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.

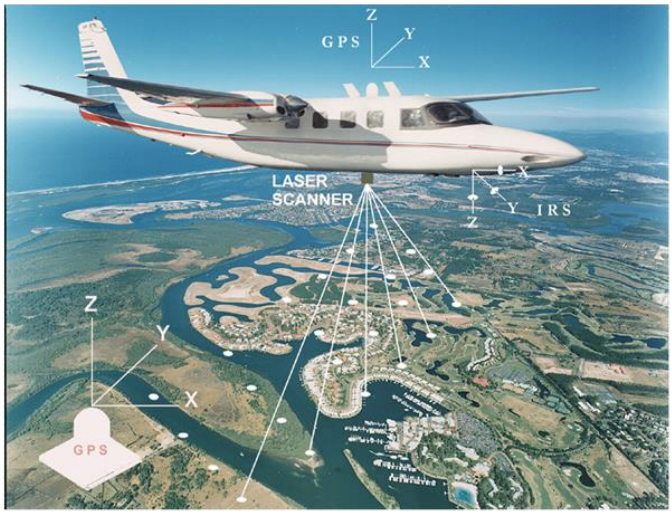
```
> M=[];i=0; % M tömb letrehozasa, i - sorok száma
> while ~feof(fid)
>    line=fgetl(fid); % egy sor beolvasása
>    i=i+1; % sorok szamat noveljuk eggyel
>    [a n]=sscanf(line,'%f'); % a – számok, n - darabszám
>    M(i,1:n)=a;
> end
> fclose(fid);
> M
```

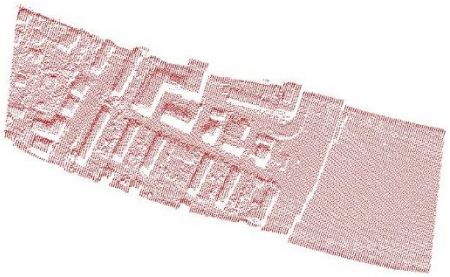
Eredmények:

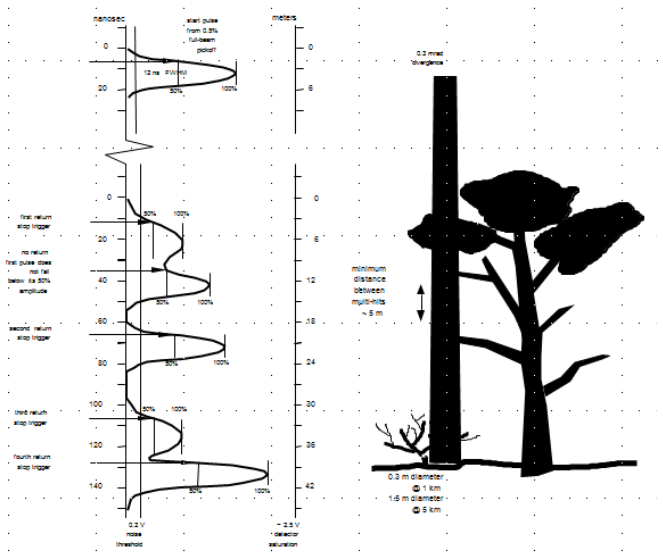
```
M =

     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
```

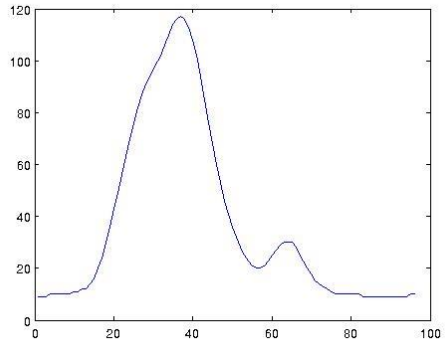
TELJES HULLÁMALAKOS LÉZERSZKENNER ADATOK BEOLVASÁSA, MEGJELENÍTÉSE

A teljes hullámalakos lézerszkennerek mérési eredményei, a lézerszkenner által kibocsátott és a felszínről visszaverődött jelek, szintén különböző hosszúságúak lehetnek.









Erdős területről visszaverődött jel

Olvassuk be a waveform_sample állományt az előbb megírt programmal!

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

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
```

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'); % kép mentése
>   % print(filenev, '-S640,480','-djpeg');
>   % 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);
```