigonometrikus függvények: sin, cos, tan, atan,
- szöges trigonometrikus függvények: sind, cosd, tand, atand,
- sqrt (gyökvonás), exp (e^x), pow (x^y), pow2 (2^y), log (ln(x)), log10 (tízes alapú), factorial (x!), factor (prímtényezőket adja vissza vektorban), primes (x-nél kisebb prímszámok), round (kerekít), floor (mindig felfele kerekít), ceil (mindig lefele kerekít = egészrész), abs (abszolút érték)
- ~ -- ez a negálás MatLabban (használható így is: ~=, teljes zárójeles tagot is negálhatunk stb.)

Vektorok:

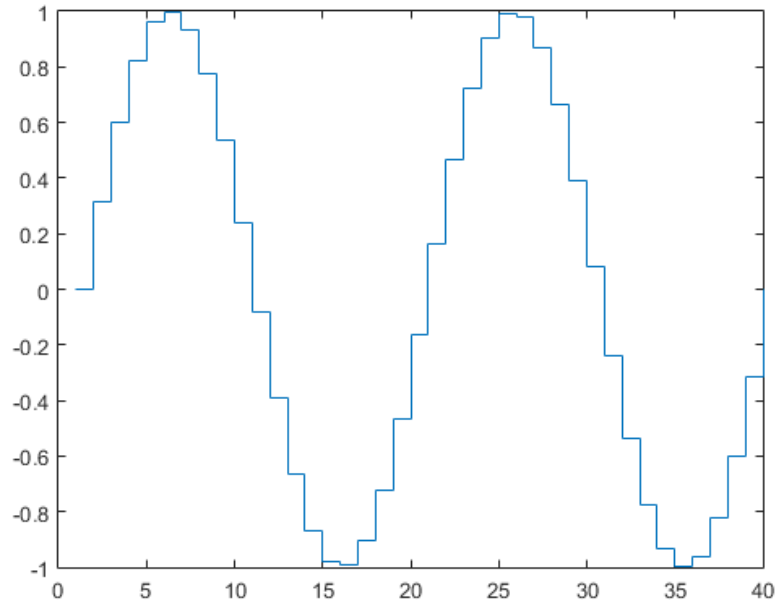
- *transzponált : A'*
- *összefűzés: X = [A, B; C, D] – vesszővel elválasztva egy sorba kerülnek, pontosvesszővel új sorba*
- *indexelés :*
  - *sima: v(5:8, :) megj: a kettőspont azt jelenti, hogy az adott dimenzió mentén az összes elem*
  - *logikai: x =(v < 5) >> olyan vektort ad vissza, ahol azokon a helyeken, ahol 5-nél nagyobb szám áll, egyes lesz, a többi helyen 0. v(x) kifejezés pedig visszaadja azokat az értékeket, amik nagyobbak, mint 5, a többit pedig kihagyja. Hasznos még: v(x) = 0 – kinullázza az 5-nél kisebb elemeket.*
- length, size (vektor hossza -- mindkettő)
- ones (n) – n hosszú vektort készít, amiben csupa egyesek vannak
- zeros(n) – ugyanaz, mint a ones, csak nullákkal
- rand(n) – n hosszú vektor 0-1 közti véletlen számokkal feltöltve (intervallum: [0;1) !!)
- diag(v) – a paraméterként átadott vektorból diagonális mátrixot készít (vektor elemei a főátlóba kerülnek)
- eye(n) – nxn-es egységmátrixot készít
- linspace(a,b,c) – a-tól b-ig terjedő intervallumot (vektort) ad vissza, amit c db elemet tartalmaz. Az elemek közt a köz egyenletes. (logspace – logaritmikus felosztású intervallum)
- find(X < 10, k) – X elemei közül az első k szám, ami megfelel a feltételnek
- *beépített fv.-ek vektorokon (sin, abs, exp, round, stb.) – értelemszerűen (pl: sin(X))*
- sum(v) – vektor elemeinek összege, prod(v) – vektor elemeinek szorzata, mean(v) – v átlaga, min(v) – legkisebb elem, max(v) – legnagyobb elem
- *elemenkénti műveletvégzés (a.*b, a.^2, stb.)*

2D ábrázolás:

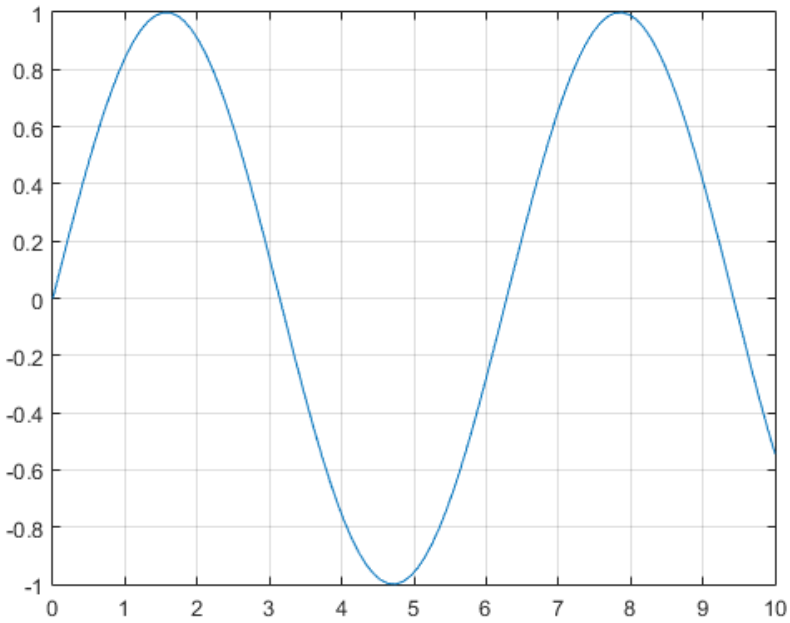
- figure(k) – új grafikonablak nyitása (paraméter opcionális)
- plot(y) – kirajzolja a y sorozat elemeit, az x tengelyen egész számok vannak
- plot(x,y, 'Linespec') – y függvény kirajzolása x tartomány fölött
- stem(y), stem(x,y) olyan grafikont rajzolt, ahol y értékekek pontokként ábrázolja, és a pontokat összeköti az x tengellyel



- stairs(x), stairs(x,y):



- hold on/off – „hold on" esetén ugyanarra a grafikonra rajzolja az összes függvényt, „hold off"-nál mindig felülírja az előzőt
- subplot(n,m,p) – a grafikonablakot nxm-es táblázatra osztja és a p-dik cellába rajzolja a következő grafikont.
- grid on/off – rács bekapcsolása



- legend(’szöveg1’, ’szöveg2’,…) – jelmagyarázat
- *linespec: színek(b – kék r – piros, g – zöld, k – fekete, m – magenta(rózsaszín), y – sárga, c – világoskék, w – fehér), vonaltípus(- folytonos vonal, : pontozott, -- szaggatott vonal, -. szaggatott-pont), jelölő(o, s, +, *, ., x, ^, v, >, <), ’LineStye’, ’LineWidth’, ’MarkerStyle’, ’MarkerSize’, ’MarkerFaceColor’, ’MarkerEdgeColor’*
- xlim([a,b]), ylim([a,b]) – tengelyek tartománya
- box on/off – grafikon kerete kapcsolható ki és be

- text(pi,0,' szöveg') – a grafikonra ír szöveget x, y pozicióban (x,y a grafikon szerinti értékek)
- xlabel(’Szöveg’), ylabel(’Szöveg’) – tengelyfeliratok beállítása
- title(’Szöveg’) – grafikon címe *('FontSize', 14, 'FontWeight', 'bold')*
- set(0,'DefaultAxesFontSize',20) – összes grafikon tengelyfeliratainak betűméretének beállítása
- clf – aktuális grafikonablak ürítése, törlése

Kiíratás, szövegformázás:

- disp(v) – v változó kiírása a konzolra
- str = sprintf('%0.5f',A) – A változóban tárolt elemet szöveggé alakítja '%0.5e' formázó kifejezés szerint
- fprintf('X is %+4.2f meters or %8.3f mm\n', A1,A2) – kiírja az első paraméterként átadott szöveget úgy, hogy %4.2f helyébe A1 értékét helyettesíti be, %8.3f helyébe pedig A2 értékét.
- *formatSpec – egy szöveg, amiben kijelöljük, hogy kiíratáskor hova fognak a változók kerülni. Az előző példában az % jel adja meg, a behelyettesítendő számok helyét, az utána álló szám megadja, hogy a szövegben hány karakternyi helyet foglaljon el a beillesztett szám, a pont utáni szám a tizedes jegyek számát adja meg, f pedig azt jelző, hogy fixpontos alakban szeretnénk kiírni a számot. Lehet még: d – egész, e – exponenciális alak, g – kompakt alak: kiválasztja, hogy lebegőpontosan, vagy fixpontosan jobb) A + jel azt jelzi, hogy az előjelet akkor is tegye ki, ha pozitív a szám.*
- rats(X) – X mátrix elemeit törtalakra hozza

- format long – a konzolra írandó számok alapértelmezetten ilyen formátumban kerülnek kiírásra lehetséges értékei: long – fixpontos sok tizedesjeggyel, shortEng – mérnöki alak (lebegőpontos alak, úgy, hogy a 10 kitevője mindig 3 többszöröse), compact – kevés space-t rak a számok közé. Megj: paraméter nélkül visszaállítja az alapértelmezést.
- num2str, mat2str – számok és mátrixok szöveggé alakítása

Mátrixok, alapműveletek:

- *létrehozás: pl: A = [2, 3, 4; 5, 6, 7; 8, 9,10] megj: a vessző elhagyható*
- *transzponált, összefűzés, indexelés – ugyanúgy, mint a vektoroknál*
- length(A) – a legnagyobb kiterjedésű dimenzió menti mérete
- size(A,k) – k-dik dimenzió menti mérete
- ndim(B) – B dimenzióinak száma
- ones (n,m) – csupa 1-esből álló nxm-es mátrix
- zeros(n) – ugyanaz, mint a ones, csak nullákkal
- rand(n) – n hosszú vektor 0-1 közti véletlen számokkal feltöltve (intervallum: [0;1) !!)
- diag(A, k) – a paraméterként átadott mátrix diagonális elemeiből vekort készít (vektor elemei a főátlóbeli elemek lesznek)
- eye(n) – nxn-es egységmátrixot készít
- find(X < 10, k) – ugyanaz, mint a vektoroknál
- *beépített fv.-ek vektorokon (sin, abs, exp, round, stb.) – értelemszerűen (pl: sin(X))*
- sum(B,k) – mátrix elemeinek összege k dimenzió mentén (pl: sum(B, 2) soronként összeadja a számokat és egy oszlopvektort ad vissza), prod(B,k) – vektor elemeinek szorzata k dimenzió mentén, mean(B,k) – v átlaga … , min(B, [],k) – legkisebb elem a k-dik dimenzió mentén, max(B, [],k) – legnagyobb elem k-dik dimenzió mentén (megj: alapértelmezetten oszloponként végzi a műveleteket)
- *elemenkénti műveletvégzés (A.*B, A.^2, stb.)*
- reshape(X, n, m) – X mátrixot átalakítja nxm-es mátrixszá.

Egyenletrendszerek:

- A mátrixok körében a szorzás nem kommutatív, ezért kétféle osztás definiálható:
  - bal osztás: A\B azt az X mátrixot jelenti, amelyre A*X = B azaz
    - A\B = A-1B
  - jobb osztás: A/B azt az Y mátrixot jelenti, amelyre Y*B = A azaz
    - A/B = AB-1
- MATLAB-ban van beépített inverz számoló: inv(...), de ha kifejezetten lineáris egyenletrendszert akarunk megoldani, akkor sokkal célszerűbb a 'bal osztás' operátorával dolgoznunk:
  - A*x = b >> ~~x = inv(A)*b~~ hanem: x = A\b

Polinomok:

- gyökök kiszámítása: roots(P);
- kiértékelés adott pontban: y0 = polyval(P,x0);
- kiértékelés sok pontban: x = -1:0.01:1; y = polyval(P,x);

- polinom létrehozása a gyökeiből: r = [-1 1]; P = poly(r);
- polinom illesztés: P = polyfit(x_vektor,y_vektor,fokszam);

Deriválás, integrálás:

- diff(t) – t vektorban található szomszédos elemek különbségét eltárolja egy vektorban. Megj: a kapott vektor elemszáma eggyel kisebb, mint a bemeneti vektor elemszáma
- *differenciahányados:* $f'(x_i) = \frac{f(x_i) - f(x_{i+1})}{x_i - x_{i+1}} = \frac{diff(értékek)}{diff(tartomány)}$
- trapz(t, y) – t tartomány fölött kiszámolja az y vektorban tárolt értékek integrálját trapéz-szabály szerint
- cumsum(y)*res – y függvényértékek integrálját számolja ki, ahol res a mintavételezések köze,
- cumsum(y)*res – ugyanaz, mint a kumulált összeg, csak trapézszabály szerint
- cumtrapz(t, y) – y értékek integrálját kiszámolja t tartomány fölött kumulált összeg trapézszabály alapján
- integral(y, a, b) megadja y függvényérték-vektor elemeinek integrálját a és b indexű elemek közt.

Képek betöltése, kirajzolása:

- colormap – betöltött grafikonok színsémáját állítja be. Lehetséges értékei:

parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
gray	
bone	
copper	
pink	
lines	

colorcube	
prism	
flag	
white	

- imagesc – átskálázza a változóként tárolt képet vagy grafikont. Mi csak a képek/grafikonok megjelenítésére használjuk.

Egyéb:

- pause(t) – megállítja a program futását t másodpercig. Paraméter nélkül billentyűlenyomásig vár.
- error('Szöveg') – hibaüzenetet dob, megmondja hol és hanyadik sorban történt a hiba és a script futása megáll.
- clc – Command Window törlése
- clear all – változók törlése
- load 'fájlnév', save 'fájlnév' – változók betöltése illetve elmentése fáljba
- isequal(X, Y) –igazzal tér vissza, ha X és Y tökéletesen megegyeznek

3D ábrázolás:

- meshgrid(a:b, c:d) – létrehoz térhálót
- surf(x, y, z) – folyamatos felületű 3D-s grafikon
- mesh(x, y, z) – térhálós felületű 3D grafikon
- surfc(x, y, z) – folyamatos felületű 3D grafikon + alatta látszanak a szintvonalak
- meshc(x, y, z) – térhálós felületű 3D grafikon + alatta látszanak a szintvonalak
- contour(x,y, z, k) – szintvonalak 2D-ben, a negyedik opcionális paraméter (k) a szintvonalak sűrűségét adja meg.
- contour3(x,y, z,k) – szintvonalak 3D-ben, a negyedik opcionális…
- contourf(x,y, z,k) – szintvonalak 2D-ben, a szintvonalak által közrezárt területek színezettek. A negyedik…
- quiver(x,y, u, v), quiver3(x,y,z, u,v,w) – 2 és 3D-s vektorok ábrázolása, ahol (x,y) illetve (x,y,z) a vektorok kezdőpontjai, (u, v) és (u,v,w) pedig az irányvektoraik. Általában (x,y) mátrixot meshgriddel állítjuk elő: [x,y] = meshgrid(5:0.5:10, 5:0.5:15).
- clabel – szintvonalak feliratait lehet bekapcsolni bemeneti paramétereként a szintvonalak mátrixát kéri ezért így lehet használni:

  ```
  [C,h] = contour(x,y,z);
  clabel(C,h)
  ```

- colorbar – bekapcsolja az oldalsó színskálát (paraméterként be lehet állítani a színskála helyét: 'north', 'south', 'east', 'west', 'northoutside', 'southoutside', 'eastoutside', 'westoutside')
- view(Az, El) – 3D-s grafikonok nézőpontját lehet beállítani (Az = azimuth, El = elevation)

Cellatömbök: Olyan adattípus, melyet különböző típusú és/vagy méretű változók tárolására használhatunk. Fontos, hogy az egyes cellákban teljesen eltérő elemeket tárolhatunk, akár további cellatömböket is.

- cell(n,m) – nxm-es üres cellatömb létrehozása. Ha csak egy paramétert kap, akkor négyzetes nxn-es tömböt csinál. Akár többdimenziós cellatömb is létrehozható így.
- *elemeinek indexelése: Cellatömb elemeinek indexelése a { } zárójelekkel, rész-cellatömb indexelése a ( ) zárójelekkel történik. Magyarul, ha { } jelekkel hivatkozunk elemre, akkor az adott cellában tárolt változónak megfelelő típusú elemet kapunk, ha viszont ( ) zárójeleket használunk, akkor egy 1x1-es cellatömböt kapunk.*
- *létrehozása ismert elemekkel: pl: C ={ 1, 2, 3;*
  'text', rand(5,10,2), {11; 22; 33}}
- cellplot(c) – grafikonablakban megjeleníti a c cella tartalmát legalább a tárolt változók típusát feltűntetve

Struktúrák: A cellatömbökhöz hasonlóan különböző típusú és/vagy méretű változók tárolására használható adattípus azzal a különbséggel, hogy a tárolt adatok névvel ellátott mezőkbe kerülnek (hasonlóan pl a C nyelv struct típusához).

- pl: hallgato.nev = 'valami szöveg';
  hallgato.szuletesi_datum = '1995.01.01.'
  hallgato.tanulmanyi_atlag = 5.00

Fájlkezelés:

- filesep – a futtató operációs rendszer szerinti fáljútvonal-elválasztót adja vissza. Ez Windows alatt '\', Linux alatt viszont '/'
- fID = fopen('file_path', 'r') –elérési útvonal alapján megnyit egy fájlt olvasásra (ha 'w' lenne a második paraméter, akkor írni lehetne bele). Lehet relatív útvonalat is megadni. Pl a './szoveg.txt' fáljt az alapértelmezett mappájában fogja keresni (ahova a .m fájlokat akarja mindig menteni). Visszatérési értéke a fájl azonosítója.
- fclose(fID) – fájl bezárása
- fprintf(fID, formatSpec, adatok) – ugyanúgy működik, mint a szövegkiíratásnál , csak meg kell adni első paraméterként a fájl azonosítóját
- A = fscanf(fID, formatSpec) – a formatSpec szerint olvas be értékekek az A-ba adatfájlból.
- A = textscan(fID, formatSpec) – a formatSpec szerint olvas be értékekek az A-ba formázott szöveget tartalmazó fájlból. Az fscanf-hez képest annyi a különbség, hogy ez alkalmas formázott szöveget beolvasni és a szövegből szétválogatni az adatokat a formatSpec szerint.
  - Pl ezt be tudja olvasni egy fájlból

```
09/12/2005 Level1 12.34 45 1.23e10 inf Nan Yes 5.1+3i

10/12/2005 Level2 23.54 60 9e19 -inf  0.001 No 2.2-.5i

11/12/2005 Level3 34.90 12 2e5   10  100   No 3.1+.1i
```

```
fileID = fopen('scan1.dat');
C = textscan(fileID,'%s %s %f32 %d8 %u %f %f %s %f');
fclose(fileID);
```

- data = fread(fID, beolvasando_adatok_szama, adatok_tipusa) – bináris fájlból olvas be (aminek általában .bin a kiterjesztése, és nem lehet szövegszerkesztővel megnyitni). A beolvasando_adato_szama lehet inf is, akkor az összes adatot beolvassa.
- fwrite(fID, kiirando_ertek, ertek_tipusa) – binárisan fájlba ír. Második paraméterként stringet vár (pl: 'double').

Vezérlő szerkezetek:

`if` *feltétel1*
  parancsok1
`elseif` *feltétel2*
  parancsok2
`else`
  parancsok3
`end`
	• Nincs zárójelezés, az adott blokk végét az end jelzi.
	• A feltétel logikai értékét vizsgálja(ami nem 0, az igaz).
	• Relációs operátorok (==, ~=, <, >, <=, >=)
	• Logikai operátorok (&, &&, |, ||, ~, xor, all, any)

`for` n = 1:10 *parancsok* `end`	• Ismert számú iteráció elvégzésére. • Nincs zárójelezés, az adott blokk végét az end jelzi. • A ciklusváltozót sorvektorként definiáljuk, a ciklus törzsében értéke az aktuális elemnek megfelelő skalár. • A ciklusváltozó tetszőleges sorvektor lehet (pl.[1 3 7 5 6]).
`while` *feltétel* *parancsok* `end`	• Ismert számú iteráció elvégzésére. • Nincs zárójelezés, az adott blokk végét az end jelzi. • Nincs explicit módon megadott ciklusváltozó. • Figyeljünk a végtelen ciklus elkerülésére!

`in = input('Írjbe egyszámot: ');`
 	
`switch` in
  `case` 0
  `disp('Nullát írtál be.');`
  `case` 1
  `megoldas = 2*pi*exp(3.2);`
  `disp(megoldas);`
  `case` {2, 5}
  `disp('A bemenet 2 vagy 5.');`
  `otherwise`
  `disp('Nem jó értéket adtál meg, próbáld újra...');`
`end`	

`function` kimenet = peldafv(bemenet1, bemenet2)
 	
`%% Peldafuggveny help bejegyzese.`
 	
`% A ket bemeno parameter`
`% tetszoleges szam lehet, a`
`% kimenet ezek osszege.`
`kulonbseg = bemenet1 - bemenet2;`
`kimenet = bemenet1 + bemenet2;`
	• .m kiterjesztés,
	  ○ ne kezdődjön számmal a fájlnév,
	  ○ ne legyen benne space,
	  ○ ne egyezzen meg beépített függvény nevével (pl. plot.m)
	• function kulcsszóval kezdődik, end zárja le
	• bemenő paraméterek és lehetséges kimenetek
	• saját, lokális scope (a nem visszatérési értékként
	• megadott változók csak a függvény futása alatt

end	léteznek)

Esettanulmányok:

Virtuális műtét:

```
% adatok betöltése
load anat;
% az összes szelet megjelenítése


figure(2)
colormap gray;

for ind = 1:size(anat,3)
  szelet = anat(:,:,ind);
  imagesc(szelet);
  axis off;
  pause(0.02)
end
```

Ház forgatása billentyűnyomásra

```
load house.mat % ház képét egy H nevű változóba betölti

figure(3);
for ang = 0:15:90
    % aktualis forgatasi szog
    phi = ang;

    % aktualis forgatasi matrix
    R = [cosd(phi) -sind(phi);
        sind(phi) cosd(phi)];

    % forgatas elvegzese
    HR = house*R;

    % kirajzolas
    plot(HR(:,1),HR(:,2), 'LineWidth', 3);
    xlim([-10 10]);
    ylim([-10 10]);
    title(strcat('Forgatas ', num2str(ang, 4), ' fokkal'), ...
        'FontSize', 12, 'FontWeight', 'bold');

    % a futas megallitasa tetszoleges billentyu lenyomasaig:
    pause;
end
```

Illesztés, deriválás:

```
%% Betoltes, Illesztes
load polinom
meresiPozicio=meresiPozicio;
mertErtekek=mertErtekek;
% Illesztett polinom egyutthatok
polinomEgyutthatokHatodfokra = polyfit(meresiPozicio,mertErtekek,6);

% Polinom ertekei a [-4;8] intervallumon es kirajzolas
```

```
szazPont=linspace(-4,8,100);
szazPontosErtekekHatodfokra =
polyval(polinomEgyutthatokHatodfokra,szazPont);

figure(1)
plot(szazPont,szazPontosErtekekHatodfokra,'r.-')


%% Numerikus derivalt
% Kolunbsegek kiszamitasa
% Ertelmezesi tartomany
dSzazPont = diff(szazPont);
% Ertekkeszlet
dSzazPontosErtekekHatodfokra = diff(szazPontosErtekekHatodfokra);

% Differencia hanyados
differenciaHanyadosSzazPontos = dSzazPontosErtekekHatodfokra./dSzazPont;

%% Derivalas egy adott pontban
% Derivalt szamitasa a t0=6 helyen
t0 = 6;
% Fuggveny ertek
y0 = polyval(polinomEgyutthatokHatodfokra,t0);
% a derivalas "felbontasa"
dt0 = 1E-6;

% t0 koruli kornyezet
tTartomany = [t0-dt0 t0+dt0];
% a fv. ertekei ugyanitt
yTartomany = polyval(polinomEgyutthatokHatodfokra,tTartomany);

%kulonbsegek
dtTartomany = diff(tTartomany);
dyTartomany = diff(yTartomany);
% derivalt
differenciaHanyados = dyTartomany/dtTartomany;

% kirajzolas
figure(2)
plot(t0,y0,'ko')
plot([t0-1 t0+1],[y0-differenciaHanyados y0+differenciaHanyados],'g')
legend('10 pontos',...
    '100 pontos',...
    'Derivalasi hely', ...
    'Erinto egyenes',...
    'Location','SouthWest')
```

Integrálás:

```
%% Osszeadassal
integralOsszeadassal = sum(szazPontosErtekekHatodfokra)*(szazPont(2)-
szazPont(1));
% Kirajzolas
figure(4);
hold on;
bar(tizPont,tizPontosErtekekHatodfokra,1,'w','EdgeColor','b','LineStyle'
,'-');
plot(szazPont,szazPontosErtekekHatodfokra,'r-');
```

```
%% Trapezzal
integralTrapezzal = trapz(tizPont,tizPontosErtekekHatodfokra);

% Kirajzolas
figure(5);
plot(szazPont,szazPontosErtekekHatodfokra,'r-');
stem(tizPont,tizPontosErtekekHatodfokra,'b-');

%% Anonim fuggveny segitsegevel
% Anonim fuggveny letrehozasa
hatodfokuPolinomFuggveny = @(x) polyval(polinomEgyutthatokHatodfokra,x);

% Integralas
integralFuggvennyel =
integral(hatodfokuPolinomFuggveny,gyokok(6),gyokok(1));

%% Kiiratas
fprintf(['Numerikus integralas eredmenye:\n\tOsszeadassal: %3.3f\n\t'...
'Trapez szaballyal: %3.3f\n\tFuggvennyel: %3.3f\nAnalitikusan:
%3.3f\n'],...
integralOsszeadassal,integralTrapezzal,integralFuggvennyel,analitikusan)
;
```

Bölcsesség:

- Integrálásnál a sima összeadást ne használjuk
- Vektorokhoz a trapéz szabály
- Függvényekhez (szögfüggvények, polinomok) pedig az integrálfüggvényt

```
%% +1: kumulált integrálösszeg -- integrálfüggvény
% vizsgált intervallum
res = 0.001;
x = 0:res:pi/2;
% alap kumulált összeg (fontos beszorozni a lépésközzel,
% mivel a cumsum csak a fv. értékeit adja össze)
z3 = cumsum(sin(x))*res;
% az eredményvektor utolsó eleme megadja az adott intervallumra
% vonatkozó integrál (a görbe alatti terület) értékét
z3(end)
% kumulált összeg trapézszabály alkalmazásával (ezt ajánlott használni)
% 1-es lépésközt feltételez
z4 = cumtrapz(sin(x))*res;
z4(end)
% így az x határozza meg a lépésközt, nem kell szorozni
z5 = cumtrapz(x,sin(x));
z5(end)
```

Differenciál egyenletrendszerek:

1. Külön megírt függvénnyel:

```
function lab07_pelda3_matlab2015
% elsorendu, ketvaltozos differencialegynelet megoldasa,
% folyadekok keveresekor a koncentracioszintek viselkedese

    format long;

    % az ODE megadasa kulon fv.-ben (chem.m) tortent
    [t y] = ode45('chem',[0 0.5], [0 1]);
```

```
        % VAGY:
        % [t y] = ode45(@chem,[0 0.5], [0 1]);

        % VAGY: (ha ugyanebben a fáljban van a függvény)
        % [t y] = ode45(@(t,y) chem(t,y),[0 0.5], [0 1]);

    end
```

---------- külön fálj ----------

```
        function dydt = chem(t, y)
        % folyadekok keveresekor a koncentracioszintek viselkedese.
        % maga a diffegyenlet:

            % y - allapotvaltozo
            dydt = zeros(2,1);

            % dA/dt
            dydt(1) = -10*y(1) + 50*y(2);
            % dB/dt
            dydt(2) = 10*y(1) - 50*y(2);
        end
```

---

2. Anonim függvénnyel (+ fázisgörbe):

```
function lab07_pelda5_matlab2015
% elsorendu, ketvaltozos differencialegynelet megoldasa,
% Lotka-Volterra ragadozo-zsakmany modell
% 2015. 04. 18.

    format long;

    % kornyezeti kapacitasok (eltartokepesseg)
    mu1 = 200; % zsakmany
    mu2 = 300; % ragadozo
    % a rendszert leiro diffegyenlet-rendszer
    PredPrey = @(t,y) [(1-y(2)/mu2)*y(1);
                       -(1-y(1)/mu1)*y(2)];
    % kezdeti ertekek
    y0 = [100;  % zsakmany
          150]; % ragadozo

    % megoldas
    [t_pp y_pp] = ode45(PredPrey, [0 20], y0);

    % kirajzolas
    figure;
    subplot(2, 1, 1);
    hold on;
    plot(t_pp,y_pp(:, 1), 'g', 'LineWidth', 2);
    plot(t_pp,y_pp(:, 2), 'b', 'LineWidth', 2);
    title('Predator-Prey Modell', 'FontSize', 14);
    xlabel('t', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('egyedszam', 'FontSize', 12, 'FontWeight', 'bold');
    legend('zsakmany','ragadozo');

    % fazisgorbe
    subplot(2, 1, 2);
```

```
        plot(y_pp(:,1), y_pp(:,2));
        title('Fazisgorbe', 'FontSize', 14);
        xlabel('zsakmanyok szama', 'FontSize', 12, 'FontWeight', 'bold');
        ylabel('ragadozok szama', 'FontSize', 12, 'FontWeight', 'bold');

    end
```

Fájlkezelés:

1. Kiírás (fprintf)

```
%% formázott szöveg kiírás

% kiirando jel generalasa
% mintaveteli frekvencia
Fs=1000; % Hz
% idovektor
t=0:1/Fs:10; % s
% körfrekvencia
f=5; % Hz
% szinusz jelalak
s=sin(2*pi*f*t);

% az adott helyhez kepest vett relativ cimzes
% (lehet abszolut is, teljes utvonallal)
fs = filesep;
DIR_PATH=['.' fs];

% a kiirando fajl neve
filename='szinusz_fprintf.txt';

% a fajl eleresi utvonala (ha csak a fajlnevet
% adom meg, az aktualis konyvtarba menti)
file_path=[DIR_PATH filename];

% fajl megnyitasa irasra, ez utan az fid-val
% hivatkozok erre a fajlra
fid=fopen(file_path,'w');

% header irasa, hogy tudjuk, mi van a fajlban
% pl.: Szinusz (Fs = 1000)
fprintf(fid,['Szinusz (Fs = ' num2str(Fs) ') \n']);
fprintf(fid,'t sin \n');

% az iras utan nyitva marad a fajl, ezert irhatok meg bele formazott
adatot

% irjuk bele a kiszamolt szinusz fuggvenyt
fprintf(fid,'%5.2f %4.4f\n',[t;s]);

% fajl bezarasa - FONTOS, ne felejtsuk el!
fclose(fid);
```

2. Beolvasás (fscanf)

```
%% Fajlbol olvasas formázott szövegként 1 (fscanf)
% az fscanf csak az alap formazo
% karaktereket ismeri (lasd HELP)
```

```
% a beolvasando fajl neve (amit
% az elobb kiirtunk)
filename='szinusz_fprintf.txt';

% a fajl eleresi utvonala
file_path=[DIR_PATH filename];

% fajl megnyitasa olvasasra
fid=fopen(file_path);

% fajl beolvasasa formazott
% szovegkent
% eloszor a header beolvasasa
% stringkent
Sheader=fscanf(fid,'%s',6); % 6 darab sztringet olvas be
% majd az adatok beolvasasa
% fixpontos szamkent
Sdata=fscanf(fid,'%f %f',[2 inf]); % [m,n]: Read at most m*n elements
in column order. n can be inf, but m cannot.

% fajl bezarasa
fclose(fid);

% beolvasott adatok kirajzolasa
figure;
plot(Sdata');
legend('t','sin');
title(Sheader);%% Fajlbol olvasas formázott szövegként 1 (fscanf)
```

3. Beolvasás (textscan)

```
%% Fajlbol olvasas formázott szövegként 2 (textscan)
% a textscan-nek lehet regularis
% kifejezest is adni (lasd HELP)
%
% EESS
%
% a kimenetet cell-array-be teszi

% a beolvasando fajl neve (meg
% mindig ugyanaz)
filename='szinusz_fprintf.txt';

% a fajl eleresi utvonala
file_path=[DIR_PATH filename];

% fajl megnyitasa olvasasra
fid=fopen(file_path);

% fajl beolvasasa cellatömbbe
% header beolvasasa
Sheader_ts=textscan(fid,'%s',6);
% adatok beolvasasa
Sdata_ts=textscan(fid,'%f %f');

% fajl bezarasa
fclose(fid);

% beolvasott adatok kirajzolasa
tt = Sdata_ts{1};
```

```
data = Sdata_ts{2};

% cim konvertalasa:
abracim = cell2mat(Sheader_ts{1}(:)'); % egy elemu cell-array,
melyben
% stringek vannak egy oszlopban... ezt meg karakter-matrix formatumra
% alakitjuk a cell2mat-tal

figure;
plot(tt,data);
title(abracim);
```

4. Beolvasás (fread)

```
format long;

%{
% adatfajl eloallitasa:
P = [180, 190, 130, 170, 0, 6];
filename='lab10_pelda2_adat.bin';
fs = filesep;
dir_path = ['.', fs];
file_path=[dir_path, filename];
fid=fopen(file_path,'w');
fwrite(fid, P, 'double');
fclose(fid);
return;
%}

filename='lab10_pelda2_adat.bin';
fs = filesep;
dir_path = ['.', fs];
file_path=[dir_path, filename];
fid=fopen(file_path);
d = fread(fid, 6, 'double');
fclose(fid);


p = d(1);
q = d(2);

modell = @(t,y) [(1-y(2)/q)*y(1);
                 -(1-y(1)/p)*y(2)];

y0 = [d(3);
      d(4)];

tspan = [d(5) d(6)];

[T Y] = ode45(modell, tspan, y0);
```