MATLAB GRAFIKA

Az első fejeztben már megismerkedtünk a Matlab alapjaival, függvények, programozási szerkezetek használatával, egyszerűbb fájlkezelési műveletekkel, illetve alap grafikai megjelenítéseket is végeztünk egyváltozós esetekben. A mérnöki gyakorlatban kiemelt szerepe van a megjelenítéseknek. A mérési, számítási eredményeinket sok esetben ábrákkal tudjuk a legjobban szemléltetni, mások számára is érthetővé tenni, ezért ebben a fejezetben részletesebben fogjuk tárgyalni a különböző megjelenítési lehetőségeket, illetve összefoglaljuk a korábban használt egyváltozós megjelenítési lehetőséget is.

Az építőmérnöki gyakorlatban sokféle ábrázolás szükséges. Gyakran használunk egyszerűbb kétdimenziós ábrákat, mint például a szakítószilárdság vizsgálatának grafikonja, vagy az igénybevételi ábrák. Szükség van háromdimenziós felületek ábrázolására is felmért domborzatok, folyómeder geometria, vagy végeselemes modellezés esetében a feszültségek megjelenítése érdekében. Vektormezők ábrázolása is szükséges lehet, például áramlási sebességmezők meghatározása esetében egy műtárgy körül, vagy folyókanyarban. A következőkben áttekintjük a két- és háromdimenziós ábrázolásokat és azokat a lehetőségeket is, amikor több dimenzió megjelenítése is szükséges, például hőmérséklet értékek vagy sebességek megjelenítése áramlatokban térben.

KÉTDIMENZIÓS MEGJELENÍTÉSEK

A kétdimenziós ábrázolásról már volt szó a korábbiakban is, itt most részletesebben nézzük meg a lehetőségeinket. Kétdimenziós megjelenítéskor szükség lehet adatsorok megjelenítésére, pl. mérési eredmények ábrázolása, illetve valamilyen matematikai formulával, illetve függvénnyel leírható kapcsolat ábrázolására.

ADATSOROK MEGJELENÍTÉSE, FORMÁZÁSA

Nézzük át egy vízépítő mérnöki példán keresztül az adatsorok megjelenítését. A víztározók egyik alapadata a tározó morfológiai jelleggörbéje, ami megadja, hogy egy adott vízálláshoz mekkora víztérfogat és felület tartozik. Adottak a következő adatok, amelyek a tarozas.txt fájlban is megtalálhatóak:

Vízállás H [cm]	Térfogat V [10^6 m^3]	Felület F [km^2]
336	0.16	0.05
504	0.36	0.09
714	0.79	0.19
976	1.73	0.37
1302	3.31	0.62
1628	5.83	0.90
1812	7.72	1.05
1932	8.98	1.16
2142	11.50	1.27

Ábrázoljuk az adatainkat egy ábrában jelmagyarázattal kiegészítve.

Először olvassuk be az adatokat és válasszuk szét a változókat, majd jelenítsük meg egy ábrában a vízállás függvényében a tározó térfogatát és ugyancsak a vízállás függvényében a tározó felületét is.

```
> % Kétdimenziós ábrázolás
> clc; clear all; close all;
> % Adatsorok beolvasása
> format shortG
> tarozo = load('tarozas.txt')
> H = tarozo(:,1); % Vízállás [cm]
> V = tarozo(:,2); % Térfogat, millió köbméter
> F = tarozo(:,3); % Felület négyzet km
> % Adatsorok ábrázolása
> figure(1);
```

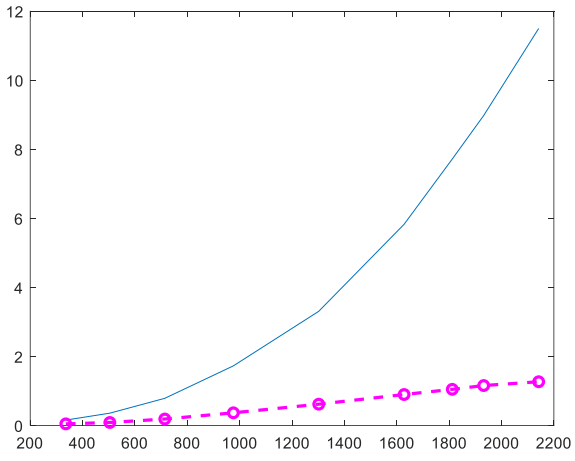
Amennyiben nem adunk meg számot a **figure** parancs után zárójelben, akkor egy új, következő ábrát nyit automatikusan a Matlab, viszont ha később szeretnénk erre az ábrába visszatérni egy másik után, akkor célszerű számozni.

A legegyszerűbb a két ábra együttes megjelenítésére, ha egy **plot** parancsban adjuk meg őket.

```
> plot(H,V,H,F)
```

Jobban paraméterezhető azonban a görbék megjelenítése, ha két külön plot parancsot használunk. Ehhez szükséges a **hold on** parancs használata is, e nélkül az új grafikon rajzolása kitörölné az előzőt. A **hold off** parancs kiadásával visszaállítható az alapértelmezett állapot. Egy ábra esetében elég egyszer kiadni a **hold on** parancsot.

```
> plot(H,V);
> hold on;
> h2 = plot(H,F,'om--','LineWidth',2);
```



A **plot** parancsot meghívhatjuk egyszerűen megadva a vektorokban tárolt összefüggő adatokat. Ebben az esetben egy alapértelmezett színű, folytonos vonallal fogja összekötni a pontokat. A **plot** parancs meghívható különböző opcionális paraméterekkel is. Megadhatjuk, hogy milyen szimbólumokat tegyen a pontokra (pl. *,+,o,x,d,s), amennyiben megadunk szimbólumot, vonalstílus nélkül, akkor nem köti össze a pontokat. Megadhatjuk, hogy milyen vonalstílust alkalmazzon (– folytonos, - -

szaggatott, : pontozott), milyen színnel jelenítse meg a vonalat (m – lila, b – kék, r – piros, k – fekete stb., az angol megfelelőik első vagy utolsó betűje alapján). Ezeket a tulajdonságokat egyben is megadhatjuk egyszeres idézőjelek között. További tulajdonságokat is beállíthatunk (pl. vonalvastagság, pont szimbólum kitöltése), külön megadva idézőjelben a tulajdonság nevét, majd vesszővel elválasztva az értékét. Összefoglalva néhány beállítható tulajdonság megtalálható az alábbi táblázatokban.

Hasznos paraméterek:

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

A második grafikonnál nem csak az alapértelmezett tulajdonságokat állítottuk át, hanem el is neveztük h2 néven. Bármilyen megjelenítési parancs eredményét elmenthetjük változókba és utána tudjuk módosítani a tulajdonságait, vagy akár törölhetjük is őket az ábrából.

Kérdezzük le a h2 elem tulajdonságait! Ehhez a **get** parancs használható:

```
> get(h2)
```

Eredménye:

```
    AlignVertexCenters: off
           Annotation: [1×1 matlab.graphics.eventdata.Annotation]
         BeingDeleted: off
           BusyAction: 'queue'
        ButtonDownFcn: ''
             Children: [0×0 GraphicsPlaceholder]
             Clipping: on
```

```
                Color: [1 0 1]
            ColorMode: 'manual'
          ContextMenu: [0×0 GraphicsPlaceholder]
            CreateFcn: ''
       DataTipTemplate: [1×1 matlab.graphics.datatip.DataTipTemplate]
            DeleteFcn: ''
          DisplayName: ''
     HandleVisibility: 'on'
              HitTest: on
        Interruptible: on
             LineJoin: 'round'
            LineStyle: '--'
        LineStyleMode: 'manual'
            LineWidth: 2
               Marker: 'o'
       MarkerEdgeColor: 'auto'
       MarkerFaceColor: 'none'
         MarkerIndices: [1 2 3 4 5 6 7 8 9]
            MarkerMode: 'manual'
            MarkerSize: 6
                Parent: [1×1 Axes]
         PickableParts: 'visible'
              Selected: off
    SelectionHighlight: on
           SeriesIndex: 2
                   Tag: ''
                  Type: 'line'
              UserData: []
               Visible: on
                 XData: [336 504 714 976 1302 1628 1812 1932 2142]
            XDataMode: 'manual'
          XDataSource: ''
                 YData: [0.05 0.09 0.19 0.37 0.62 0.9 1.05 1.16 1.27]
          YDataSource: ''
                 ZData: [1×0 double]
          ZDataSource: ''
```

Látjuk, hogy sokféle tulajdonsága van egy egyszerű vonalas objektumnak is. Ha bármit módosítani szeretnénk rajta, akkor ehhez a **set** parancsot használhatjuk, megadva a grafikus objektum nevét, majd páronként a módosítani kívánt tulajdonságot és a hozzá tartozó új értéket. Ugyanez működik természetesen nem csak vonalas rajzokkal, hanem felületekkel, szintvonalas ábrával illetve bármilyen egyéb grafikus elemmel is.

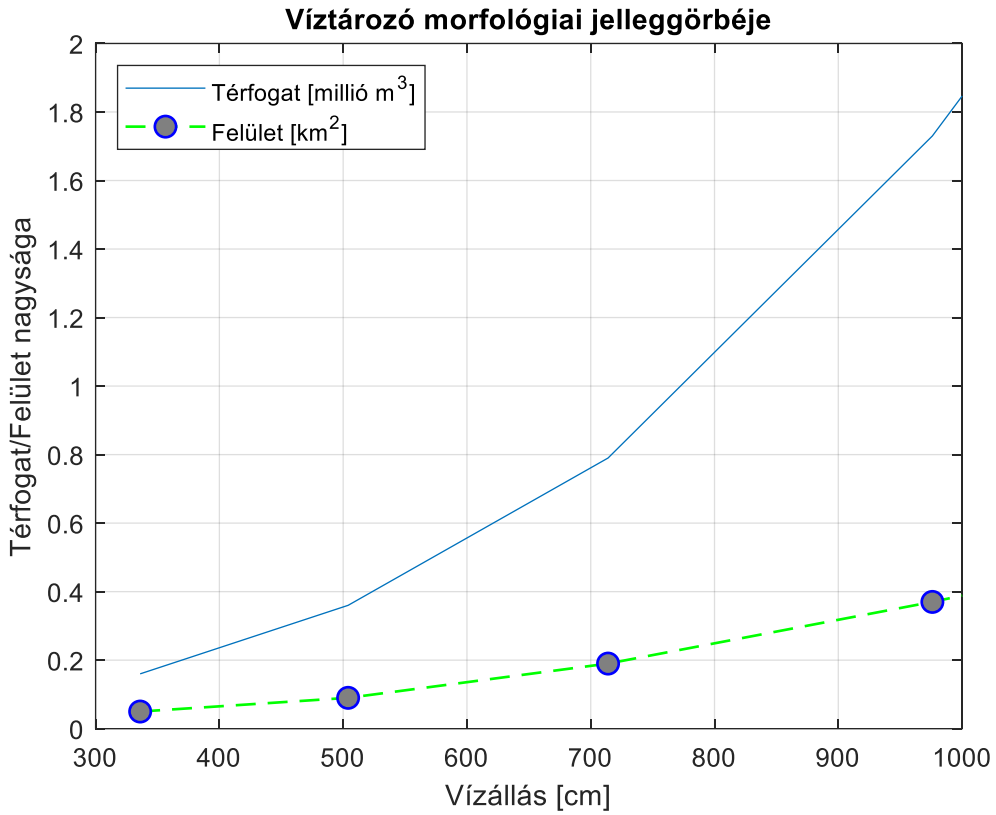
Használata:

```
> % Tulajdonságok módosítása
> set(h2,'Color','g','LineWidth',1,'MarkerSize',8,...
>     'MarkerEdgeColor','b','MarkerFaceColor', [0.5,0.5,0.5]);
```

Elnevezhetjük a tengelyeket (**xlabel**, **ylabe**l parancs), címet (**title**), jelmagyarázatot (**legend**) is fűzhetünk hozzá, illetve bekapcsolhatjuk a koordinátahálót is, hogy könnyebb legyen leolvasni az értékeket (**grid on** parancs). A tengelyek maximális és minimális értékeit is beállíthatjuk az **axis([xmin,xmax,ymin,ymax])** paranccsal, ránagyítva ezzel a megadott területre.

```
> % Címek, jelmagyarázat, koordinátaháló, határok beállítása
> xlabel('Vízállás [cm]')
> ylabel('Térfogat/Felület nagysága')
> title('Víztározó morfológiai jelleggörbéje')
> grid on;
> axis([300,1000,0,2])
> legend('Térfogat [millió m^3]',...
```

```
> 'Felület [km^2]','Location','NW')
```



A szövegekben használhatunk **alsó indexet**, pl. **x_1**, vagy **felső indexet** pl. **m^3**, illetve **görög betűk**et is egy fordított törtvonal (backslash) után a görög betű nevét megadva, pl. **\delta** eredménye: δ.

A jelmagyarázat után olyan sorrendben kell megadni a grafikonokhoz tartozó szövegeket, amilyen sorrendben megjelenítettük őket. Opcionális paraméterként megadhatjuk a jelmagyarázat helyét is ('Location'), földrajzi égtájakat használva, pl. 'NorthWest', vagy 'NW', stb. illetve a 'best' paramétert megadva a Matlab megpróbálja kitalálni a jelmagyarázat ideális helyét, ahol a legkevesebb ábrát takarja ki (ez utóbbi Octave esetében nem használható). Lehetőség van csak bizonyos grafikonok feliratozására is, amennyiben elneveztük őket (több feliratozandó elem esetében egy vektorban kell megadni az összeset, pl. [h1,h2]). Próbáljuk ki a következőt.

```
> legend(h2,'Felület [km^2]')
```

Az elnevezett grafikonokat törölhetjük is az ábránkból a **delete** paranccsal.

```
> delete(h2)
```

Az egész aktuális ábra tartalmát törölhetjük a **clf** (clear figure) paranccsal.

```
> clf
```

Végül a **close** paranccsal bezárhatjuk az aktuális ábrát, a **close all** paranccsal pedig az összeset.

```
> close
```

EGYVÁLTOZÓS, EXPLICIT FÜGGVÉNYEK MEGJELENÍTÉSE

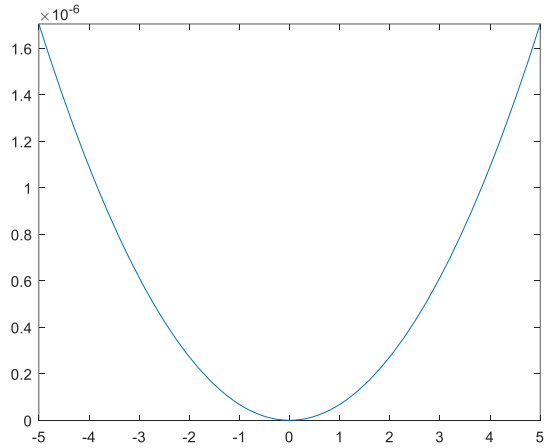
Az előző részben a vektorokban tárolt összetartozó adatsorok megjelenítésével foglalkoztunk, amihez a **plot** parancsot használtuk. Most áttérünk a matematikai formulával leírható függvények, görbék megjelenítésére. Ezek megjelenítésének egyik módja lehetne, hogy viszonylag sűrűn felveszünk pontokat, kiszámoljuk a felvett pontokban a függvény értékeket és utána ezeket jelenítjük meg **plot** paranccsal. A Matlab/Octave azonban kínál egyszerűbb megjelenítési lehetőséget is a számunkra az **ezplot** illetve az **fplot**, **fimplicit** parancsok formájában. A korábbi Matlab verziókban és az Octave-ban az explicit és implicit alakban megadott függvények, görbék megadására az **ezplot** parancs használható. Az újabb Matlab verziókban már nem támogatott az **ezplot** használata, hanem helyette az **fplot** és **fimplicit** parancsok használata jött be, de az R2020a verzióban még mindkettő megtalálható.

Nézzünk meg először egy egyszerűbb, trigonometriai magasságmérésben használt függvényt, amivel a földgörbület és refrakció hatása írható le a távolság függvényében.

$$\Delta = (1-k)\cdot\frac{d^2}{2\cdot R}$$

A képletben k a refrakciós együttható, értéke +0.13, R a Föld közelítő sugara, értéke 6378 km, d pedig a vízszintes távolság. A trigonometriai magassági számításokban ezt a tagot többnyire elhanyagoljuk, amennyiben a hatása nem éri el az 1 cm-t. Nézzük meg egy ábra segítségével, hogy körülbelül mekkora az a távolság ahol nem kell foglalkoznunk ezzel a hatással. Ez a képlet egy egyszerű egysoros függvénnyel (anonymous function) leírható.

```
> % Függvények megjelenítése
> R = 6378000; k = 0.13;
> delta = @(d) (1-k)*d.^2/(2*R)
> figure(3)
> fplot(delta) % vagy ezplot(delta)
```



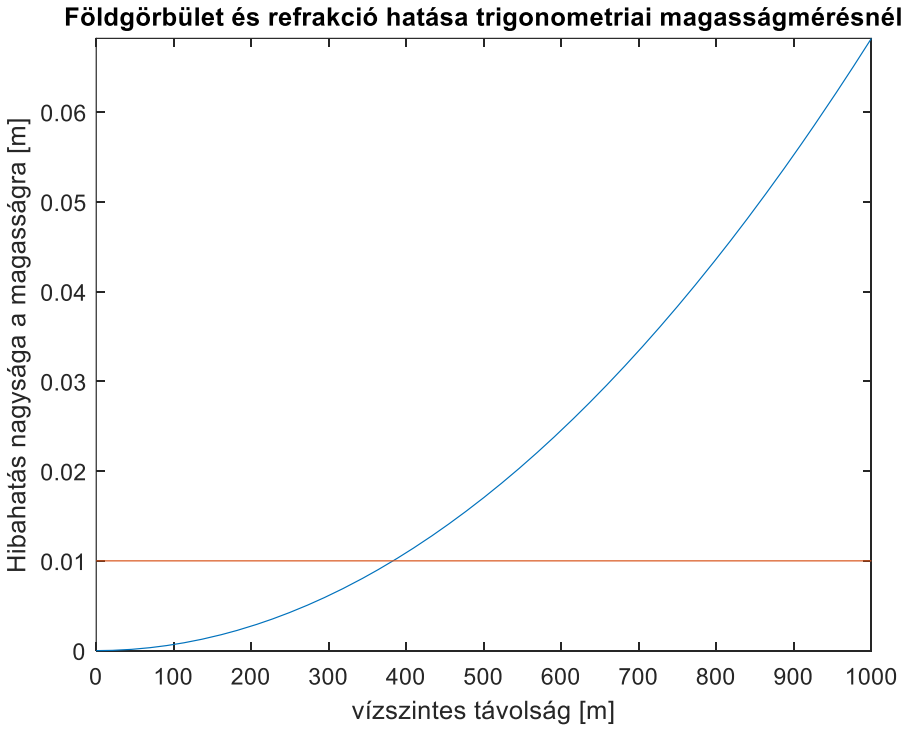
A fenti az **fplot** vagy **ezplot** parancs legegyszerűbb meghívása, ami egy alapértelmezett tartományon jeleníti meg a függvényt. A mi céljainkra azonban most nagyon nem felel meg az alapértelmezett tartomány. Egyrészt negatív távolságokat nem mérünk, másrészt itt a maximális y érték $\sim 1.6\cdot 10^{-6}$, vagyis ezred milliméter nagyságrendű, minket pedig az érdekel, hogy mekkora távolságnál haladja meg ez az érték a centimétert, vagyis 10^{-2} métert. Ezen segíthetünk, ha a függvény

megadásakor megadjuk az értelmezési tartományt is, legyen ez most ebben egy reálisan megmérhető távolság, mondjuk 0 és 1000 m közötti érték. Rajzoljunk be egy vízszintes vonalat is a 0.01 m-es értékhez, hogy lássuk, hogy nagyságrendileg hol metszi a függvényt!

```
> fplot(delta,[0,1000]) % vagy ezplot(delta,[0,1000])
> hold on;
> plot(xlim,[0.01,0.01]) % vagy plot([0,1000],[0.01,0.01])
> title('Földgörbület és refrakció hatása trigonometriai
  magasságmérésnél')
> xlabel('vízszintes távolság [m]')
> ylabel('Hibahatás nagysága a magasságra [m]')
```

A vízszintes vonal berajzolásakor használhatjuk a **plot** parancsot (illetve hasonlóképp meghívva a **line** parancsot is), megadva az egyenes két végpontjának x és y koordinátáit egy-egy vektorban. Mivel az egyenest az ábra elejétől a végéig be akarjuk rajzolni, itt az értelmezési tartomány újbóli megadása helyett használhatjuk az **xlim** parancsot is, ami a megnyitott ábra x tengelyének határait jelenti.

A lenti ábrán láthatjuk, hogy nagyjából 400 m távolság az, ahol a földgörbület és refrakció hatása eléri az 1 centimétert, ennél kisebb távolságok esetében nyugodtan elhanyagolhatjuk a trigonometriai magasságmérés képletéből ezt a tagot, utána viszont meredeken nő az érték, mivel négyzetesen arányos a hatás a távolsággal.



Az **fplot** parancs esetében ugyanazokat a formázási tulajdonságokat használhatjuk, mint amiket a **plot** esetében is használtunk, pl.:

```
> fplot(delta,[0,1000],'g--','LineWidth',2)
```

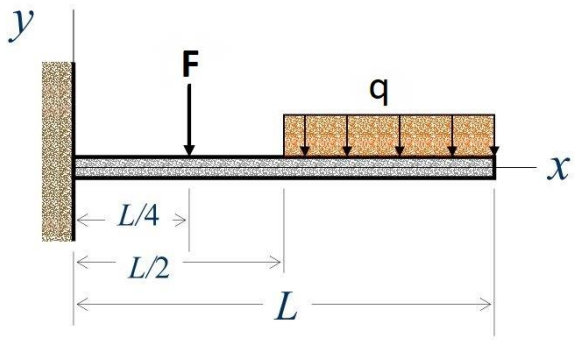
Az **ezplot** esetében a **set** paranccsal változtathatjuk meg a grafikon tulajdonságait:

```
> h3 = ezplot(delta,[0,1000])
> set(h3,'Color','g','LineStyle','--','LineWidth',2)
```

TÖBB GRAFIKON MEGJELENÍTÉSE EGYMÁS ALATT/MELLETT

Nézzünk most egy olyan ábrát, ahol jellemzően több egymással összefüggő ábrát szeretnénk egymás alatt megjeleníteni. Ilyenek például az igénybevételi ábrák mechanikából.

Rajzoljuk fel egy L hosszúságú konzol igénybevételi ábráit, ahol L/4 helyen F erő hat és L/2-től a konzol végéig q megoszló terhelés. Ábrázoljuk egymás alatt a nyíróerő (T) ábrát és a nyomatéki (M) ábrát!



$L = 8\ m, F = 10\ kN, q = 1\ kN/m$

Ebben az esetben a nyíróerő és a nyomaték függvénye nem fejezhető ki egyetlen analitikus függvénnyel, hanem három intervallumra kell külön-külön felírnunk. A nyíróerőre (T) és a nyomatékra (M) a következő függvényeket kapjuk:

$$T(x) = \begin{cases} F + \frac{q \cdot L}{2}, & 0 \le x \le L/4 \\ \frac{q \cdot L}{2}, & L/4 \le x \le L/2 \\ \frac{q \cdot L}{2} - q \cdot \left(x - \frac{L}{2}\right), & L/2 \le x \le L \end{cases}$$

$$M(x) = \begin{cases} -\frac{F \cdot L}{4} - \frac{3 \cdot q \cdot L^2}{8} + F \cdot x + \frac{q \cdot L}{2} \cdot x, & 0 \le x \le L/4 \\ -\frac{3 \cdot q \cdot L^2}{8} + \frac{q \cdot L}{2} \cdot x, & L/4 \le x \le L/2 \\ -\frac{3 \cdot q \cdot L^2}{8} + \frac{q \cdot L}{2} \cdot x - \frac{q}{2} \cdot \left(x - \frac{L}{2}\right)^2, & L/2 \le x \le L \end{cases}$$

Ezt az összefüggést már nem tudjuk Matlab-ban egysoros függvényben megírni, hanem egy külön függvényt használhatunk egy bemenettel (x) és több kimenettel (T,M). Nyissuk meg Matlab-ban a konzol.m függvényt! Ebben 3 feltétellel, logikai indexeléssel van megoldva, hogy a 3 különböző szakaszon 3 különböző függvényt számoljunk ki. Itt 3 kimeneti érték van, az összetartozó x,T(x),M(x) értékekekel. A kimenetek között azért szerepel x is, ami a bemenet is, mivel a szakasz határokon két különböző értéket is fel kell vegyen T és M, így ezek duplán szerepelenek a kimenetben.

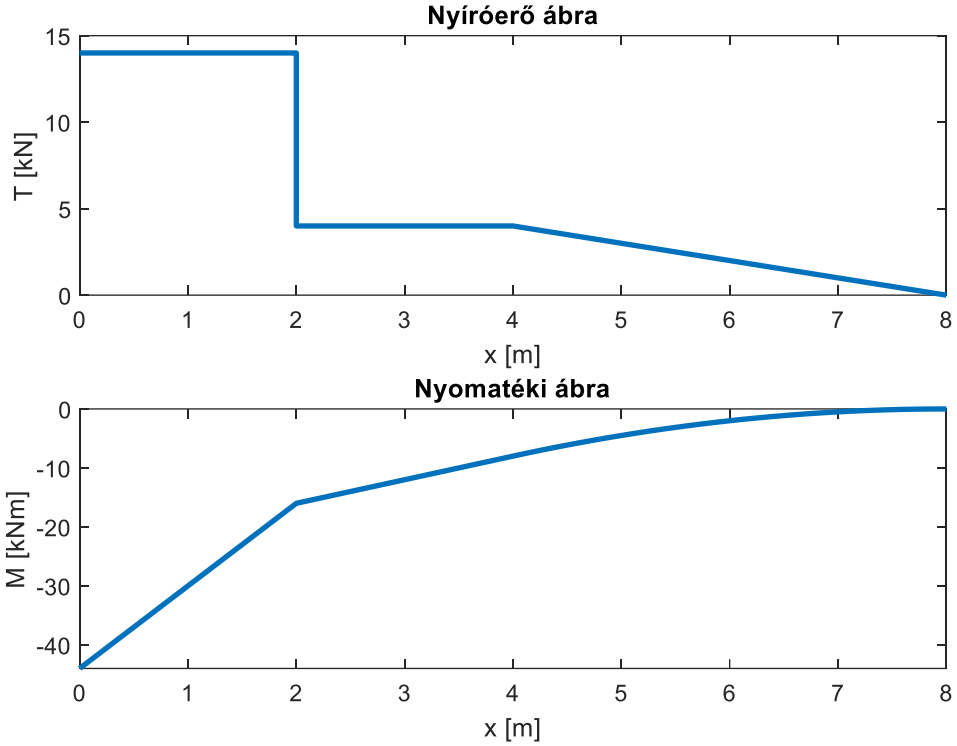
```
> function [X,T,M] = konzol(x)
>     % Konzol igénybevételi értékei, ahol L/4 helyen F erő hat és
>     % L/2-től a konzol végéig q megoszló terhelés. T-nyíróerő,
>     % M-nyomaték
```

```
>       F = 10; L = 8; q = 1;
>       cond1 = and(x>=0,x<=L/4);   % 1. szakasz
>       cond2 = and(x>=L/4,x<=L/2); % 2. szakasz
>       cond3 = and(x>=L/2,x<= L);  % 3. szakasz
>       x1 = x(cond1); % logikai indexelés
>       x2 = x(cond2); % logikai indexelés
>       x3 = x(cond3); % logikai indexelés
>       T1 = []; T2 = []; T3 = []; M1 = []; M2 = []; M3 = [];
>       for i=1:length(x1)
>          T1(i) = F + q*L/2;
>          M1(i) = -F*L/4 - 3*q*L^2/8 + F*x1(i) + q*L/2*x1(i);
>       end
>       for i=1:length(x2)
>          T2(i) = q*L/2;
>          M2(i) = -3*q*L^2/8 + q*L/2*x2(i);
>       end
>       for i=1:length(x3)
>          T3(i) = q*L/2 - q*(x3(i)-L/2);
>          M3(i) = -3*q*L^2/8 + q*L/2*x3(i) - q/2*(x3(i)-L/2).^2;
>       end
>       X = [x1,x2,x3];
>       T = [T1,T2,T3];
>       M = [M1,M2,M3];
>   end
```

A fenti függvényt felhasználva jelenítsük meg most két egymás alatti ábrán az igénybevételi ábrákat, felül a nyíróerő ábrát, alul a nyomatéki ábrát! Egymás alatti, melletti ábrákat a **subplot(n,m,i)** parancs segítségével adhatunk meg, ahol először mint egy mátrixot meg kell adjuk, hogy hány sorban (n) és oszlopban (m) szeretnénk ábrákat megjeleníteni, majd meg kell adni, hogy éppen hányadik (i) ábrába szeretnénk rajzolni (a számozás balról jobbra és fentről lefelé történik). A **subplot** parancs esetén minden ábrának adhatunk címet a **title** paranccsal és az egésznek is az **sgtitle** paranccsal (a Matlab R2018b változatától).

```
>   %% Töréseket tartalmazó függvény - igénybevételi ábrák
>   % Több ábra egymás alatt
>   clc; clear all; close all;
>   % kiszámolhatjuk a nyíróerő és nyomaték értékét egy tetszőleges
    pontban:
>   [X0,T0,M0] = konzol(3) % X0 = 3, T0 = 4, M0 = -12
>   % Illetve egy vektor elemeiben is, a kirajzoláshoz
>   x = 0:0.1:8;
>   [X1,T1,M1] = konzol(x);
>   figure();
>   subplot(2,1,1)
>   plot(X1,T1,'LineWidth',2)
>   title('Nyíróerő ábra'); xlabel('x [m]'); ylabel('T [kN]')
>   subplot(2,1,2)
>   plot(X1,M1,'LineWidth',2)
>   title('Nyomatéki ábra'); xlabel('x [m]'); ylabel('M [kNm]')
>   sgtitle('Igénybevételi ábrák')
```

Igénybevételi ábrák



Kiegészítő anyag (igénybevételi ábrák szingularitás függvényekkel)[1]

Az igénybevételi ábrák értékeinek kiszámítását, megrajzolását lényegesen leegyszerűsítik, ha szingularitás függvényeket (’singularity functions’) használunk, amelyek magukba foglalják az egységugrás, Dirac-delta és hasonló függvényeket. Szingularitás függvények használatával egyetlen analitikus függvénnyel leírhatjuk mind a nyíróerő, mind a nyomaték függvényét az egész tartóra. (Lásd: Dr. Ibrahim A. Assakkaf (2003): Beams: Deformation by singularity functions, ENES 220 – Mechanics of Materials, http://www.assakkaf.com/courses/enes220/lectures/lecture17.pdf).

A szingularitás függvények általánosan a következő formában írhatók le:

$$\langle x - x_0 \rangle^n = \begin{cases} (x - x_0)^n, & ha\ n > 0\ és\ x \geq x_0 \\ 0, & ha\ n > 0\ és\ x < x_0 \end{cases}$$

$$\langle x - x_0 \rangle^0 = \begin{cases} 1, & ha\ x \geq x_0 \\ 0, & ha\ x < x_0 \end{cases}$$

Ebben az esetben a teherfüggvény szinguláris függvényekkel a következőképpen írható le, ha a fellépő reakció erő R (vetületi egyenletekből: $R = F + q \cdot L/2$):

$$q(x) = R \cdot \langle x - 0 \rangle^{-1} - F \cdot \langle x - L/4 \rangle^{-1} - q \cdot \langle x - L/2 \rangle^0$$

Ebből integrálásokkal kaphatjuk meg nyíróerő és a nyomaték függvényét:

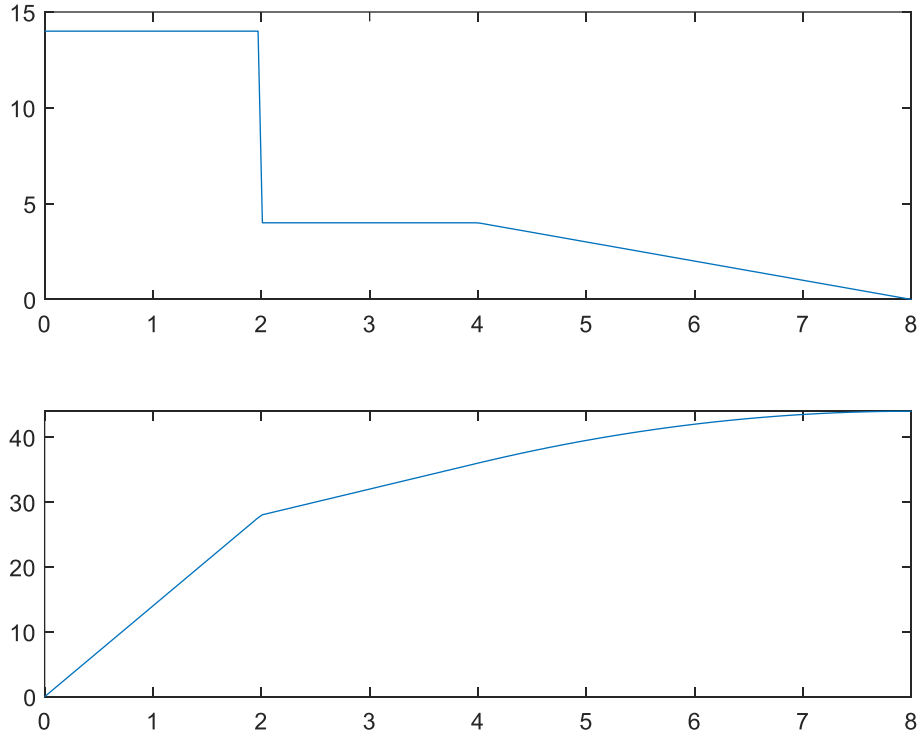
$$F(x) = R \cdot \langle x \rangle^0 - F \cdot \langle x - L/4 \rangle^0 - q \cdot \langle x - L/2 \rangle^1$$

$$M(x) = R \cdot \langle x \rangle^1 - F \cdot \langle x - L/4 \rangle^1 - \frac{q}{2} \cdot \langle x - L/2 \rangle^2$$

[1] Otthoni átnézésre

A fenti függvények Matlab-ban nagyon könnyen megvalósíthatók logikai indexelés használatával. Ebben az esetben nincs szükség meghatározni minden szakaszra külön a nyíróerő és nyomaték függvényét, a teher függvény felírása utána automatikusan adódik a másik kettő.

```
> % Megoldás szingularitás függvényekkel
> F = 10; L = 8; q = 1;
> x = linspace(0,8,200);
> % Reakció erő Fy=0 egyensúlyi egyenlet alapján
> R = F + q*L/2 % 14
> % Nyíróerő ábra és nyomatéki ábra szingularitás függvényekkel
> T = R - F*(x>=L/4) - q*(x-L/2).*(x>=L/2)
> M = R*x - F*(x-L/4).*(x>=L/4) - q/2*(x-L/2).^2.*(x>=L/2)
> figure(); subplot(2,1,1); plot(x,T);
> subplot(2,1,2); plot(x,M)
```



IMPLICIT FÜGGVÉNYEK MEGJELENÍTÉSE

Eddig csak explicit függvények megjelenítésével foglalkoztunk, ahol y egyértelmű függvénye volt x-nek. Mi a helyzet, akkor, amikor y nincs explicit módon kifejezve, hanem implicit formában adott a megjelenítendő összefüggés, például egy kör, vagy egy ellipszis esetében? Jelenítsük meg Matlab-ban a (2;5) középpontú a=4, b=3 fél nagy- és kistengelyű ellipszist! Ezt Descartes-koordináta rendszerben a következőképpen írhatjuk fel:

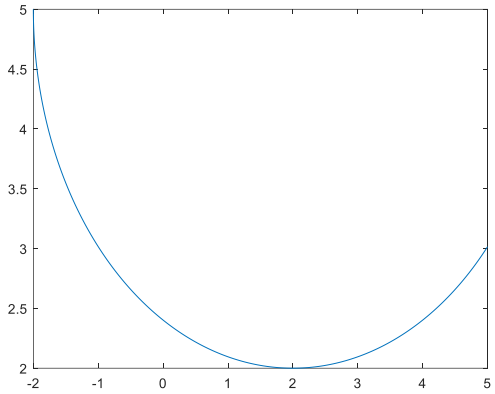
$$\left(\frac{x-2}{4}\right)^2+\left(\frac{y-5}{3}\right)^2=1$$

Ennek a megjelenítése Matlab-ban az **fimplicit** paranccsal történhet. az **fplot** csak explicit függvényeket tud megjeleníteni. A korábbi verziókból egyelőre még megmaradt (és Octave-ban is használható) **ezplot** parancs is használható explicit és implicit függvények megjelenítésére is.

A megjelenítéshez először 0-ra kell redukánunk az egyenletet:

$$\left(\frac{x-2}{4}\right)^2 + \left(\frac{y-5}{3}\right)^2 - 1 = 0$$

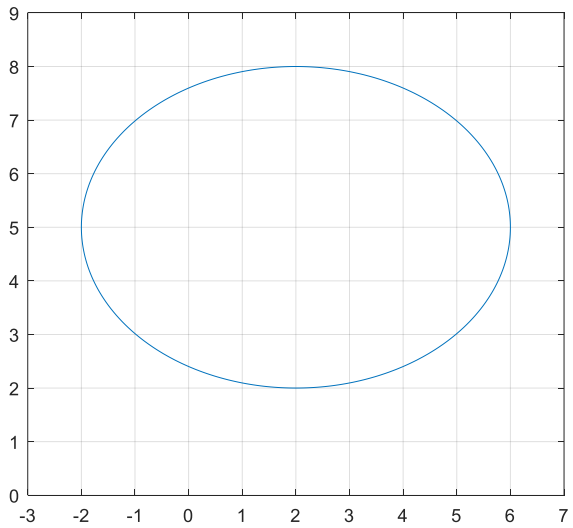
```
> %% Implicit függvények
> f = @(x,y) ((x-2)/4).^2 + ((y-5)/3).^2 - 1
> figure();
> fimplicit(f)
```



Meghívhatjuk az **fimplicit** függvényt önállóan, de akkor egy alapértelmezett tartományon fogja megjeleníteni a függvényt. Célszerű megadni egy vektorban a megjelenítendő terület határait: [x_{min} x_{max} y_{min} y_{max}].

```
> fimplicit(f,[-3 7 0 9]) % vagy Octave-ban: ezplot(f,[-3 7 0 9])
> grid on; axis equal;
```

Plusz beállításként egy rácshálót is bekapcsoltunk itt és egyforma beosztásra állítottuk a tengelyeket.



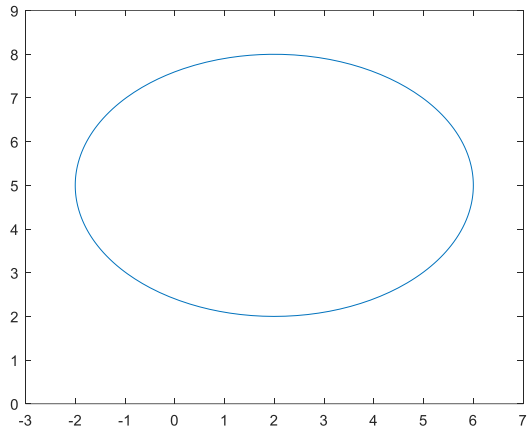
PARAMÉTERESEN ADOTT FÜGGVÉNYEK MEGJELENÍTÉSE

Az előző ellipszist megadhatjuk paraméteresen is, ahol a paraméter az elfordulási szögérték lesz.

$$\begin{cases} x(t) = 2 + 4 \cdot \cos(t) \\ y(t) = 5 + 3 \cdot \sin(t) \end{cases} \quad t \in [0,2\pi]$$

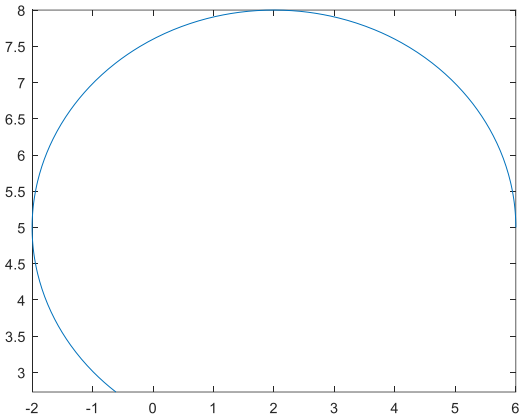
Nézzük meg, hogyan lehet paraméteresen megadott görbét ábrázolni! Először fel kell írnunk x-t és y-t a paraméter (t) függvényeként, majd az **fplot** (vagy **ezplot**) parancsban megadni egymás után a két paraméteres függvényt és a paraméter minimális maximális értékét. Ebben az esetben az **axis** paranccsal módosíthatjuk a tengelyek minimális, maximális értékeit.

```
> %% Paraméteresen megadott görbe ábrázolása
> x = @(t) 2+4*cos(t)
> y = @(t) 5+3*sin(t)
> figure()
> fplot(x,y,[0,2*pi()]) % vagy ezplot(x,y,[0,2*pi()])
> axis([-3 7 0 9])
```



Próbáljuk ki, hogy mi történik, ha módosítjuk a paraméter értékét!

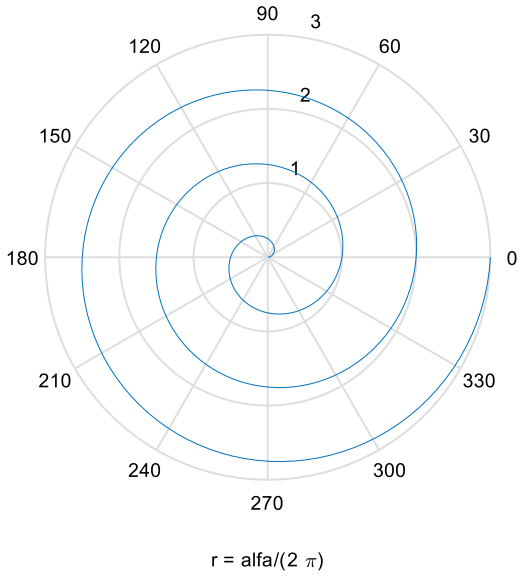
```
> fplot(x,y,[0,4])
```



POLÁRKOORDINÁTÁS ÁBRÁZOLÁS

A derékszögű koordinátás megadás mellett bizonyos esetekben szükség lehet polár koordinátás megjelenítésre is, ehhez az **ezpolar** parancsot használhatjuk. Jelenítsünk meg ezzel egy lineáris spirált, ami háromszor fordul körbe, az elfordulással arányos méretű sugárral!

```
> %% Polár koordinátás megjelenítés
> f = @(alfa) alfa/(2*pi)
> figure()
> ezpolar(f,[0,3*2*pi()])
```



A FEJEZETBEN HASZNÁLT FÜGGVÉNYEK

figure	-	Új grafikus ablak nyitása
plot	-	Adatsorok megjelenítése
hold on; hold off	-	Új ábra rajzolásakor megtartsa-e az előzőt, vagy felülírja
get,set	-	Ábra tulajdonságainak lekérdezése, megváltoztatása
xlabel,ylabel	-	Tengelyek feliratozása
title	-	Cím
legend	-	Jelmagyarázat
grid on; grid off	-	Rácsháló ki/bekapcsolása
axis([x_{min},x_{max},y_{min},y_{max}])	-	Tengelyek minimális, maximális értékének beállítása
delete	-	Névvel elmentett ábra törlése
clf	-	Grafikus ablak tartalmának törlése
close, close all	-	Aktuális/összes grafikus ablak bezárása
fplot, ezplot	-	Explicit módon megadott függvények ábrázolása
subplot	-	Egymás alatti/melletti ábrák megjelenítése mátrix szerűen
sgtitle	-	Subplot-tal részekre osztott ábra egészének a címe
fimplicit, ezplot	-	Implicit módon megadott függvények ábrázolása
ezpolar	-	Polárkoordinátás ábrázolás
fsurf, ezsurf	-	Függvénnyel megadott felület megjelenítése
fcontour, ezcontour	-	Függvénnyel megadott felület szintvonalainak a megjelenítése
axis equal	-	Egyforma tengelybeosztás beállítása