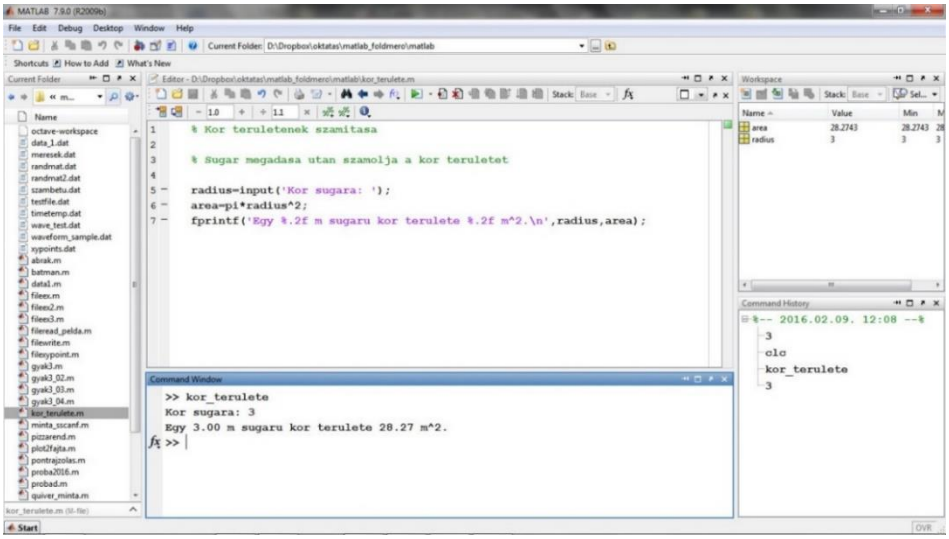
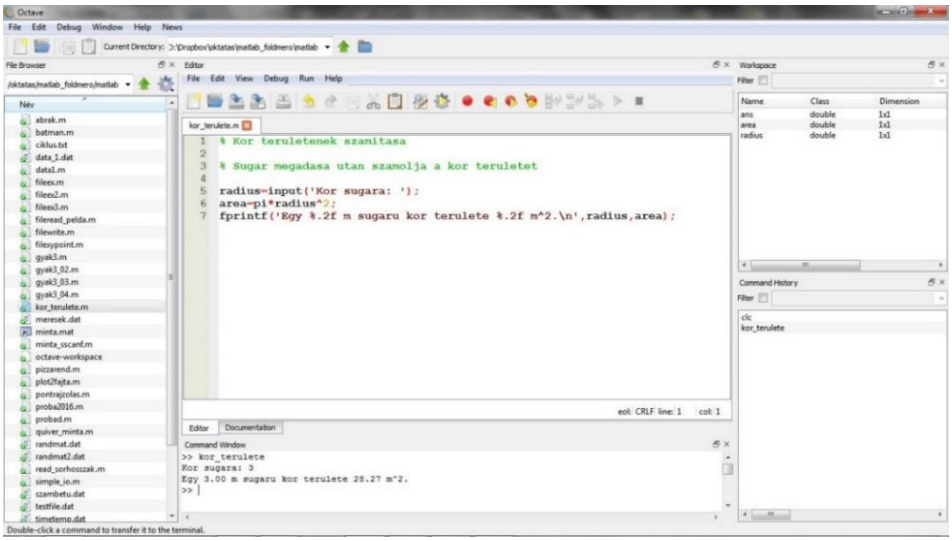
# MATLAB/OCTAVE/ ALAPOZÓ[1]

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). Az Octave a https://www.gnu.org/software/octave/ oldalról tölthető le, jelenleg a 4.2.0 változat (2016. novembere óta). 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).




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, January 20, 2017, http://www.math.ohiou.edu/courses/math3600/book.pdf

## MATLAB/OCTAVE MUNKAKÖRNYEZET, ALAPOK

A grafikus felület fontosabb részei: az éppen aktuális könyvtár, ahová mindent ment a program (**current folder/directory**), 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.

A későbbiekben amire mindig figyeljünk, hogy a használt fájl nevek nem kezdődhetnek számmal és lehetőleg ne legyenek benne ékezetes karakterek 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)

`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 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

Célszerű kerülni az ékezeteket, Octave-ban es Matlab-ban azok ugyanis nem kompatibilisek, az egyikkel létrehozott ékezetes betűk a másikban rosszul jelennek meg.

### ÉRTÉKADÁS, VÁLTOZÓ TÍPUSOK

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

A Matlab/Octave alap objektumai a mátrixok. A vektorok speciális mátrixok, pl. 1xn-es sorvektorok, vagy mx1-es oszlopvektorok. Mátrix/vektor definiálásához szögletes zárójelet használunk.

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

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

```
> t(2) % eredménye: 4
> 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:

```
> t(2:4) % eredménye az előző parancs után: 47 22 66
```

Lehet sorvektorból oszlopvektor készíteni és fordítva

```
> to = t' % t transzponáltja, oszlopvektor
```

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 = linspace(1,10,10) % % 1 és 10 között 10 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
> XY = [X Y] % 2x7 elemű mátrix, vízszintesen összefűzve X és Y
> XZ = [X; Z] % 5x3-as mátrix, függőlegesen összefűzve X, Z
> XY(1,:) % XY első sora (: az adott sorban az összes elem)
> XZ(:,2) % XZ első 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)
```

## EGYSZERŰ PLOTTOLÁS

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

| ε [%] | 0 | 0.5 | 5 | 23 | 27 |
|---|---|---|---|---|---|
| σ [N/mm$^2$=Mpa] | 0 | 215 | 225 | 380 | 350 |

1. TÁBLÁZAT BETONACÉL FESZÜLTSÉG-FAJLAGOS ALAKVÁLTOZÁS DIAGRAMJA

Írjuk be a parancssorba:

```
> x = [0 0.5 5 23 27]
> y = [0 215 225 380 350]
```

Ha beírjuk a parancssorba egy létező változó nevét, akkor kiírja az aktuális tartalmát.

```
> 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*-')
```

## FÜGGVÉNYEK HASZNÁLATA

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.

```
> sin(pi) % értéke 0 a számábrázolás pontosságán belül
> 3^4 % értéke: 81
> 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 FÜGGVÉNYEK

Többféleképp lehet Matlab-ban saját függvényeket megadni, az egyszerűbb függvények leggyakoribb módja az anonim függvény használata (anonymous function), pl adjuk meg az $f(x) = 2x^2 - 3x + 1$ függvényt:

```
> f = @(x) 2*x.^2 - 3*x + 1
```

Ellenőrizzük le egy adott x értékére:

```
> f(1.2345) % értéke: 0.3445
```

A függvényt úgy adtuk meg, hogy nem csak egyedüli értékekre, hanem vektorokra is működik, pl.

```
> x = -2:.2:2
> y = f(x)
> plot(x,y)
```

Azért működött a függvényünk vektorokra is, mert a $x^2$ értékét x.^2 módon adtuk meg. Itt szerepel egy plusz pont a hatvány művelet előtt. Így kell megadni a szorzást, osztást és hatványozást is, ha vektor bemeneten is használni szeretnénk a függvényt: .*,./,.^. Összeadásnál nem kell megadni a plusz pontot, se skalárral való osztás/szorzáskor. A Matlabnak az egyik erőssége, hogy egy vektorba téve sok értéket, a Matlab egyszerre tud az összesen műveleteket végezni, és nem egyesével.

## MATLAB PROGRAMOK

Matlab programokat *.m kiterjesztésű szöveges fájlokba írhatunk. Kétféle típust fogunk használni: script fájlt és függvényt (function).

## FÜGGVÉNY ÍRÁSA

Írjuk meg külön függvény fájlba az előbb megadott függvényt! Kattintsunk a bal felső sarokban a plusz jelre (new), és megnyílik az Editorban egy üres lap. írjuk be a következőket, majd mentsük el probafv.m néven az aktuális könyvtárba.

```
> function y = probafv(x)
>     y = 2*x.^2 - 3*x + 1;
> end
```

Írjuk be utána a parancssorba:

```
> x1 = -2:.1:2;
> y1 = probafv(x1);
> plot(x1,y1)
```

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 nem fognak megjelenni a workspace-ben, és a függvény sem fér hozzá a workspace-ben lévő változókhoz, csak, amit megadtunk a bemenetnél.

Egy függvénynek lehet több bemenete is, pl módosítsuk az előző függvényünket az alábbi módon:

```
> function y = probafv2(x,p)
>     y = 2*x.^p - 3*x + 1;
> end
```

Majd mentsük el probafv2.m néven és hívjuk meg:

```
> y1 = probafv2(x1,3);
> plot(x1,y1)
```

Egy függvénynek lehet több kimenete is egy vektorban összegyűjtve:

```
> function [x2 x3 x4] = hatvany(x)
>     x2 = x.^2;
```

```
>     x3 = x.^3;
>     x4 = x.^4;
> end
```

Mentsük el a fenti fájlt hatvany.m néven, és készítsünk néhány ábrát vele.

```
> x = -1:.1:1
> [x2 x3 x4] = hatvany(x);
> plot(x,x,x,x2,x,x3,x,x4)
```

Mi is adhatunk egyéni színeket hozzá:

```
> plot(x,x,'black',x,x2,'blue',x,x3,'green',x,x4,'red')
```

Vagy jelmagyarázatot is fűzhetünk hozzá

```
> legend('x','x^2','x^3','x^4','Location','Best')
```

## SCRIPT ÍRÁSA

A másik általunk használt program típus, a script fájl, ami több dologban eltér a függvénytől:

- Nincsenek bemenetek és kimenetek.
- A scripttel létre tudunk hozni változókat a workspace-ben, ezeket utána módosíthatjuk, törölhetjük.

A következő script ugyanazt csinálja, mint az előző függvény,

```
> x2 = x.^2;
> x3 = x.^3;
> x4 = x.^4;
> plot(x,x, 'black ',x,x2, 'blue ',x,x3, 'green ',x,x4, 'red ')
```

Mentsük el grafikonok.m néven az aktuális könyvtárba. A **clear** paranccsal töröljünk minden korábbi változót a workspace-ből, majd adjuk meg:

```
> x = -1:.1:1;
> grafikonok
```

Vegyük észre, hogy a program még úgyis használta az x változó értékét, hogy az a parancssorban lett megadva, nem a script fájlban. Általában a parancssori gépelés helyett script fájlokba szoktunk dolgozni, főleg, ha egysorosnál hosszabb számításokat végzünk. Könnyebb utólag javítani. A script fájlokat futtathatjuk a nevük begépelésével, de egyszerűbb az **F5** megnyomásával, ez rögtön menti és futtatja a programot. Ha nem akarjuk az egész programot lefuttatni betehetünk egy **return** parancsot valahová, és akkor csak addig fog lefutni a program. Illetve az **F9** lenyomásával a kijelölt rész futtatható.

## 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 szükséges megadni, hogy mi a függvény célja, milyen bemeneti és kimeneti értékekek szerepelnek benne. Írjunk megfelelő kommenteket egyik előző függvényünkhöz, majd mentsük el!

```
> function y = probafv(x)
> % Kiszámítja a 2*x^2 - 3*x + 1 függvényértéket.
> % Bemenet: x – egy szám vagy vektor
> % Kimenet: y – szám vagy vektor (x-szel egyező méretű)
>     y = 2*x.^2 - 3*x + 1;
> end
```

A Matlab help parancsa kiírja a képernyőre a fájl elején található kommenteket.

```
> help probafv
```

Kiírja a képernyőre a fent beírt kommenteket!

## MATLAB/OCTAVE HIBAÜZENETEK

A programok írása során számos hibaüzenettel találkozunk. 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 'cler' for input arguments of type 'char'.
error: 'cler' undefined near line 1 column 1
```

A Matlab/Octave 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 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
```