Matlab/Octave a geoinformatikában

Dr Laky Piroska
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
ÉPÍTŐMÉRNÖKI KAR

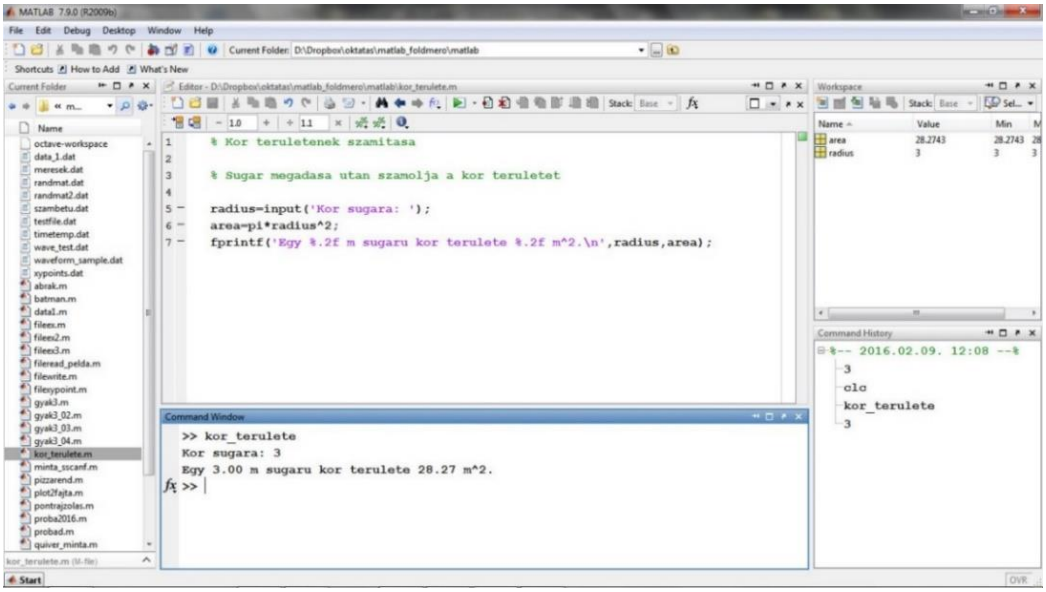
2018

TARTALOM

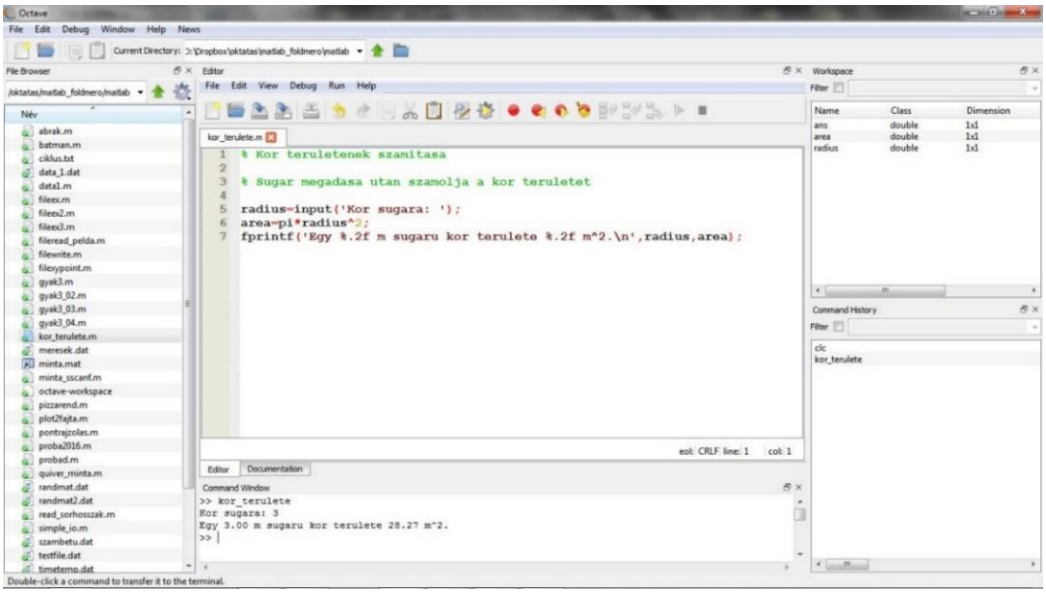
1. MATLAB/OCTAVE/ ALAPOZÓ[1]

A gyakorlatok során Octave/Matlab matematikai környezet használatát fogjuk elsajátítani geodéziai, térinformatikai példákon keresztül. Mind a MATLAB, mind az Octave alapvetően numerikus számítások végzésére kidolgozott program környezet. Az utóbbi egy ingyenes, nyílt forráskódú program, ahol lényegében ugyanazokat a parancsokat használhatjuk, mint a Matlab-ban, mivel az Octave készítői törekednek a Matlab kompatibilitásra. Még a grafikus felület is nagyon hasonló a két programban, tetszés szerint átrendezhető ablakokkal (lásd 1., 2. ábra).

1MATLAB GRAFIKUS KÖRNYEZETE



2OCTAVE GARFIKUS KÖRNYEZETE



[1] A segédlet készítése során felhasználva: Todd Young and Martin J. Mohlenkamp: Introduction to Numerical Methods and Matlab Programming for Engineers, Department of Mathematics, Ohio University, July 24, 2018, http://www.ohiouniversityfaculty.com/youngt/IntNumMeth/book.pdf

Az Octave a https://www.gnu.org/software/octave/ oldalról tölthető le, jelenleg a 4.4.1 változat, ami 2018. augusztus 9-én jött ki. Sok kiegészítő csomag is van hozzá, lásd https://octave.sourceforge.io/ illetve https://octave.sourceforge.io/packages.php , ezek egy jó része telepítve is van, csak be kell tölteni használatkor. Le lehet kérdezni, hogy mi van telepítve a **pkg list** paranccsal).

Otthoni gyakorláshoz használható, 2017 márciusától ingyenesen, a Matlab szoftver is egyetemi licenccel. A Matlab oldalán létre lehet hozni egy MathWorks account-ot (célszerű BME-s email címet használni, ha az Online Matlab-hoz is szeretnénk hozzáférni). A MathWorks account-tal lehetőség van megoldani a **Matlab Onramp**-ben található alap gyakorló feladatokat, ezt mindenkinek ajánlott végigcsinálni (kb. 2-3 óra). További jó, online gyakorlási lehetőségek vannak a **Matlab Cody** oldalán.

A Matlab telepítő csak az egyetemi gépekről érhető el: http://net.bme.hu/sw/. Leírás a telepítéshez: https://viki.eik.bme.hu/doku.php?id=mathworks:mathworks. Van telepítés nélkül is használható on-line Matlab is BME-s email címmel regisztrált felhasználóknak. BME-s email címet a címtár oldalán lehet létrehozni: https://login.bme.hu/admin Neptun-kóddal és jelszóval kell belépni. Itt ki kell választani a 'Felhasználónév beállítása a Microsoft Office 365 csomaghoz', majd meg kell adni a kívánt felhasználónevet. Egy kis idő múlva elkészül a felhasznalonev@edu.bme.hu email cím. Erről értesítést küld a rendszer a neptunban megadott email címre. Belépni a https://portal.office.com oldalon lehet. Ezzel az email címmel lehet utána regisztrálni a Matlab oldalra.

MATLAB MUNKAKÖRNYEZET, ALAPOK

A grafikus felület fontosabb részei: az éppen aktuális könyvtár, ahová mindent ment a program (**current folder**), a parancssor (**command window**), a munkakörnyezet (**workspace**) az éppen használt változókkal, az eddig futtatott parancsok (**command history**) és a szerkesztő (**editor**). Az adott panel nevére kattintva, lenyomva tartott bal egér gombbal áthúzhatóak a panelek, és a mozgatás során a kékre színezett területre helyezhetőek. Célszerű lehet rögzíteni a szerkesztőt (dock editor), ezt az Editor ablak jobb felső részén lévő nyílra kattintva tehetjük meg.

A későbbiekben amire mindig figyeljünk, hogy a Matlabban használt fájl nevek nem kezdődhetnek számmal és ne legyenek benne ékezetes karakterek, szóközök se! A program mindig azokat a fájlokat, függvényeket tudja éppen használni, amik a beállított aktuális könyvtárban vannak.

Octave-ban, ha azt szeretnénk, hogy bármilyen művelet eredménye folyamatosam jelenjen meg, és ne oldalanként, akkor használjuk a `page_screen_output(0)` parancsot!

SEGÍTSÉG (HELP, DOCUMENTATION)

Nagyon sokat tanulhatunk a dokumentáció használatából is (lásd a documentation fül, vagy az alábbi weblap), persze kellő angoltudás függvényében. Nyugodtan lehet nem

csak az Octave saját dokumentációját nézni, hanem a Matlabét is az interneten, ott több, sokszor részletesebb példát is láthatunk az adott parancs használatára.

Octave help-je: https://www.gnu.org/software/octave/doc/interpreter/

Matlab help-je: http://www.mathworks.com/help/matlab/

Sok hasznos segítség, letölthető matlab fájl található a matlab centralon is: http://www.mathworks.com/matlabcentral/

Az Octave-on és a Matlabon belül alapvető a help használata. Matlabon belül ha beírjuk a parancssorba a **help** parancsot önmagában, akkor megkapjuk a témakörök listáját (ez octave-ban nincs). Matlab-ban be lehet írni a **help** után egy adott témakör nevét is pl (> jel jelzi a Matlab-ba/Octave-ba begépelendő parancsokat, ezt a jelet nem kell begépelni!)

```
> help elfun
```

Ez az alap függvények (elementary math function) listáját adja. Octave-ban ez elérhető a **Documentation** fülön az 'Arithmetic' témakörre kattintva.

Ha tudjuk egy adott függvény nevét, akkor mind a Matlab-ban, mind az Octave-ban használhatjuk következőt:

```
> help 'parancsnév'
```

Ez megadja az adott parancs leírását, használatának módját. Pl.:

```
> help rand
```

Megadja, hogy a **rand** parancs 0-1 közötti egyenletes eloszlású véletlen számot generál, megadja a használatának módját, hogy lehet egy vagy több bemenettel is hívni és megadja a kapcsolódó parancsokat is (pl. **randn**, ami standard normális eloszlású véletlen számokat generál 0 várható értékkel és 1 szórással).

A **doc** parancs a help-hez hasonlóan működik. Octave-ban megnyitja az adott parancs leírását a **Documentation** fülön (ez egyezik a help parancs tartalmával, csak a **help** eredménye a **Command window**-ban jelenik meg). Matlab esetében a **doc** parancs egy külön ablakban nyitja meg az eredményt, ami a help-hez képest egy jóval részletesebb leírás, sok példával. De ugyanez elérhető a fent említett Matlab help internetes oldalon is. Próbáljuk ki a következő parancsot is:

```
> doc rand
```

Próbáljuk ki a **pwd** parancsot! Kérdezzük le help-pel ez mit is csinál!

Másik hasznos parancs a **lookfor** utasítás. Ezzel parancs részletekre is rákereshetünk, vagy bármilyen szóra, ami a parancs leírásban szerepel. Próbáljuk ki a következőt:

```
> lookfor rand
```

Ez minden parancsot kilistáz, aminek a nevében, vagy a rövid leírásában szerepel a rand szó. Ha túl sokáig tartana a keresés, akkor megszakíthatjuk a parancsot a **CTRL + C** billentyű kombinációval.

NÉHÁNY HASZNOS PARANCS

`clc` – kitörli a command window ablak tartalmát

`clear` – kitörli a változókat (lásd workspace)

`close` – bezárja az aktuális ábrát, vagy az összeset (close all)

`CTRL+C` – félbeszakítja az adott parancsot (kilépés pl. végtelen ciklusból)

`%` – megjegyzés (a program figyelmen kívül hagyja ami ez után van a sorban)

`;` – parancs végén a ; hatására nem jelenik meg az eredmény

`page_screen_output(0)` -csak Octave-ban! A műveletek eredményét folyamatosan jelenítse meg a Command Windowban, ne oldalanként tördelve.

'TAB' GOMB ÉS A NYILAK HASZNÁLATA A PARANCSSORBAN

Nagyon hasznos a 'tab' használata. Ha nem tudjuk pontosan egy adott parancs nevét, csak az elejét, és elkezdjük begépelni a parancssorba pl, hogy pref, majd utána nyomunk egy **tab**-t ha csak egyféle pref kezdetű parancs van, akkor kiegészíti, ha több, akkor megadja a lehetséges parancsokat. Itt több parancs is van ezzel a kezdettel pl. **prefdir** (annak a könyvtárnak a neve, ahol a beállítások, history stb. található) vagy a **preferences**, ami megnyitja a beállítások ablakot.

Szintén nagyon hasznos a nyilak használata a parancssorban, amivel korábbi parancsokat lehet újra előhozni, lefuttatni, módosítani. A korábbi parancsokat újra le lehet futtatni a **command history**-t használva is, dupla kattintással az adott parancson.

ÉRTÉKADÁS, VÁLTOZÓ TÍPUSOK, FÜGGVÉNY HASZNÁLAT

```
> %% Értékadás, változótípusok
> % Egyszerű értékadás (0.01 háromféleképpen)
> a = 0.01
> b = 1e-2
> c = 1d-2;
> a+c
> clear a % kitörli az 'a' változót
> clear % vagy 'clear all' kitörli az összes változó értekét
> pi % beépített érték (3.14)
> e = exp(1) % e^1 = e = 2.71
> b = e^-10 % hatványozás
```

Néhány megjelenítési, formázási lehetőség:

```
> format long % több tizedes jegy megjelenítése
> e, b
> format short % rövidebb megjelenítés
> e, b
```

A Matlab alap objektumai a mátrixok. A vektorok speciális mátrixok, pl. 1xn-es sorvektorok, vagy mx1-es oszlopvektorok. Mátrix/vektor definiálásához szögletes zárójelet használunk. Az elemeket egy soron belül vesszővel, vagy szóközzel választjuk el, sorok között az elválasztó a pontosvessző.

```
> z = [1 3 45 33 78] % sorvektor
> z = [1,3,45,33,78] % sorvektor másképp megadva
> t = [2; 4; 22; 66; 21] % oszlopvektor
> M = [1,2,3; 4,5,6] % 2x3-as méretű mátrix/tömb
```

Le lehet kérdezni egy vektor tetszőleges elemét, kerek zárójelbe téve az elem sorszámát:

```
> t(2) % eredménye: 4
> M(2,3) % eredménye: 6
> z(end) % z utolsó eleme: 78
```

Vagy felül lehet írni bármelyik elem értékét:

```
> t(2)=47
> p = [] % üres vektor
> z(3)=[]; % kitörli a 3. elemet, utána z = 1 3 33 78
```

Le lehet kérdezni egy részét a vektornak, mátrixnak:

```
> t(2:4) % eredménye az előző parancs után: 47 22 66
> t(1:29 % elgépelés esetén hibaüzenet
t(39
    ?
 Error: Expression or statement is incorrect--possibly unbalanced (, {, or
[.
> t(1:29)
Index exceeds matrix dimensions.

> M(1:2,2:3) % eredménye: [2,3; 5,6]
```

Vektor, mátrix transzponáltja (felcseréli a sorokat, oszlopokat):

```
> tt = t' % t transzponáltja, sorvektor
> Mt = M' % eredménye: [1,4; 2,5; 3,6]
```

Vannak hasznos parancsok, amelyekkel egyszerűen lehet vektorokat előállítani:

```
> x1 = 1:10 % sorvektor 1-10-ig egész számok
> x2 = 1:0.3:10 % sorvektor 1-tol 10-ig, 0.3 osztásközzel
> x3= (1:0.3:10)' % oszlopvektor 1-tol 10-ig, 0.3 osztásközzel
> x4 = linspace(1,10,4) % 1 és 10 között 4 pontot vesz fel
```

Könnyű összefűzni vízszintesen, függőlegesen azonos sor/oszlop számú vektort/mátrixot.

```
> X = rand(2,3) % 2x3-as [0,1] közti véletlen számokból álló mátrix
> Y = ones(2,4) % 2x4-es egyesekből álló mátrix
> Z = eye(3) % 3x3-as egységmátrix
> W = zeros(2,4) % 2x4-es nullákból álló mátrix
> XY = [X, Y] % 2x7 elemű mátrix, vízszintesen összefűzve X és Y
> XZ = [X; Z] % 5x3-as mátrix, függőlegesen összefűzve X, Z
> XY2 = [X; Y] % hibaüzenet
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
> XZ2 = [X,Z] % hibaüzenet
Error using horzcat
Dimensions of matrices being concatenated are not consistent.
```

Sorok oszlopok leválogatása

```
> XY(1,:) % XY első sora (: az adott sorban az összes elem)
> XY(end,:) % XY utolsó sora (: az adott sorban az összes elem)
> XY(:,1) % XY első oszlopa (: az adott oszlopban az összes elem)
```

```
> XY(:,end-1) % XZ utolsó előtti oszlopa (: az adott oszlop összes
  eleme)
```

Szövegek, mint karakterekből álló vektorok

```
> s = 'p' % szöveges/karakter (string) típusú változó 1x1 méretű
> me = 'Műszaki Egyetem' % string típusu változo 1x15 méretű
> bme = ['Budapesti ',me] % Budapest Műszaki Egyetem
> bme(13:17)
```

SCRIPT ÍRÁSA

Eddig parancssorból dolgoztunk, de egy bonyolultabb számítást, programot már nehéz parancssorban megírni, szükség esetén javítani. Célszerűbb lehet egy fájlba összegyűjteni a használt parancsokat, ez a script fájl. Itt is használhatunk minden korábbi Matlab függvényt, de egyszerűbb a javítás, futtatás. A Matlab alapértelmezett fájl típusa a *.m fájl, ez egy egyszerű szövegfájl, amit bármilyen szövegszerkesztővel szerkeszthetünk. Ezen kívül az újabb Matlab verziókban már használhatjuk a livescript fájltípust is (*.mlx), ez utóbbiban vegyesen használhatunk Matlab utasításokat és formázott szövegeket, képeket, illetve láthatjuk rögtön az eredményeket is. Ez viszont a Matlab saját formátuma, amit csak Matlab programon belül tudunk megnyitni.

A script fájlokat futtathatjuk a nevük begépelésével, de egyszerűbb az **F5** megnyomásával, ez rögtön menti és futtatja a programot. Figyeljünk, hogy a fájlnév ne kezdődjön számmal és ne legyenek benne szóközök, ékezetek! A fájlnevek csak betűvel kezdődhetnek, és csak az angol abc betűi, illetve számok és alulvonás szerepelhet bennük.

Amennyiben nem akarjuk az egész programot lefuttatni betehetünk egy **return** parancsot valahová, és akkor csak addig fog lefutni a program. Illetve az **F9** lenyomásával csak a kijelölt rész fog lefutni. Másik megoldás, ha szekciókra osztjuk a fájlt dupla % jelek használatával (%%) és utána egy szekció cím megadásával. Egy bizonyos szekcióban lévő parancsokat a **CTRL+Enter** billentyűk megnyomásával tudunk futtatni. Kezdjünk egy új script fájlt a bal felső sarokban a plusz jelre kattintva (new), és ezzel megnyílik az Editorban egy üres lap. Mentsük el az aktuális könyvtárunkba gyak1.m fájl néven! A továbbiakban ebbe fogunk dolgozni.

EGYSZERŰ PLOTTOLÁS

Nézzük az alábbi táblázat adatait egy betonacél feszültség-fajlagos alakváltozás (σ-ε) diagramjából:

ε [%]	0	0.2	2	20	25
σ [N/mm^2=Mpa]	0	300	285	450	350

1. TÁBLÁZAT BETONACÉL FESZÜLTSÉG-FAJLAGOS ALAKVÁLTOZÁS DIAGRAMJA

Írjuk be a gyak1.m script fájlba:

```
> %% Betonacel alakváltozása
> x = [0,0.2,2,20,25] % kiírja a képernyőre a tartalmát
> y = [0,300,285,450,350]; % nem írja ki a képernyőre a tartalmát
```

Futtassuk vagy az **F5** paranccsal az egész fájlt, vagy egy kijelölt részt az **F9**-cel, vagy **CTRL+Enter**-rel a szakaszt!

Ha beírjuk a script fájlba, parancssorba egy létező változó nevét, akkor kiírja az aktuális tartalmát, még akkor is, ha az előző parancsnál az y után ; szerepelt, ezért ott ugyan nem írta ki a képernyőre a változó értékét, de a workspace-be elmentette!

```
> x, y
```

Vektor formátumban lévő adatokat a plot paranccsal rajzolhatunk ki:

```
> plot(x,y)
```

Ez egy vonallal össze fogja kötni a pontokat. Ha a pontokat szimbólumokkal szeretnénk jelölni, próbáljuk ki a következőket:

```
> plot(x,y,'x')
> plot(x,y,'o-')
> plot(x,y,'r*-')
```

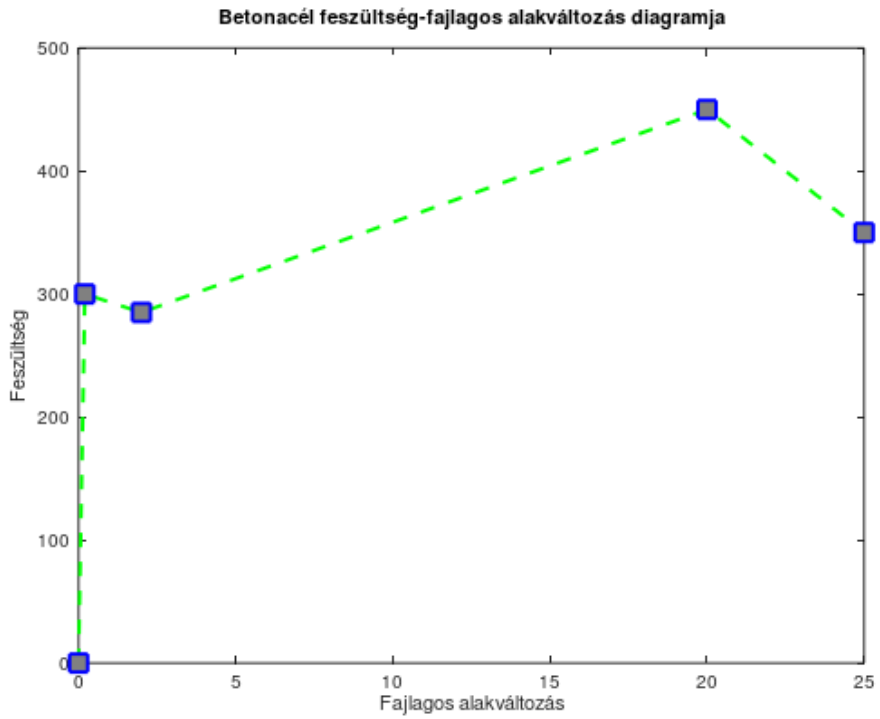
További formázási lehetőségek:

```
> plot(x,y,'gs--')
> plot(x,y,'gs--','LineWidth',2,'MarkerSize',10,...
>     'MarkerEdgeColor','b','MarkerFaceColor',[0.5,0.5,0.5])
```

A '--' szaggatott vonalat jelent, 's'-green (zöld), 's'-square (négyzet), 'LineWidth'- vonalvastagság, 'MarkerSize'- pont szimbólum mérete, 'MarkerEdgeColor'- pont szimbólum határvonala, 'MarkerFaceColor'- pont szimbólum kitöltésének színe. Ez csak néhány példa a beállítási lehetőségekre.

Elnevezhetjük a tengelyeket, vagy jelmagyarázatot, címet is fűzhetünk hozzá

```
> xlabel('Fajlagos alakváltozás')
> ylabel('Feszültség')
> title('Betonacél feszültség-fajlagos alakváltozás diagramja')
```



Vagy bezárhatjuk az ábrát:

```
> close % ábra bezárása
```

Hasznos paraméterek:

Line Style	Description
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Marker	Description
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram

Long Name	Short Name	RGB Triplet
'yellow'	'y'	[1 1 0]
'magenta'	'm'	[1 0 1]
'cyan'	'c'	[0 1 1]
'red'	'r'	[1 0 0]
'green'	'g'	[0 1 0]
'blue'	'b'	[0 0 1]
'white'	'w'	[1 1 1]
'black'	'k'	[0 0 0]

LineWidth	vonalvastagság
MarkerEdgeColor	pont szimbólumok határoló vonalának a színe
MarkerFaceColor	pont szimbólumok kitöltése
MarkerSize	pont szimbólumok mérete

FÜGGVÉNYEK

Nagyon sok matematikai és egyéb beépített függvény van, ezek megismeréséhez célszerű a **help**-et, dokumentációt böngészni. Nézzük meg néhány függvényt az alap matematikai függvények közül (Elementary math functions) – ld. **help elfun**

A függvények változói mindig kerek zárójelben vannak. A trigonometrikus függvényeknél az alapértelmezett szögegység a radián!

```
> sin(pi) % értéke 0 a számábrázolás pontosságán belül
> cos(pi) % -1
> tan(pi) % végtelen helyett egy nagy szám
> log(100) % természetes alapú logaritmus
> log10(100) % 10-es alapú logaritmus
```

```
> 3^4 % értéke: 81
> sqrt(81) % 9
> abs(-6) % 6
> exp(0)
```

Az **exp(0)** az e^0, értéke természetesen 1.

A beépített függvények nem csak számokon, hanem vektorokon is működnek.

```
> x = linspace(0,2*pi,40)
> y = sin(x)
> plot(x,y)
```

További beállítások lásd **help plot**!

FELHASZNÁLÓ ÁLTAL DEFINIÁLT EGYSOROS FÜGGVÉNYEK

Többféleképp lehet Matlab-ban saját függvényeket megadni, az egyszerűbb esetekben leginkább az egysoros függvényeket használjuk (ezt az angol nyelven 'anonymous function'-nek nevezik, ami nincs elmentve külön programként, csak egy változóhoz van hozzárendelve). Néhány példa:

```
> f = @(x) cos(2*x)
> ma = @(a,b) a*b+12
> f(pi) % 1
> ma(5,7) % 47
```

Itt először egy @ szimbólum után meg kell adni a függvény független változóit, majd utána jön a formula. Rajzoljuk fel az előbb felrajzolt szinusz függvény mellé a cos(2*x) függvényt is, most nem előre megadott pontpárokat használva, hanem szimbolikusan az **ezplot** vagy **fplot** paranccsal! (Megj.: Octave-ban és régebbi Matlab verzióba az ezplot parancs használható, újabb Matlab-ban az fplot).

```
> hold on
> ezplot(f,[0,2*pi])
```

A **hold on** parancs nélkül, ha egy új rajzolási parancsot adok ki, akkor letörli az előző ábrát és úgy rajzolja fel az újat, **hold on** esetén megtartja a régit, és mellé rajzolja az újat. A **hold off** paranccsal visszaállítható az alapértelmezett mód. Az **fplot** esetén meg lehet adni az intervallumot, ha nem az alapértelmezett tartományon szeretnénk a függvényt megjeleníteni.

Az ábra törlését, grafikus ablak bezárását a következő parancsokkal tehetjük meg:

```
> clf % Clear current figure
> close % Close figure
```

Állítsuk elő saját függvényt használva 1-10-ig a számok négyzetét!

```
> f1 = @(x) x^2
```

Ellenőrizzük le egy adott x értékére:

```
> f1(3) % értéke: 9
```

Állítsuk elő az f függvényt használva 1-10-ig a számok négyzetét!

```
> x = 1:10;
> y = f1(x)
```

Erre hibaüzenetet kapunk:

```
Error using  ^
One argument must be a square matrix and the other must be a scalar. Use
POWER (.^) for elementwise power.
Error in gyak1>@(x)x^2
Error in gyak1 (line 100)
y = f(x)
```

Miért? Azért mert az x változónk egy sorvektor, négyzetre emeléskor pedig két sorvektort próbálunk összeszorozni, ami matematikailag helytelen. Ha nem vektor/mátrix műveletet szeretnénk végrehajtani, hanem azt szeretnénk, hogy elemenként hajtódjon végre a művelet, akkor egy pontot kell tenni a műveleti jel elé.

```
> f1 = @(x) x.^2
> y = f1(x) % 1  4   9  16  25  36  49  64  81  100
```

Mivel vektorok/mátrixok esetén az összeadás, kivonás, skalárral való osztás/szorzás eleve elemenként történik, ezért ezekben az esetekben nem kell pontot tenni a műveleti jel elé, csak a szorzás, osztás és hatványozás esetén: .*,./,.^.

A Matlabnak az egyik erőssége a vektorokkal, mátrixokkal való műveletek végzése, így sok esetben elkerülhetjük a lassabb ciklus műveletek alkalmazását.

FÜGGVÉNYEK KÜLÖN FÁJLBAN

A Matlab alapértelmezett fájltípusa a *.m kiterjesztésű szöveges fájl. Ennek két fő típusa van, script fájl és függvény (function). Az előbbit már használtuk az eddigi munkánk során, nézzük meg miben különbözik ettől a függvények írása!

A függvények külön fájlba történő megírásának egyik előnye, hogy nem csak abból a script fájlból használhatjuk, ahol épp dolgozunk, hanem bármely fájlból meghívhatóak, így ha van olyan feladat, ami gyakran ismétlődik, akkor azt célszerű egy külön függvényben megírni. Az egysoros függvényekkel szemben sokkal bonyolultabb számítások, műveletek elvégzésére használhatjuk őket, könnyebben paraméterezhetjük, hogy hány ki és bemenete legyen, magyarázatokat fűzhetünk hozzá.

Írjuk meg külön függvény fájlba az előbb megadott négyzet függvényt! Kattintsunk a bal felső sarokban a plusz jelre (new), és megnyílik az Editorban egy üres lap. írjuk be a következőket, majd mentsük el negyzetfv.m néven az aktuális könyvtárba. Fontos: a függvény neve (ami vastaggal van írva a következő kódban) meg kell egyezzen a fájl nevével, különben nem lehet meghívni!

```
> function y = negyzetfv(x)
> % Kiszámolja a bemenet négyzetét
>     y = x.^2;
> end
```

A függvények néhány fontos tulajdonsága:

- function szóval kezdődik
- Van legalább egy-egy kimenet és bemenet
- A kimenet, a függvény neve és a bemenet az első sorban található, és a függvény neve meg kell egyezzen az *.m fájl nevével
- A függvény belsejében valahol értéket kell adjunk a kimenetnek

- A függvény belső változói lokális változók, nem fognak megjelenni a workspace-ben, és a függvény sem fér hozzá a workspace-ben lévő változókhoz, csak ahhoz, amit megadtunk a bemenetnél.
- A függvényt nem lehet futtatni, csak egy másik fájlból, vagy parancssorból meghívni! A meghíváshoz a függvénynek az aktuális könyvtárban kell lennie (vagy egy olyan könyvtárban, ami benne van az elérési útban (path).
- A függvény első sora után megadott kommentek tartalmát listázza ki a help parancs az adott függvényre!

Hívjuk meg a megírt függvényt az x vektorunkra! Ehhez váltsunk vissza a gyak1.m script fájlra!

```
> negyzetfv(11) % 121
> negyzetfv(x) % 1  4  9  16  25  36  49  64  81  100
> help negyzetfv %  Kiszámolja a bemenet négyzetét
```

Egy függvénynek lehet több bemenete is, pl módosítsuk az előző függvényünket az alábbi módon, és mentsük el hatvany.m néven!

```
> function y = hatvany(x,p)
>     y = x.^p
> end
```

Egy függvénynek lehet több kimenete is egy vektorban összegyűjtve (hatvanyok.m):

```
> function [x2 x3 x4] = hatvanyok(x)
>     x2 = x.^2;
>     x3 = x.^3;
>     x4 = x.^4;
> end
```

Hívjuk meg a fenti függvényeket is a script fájlunkból!

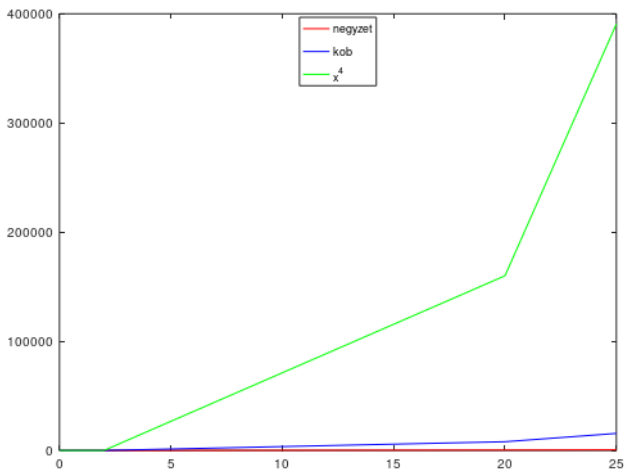
```
> hatvany(x,3) % 1  8  27  64  125  216  343  512  729  1000
> [a b c] = hatvanyok(x)
> % a = 1 4  9 16 25 36 49 64 81 100
> % b = 1 8 27 64 125 216 343 512 729 1000
> % c = 1 16 81 256 625 1296 2401 4096 6561 10000
```

Ábrázoljuk az eredményeket egy új 2-es számú ábrán a **figure** parancs használatával! A **plot** parancsnál egymás után több összetartozó értéket is felsorolhatunk!

```
> figure(2)
> plot(x,a,x,b,x,c)
```

Mi is adhatunk egyéni színeket hozzá, illetve jelmagyarázatot is fűzhetünk hozzá olyan sorrendben megadva a jelmagyarázat szövegét, ahogy felrajzoltuk az ábrákat:

```
> plot(x,a,'red',x,b,'blue',x,c,'green')
> plot(x,a,'r',x,b,'b',x,c,'g')
> legend('negyzet','kob','x^4','Location','North')
```



A jelmagyarázatnál megadhatjuk égtájak szerint angolul, hogy hová kerüljön a jelmagyarázat, Matlab-ban van egy olyan lehetőségünk is, hogy 'Best' módszer megadása esetén megpróbálja eldönteni a program, hogy hol zavar a legkevésbé a felirat (ez Octave-ban nem működik).

```
> legend('negyzet','kob','x^4','Location','Best')
```

PROGRAM KOMMENTEK

A több sorból álló programok fontos részei a kommentek. Egyrészt mások is megértik a programkódunkat, másrészt mi is emlékezni fogunk rá, ha később újra használni akarjuk. Célszerű nem csak a program elején, hanem minden új szekcióhoz is kommenteket írni. A Matlab-ban % jel után írhatunk kommenteket. Egy függvény esetében a kommentben célszerű megadni, hogy mi a függvény célja, milyen bemeneti és kimeneti értékek szerepelnek benne. Függvény esetében az első sor után megadott kommentek fognak megjelenni a help utasítás hatására segítségként.

MATLAB HIBAÜZENETEK

A programok írása során számos hibaüzenettel találkozunk, eddig is láttunk már néhányat. Fontos, hogy tudjuk ezeket értelmezni, ez segíthet kijavítani a hibáinkat! (piros – Matlab, kék – Octave). Nézzünk egy példát elírásra a clear all helyett!

```
> cler all
Undefined function or variable 'cler'.
Did you mean:
>> clear all
error: 'cler' undefined near line 1 column 1
```

A Matlab érzékeny a kis nagy betűk változására is:

```
> x = 3/4; x
Undefined function or variable 'x'.
error: 'x' undefined near line 1 column 10
```

Nézzünk példát egy szintaktikailag hibás Matlab utasításra:

```
> 1 x
```

```
1 x
   ↑
Error: Unexpected MATLAB expression.
parse error:
  syntax error
>> 1 x
     ^
```

Túl sok bemenő paraméter:

```
   >  sin(pi,3)
Error using sin
Too many input arguments.
error: Invalid call to sin.  Correct usage is:
 -- sin (X)
```

Nem egyezik a mátrixban lévő sor/oszlop elemeinek száma:

```
   >  M = [1 2;3]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
error: vertical dimensions mismatch (1x2 vs 1x1)
```

```
   >  [3, 4, 5] * [1; 2; 3; 4]
Error using  *
Inner matrix dimensions must agree.
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 4x1)
```

```
   >  a = 1:5, b = 1:3
   >  [a;b]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
```

Könnyen elgépelhető hiba lehet, hogy zárójel helyett 8 v 9 kerül beírásra:

```
   >  sin(pi9
sin(pi9
       ↑
Error: Expression or statement is incorrect--possibly unbalanced (, {,
or [.
```

Elemenkénti műveletet akarunk végezni vektoron, de lemarad a pont:

```
   >  v =1:4;
   >  1/v
Error using  /
Matrix dimensions must agree.
```

A legrosszabb, amikor nincs hibaüzenet, az eredmény mégis hibás. Példa: számítsuk ki $\frac{1}{2\pi}$ értékét a következő utasítással! Miért hibás az eredmény?

```
   >  1 / 2*pi %  1.5708
```

LOGIKAI MŰVELETEK

A logikai műveletek (1-igaz/0-hamis) ismerete is nagyon fontos, különösen, ami a vektorok/mátrixok elemeinek módosítását, lekérdezését illeti. Sok olyan feladatot is meg tudunk oldani a mátrixokkal és logikai változókkal, amihez különben valamilyen ciklus kellene.

```
> % egyenlo ==, nem egyenlo ~=
> b = 3==4
> whos b
> b = 5~=6
> vs = [1 2 3 4 5 6] % sorvektor
> vs(5)>5
> vs(5)>=5
> % vagy (or) ||, es (and) &&
> vs(1)>2 || vs(4)>2 % igaz, mert a ket feltetel kozott van igaz
> vs(1)>2 && vs(4)>2 % hamis, mert csak az egyik feltetel igaz
```

Nézzünk egy példát, ahol egy vektor adott tulajdonságú elemeit kérdezzük le logikai változóval. Legyen egy műegyetemi tanár, aki véletlenszerűen osztogatja a jegyeket a diákoknak a vizsgán. Most éppen 6 vizsgázója van (a,b,c,d,e,f). Kapjon mindenki egy jegyet 1-5 között és utána kérdezzük le hányan buktak és kik?

```
> vizsgazok = ['a';'b';'c';'d';'e';'f']
> vizsga = ceil(rand(1,6)*5)
> bukott = vizsga<2
> vizsgazok(bukott)
```

A bukott=vizsga<2 eredménye egy 6 elemű vektor lesz, ahol 1-es áll azokon a helyeken, amire igaz volt a feltétel, 0 a többin. Ha a vizsgázók nevei közül le akarjuk kérdezni azokat, akik megbuktak, akkor nem kell mást tennünk, mint meghívni a vizsgazok(bukott) parancsot, ez csak azokat a neveket fogja visszaadni, ahol 1-es állt a bukott vektorban. Egy ilyen lekérdezést matlab-ban ciklus nélkül is meg lehet oldani, logikai változókat használva. Megjegyzés: kerekítési parancsok: round, ceil, floor, fix (lásd help)

FORMÁZOTT SZÖVEGEK

Gyakran van szükség arra, hogy az eredményeinket egy adott formátumban jelenítsük meg. Nézzük például a szögekkel való műveleteket. A legtöbb matematikai művelet végzésére alkalmas szoftver (pl. Matlab, Octave, Excel…) a szögeknél a radiánt tekinti alapértelmezettnek, ha ettől eltérő formátumban szeretnénk látni az eredményeket, akkor nekünk kell erről gondoskodni. Matlab és Octave alatt a radiánban értelmezett trigonometriai függvényeknek (pl. sin, cos, tan, atan, atan2…) megvannak a fokokkal számoló változatai (pl. sind, cosd, tand, atand, atan2d…), de ha már fok-perc-másodpercben szeretnénk megjeleníteni az eredményeket (pl 302-06-23), esetleg adott számú tizedesjegyre (23-03-48.5831), akkor ezt a formázott szövegekkel tehetjük meg. Ugyanígy, ha pl. képeket szeretnénk automatikusan elnevezni egy ciklusban pl. IMG0001, IMG0002 stb. akkor ehhez is a formázott szövegeket hívhatjuk segítségül.

Az `fprintf` paranccsal a fájlba írhatunk formázott szövegeket vagy a képernyőre, az `sprintf` használatával pedig egy stringbe (szöveges változóba)/képernyőre. A következő formátumjelölőket alkalmazhatjuk:

- %d – egész szám, %s – szöveg, %f – valós szám (lebegőpontos), %c – karakter, %u – nem előjeles egész
- %e – normál alak pl. 3.14e+00, %E – 3.14E+00
- %g – kompakt forma, %f vagy %e közül a rövidebbik, fölös 0-k nélkül

A típust jelző betű előtt szerepelhet még pl. + jel, akkor előjelesen írja ki a számot; mező szélesség; tizedesjegyek száma; 0, akkor 0-kal tölti fel elől az üres helyeket a mező szélességéig.

Próbáljuk ki a következőket!

```
> clc; disp('Hány éves a kapitány?')
> ev = 33; ho = 3; nap = 3;
> ev2 = ev + ho/12 + nap/365;
> fprintf('A kapitány 33 éves')    % nem rak be sortörést a végére
> fprintf('A kapitány 33 éves\r\n')% \r\n - sortörés
> sprintf('A kapitány 33 éves')   % szöveges változóba teszi az
  eredményt
> sprintf('A kapitány %d éves, %d hónapos és %d napos',ev,ho,nap) %  'A
  kapitány 33 éves, 3 hónapos és 3 napos'
> sprintf('A kapitány %f éves', ev2) % 'A kapitány 33.258219 éves'
> sprintf('A kapitány %.2f éves', ev2) % 'A kapitány 33.26 éves'
> sprintf('A kapitány %8.2f éves', ev2) % 'A kapitány    33.26 éves'
> sprintf('A kapitány %08.2f éves', ev2) % 'A kapitány 00033.26 éves'
> sprintf('A kapitány %+6.2f éves', ev2) % 'A kapitány +33.26 éves'
```

A `%+6.2f` kifejezés azt jelzi, hogy 6 karakteren írja ki a változót (tizedespontot, előjelet is beleértve!), egy valós számot (`f`), ahol a tizedesjegyek száma 2 (`.2`), és kiírja az előjelet is (`+`). Ha `0` is szerepel benne, akkor az üres helyeket 0-kkal tölti ki. (`0`). Ha az eredmény hosszabb lenne, mint a mező szélessége, akkor nem veszi figyelembe a megadott mező szélességet.

Ha az eredmény hosszabb, mint a mező szélessége, akkor nem veszi figyelembe a megadott mező szélességet. Pl.

```
> sprintf('%2.5f',1/eps) % 4503599627370496.00000
> sprintf('%2.5g',1/eps) % 4.5036e+015
```

SZÖGEK KIÍRÁSA FOK-PERC-MÁSODPERCBEN (GEODÉZIAI FORMÁTUM)

Írjunk egy saját programot, ami a tizedfokban megadott eredményünket a geodéziában szokásos fok-perc-másodperc formában írja ki. A perc, másodperc értékét mindig két számjeggyel írja ki pl. 192-03-12.

```
> function str = fpm(x);
> % A fuggveny tizedfokbol fok-perc-masodperc ertekekbe szamol at.
> % A kimenet egy formázott szöveg (ddd-mm-ss)
>   f = fix(x);
>   p = fix((x-f) .* 60);
>   m = ((x-f).*60-p).*60;
>   str = sprintf('%4d-%02d-%02.0f', f, abs(p), abs(m));
> end
```

Próbáljuk ki! (Az abszolút érték a negatív szögek jó megjelenítéséhez kell.)

```
> x = 123.456789, y = -23.00987
> fpm(x), fpm(y)
```

CIKLUSOK

Persze azért mindent nem lehet ciklusok használata nélkül megoldani, úgyhogy ismerkedjünk meg kicsit a ciklusokkal! Kétféle ciklust használunk, számlálással és feltétellel vezérelt ciklust.

SZÁMLÁLÁSSAL VEZÉRELT CIKLUS

Írjunk egy programot, ami kirajzolja x^1, x^2, x^3, x^4, x^5 függvényt [-10, 10] intervallumon és el is menti ezeket képként! Először csak plottoljuk ki x^2 függvényt és mentsük el!

```
> x = -10:0.1:10
> y = x.^2
> plot(x,y)
> print('proba.jpg','-djpeg')
```

Nézzük meg a megoldást számlálással vezérelt for ciklust használva!

```
> for i=1:5
>   y = x.^i;
>   plot(x,y);
>   fajlnev = sprintf('hatvany%4d.jpg',i)
>   print(fajlnev,'-djpeg')
>   title(sprintf('x^%d függvény',i)
> end
```

Az **sprintf** függvény segítségével tudunk egy szövegbe változókat beírni, adott formátumban. A **%** jel jelzi, hogy itt egy változó következik, utána a **d** jelölő az egész szám típust jelenti. A **%d** helyére egy egész szám típusú változót vár a program, amit a vessző után tudunk megadni, pl. most az *i* változó tartalma egy sorszám.

FELTÉTELLEL VEZÉRELT CIKLUS

Egy másik műegyetemi oktató kicsit tudományosabban adja a jegyeket a vizsgáján, nem egyenletes eloszlásban, hanem normális eloszlásban, 3-as várható értékkel és 1-es szórással. Vajon hányszor kell elmenjek hozzá vizsgázni, ha 5-öst szeretnék kapni? Írjunk egy programot, ami előállít egy véletlen számot először egy for ciklusban 100-szor, 3-as várható értékkel, 1-es szórással, egészre kerekítve. Előfordulhat, hogy 1-esnél rosszabb vagy 5-ösnél jobb jegyet kapunk (nézzük meg a min, max értékét!). Ezeket logikai változók használatával módosítsuk 1-esre és 5-ösre. Ábrázoljuk hisztogramon! Majd írjunk egy feltétel vezérelt ciklust, ami a saját vizsgáinkra vonatkozik, ami akkor áll le, ha végre 5-öst kaptunk!

Először nézzünk meg 100 véletlenszerű osztályozást!

```
> r = round(randn(1,100))+3
> min(r), max(r)
> r(r<1)=1; r(r>5)=5;
> min(r), max(r)
> hist(r)
```

Most a saját vizsgáinkat nézzük, addig fusson a ciklus, amíg 5-öst (vagy jobbat) nem kapunk! Hányadikra sikerült?

```
> r = 0; i =0; R = [];
> while r<5
>   r = round(randn(1)+3);
>   R = [R r];
>   i = i+1;
> end
> R
> i
```

Ha szükséges a `break` utasítással ki lehet lépni pl. bizonyos feltétel teljesülése esetén mind a `for`, mind a `while` ciklusból. A `continue` parancs adott feltétel esetén kihagyja a ciklus további részét és a következő iterációs lépésre ugrik.

ELÁGAZÁSOK

MÁSODIK GEODÉZIAI FŐFELADAT MEGOLDÁSA

Írjunk saját függvényt a második geodéziai főfeladat meghatározására, vagyis két koordináta pár alapján távolság és irányszög meghatározására. Vegyük figyelembe a különböző síknegyedeket is!

A képletek: $t = \sqrt{\Delta x^2 + \Delta y^2}$; $\alpha = arctan\left|\frac{\Delta y}{\Delta x}\right|$

Síknegyedenként: 1. $\delta = \alpha$, 2. $\delta = 180° - \alpha$, 3. . $\delta = 180° + \alpha$, 4. $\delta = 360° - \alpha$

```
> function [t delta] = tav_irany(y1,x1,y2,x2)
>       dy = y2 - y1;
>       dx = x2 - x1;
>       t = sqrt(dy^2+dx^2);
>       alfa = atand(abs(dy/dx));
>       if dy>=0 && dx>=0
>             delta = alfa;
>       elseif dy>=0 && dx<0
>             delta = 180 - alfa;
>       elseif dy<0 && dx<0
>             delta = 180 + alfa;
>       else
>             delta = 360 - alfa;
>       end
> end
```

Ellenőrizzük le parancssorból a 4 negyedet!

```
> tav_irany(0,0,10,10)
> tav_irany(0,0,10,-10)
> tav_irany(0,0,-10,-10)
> tav_irany(0,0,-10,10)
```

Egyszerűbb a megoldás, ha ismerjük az `atan2` (eredménye radian) illetve `atan2d` (eredménye fok) parancsot! Ez síknegyedekre lebontva számolja a szögeket, az utóbbi -180 és +180 fok között. Itt csupán 360 fokot hozzá kell adni a negatív értékekhez.

```
> function [t delta] = tav_irany2(y1,x1,y2,x2)
>       dy = y2 - y1;
```

```
>       dx = x2 - x1;
>       t = sqrt(dy^2+dx^2);
>       delta = atan2d(dy,dx);
>       if delta<0 delta=delta+360; end
> end
```

Az elágazások másik alakja a switch-case utasítások.

```
> muszer = menu('Milyen muszert kersz?','Szintezo', 'Teodolit', 'GPS');
> switch muszer
>   case 1
>     disp('Szintezo kiadva!')
>   case 2
>     disp('Teodolit kiadva!')
>   case 3
>     disp('GPS kiadva!')
> end
```

TRY - CATCH ALGORITMUS

Sokszor előfordul, hogy bizonyos műveletek során hiba lép fel. Alapesetben erre leáll a program futása. Ha a hiba ellenére is szeretnénk tovább futtatni a programot, akkor használhatjuk a try-catch algoritmus, ilyenkor megadhatjuk, hogy hiba esetén (pl. hiányzó fájl, amit be akarunk tölteni, hibás művelet) mi történjen. Próbáljuk ki a következőt!

```
> A = rand(5,3)
> B = rand(5,4)
> C = [A; B]
> D = [A B]
```

Hibaüzenettel leállt a program a C után. Az alábbi esetben viszont csak egy figyelmeztető üzenetet ír ki, de fut tovább.

```
> try
>    C = [A; B]
> catch
>    disp('Nem egyeznek a matrixok meretei')
> end
> D = [A B]
> disp('Itt a vege!')
```

HASZNÁLT FÜGGVÉNYEK

`page_screen_output(0)`	-	csak Octave-ban van, a műveletek eredményét ne oldalakra tördelve jelenítse meg
help	-	matlab helpjének kategóriái, vagy segítség megadott témakörhöz, függvényhez
rand	-	Véletlen számok 0-1 között egyenletes eloszlásban
randn	-	Véletlen számok sztenderd normális eloszlásban, 0 várható értékkel és 1 szórással
doc	-	részletes dokumentáció adott függvényhez, parancshoz
lookfor	-	keresés a help-ben adott szóra, szórészletre
clc	-	kitörli a command window ablak tartalmát
clear, clear all	-	kitörli a megadott változókat, vagy az összes változót
close, close all	-	bezárja az aktuális ábrát, vagy az összeset
CTRL+C	-	félbeszakítja az adott parancsot (kilépés pl. végtelen ciklusból)
%	-	megjegyzés (a program figyelmen kívül hagyja ami ez után van a sorban)
;	-	parancs végén a ; hatására nem jelenik meg az eredmény a Command Window-ban
Tab gomb	-	elkezdett parancsot kiegészíti
↑↓ gombok	-	korábbi parancsokat vissza lehet hozni a Command Window-ba
preferences	-	megnyitja a beállítások ablakot
pi	-	3.14…. (pi szám)
exp(1), exp(n)	-	e^1 =2.71…, e^n
^	-	hatványozás
format long/short	-	több tizedes jegy megjelenítése/ rövidebb megjelenítés
[1, 2, 3; 4, 5, 6]	-	vektor, mátrix megadása
'	-	vektor, mátrix transzponáltja
[A,B] vagy [A B]	-	mátrixok összefűzése egymás mellé (sorok száma egyenlő)
[A;B]	-	mátrixok összefűzése egymás alá (oszlopok száma egyenlő)
A(1,:), A(:,1), A(:,end)	-	mátrix első sora, mátrix első/utolsó oszlopa
linspace(x1,x2,n)	-	[x1,x2] intervallumban n pont felvétele egyenletesen
ones, zeros	-	egyesekből álló mátrix, nullákból álló mátrix
eye	-	egységmátrix
figure	-	új ábra nyitása
plot	-	összetartozó pontpárok felrajzolása
xlabel, ylabel	-	x,y tengely feliratozása
title	-	ábra címe
'LineWidth'	-	vonalvastagság
'MarkerSize'	-	pont szimbólum mérete
'MarkerEdgeColor'	-	pont szimbólum határvonalának a színe
'MarkerFaceColor'-	-	pont szimbólum kitöltésének színe

sin, cos, tan	-	szögfüggvények (alapértelmezett mértékegység a radián!)
log	-	természetes alapú logaritmus
log10	-	10-es alapú logaritmus
sqrt	-	négyzetgyök
abs	-	abszolút érték
hold on, hold off	-	felülírja, vagy ne írja felül a meglévő ábrát az új ábrával
ezplot, fplot	-	függvények felrajzolása
.* ./ .^	-	elemenkénti szorzás, osztás, hatványozás vektoroknál
clf	-	ábra törlése (nem zárja be az ablakot)
legend	-	jelmagyarázat
function	-	saját függvény kezdete
==	-	Logikai egyenlőség
~=	-	Logikai 'nem egyenlő'
and(felt1, felt2), felt1 && felt2,	-	Logikai ÉS
or(felt1, felt2), felt1 || felt2,	-	Logikai VAGY
for	-	Számlálással vezérelt ciklus
while	-	Feltétellel vezérelt ciklus
break	-	Ki lehet lépni pl. bizonyos feltétel teljesülése esetén mind a `for`, mind a `while` ciklusból
continue	-	Adott feltétel esetén kihagyja a ciklus további részét és a következő iterációs lépésre ugrik
sprintf	-	Formázott szöveg írása string típusú változóba
round	-	Kerekítés matematikai értelemben
min, max	-	Egy vektor legkisebb/legnagyobb eleme
hist	-	Ábrázolás hisztogramon
atan, atand	-	Arkusz tangens függvény radián, fok mértékegységben
atan2, atan2d	-	Arkusz tangens függvény radián, fok mértékegységben, a síknegyedek figyelembe vételével a teljes körre vonatkozóan
if, elseif, else, end	-	Kétirányú feltételes elágazás
switch, case	-	Többirányú elágazás
menu	-	Felugró menü készítése listából kiválsztáshoz
try, catch,end	-	Hibák kezelése, ha hiba lép fel, akkor helyette mit történjen

2. FÁJLMŰVELETEK

EGYSZERŰ ADATBEOLVASÁS/KIÍRÁS (LOAD, SAVE)

A legegyszerűbb adatbeolvasás/kiírás a `load` illetve `save` paranccsal történhet. Nézzünk rá példát! Kezdjük ezt a gyakorlatot is egy új script fájl írásával, válasszuk ki az aktuális könyvtárat, ahová dolgozunk, adjuk meg, hogy az eredményeket ne laponként hanem folyamatosan írja ki a képernyőre (ehhez a page_screen_output változó értékét 1-ről 0-ra kell állítani), majd töröljünk ki minden korábbi változót a workspace-ből. Amikor elmentjük a fájlt, figyeljünk, hogy nem kezdődhet számmal a fájlnév, nem lehetnek benne szóközök és ékezetes betűk, de a nagybetűket is célszerű kerülni! Legyen ez pl. gyak3.m fájl!

```
> page_screen_output(0)
> clc; clear all;
```

Hozzunk először létre egy 4x4-es (*A*) és egy 4x1-es (*b*) mátrixot/vektort, 0 és 1000 között egyenletes eloszlással, majd mentsük el későbbi felhasználásra! Hozzunk létre egy *A1* 4x4-es és egy *b1* 4x1-es mátrixot/vektort is, csak most 0 várható értékkel 1000 szórással! Mentsük el ezt is!

```
> A = rand(4)*1000
> b = rand(4,1)*1000
> save veletlen.mat A b
> A1 = randn(4)*1000
> b1 = randn(4,1)*1000
> save('veletlen2.mat','A1','b1')
```

Vegyük észre, hogy a save-et kétféleképpen is meghívhatjuk, itt először parancsként, másodjára függvényként hívtuk meg. A függvényként hívás akkor lehet célszerű, ha paraméterezni szeretnénk, és pl. egy ciklusból többször meghívni különböző automatikusan előállított fájlnevekkel. Ha nem adunk meg változóneveket a fájlnév után, akkor a workspace összes aktuális változóját elmenti.

Töröljünk ki mindent és töltsük be az eredményeket fájlból! Fűzzük össze az A és b ill. A1 és b1 változókat, tegyük vízszintesen egymás mellé őket!

```
> clear all; clc;
> load veletlen.mat
> load('veletlen2.mat')
> Ab = [A b]
> Ab1 = [A1 b1]
```

Mentsük el a matlab/octave *.mat kiterjesztésű bináris fájlja helyett egy szöveges fájlba az Ab változót, majd fűzzük utána az Ab1 változót! Töröljünk ki mindent és töltsük be az Ab.txt fájlt! Nézzük meg mi van az Ab változóban és az Ab.txt fájlban!

```
> save Ab.txt Ab -ascii
> save Ab.txt Ab1 -ascii -append
> clear all;
> load Ab.txt
> Ab
> type Ab.txt
```

Eredmény:

```
Ab =
    603.976    588.999    698.522    799.056    597.727
    189.623    258.823    810.762    741.362    755.103
...
 6.03975786e+002     5.88999045e+002     6.98521807e+002     7.99055866e+002
5.97726799e+002
 1.89622699e+002     2.58823496e+002     8.10761738e+002     7.41362023e+002
7.55102519e+002
...
```

Megjegyzések: Szöveges fájl beolvasásánál azonos típusú adatok és azonos sorhosszak szükségesek. Az eredmény egy mátrixba kerül, amiben csak azonos típusú elemek lehetnek.

save-vel elmenthetjük a munkakörnyezet összes vagy néhány változóját *.mat kiterjesztésű MATLAB bináris fájlba, amiből utána **load**-dal ezek betölthetőek, de ez más program számára nem igazán értelmezhető formátum. Ha szöveges állományba kívánjuk menteni egy mátrix tartalmát, akkor a **save filename variables -ascii** paranccsal tehetjük meg. Ha egy létező fájlhoz akarunk hozzáírni valamit, akkor a **save filename variables -ascii -append** parancsot használhatjuk. Formázott szövegeket az **fprintf** használatával írhatunk ki.

FORMÁZOTT KIÍRÁS (FPRINTF)

Látjuk, hogy a **save** eredményeképpen a szöveges állományba normálalakban kerültek a számok. Szeretnénk inkább hagyományosan kiírni a számokat, 3 tizedes jegyre! Ehhez a múlt órán már tanult formázott szöveg kiírását válasszuk, csak most nem az **sprintf**, hanem az **fprintf** utasítást! Ehhez először meg kell nyissuk írásra a fájlt, majd elvégezni a műveleteket, végül lezárni. Ezek általánosan a következőképp néznek ki:

- fájl megnyitása (**fopen**)
- beolvasás, írás, hozzáfűzés a fájlhoz
- fájl bezárása (**fclose**)

Az **fopen** használata során megadhatjuk, hogyan kívánjuk megnyitni a fájlt, 'r'-csak olvasásra (alapértelmezett, ha nem adunk meg semmit), 'w'-írásra, 'a'-hozzáfűzéshez:

`fileID = fopen(filename,'w')` – fájl megnyitásra írásra

A fájlokat bezárhatjuk egyenként: **fclose(fileID)**, vagy egyszerre az összeset: **fclose('all')**.

```
> fid = fopen('veletlen3.txt','w');
> for i=1:size(Ab,1)
>   fprintf(fid,'%9.3f ',Ab(i,:));
>   fprintf(fid,'\r\n');
> end
> fclose(fid);
> type veletlen3.txt
```

A fenti művelet megoldható ciklus nélkül is, ha tudjuk, hogy hány elem van egy sorban:

```
> fid = fopen('veletlen3.txt','w');
> fprintf(fid,'%9.3f %9.3f %9.3f %9.3f %9.3f\r\n',Ab);
> fclose(fid);
> type veletlen3.txt
```

ADATOK BEOLVASÁSA/KIÍRÁSA FÁJLBA

Az előző órán láttuk, hogy load és save használatával be lehet olvasni, illetve el lehet menteni szöveges állományokat. A szöveges fájl beolvasásánál azonos típusú adatok és azonos sorhosszak szükségesek. Az eredmény egy mátrixba kerül, amiben csak azonos típusú elemek lehetnek. A **save** paranccsal elmenthetjük a munkakörnyezet összes vagy néhány változóját *.mat kiterjesztésű MATLAB bináris fájlba, amiből utána **load**-dal ezek betölthetőek, de ez más program számára nem igazán értelmezhető formátum. Ha szöveges állományba kívánjuk menteni egy mátrix tartalmát, akkor a **save filename variables -ascii** paranccsal tehetjük meg. Ha egy létező fájlhoz akarunk hozzáírni valamit, akkor a **save filename variables -ascii -append** parancsot használhatjuk. Formázott szövegeket az **fprintf** használatával írhatunk ki. Mi helyzet azonban akkor, amikor ennél bonyolultabb szerkezetű fájlt kell beolvasni? Nem azonos típusú adatok szerepelnek benne, nem azonos az egy sorban lévő adatok száma? Ilyenkor más megoldást kell keresnünk a beolvasásra.

SORONKÉNTI BEOLVASÁS (FGETL, FGETS)

Az **fgetl** és **fgets** parancsokkal soronként lehet beolvasni egy fájl tartalmát. Az **fgetl** levágja belőle a sorvége karaktert (\n vagy \r\n)[2], míg az **fgets** megtartja. A beolvasás eredménye egy string változóba kerül. Az egész fájl tartalom beolvasásához egy feltételes ciklusra van szükség (**while**), hogy addig olvasson, amíg el nem érünk a fájl vége jelhez (**feof** - end-of-file).

Olvassuk be a következő állományt, amiben betűk és számok is vannak, szambetu.dat:

```
> 5.3 a
> 2.2 b
> 3.3 a
> 4.4 a
> 1.1 b
```

Először csak nyissuk meg az állományt és olvassunk be két sort. Megj.: A fájl megnyitása után egy fájl pointer figyeli, hogy épp hányadik bájtig olvastuk be a fájlt, amit akár le is kérdezhetünk az ftell(fid) paranccsal.

```
> clear all; clc;
> type szambetu.dat
> fid=fopen('szambetu.dat');
> line=fgetl(fid) % egy sor beolvasása
> line=fgetl(fid) % egy sor beolvasása
> fclose(fid);
```

Jó lenne ezt egy ciklusba tenni, ami addig futna, amíg a fájl végére nem érünk, akkor nem kell tudni előre hány sort olvassunk be! Használjunk ehhez egy feltétel vezérelt ciklus, amíg a fájl végére nem érünk (feof igaz nem lesz).

```
> fid=fopen('szambetu.dat');
> while feof(fid)==0
```

[2] A sor vége jel Windows esetében: \r\n, Mac (OS 9-) esetében \r, Unix/Linux esetében: \n.

```
>    line=fgetl(fid) % egy sor beolvasása
>  end
>  fclose(fid);
```

Most minden új sor beolvasásakor felülírjuk az előzőt. Jó lenne elmenteni külön egy mátrixba a számokat és külön a betűket! Nézzük meg hogyan tudjuk szétválasztani őket!

```
>  szam = str2num(line(1:3))
>  betu = line(5)
```

Ez az egyik lehetőség, ha ismerjük, hogy hány karakter a szám az elején és hol van a betű, a másik lehetőség, hogy az első szóköz mentén szétvágjuk a szöveget. Van még sok szöveges állomány kezelő parancs, amiket érdemes lehet nézegetni, ha ilyen feladata van az embernek (lásd pl. az Octave Documentation fülnél a string menü, vagy https://www.mathworks.com/help/matlab/characters-and-strings.html).

```
>  [szam betu] = strtok(line)
>  szam = str2num(szam)
```

Válasszuk ki az egyik megoldást és a ciklusban fűzzük össze egy-egy tömbbe a számokat, betűket!

```
>  szamok=[]; betuk=''; % számok és betűk tömbök létrehozása
>  fid=fopen('szambetu.dat');
>  % Sorok beolvasása, számok, betűk szétválasztása
>  while feof(fid)==0 % ciklus, amíg a fájl végére érünk
>     line=fgetl(fid); % egy sor beolvasása
>     % szöveg szétválasztása számra, betűre
>     szam = str2num(line(1:3))
>     betu = line(5)
>     szamok=[szamok;szam]; % szam hozzáfűzése a szamok tömbhöz
>     betuk=[betuk;betu]; % betű hozzáfűzése a betuk tömbhöz
>  end
>  fclose(fid);
>  szamok, betuk
>  % Írassuk ki a számok összegét a képernyőre!
>  szumma=sum(szamok);
>  fprintf('A szamok osszege: %.2f\n',szumma)
```

Az eredmény: A számok összege: 16.30

BEOLVASÁS FSCANF, TEXTSCAN HASZNÁLATÁVAL

Az **fscanf** paranccsal egyszerre az egész fájlt be tudjuk olvasni egy megadott formátum szerint. Az eredmény egy mátrixba kerül. Ezzel a paranccsal különböző hosszúságú sorokat nem tudunk beolvasni, azoknál célszerű az **fgetl**, **fgets** parancsot használni. Nézzük meg a számok/betűk szétválasztását az **fscanf** parancsot használva!

A beolvasás oszloponként történik, megadjuk a formátumot, amiben az adatok le vannak tárolva most először egy szám, szóköz, aztán egy betű. Ekkor először az első oszlop kerül beolvasásra, amiben a számok vannak, ez kerül az első sorba, utána a második oszlop a betűkkel, ez kerül a második sorba. Transzponáljuk, hogy a forma megfeleljen az eredetinek.

```
>  clear all; clc;
```

```
> fid=fopen('szambetu.dat');
> mat=fscanf(fid,'%f %s',[2 inf])'
> fclose(fid);
```

Eredmény:

```
5.3000   97.0000
2.2000   98.0000
3.3000   97.0000
4.4000   97.0000
1.1000   98.0000
```

Miért lett ilyen furcsa az eredmény? Hol vannak a betűk? Mivel egy mátrixban csak egyféle adatot lehet tárolni (csak szám vagy csak szöveg), a betűk helyett azok ascii kódja került eltárolásra. Válasszuk szét a változókat és alakítsuk vissza szöveggé!

```
> szamok = mat(:,1)
> betuk = char(mat(:,2))
```

Az `fscanf`-hez hasonlóan használható az `sscanf`, csak ez nem fájlból olvas be adott formátum szerint, hanem stringből!

Hogyan tudnánk az adatokat úgy beolvasni, hogy a különböző formátumok ne okozzanak gondot? Van egy másik változó típus, a cellatömb, amiben különböző típusú változókat is eltárolhatunk. A megadása hasonló a mátrixokhoz, csak itt kapcsos zárójelet {} használunk szögletes [] helyett. A **`textscan`** parancs hasonló az **`fscanf`**-hez, de ez cellatömbbe olvas be adatokat.

```
> clear all; clc;
> fid=fopen('szambetu.dat');
> adatok=textscan(fid, '%f %s');
> fclose(fid);
> % Eredmények szétválasztása a cella tömbből
> szamok=adatok{1} % mátrix számokkal
> betuk=adatok{2} % ez egy cellatömb karakterekkel
> betuk{1,1} % ez már maga az eltárolt betű
> betuk = char(betuk) % karaktertömböt (vektort) készít a cellákból
```

Nézzünk még egy példát az fscanf használatára. Olvassuk be az *xypoints.dat* állományt!

```
x2.3y4.56
x7.7y11.11
x12.5y5.5
```

Itt x,y koordináták vannak, csak előttük ott áll, hogy x vagy y. A számok különböző hosszúak, 3-5 karakteren tárolódnak, így a karakterszám szerinti szétválasztás nem megy. Formázott szövegként viszont be tudjuk olvasni. Ha az fscanf vagy textscan részeként konkrét szöveget adunk meg (itt pl. x vagy y), akkor azok nem kerülnek beolvasásra, hanem csak a köztük lévő számok, amit pl. %f alakban adhatunk meg. Most nincs problémánk a különböző típusokkal, hiszen a lényeges információ csak számokból áll.

```
> fid = fopen('xypoints.dat')
> xy = fscanf(fid, 'x%fy%f\r\n', [2 inf])'
> fclose(fid)
```

Érdemes tanulmányozni ezeknek a parancsoknak a help-jét, mivel számtalan opciót lehet még megadni. Pl. **`fscanf`**-nél, ha %*s vagy %*f alakot adok meg, akkor átugrik egy bizonyos karaktert/számot, **`textscan`** parancsnál meg lehet adni fejlécet,

kommentstílust, amiket kihagy a beolvasásból, meg lehet adni elválasztót a szövegek között (szóköz, pont, vessző...) stb.

Leggyakrabban azonban az fgetl parancsot használjuk a beolvasásra, amikor nem adott formában vannak az adatok, esetleg változó sorhosszúságúak és egyenként kell minden sort megvizsgálni.

INTERFEROMÉTERES ADATOK BEOLVASA (FSCANF)

Nézzünk egy geodéziai példát bonyolultabb adatbeolvasásra. A következő fájl interferométerrel készült méréseket tartalmaz. Van egy fejléc része (ez mindig ugyanannyi sorban van tárolva, most épp 26-ban), utána jönnek a tényleges mérések (ennél nem tudjuk előre a sorok számát), majd a végén még egyéb adatok jönnek (pl. légnyomás stb.).

teszt_interfero.txt

```
HEADER
File type  : rtl
…
Run Target Data:
1   1       7.403
1   2    -994.335
2   2   -1008.836
…
ENVIRONMENT::
Air temp   : 22.445311 22.460936 0
…
EOF
```

A lényeges információ 3 oszlopban van, és tetszőleges számú sorban. Ezt lenne jó beolvasni egy mátrixba valamilyen módon. Itt nem használhatjuk az end-of-file opciót a ciklushoz, mert az adatok nem a fájl végéig tartanak, más módot kell keresnünk. Az első 26 sornyi fejlécet könnyen át tudjuk ugrani, ha **`fgetl`** paranccsal beolvasunk 26 sort. A többit beolvashatjuk formázott szövegként az **`fscanf`** paranccsal. Ez ugyanazokat a formátumokat használja, mint a korábban már használt sprintf, fprintf. Megadhatjuk neki, hogy 3 szám van egy sorban szóközzel (vagy tabbal *\t* elválasztva, jelen esetben mindegy melyiket használjuk), és megadhatjuk a méretet is. Az fscanf oszlopokat olvas be, amikor a méretet adjuk meg, először az oszlopok számát kell megadni, és utána a sorokét, de ezt állíthatjuk előre nem meghatározottra (itt végtelen - inf) is. A parancs leáll, ha olyan sort talál, ami nem felel meg a formátumnak.

```
> fid = fopen('teszt_interfero.txt');
> for i=1:26; fgetl(fid); end % Kihagyjuk az első 26 sornyi fejlécet
> % számok beolvasása 3 oszlopba, az első szövegnél leáll
> interfero=fscanf(fid,'%f %f %f\n',[3 inf])
> fclose(fid);
> % transzponáljuk hogy 3 oszlopban legyenek az adatok, ne 3 sorban
> interfero = interfero'
```

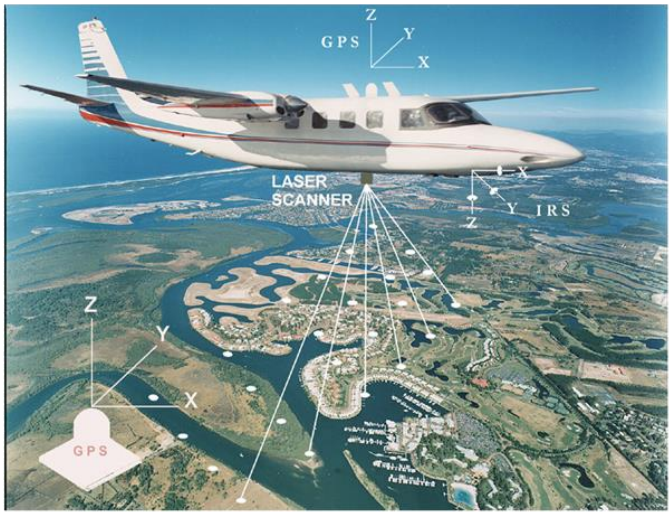
FONTOSABB INPUT/OUTPUT PARANCSOK ÖSSZEFOGLALÁSA ANGOLUL

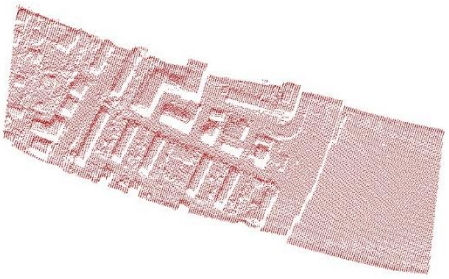
```
load                - Load workspace variables from disk, load filename
save                -  Save  workspace  variables  to  disk,  save  filename
variables
-ascii -append
```

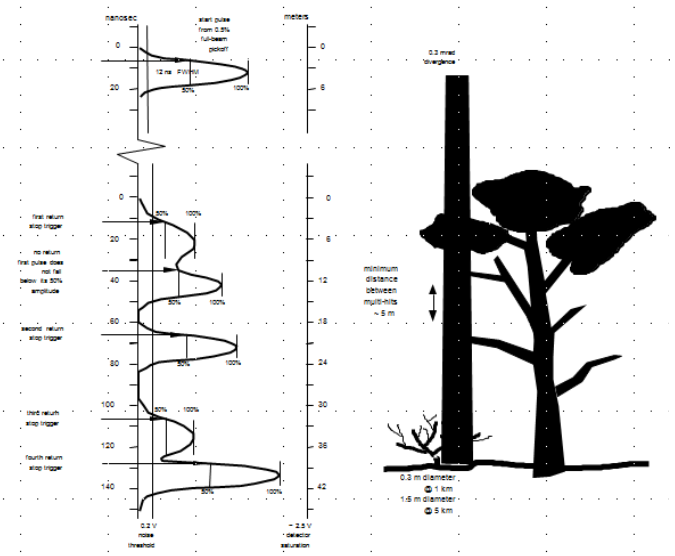
```
fopen             - Open file, or obtain information about open files,
fileID = fopen(filename)
fclose            - Close one or all open files, fclose(fileID)
fseek             - Move to specified position in file, fseek(fileID,
offset, origin)
feof              - Test for end-of-file, feof(fileID)
fgetl             - Read line from file, removing newline characters,
fgetl(fileID)
fgets             - Read line from file, keeping newline characters,
fgets(fileID)
fscanf            - Read formatted data from a text file, converts data into
array, fscanf(fileID, format)
sscanf            - Read formatted data from string, sscanf(str, format)
textscan          - Read formatted data from text file or string, returns a
cell array, textscan(fid, 'format')
fprintf           - Write data to text file, fprintf(fileID, format, A, ...)
sprintf           - Format data into string, sprintf(format, A, ...)
dlmread           - Read ASCII-delimited file of numeric data into matrix,
M = dlmread(filename, delimiter)
dlmwrite          -   Write   matrix   to   ASCII-delimited   file,
dlmwrite(filename, M, 'D')
fileread          -  Read  contents  of  file  into  string,  text  =
fileread(filename)
fread             - Read data from binary file
```

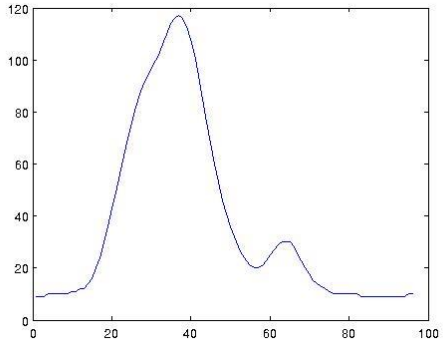
3. ANIMÁCIÓ KÉSZÍTÉS TELJES HULLÁMALAKOS LÉZERSZKENNER ADATOKBÓL

A teljes hullámalakos lézerszkennerek mérési eredményei, a lézerszkenner által kibocsátott és a felszínről visszaverődött jelek. A visszaérkező jelek nem azonos hosszúságúak, ezért nézzük meg először hogyan lehet különböző hosszúságú adatokat beolvasni!









Erdős területről visszaverődött jel

KÜLÖNBÖZŐ HOSSZÚSÁGÚ SOROK BEOLVASÁSA

Nézzünk most egy olyan példát, ahol különböző hosszúságú soraink vannak, pl. a rögzített mérések száma eltér egymástól. Ilyen lehet pl. egy légi lézerszkenner visszaverődéseinek rögzítése.

Legyen például a következő fájlunk:

meresek.dat

```
3 4 5 23 56 67 43 21 11
1 4 7 15 26 11
4 17 35 78 43 32 21
```

Hogyan tudjuk ezt beolvasni? Azt szeretnénk, hogy az eredmény egy tömbbe kerüljön, és a tömb üres elemei (az eltérő sorhosszak miatt) töltődjenek fel 0-val.

Érdemes soronként beolvasni az **fgetl**-lel, és utána egy ciklussal beleírni az elemeket a mátrix következő sorába. A hiányzó elemek automatikusan 0-k lesznek. Nézzük a megvalósítást. A fájl megadásakor válasszuk a meresek.dat fájlt!

```
> clear all; close all; clc;
> page_screen_output(0); % laponkénti megjelenítés leállítása Octave-
  ban
>
> % beolvasni kívánt fájl kiválasztása
> [inputfilename, inputpathname] = uigetfile('*.dat', 'Valaszd ki a
  beolvasando fajlt!');
> fajlnev = [inputpathname inputfilename]
> fid = fopen(fajlnev, 'r'); % fájl megnyitása olvasásra
```

Először olvassunk be egy sort az **fgetl** paranccsal, utána vizsgáljuk meg, hogyan tudnánk a sorban lévő számokat egy mátrixba beírni! Erre több lehetőségünk is van.

Először nézzük meg a szövegfeldolgozó parancsokat! Ezek között nagyon sok olyan van, ami hasznos lehet a számunkra, pl. a **strsplit** utasítás, ami szétdarabolja a szóközök (vagy opcionálisan megadott más határoló karakter mentén, például vessző - **strsplit(str,',')**) a szöveget, és a darabokat egy cellatömbbe teszi. Ilyenkor a cellatömb elemei szövegek, de ezt át lehet könnyen alakítani számmá a **str2double** paranccsal. Most a kimenet egy sorvektor lesz ami a számokat tartalmazza.

```
> a = strsplit(line)
> a = str2double(a)
```

Egy másik megoldás, ha használhatjuk a **sscanf** utasítást, ami ugyanaz, mint az **fscanf** (amit már használtunk korábban például az interferométeres adatok beolvasásakor), csak nem fájlból, hanem string típusú változóból olvas be formázott szöveget.

```
> line = fgetl(fid)
> a = sscanf(line,'%f')
```

A fenti parancsban a %f, azt jelenti, hogy lebegőpontos számokat (floating point number) olvas be a stringből, amíg talál ilyeneket. A kimenet egy oszlopvektor lesz ami a számokat tartalmazza. Ez tökéletesen megfelel a céljainkra. Meg lehet hívni két kimenettel is, ilyenkor a második kimenetbe az kerül, hogy hány darabot talált az adott formátumból. Hívjuk meg mi is így később még szükség lehet a darabszámra.

```
> [a n]=sscanf(line,'%f')
```

Bármelyik megoldást is használjuk a végeredmény egy (oszlop vagy sor) vektor lesz, ami tartalmazza az adott sorban lévő számokat. Miután több sort beolvastunk ezeket kéne összefűzni egy mátrixba. A gond csak az, hogy ezek a sorok nem egyenlő hosszúak, nem lehet egy [a1; a2] paranccsal összefűzni őket.

Hogyan tudunk eltérő hosszúságú sorokat mátrixba összefűzni? Hozzunk létre egy tetszőleges, de a-tól eltérő darabszámú b sorvektort, és egy M üres mátrixot. Az M

mátrixba fűzzük össze az a és b vektort a következő módon (az üres helyekre 0-k fognak kerülni):

```
> b = [1 2 3], M=[]
> M(1,1:length(a))=a
> M(2,1:length(b))=b
```

Itt a length parancsot használtuk, hogy megszámoljuk hány elem van a mátrixban, de ha korábban az sscanf parancsot használtuk, akkor az rögtön megadja a darabszámot is! Töröljünk (vagy kommenteljünk) ki mindent az **fopen** parancs után, és most próbáljuk meg egy ciklusban beolvasni M mátrixba a sorokat, 0-val feltöltve az üres helyeket.

```
> M=[];i=0; % M tömb letrehozasa, i - sorok száma
> while ~feof(fid)
>    line=fgetl(fid); % egy sor beolvasása
>    i=i+1; % sorok szamat noveljuk eggyel
>    [a n]=sscanf(line,'%f'); % a – számok, n - darabszám
>    M(i,1:n)=a;
> end
> fclose(fid);
> M
```

Eredmények:

```
M =
     3     4     5    23    56    67    43    21    11
     1     4     7    15    26    11     0     0     0
     4    17    35    78    43    32    21     0     0
```

TELJES HULLÁMALAKOS LÉZERSZKENNER MÉRÉSI ADATAINAK BEOLVASÁSA

Olvassuk be a waveform_sample állományt az előbb megírt programmal!

(Ez az állomány 1132 sorból áll, a maximális sorhossz 272)

Hogyan tudjuk ezt beolvasni? Azt szeretnénk, hogy az eredmény egy tömbbe kerüljön, és a tömb üres elemei (az eltérő sorhosszak miatt) töltődjenek fel 0-val. Érdemes soronként beolvasni az **fgetl**-lel, és utána egy ciklussal beleírni az elemeket a mátrix következő sorába. A hiányzó elemek automatikusan 0-k lesznek. Nézzük a megvalósítást. A fájl megadásakor válasszuk a meresek.dat fájlt!

```
> clear all; close all; clc;
> page_screen_output(0); % laponkénti megjelenítés leállítása Octave-
  ban
>
> % beolvasni kívánt fájl kiválasztása
> [inputfilename, inputpathname] = uigetfile('*.dat', 'Valaszd ki a
  beolvasando fajlt!');
> fajlnev = [inputpathname inputfilename]
> fid = fopen(fajlnev, 'r'); % fájl megnyitása olvasásra
>
> M=[];i=0; % M tömb letrehozasa, i - sorok száma
> while ~feof(fid)
>    line=fgetl(fid); % egy sor beolvasása
>    i=i+1; % sorok szamat noveljuk eggyel
>    [a n]=sscanf(line,'%f'); % a – számok, n - darabszám
>    M(i,1:n)=a;
```

```
> end
> fclose(fid);
> whos M
```

Eredmény:

```
  Attr Name        Size                     Bytes  Class
  ==== ====        ====                     =====  =====
       M        1136x272                   2471936  double
```

Jelenítsük meg grafikusan minden 50. mintát!

```
> figure(1);clf;
> for i=1:50:length(m)
>   plot(m(i,:));
>   pause(0.5);
> end
```

Csináljunk ebből egy animációt.

ANIMÁCIÓ KÉSZÍTÉS KÉPEKBŐL

Matlabban erre lehet használni az avifile és az addframe parancsokat, sajnos ezek Octave-ból nem elérhetőek egyelőre. Octave-ban a megoldás az lehet, ha elmentjük az animáció egyes képkockáit és egy külön programmal (pl. most az Imagemagick segítségével) animált gif-et készítünk belőle.

MEGOLDÁS OCTAVE-BAN

Először hozzunk létre az aktuális munkakönyvtárban egy mappát az **`mkdir`** paranccsal, amibe majd a képek fognak kerülni (ha már létezik a könyvtár, akkor nem csinál semmit). Töröljük ki az összes fájlt (*.*) a könyvtárból a **`delete`** paranccsal!

```
> mkdir('kepek'); % konyvtár letrehozasa a kepek szamara
> delete('kepek/*.*'); % benne levo korabbi kepek torlese
```

A video készítéséhez zárjuk be a megnyitott grafikus ablakokat és állítsuk át a grafikus megjelenítéshez használt eszközt **gnuplot**-ra Octave-ban az alapértelmezett '**fltk**' helyett, különben előfordulhat, hogy az előtt menti a képet, mielőtt teljesen kirajzolta volna, vagy akár ki is maradhat néhány kép.

```
> close all;
> % elerheto graphics_toolkit-ek:
> available_graphics_toolkits
> % az alap 'fltk' helyett gnuplot szukseges a kepek helyes mentesehez
> graphics_toolkit('gnuplot');
```

Plottoljuk ki a képeket, egyelőre csak minden100. beérkező hullámformát (waveform) úgy, hogy a tengelyek fixek legyenek – **`axis`** (animációban zavaró, ha mindig más a tengelyek léptéke). Az x tengely maximális értéke a leghosszabb sor darabszáma, vagyis az M mátrix oszlopainak száma. Az y tengely maximális értéke 255, mivel 0-255 között tárolják a visszaérkező hullámokat. Legyen az ábra címe, hogy éppen hányadik visszaérkező hullámnál tartunk (pl. 201. hullám). Generáljunk automatikusan fájlneveket is a következő alakban: *hullam0201.jpg*. A szám a végén a hullám sorszáma, 4 mezőbe kiírva, ha szükséges 0-kal feltöltve az elejét. Utána mentsük el a

képet a **print** paranccsal! Ha túl sok kép van, akkor esetleg a felbontást is csökkenthetjük az Octave-ban, ha megadunk a print-nél még egy méretre vonatkozó opciót, hogy hányszor hány pixel legyen a mentett kép '-S640,480' ez az opció Matlab-nál nincs.

```
> % Plottolás és a képek mentése
> figure(1);clf;
> oszlop = size(M,2);
> sor = size(M,1);
> for i=1:100:sor
>   plot(m(i,:));
>   axis([0 oszlop  0 255]);
>   title(sprintf('%d. visszaverodes', i));
>   filenev = sprintf('kepek/hullam%04d.jpg', i) % fájlnév generálása
>   % print(filenev, '-djpeg');
>   print(filenev, '-S640,480','-djpeg'); % kép mentése
>   % kép mentése adott felbontásba Octave-ban!
> end
```

Az animáció készítéséhez le kell tölteni egy parancssorból is futtatható egyszerű képszerkesztőt. Töltsük le az ImageMagick szoftvert a következő oldalról: https://www.imagemagick.org/script/index.php

A letöltésnél válasszuk a gépünknek megfelelő formát a download menü alatt, pl. Unix, Mac , Windows, ezen belül van 64 és 32 bites változat is, kinek milyen szoftvere van, pl. win64 static, vagy win32 static. A statikus változatot használjuk, amiben benne van minden dll is, ami kell. Telepítsük, lehetőleg olyan könyvtárba, aminek nincs a nevében szóköz pl a D:/ImageMagick könyvtárba.

Hívjuk meg Octave alatt a programot a **system** paranccsal (minden parancssori programot meghívhatunk így). A magick.exe programmal tudunk egy könyvtárban lévő fájlokból animált gif fájlt készíteni. meg kell adni egy delay-késleltetés opciót is, itt pl. ez 25, ami azt jelenti, hogy minden képet 25-ször vetít le, hogy lássuk az animációt.

```
> system('d:/ImageMagick/magick.exe -delay 25 kepek/*.jpg hullam.gif');
```

MEGOLDÁS MATLAB-BAN

Matlab-ban is használhatjuk a fenti megoldást, de ott van egy video készítő parancs is, azzal is megoldhatjuk az animációt:

```
> mozi = avifile('lidar_anim.avi','fps',4);
> fig1 = figure(1);
> % Ha adott felbontásban (pl. 640x480 szeretnénk menteni)
> set(fig1,'Units','pixel');
> pos = get(fig1,'Position')
> set(fig1,'Position',[pos(1) pos(2) 640 480]);
> k=1;
> oszlop = size(m,2);
> for i=1:20:length(m);
>    plot(m(i,:));
>    axis([0 oszlop  0 255]);
>    title(sprintf('%d. visszaverődés', i));
>    frame = getframe(fig1);
>    mozi = addframe(mozi, frame);
>
>    k=k+1;
>     pause(0.1);
> end
> mozi=close(mozi);
```

4. DXF FÁJL ÍRÁS, BEOLVASÁS, GUI ALAPOK

DXF VONALLÁNC ÍRÁSA KOORDINÁTÁKBÓL

Feladat egy szöveges állományban tárolt koordinátajegyzék beolvasása, majd ebből DXF vonallánc készítése, AutoCAD-be történő beolvasása.

Az xydata.txt állományban pontok koordinátái vannak eltárolva.

```
67.085 37.540
67.617 37.562
67.986 37.413
68.237 37.070
…
```

Ezek a pontok egy folytonos vonal pontjai. Olvassuk be őket MATLAB-ba, majd készítsünk belőle egy alap DXF fájlt, ahol a pontok vonallánccal vannak összekötve!

A feladat első része nagyon egyszerű, mivel a fájlban azonos típusú adatok vannak azonos sorhosszakkal, így a megoldás történhet az egyszerű `load` paranccsal, majd válogassuk szét őket x és y koordinátákra!

```
> page_screen_output(0); % csak Octave-ban kell
> clc; clear all;
> load 'xydata.txt'
> x = xydata(:,1)
> y = xydata(:,2)
> plot(x,y,'r+')
```

Ezután jön a DXF fájl készítése, amihez ismernünk kell a DXF fájlok szerkezetét (lásd DXF formátum segédlet a mellékletben).

Az adott fájlban egy darab folytonos vonal pontjai találhatóak. Ha ezt AutoCAD-ben szeretnénk megjeleníteni, akkor választhatjuk a vonal, vagy a vonallánc parancsot is. A vonal esetében két pontot kell megadnunk, kezdőpontot és végpontot. Egy folytonos vonal esetében ez azt jelentené, hogy a közbülső pontokat duplán kellene megadni, egyszer, mint az előző vonal végpontja, majd mint a következő vonal kezdőpontja. Ilyenkor a vonallánc célszerűbb, mert akkor elég egyszer megadni minden pontot, csomópontként.

Az AutoCAD DXF fájlokban a sorok kettesével összetartoznak, először egy kód van, hogy mi fog következni (pl. szekció eleje, vége, vonallánc típusú rajzi elem, rétegmegadás, szín stb.,, utána pedig a hozzá tartozó érték, vagy az érték kódja). A fájl első része teljesen általános, szekció eleje, rajzi elemek szekció. Meg kell adni a rajzi elem típusát is (vonallánc), hogy milyen rétegre kerüljön, utána megadhatjuk a színét (nem kötelező), végül következnek a csomópontok 66-os kóddal az elején, majd VERTEX megnevezéssel az egyes csomópontok. A fájl végén jön egy csomópontok vége jel, majd szekció vége, fájl vége.

Példa DXF vonalláncra (mindig két összetartozó érték van: következő művelet kódja, majd az értéke):

0	Új művelet
SECTION	Szekció kezdődik
2	Szekció típusa
ENTITIES	Rajzi elemek
0	Rajzi elem típusa
POLYLINE	Vonallánc
8	Rétegmegadás
vonal	'vonal' nevű réteg
62	Szín megadása
1	1-piros (2-sárga, 3-zöld, 4-v.kék, 5-s.kék, 6-lila, 7-fekete, 8-szürke)
66	Csomópontok következnek[3]
1	(értéke mindig 1)
0	Új művelet
VERTEX	Új csomópont
8	Rétegmegadás[4]
vonal	'vonal' nevű réteg
10	X koordináta (kezdőpont)
32.65	értéke
20	Y koordináta (kezdőpont)
67.89	értéke
…	…
0	Új művelet
SEQEND	Csomópontok megadásának vége
0	Új művelet
ENDSEC	Szekció vége
0	Új művelet
EOF	Fájl vége

Először nyissuk meg írásra a fájlt, amibe menteni szeretnénk a vonalláncot, majd írjunk a fájl elejére egy megjegyzést, hogy ez egy MATLAB-bal készült vonallánc. A DXF-ben a megjegyzés kódja a 999, az AutoCAD a 999 után következő sort nem olvassa be, azt megjegyzésnek tekinti. MATLAB-ban ezt az **`fprintf`** használatával írhatjuk ki a fájlba!

```
> fajlnev='xypolyline.dxf'; % Fájlnév megadása
> fid=fopen(fajlnev,'w'); % Fájl megnyitása írásra
> % Megjegyzes AutoCAD dxf fajlban: 999
> fprintf(fid,'999\r\nPolyline with matlab\r\n');
```

Be kell iktatnunk sorvége jeleket is a fájlba **`\r\n`**, itt vegyük figyelembe, hogy a sorvége karaktert különböző operációs rendszerek máshogy jelölik. A sor vége jel Windows esetében: \r\n, Mac (OS 9-) esetében \r, Unix/Linux esetében: \n. Az adott programtól is függ, hogy mit tud értelmezni sor végének. Sok olyan program van, ami mind a \n

[3] A 66-os kód jelöli, hogy csomópontok következnek, ez nem minden CAD programnál kötelező elem, de a tanszéki AutoCAD MAP 2008 változatnál igen.
[4] Lásd mint előbb. Nem minden CAD programnál kötelező minden csomópontnál megadni a réteg nevét is, de az AutoCAD MAP 2008 esetében igen.

mind a \r\n jelet elfogadja. Ilyen például a Wordpad, de ha Notepad-et használunk, akkor az csak a \r\n jelet tudja értelmezni. Ugyanígy van olyan CAD program, ami elfogadja mindkét jelölést, de van, ami csak a \r\n-t Windows alatt. Miután mindig két összetartozó érték van két különböző sorban, egyszerűsíthetjük a dolgot, ha készítünk egy anonim függvényt, ami a két érték után berak egy-egy sorvége jelet, hogy jobban áttekinthető legyen a programunk a későbbiekben.

```
> % Függvény összetartozó értékpárok két sorba írásához
> dxf = @(a,b) [a '\r\n' b '\r\n']
> dxf('10','20.124') % tesztelés:  10\r\n20.124\r\n
```

Ezután jöhetnek a tényleges rajzi elemek.

```
> fprintf(fid, [dxf('0','SECTION'), dxf('2','ENTITIES')]);
> fprintf(fid, [dxf('0','POLYLINE'), dxf('8','vonal')]);
> fprintf(fid, [dxf('62','1'), dxf('66','1')]);
> % A csomópontok adatai egyenként ciklusban
> for i=1:length(x)
>   fprintf(fid, [dxf('0','VERTEX'), dxf('8','vonal')]);
>   xs = num2str(x(i)); ys = num2str(y(i));
>   fprintf(fid, [dxf('10', xs), dxf('20', ys)]);
> end
> % lezárások
> fprintf(fid, [dxf('0','SEQEND'), dxf('0','ENDSEC'), dxf('0','EOF')]);
> fclose(fid); % Fájl lezárása
> type('xypolyline.dxf') % Nézzük meg mi van a fájlban
```

Ezután próbáljuk megnyitni valamilyen CAD-es programmal a fájlunkat (pl. AutoCAD, AutoCAD Map, BricsCAD). Ha minden jól megy és nem írtunk el semmit, akkor végre megnézhetjük, hogy mi is volt a fájlban! Méghozzá piros színnel, mivel a szín beállításánál 1-es kódot (piros) adtunk meg. Mentsük is el az állományunkat valamilyen más néven, de szintén ASCII DXF fájlként a programból! Nyissuk meg egy szövegszerkesztővel a két fájlt (pl. notepad++). Mit tapasztalunk? Az általunk létrehozott fájl kb. 300 sorból áll, a CAD-es szoftverből elmentett változat pedig több ezerből, pedig semmit nem változtattunk a rajzon!

ALAP PROGRAM EGYBEN

```
> page_screen_output(0); % csak Octave-ban kell
> clc; clear all;
>
> load 'xydata.txt'
> x = xydata(:,1)
> y = xydata(:,2)
> plot(x,y,'r+')
>
> fajlnev='xypolyline.dxf'; % Fájlnév megadása
> fid=fopen(fajlnev,'w'); % Fájl megnyitása írásra
> % Megjegyzes AutoCAD dxf fajlban: 999
> fprintf(fid,'999\r\nPolyline with matlab\r\n');
>
> % Függvény összetartozó értékpárok két sorba írásához
> dxf = @(a,b) [a '\r\n' b '\r\n']
> dxf('10','20.124') % tesztelés:  10\r\n20.124\r\n
>
> fprintf(fid, [dxf('0','SECTION'), dxf('2','ENTITIES')]);
```

```
> fprintf(fid, [dxf('0','POLYLINE'), dxf('8','vonal')]);
> fprintf(fid, [dxf('62','1'), dxf('66','1')]);
> % A csomópontok adatai egyenként ciklusban
> for i=1:length(x)
>   fprintf(fid, [dxf('0','VERTEX'), dxf('8','vonal')]);
>   xs = num2str(x(i)); ys = num2str(y(i));
>   fprintf(fid, [dxf('10', xs), dxf('20', ys)]);
> end
> % lezárások
> fprintf(fid, [dxf('0','SEQEND'), dxf('0','ENDSEC'), dxf('0','EOF')]);
> fclose(fid); % Fájl lezárása
```

INTERAKTÍV LEHETŐSÉGEK – GUI ALAPOK

Most egy kicsit tegyük interaktívabbá a programunkat néhány grafikus ablakkal (Graphical User Interface (GUI)), amikben a felhasználó adhat meg néhány opciót, például, hogy milyen színű legyen a vonallánc, milyen fájlból olvassuk be az adatokat, milyen néven és hová mentsük az elkészült állományt, és mi legyen az AuotCAD-es réteg neve, amire a rajz kerülni fog!

A grafikus felhasználói felület adatbeviteli parancsait a program elejére írjuk be, a **load** parancs elé!

A következőkben a felhasználó válassza ki a betölteni kívánt koordináta állományt az **uigetfile** paranccsal, majd ezt töltsük is be!

```
> % GUI - koordináta fájl választás
> [koordfajl koordmappa] = uigetfile('*.txt',...
>    'Valassza ki a beolvasni kivant koordinata fajlt!')
> koordfajl = [koordmappa koordfajl]
> xydata = load(koordfajl)
```

Az ezt követő **load** parancsot ezután ki is törölhetjük, vagy kommentelhetjük egy % jellel.

Korábban már használtuk a menu parancsot, ahol egy listából lehet kiválasztani elemeket, és a kiválasztott elem sorszáma lesz a visszaadott érték. Ezt célszerű használni a színek megadásához, ott ugyanis minden színnek van megfelelő kódja (1-piros, 2-sárga, 3-zöld, 4-világoskék, 5-sötétkék, 6-lila, 7-fekete, 8-szürke). Ha a **menu** parancsban ilyen sorrendben adjuk meg őket, akkor a megfelelő kódot kapjuk vissza. Ezt utána már csak szöveggé kell alakítanunk és később használni a vonallánc kiírásához.

```
> % GUI – szín kiválasztása menüből
> % színek kódjai: 1-piros ,2-sárga, 3-zöld, 4-v.kék, 5-s.kék, 6-lila,
> % 7-fekete, 8-szürke
> szin = menu('Adja meg a vonallanc szinet!','piros','sarga',...
>             'zold','vilagoskek','sotetkek','lila','fekete','szurke')
> szin = num2str(szin)
```

Utána kérdezzük meg a felhasználótól, hogy mi legyen a dxf fájl, és azon belül a réteg neve, amire a rajz kerülni fog. Ezt az **inputdlg** paranccsal tehetjük meg. Itt nem egy menüből kell kiválasztani a megfelelőt, hanem szöveges értéket/értékeket vár a program. Ha több dolgot szeretnénk megkérdezni, akkor egy kapcsos zárójelekből {'kerdes1','kerdes2'} álló cellatömbben kell megadni a kérdéseket. A kimenet

mindenképpen egy cellatömb lesz, aminek az elemeit szintén kapcsos zárójelekkel kérdezhetjük le! Ha csak egy értéket kérünk a felhasználótól, akkor a következőképpen tehetjük meg:

```
> nev = inputdlg('Fajlnev: ')
```

Ez a legegyszerűbb megadás, ilyenkor az ablak címe az alapértelmezett 'input dialog' lesz. Ha már kettő adatot kérdezünk a felhasználótól és címet is akarunk adni az ablaknak, akkor a következőképpen tehetjük meg!

```
> nevek = inputdlg({'Fajlnev: ','Milyen retegre keruljon a rajz? '},…
> 'Adja meg az adatokat!')
```

Ha azt szeretnénk, hogy legyen valamilyen alapértelmezett érték is rögtön beírva, akkor először meg kell adni, hogy hány sorba szeretnénk kiíratni az alapértelmezett értékeket (legyen ez most 1) és egy cellatömbben meg kell adni a kívánt szövegeket.

```
> % GUI - fájlnév, rétegnév megadása
> nevek = inputdlg({'Fajlnev: ','Milyen retegre keruljon a rajz? '},...
>         'Adja meg az adatokat!',1,{'proba1','reteg1'})
> fajlnev = [nevek{1} '.dxf']
> reteg = nevek{2}
```

Alapértelmezett esetben az új fájl az aktuális munkakönyvtárba kerül, amit a **pwd** paranccsal lehet lekérdezni. A felhasználó az **uigetdir** paranccsal választhat másik könyvtárat az aktuális könyvtár helyett, ahová menteni szeretné a DXF fájlt. A kiindulási könyvtár az aktuális munkakönyvtár legyen (pwd), és adjunk címet is az ablaknak! Ez utóbbi két adat opcionális, az uigetdir parancs önmagában is működik.

```
> % GUI - könyvtár kiválasztása, fájlnév pontosítása
> mappa = uigetdir(pwd,'Adja meg a konyvtarat, ahova menteni
  szeretne!');
> fajlnev = [mappa '\' fajlnev]
```

A felhasználó által történő fájlnév megadása után ki is törölhetjük/kommentelhetjük a korábbi **fajlnev='xypolyline.dxf';** parancsot.

Ahhoz, hogy a színt és rétegnevet is használni tudjuk, cseréljük ki az összes **dxf('8','vonal')** szöveget **dxf('8',reteg)**-re és a **dxf('62','1')** szöveget **dxf('62',szin)** szövegre!

INTERAKTÍV PROGRAM GUI HASZNÁLATÁVAL

```
> page_screen_output(0); % csak Octave-ban kell
> clc; clear all;
>
> % GUI - koordináta fájl választás
> [koordfajl koordmappa] = uigetfile('*.txt',...
>    'Valassza ki a beolvasni kivant koordinata fajlt!')
> koordfajl = [koordmappa koordfajl]
> xydata = load(koordfajl)
>
> % GUI - szín kiválasztása menüből
> % színek kódjai: 1-piros ,2-sárga, 3-zöld, 4-v.kék, 5-s.kék, 6-lila,
> % 7-fekete, 8-szürke
> szin = menu('Adja meg a vonallanc szinet!','piros','sarga',...
>             'zold','vilagoskek','sotetkek','lila','fekete','szurke')
> szin = num2str(szin)
```

```
> % GUI - fájlnév, rétegnév megadása
> nevek = inputdlg({'Fajlnev: ','Milyen retegre keruljon a rajz? '},...
>         'Adja meg az adatokat!',1,{'proba1','reteg1'})
> fajlnev = [nevek{1} '.dxf']
> reteg = nevek{2}
>
> % GUI - könyvtár kiválasztása, fájlnév pontosítása
> mappa = uigetdir(pwd,'Adja meg a konyvtarat, ahova menteni
  szeretne!');
> fajlnev = [mappa '\' fajlnev]
>
> x = xydata(:,1)
> y = xydata(:,2)
> plot(x,y,'r+')
>
> fid=fopen(fajlnev,'w'); % Fájl megnyitása írásra
> % Megjegyzes AutoCAD dxf fajlban: 999
> fprintf(fid,'999\r\nPolyline with matlab\r\n');
>
> % Függvény összetartozó értékpárok két sorba írásához
> dxf = @(a,b) [a '\r\n' b '\r\n']
> dxf('10','20.124') % tesztelés:  10\r\n20.124\r\n
>
> fprintf(fid, [dxf('0','SECTION'), dxf('2','ENTITIES')]);
> fprintf(fid, [dxf('0','POLYLINE'), dxf('8',reteg)]);
> fprintf(fid, [dxf('62',szin), dxf('66','1')]);
> % A csomópontok adatai egyenként ciklusban
> for i=1:length(x)
>   fprintf(fid, [dxf('0','VERTEX'), dxf('8',reteg)]);
>   xs = num2str(x(i)); ys = num2str(y(i));
>   fprintf(fid, [dxf('10', xs), dxf('20', ys)]);
> end
> % lezárások
> fprintf(fid, [dxf('0','SEQEND'), dxf('0','ENDSEC'), dxf('0','EOF')]);
> fclose(fid); % Fájl lezárása
```

DXF FÁJL BEOLVASÁSA, KOORDINÁTÁK KINYERÉSE

Nézzük meg röviden a feladat fordítottját, ha egy AutoCAD által dxf-be elmentett vonallláncból szeretnénk kinyerni a koordinátákat és egy szöveges állományba elmenteni! Ilyenkor a beolvasáshoz alaposan meg kell ismernünk az adott fájl szerkezetét. Többféle DXF verzió is van, így különböző programok eltérő szerkezetbe menthetnek. Nézzük meg a következő **satellite_polyline.dxf** fájlt! Ebben az AutoCAD által elmentett fájlban, a csomópontok szerkezete a következőképp néz ki:

```
  0
VERTEX
  5
21A
  8
0
 10
67.084536609003976
 20
37.540497719354967
 30
0.0
```

Itt a csomópontot jelző VERTEX után a 6. sorban van az x és a 8. sorban az y koordináta, amire nekünk szükségünk van.

Először be kell olvasnunk a fájlt. A textscan paranccsal beolvashatjuk az egész fájlt egy cella tömbbe (szoveg). Ez egy 1x1-es méretű cella tömb, aminek az első (és egyetlen eleme) egy 1551x1 méretű cella lesz (t), azaz minden sor (elem) egy cellába kerül.

Egy for ciklusban menjünk végig a cella elemeken és keressük meg a VERTEX elemeket (strcmpi – string összehasonlítás parancs), majd tároljuk le az őt követő 6. és 8. elemet az x és y változókba! Ezeket előbb számmá kell konvertálni a str2num paranccsal.

Ezután már csak el kell menteni a mátrixot a save paranccsal szöveges állományba. Ezt a következő programmal tehetjük meg.

```
> clear all; clc;
> page_screen_output(0); % laponkenti megjelenites leallitasa Octave-
  ban
>
> fid=fopen('satellite_polyline.dxf');
> szoveg=textscan(fid,'%s')
> fclose(fid);
> t=szoveg{1};
> % clear szoveg;
> x=[];y=[];
> for i=1:length(t)
>     s=t{i};
>    if strcmpi(s,'VERTEX') % s==VERTEX? kis-nagy betutol fuggetlenul
>        x1=str2num(t{i+6}); % A VERTEX utani 6. elem az x koordinata
>        y1=str2num(t{i+8}); % A VERTEX utani 8. elem az y koordinata
>        x=[x;x1];
>        y=[y;y1];
>    end
> end
> plot(x,y,'r+')
> xy=[x y]
> save xydata.txt xy -ascii
```

5. DOMBORZATMODELLEZÉS, INTERPOLÁCIÓ

Terepfelmérés során többnyire szórt pontokban kapunk magassági értékeket, melyekből szeretnénk digitális domborzatmodellt készíteni, szintvonalas ábrát vagy metszeteket rajzolni az éppen aktuális feladatnak megfelelően.

A digitális domborzatmodell készítése során a szórt pontokban történő felmérés eredményéből szeretnénk egy olyan modellt előállítani, amiből a felmért terület bármely pontjában kiszámolható a magasság értéke. Ez történhet például Delaunay háromszögeléssel és utána lineáris interpolációval, vagy akár spline interpolációval is. A domborzatmodellt többnyire rácshálóra interpolált értékekkel adjuk meg.

A már elkészült domborzatmodellből levezethetünk szintvonalas térképeket vagy terepmetszeteket is.

Töltse le a meres_coo.txt és a clabel2.m fájlokat és másolja be a munkakönyvtárba.

A terepfelmérés során mért koordinátáink a meres_coo.txt fájlban találhatóak a következő formátumban (pontszám, EOV koordináták (Y,X), Balti tengerszint feletti magasság):

```
2001  577057.011 188795.517 142.042
2002  577051.903 188783.028 140.821
2003  577051.044 188772.868 140.317
2004  577053.460 188758.714 139.622
2005  577053.097 188757.082 139.478
2006  577052.829 188752.585 139.186
2007  577043.018 188773.355 139.426
…
```

Miután azonos típusú, sorhosszúságú adataink vannak a beolvasás egyszerűen megtörténhet a load-dal.

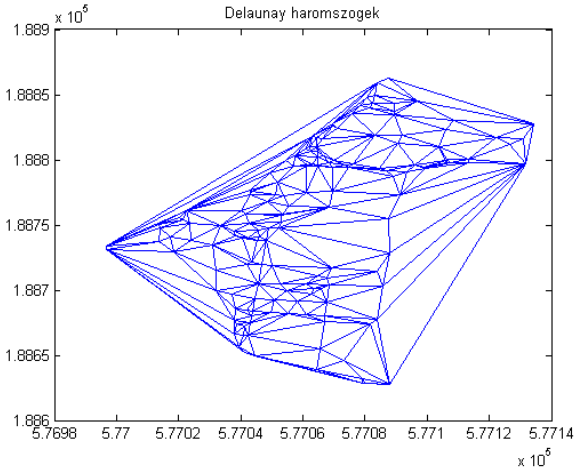
```
>  clear all; close all; clc;
>  page_screen_output(0);% laponkénti megjelenítés leállítása Octave-ban
>
>  data=load('meres_coo.txt');
>  x = data(:,2);
>  y = data(:,3);
>  z = data(:,4);
>  % Nézzük meg az adatok minimális, maximális értékeit
>  format long
>  xmin = min(x), xmax = max(x)
>  ymin = min(y), ymax = max(y)
>  zmin = min(z), zmax = max(z)
>
>  % Kiterjedés
>  xmax-xmin
>  ymax-ymin
>
```

LINEÁRIS INTERPOLÁCIÓ SZÓRT PONTOKBÓL RÁCSRA

A lineáris interpoláció szórt pontokra többnyire Delaunay háromszögelés alapján történik, ezekre a háromszögekre síkot illesztve ki lehet számolni minden pont magasságát.

Jelenítsük meg a háromszögeket!

```
> figure(1)
> % a csúcspontok indexei:
> tri = delaunay(x, y)
> triplot(tri, x, y);
> title('Delaunay haromszogek');
```



Interpoláljuk méterszer méteres rácshálóra lineárisan a mért értékeket a **griddata** parancs használatával! Ehhez először egy megfelelő rácshálót kell generálnunk a **meshgrid** használatával.

```
> % Interpoláció rácshálóra meshgrid és griddata használatával
> x_vekt=round(xmin):1:round(xmax); % egész méterenkénti x koordináták
> y_vekt=round(ymin):1:round(ymax); % egész méterenkénti y koordináták
> % kerekítési parancsok:
> % round-legközelebbi egészre, floor-lefelé, ceil-fölfelé kerekít
> [XI YI] = meshgrid(x_vekt, y_vekt); % rácsháló
> ZI=griddata(x, y, z, XI, YI,'linear'); % lineáris interpoláció rácsra
```

A **griddata** használatával rácsra interpolálhatunk. Lehetőségünk van legközelebbi szomszéd (`'nearest'`) vagy lineáris (`'linear'`) interpolációra. Ez utóbbi az alapértelmezett, ha nem adunk meg semmilyen interpolációs módszert. A Matlab-ban ezen kívül lehetőség van spline (`'cubic'`) interpolációra is, ez azonban sajnos még nincs implementálva Octave-ba. Spline interpolációhoz ezért a spline csomagot fogjuk most használni.

INTERPOLÁCIÓ RÁCSRÓL TETSZŐLEGES PONTRA

Az **interp2** paranccsal kérdezhetünk le magasság értékeket a rácshálóban előállított domborzat modellünkből. Ez pont a fordítottja a korábbi szórt pontokról rácsra történő interpolációnak, itt a rácsháló pontjait használjuk fel, hogy tetszőleges pontokban magasságot számoljunk. Kérdezzük le például az x1 = 577080.3 és y1 = 188750.3 pontban a magasságot!

```
> x1 = 577080.3; y1 = 188750.3;
> z1 = interp2(XI, YI, ZI, x1, y1)
```

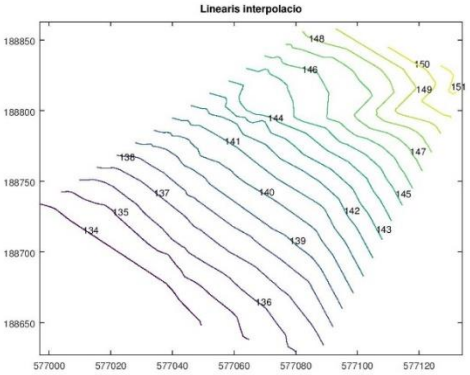
Az eredmény z1 = 141.424 m lesz.

Az interp2-nél is beállítható, hogy milyen módszerrel történjen a rácsról az adott pontra az interpoláció, lehet `'nearest'`, `'linear'` vagy `'cubic'`, ha nem adunk meg semmit, akkor lineáris az alapértelmezett.

DOMBORZAT MEGJELENÍTÉSI LEHETŐSÉGEK

SZINTVONALAS MEGJELENÍTÉS

```
> % Szintvonalak rajzolása a griddata eredményének felhasználásával
> % Figyelem: meglehetősen lassú a szintvonalak generálása!
>
>
> figure(2)
> a = ceil(zmin)
> b = floor(zmax)
> [C, h] = contour(XI, YI, ZI, a:b);
> % clabel(C, h, a:b);
> % töltsük le a clabel2.m fájlt!
> clabel2(C, a:b);
> title('Linearis interpolacio');
> % mentsük el képként!
> print('interp_lin.jpg','-djpeg')
```



Megjegyzés: a szintvonalak feliratozására a `clabel` parancsot is használhatjuk. Octave-ban azonban célszerű ennek egy módosított változatát használni (`clabel2`), mivel az eredeti bizonyos esetekben túl sűrűn feliratozza a szintvonalakat.

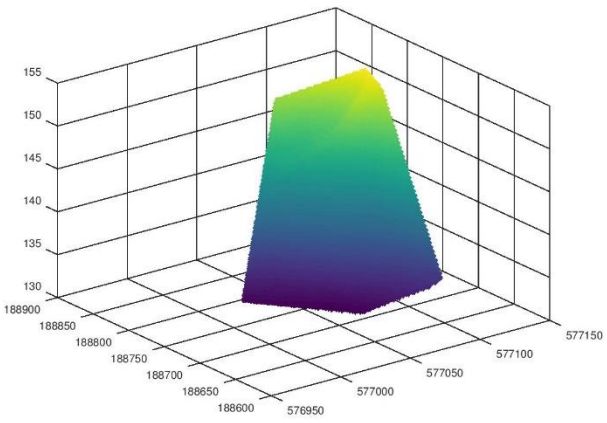
Vegyük észre, hogy extrapoláció nem történt az egész rácsra, csak a mért pontjainkat befoglaló konvex sokszögön belül vannak értékeink. (Ellenőrizzük le a ZI interpolált értékeket! Ahol nem volt adat oda NaN (not a number) értékek kerültek.)

FELÜLET ÁBRÁZOLÁS

Másik ábrázolási lehetőség a 3D-ben színezett felülettel történő megjelenítés, a `surf` paranccsal.

```
> figure(3);
> surf(XI,YI,ZI,'EdgeAlpha',0)
```

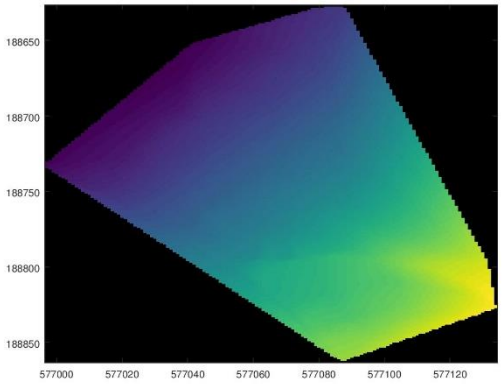
Itt az EdgeAlpha paraméter 0-ra állítása azért szükséges, hogy a zavaró felülethatárok ne rajzolódjanak ki.



MEGJELENÍTÉS SZÍNÁTMENETTEL

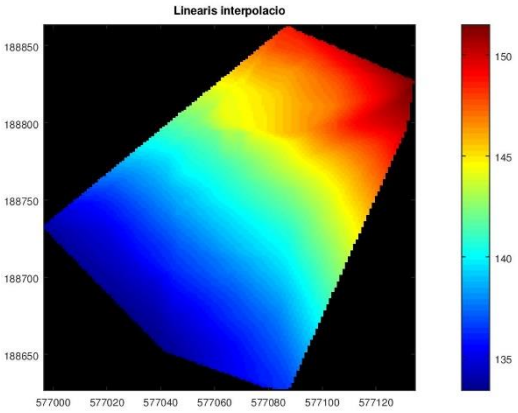
Szintvonalas megjelenítésen kívül magasság szerinti színátmenetes megjelenítéssel is ábrázolhatjuk a domborzatmodellünket. Ehhez használhatjuk az **imagesc** parancsot. Ez tulajdonképpen egy mátrixot jelenít meg képként.

```
> figure(4);
> imagesc(XI, YI, ZI);
> % imagesc(XI(1,:), YI(:,1), ZI);
```



Octave-ban a bemenő x,y,z koordináták is lehetnek a **meshgrid** által előállított mátrixos formában. Matlabban ki kell vegyünk a mátrixokból egy sort illetve egy oszlopot, mivel itt a bemenő x és y koordinátákat vektorban kell megadni, míg a z koordinátákat egy rácsháló pontjaiban.

Ha összehasonlítjuk az fenti képet a korábbiakkal, akkor feltűnik, hogy olyan a domborzatmodell, mintha függőleges irányban tükrözve lenne. Ez tényleg így van, ugyanis az imagesc koordináta rendszere a képek koordináta rendszerének felel meg (ij), azaz a bal felső sarok a kezdőpont, szemben a nálunk használt észak-keleti (xy) tájolású koordináta rendszerekkel. A kettő között az **axis xy** vagy **axis ij** paranccsal lehet váltani. A színskálát kitenni a **colorbar** paranccsal tudjuk, a színskálát változtatni pedig a **colormap** paranccsal. Próbáljunk ki különböző színskálákat! Alapértelmezett Ocatve alatt a **viridis**, Matlab alatt a **jet** skála. Egyéb pl.: **hot, hsv, summer, autumn, spring, winter, cool, gray, lines.**




```
> axis xy;
> colorbar;
> title('Linearis interpolacio');
> colormap(jet)
```

KÉPPONT KOORDINÁTÁINAK LEKÉRDEZÉSE GRAFIKUSAN

Használjuk most is az **interp2** parancsot, csak most grafikusan jelöljük ki azt a pontot, ahol szeretnénk lekérdezni a magasságot! A **ginput(n)** paranccsal a képre kattintva n pont koordinátáit kérdezhetjük le. Az **msgbox** paranccsal megjeleníthetünk egy üzenetet a felhasználónak egy grafikus ablakban. A **text** paranccsal szöveget írhatunk az ábra tetszőleges pontjára.

```
> msgbox('Kattintson a terkepen!')
> [xA yA] = ginput(1)
> zA = interp2(XI, YI, ZI, xA, yA)
> text(xA,yA,num2str(zA))
> msgbox(sprintf('A pontban a magassag: %.2f', zA))
```

SPLINE INTERPOLÁCIÓ (RÁCSRA)

A köbös spline interpoláció Matlabban megoldható a griddata egy másik beépített módszerével, a 'cubic' használatával, ezt azonban Octave alá még nem implementálták. Helyette használjuk a spline csomag **tpaps** parancsát (thin plate spline). Először nézzük meg, hogy telepítve van-e a csomag! Ha nincs telepítsük az internetről!

```
> pkg list
> % spline csomag telepítése internetről ha szükséges:
> % pkg install -forge splines
>
> pkg describe splines -verbose %kilistázza a splines csomag parancsait
> pkg load splines; % splines csomag betöltése
> ZI2 = tpaps([x y], z, 1, [XI(:) YI(:)]);
>
> % Ugyanez a Matlabban két paranccsal oldható meg:
> %   st = tpaps([x y]', z', 1);
> %   ZI2 = fnval(st, [XI(:) YI(:)]');
>
> ZI2 = reshape(ZI2, size(XI)); % visszaméretezés vektorból mátrixba
> figure(5);
> surf(XI, YI, ZI2) % extrapoláció is!
> ZI2(isnan(ZI)) = nan; % az extrapoláció kiküszöbölése
> surf(XI, YI, ZI2) % extrapoláció nélkül
```

A **tpaps** parancs ellentétben a **griddata**-val nem a **meshgrid** által mátrix formában előállított rácshálóban kéri az interpolálandó pontok helyét, hanem vektorosan összetartozó [xi yi] értékpárokat kér. Ezáltal azonban nem csak rácshálóra végezhető interpoláció, hanem bármilyen pontokra! A bemenetnél is összetartozó [x y] értékpárokra van szükség egy mátrixban, és a hozzájuk tartozó z értékekre egy külön vektorban. Utána megadható egy 0-1 közötti szám, ami a simítási tényező, ami a 'vékony lemez' merevségét jellemzi. 0 esetén teljesen merev, ekkor egy közelítő síkot illeszt a pontokra, 1 esetén pedig interpolál, azaz minden ponton átmegy a felület. A kettő között valamilyen regresszió van, minél közelebb van a szám 0-hoz, annál inkább a síkhoz hasonló sima felületet kapunk. (Megj: gyakorlatban azonban úgy tűnik nincs különbség, hogy 0-t vagy 1-et adunk meg, ugyanaz lesz a végeredmény , az Octave 4.0.1 változatán kipróbálva.)

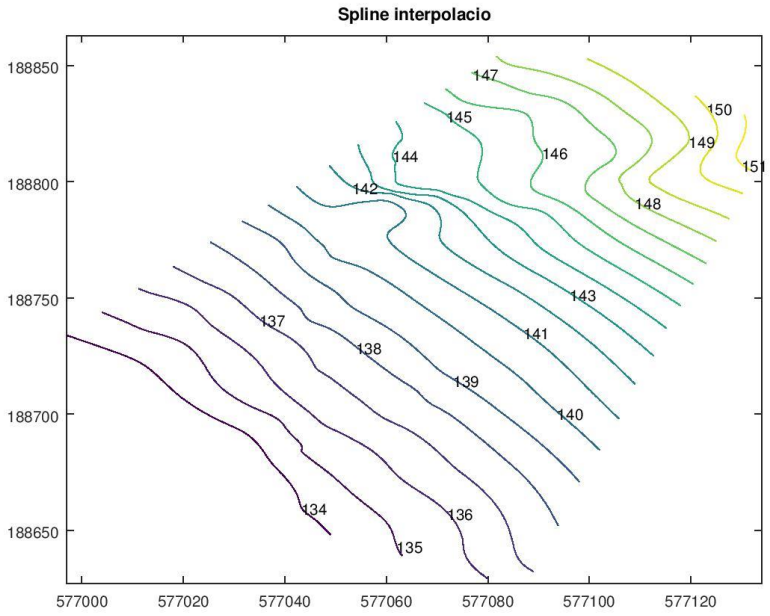
Octave-ban egy paranccsal hívható a módszer, Matlab esetén két parancs kell, először kiszámítjuk a spline együtthatóit, és utána az **fnval** paranccsal kiértékeljük az eredményt a rácspontokban.

A vektoros formában történő megadáshoz a meshgrid eredményeképpen kapott XI, YI értékekeket oszlopvetorrá alakítottuk az XI(:) és YI(:) parancsokkal.

Az eredményt a szintvonalas megjelenítéshez visszaalakítjuk mátrix alakba (**reshape**). Mivel ezzel a módszerrel extrapolálni is lehet, ahol viszont nagyon rossz eredményeink lennének, ezért az extrapolált helyeket NaN-nel töltjük fel (Not a Number). Ehhez felhasználtuk a lineáris interpoláció eredményét.

```
> % A spline interpoláció szintvonalas megjelenítése
> figure(6)
> [C, h] = contour(XI, YI, ZI2, a:b);
```

```
> clabel2(C, a:b);
> title('Spline interpolacio');
> % mentsük el képként!
> print('interp_spline.jpg','-djpeg')
```



Ez a módszer jóval simább szintvonalakat eredményez, mint a lineáris interpoláció.

Mentsük el a létrehozott domborzatmodelleket későbbi felhasználásra, a szórt pontokkal együtt!

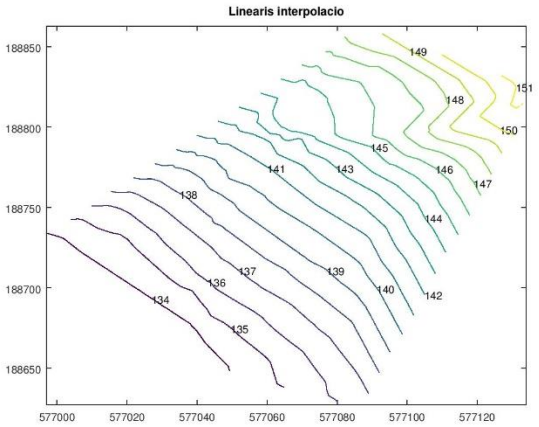
```
> save domborzat.mat XI YI ZI ZI2 x y z
```

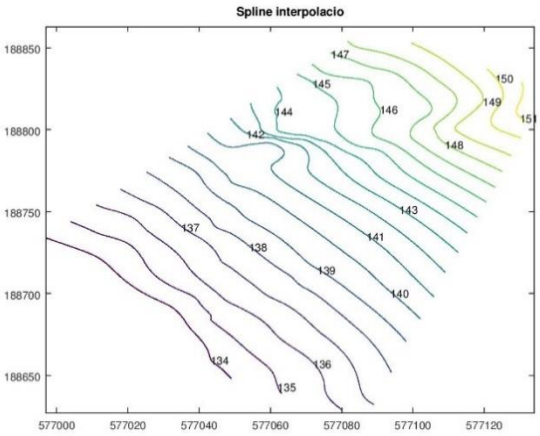
KÉTFÉLE INTERPOLÁCIÓ KÖZÖTTI ELTÉRÉSEK

A **domborzat.mat** fájlban x,y,z változók a terepfelmérés szórt pontjai, XI, YI az 1x1 méteres rácsháló pontjainak x,y koordinátái, ZI a lineáris interpoláció során kapott z értékek, ZI2 pedig a spline interpoláció során kapott z értékek.

Hasonlítsuk most össze a két megoldást!

A lineáris és spline interpolációval kapott szintvonalakat elmentettük az interp_lin.jpg és az interp_spline.jpg képekbe. A spline interpoláció jóval simább szintvonalakat eredményez, mint a lineáris interpoláció, azok kissé szögletesek. Viszont van egy terület, ahol a két ábra nagyon eltér egymástól. Vizsgáljuk ezt meg alaposabban!





ELTÉRÉS PONTOKBAN

Töltsük le a domborzat.mat fájlt, majd olvassuk be a tartalmát (XI, YI, ZI, ZI2, x,y,z)! Számoljuk ki a kétféle interpoláció között az eltéréseket. Jelenítsük meg színátmenetes térképen az eredményt! Néhány pontban kérdezzük le grafikusan az eltérés nagyságát, majd jelenítsük meg a maximális eltérés helyét kiemelve!

```
> clear all; close all; clc;
> page_screen_output(0);% laponkénti megjelenítés leállítása Octave-ban
> load domborzat.mat
>
> % Elteres a ketfele interpolacio kozott
> DZ = ZI - ZI2;
>
> % Megjelenítés színátmenetes térképen
> figure(1);
> imagesc(XI(1,:), YI(:,1), DZ); % színátmenetes térkép
> axis xy;
> colorbar; % jelkulcs
> title('Elteresek');
> colormap(jet)
>
```

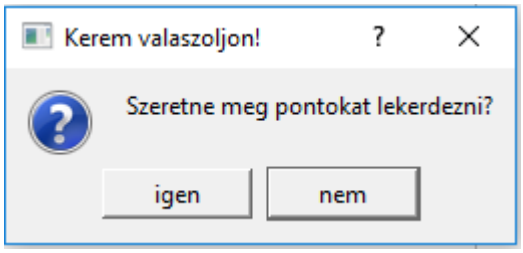
Egy felülnézeti ábrán a **ginput** paranccsal választhatunk ki egy pontot. Ezért használtuk megjelenítésre most a 3D **surf** parancs helyett az **imagesc** színátmenetes megjelenítést.

```
> msgbox('Adjon meg egy pontot a terkepre kattintva!')
> [xA yA] = ginput(1);
> % A pontban az eltérés értékét az interp2 paranccsal számíthatjuk
> zA = interp2(XI, YI, DZ, xA, yA)
```

Foglaljuk most ciklusba az egészet, és kérdezzük meg a felhasználótól szeretne-e még több pontot lekérdezni? A kérdés grafikus ablakban történő feltevéséhez többféle parancs közül választhatunk, használhatjuk az **inputdlg**, **menu** vagy a **questdlg** parancsot is. Most ez utóbbit használjuk. Gyakorlásképp próbálja ki a másik kettőt is, ami már korábban is szerepelt!

```
> kerdes = 'Szeretne meg pontokat lekerdezni?';
```

```
cim = 'Kerem valaszoljon!';
valasz = 'igen';
while strcmp(valasz,'igen')
  valasz = questdlg(kerdes, cim,'nem','igen','nem')
  if strcmp(valasz,'igen')
    [xA yA] = ginput(1)
    zA = interp2(XI, YI, DZ, xA, yA)
    text(xA,yA,num2str(zA))
  end
end
```

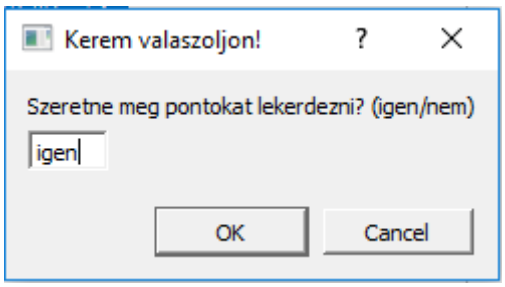


Az **questdlg** parancsnál meg lehet adni egy címet az ablaknak (pl. most: 'Kerem valaszoljon!'), két gombot és hozzá a végén, hogy melyik legyen az alapértelmezett, ha csak simán 'enter'-t nyomunk. A ciklus addig menjen, amíg a válasz igen. Mivel a ciklus elején történik a feltételvizsgálat, ezért, a kérdésfeltevés után is meg kell vizsgálni a feltételt, és csak akkor folytatni a lekérdezést, ha a válasz 'igen'. Másik megoldás lehet, ha azt vizsgáljuk, hogy 'nem' választ kaptunk-e, és ebben az esetben egy **break** utasítással is ki lehet lépni a ciklusból. A break utasítással így nézne ki a megoldás:

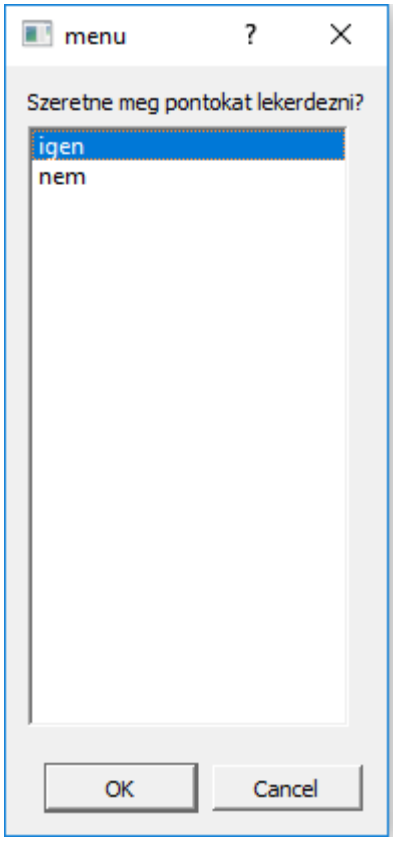
```
% Megoldás ciklussal, nem válasz esetén break használatával
valasz = 'igen';
while strcmp(valasz,'igen')
  valasz = questdlg(kerdes, cim,'nem','igen','nem')
  if strcmp(valasz,'nem')
    break
  end
    [xA yA] = ginput(1)
    zA = interp2(XI, YI, DZ, xA, yA)
    text(xA,yA,num2str(zA))
    msgbox(sprintf('A pontban az elteres: %.2f', zA));
end
```

Ismétlésként nézzük meg az inputdlg-s és a menu-s megoldások is!

```
% Megoldás inputdlg használatával
cim = 'Kerem valaszoljon!';
kerdes = {'Szeretne meg pontokat lekerdezni? (igen/nem)'};
def = {'igen'};
valasz = {'igen'};
while strcmp(valasz,'igen')
  valasz = inputdlg(kerdes,cim,[1,4],def)
  if ~strcmp(valasz,'igen')
    break
  end
    [xA yA] = ginput(1)
    zA = interp2(XI, YI, DZ, xA, yA)
    text(xA,yA,num2str(zA))
    msgbox(sprintf('A pontban az elteres: %.2f', zA));
end
```



Az **inputdlg** esetén szöveges visszajelzést adhatunk meg. Ebben az esetben a kérdéseket, válaszokat, alapértelmezett (default) értékeket cellatömbként kell megadni { } zárójelek között.



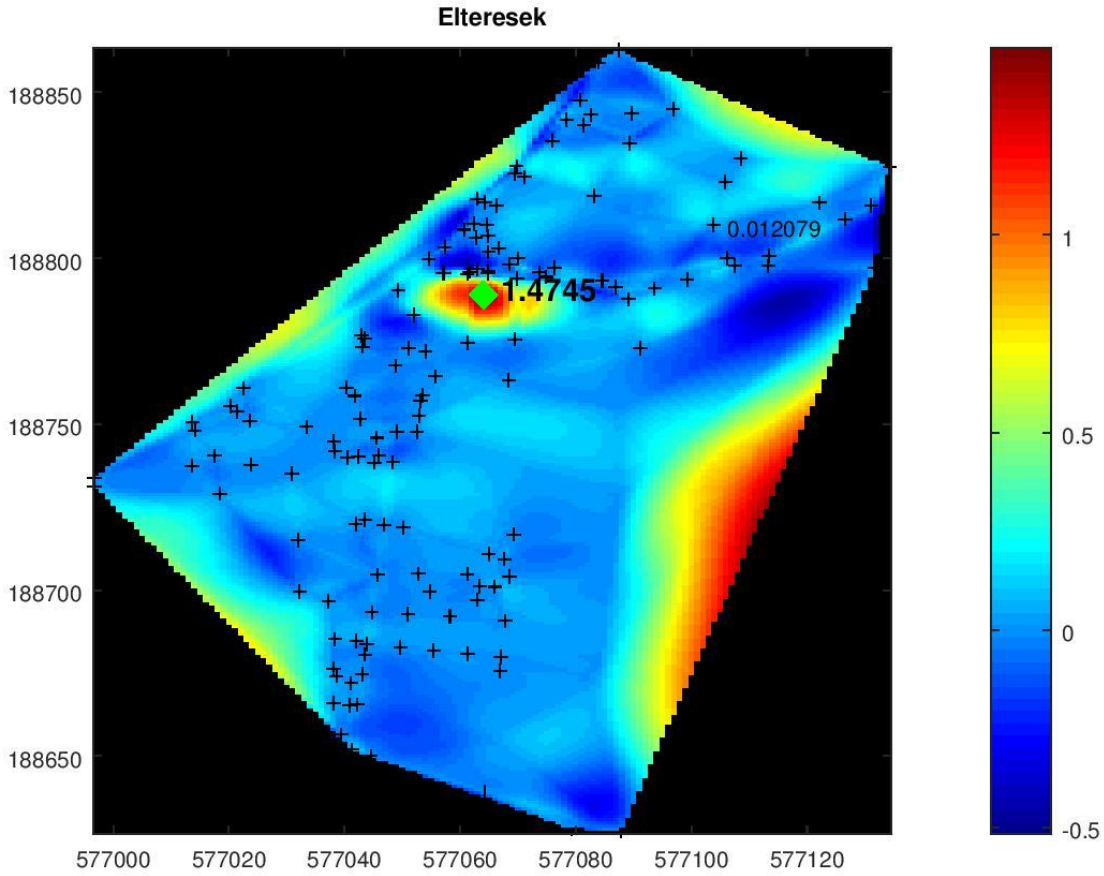

```
> % Megoldás menu használatával
> cim = 'Szeretne meg pontokat lekerdezni?';
> valasz = '1';
> while valasz
>   valasz = menu(cim, {'igen','nem'});
>   if valasz~=1
>     break
>   end
>     [xA yA] = ginput(1)
>     zA = interp2(XI, YI, DZ, xA, yA)
>     text(xA,yA,num2str(zA))
>     msgbox(sprintf('A pontban az elteres: %.2f',
zA));
> end
```

A **menu** parancs esetén egy legördülő menüből választhatunk, az első válasz esetén 1-es lesz az eredmény, a másodiknál 2-es, a harmadiknál 3-as stb.

MAXIMÁLIS ELTÉRÉS HELYE, ÉRTÉKE

Határozzuk meg a maximális eltérés értékét, helyét és jelenítsük meg!

```
> DZmax = max(max(abs(DZ))) % maximális eltérés a rácshálóban
> % a max. elteres helye a mátrixban (sor, oszlop)
> [imax jmax] = find(abs(DZ) == DZmax)
> % a max. elteres X, Y koordinátái a méterszer méteres rácshálóban
> xmax = XI(imax,jmax)
> ymax = YI(imax,jmax)
> % Kiplottolása nagyobb méretű zöld rombusszal, zöld kitöltéssel
> hold on;
> plot(xmax, ymax, 'gd', 'MarkerSize', 12, 'MarkerFaceColor', 'g');
> % Írjuk fel az eltérés értékét az ábrára!
> x0 = 3; y0 = 3; % eltolás értékek a szöveghez
> t = text(xmax+x0, ymax+y0, num2str(DZmax));
> set(t, 'fontSize', 14, 'FontWeight', 'bold');
> % Eredmény megjelenítése külön grafikus ablakban:
> msgbox(sprintf('A maximalis elteres: %.2f', DZmax))
> % Mérési pontok feltüntetése
> plot(x, y, 'k+'); % mérési pontok
```



Az eltérés ott a legnagyobb, ahol nincsenek mért pontjaink. Ezeken a helyeken a lineáris interpoláció egyszerűen levágja az értékeket egy egyenessel, a spline pedig a korábbi görbületekkel folytatva számítja a felületet. Ezt a legkönnyebben metszetek felvételével nézhetjük meg.

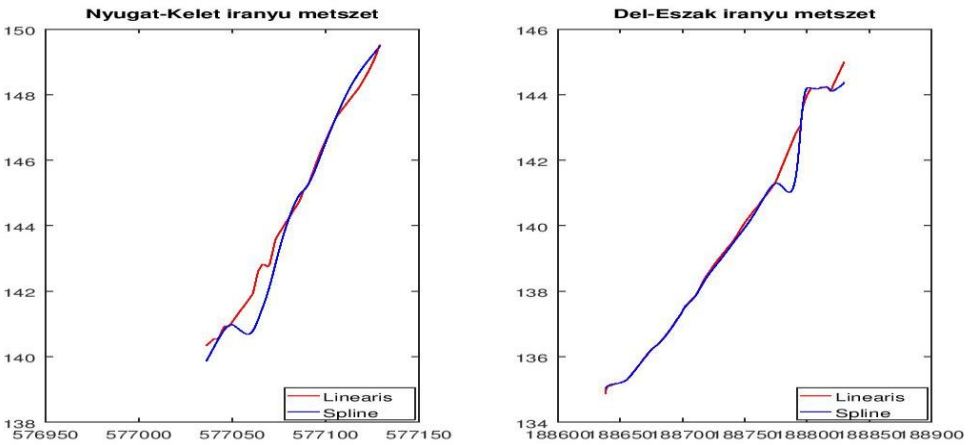
TEREPMETSZETEK KÉSZÍTÉSE

TEREPMETSZETEK É-D ÉS K-NY IRÁNYOKBAN

Készítsünk Észak-Dél és Kelet-Nyugat irányú metszeteket a maximális eltérés helyén! A metszetekhez kiválasztjuk a méterszer méteres rácshálónkból (XI, YI, ZI) azokat a sorokat (K-Ny) vagy oszlopokat (É-D), amelyek megfelelnek a maximális eltérésnek.

```
> % Nyugat-Kelet irányú metszet a maximális eltérés helyén
> figure(2);
> subplot(1,2,1)
> plot(XI(imax,:), ZI(imax,:), 'r'); hold on;
> plot(XI(imax,:), ZI2(imax,:), 'b')
> legend('Linearis', 'Spline','Location','southeast');
> title('Nyugat-Kelet iranyu metszet');
> xlabel('x');ylabel('z');
>
> % Észak-Dél irányu metszet a maximális eltérés helyén
> subplot(1,2,2)
> plot(YI(:,jmax), ZI(:,jmax), 'r');hold on;
```

```
> plot(YI(:,jmax), ZI2(:,jmax), 'b');
> legend('Linearis', 'Spline','Location','southeast');
> title('Del-Eszak iranyu metszet');
> xlabel('y');ylabel('z');
>
> % A szebb megjelenítés miatt nyújtsuk meg az ábrát x irányban!
> % Kérdezzük le az ábra alap pozícióját, méreteit
> % [pos_x, pos_y, size_x, size_y]
> pos = get(gcf, 'Position')
> % Nyújtsuk meg a x irányú méretet a kétszeresére!
> pos(3) = pos(3)*2
> set(gcf, 'Position', pos)
```



TEREPMETSZETEK TETSZŐLEGES IRÁNYBAN

A tetszőleges metszet felvételéhez két pontot kell kiválasztani a **`ginput`** paranccsal!

```
> % Metszet tetszőleges irányban
> pause(1); close(gcf); % bezárjuk az aktuális ábrát 1 mp után
> msgbox('Adjon meg ket pontot metszet keszitesehez!');
> figure(1);
> [xA yA] = ginput(2)
> hold on;
> plot(xA, yA, 'r*-','LineWidth',4);
```

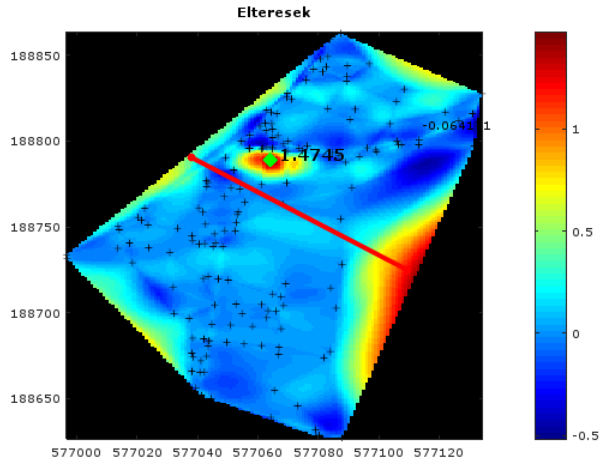
Ezután a két végpont között felveszünk 100 pontot, és ezekben kiszámoljuk interpolációval a z értékeket, akár a lineáris, akár a spline interpoláció eredményét felhasználva az **`interp2`** paranccsal.

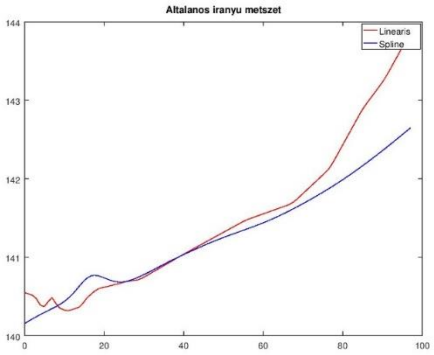
```
> xm = linspace(xA(1), xA(2), 100);
> ym = linspace(yA(1), yA(2), 100);
> zm = interp2(XI, YI, ZI, xm, ym);
> zm2 = interp2(XI, YI, ZI2, xm, ym);
```

A z értékeket a kezdőponttól mért távolság függvényében fogjuk megjeleníteni. Ehhez ki kell számolnunk a távolságokat!

```
> s = sqrt((xm-xA(1)).^2 + (ym-yA(1)).^2);
> figure(3);
> plot(s, zm, 'r'); hold on;
> plot(s, zm2, 'b');
```

```
> legend('Linearis', 'Spline'); title('Altalanos iranyu metszet');
```

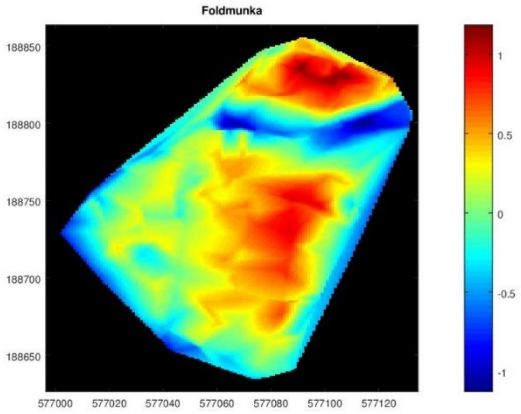




TEREPRENDEZÉS UTÁNI FÖLDTÉRFOGATOK SZÁMÍTÁSA

A területen elegyengették a kisebb nagyobb egyenetlenségeket, ezután készült egy újabb felmérés a területről. számoljuk ki ezek alapján a szükséges földmunka mennyiségét!

Először töltsük be a simított állományt!




```
> simitas = load('simitas_coo.txt');
> xs = simitas(:,2);
> ys = simitas(:,3);
> zs = simitas(:,4);
```

Interpoláljunk lineárisan rácshálóra, majd nézzük meg a különbséget a korábbi felületeinkkel!

```
> ZI3 = griddata(xs, ys, zs, XI, YI);
```

Számoljuk ki a szükséges földmunka mennyiségét, majd ábrázoljuk ezt! Ehhez számoljuk ki a két felület magasság különbségét a rácspontokban, rendezés előtt és után!

```
> foldmunka = ZI3 - ZI;
> figure(4);
> imagesc(XI(1,:), YI(:,1), foldmunka);
> axis xy;
> colorbar; colormap(jet);
> title('Foldmunka');
```

Számoljuk ki a szükséges töltés/bevágás mennyiségét! Mivel a rácsháló méterszer méteres így könnyű dolgunk van, csak összegezni kell a ’pixelek’ számát.

```
> toltes = sum(foldmunka(foldmunka > 0)) % 4054.9
> bevagas = sum(foldmunka(foldmunka < 0)) % -2011.9
> egyenleg = toltes+bevagas % 2043.0
```

6. SEGÉDLET A HÁZI FELADATOKHOZ

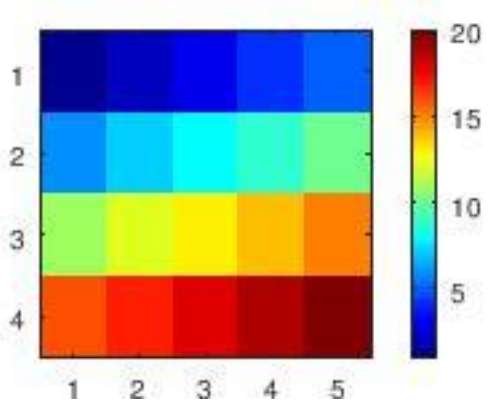
KÉPEK BETÖLTÉSE, MEGJELENÍTÉSE

IMAGESC – MÁTRIXOK/KÉPEK MEGJELENÍTÉSE

Többféle házi feladatban is előjön, hogy egy képet/mátrixot kell megjeleníteni, esetleg ezekből animációt készíteni. Legtöbbször az **imagesc** parancsot használjuk ezekre a megjelenítésekre, azonban néhány felmerülő problémával nem árt tisztában lenni.

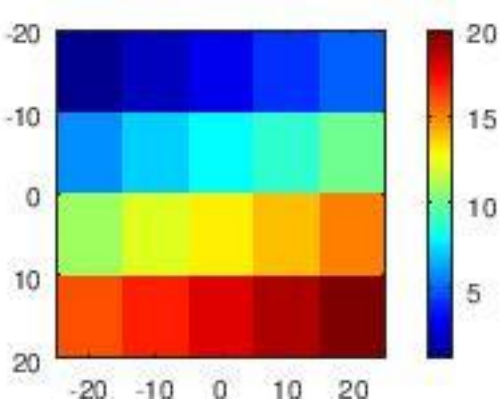
Az **imagesc** paranccsal akár mátrixokat, akár beolvasott képeket meg lehet jeleníteni (a képek szintén mátrixokként tárolódnak) színátmenetekkel. Hozzunk létre egy mátrixot, amiben számok vannak 1-től 20-ig, 4 sorban, 5 oszlopban, majd ezt jelenítsük meg színátmenetekkel!

```
> clear all; close all; clc;
> page_screen_output(0); % laponkénti megjelenítés leállítása Octave-ban
>
> M = [1 2 3 4 5;
>      6 7 8 9 10;
>      11 12 13 14 15;
>      16 17 18 19 20]
>
> figure(1)
> imagesc(M)
> colormap(jet); colorbar;
> % szépen megjeleníti a soroknak, oszlopoknak megfelelően
> print m01.jpg '-S200,150'
> % mentsük el a képet 200x150 pixel méretben
```



Szépen megjelenítette a mátrixot a sorok, oszlopok számának megfelelően a beállított színekkel. Kisebb számok kékkel, nagyobb számok pirossal. Mi történik azonban, ha ez a mátrix egy műholdfelvételnek felel meg, ahol egy pixel 10x10 fokot jelent, és az adataink (ha a pixelek középpontját nézzük) -20 és +20 földrajzi hosszúság illetve -15 és +15 földrajzi szélesség között helyezkednek el? Az **imagesc**-t többféle módon is meg lehet hívni, korábban a domborzatmodellezésnél **meshgrid**-del előállított rácshálóval használtuk, ahol a magasság adatokon kívül megadtuk az X, Y koordinátákat is. Ez működhet ennél a megoldásnál is, azonban az **imagesc**-nek van egy olyan meghívási módja, ahol megadhatjuk a minimális és maximális X, Y értékeket. Nézzük meg ezt.

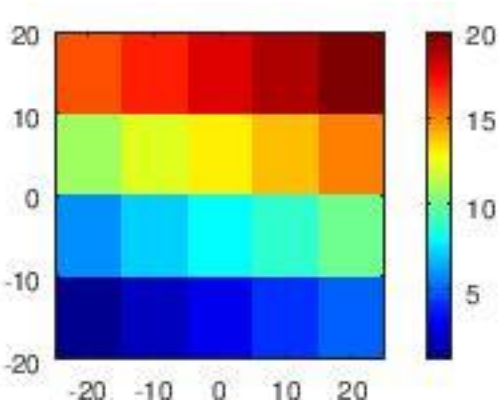
```
> % Az adatok [-20,20] és [-15,15] fok között 10 fokonként vannak.
> figure(2)
> imagesc([-20 20],[-15 15],M)
> colormap(jet); colorbar;
> % függőleges tengely számozása fentről lefelé nő, nem a földrajzi koord. szerint!
```



A függőleges (szélesség) koordináták számozása nem felel meg a földrajzi koordinátáknak, fentről lefelé nő, nem fordítva!

A korábbi módszernek megfelelően állítsuk át az alapértelmezett kép koordináta rendszert (**axis ij**) matematikai koordináta rendszerre (**axis xy**)!

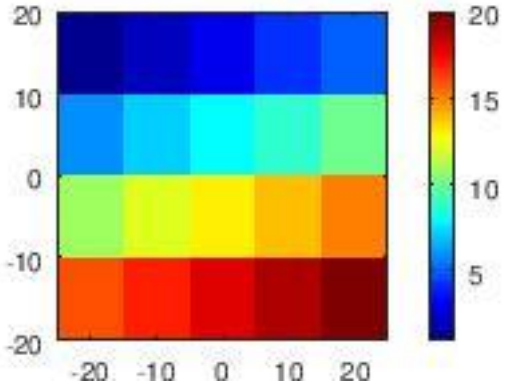
```
> figure(3)
> imagesc([-20 20],[-15 15],M)
> colormap(jet); colorbar;
> axis xy
> % az adatok is megfordultak!
```



Most viszont nem csak a koordináta rendszer, hanem az adatok is megfordultak! Hogyan lehetne ezt megoldani?

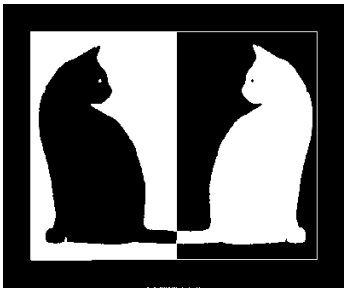
Megpróbálhatjuk pl. tükrözni függőlegesen az adatainkat (flipud), vagy létrehozhatunk meshgrid-del megfelelő rácsot és a szerint jelenítjük meg, vagy a legegyszerűbb, ha áttérünk az imagesc képkoordináta rendszer logikájára és a kép bal felső sarokpontból kiindulva adjuk meg a koordinátákat, magyarán a függőleges koordinátákat megfordítjuk és +15-től adjuk meg -15-ig! Nézzük meg mindegyik megoldást, a legegyszerűbbel kezdve!

```
> % megoldás 1: függőleges koordinátákat fordított sorrendben adjuk meg
> % a bal felső sarokból kiindulva, a kép koordináta rendszere szerint
> figure(4)
> imagesc([-20 20],[15 -15],M)
> colormap(jet); colorbar; axis xy
> % megoldás 2: tükrözzük az adatokat függőlegesen: flipud
> figure(5)
> imagesc([-20 20],[-15 15],flipud(M))
> colormap(jet); colorbar; axis xy
> % megoldás 3: rácsot adunk meg a meshgrid használatával
> x = -20:10:20
> y = 15:-10:-15
> [X Y] = meshgrid(x,y)
> figure(6)
> imagesc(X,Y,M)
> colormap(jet); colorbar; axis xy
>
```



KÉPEK BETÖLTÉSE

Az **imread** paranccsal lehet betölteni képeket, megjeleníteni pedig az **imshow** vagy a korábban is használt **imagesc** (színskálás megjelenítés) paranccsal.



```
> cicak = imread('cicak.gif');
> figure(1)
> imshow(cicak)
> figure(2)
> imagesc(cicak)
```

Mivel a cicak.gif egy fekete fehér kép, így egy egyszerű mátrixba kerül beolvasás után, amiben 0-k és 1-esek vannak (cicak: 333x400). Ennek egy tetszőleges részét ugyanúgy kivehetjük, mint egy mátrix egy részét.

```
> cicak1 = cicak(50:150,40:150);
> imshow(cicak1)
```

Színes képek egy 3 dimenziós mátrixba kerülnek beolvasásra (kolibri: 210x240x3).

```
> kolibri = imread('colibri.jpg');
> figure(1)
> imshow(kolibri)
```

INTERNETES FÁJLOK AUTOMATIKUS LETÖLTÉSE, KITÖMÖRÍTÉSE

FÁJLOK AUTOMATIZÁLHATÓ LETÖLTÉSE

A legtöbb házi feladatnál valamilyen ingyenesen elérhető térbeli adat feldolgozása, megjelenítése a cél. Ezeket az adatokat le is kell tölteni, sokszor ki is kell tömöríteni. Néhány adat esetében ez megoldható automatizálás nélkül is, azonban, ha sok adat van, esetleg interaktív a feladat és a felhasználó választhatja ki, hogy melyik az az adat, ami őt is érdekli, akkor ezt csak a programból meghívva tehetjük meg (**`urlwrite`**). A kitömörítéshez (akárcsak az animáció készítéséhez) itt is egy ingyenes parancssorból hívható külső programot fogunk használni, a most 7zip-et.

Nézzük meg, hogyan tudunk adatokat letölteni egy ftp szerverről!

Először meg kell adni az URL-t. Ha napi/havi/éves adatokat töltünk le, akkor ezeknek az adatoknak többnyire előállítható automatikusan (pl. sprintf-et használva) a neve valamilyen ismert szabályszerűség alapján. Először töltsünk le egy fájlt egy ftp szerverről. Nagyon sok adatot találhatunk az Astronomical Institute at the University of Bern (http://www.aiub.unibe.ch/index_eng.html) ftp szerverén (ftp.aiub.unibe.ch/)

Töltsük le először az AIUB adatait leíró txt fájlt Matlab/Octave alól:

ftp://ftp.aiub.unibe.ch/AIUB_AFTP.TXT

```
> fajlnev = 'AIUB_AFTP.TXT'
> url = ['ftp.aiub.unibe.ch/' fajlnev]
> [f, success] = urlwrite(url,fajlnev)
```

Itt a második kimenet azt adja meg, hogy sikeres volt-e a letöltés (ha igen, értéke 1). A parancs a megadott URL alól letölti és a megadott fájlnéven menti az állományt. Töltsünk le pl. GPS földi állomás koordinátákat adott napra!

Fájlnév szerkezete: CODyyddd.CRD.Z – yy=év, ddd=hányadik nap az évben.

Legyen ez a nap 2018. február. 1. (az év 32. napja), ekkor a fájlnév: COD18032.CRD.Z

(http://ftp.aiub.unibe.ch/BSWUSER52/STA/2018/COD18032.CRD.Z)

Állítsuk elő a fájlnevet **`sprintf`** segítségével, úgy hogy adott az év és nap az évben!

```
> ev = 2018; nap = 32;
> ev = num2str(ev);
> ev = ev(3:4)
> fajlnev = sprintf('COD%s%03d.CRD.Z',ev,nap)
```

```
> url = [' http://ftp.aiub.unibe.ch/BSWUSER52/STA/2018/' fajlnev]
> % fajlnev = COD18032.CRD.Z
> [f, success] = urlwrite(url,fajlnev)
```

Ez utóbbi fájlt még nem tudjuk megnyitni, beolvasni, mert be van tömörítve (*.Z a kiterjesztése).

FÁJLOK AUTOMATIKUS KITÖMÖRÍTÉSE

Fájlok kitömörítéséhez használhatjuk a 7zip program parancssori változatát. Töltsük le a

http://www.7-zip.org/download.html oldalról a 7-Zip Extra: standalone console verziót!

https://www.7-zip.org/a/7z1801-extra.7z

Tömörítsük ki és másoljuk be a **7za.exe** fájlt abba a könyvtárba, ahová dolgozunk, utána futtathatjuk Octave-ból. Az 'e' opció az extract – kitömörítés, a '-aos' kihagyja a már létező fájlokat. Több fájl együttes kitömörítése pl. *.Z beírásával történhet (adott könyvtárban minden Z kiterjesztésű fájl kitömörítése pl. system('7za.exe e *.Z').

Egy fájl kitömörítése pl. a COD18032.CRD.Z fájlé közvetlenül beírva:

```
> system('7za.exe e -aos COD18032.CRD.Z')
```

vagy változóból véve a fájlnevet:

```
> fajlnev = 'COD18032.CRD.Z'
> system(['7za.exe e -aos ',fajlnev])
```

DÁTUMBÓL ÉV NAPJA, GPS HÉT SZÁMÍTÁSA

Sok fájlnévben nem a dátum szerepel, hanem az, hogy az év hányadik napja. Ezt a **datenum** parancs használatával számolhatjuk ki.

DAYS = datenum (YEAR, MONTH, DAY)

A fenti parancs megszámolja a napokat 0. év. január 1-hez viszonyítva.

Egy adott évre vonatkoztatva a következőképp tehetjük meg:

```
> evnap = datenum(2018,03,27) - datenum(2018,1,0)
```

Kivonjuk belőle az adott év 0. napját (ami megfelel az előző év utolsó napjának), és megkapjuk, hogy éppen hányadik nap van.

Más esetekben a GPS hét van megadva, ezt a Geodetic Toolbox dates parancsának használatával kereshetjük meg a dátum alapján. A Geodetic Toolbox a Matlabcentral-on található kiegészítés, amit Mike Craymer írt:

https://www.mathworks.com/matlabcentral/fileexchange/15285-geodetic-toolbox/

A Matlabcentral (https://www.mathworks.com/matlabcentral/) oldalán, a file exchange menü alatt, sok hasznos letölthető programot találhatunk.

Érdemes lehet még földmérőknek az Octave Mapping csomagját is megnézni, ez feletehetően telepítve van, le lehet kérdezni a pkg list paranccsal, hogy mi van telepítev. Utána be lehet tölteni a pkl load paranccsal, és lekérdezni a benne lévő parancsokat a pkg describe paranccsal vagy megnézhetjük a honlapján is:

https://octave.sourceforge.io/mapping/overview.html

```
> pkg list
> pkg load mapping
> pkg describe mapping -verbose
```

LOGIKAI INDEXEK HASZNÁLATA

Legyen egy mátrixunk, amiben az év napjai és az azokon mért hőmérsékletek találhatóak.

```
> S = [2016 01 11 -2;
>      2016 01 21 3;
>      2016 03 30 21;
>      2017 02 12 -10;
>      2017 02 23 -5]
```

Kérdezzük le azoknak a soroknak az indexeit, amelyek 2016-ra vonatkoznak, vagyis, ahol az első oszlop 2016-tal egyenlő!

```
> S(:,1)==2016
```

Az eredménye egy oszlopvektor: [1; 1; 1; 0; 0]

Ezt használva indexnek, ki tudjuk válogatni az S mátrixból azokat a sorokat (és az összes oszlopot), ami 2016-ra vonatkozik!

```
> S_2016 = S(S(:,1)==2016,:)
S_2016 =
2016    1   11   -2
2016    1   21    3
2016    3   30   21
```

Az előzőhöz hasonlóan lekérdezhetjük a 2016 januári adatokat!

```
> S_2016_01 = S_2016(S_2016(:,2)==01,:)
S_2016_01 =
2016    1   11   -2
2016    1   21    3
```

FIX POZÍCIÓJÚ ELEMEKET TARTALMAZÓ SZÖVEG FELDOLGOZÁSA

Nézzük meg a **'nevsor.txt'** fájl tartalmát!

```
neptun nev                    szul       telepules      LAT      LON
abcdef Kiss Aladin Achillesz  2000/12/06 GYOR           +47.617 +017.783
bcdefg Nagy Aporka Szidonia   2001/03/15 SIOFOK         +46.917 +018.050
cdefgh Kovacs Antigon         2002/04/01 NYIREGYHAZA    +47.984 +021.692
defghi Halasz Augusztina      1998/05/01 DEBRECEN       +47.489 +021.615
efghij Szabo Arisztid         1997/10/23 KECSKEMET      +46.918 +019.749
```

Olvassuk be a **'nevsor.txt'** fájlt, ahol a különböző mezők megadott karakterek szerinti helyet foglalnak el. Mentsük el a változókat cellatömbbe. Jelenítsük meg azon diákok születési helyét egy térképen, akik 2000 előtt születtek. Ehhez jelenítsük meg Magyarország határvonalát is a **'hungary_border.dat'** fájlt használva.

```
> clc; clear all; close all;
> fid = fopen('nevsor.txt');
> % strsplit - szóközöknél szétszedi a sort
> for i=1:1; fgetl(fid); end; % fejléc
> neptun = {}; nev = {}; szul = {}; varos = {}; lat = {}; lon = {};
```

```
> i = 0;
> while ~feof(fid)
>   i=i+1;
>   line = fgetl(fid);
>   neptun0 = line(1:6);neptun{i,1} = neptun0;
>   nev0 = line(8:33); nev{i,1} = nev0;
>   szul0 = line(34:43); szul{i,1} = szul0;
>   varos0 = line(45:59); varos{i,1} = varos0;
>   lat0 = line(60:66); lat{i,1} = lat0;
>   lon0 = line(68:75); lon{i,1} = lon0;
> end
> fclose(fid);
> adat = [neptun, nev, szul, varos, lat, lon];
```

Most minden adat egy cellatömbbe van beolvasva:

```
adat = 5×6 cell array

{'abcdef'} {'Kiss Aladin Ach…'} {'2000/12/06'} {'GYOR       '} {'+47.617'}
{'+017.783'}
{'bcdefg'} {'Nagy Aporka Szi…'} {'2001/03/15'} {'SIOFOK     '} {'+46.917'}
{'+018.050'}
{'cdefgh'} {'Kovacs Antigon …'} {'2002/04/01'} {'NYIREGYHAZA'} {'+47.984'}
{'+021.692'}
{'defghi'} {'Halasz Auguszti…'} {'1998/05/01'} {'DEBRECEN   '} {'+47.489'}
{'+021.615'}
{'efghij'} {'Szabo Arisztid …'} {'1997/10/23'} {'KECSKEMET  '} {'+46.918'}
{'+019.749'}
```

Alakítsuk át a születési dátumot egy mátrixxá, az oszlopokban az év, hónap, nap adatokkal. Logikai indexeléssel válogassuk le a 2000 előtt születetteket!

```
> datumform = 'yyyy/mm/dd'
> szuldat = datevec(szul, datumform)
> % feltetel: ki született 2000 előtt? - logikai indexelés
> felt1 = szuldat(:,1)<2000;
> nev(felt1)
> adat(felt1,:)
```

Az eredmény:

```
ans =  2×6 cell array
{'defghi'} {'Halasz Auguszti…'} {'1998/05/01'} {'DEBRECEN '} {'+47.489'}
{'+021.615'}
{'efghij'} {'Szabo Arisztid …'} {'1997/10/23'} {'KECSKEMET'} {'+46.918'}
{'+019.749'}
```

Alakítsuk számokká a földrajzi szélesség, hosszúság adatokat. Rajzoljuk fel Magyarország határait és rajzoljuk be a diákok születési helyét, feliratozva a városok szöveggé alakított nevét is.

```
> % latitud, longitud számmá alakítása
> lat = cellfun(@str2num,lat)
> lon = cellfun(@str2num,lon)
> figure(1);
> plot(lon,lat,'+')
> hatar = load('hungary_border.dat');
> hold on; plot(hatar(:,1),hatar(:,2))
> text(lon,lat,char(varos))
```

MŰHOLD PÁLYA ADATOK FORMÁTUMA

Segítség: Műhold/űrállomás 2-soros pályaelemek:

http://www.celestrak.com/NORAD/elements/

24 órás pálya koordináták számítása 1 perces bontásban pályaelemekből:

http://www.satellite-calculations.com/TLETracker/SatTracker.htm

```
% Formatum:
%    [1,1] = 1
%    [1,2] =Mon
%    [1,3] = Apr
%    [1,4] = 10
%    [1,5] = 2017
%    [1,6] = 10:26:18
%    [1,7] = GMT+0200
%    [1,8] = 35:47:07.491
%    [1,9] = T=1.49105892222087
%    [1,10] = 132.495353
%    [1,11] = West
%    [1,12] = 4.55878583
%    [1,13] = North
```

IONOSZFÉRA, TROPOSZFÉRA ADATOK FORMÁTUMA

IONOSZFÉRA ADATOK

Ionoszféra térképek 2018-ra:

http://ftp.aiub.unibe.ch/CODE/IONO/2018/

CKMGddd0.yyI.Z file-ok (A letöltött fájlokat még ki kell tömöríteni beolvasás előtt. ddd- az év hányadik napja, yy-év utolsó két számjegye)

GPSGddd0.yyI.Z file-ok (A letöltött fájlokat még ki kell tömöríteni beolvasás előtt. ddd- az év hányadik napja, yy-év utolsó két számjegye)

Pl. 2018. január 4., az év 4. napja:

GPSG0040.18I.Z - 004 - day of year, 18 - year (2018)

2018. év 80. napja (2018. március 21.)

GPSG0800.18I.Z' -080 - day of year, 18 - year (2018)

Tartalma: Óránkénti ionoszféra térképek az adott napra, rácshálóban a TEC értékek, $-87.5° < \varphi < +87.5°$; $-180° < \gamma < +180°$ tartományban 2.5 illetve 5 fokonként.

Egymás után összesen 25 órányi adat, az adott nap 0 órájától a következő nap 0 órájáig. Minden órában meg vannak adva a fenti rácshálóra vonatkoztatva a TEC értékek. Érdemes ciklusban egymás után beolvastatni az adott órához tartozó rácsot(TEC térképet)!

TROPOSZFÉRA ADATOK

Troposzféra térképek 2018-ra: http://ftp.aiub.unibe.ch/CODE/2018/

CODwwwwn.tro.z file-ok (A letöltött fájlokat még ki kell tömöríteni beolvasás előtt. wwww- a GPS hét száma, n - a hét napja (0-6))

Tartalma: fejléc, majd állomás X, Y, Z koordináták, majd a kétóránként adatok állomásonként pl. *SITE ____EPOCH___ TROTOT STDDEV TGNTOT STDDEV TGETOT STDDEV

ABMF 16:051:03600 2560.6 0.5 -0.105 0.043 -0.338 0.045

ABMF 16:051:10800 2561.5 0.4 -0.144 0.038 -0.378 0.040…

Pl. epoch: 2016: 51. nap az évben: 3600. másodperc az adott napon (adatok 7200 másodpercenként, vagyis 2 óránként, 1 órától 23 óráig összesen 12 időpontban)

METEOROLÓGIAI ADATOK FORMÁTUMA

ÁLLOMÁSOK KOORDINÁTÁI

Állomások listája

ftp://ftp.ncdc.noaa.gov/pub/data/noaa/isd-history.txt

Minta:

```
128820 99999 DEBRECEN                       HU     LHDC  +47.489 +021.615 +0110.0 19310104
20170315
128920 99999 NYIREGYHAZA                    HU     LHNY  +47.984 +021.692 +0103.0 19730101
20170315
```

Mezők:

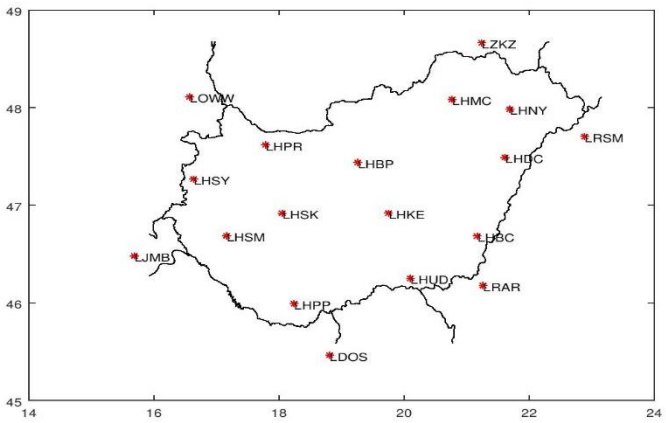
```
USAF = Air Force station ID. May contain a letter in the first position.
 WBAN = NCDC WBAN number
 CTRY = FIPS country ID
   ST = State for US stations
 ICAO = ICAO ID
  LAT = Latitude in thousandths of decimal degrees
  LON = Longitude in thousandths of decimal degrees
 ELEV = Elevation in meters
BEGIN = Beginning Period Of Record (YYYYMMDD). There may be reporting gaps within the P.O.R.
  END = Ending Period Of Record (YYYYMMDD). There may be reporting gaps within the P.O.R.
```

Az interpolációval előállított térképekhez célszerű használni néhány Magyarországon kívüli állomás adatát is, hogy az országhatár közelében is legyen adatunk, ne csak az ország belsejében. Pl. a következőket lehet használni:

```
LZKZ – Kosice (SK) - Kassa
LRSM – Satu Mare (RO) - Szatmárnémeti
LRAR – Arad (RO) - Arad
LJMB – Maribor (SI)
LOWW – Viena/Schwechat (AT) – Bécs
LDOS – Osijek (HR) - Eszék
```

Egyszerűsítésként Magyarországra sem szükséges az összes állomást használni, elég a következőket:

```
MISKOLC        - LHMC
SZOMBATHELY    - LHSY
GYOR           - LHPR
FERIHEGY       - LHBP
BALATON        - LHSM
DEBRECEN       - LHDC
NYIREGYHAZA    - LHNY
SIOFOK         - LHSK
PECS_SOUTH     - LHPP
KECSKEMET      - LHKE
SZEGED_(AUT)   - LHUD
BEKESCSABA     - LHBC
```



Az állományok adatai letölthetőek a www.agt.bme.hu/~piri/allomasok_mo.txt helyről.

ÓRÁNKÉNT HŐMÉRSÉKLETI ADATOK - ISD LITE FORMAT

Az időjárási adatok az amerikai NOAA (National Oceanic and Atmospheric Administration) oldaláról tölthetőek le, pl. a következő linken:

https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets

Óránkénti adatokat (NOAA Integrated Surface Database (ISD) Data Access): a fenti NOAA link Integrated Surface Database (ISD) menüpontja alól lehet elérni, vagy az alábbi link alól: https://www.ncdc.noaa.gov/isd/data-access . Ezen belül célszerű az ISD Lite FTP Access menüpontot választani, abban már egy javított, szűrt adatállomány van, amiben nincsenek ismétlődő adatok, és egész órákra tartalmazza a meteorológiai adatokat.

Ftp link: ftp://ftp.ncdc.noaa.gov/pub/data/noaa/isd-lite

Formátum: ftp://ftp.ncdc.noaa.gov/pub/data/noaa/isd-lite/isd-lite-format.txt

Évenként és állomásonként külön fájlban vannak tárolva az adatok. Egy fájl egy állomás összes adatát tartalmazza az adott évre, óránként bontásban!

Minta fájlnév: pl. **127720-99999-2017.gz**

Fájlnév: USAF kód - WBAN szám - évszám.gz (tömörített fájl!)

Mezők: évszám, hónap, nap, óra, hőmérséklet [ºC]*10, harmatpont [ºC]*10,légnyomás [HPa]*10, szélirány, szélsebesség [m/s]*10, felhőborítottság kód, óránként csapadék [mm]*10, 6 óra alatti csapadék [mm]*10

Figyeljenek a 10-es szorzókra! Egész számként vannak tárolva az adatok, ezért pl. a 10.5 fok 105-ként szerepel a fájlban!

Beolvasáskor érdemes megnézni a fájlformátum leírását (lásd fent), ott részletesen megvan, hogy egy sorban hányadik karaktertől hányadik karakterig mi található a fájlban.

Minta egy állomás adataira (egy fájl egy állomás egész éves óránkénti adatait tartalmazza!):

```
2017 01 01 10   -46   -60 10275   120    10     9 -9999 -9999
2017 01 01 11   -46   -60 10268     0 -9999     9 -9999 -9999
2017 01 01 12   -45   -59 10262     0 -9999     9 -9999     0
2017 01 01 13   -45   -59 10257     0 -9999     9 -9999 -9999
2017 01 01 14   -44   -56 10256    90    10     9 -9999 -9999
2017 01 01 15   -45   -59 10252   140    10     9 -9999 -9999
2017 01 01 16   -45   -59 10251     0 -9999     9 -9999 -9999
```

-9999 - nincs adat, Matlab-ban a -9999-et ki kell cserélni NaN-re (not a number)!

HAVI HŐMÉRSÉKLETI ADATOK

Vigyázat a hőmérsékleti adatok Fahrenheit-ben vannak megadva, nem Celsius fokban, a szélsebesség csomóban nem m/s vagy km/h-ban, a csapadék, hó mennyisége hüvelykben (inch), nem milliméterben. Át kell váltani őket! Az időjárási adatok az amerikai NOAA (National Oceanic and Atmospheric Administration) oldaláról tölthetőek le, pl. a következő linken:

https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets

Napi összefoglaló adatokat a fenti NOAA link Global Summary of the Day (GSOD) menüpontja alól lehet letölteni.

Ftp linkről: ftp://ftp.ncdc.noaa.gov/pub/data/gsod/

Formátum: ftp://ftp.ncdc.noaa.gov/pub/data/gsod/GSOD_DESC.txt

Évenként és állomásonként külön fájlban vannak tárolva az adatok. Egy fájl egy állomás összes adatát tartalmazza az adott évre, napi bontásban!

Minta fájlnév pl. **128390-99999-2016.op.gz**

Fájlnév: USAF kód - WBAN szám - évszám.op.gz (tömörített fájl!)

Ha egy mezőben csupa 9-es van az azt jelenti, hogy nincs adat. Matlab-ban ez az NaN (not a number)! Erre kell kicserélni ezeket!

Fontosabb mezők: STN – állomás azonosítója, YEARMODA – év- hónap-nap, TEMP – hőmérséklet [F], STP – az állomáson a közepes légnyomás (millibar-ban [mbar]), WDSP – közepes szélsebesség (knot (azaz csomó) mértékegységben), MXSPD – maximális szélsebesség, MAX, MIN – maximális!minimális hőmérséklet [F], PRCP – napi csapadék [inch], SNDP – hó mennyisége [inch]

Beolvasáskor érdemes megnézni a fájlformátum leírását (lásd fent), ott részletesen megvan, hogy egy sorban hányadik karaktertől hányadik karakterig mi található a fájlban. Minta egy fájlból:

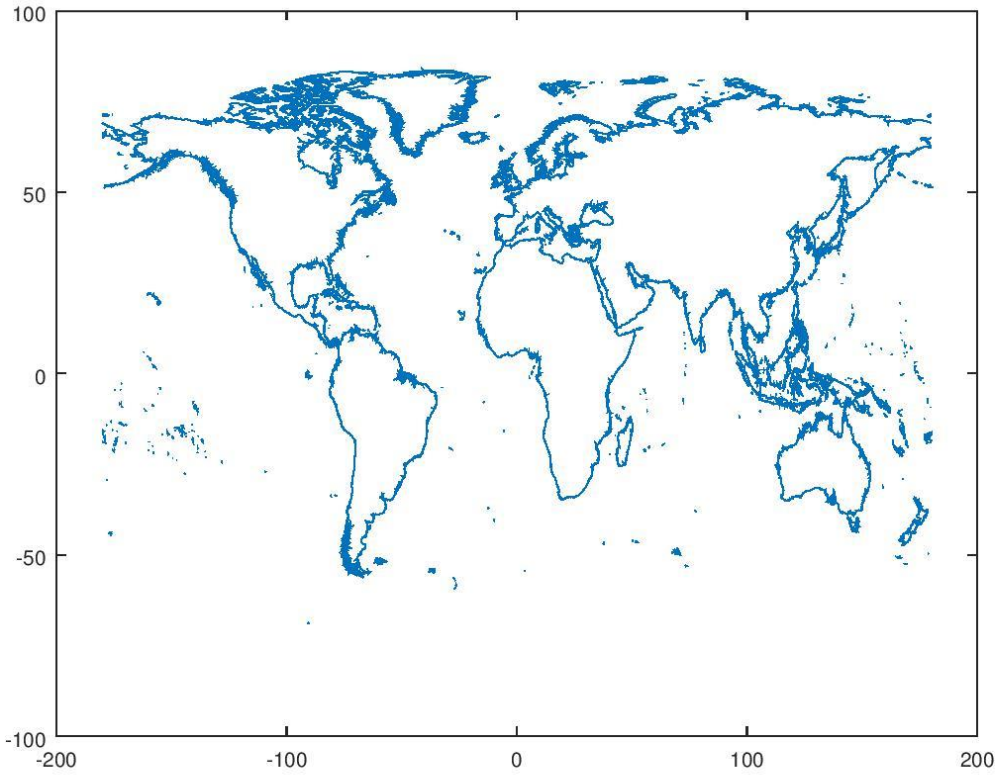
```
STN--- WBAN   YEARMODA    TEMP       DEWP      SLP        STP        VISIB      WDSP     MXSPD
GUST    MAX     MIN   PRCP   SNDP   FRSHTT

128390 99999  20160101    20.5 24    16.9 24  9999.9  0  9999.9  0    2.5 17    2.4 24    6.0
999.9    26.6*   10.4* 99.99  999.9  001000

128390 99999  20160102    20.3 24    16.9 24  9999.9  0  9999.9  0    3.0 22    6.8 24   12.0
999.9    23.0*   14.0* 99.99  999.9  001000

128390 99999  20160103    17.1 24     6.8 24  9999.9  0  9999.9  0  999.9  0    7.4 24   15.0
999.9    23.0*    8.6*  0.00I 999.9  000000
```

7. 3D ÁBRÁZOLÁS, NMEA-KML FORMÁTUMOK

FÖLDGÖMB ÁBRÁZOLÁS 3D-BEN

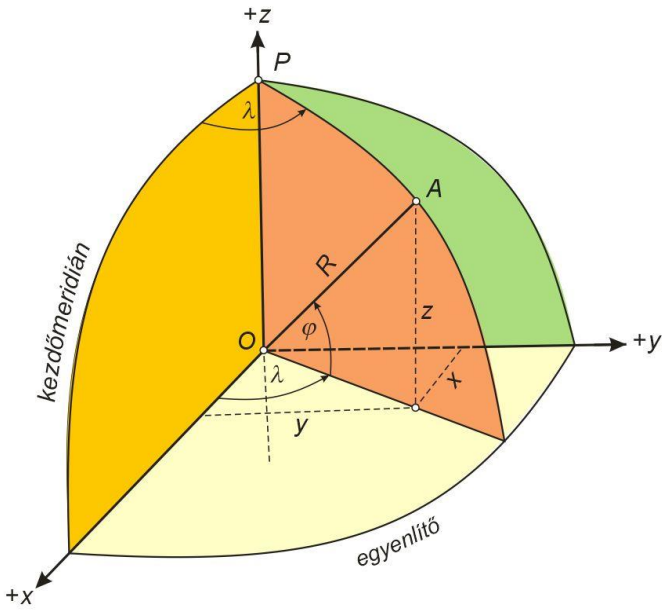
A coastline.txt fájlban a kontinensek partvonalai találhatóak. ábrázoljuk ezeket 2D-ben és 3D-ben is!

```
> clear all; close all; clc; page_screen_output(0)
>
> % partvonal koordinátáinak betöltése
> % síkbeli ábrázolás (fi, lambda alapján)
> a = load('coastline.txt');
> figure(1)
> plot(a(:,1),a(:,2))
```



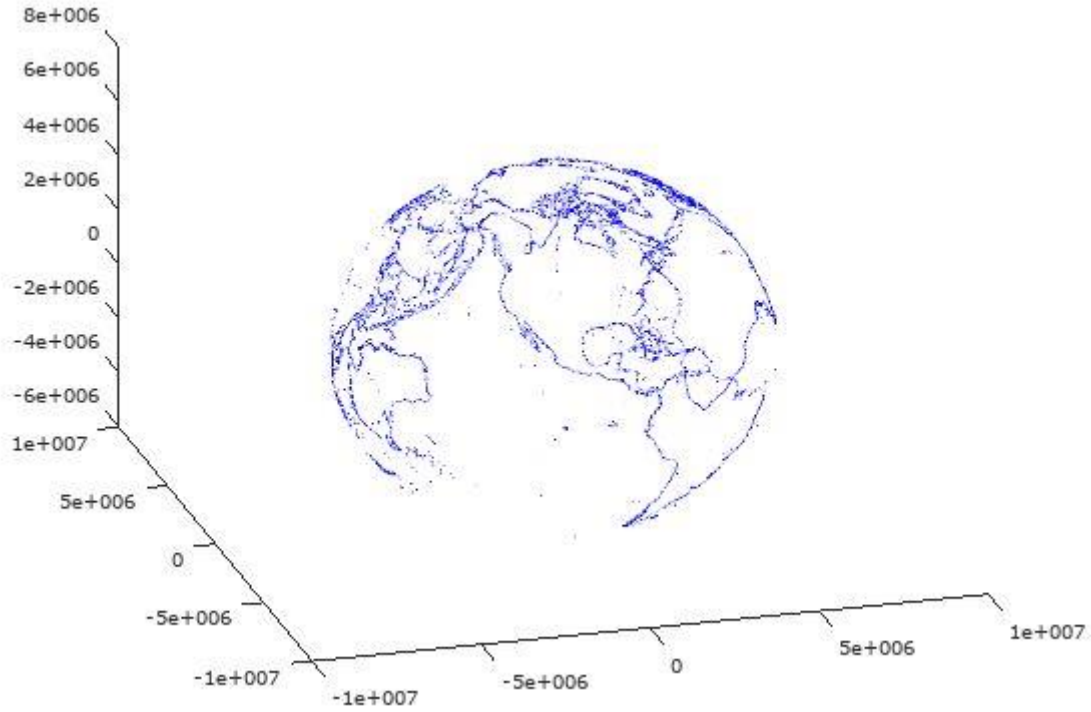

```
>
```

A térbeli koordináták számításához gömbi földrajzi koordináta rendszerből térjünk át térbeli derékszögű koordinátákra az alábbi ábra alapján:



$$x = R \cos\varphi \cos\lambda,$$
$$y = R \cos\varphi \sin\lambda$$
$$z = R \sin\varphi.$$

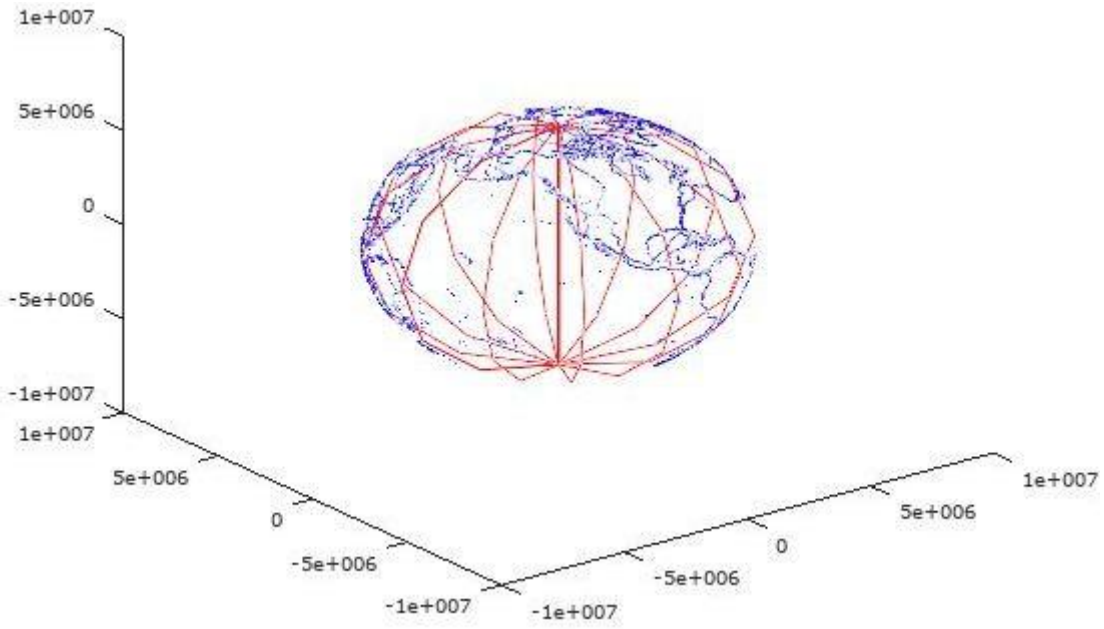
```
> % térbeli koordináták kiszámítása
> R = 6378000;
> z = R*sind(a(:,2));
> p = R*cosd(a(:,2)); % paralelkör sugara
> x = p.*cosd(a(:,1));
> y = p.*sind(a(:,1));
>
> % térbeli ábrázolás
> figure(2);
> plot3(x, y, z, 'b');
```



Ábrázoljuk a fokhálózatot is 30 fokonként! Először vegyük fel a földrajzi hosszúság (lambda) és földrajzi szélesség (fi) értékeit 30 fokonként. Utána állítsuk elő a meridián

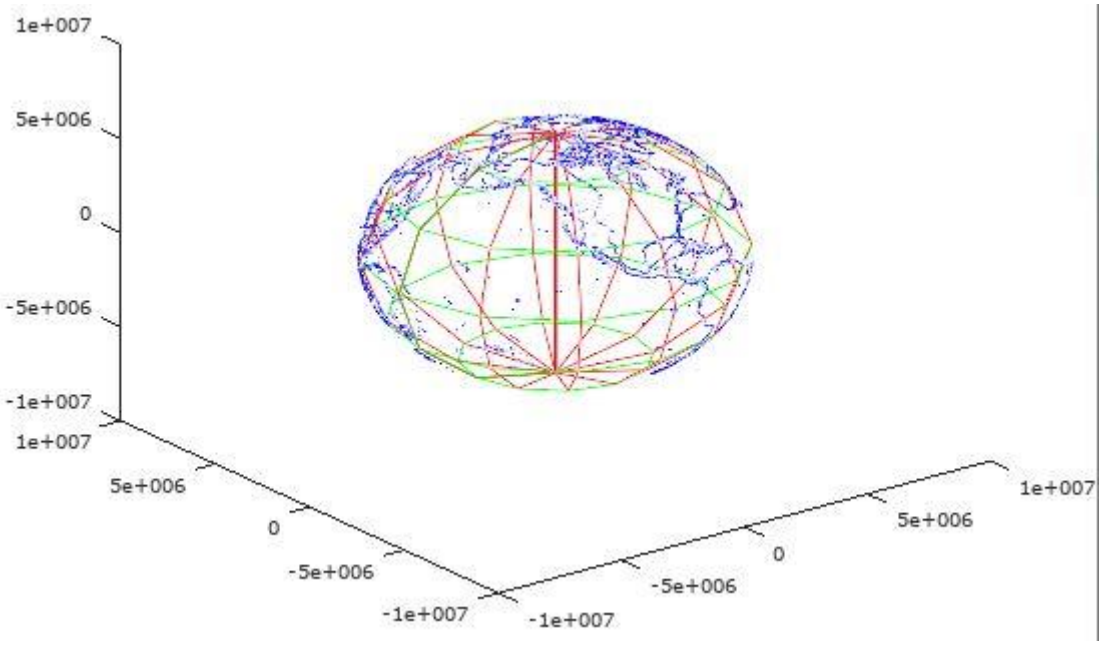
görbék pontjait egy vektorba. Ezeket számítsuk át az előző módon xyz térbeli koordinátákká és ábrázoljuk őket. ismételjük meg a műveletet a paralelkörökkel is! Figyeljünk, hogy egy adott meridiánnál a lambda állandó (ez kerül a külső ciklusba), míg egy adott paralelkörnél a fi lesz állandó!

```
> % fokhálózati vonalak felvétele 30 fokonként
> lambda = -180:30:180;
> fi = -90:30:90;
>
> % meridiánok görbéi (pontok 30 fokonként)
> meridian = [];
> for i=1:numel(lambda)
>   for j=1:numel(fi)
>     meridian = [meridian; lambda(i) fi(j)];
>   end
> end
> % meridiánok pontjai xyz koord. rendszerben
> zm = R*sind(meridian(:,2));
> pm = R*cosd(meridian(:,2));
> xm = pm.*cosd(meridian(:,1));
> ym = pm.*sind(meridian(:,1));
> % meridiánok ábrázolása 3D-ben
> figure(2); hold on;
> plot3(xm, ym, zm, 'r');
```




```
> % paralelkörök görbéi (pontok 30 fokonként)
> paralel = [];
> for i=1:numel(fi)
>   for j=1:numel(lambda)
>     paralel = [paralel; lambda(j) fi(i)];
>   end
> end
>
> % paralelkörök pontjai xyz koord. rendszerben
> zp = R*sind(paralel(:,2));
> pp = R*cosd(paralel(:,2));
> xp = pp.*cosd(paralel(:,1));
```

```
> yp = pp.*sind(paralel(:,1));
> % paralelkörök ábrázolása 3D-ben
> plot3(xp, yp, zp, 'g');
```



NMEA GPS ADATOK GOOGLE EARTH (KML) FORMÁBA ALAKÍTÁSA

Nyaralás alatt hajókázni mentünk és az útvonalat szeretnénk megmutatni az ismerőseinknek is. Az útvonalat GPS-szel rögzítettük, és az adatokat a navigációban gyakran használt NMEA 0183 formátumban kaptuk meg. Ezt szeretnénk megmutatni az ismerőseinknek. Először olvassuk be az adatokat, majd jelenítsük meg a saját korábbi ábránkban. Ezután alakítsuk át a Google Earth KML formátumába, hogy látványosabban tudjuk megjeleníteni az útvonalat.

NMEA ADATOK - RÖVID ISMERTETŐ

Most csak röviden összefoglaljuk az NMEA forméátumról, ami a mi feladatunkhoz kell, akit részletesebben érdekel a téma megnézheti a http://www.nmea.org oldalt, vagy magyar nyelven pl. Ferencz Viktória 2006-os TDK dolgozatát: http://vit.bme.hu/tdk/2006/dolgozatok/ferenczv.pdf. Az alábbi összefoglaló Primusz Pétertől (http://primuszpeter.blogspot.hu/2009/03/modern-navigacio.html) és Ferencz Viktóriától származik.

Az NMEA 0183 egy nemzetközi hajózási elektronikai szabvány, melyet az NMEA (National Marine Electronics Association) nevű nonprofit szervezet ad ki, és gondoz. Eredetileg különféle hajónavigációs eszközök (LORAN, radarok, OMEGA stb.) közötti kommunikációra kialakított szabvány, amelyet kibővítettek GPS-specifikus formátummal is. Napjainkra a GPS-vevőkből kommunikációs porton keresztül nyert információk leggyakrabban használt formája. Kedvelt, hiszen könnyen értelmezhető, rövid, szabadon sorrendezhető és összeállítható ASCII üzenetcsomagokról van szó. Minden küldött mondat $ karakterrel kezdődik, amelyet a küldő azonosítója követ, pl.: a GPS-t a GP karakterpár fog azonosítani, a hajónavigációs rendszert pedig az II azonosító jelöl.

Az NMEA 0183 szabvány szerint nagyon sok ún. "GPS adatmondat", parancs, lekérdezés létezik, a legáltalánosabb, a Globalsat összes GPS eszköze által használt parancsok jelentése a következő:

- GGA - Pozíció adat hibaértékkel, magasságadatokkal
- GSA - Aktív műholdak
- GSV - Látható műholdak
- RMC - Pozíció és UTC adatok
- GLL - Csak Pozíció adat (hosszúsági és szélességi)
- VTG - útirány és sebesség adatok

Az azonosító ismeretében tudható, hogy a következő (vesszőkkel elválasztott, azaz CSV formátumú) adatok éppen koordinátát, sebességet vagy valami mást jelentenek. A mondatok végén egy ellenőrzőösszeg és egy soremelés vagy sortörés karakter található.

Néhány példa 'GPS adatmondatokra' (minden ilyen mondat $GP-vel kezdődik):

Az egyik legfontosabb üzenet a 'GGA ' karakterhármast tartalmazó mondat, amely tartalmazza a 3D helymeghatározó adatokat és a pontosságra vonatkozó információt. Egy ilyen mondat a következőképpen értelmezhető:

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

GGA	GPS észlelési adatok
123519	észlelési időpont (12:35:19 UTC)
4807.038,N	földrajzi szélesség (48° 07.038',É)
01131.000,E	földrajzi hosszúság (11° 31.000', K)
1	Észlelés típusa (jóság):
	0 = érvénytelen, 1 = GPS észlelés (SPS) , 2 = DGPS észlelés, 3 = PPS észlelés,
	4 = RTK észlelés, 5 = Float (lebegő) RTK, 6 = értékelt (számítás nélkül), 7 = manuális bevitel, 8 = szimuláció
08	műholdak száma
0.9	HDOP
545.4,M	tengerszint feletti magasság [m]
46.9,M	geoid-ellipszoid (WGS-84) távolság
(üres karakter)	az utolsó DGPS frissítés óta eltelt idő [s]
(üres karakter)	DGPS állomás ID (azonosító)
*47	checksum adat (minden esetben * az első karakter)

A 'GLL' kódot tartalmazó mondat a szélességi és hosszúsági adatokra vonatkozó értékeket adja meg.

$GPGLL,4916.45,N,12311.12,W,225444,A,*31

GLL	geográfiai helyzet, földrajzi szélesség és hosszúság
4916.46,N	földrajzi szélesség (49° 16.45', É)
12311.12,W	földrajzi hosszúság (123° 11.12', NY)
225444	észlelés ideje (22:54:44 UTC)
A	adatok státusza (aktív vagy érvénytelen)
*31	checksum

NMEA ADATOK BEOLVASÁSA

Első feladatként olvassuk be az NMEA adatokat Octave-ba (vagy Matlabba)! Töltsük le a hb_nmea.txt fájlt az oktatas.epito.bme.hu oldalról.

Ebben a fájlban csak $GPGLL kezdetű sorok vannak. Pl.

$GPGLL,5156.9051,N,00117.1178,E*69

Egy sor maximum 80 karakter lehet. Meghatározott hosszúságú mezők vannak vesszővel elválasztva, elég könnyen beolvasható, feldolgozható fájl. 5156.9051,N = 51° 56.9051', É-i szélesség, 00117.1178,E = 1° 17.1178', K-i hosszúság. A földrajzi szélességnél az első két karakter a fok érték, utána perc, a hosszúságnál az első 3 karakter a fok érték, utána perc (mivel előbbi -90-+90, utóbbi -180-+180-ig terjed). Szélességnél N (Észak) pozitív, S (Dél) negatív, Hosszúságnál E (Kelet) pozitív, W (Nyugat) negatív.

Olvassuk be a meghatározott formátum szerint az fscanf paranccsal:

```
> clear all; clc; close all;
> page_screen_output(0);
>
> fid = fopen('hb_nmea.txt');
> mat = fscanf(fid,'$GPGLL,%2d%f,%1s,%3d%f,%1s*%*2s\r\n', [6 inf]);
> fclose(fid);
> mat = mat'
```

A beírt szöveget ($GPGLL,) átugorja, utána 2 karakteres egész szám (%2d), majd egy valós szám (%f), utána egy betű (%1s), majd 3 karakter egész szám (%3d), majd egy valós szám (%f), egy karakter (%1s), majd * és utána kihagy 2 karaktert (%*2s) a sorvége jel előtt. A % utáni értékeket elmenti a mat nevű tömbbe, 6 sorba és előre meg nem határozott számú oszlopba (amennyi adat van). Transzponáljuk a mátrixot, hogy az egy ponthoz tartozó adatok egy sorba kerüljenek egy oszlop helyett. Az fscanf csak

számokat képes tárolni egy tömbben, a textscan-nel ellentétben, ami cellatömbben többféle adatot is tárolhat, így az N/S, E/W (Észak/Dél, kelet/Nyugat) betűk is csak az ASCII kódjukkal lesznek eltárolva. Az ASCII kódokat lekérdezhetjük, pl. double('N') paranccsal. N = 78, S = 83, E = 69, W = 87. Fordítva pedig a char paranccsal kérdezhetjük le, hogy egy adott kódhoz milyen betű tartozik pl. char(78).

Válogassuk le a szélesség, hosszúság adatokat, alakítsuk tizedfokká, és ha déli szélesség (S) vagy nyugati (W) hosszúság szerepel benne, akkor szorozzuk meg (-1)-gyel az adott értéket. Ezt megoldhatjuk a szokásos módon 'for' ciklust és feltételt használva, de kihasználhatjuk hogy a Matlabot elsősorban mátrix/vektor műveletekre optimalizálták. Ez a 'vektorizálás' általában rövidebb, könnyebben áttekinthető és gyorsabb kódot eredményez. Nézzük meg a szélességeknél a hagyományos módon ciklussal, a hosszúságnál pedig vektorizációval.

```
> lat = mat(:,1) + mat(:,2)/60.0;
> for i=1:size(mat,1)
>   if mat(i,3)==double('S');
>     lat(i) = lat(i) * (-1);
>   end
> end
>
> long = mat(:,4) + mat(:,5)/60.0;
> % Ciklus nélkül megoldva ugyanez a hosszúságra vektorizációval
> % Megkeressük azokat az indexeket, amikre igaz a feltétel
> ii = mat(:,6) == double('W');
> % és az adott indexűeket megszorozzuk (-1)-gyel.
> long(ii) = long(ii) * (-1.0);
```

A formázott szövegként történő beolvasás helyett választhatjuk a soronkénti beolvasást is. Így a következőképp tudjuk előállítani a szélesség, hosszúság értékeket.

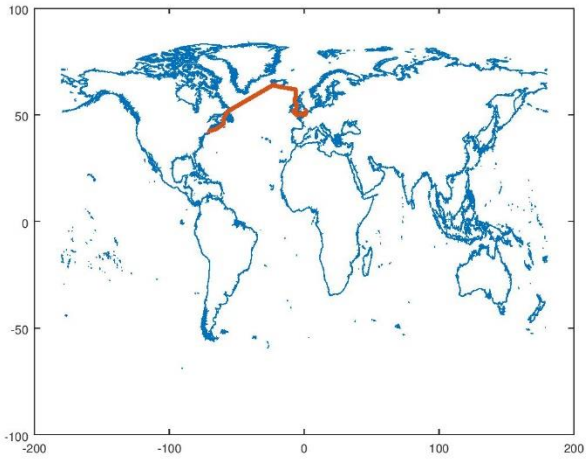
```
> %Adatok beolvasása méslépp, soronként, utólagos feldolgozással,
> % ha ismnerjük, hogy mi hányadik karakter
> fid = fopen('hb_nmea.txt');
>
> %Olvassuk be ciklussal az összes elemet,
> % és tároljuk a lat1, long1 vektorban a szélesség, hosszúság
  értékeket
> lat1 = []; long1 = [];
> while feof(fid)==0
>   line = fgetl(fid);
>   fi_fok = line(8:9);   fi_perc = line(10:16);
>   fi = str2num(fi_fok)+str2num(fi_perc)/60;
>   NS = line(18);   if NS=='S'; fi=fi*-1; end;
>   lambda_fok = line(20:22);  lambda_perc = line(23:29);
>   lambda = str2num(lambda_fok)+str2num(lambda_perc)/60;
>   EW = line(31);   if EW=='W'; lambda=lambda*-1; end;
>   lat1 = [lat1; fi]; long1 = [long1; lambda];
> end
> fclose(fid)
```

ÁBRÁZOLÁS

Ábrázoljuk az adatokat a korábbi 2D ábránkba és mentsük el az eredményt!

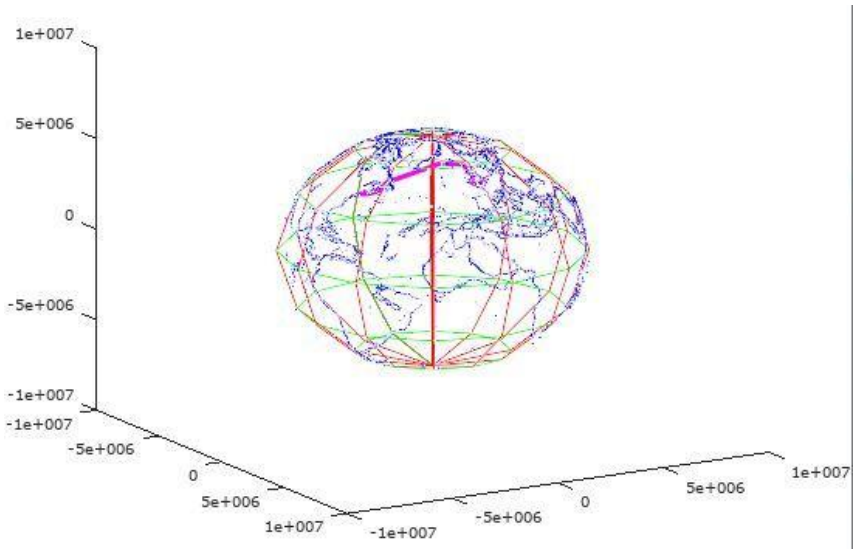
```
> figure(1); hold on;
```

```
> plot(long, lat,'LineWidth',3);
> save longlat.mat long lat
```



Rajzoljuk be a 3D ábránkba is!

```
> % útvonal pontjai xyz koord. rendszerben
> zu = R*sind(lat);
> pu = R*cosd(lat);
> xu = pu.*cosd(long);
> yu = pu.*sind(long);
> % paralelkörök ábrázolása 3D-ben
> figure(2);
> plot3(xu, yu, zu, 'm','LineWidth',3);
```



Lássuk be a fenti ábra nem igazán látványos, nehéz lenne ezzel dicsekedni a transzatlanti nyaralásunkkal. Jobb lenne, ha sokak által ismert és használt Google Earth-be tudnánk felrakni az adatainkat. A Google Earth formátuma a KML. Ismerkedjünk meg egy kicsit vele.

KML FORMÁTUM RÖVIDEN

A KML vagy Keyhole Markup Language egy földrajzi jellemzők, például pontok, vonalak, képek, sokszögek és megjelenítési modellek tárolására és modellezésére

szolgáló XML fájlformátum a Google Föld, a Google Térkép és egyéb alkalmazásokban. A KML segítségével helyeket és információkat oszthat meg ezeknek az alkalmazásoknak a többi felhasználójával.

Egy KML fájlt a Google Föld hasonlóan dolgoz fel ahhoz, ahogy a webböngészők feldolgozzák a HTML és XML fájlokat. HTML-hez hasonlóan a KML is névvel és attribútumokkal rendelkező jelölőket alkalmaz a meghatározott megjelenítési céljaira. Ily módon a Google Föld a KML fájlok böngészőjeként viselkedik. (lásd: https://support.google.com/earth/answer/148118?hl=hu)

Egy egyszerű mintapélda (Bodroginé Dr.Zichar Marianna: KML.pdf, https://w1.inf.unideb.hu/web/noir/gis, Josie Wernecke: The KML handbook alapján).

Megjegyzés sor: <!-- tetszőleges szöveg - ->

HelloFold.kml

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Helló Világ</name>
    <description>
    <![CDATA[
      <p><b>Itt fejlesztik a Google Earth-t!</b> </p>
    ]]>
    </description>
    <Point>
      <coordinates>
        -122.084583,37.42227,0
      </coordinates>
    </Point>
  </Placemark>
</kml>
```

SimpleLineString.kml

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Simple LineString</name>
    <Placemark>
      <name>Juan de Fuca Plate</name>
      <LineString>
        <coordinates>
          -130.597293,50.678292,0
          -129.733457,50.190606,0
          -130.509877,49.387208,0
          -128.801553,48.669761,0
```

```
          -129.156745,47.858658,0
          -128.717835,47.739997,0
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

ADATOK KML FORMÁTUMBA MENTÉSE

Látjuk, hogy a legegyszerűbb KML fájlok (pontok, egyszerű vonalláncok) nem túl bonyolultak, viszonylag könnyen elő lehet állítani akár Matlabot/Octave-ot használva is, ha ismerjük a fájl specifikációt. Ha már vonalstílust, pontstílust, egyebeket akarunk állítani, akkor már kicsit nehezebb, utána kell nézni alaposabban.

Viszont koordináta listából KML fájlt készíteni egy meglehetősen gyakran felmerülő feladat. Könnyen lehet, hogy vannak erre már kész matlab megoldások. Ilyenkor lehet érdemes esetleg körülnézni a neten (pl. a Matlab Central-on, ami egy Matlab felhasználói közösség: http://www.mathworks.com/matlabcentral/), ahelyett, hogy rögtön nekiállnánk megírni a saját matlab programunkat.

A Matlab Central oldalon válasszuk ki a File exchange menüt, majd keressünk rá a 'KML write' kifejezésre! Jó pár találatot kapunk, nézzük meg őket. Nekünk földrajzi koordinátákból kell egy vonalláncot készíteni. Használhatnánk akár a 'KML Toolbox'-ot Rafael Oliveirától, ami egy sokoldalú KML eszköztár, de a mi feladatunkhoz pont megfelelő a 'kml line plot' program Cameron Sparr-tól, ami lényegesen egyszerűbben kezelhető és pont azt a feladatot végzi, amire szükségünk van (Draw nan-separated lines (or a single line) onto Google Earth.). A Matlab Central oldaláról regisztráció után lehet letölteni fájlokat (érdemes esetleg a 'geodetic' kulcsszóra is rákeresni hasznos függvényekért!). Hogy most ne kelljen mindenkinek regisztrálni, az oktatas.epito.bme.hu oldalra feltöltöttem a kml_line.zip fájlt. Töltsük le és tömörítsük ki, majd nyissuk meg a kml_line.m fájlt.

Nézzük meg a specifikációt:

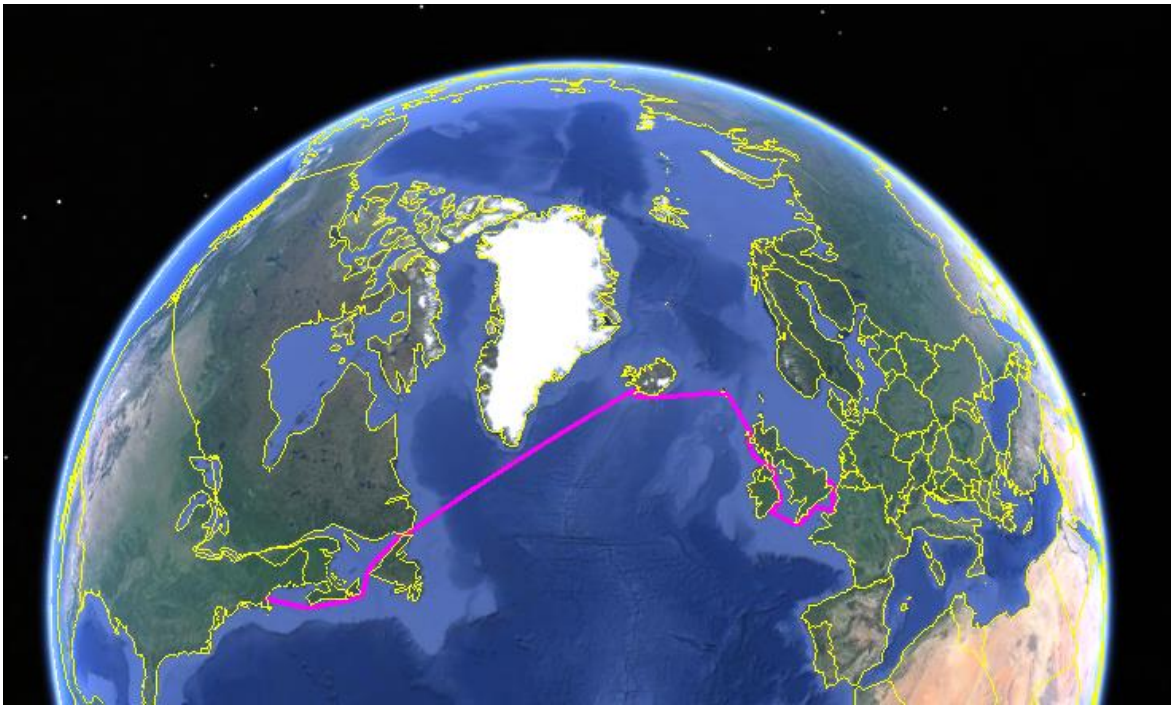
```
% KML_LINE    Draw nan-separated lines (or single line) onto Google Earth.
%
% Syntax:
%     KML_LINE(LON, LAT) writes nan-separated lines specified
%     in LON and LAT to an output file, doc.kml
%     KML_LINE(LON, LAT, NAME) writes nan-separated lines specified
%     in LON and LAT to an output file, NAME.kml
%     KML_LINE(LON, LAT, NAME, COLOR) writes lines same as above but with
%     MATLAB color value (default is 'w').
%     KML_LINE(LON, LAT, NAME, WIDTH) writes lines same as above but with
%     width WIDTH (default is 1).
%     KML_LINE(LON, LAT, NAME, COLOR, WIDTH) writes lines same as above but
%     with width WIDTH and color COLOR.
```

```
%
% Input:
%     LON: 1-D array of longitude line values. Separate lines are separated
%     by a NaN. Must correspond to LAT.
%     LAT: 1-D array of latitude line values. Separate lines are separated
%     by a NaN. Must correspond to LON.
%     NAME: String name of output file (without the .kml extension)
%     COLOR: MATLAB color (default is 'w'), supports vector colors.
%     WIDTH: int or float width value (default is 1)
%
% Output:
%     This function creates a kml file called NAME.kml in the current
%     working directory
%
% Examples:
%     % four different ways of calling kml_line:
%     load('palau_coastline.mat');
%     kml_line(lon_coast, lat_coast, 'palau_coastline');
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 'r');
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 1.5);
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 'magenta', 2);
%
%
% Cameron Sparr - Nov. 31, 2011
% cameronsparr@gmail.com
%
```

Mi most a legutolsó hívási módot fogjuk választani, hogy színt és vonalvastagságot is megadhassunk:

```
> kml_line(long, lat, 'hajokazas', 'magenta', 3);
```

Ezután Google Earth-be töltsük be a hajokazas.kml fájlt, és nézzük meg merre is volt a nyaralás!



Átkelés útvonala az Atlanti Óceánon Harwichtól Bostonig luxushajóval.

8. AZ OCTAVE MAPPING ÉS GEOMETRY CSOMAGJA

A különböző csomagok listája, telepítése, betöltése:

```
> pkg list % csomagok listázása
> % mapping csomag telepítése internetről közvetlenül, ha nincs
  telepítve
> % pkg install -forge mapping;
> pkg describe -verbose mapping % a mapping csomag parancsainak listája
> pkg load mapping; % mapping csomag betöltése
```

MAPPING PACKAGE

Ha már a neten elérhető megoldásokról beszélünk, akkor érdemes lehet kicsit belenézni az Octave Mapping és Geometry Package utasításaiba is, hogy mi az, amit mi is használni tudunk belőle. Néhányat már láttunk, hogyan lehet fok-perc-másodpercből tizedfokba és vissza váltani, fokból radiánba és vissza stb. (deg2rad;degtorad;rad2deg;radtodeg; degrees2dm; degrees2dms; dm2degrees; dms2degrees; függvények).

Milyen egyéb számunkra érdekes függvényei vannak még a mapping csomagnak?

Lehet a **gömbön** szögeket, ívhosszakat számítani, ami vetülettanból lehet ismerős.

azimuth: Azimut számítása a legnagyobb gömbi körön P_1 és P_2 között.

distance: távolság a legnagyobb gömbi kör mentén P_1 és P_2 között és opcionálisan az azimut is.

reckon: Első geodéziai alapfeladat a gömbön: adott pontból azimut és gömbi távolság (az ívhez tartozó középponti szög) megadásával a végpont koordinátáinak kiszámítása.

km2deg, km2rad, deg2km, rad2km: Gömbi távolságok középponti szögekké alakítása és vissza. Az alapértelmezett sugár 6371 km, de meg lehet adni tetszőleges sugarat, vagy a holdat, Marsot, és egyéb bolygókat is.

Mértékegységek közti átváltások:

km2sm, km2nm, nm2km, nm2sm, sm2km, sm2nm : Átváltás kilométerből mérföldbe, tengeri mélföldbe és vissza.

unitsratio: átváltási arány különböző mértékegysége között, pl. km, cm, láb, hüvelyk

wrapTo180, wrapTo360, wrapToPi, wrapto2Pi: szögek átalakítása a -180-+180 fok tartományra, vagy a 0-360 fok tartományra, vagy ugyanez radiánban.

Ezenkívül shape fájlok, rasztre fájlok olvasása, írása.

GEOMETRY PACKAGE

Ebben rengeteg utasítás van, most csak néhány fontosabb parancsot nézzünk meg:

centroid – súlypont számítás ponthalmazra

distancePoints – pontok közötti távolság minden kombinációban

findClosestPoint – legközelebbi pont indexének és távolságának megkeresése egy pontfelhőhöz képest

isPointInCircle – adott körön belül van-e a pont?

midPoint – középpont egy szakaszon

minDistancePoints – Legkisebb távolság sok pont között

polarPoint – Poláris pont számítás (első geodéziai alapfeladat) (mért távolság + irányszög)

angle2Points – 2 ponttal adott egyenes x tengellyel bezárt szöge

angle3Points – Térbeli szög kiszámítása P1, P2 és P3 között.

distancePointLine – Minimális távolság pont és egyenes között

isParallel – Két vektor párhuzamosság vizsgálata

isPerpendicular – Két vektor merőlegesség vizsgálata

pointOnLine – Pont létrehozása egyenesen adott távolságra

transformPoint – Pont transzformáció affin transzformációval

fillPolygon – Pontlista által megadott polygon kitöltése

A fenti csak néhány példa ebből a csomagból, ezeken kívül sok más is van a csomagban, például különböző tarnszformációk, körökhöz, ellipszisekhez, vonalakhoz, poligonokhoz tartozó parancsok 2D-ben, 3D-ben. Pl metszések, legrövidebb távolságok.

GEODÉZIAI, VETÜLETTANI ALAPFELADATOK

ELSŐ ÉS MÁSODIK GEODÉZIAI ALAPFELADAT A SÍKON

```
> %% Második geodéziai alapfeladat
> pkg load geometry;
> pkg load mapping;
> A = flip([657705.45 247565.56]);
> B = flip([658310.44 248489.88]);
> C = flip([658604.69 247832.58]);
> D = flip([658077.70 247431.38]);
> deltaAC = geodszog(rad2deg(angle2Points(A,C)))
> dAC = distancePoints(A,C)
> deltaBD = geodszog(rad2deg(angle2Points(B,D)))
> dBD = distancePoints(B,D)
>
> %% Első geodéziai alapfeladat
> A = flip([657705.45 247565.56]);
> tavAC = 938.05
> deltaAC = deg2rad(dms2degrees([73 27 42]))
> C_polar = flip(polarPoint(A, tavAC, deltaAC))
```

Eredmények:
```
deltaAC =  73-27-42.3313
dAC =  938.047044662461
deltaBD = 192-24-2.5431
dBD =  1083.78510674395

tavAC =  938.050000000000
deltaAC =  1.28214795733590
C_polar =

   658604.692404206   247832.582285545
```

Mint látjuk megoldható az első és a második geodéziai alapfeladat beépített függvényekkel, de be kell hozzá olvasni két csomagot is (mapping a rad2deg függvényhez és geometry a az angle2Points függvényhez) és figyelembe kell venni, hogy a matematikai és a geodéziai koordináta rendszerek eltérnek, ezért a flip paranccsal meg kell cserélni Y,X sorrendjét. Ebben az esetben lehet, hogy egyszerűbb, ha saját magunknak írunk egy távolság és irányszög számító függvényt, mint, ahogy ezt már egyszer meg is tettük a 2. gyakorlaton:

tav_irany.m fájl:
```
> function [t delta] = tav_irany(y1,x1,y2,x2)
>       dy = y2 - y1;
>       dx = x2 - x1;
>       t = sqrt(dy^2+dx^2);
>       delta = atan2d(dy,dx); % Octave esetén
>       %delta = atan2(dy,dx); % Matlab esetén
>       %delta = radtodeg(delta); % Matlab esetén
>       if delta<0 delta=delta+360; end
> end;
```

Használata:
```
> %% Második geodéziai alapfeladat saját függvénnyel
> A = [657705.45 247565.56];
> C = [658604.69 247832.58];
> [dAC deltaAC] = tav_irany(A(1),A(2),C(1),C(2))
> deltaAC = geodszog(deltaAC)
```

ELSŐ ÉS MÁSODIK GEODÉZIAI FŐFELADAT A GÖMBÖN

```
> page_screen_output(0);clc; clear all; close all;
> pkg load mapping % mapping csomag betöltése
> %% Vetülettan - első és második geodéziai főfeladat a gömbön
> % Kiinduló adatok
> fiA = dms2degrees([47 13 58.4976]);
> lA = dms2degrees([-1 3 14.6857]);
> fiC = dms2degrees([47 15 41.3749]);
> lC = dms2degrees([-2 14 27.8952]);
> alfaAB = dms2degrees([223 44 26.3670]);
> sAB = 222342.613;
> R = 6379743.001;
>
> % Első geodéziai főfeladat
> % Adott [fiA, lambdaA], az azimut (alfaAB) és az ívhossz (sAB)
> % Keresett [fiB, lambdaB]
> tetaAB = km2deg(sAB/1000, R/1000) % Ívhosszból középponti szög
> disp(degrees2dms(tetaAB)) % megjelenítés fok-perc-mp
> % megjelenítés fok-perc-mp saját függvénnyel geodéziai formában
```

```
> disp(geodszog(tetaAB))
> [fiB, lB] = reckon(fiA,lA,tetaAB,alfaAB) % első geodéziai főfeladat
> disp(geodszog(fiB))
> disp(geodszog(lB))
> alfaBA = azimuth(fiB,lB,fiA,lA) % ellentett irány azimutja
> disp(geodszog(alfaBA))
>
> % Második geodéziai főfeladat
> % Adott [fiA, lambdaA] és [fiC, lambdaC]
> % Keresett az azimut (alfaAC) és az ívhossz (sAC)
> [tetaAC alfaAC] = distance(fiA,lA,fiC,lC) % második geodéziai
  főfeladat
> disp(geodszog(alfaAC))
> alfaCA = azimuth(fiC,lC,fiA,lA) % ellentett irány azimutja
> disp(geodszog(alfaCA))
> format long;
> sAC = deg2km(tetaAC, R/1000)*1000 % középponti szögből ívhossz
```

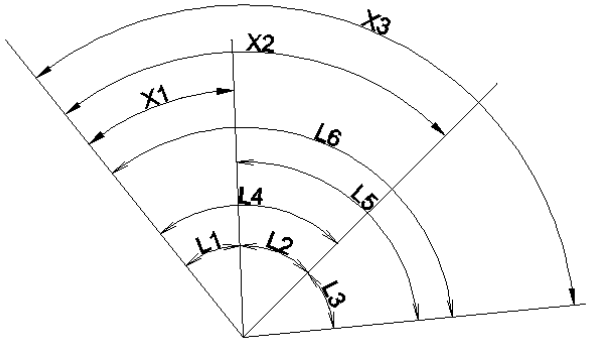
Eredmények:

```
tetaAB =  1.99683487701836
    1.00000000000000   59.00000000000000   48.60555726610255
  1-59-48.6056
fiB =  45.7728784819162
lB = -3.03341788401119
 45-46-22.3625
 -4-57-59.6956
alfaBA =  42.3046429670264
 42-18-16.7147

tetaAC =  0.806279583720703
alfaAC =  272.466880671765
272-28-0.7704
alfaCA =  91.5952628357265
 91-35-42.9462
sAC =  89777.2327177581
```

9. KIEGYENLÍTŐ SZÁMÍTÁSOK

SZÖGMÉRÉS KIEGYENLÍTÉSE

Határozzuk meg 4 irány által bezárt X_1, X_2 és X_3 szögeket, úgy, hogy a közbezárt szögeket minden kombinációban megmértük (L_1, L_2, L_3, L_4, L_5, L_6)!



(A példát lásd Detrekői: Kiegyenlítő számítások, Tankönyvkiadó, Budapest, 1991., 145-150. oldal)

A mérési eredmények a következőek:

L_1 = 30-40-24 L_4 = 72-46-40

L_2 = 42-06-12 L_5 = 88-18-27

L_3 = 46-12-20 L_6 = 118-58-59

a) A mérésekről először feltételezzük, hogy azonos pontosságúak.

b) Az L_1, L_2, L_3 méréseket μ=2”, az L_4, L_5, L_6 méréseket μ=4” középhibával jellemezhetjük.

ADATOK MEGADÁSA MATLAB/OCTAVE HASZNÁLATÁVAL

Adjuk meg a mérési eredményeket vektorokban, majd fűzzük össze őket egy mátrixba!

```
> page_screen_output(0);
> clc; clear all; close all;
>
> L1 = [30,40,24];
> L2 = [42,06,12];
> L3 = [46,12,20];
> L4 = [72,46,40];
> L5 = [88,18,27];
> L6 = [118,58,59];
> L = [L1;L2;L3;L4;L5;L6]
```

SAJÁT FÜGGVÉNYEK KÉSZÍTÉSE FOK-PERC-MÁSODPERC ÉRTÉKEK KEZELÉSÉHEZ

FOK-PERC-MÁSODPERC ÁTVÁLTÁSA TIZEDFOKBA

Az Octave/Matlab alapértelmezésként a radiánt tekinti a szög mértékegységének (pl. sin, cos stb függvények esetében), illetve még tizedfokban megadott értékekkel is tud dolgozni (pl. sind, cosd stb) fok-perc-másodperc értékekkel azonban nem.

Első feladat, hogy valahogy kezelhetővé tegyük a fok-perc-másodperc értékeket. Írjunk függvényt, ami átváltja a szöget tizedfokba, illetve egy másikat, ami radiánba! Itt el kell döntenünk, hogyan szeretnénk megadni a fok-perc-másodperc értékeket az adott függvénynek, pl. lehetnek ezek különálló bemenetek vesszővel elválasztva, illetve egy bemenet, ahol vektorban tárolódnak az értékek. Ezt megtehetjük ún. anonym függvény írásával, ahol az m fájlunkban definiáljuk közvetlenül a függvényünket.

Pl. függvény 3 vesszővel elválasztott bemeneti változóval.

```
> fpm = @(f,p,m) f + p/60 + m/3600;
```

Teszteljük:

```
> fpm(30,40,24)
```

Vagy függvény egy vektor bemenettel, ahol a vektor elemeiben vannak megadva a fok-perc-másodperc értékek.

```
> fpm2 = @(f) f(1) + f(2)/60 + f(3)/3600;
> fpm2(L1)
```

Célszerűbb lehet azonban külön m fájlban definiálni ezeket, saját függvényként. így egyrészt más feladat során is használhatjuk ezeket, illetve több lehetőségünk van a függvény paraméterezésére. Megadhatunk pl. több kimenetet, ha szükséges, illetve a bemenetek számának, jellegének függvényében változhat a feldolgozás is, így akár egy függvényben tudjuk kezelni mind a kettő fenti esetet, hogy akár vektorként, akár külön bemeneti változókként megadhatjuk a fok-perc-másodperc értékeket.

A legegyszerűbb eset az alábbi:

```
> function fok = fpm2fok(f,p,m)
> fok = f + p/60 + m/3600;
> end
```

Most a függvénynek az fpm2fok nevet adtuk, ilyenkor fpm2fok.m fájlként is kell elmenteni. A függvény hívása egyszerűen az fpm parancs kiadásával történhet, abból a könyvtárból, ahová ezt az m fájlt mentettük.

Teszteljük:

```
> %L1fok = fpm2fok(L1) % hibaüzenetet kapunk
> L1fok = fpm2fok(L1(1), L1(2), L1(3))
>
> Lf = L(:,1)
> Lp = L(:,2)
> Lm = L(:,3)
> Lfok = fpm2fok(Lf, Lp, Lm)
```

Módosítsuk ezt a függvényt, hogy többféle bemenettel is működjön, akár külön-külön vannak megadva a fok-perc-másodperc értékek, akár egy 3 oszlopú sorvektorban, akár egy 3 oszlopú oszlopvektorban több szög egyszerre.

```
> function fok = fpm2fok(f,p,m)
> % A fuggveny fok-perc-masodperc ertekekbol tizedfokot szamol.
> % A bemenet lehet vesszovel elvalasztva fok,perc, masodperc vagy
> % lehet vektorban megadva [fok perc másodperc] alakban is.
> % Matrixban adott bemenetet is elfogad a fuggveny, ahol
> % az 1. oszlop a fok, a 2. a perc, a 3. a masodperc
> %
> % Kimenet: tizedfok (skalar vagy vektor bemenettol fuggoen)
>
> switch nargin
>     case 3
>         fok = f + p/60 + m/3600;
>     case 1
>         fok = f(:,1) + f(:,2)/60 + f(:,3)/3600;
> end
> end
```

A függvény első sora után megadott megjegyzések (%) alkotják a help parancsra kiírt dokumentációt (a 'help fpm2fok' után ez jelenik meg a parancssorban). Saját függvényeket célszerű jól ledokumentálni, hogy később is tudjuk milyen bemenetet vár a program, és milyen kimenetet kapunk.

A nargin (Number of function input arguments) változó a bemenetek számát tárolja. Ha előre tudjuk, hogy maximum hány változót vár a függvény, akkor egyszerűen megadhatunk ennyi változót bemenetként, és utána a nargin lekérdezése után mindig azt a megoldást hívjuk meg, ami megfelel a változók számának. Ha a maximális változó szám ismeretlen, akkor a változó lista végére meg kell még adni a varargin változót is (Variable-length input argument list).

Használjuk a switch elágazást a változók számának megfelelő parancs hívására!

Ha külön vannak tárolva a fok-perc-másodperc értékek, akkor az f + p/60 + m/3600 parancs működik mind skalár, mind vektor adatokkal, viszont, ha egy 3 oszlopú mátrixban vannaz az adatok, akkor a f(1) + f(2)/60 + f(3)/3600 parancs nem működik az összes tárolt szögre (egy sor egy szög), ahhoz a f(:,1) + f(:,2)/60 + f(:,3)/3600 parancsot kell kiadni.

Teszteljük különböző bemenetekkel:

```
> L1fok = fpm2fok(30,40,24)
> L1fok = fpm2fok(L1)
> Lfok = fpm2fok(L)
> Lfok = fpm2fok(Lf,Lp,Lm)
```

Egy függvénynek nem csak egy, hanem több kimenete is lehet, jelen esetben például lehet a függvény egy másik kimenete a tizedfok mellet a szög radiánban kapott értéke is. Ehhez az alábbiak szerint módosítsuk a függvényt:

```
> function [fok rad]= fpm2fok(f,p,m)
> % A fuggveny fok-perc-masodperc ertekekbol tizedfokot és radiant
  szamol.
> % A bemenet lehet vesszovel elvalasztva fok,perc, masodperc vagy
> % lehet vektorban megadva [fok perc másodperc] alakban is.
```

```
>  % Matrixban adott bemenetet is elfogad a fuggveny, ahol
>  % az 1. oszlop a fok, a 2. a perc, a 3. a masodperc
>  %
>  % 1. kimenet: tizedfok (skalar vagy vektor bemenettol fuggoen)
>  % 2. kimenet: a szog erteke radianban
>
>  switch nargin
>      case 3
>          fok = f + p/60 + m/3600;
>      case 1
>          fok = f(:,1) + f(:,2)/60 + f(:,3)/3600;
>  end
>  rad = fok *pi / 180;
>  end
```

Ilyenkor, ha egy kimenettel hívjuk meg a függvényt az első kimenetet fogja csak visszaadni, a tizedfok értékét, több kimenet estében azonban mind a tizedfokot, mind a radián visszaadja. Például:

```
>  [a b] = fpm2fok(30,40,24)
>  % a =  30.673
>  % b =  0.53535
>
>  [Lfok Lrad] = fpm2fok(L)
```

FOK ÁTVÁLTÁSA RADIÁNBA (OTTHONI ÁTNÉZÉSRE)

Írjunk egy olyan függvényt, ami csak radiánba vált át fok értéket. Viszont többféle bemenetet kezeljen. A fok értéket meg lehessen adni akár tizedfokban, akár fok-perc-másodperc értékekben, és ez utóbbi is működhet a korábban megadott módokon (3 elemű sorvektorban, vagy vesszővel elválasztva).

```
>  function rad = fok2rad(f,p,m)
>  % Fokban lévő szoget radianba alakit.
>  % Ha egy bemenet van, akkor megvizsgalja, hogy 1 vagy 3 oszlopa van-
   e.
>  % Egy oszlop eseten tizedfokban adott szoget alakit at,
>  % ha 3 oszlopa van, akkor fok-perc-masodperc ertekeket alakit at
   radianna.
>  % Ha 3 bemenete van, akkor azt fok-perc-másodpercnek tekinti.
>
>    switch nargin
>      case 3
>           rad = (f + p/60 + m/3600) * pi / 180;
>      case 1
>      if size(f,2)==1
>          rad = f * pi / 180;
>      end
>      if size(f,2)==3
>          rad = (f(:,1) + f(:,2)/60 + f(:,3)/3600) * pi / 180;
>      end
>    end
>  end
```

Ez utóbbi a (*pi/180) átváltáson kívül abban tér el az előzőtől, hogy egy bemenet esetén vizsgálja, hogy az a bemenet 1 vagy 3 oszlopból áll-e. Egy oszlop esetén feltételezi, hogy a szögek tizedfokban adottak, 3 esetén pedig fok-perc-másodpercben.

A size parancs a mátrix méretét adja vissza. size(M) hívása esetén először a sor, utána az oszlopok számát, size(M,1) a sorok számát, size(M,2) az oszlopok számát adja vissza.

TIZEDFOK ÁTVÁLTÁSA FOK-PERC-MÁSODPERCRE (OTTHONI ÁTNÉZÉSRE)

Az első függvényünk fordítottja is kelleni fog, ha az eredményeket fok-perc-másodperc értékben szeretnénk kiíratni, ahogy az geodéziában szokásos.

```
> function fpm = fok2fpm(x);
> % A fuggveny tizedfokbol fok-perc-masodperc ertekekbe szamol at.
> % A bemenet lehet egy skalar vagy vektor is.
> % A kimenet egy vektor vagy egy matrix, bemenettol fuggoen, ahol
> % az 1. oszlop a fok, a 2. a perc, a 3. a masodperc
>
>   f = fix(x);
>   p = fix((x-f) * 60);
>   m = ((x-f)*60-p)*60;
>
>   fpm = [f p m];
> end
```

Ez a függvény tizedfokból számol át fok-perc-másodpercbe. Ez is működik akár vektorban adott bementtel is, több szög együttes átszámítására. A fix parancs mindig a 0 felé kerekít egészre. Azért célszerű ezt használni round helyett, hogy akár negatív szögekre is működjön az átváltás.

FOK-RADIÁN ÁTVÁLTÁS MAPPING PACKAGE HASZNÁLATÁVAL

A fenti függvényeken megtanulhattuk, hogyan tudunk olyan saját függvényeket írni, amik többféle bemenetet is elfogadnak és többféle kimenetet engedélyeznek. A tizedfokból fok-perc-másodpercbe, radiánba és vissza átszámító függvényekre gyakran szükség van a geodéziai számítások során. A feladat gyakorisága folytán természetesen lehet olyan kész függvénykönyvtárat is találni, amiben a fentiek benne vannak. Octave esetében ez a Mapping csomag (package), a Matlabban pedig a Mapping Toolbox. Nézzük meg, hogy milyen függvényeket találhatunk ebben, és hogyan tudjuk ezeket használni!

Ellenőrizzük le, hogy telepítve van-e a Mapping package:

```
> pkg list
```

Ha nincs telepítsük az Octave-Forge oldalról:

```
> % mapping csomag telepítése internetről közvetlenül:
> pkg install -forge mapping
```

Nézzük meg milyen parancsok vannak benne:

```
> pkg describe -verbose mapping % kilistázza a mapping csomag
  parancsait
```

Töltsük be a csomagot!

```
> pkg load mapping; % mapping csomag betöltése
```

A számunkra most fontos parancsok:

```
deg2rad;degtorad;rad2deg;radtodeg;
degrees2dm; degrees2dms; dm2degrees; dms2degrees;
```

Teszteljük a dms2degrees és a deg2rad függvényt! Kérdezzük le help-pel milyen bemenetet várnak!

```
> L1degree = dms2degrees(L1)
> Ldegree = dms2degrees(L)
> Lrad = deg2rad(Ldegree)
```

A SZÖGMÉRÉS KIEGYENLÍTÉSÉNEK ALAPELVE

Az előző segédfüggvények elkészítése (vagy betöltése) után hozzá is láthatunk a megoldáshoz.

Az ábra alalpján a 3 meghatározandó szög értékre 6 egyenletet írhatunk fel a 6 mérési eredmény felhasználásával. Ez egy lineáris egyenletrendszert alkot.

```
> % A megoldando linearis egyenlet rendszer
> % L1 = X1;
> % L2 = X2-X1;
> % L3 = X3-X2;
> % L4 = X2;
> % L5 = X3-X1;
> % L6 = X3;
```

Egy lineáris egyenletrendszer általános alakja a következő:

$$A \cdot x = b$$

Ennek a megoldása, ha egyértelmű a megoldás:

$$x = A^{-1} \cdot b$$

Jelenleg azonban 6 egyenletünk van 3 ismeretlenre. Ez egy túlhatározott egyenlet rendszer. Ilyenkor nincs egyértelmű megoldás, hanem a maradék eltérések négyzetösszegét minimalizáljuk.

$$\|A \cdot x - b\|^2 \rightarrow min.$$

Ez a célfüggvény. Egy függvénynek akkor lehet minimuma, ha az első derivált értéke zérus.

Nagyon leegyszerűsítve, ha az A és a b nem mátrix, illetve vektor lenne, hanem egy-egy skalár, akkor az $(A \cdot x - b)^2$ függvény deriváltja (felhasználva az összetett függvény deriválására vonatkozó szabályt a $2 \cdot (A \cdot x - b) \cdot A = 2 \cdot A \cdot A \cdot x - 2 \cdot A \cdot b$ lenne. Ha ez egyenlő nullával, akkor 2-vel le lehet osztani nyugodtan, így a megoldandó egyenlet a $A \cdot A \cdot x - A \cdot b = 0$ lenne. Mátrixok, illetve vektorok esetében a megoldandó egyenlet a következő lesz:

$$A^T \cdot A \cdot x - A^T \cdot b = 0$$

Ha ezt megoldjuk x-re, akkor a következő egyenletet kapjuk:

$$x = (A^T \cdot A)^{-1} \cdot A^T \cdot b$$

Ez a megoldás egységsúlyú esetben igaz, ha a méréseink különböző súlyúak, akkor a súlymátrixot is bele kell vennünk:

$$x = (A^T \cdot P \cdot A)^{-1} \cdot (A^T \cdot P \cdot b)$$

Ez tulajdonképpen a pszeudoinverzzel történő megoldás.

MEGOLDÁS MATLAB/OCTAVE HASZNÁLATÁVAL

A korábban felírt lineáris egyenletrendszert írjuk fel $A \cdot x = b$ alakban, ahol A az ismeretlenek (X1, X2, X3) együtthatóit tartalmazza az egyes egyenletekben, b pedig a mérési eredményeket. Az A mátrixokt alakmátrixnak szokás nevezni a geodéziában. A fok-perc-másodper értékeket alakítsuk át tozedfokba a számításhoz!

```
> % A megoldando linearis egyenlet rendszer
> % L1 = X1;
> % L2 = X2-X1;
> % L3 = X3-X2;
> % L4 = X2;
> % L5 = X3-X1;
> % L6 = X3;
>
>  A = [1 0 0;
>     -1 1 0;
>     0 -1 1;
>     0 1 0;
>     -1 0 1;
>     0 0 1]
>
> b = fpm2fok(L)
> % vagy
> b = dms2degrees(L)
```

A megoldás:

```
> X = inv(A'*A)*(A'*b)
```

Megjegyzés: ugyanez az eredmény megkapható a beépítet pszeudoinverz függvény alkalmazásával is, vagy az azzal egyenértékű \ jel használatával:

```
> X = pinv(A)*b
```

vagy a legrövidebben:

```
> X=A\b
```

Eredmény:

```
X =
   30.6742
   72.7774
  118.9826
```

Alakítsuk át a megoldást fok-perc-másodperccé!

```
> Xfpm = degrees2dms(X)
```

Az eredmény:

```
xfpm =
   30.0000   40.0000   27.0000
   72.0000   46.0000   38.7500
  118.0000   58.0000   57.2500
```

Ha nem lineáris egyenletet szeretnénk megoldani legkisebb négyzetek módszerével, akkor először linearizálni kellene az egyenletek egy kezdőérték körül (Tayor-sorba fejtéssel), és úgy kellene megoldani a feladatot.

Nézzük meg a súlyozott megoldást, amikor az L_1, L_2, L_3 méréseket μ=2”, az L_4, L_5, L_6 méréseket μ=4” középhibával jellemezhetjük! Vagyis az első 3 szöget pontosabban mértük, tehát nagyobb súllyal vesszük őket figyelembe. Ebben az esetben először a súlymátrixot kell felvennünk, ami az egyes mérések súlyát tartalmazó diagonálmátrix.

A súlyok és a középhibák kapcsoltát az alábbi aránnyal írhatjuk le:

$$p_1 : p_2 = \frac{1}{{m_1}^2} : \frac{1}{{m_2}^2}$$

Mivel ez egy arányosság az egészet beszorozhatjuk egy tetszőleges számmal, például a nevezők legkisebb közös többszörösével (least common multiple - lcm), hogy egész számokat kapjunk a súlyokra!

```
> disp('Sulyozott megoldas')
>
> m1 = 2; m2 = 4; % középhibák
> c = lcm(m1^2,m2^2) % középhiba négyzetek legkisebb közös többszöröse
> p1 = c/m1^2 % súlyok
> p2 = c/m2^2 % súlyok
> % súlymátrix
> P = diag([p1 p1 p1 p2 p2 p2])
```

Az eredmény:

```
c2 = 16
p1 = 4
p2 = 1
P =  4     0     0     0     0     0
     0     4     0     0     0     0
     0     0     4     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

A megoldás:

```
> X2 = inv(A'*P*A)*(A'*P*b);
> X2fpm = degrees2dms(X)
```

Az eredmény:

```
X2fpm =
   30.0000   40.0000   25.2293
   72.0000   46.0000   37.3268
  118.0000   58.0000   56.7561
```

EGYENES ILLESZTÉSE

A következő feladat egy egyenes illesztése lesz legkisebb négyzetek módszerével. Adjuk meg az egyenes pontjait, mint mérési eredményeket. Egy adott egyenes 10 pontját „rontsunk el” mérési hibákkal!

Az egyenes egyenletét az $y = m \cdot x + b$ egyenlettel adjuk meg. Az egyenes két paramétere az y tengellyel vett metszete (b) és a meredeksége (m).

```
> disp('Egyenes illesztése')
> b = 8.765; m = 1.234;
> f = @(x) b + m*x
> figure(1);
> h = ezplot(f,[0, 11])
> set(h, 'LineWidth', 2)
```

Az egyenes 10 pontját számoljuk ki 1:10 közötti x koordináták esetében, majd adjunk hozzá mind x, mind y értékekhez egy normális eloszlású véletlen hibát (randn).

```
> x = [1:10]'
> y = f(x)
> x = x + randn(numel(x),1)
> y = y + randn(numel(y),1)
> hold on;
> plot(x,y,'r*')
```

Illesszük ezekre a pontokra a legkisebb négyzetek értelmében legjobban illeszkedő egyenest! A 10 pontra 10 egyenletet tudunk felírni, miközben két ismeretlenünk van, ez egy erősen túlhatározott egyenlet.

$$b + m \cdot x_1 = y_1$$
$$b + m \cdot x_2 = y_2$$
$$\dots$$
$$b + m \cdot x_{10} = y_{10}$$

A két paramétert adjuk meg a p vektorban, p(1)=b és p(2)=m.

Mátrixosan felírva $A \cdot p = y$:

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_{10} \end{pmatrix} \cdot \begin{pmatrix} b \\ m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_{10} \end{pmatrix}$$

Állítsuk elő az A alakmátrixot a Matlab-ban! Ehhez egy 10 elemű (amilyen hosszú az x vektor) egyesekből álló oszlopvektorhoz hozzá kell fűzzük az x értékeket tartalmazó vektort. A numel parancs visszadja egy mátrix vagy vektor elemeinek a számát. A ones(n,m) parancs egy csupa egyesből álló mátrixot hoz létre, n sorral, m oszloppal.

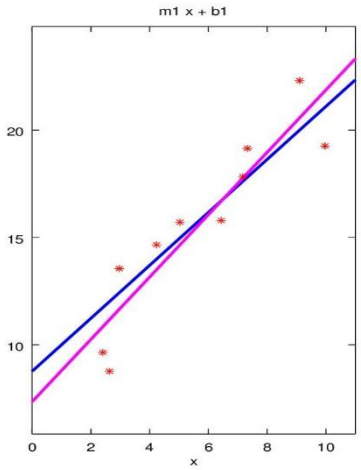
```
> A = [ones(numel(x),1) x]
```

Oldjuk meg az egyenletrendszert a legkisebb négyzetek elve szerint!

```
> p = inv(A'*A)*A'*y
> % vagy
> p = A\y
```

Az eredmény (természetesen többszöri futtatásra a randn függvény miatt nem állandó), pl.:

```
p =
   7.3466
   1.4525
```



Ábrázoljuk:

```
> b1 = p(1); m1 = p(2);
> f1 = @(x) m1*x + b1
> h = ezplot(f1, [0,11])
> set(h,'Color','m','LineWidth', 2)
> axis('equal');
> print egyenes.jpg;
```

SÍK ILLESZTÉSE

Olvassuk be a domborzatmodellezéskor már használt mérési állományunkat (meres_coo.txt)! Korábban láttuk a szintvonalas domborzatnál, hogy a terep meglehetősen síknak tekinthető. Illesszünk egy kiegyenlítő síkot a pontokra! Keressük meg a sík paramétereit.

```
2001                577057.011 188795.517  142.042
2002                577051.903 188783.028  140.821
2003                577051.044 188772.868  140.317
2004                577053.460 188758.714  139.622
...
```

A sík általános egyenlete a következő:

$$a_0 + a_1 \cdot x + a_2 \cdot y = z$$

Ahány pontunk van, annyi egyenletet tudunk felírni (163), míg összesen 3 ismeretlenünk van az a_0, a_1, a_2 paraméter, tehát ismét túlhatározott lineáris egyenletrendszert kell megoldani. Mátrix alakban felírva a következő egyenletrendszert kell megoldanunk:

$$\begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \dots & \dots & \dots \\ 1 & x_n & y_n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{pmatrix}$$

vagyis:

$A \cdot p = z$, ahol p a paraméter vektor $p = [a_0; \quad a_1; \quad a_2]$.

A megoldást a már korábban levezetett alakban kapjuk meg:

$$p = (A^T \cdot A)^{-1} \cdot A^T \cdot z$$

MEGOLDÁS – PROBLÉMÁK

Oldjuk meg ezt Matlabban/Octave-ban, majd jelenítsük meg az eredményt!

```
> clear all; close all; clc;
> page_screen_output(0); % ez csak Octave-ban kell!
```

```
> data=load('meres_coo.txt');
> x = data(:,2);
> y = data(:,3);
> z = data(:,4);
> A = [ones(size(x)) x y];
> p = inv(A'*A)*(A'*z)
```

A futtatás után kapunk egy figyelmeztetést:

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 4.191364e-021.
```

Ez arra utal, hogy a megoldás pontatlan lehet, és nagyon bizonytalan. Kérdezzük le az $A^T \cdot A$ mátrix kondíciószámát!

```
> c=cond(A'*A)
```

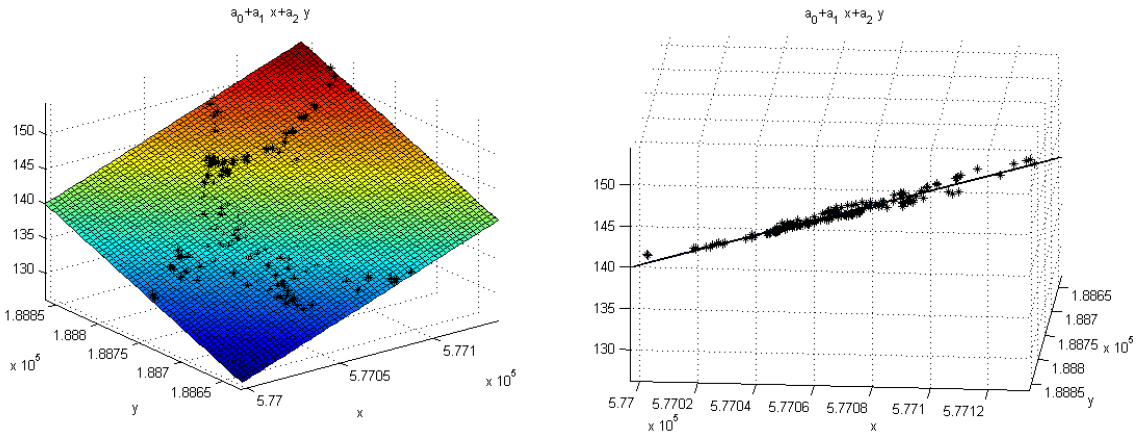
Eredmény:

```
c =  2.1001e+020
```

Minél nagyobb a kondíciószám, annál bizonytalanabb a megoldás, mivel a kondíciószám egy hányados a kimenet és a bemenet relatív hibája között. Egy kis változás a bemeneti adatokban a megoldás nagy változását okozhatja.

Ábrázoljuk azért a megoldást!

```
> a0=p(1)
> a1=p(2)
> a2=p(3)
> f = @(x,y) a0+a1*x+a2*y
> figure(1); hold on;
> ezsurf(f, [min(x) max(x) min(y) max(y)])
> plot3(x,y,z, 'k*')
```



Ha megnézzük az adatokat, akkor látszik, hogy több százezres nagyságrendű EOV y,x koordinátákkal dolgoztunk, hozzá 100-200 m körüli z értékekkel. Ilyenkor a numerikus számítás bizonytalanságát csökkenthetjük, ha áttérünk súlyponti koordinátákra és nem a több százezres koordinátákkal dolgozunk.

MEGOLDÁS SÚLYPONTI KOORDINÁTÁKKAL

A súlyponti y,x koordinátákhoz ki kell számolni a koordináták átlagát és kivonni ezt a mérési eredményekből. A végleges eredménynél és megjelenítésnél sem szabad azonban elfelejteni, hogy ezeket az átlagokat ki kell vonni az y,x értékekből!

```
> % sulyponti koordinatak
> XS = mean(x)
> YS = mean(y)
>
> xs = x - XS;
> ys = y - YS;
>
> As = [ones(size(xs)) xs ys];
> ps = inv(As'*As)*(As'*z)
> cs=cond(As'*As)
>
> a0s=ps(1)
> a1s=ps(2)
> a2s=ps(3)
>
> fs = @(x,y) a0s + a1s * (x-XS) + a2s * (y-YS)
>
> figure(2)
> ezsurf(fs, [min(x) max(x) min(y) max(y)])
> hold on;
> plot3(x,y,z, 'b.', 'MarkerSize', 15)
```

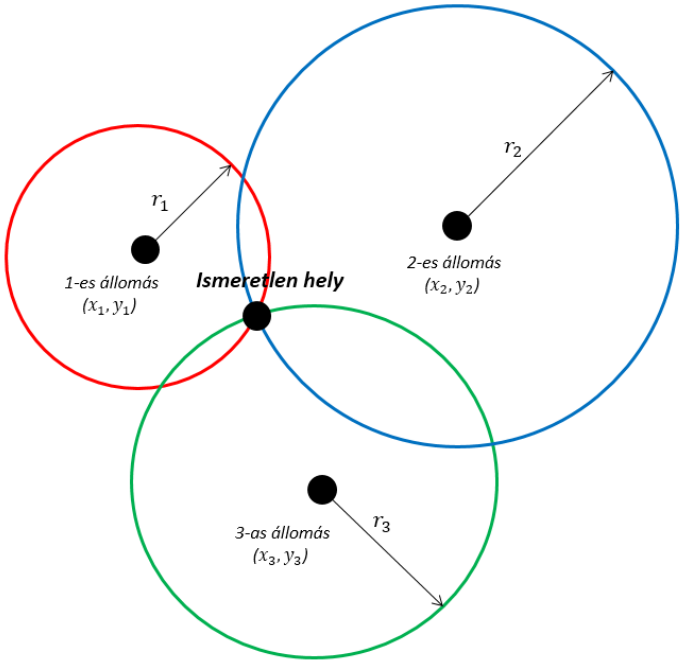
A súlypont koordinátáit YS és XS jelöli. A kiszámított súlyponti koordináták pedig ys és xs. A minimális és maximális xs: [-66.342; 71.208], a minimális és maximális ys: [-126.7668; 108.612]. Ezekkel az értékekkel számolva jóval kisebb a numerikus számítás pontatlansága, a kondíciószám 10^{20} nagyságrend helyett mindössze 10^{3} nagyságrendű. Most nem is kapunk figyelmeztetést, csak a megoldást.

Egy másik módja a megoldás pontosításának, ha nem az $x = (A^T \cdot A)^{-1} \cdot A^T \cdot b$ alakban oldjuk meg a problémát, hanem használjuk az Octave/Matlab valamelyik beépített megoldó módszerét túlhatározott egyenletekre. Pl az $A \cdot x = b$ egyenletet túlhatározott esetben is megoldhatjuk az $x = A \backslash b$ alakban. Ez Octave esetében SVD (Singular Value Decomposition) felbontással történő megoldást jelent, ami numerikusan sokkal stabilabb. Ha ezzel oldjuk meg a feladatot, akkor az eredeti koordinátákkal sem lép fel a rosszul kondicionáltság problémája. Az eredmények némileg eltérnek a másik módszerrel kapott eredményektől az eredeti koordinátákat használva, míg a súlyponti koordinátáknál megegyeznek.

POZÍCIÓ MEGHATÁROZÁS MOBILTELEFONOKKAL

A mobiltelefonok pozíciójának meghatározásakor rendelkezésre áll az eszköz és a mobiltornyok távolsága. A távolságok egy-egy kört határoznak meg a bázisállomások körül (lásd ábra). A körök másodfokú egyenletek, és ezek metszéspontja adja a mobiltelefon pozícióját. Ez a probléma az ívmetszés (angolul lateration). Amennyiben

legalább 3 távolság, azaz 3 kör, valamint a bázisok koordinátái ismertek, az ismeretlen hely meghatározható. Több távolság esetében kiegyenlítésre van szükség, ami tekintve az egyenleteket, most nemlineáris egyenletrendszerre alkalmazott legkisebb négyzetek módszerével történhet.



Az egyes mobiltornyok koordinátáit és a távolságméréseket az alábbi táblázat foglalja össze.

Mobil torony sorszáma	**X koordináta** (x_i) **[m]**	**Y koordináta** (y_i) **[m]**	**Mért bázis-terminál távolság** (r_i) **[m]**
1	561	1487	2130
2	5203	4625	5620
3	5067	-5728	6040
4	1012	5451	5820

Az egyenleteket a következő implicit alakban adhatjuk meg:

$$(x - x_i)^2 + (y - y_i)^2 - r^2 = 0,$$

ahol x_i, y_i a mobiltornyok koordinátái, x,y pedig a keresett álláspont.

A megoldáshoz ismét az eltérések négyzetösszegét kell minimalizálni. Ehhez most az Octave/Matlab beépített szimplex módszerét fogjuk használni, az fminsearch parancsot (de akár az fminunc parancs is használható, ami kvázi-Newton minimalizálást alkalmaz). Itt viszont már szükség lesz kezdőérték megadására, ami

lineáris esetben még nem volt követelmény. A kezdőértékeket most ábrából vesszük, így először szükséges ábrázolni az egyenleteket.

NEMLINEÁRIS LEGKISEBB NÉGYZETEK MÓDSZERE

Adjuk meg először a mobiltornyok koordinátáit, a mért távolságokat, és ábrázoljuk a pontokat!

```
> clear all; clc; close all;
> page_screen_output(0); % ez csak Octave-ban kell!
>
> xt = [561; 5203; 5067; 1012]
> yt = [487; 4625; -5728; 5451]
> rm = [2130; 5620; 6040; 5820]
>
> figure(1); hold on;
> plot(xt, yt, 'r*')
```

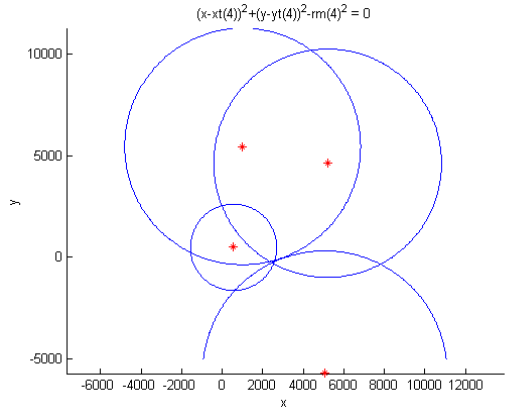
Definiáljuk ezek után a tornyoktól mért távolságokat függvényekkel és ábrázoljuk ezeket!

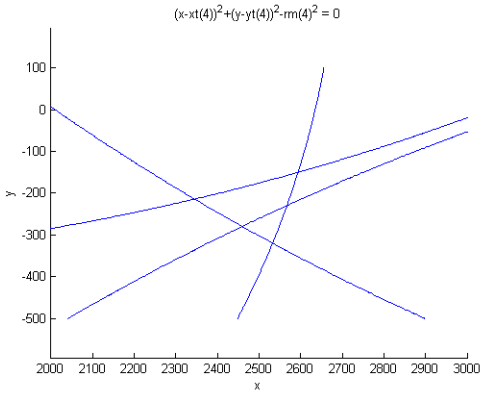
```
> eq1 = @(x,y) (x-xt(1)).^2 + (y-yt(1)).^2 - rm(1).^2
> eq2 = @(x,y) (x-xt(2)).^2 + (y-yt(2)).^2 - rm(2).^2
> eq3 = @(x,y) (x-xt(3)).^2 + (y-yt(3)).^2 - rm(3).^2
> eq4 = @(x,y) (x-xt(4)).^2 + (y-yt(4)).^2 - rm(4).^2
>
> ezplot(eq1, [-5000 12000])
> ezplot(eq2, [-5000 12000])
> ezplot(eq3, [-5000 12000])
> ezplot(eq4, [-5000 12000])
> axis equal
```

Figyeljünk arra, hogy a függvények definiálásakor használjunk pontot (.) a hatványozás, szorzás, osztás művelete előtt, hogy vektorokra is hívható legyen elemenként a függvény.

Nagyítsunk rá a minket érdeklő területre!

```
> figure(2); hold on;
> ezplot(eq1, [2000 3000 -500 100])
> ezplot(eq2, [2000 3000 -500 100])
> ezplot(eq3, [2000 3000 -500 100])
> ezplot(eq4, [2000 3000 -500 100])
> axis equal
```





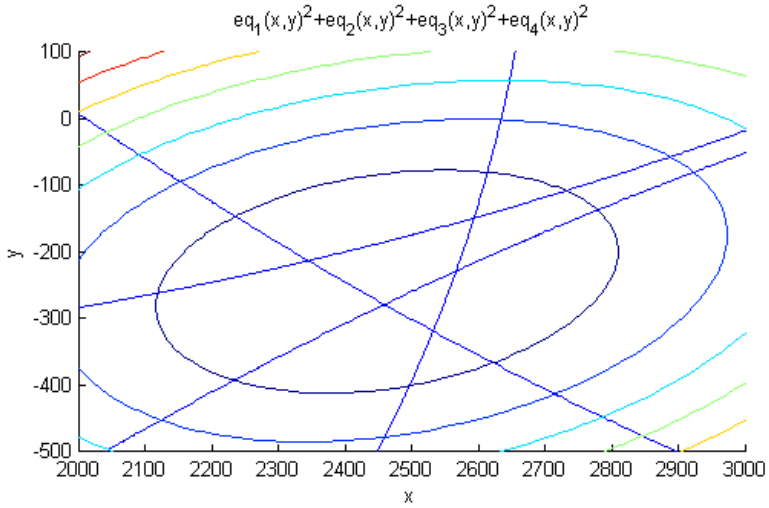
Definiáljuk a minimalizálandó függvényt, az eltérések négyzetösszegét!

```
> err = @(x,y) eq1(x,y).^2 + eq2(x,y).^2 + eq3(x,y).^2 + eq4(x,y).^2;
```

Ábrázoljuk ezt is a fenti rajzon szintvonalakkal!

```
> ezcontour(err,[2000 3000 -500 100]);
```

(Színezett szintvonalakkal ugyanez: `ezcontourf(err,[2000 3000 -500 100]);`)



A megoldás a fenti függvény minimuma lesz. Ehhez fel kell vennünk egy kezdőértéket. A rajz alapján a kezdőérték legyen:

```
> x0 = [2400; -300]
```

A megoldáshoz az ’err’ függvényt vektor változóssá kell alakítanunk! Rajzoljuk ki a megoldást is!

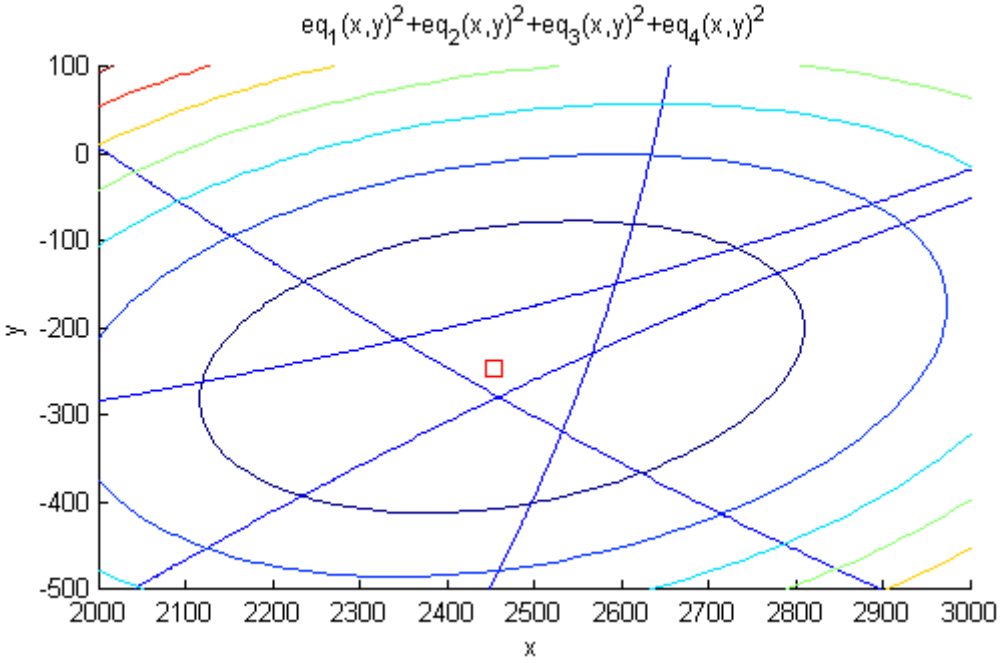
```
> err1 = @(x) err(x(1),x(2));
> sol = fminsearch(err1,x0)
> % sol2 = fminunc(err1,x0)
> plot(sol(1), sol(2), 'rs')
> plot(xm, ym, 'r*')
```

Nézzük meg az eltérést a mért távolságok és kiegyenlített álláspont - mobiltornyok távolságai között!

```
> % hibák
> ex = xt - sol(1);
> ey = yt - sol(2);
> er = rm - sqrt(ex.^2+ey.^2)
```

Az eltérések:

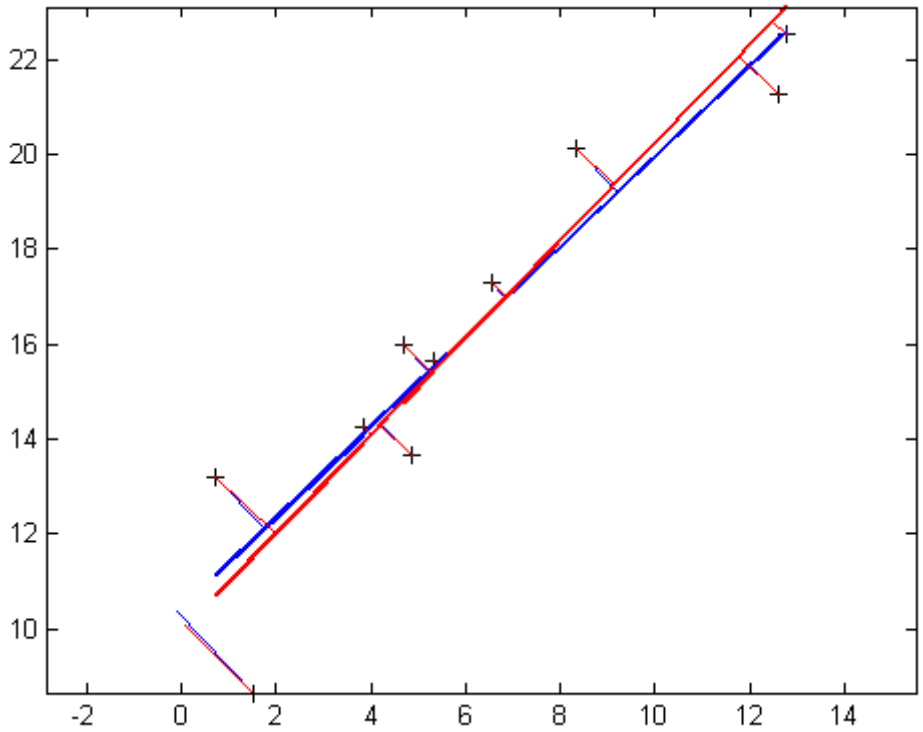
```
er =
   99.0847
   26.3188
  -31.7595
  -57.7440
```



TELJES LEGKISEBB NÉGYZETEK MÓDSZERE – EGYENES ILLESZTÉSE

A következőkben egy példa lesz a teljes legkisebb négyzetek módszerével történő egyenes illesztésére. A korábbi egyenes illesztésnél az y irányú eltérések négyzetösszegét minimalizáltuk az egyeneshez képest, feltételeztük, hogy az x koordináta hibátlan. Ez a feltételezés azonban többnyire nem állja meg a helyét, többnyire egy mérésnél mind a két koordináta hibával terhelt, ezért jobb megközelítés, ha a pontok egyenestől való távolságának négyzetösszegét minimalizáljuk.

Az alábbi ábrán az előző gyakorlatban a hagyományos legkisebb négyzetek módszerével illesztett egyenes látható kékkel és pirossal a teljes legkisebb négyzetek módszerével illesztett egyenes, berajzolva az egyenestől mért távolságokat.



A nehézséget az jelenti a feladatban, hogy egy olyan egyenestől való távolságot kell meghatározni, aminek egyelőre nem ismerjük az egyenletét, és maga a probléma sem lineáris. A feladat iterációkkal határozható meg, szükség van kezdőértékekre a paraméterekhez az elején, ez lehet például a hagyományos legkisebb négyzetek módszerével kapott egyenes két paramétere. Fel kell vennünk egy célfüggvényt, amit minimalizálni szeretnénk, ez a minimalizálandó távolság négyzetösszeg lesz, ez egy külön függvényben kerül definiálásra, és szükség lesz egy függvényre, ami pont-egyenes távolságot számol.

A megoldás a következő lesz:

A célfüggvény:

```
>  function celertek = celfuggveny(p)
>    global x;
>    global y;
>
>    celertek = 0;
>    for i=1:numel(x)
>      P = [x(i); y(i)];
>      celertek = celertek + pont_egy_tav(p, P)^2;
>    end
>  end
```

Pont-egyenes távolságot számító függvény:

```
>   function [t n Dy] = pont_egy_tav(p_egyenes, P_pont)
>    % Egyenes egyseghosszu iranyvektora
>    b = [1; p_egyenes(2)] / sqrt(1^2 + p_egyenes(2)^2);
>    % Egyenesre meroleges egyseghosszu vektor
>    n = [b(2); -b(1)];
>    % Az egyenes es a pont fuggoleges tavolsaga
>    Dy = P_pont(2) - (p_egyenes(1) + p_egyenes(2)*P_pont(1));
```

```
>    % Ennek vektor megfeleloje
>    Dyv = [0; Dy];
>    % Vetulete ez egyenes normalisara
>    t = Dyv'*n;
>    % Az egyseghosszu normalis vektor nyujtasa t hosszura
>    n = t * n;
>  end
```

A teljes legkisebb módszerek szerinti kiegyenlítés (kezdőérték a hagyományos legkisebb négyzetek módszerével), minimalizálás az fminunc függvénnyel.

```
>  clear all; clc; close all; %
>  page_screen_output(0);
>  global x;
>  global y;
>
>  p = [8.765; 1.234];
>  x = [1:10]';
>  y = p(1) + p(2)*x;
>
>  x = x + randn(numel(x),1);
>  y = y + randn(numel(y),1);
>
>  A = [ones(numel(x),1) x];
>  p1 = inv(A'*A)*A'*y;
>  p
>  p1
>
>  figure(1);
>  hold off;
>  plot(x, y, 'k+');
>  hold on;
>  plot(x, p1(1) + p1(2)*x, 'b', 'LineWidth', 2);
>  axis('equal');
>
>  for i=1:numel(x)
>    P = [x(i); y(i)];
>    [t n Dy] = pont_egy_tav(p1, P);
>  %  plot([P(1); P(1)], [P(2); P(2)-Dy], 'g');
>    plot([P(1); P(1)-n(1)], [P(2); P(2)-n(2)], 'b');
>  end
>
>  p2 = fminunc('celfuggveny', p1);
>
>  plot(x, p2(1) + p2(2)*x, 'r', 'LineWidth', 2);
>
>  celfuggveny(p1)
>  celfuggveny(p2)
>
>  for i=1:numel(x)
>    P = [x(i); y(i)];
>    [t n Dy] = pont_egy_tav(p2, P);
>    plot([P(1); P(1)-n(1)], [P(2); P(2)-n(2)], 'r');
>  end
```

MELLÉKLET

EGYSZERŰ AUTOCAD DXF FÁJL SZERKEZETE

FÁJL STRUKTÚRA

A DXF az AutoCAD egy sima ASCII szöveges állománya, meghatározott szerkezet szerint felépítve, ahol két egymást követő sor mindig összetartozik, az első sor egy típuskódot ad meg, a második az adott típushoz/elemhez tartozó értéket.

Alap felépítés:

A DXF fájl különböző szekciókból állhat, de ebből csak az ENTITIES kötelező, utána egy fájl vége jel van (EOF – end of file).

1. HEADER szekció – Fejléc. Általános információk, változók – opcionális
2. CLASSES szekció – opcionális
3. TABLES szekció – Különböző definíciók – opcionális
    a. Linetype table (LTYPE)
    b. Layer table (LAYER) – itt lehetne rétegeket definiálni, de új réteget az entitás szekción belül is létre lehet hozni, csak a réteg kóddal (8) és a réteg név megadásával, alap beállításokkal
    c. Text Style table (STYLE)
    d. View table (VIEW)
    e. User Coordinate System table (UCS)
    f. Viewport configuration table (VPORT)
    g. Dimension Style table (DIMSTYLE)
    h. Application Identification table (APPID)
4. BLOCKS szekció – Blokk definíciók – opcionális (ha nincs blokk a fájlban, akkor nem kell)
5. ENTITIES section – Rajzi elemek (pontok, vonalak stb.) – KÖTELEZŐ!
6. OBJECTS szekció – opcionális

Szekciók előtt meg kell adni, hogy most egy új szekció jön (0 kód, SECTION), majd utána a szekció nevét (2-es kód, név). Majd a tartalom után jön a szekció vége (0 kód, ENDSEC), legvégül fájl vége jel (0 kód, EOF). Pl.

```
0
SECTION
2
ENTITIES
…
0
ENDSEC
…
0
EOF
```

További kódokat lásd az 1. táblázatban a szöveg után, néhány gyakran használt kódot vastaggal kiemelve. Néhány rajzi elem leírását az ENTITIES szekción belül lásd a 2. táblázatban. Egyszerű példák DXF fájlra:

PONT MEGADÁSA DXF FÁJLBAN

A következő példa egy pont rajzol a 35.23, 12.89 koordinátákhoz.

Simple_point.dxf:

0	
SECTION	Szekció eleje
2	
ENTITIES	Rajzi elemek szekció követezik
0	
POINT	Pont típusú elem
8	
pontok	Rétegmegadás kódja
10	’pontok’ a réteg neve
35.23	X koordináta
20	
12.89	Y koordináta
0	
ENDSEC	Szekció vége
0	
EOF	Fájl vége

A fenti fájl egy pontot tesz le, de a pont stílusa az alap beállításként használt pici pötty, ha ezt meg akarjuk változtatni, akkor kell a HEADER szekció is, és be kell állítani a pont stílusát a PDMODE változóban. PDMODE értéke lehet pl. 2 (+), 3 (✕), 34 (⊕). Ehhez az előbbi előtt a HEADER szekciót is definiálni kell.

0	
SECTION	Szekció eleje
2	
HEADER	Fejléc szekció követezik
9	
$PDMODE	Változó definiálás jön
70	PDMODE változó
34	Változó értékadása (egész típusú)
0	értéke: 34 (⊕)
ENDSEC	Szekció vége

Ha különböző színekkel akarunk rajzolni, akkor a POINT, LINE vagy POLYLINE szekcióban is adhatunk meg színeket. A 62-es kód jelenti a színt, utána pedig az adott szín kódja pl az AutoCAD-ben: 1-piros, 2-sárga, 3-zöld, 4-világoskék, 5-sötétkék, 6-lila, 7-fekete/fehér, 8-szürke.

VONAL RAJZOLÁSA

A következő dxf fájl egy vonal a (0.0,0.0) kezdőpontból a (100,200) pontba.

```
0
SECTION
2
ENTITIES
0
LINE
8
vonal
10
0.0
20
0.0
11
100
21
200
0
ENDSEC
0
EOF
```

Szekció eleje

Rajzi elemek szekció követezik

Vonal típusú elem

Rétegmegadás kódja
’vonal’ a réteg neve
X koordináta (kezdőpont)
0.0
Y koordináta (kezdőpont)
0.0
X koordináta (végpont)
100
Y koordináta (végpont)
200
Szekció vége

Fájl vége

EGYSZERŰ SZÖVEG ÍRÁSA

írjuk ki a ’Haliho’ szöveget a (3,8) pontba, szöveg rétegbe, 1 egység magassággal.

```
0
SECTION
2
ENTITIES
0
TEXT
8
szöveg
10
3.0
20
8.0
40
1.0
1
Haliho!
0
ENDSEC
0
EOF
```

Szekció eleje

Rajzi elemek szekció követezik

Szöveg típusú elem

Rétegmegadás kódja
    ’szöveg’ a réteg neve
X koordináta
    3.0
Y koordináta
    8.0
Szöveg magassága
    1 rajzi egység
Szöveg tartalma:
    ’Haliho’
Szekció vége

Fájl vége

VONALLÁNC RAJZOLÁSA

Vonallánc a (0.0,0.0) kezdőpontból a (34.5,23.2) ponton át a (56.0,16.2) pontba.

0	Szekció eleje
SECTION	
2	Rajzi elemek szekció követezik
ENTITIES	
0	Vonallánc típusú elem
POLYLINE	
8	Rétegmegadás kódja
vonal	’vonal’ a réteg neve
66[5]	Csomópontok következnek
1	értéke mindig 1
0	Csomópont következik
VERTEX	
8[6]	Rétegmegadás kódja
vonal	’vonal’ a réteg neve
10	X koordináta (kezdőpont)
0.0	0.0
20	Y koordináta (kezdőpont)
0.0	0.0
0	Csomópont következik
VERTEX	
8	Rétegmegadás kódja
vonal	’vonal’ a réteg neve
10	X koordináta (kezdőpont)
34.5	34.5
20	Y koordináta (kezdőpont)
23.2	23.2
0	Csomópont következik
VERTEX	
8	Rétegmegadás kódja
vonal	’vonal’ a réteg neve
10	X koordináta (kezdőpont)
56.0	56.0
20	Y koordináta (kezdőpont)
16.2	16.2
0	Csomópontok vége
SEQEND	
0	Szekció vége
ENDSEC	
0	Fájl vége
EOF	

[5] A 66-os kód jelöli, hogy csomópontok következnek, ez nem minden CAD programnál kötelező elem, de a tanszéki AutoCAD MAP 2008 változatnál igen.

[6] Lásd mint előbb. Nem minden CAD programnál kötelező minden csomópontnál megadni a réteg nevét is, de az AutoCAD MAP 2008 esetében igen.

ÖSSZES RAJZI ELEM EGYÜTT

simple_dxf_example.dxf

```
0
SECTION
  2
HEADER
  9
$PDMODE
 70
34
  0
ENDSEC
  0
SECTION
  2
ENTITIES
  0
POINT
  8
pontok
 10
35.23
 20
12.89
  0
LINE
  8
vonal
 10
0.0
 20
0.0
 11
100
 21
200
  0
TEXT
  8
szöveg
 10
3.0
 20
8.0
 40
1.0
  1
Haliho!
  0
POLYLINE
  8
vonal
 66
1
  0
VERTEX
  8
vonal
 10
0.0
 20
0.0
  0
VERTEX
  8
vonal
 10
34.5
 20
23.2
  0
VERTEX
  8
vonal
 10
56.0
 20
16.2
  0
SEQEND
  0
ENDSEC
  0
EOF
```

DXF FÁJLBAN TALÁLHATÓ KÓDOK LISTÁJA

2. TÁBLÁZAT ÁLTALÁNOS KÓDOK JELENTÉSE A DXF FÁJLBAN[7]

0	**Text string indicating the entity type (fixed)**
1	The primary text value for an entity
2	**Name (attribute tag, block name, and so on)**
3-4	Other textual or name values
5	Entity handle. Text string of up to 16 hexadecimal digits (fixed)
6	Line type name (fixed)

[7] lásd részletesebben: http://www.autodesk.com/techpubs/autocad/acadr14/dxf/index.htm

7	Text style name (fixed)
8	**Layer name (fixed)**
9	**Variable name identifier (used only in HEADER section of the DXF file) (Example: $PDMODE – point style)**
10	**Primary point. This is the start point of a line or text entity, center of a circle, and so on. DXF: *X* value of the primary point (followed by *Y* and *Z* value codes 20 and 30)**
11-18	Other X coordinates
20	**Primary Y coordinate. 2n values always correspond to 1n values and immediately follow them in the file**
21-28	Other Y coordinates
30	**Primary Z coordinate. 3n values always correspond to 1n and 2n values and immediately follow them int he file**
31-37	Other Z coordinates
38	DXF: entity's elevation if nonzero.
39	Entity's thickness if nonzero (fixed)
40-48	Floating-point values (text height, scale factors, etc.)
49	Repeated floating-point value. Multiple 49 groups may appear in one entity for variable-length tables (such as the dash lengths in the LTYPE table). A 7*x* group always appears *before* the first 49 group to specify the table length.
50-58	Angles
62	**Color number (fixed)**
66	**"Entities follow" flag (fixed)**
67	Space--that is, model or paper space (fixed)
68	Identifies whether viewport is on but fully off screen, is not active, or is off
69	Viewport identification number
70-78	Integer values such as repeat counts, flag bits, or modes
210, 220, 230	X, Y, and Z components of extrusion direction (fixed)
999	**Comments**
etc.	

3. TÁBLÁZAT RAJZI ELEMEK (ENTITIES SECTION)[8]

LINE	10, 20, 30 (start point), 11, 21, 31 (end point).
POINT	10, 20, 30 (point), 50 (angle of X axis for the UCS in effect when the Point was drawn -optional 0, for use when PDMODE is nonzero). PDMODE:34
CIRCLE	10, 20, 30 (center), 40 (radius).
ARC	10, 20, 30 (center), 40 (radius), 50 (start angle), 51 (end angle).
POLYLINE	66 ("vertices follow flag"), 70 (Polyline flags), 40 (default starting width), 41 (default ending width), 71 and 72 (poly- gon mesh M and N vertex counts -optional 0), 73 and 74 (smooth surface M and N densities -optional 0), 75 (smooth surface type -optional 0). The default widths apply to any vertex that doesn't supply widths (see below). The "vertices follow" flag is always 1, indicating that a series of VERTEX entities is expected to follow the POLYLINE, terminated by a sequence end (SEQEND) entity. The "polyline flags" group is a bit-coded field with bits defined as follows: Flag bit value Meaning 1 This is a closed Polyline (or a polygon mesh closed in the M direction) 2 Curve-fit vertices have been added 4 Spline-fit vertices have been added 8 This is a 3D Polyline 16 This is a 3D polygon mesh. Group 75 indicates the smooth surface type, as follows:

[8] lásd részletesebben: http://www.autodesk.com/techpubs/autocad/acadr14/dxf/index.htm

	0 = no smooth surface fitted 5 = quadratic B-spline surface 6 = cubic B-spline surface 8 = Bezier surface 32 The polygon mesh is closed in the N direction
VERTEX	10, 20, 30 (location), 40 (starting width -optional, see above), 41 (ending width -optional, see above), 42 (bulge), 70 (vertex flags), 50 (curve fit tangent direction -optional). The bulge is the tangent of 1/4 the included angle for an arc segment, made negative if the arc goes clockwise from the start point to the end point; a bulge of 0 indicates a straight segment, and a bulge of 1 is a semicircle. The meanings of the bit-coded "vertex flags" are shown in the following table. Flag bit value Meaning 1 Extra vertex created by curve fitting 2 Curve fit tangent defined for this vertex. A curve fit tangent direction of 0 may be omitted from the DXF output, but is significant if this bit is set. 4 Unused (never set in DXF files) 8 Spline vertex created by spline fitting 16 Spline frame control point 32 3D Polyline vertex 64 3D polygon mesh vertex
SEQUEND	No fields. This entity marks the end of vertices (VERTEX type name) for a Polyline, or the end of Attribute entities (ATTRIB type name) for an INSERT entity that has Attributes (indicated by 66 group present and nonzero in INSERT entity).
TEXT	10, 20, 30 (insertion point), 40 (height), 1 (text value), 50 (rotation angle -optional 0), 41 (relative X scale factor -optional 1), 51 (obliquing angle -optional 0), 7 (text style name -optional "STANDARD"), 71 (text generation flags -optional 0), 72 (justification type -optional 0), 11, 21, 31 (alignment point -optional, appears only if 72 group is resent and nonzero). The "text generation flags" are a bit-coded field with meanings as follows: Flag bit value Meaning 2 Text is backwards (mirrored in X) 4 Text is upside down (mirrored in Y) The "justification type" value (not bit-coded) indicates the text justification style used on this entity, as shown in the following table. Value Meaning 0 Text is left justified 1 Text is centered along its baseline 2 Text is right justified 3 Text is aligned between two points (height varies) 4 Text is "middle" (fully) centered 5 Text is fit between two points (width varies) If the justification is anything other than 0 (left justified), 11, 21, and 31 groups will also appear in the entity to specify the alignment point of the text (center, right-most, or second alignment point). DXFOUT handles ASCII control characters in text strings by expanding the character into a "^" (caret) followed by the appropriate letter. For example, an ASCII Control-G (BEL, decimal code 7) is output as "^G". If the text itself contains a caret character, it is expanded to "^ " (caret, space). DXFIN performs the complementary conversion.