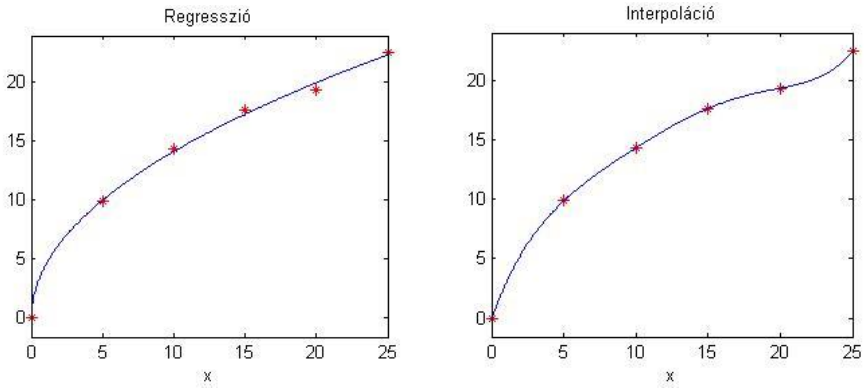
INTERPOLÁCIÓ

A korábbi gyakorlaton regresszióval, függvényillesztéssel foglalkoztunk, amikor meghatároztuk az adott pontokra legjobban illeszkedő egyenest, parabolát, exponenciális függvényt stb. Az illesztett függvény többnyire nem megy át a mért pontokon, de ideális esetben közel halad hozzájuk.

Az interpoláció egy olyan matematikai összefüggés, ami a megadott pontokat tökéletesen reprezentálja, visszaadja ezekben a pontokban a mérési értékeket, a pontok között pedig becslésre használható. Grafikusan ábrázolva a függvény átmegy a mérési pontokon.



INTERPOLÁCIÓ EGYETLEN POLINOMMAL

Egy polinom általános alakja:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

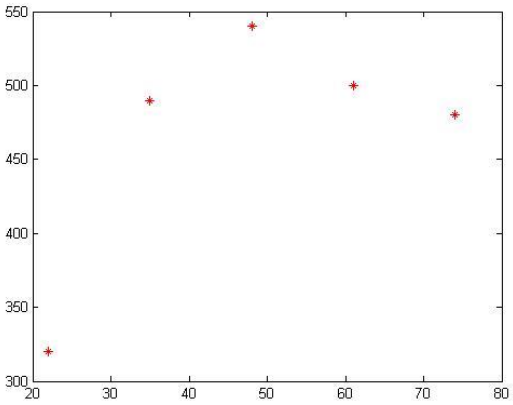
Az $a_n, a_{n-1}, \cdots, a_1, a_0$ együtthatók valós számok, n pedig egy nem negatív egész szám, a polinom fokszáma. Az elsőfokú polinom egy lineáris függvénykapcsolat, képe egyenes, a másodfokú polinom (parabola) és a magasabb fokú polinomok képei pedig valamilyen görbék, minél magasabb a fokszám, annál több 'kanyar', inflexiós pont lehet benne, annál bonyolultabb lehet a képe.

Ha *n* pontból álló adatállományunk van, akkor ezt különböző fokszámú polinomokkal lehet közelíteni, egészen *(n-1)* fokszámig. Alacsonyabb fokszámú polinom illesztése esetén regresszióról beszélhetünk, *(n-1)* fokú polinommal pedig interpolációt kapunk, ekkor a polinom minden ponton keresztül halad. Ez lesz a polinomiális interpoláció esete. Ez egy **globális interpoláció**, mivel az összes adatra egyetlen függvényt illesztünk. Nézzünk egy példát rá!

A szélerőművek teljesítménye a szélsebességgel változik. Egy kísérletben a következő 5 mérési eredményt kaptuk erre vonatkozóan:

Szélsebesség [km/h]	22	35	48	61	74
Teljesítmény [W]	320	490	540	500	480

Az 5 mérési eredményre negyedfokú interpolációs polinom illeszthető. Interpolációval határozzuk meg a szélerőmű teljesítményét 42 és 68 km/h-s szél esetén! Mekkora szélsebességnél lesz a teljesítmény 400 W? Ehhez töltsük be a szeleromu.txt állományt!




```
> % szélerőmű adatai
> clear all; close all; clc;
> data = load('szeleromu.txt')
> v = data(:,1) % szélsebesség
> p = data(:,2) % teljesítmény
> figure(1)
> plot(v,p,'r*')
```

A negyedfokú interpolációs polinom $f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ együtthatóit kiszámolhatjuk egy lineáris egyenletrendszert megoldva, a regressziónál már megismert módon, 5 egyenletet felírva az öt pontra. Az *A* együttható mátrixot **Vandermonde mátrix**nak is nevezik.

$$\begin{aligned} y_1 &= a_4x_1^4 + a_3x_1^3 + a_2x_1^2 + a_1x_1 + a_0 \\ y_2 &= a_4x_2^4 + a_3x_2^3 + a_2x_2^2 + a_1x_2 + a_0 \\ y_3 &= a_4x_2^4 + a_3x_2^3 + a_2x_2^2 + a_1x_2 + a_0 \\ y_4 &= a_4x_3^4 + a_3x_3^3 + a_2x_3^2 + a_1x_3 + a_0 \\ y_5 &= a_4x_4^4 + a_3x_4^3 + a_2x_4^2 + a_1x_4 + a_0 \end{aligned} \quad \rightarrow A = \begin{pmatrix} {x_1}^4 & {x_1}^3 & {x_1}^2 & x_1 & 1 \\ {x_2}^4 & {x_2}^3 & {x_2}^2 & x_2 & 1 \\ {x_3}^4 & {x_3}^3 & {x_3}^2 & x_3 & 1 \\ {x_4}^4 & {x_4}^3 & {x_4}^2 & x_4 & 1 \\ {x_5}^4 & {x_5}^3 & {x_5}^2 & x_5 & 1 \end{pmatrix}; b = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$$

Vagy használhatjuk itt is a **polyfit, polyval** beépített Matlab függvényeket. Most az egyszerűség kedvéért használjuk ez utóbbiakat!

```
> a = polyfit(v,p,4) %  0.0001   -0.0171    0.5627   12.0190  -62.0517
```

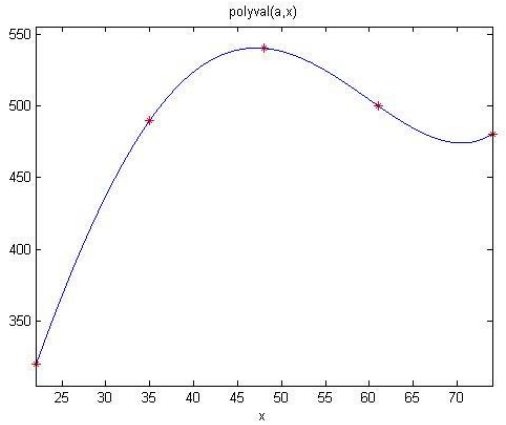
A polyfit-et használva megadhatjuk az illeszteni kívánt polinom fokszámát, és megkapjuk a polinom együtthatóit a legmagasabb fokú tagtól visszafelé a konstans tagig: $a_4 = 0.0001;\ a_3 = -0.0171;\ a_2 = 0.5627;\ a_1 = 12.0190;\ a_0 = -62.0517.$

A polyval parancs a megadott együtthatókból ki tudja számolni egy tetszőleges helyen a polinom értékét, pl. a kérdéses 42 km/h-nál:

```
> polyval(a,42) % 531.7853
```

Vagy definiálhatjuk az együtthatókkal a polinom függvényét is, és akkor ábrázolni is könnyen tudjuk:

```
> fp = @(x) polyval(a,x)
> fp(68) % 476.5008
> hold on;
> fplot(fp,[min(v) max(v)])
```



Ahhoz, hogy megkapjuk, hogy mekkora szélsebességnél lesz a teljesítmény 400 W, egy nemlineáris egyenletet kell megoldanunk! A megoldandó nemlineáris egyenletünk, nullára rendezve: $a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 - 400 = 0$. Szükség lesz egy kezdőértékre is, amit az ábrából vehetünk.

```
> h = @(x) fp(x)-400
> x200 = fzero(h,30) % 27.1296
```

Tehát 42 km/h-s szélnél 532 W a teljesítmény, 68 km/h-nál pedig 477 W. 400 W-os teljesítményt pedig 27 km/h-s szélnél kapunk.

A sztenderd alakba írt *n*-ed fokú polinom $f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0$ illesztéséhez *(n+1)* lineáris egyenletből álló egyenletrendszert kell megoldani. Az egyenletekhez az összes rendelkezésre álló pont koordinátáit be kell helyettesíteni az általános alakba. Gyakorlatban, különösen magasabb fokú polinomok esetében nem hatékony megoldani ezt az egyenletrendszert, gyakran rosszul kondicionált lesz az együttható mátrixunk. Nézzük meg az előző példánk együttható mátrixának kondíciószámát! Ehhez írjuk fel az együttható mátrixot, ami a lineáris egyenletrendszer megoldásához kellene. A felírást megtehetjük a regressziónál megismert módon is, de a Matlabnak van egy külön parancsa a Vandermonde mátrix előállítására, azt is használhatjuk, ugyanaz lesz az eredmény:

```
> A = [v.^4 v.^3 v.^2 v.^1 v.^0] % hagyományos felírás
> A = vander(v) % másik megoldás: Vandermonde mátrixként
> cond(A) %  1.5378e+09
```

Ez a szám már most is nagyon nagy (10^9 nagyságrendű), pedig csak 4-ed fokú, vagyis nem is nagyon magas fokszámú polinomot illesztettünk. Ellenőrzésképp megnézhetjük, a lineáris egyenletrendszer megoldását, hogy ugyanazt adja-e, mint a polyfit!

```
> x = A\p % 0.0001; -0.0171; 0.5627; 12.0190; -62.0517
```

LAGRANGE ÉS NEWTON INTERPOLÁCIÓS POLINOMOK

Adott pontokon keresztül egy interpolációs polinom írható fel, ez viszont többféle alakban is megadható, nézzünk meg az általános alak mellett röviden két másikat, amit gyakran célszerűbb használni, könnyebb felírni és nem lesz rosszul kondicionált a feladat. Az egyik ilyen alak a Lagrange interpolációs polinom, a másik a Newton.

LAGRANGE INTERPOLÁCIÓS POLINOM

Ez a fajta polinom mindenféle számítás, egyenletrendszer megoldás nélkül, a pontok koordinátáiból felírható, a következő alakban *n* pontra:

$$f(x) = \sum_{i=1}^{n} y_i L_i(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

ahol $L_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$ -t Lagrange függvényeknek hívjuk. Két pontra felírva:

$$f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2$$

Három pontra felírva:

$$f(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3$$

- Látszik, hogy a pontok koordinátáit használva, előzetes számítások nélkül is felírható az interpolációs polinom.
- Nehézkes vele dolgozni, minden egyes interpolálandó pontnál fel kell írjuk újra az adott pontra az egész egyenletet, nem elég csak az együtthatókat behelyettesíteni, mint az általános alaknál.
- Ha a ponthalmazunk új ponttal bővül, akkor az összes Lagrange függvényt újra kell számolni, ebben különbözik a Newton formától, ahol új pontok esetén csak az új tagokat kell kiszámolni.

NEWTON-FÉLE INTERPOLÁCIÓS POLINOM[1]

A Newton-féle polinom az ún. osztott differenciák segítségével írható fel, rekurzív úton. Általános alakja a polinomnak:

$$f(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + \cdots + a_n(x - x_1)(x - x_2) \cdots (x - x_{n-1})$$

Két pontra felírva az együtthatók:

$$a_1 = y_1; a_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

Definiáljuk az elsőrendű osztott differenciákat a következőképpen:

$$f[x_{i+1}, x_i] = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

Ez tulajdonképpen az egyenes meredeksége, és két pontra megegyezik a_2 együtthatóval.

Három pontra felírható a másodrendű osztott differencia $f[x_3, x_2, x_1]$, ami két elsőrendű osztott differencia különbsége osztva $(x_3 - x_1)$ -gyel, ez lesz az a_3 együttható (az első kettő együttható megegyezik a korábbiakkal).

[1] Otthoni átnézésre

$$a_3 = f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}$$

Hasonlóképp négy pontra felírható a harmadrendű osztott differencia két másodrendű osztott differencia különbségét elosztva $(x_4 - x_1)$-gyel, és így tovább.

Általánosan a k-adrendű osztott differencia:

$$f[x_{i+k}, \cdots, x_i] = \frac{f[x_{i+k}, \cdots, x_{i+1}] - f[x_{i+k-1}, \cdots, x_i]}{x_{i+k} - x_i}, (k = 1,2, \cdots, n) és\ (i = 0, \cdots, n - k).$$

- Látszik, hogy ebben az esetben is a pontok koordinátáit használva, előzetes számítások nélkül is felírható az interpolációs polinom.
- Ezzel már nem olyan nehézkes dolgozni, ha egyszer már meghatároztuk $a_1 \cdots a_n$ együtthatókat, akkor ezeket felhasználhatjuk bármelyik pont interpolációjánál.
- Ha a ponthalmazunk új ponttal bővül, akkor nem kell mindent újraszámolni, csak az új együtthatót. Ezáltal könnyen bővíthető új ponttal a halmaz, és a pontoknak nem kell sorrendben lennie.

TÁROZÓ JELLEGÖRBE INTERPOLÁCIÓJA

Nézzünk interpolációra egy vízépítő mérnöki feladatot. A tározóméretezések egyik elengedhetetlen alapadata a tározó morfológiai jelleggörbéje, ami megadja, hogy egy adott vízálláshoz mekkora víztérfogat és felület tartozik. Adottak a következő adatok:

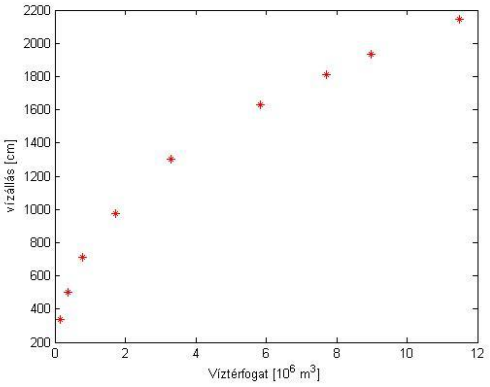
Vízállás H [cm]	Térfogat V [10^6 m^3]	Felület F [km^2]
336	0.16	0.05
504	0.36	0.09
714	0.79	0.19
976	1.73	0.37
1302	3.31	0.62
1628	5.83	0.90
1812	7.72	1.05
1932	8.98	1.16
2142	11.50	1.27

Ábrázoljuk a térfogatot leíró jelleggörbét, szokás szerint, úgy, hogy a vízszintes tengelyen a térfogat, a függőleges tengelyen pedig a vízállás legyen. Határozzuk meg ezek alapján interpolációval, hogy mekkora víztérfogat tartozik 15 m-es vízszinthez, illetve mekkora vízszint tartozik 12 millió m^3 térfogathoz!

Töltsük be a jelleggörbékhez tartozó adatokat a tarozo.txt állományból!

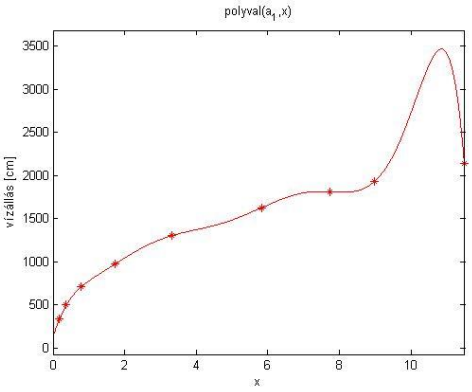
```
> clc; clear all; close all;
> adat = load('tarozas.txt')
```

```
> H = adat(:,1); % cm
> V = adat(:,2); % millió m^3
> F = adat(:,3); % km^2
>
> figure(1)
> plot(V,H,'*'); hold on;
> ylabel('vízállás [cm]')
> xlabel('Víztérfogat [10^6 m^3]')
```



Interpoláció esetén *n* pontra, *(n-1)*-ed fokú polinomot illeszthetünk. Nézzük meg, ez mit jelent a mostani esetben. Most 9 adatunk van, erre 8-ad fokú polinomot illeszthetünk!

```
> n = length(V) % 9
> a1 = polyfit(V,H,n-1)
> p1 = @(x) polyval(a1,x)
> fplot(p1,[0,max(V)])
```



Mi történt? Jó lehet a fenti görbe interpolációra? Nyilvánvaló hogy nem. Ez a túlzott illeszkedés esete, amikor a magas fokszámú polinom tökéletesen visszaadja a ismert pontok értékeit, de köztük (különösen a széleken) oszcillálni kezd, jelentősen eltér az adatok trendjétől, ezért nem alkalmazható megbízhatóan interpolációra, extrapolációra. Ezt a hullámzást, oszcillációs jelenséget **Runge jelenség**nek hívják. Kevés pont estén, amikor a polinom fokszáma alacsony, többnyire jól használhatóak az interpolációs polinomok, sok pont, magas fokszámú polinom esetén azonban más megoldást kell keresni!

A polyfit parancs futása után a Matlab egy figyelmeztető üzenetet is kiír:

```
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
```

Ha lekérdezzük a Vandermonde mátrix kondíció számát, akkor egy nagyon nagy számot kapunk (10^{10} nagyságrend), ami rosszul kondicionált mátrixot jelent.

```
> A = vander(V);
> cond(A) % 2.7260e+10
```

SPLINE INTERPOLÁCIÓ

Ha sok pont között szeretnénk interpolációt végezni, akkor jobb eredményt lehet elérni több, alacsony fokszámú polinom alkalmazásával, mint egy magas fokúval. Minden egyes alacsony fokszámú polinom csak egy adott szakaszra, intervallumra érvényes, két vagy több pont között. Általában minden polinomnak megegyezik a fokszáma, csak az együtthatókban térnek el egymástól. Amikor elsőfokú polinomokat használunk, akkor a pontokat egyenes vonalak kötik össze, másodfokú (négyzetes) vagy harmadfokú (köbös) esetben a pontok görbékkel vannak összekötve. Ez a szakaszonként eltérő paraméterű polinomiális interpoláció, amit **spline** interpolációnak neveznek. Ez tulajdonképpen egy fajta lokális interpoláció, mivel egy-egy szakaszra csak a környező pontokat vesszük figyelembe.

Az n-edfokú spline-ban legfeljebb n-edfokú polinom szakaszok csatlakoznak egymáshoz. A legegyszerűbb eset, ha az adott pontok közé lineáris függvényeket (elsőfokú polinomokat) illesztünk. Ez az Euler-féle töröttvonalak módszere. Gyakran szükséges, hogy a csatlakozási pontokban ne csak folytonos, hanem differenciálható is legyen a függvény. Ilyen lehet például egy másodfokú spline, ahol a csatlakozási pontokban a függvénynek jobbról és balról megegyeznek a deriváltjai is. Ezt négyzetes (kvadratikus) spline-nak is nevezik.

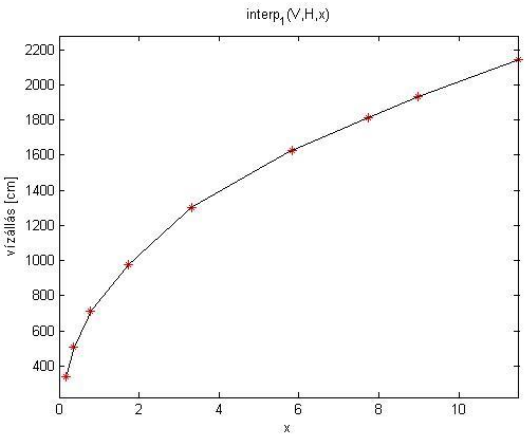
Általánosan k-ad fokú és m-ed rendű spline az, ahol szakaszonként k-ad fokú polinomokat illesztünk, és a közbülső pontokban m-ed rendig megegyeznek a deriváltak. Az egyik leggyakrabban használt spline a köbös, másodrendű spline interpoláció, amikor is az adott pontok közé harmadfokú polinomokat illesztünk úgy, hogy a csatlakozási pontokban a függvénynek megegyeznek a deriváltjai és második deriváltjai is. Létezik köbös elsőrendű spline interpoláció is, amikor szintén harmadfokú polinomokat illesztünk a pontokra, de csak az első deriváltak egyezését írjuk elő (pl. Hermite interpolációs polinomok).

LINEÁRIS SPLINE INTERPOLÁCIÓ

Illesszünk lineáris függvényeket a pontok közé! Adott *n* pont esetén *(n-1)* szakasz van, amire *(n-1)* egyenest kell meghatároznunk. Ezt a legegyszerűbben a két pontra felírt Lagrange-polinom segítségével tehetjük meg:

$$f_i(x) = \frac{(x - x_{i+1})}{(x_i - x_{i+1})} y_1 + \frac{(x - x_i)}{(x_{i+1} - x_i)} y_2$$

Matlab-ban az **interp1** függvényt használhatjuk általánosan spline interpolációra. Ennél egy paraméterben megadhatjuk, hogy milyen spline interpolációt szeretnénk használni, ha nem adunk meg semmit, akkor az alapértelmezett a lineáris interpoláció. Lehetséges interpolációs módszerek az **interp1**-nél: 'linear' - alapértelmezett, 'nearest' - legközelebbi szomszéd, 'pchip' - köbös elsőrendű spline, 'spline' - köbös másodrendű spline.




```
> figure(2);
> plot(V,H,'r*');hold on;
> ylabel('vízállás [cm]');
> xlabel('Víztérfogat [10^6 m^3]');
> sp = @(x) interp1(V,H,x)
> fplot(sp,[0,max(V)])
```

Lineáris függvényillesztésnél folytonos lesz ugyan a függvényünk, de a meredekségekben törések lesznek. Ha simább függvényt szeretnénk kapni, akkor magasabb fokú spline-t kell használni.

NÉGYZETES SPLINE INTERPOLÁCIÓ[2]

Négyzetes (kvadratikus, másodfokú) spline esetében az adott pontok közé másodfokú polinomokat ($y = a \cdot x^2 + b \cdot x + c$) illesztünk úgy, hogy a csatlakozási pontokban a függvénynek jobbról és balról megegyeznek az első deriváltjai is.

Egy másodfokú polinomnak 3 ismeretlen együtthatója van, és *n* pont esetén *(n-1)* szakaszra kell ezeket meghatározzuk, vagyis *3*(n-1)=3n-3* ismeretlenünk van, ezekre kell egyenleteket felírnunk, feltételeket megadnunk.

- Minden polinomnak át kell mennie a szakasz végpontjain, ebből szakaszonként két egyenlet írható fel a következő alakban: $f_i(x) = a_i \cdot x^2 + b_i \cdot x + c_i$, vagyis összesen *2*(n-1)=2n-2* egyenlet.
- A közbülső *(n-2)* pontban a deriváltak ($f'(x) = 2a_i x + b_i$) is megegyeznek, erre *(n-2)* egyenletet lehet felírni a következő alakban: $2\ a_{i-1}x_i + b_{i-1} = 2\ a_i x_i + b_i$
- Mivel a fenti két feltétel még csak *(3n-4)* egyenletet jelent (*3n-3)* ismeretlen meghatározására, még egy feltételt meg kell adni. Itt több lehetőség is van, lehet ez például az, hogy a második derivált értéke legyen 0 az első (vagy az utolsó) pontnál: $f_1''(x) = 2a_1$, vagyis $a_1 = 0$. Ez tulajdonképpen azt jelenti, hogy az első két pontot (vagy az utolsó két pontot) egyenessel kötjük össze.

A négyzetes spline-k illesztéséhez *(3n-4)* egyenletet kell felírjuk. A köbös, másodrendű spline-k használata elterjedtebb, részben mivel a második deriváltak (görbületek) is megegyeznek és emiatt simább a függvény, részben pedig amiatt, hogy levezethető egy olyan alakja is, ahol csak *(n-2)* egyenletet kell megoldani. A Matlab beépített függvényei között is csak köbös spline szerepel, négyzetes nem.

KÖBÖS MÁSODRENDŰ SPLINE INTERPOLÁCIÓ

Nézzük most az egyik leggyakrabban használt spline a köbös, másodrendű spline interpoláció levezetését az általános harmadfokú polinom képlete alapján!

Ebben az esetben az adott pontok közé úgy illesztünk harmadfokú polinomokat ($y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$) , hogy a csatlakozási pontokban a függvénynek megegyeznek az első és a második deriváltjai is. Adott *n* pont esetén *(n-1)* szakaszunk van, ezekre kell meghatároznunk harmadfokú polinomokat. Minden egyes harmadfokú polinomnak 4 ismeretlen együtthatója van, vagyis *4*(n-1)=4n-4* ismeretlenünk van, ezekre kell egyenleteket felírnunk, feltételeket megadnunk.

- Minden polinomnak át kell mennie a szakasz végpontjain, ebből szakaszonként két egyenlet írható fel a következő alakban: $y_i = a_i \cdot x^3 + b_i \cdot x^2 + c_i \cdot x + d_i$, vagyis összesen *2*(n-1)=2n-2* egyenlet.
- A közbülső *(n-2)* pontban megegyeznek a deriváltak ($f'(x) = 3ax^2 + 2bx + c$) is, erre *(n-2)* egyenletet lehet felírni a következő alakban: $3\ a_{i-1}{x_i}^2 + 2b_{i-1}x_i + c_{i-1} = 3\ a_i{x_i}^2 + 2b_i x_i + c_i$
- A közbülső *(n-2)* pontban a második deriváltak ($f''(x) = 6ax + 2b$) is megegyeznek, erre is *(n-2)* egyenletet lehet felírni a következő alakban: $6\ a_{i-1}x_i + 2b_{i-1} = 6\ a_i x_i + 2b_i$

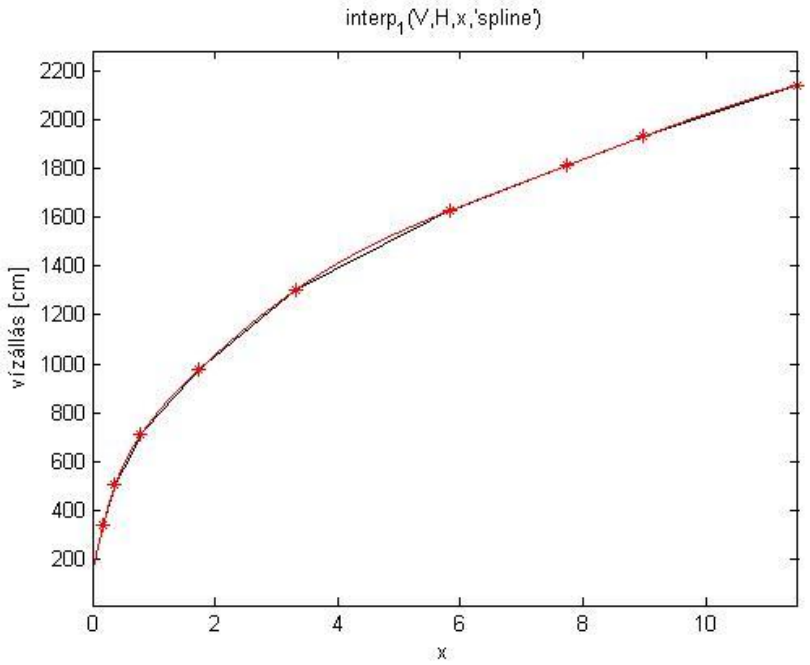
[2] Otthoni átnézésre

- Összesen eddig *(4n-6)* egyenletet írtunk fel *(4n-4)* ismeretlenre, tehát még két feltételt meg kell adni, amit többféleképpen lehet. Az egyik lehetőség, hogy a végpontokban a második deriváltak legyenek 0-k, ezt a fajta spline-t hívják természetes köbös spline-nak. Egy másik módszer, amit a Matlab beépített függvénye is használ, a „not-a-knot" feltétel. Ezt azt jelenti, hogy a második és az utolsó előtti pontban a harmadik deriváltak is megegyeznek.

Megjegyzés: levezethető egy másik alakja is ennek a köbös spline-nak, ahol nem *(4n-4)* egyenletet kell megoldani, hanem elég egy (n-2) lineáris egyenletből álló rendszert megoldani. Ez a levezetés a második deriváltakból indul ki és a Lagrange alakot használja. Lásd pl.: Amos Gilat, Vish Subramaniam (2011): Numerical Methods, An Introduction with Applications Using MATLAB, John Wiley & Sons

Illesszünk harmadfokú köbös spline-t Matlab-ban a tározó jelleggörbéjére! Ehhez használjuk ismét az **interp1** parancsot, csak most adjunk meg neki egy módszert is, az alapértelmezett 'linear' helyett legyen 'spline'!

```
> sp2 = @(x) interp1(V,H,x,'spline')
> fplot(sp2,[0,max(V)])
```



Látjuk, hogy ez egy simább lefutású függvényt eredményezett. Mint korábban szó volt róla, ez nem a természetes spline-t használja, hanem a not-a-knot", vagyis 'nem csomópont' feltételt. Ezt azt jelenti, hogy a második és az utolsó előtti pontban a harmadik deriváltak is megegyeznek. Mivel itt harmadfokú polinomokról van szó, ezért az első kettő és az utolsó kettő szakaszon is teljesen megegyeznek a paraméterek, tehát tulajdonképpen az első 3 és az utolsó 3 pontra egy-egy polinomot illesztünk. Innen származik a neve, hogy a második és az utolsó előtti pont nem igazi csomópont.

Mivel a leggyakrabban ezt a köbös, másodrendű spline interpolációt alkalmazzák, ezért erre van egy külön parancs is, **spline** parancsként, aminek a működése egyenértékű az **interp1** 'spline' módszerével:

```
> sp2 = @(x) spline(V,H,x)
```

Térjünk vissza az eredeti kérdésre és az illesztett spline alapján határozzuk, hogy mekkora vízszint tartozik 12 millió m^3 térfogathoz, illetve mekkora víztérfogat tartozik 15 m-es (1500 cm-es) vízszinthez!

```
> % Mekkora vízállás tartozik 12 millió m^3 vízhez?
> H15 = sp2(12) % 2174 cm
> % 1500 cm-es vízállás mekkora víztérfogatot jelent?
> f = @(x) sp2(x)-1500
> V1500 = fzero(f,5) % 4.6699 millió m^3
```

KÖBÖS ELSŐRENDŰ SPLINE INTERPOLÁCIÓ

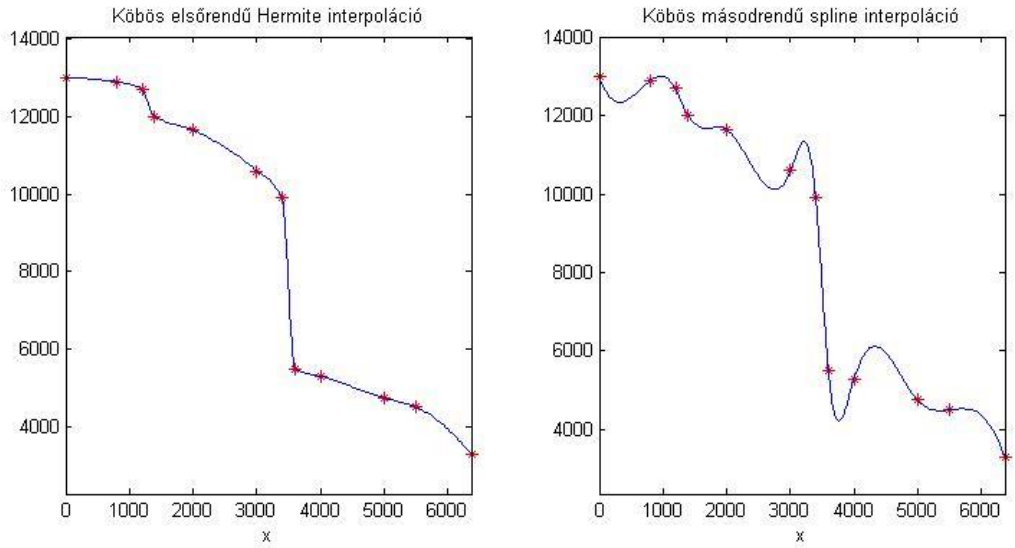
Matlab-ban van egy másik köbös spline interpoláció is. Az **interp1** parancsot meghívhatjuk 'pchip' módszerrel is (piecewise hermite interpolation). Ez egy köbös elsőrendű interpoláció, ahol ugyan harmadfokú polinomokat illesztünk az egyes szakaszokra, de csak az első deriváltak folytonosságát kötjük ki. Ez egy köbös Hermite interpoláció, ahol a deriváltakat is meghatározza a Matlab numerikus differenciálásból 3 pontra a közbülső pontok esetében és 2 pontból a függvény elején/végén. Nézzük meg mikor lehet hasznos ez a módszer!

Vegyünk most egy felsőgeodéziához köthető példát. A Föld gravitációs erőterének számításaihoz szükséges ismerni a Föld sűrűségének változásait. A Föld sűrűsége (ρ) a sugarával (R) együtt változik, megközelítően az alábbiak szerint:

Sugár [km]	0	800	1200	1400	2000	3000	3400	3600	4000	5000	5500	6370
Sűrűség [kg/m^3]	13000	12900	12700	12000	11650	10600	9900	5500	5300	4750	4500	3300

Illesszünk spline görbét a sugár-sűrűség értékekre, majd az interpolációs görbe alapján számítsuk ki mekkora lesz a Föld sűrűsége 3200 km-es sugárnál, illetve mekkora sugárnál lesz a sűrűség értéke pont 4000 kg/m^3? Ehhez töltsük be a fold_surusege.txt fájlt, majd illesszünk a pontokra köbös másodrendű és köbös elsőrendű spline-t is! Nézzük meg melyik illeszkedik jobban!

```
> data = load('fold_surusege.txt')
> r = data(:,1) % kilométerben a sugár
> ro = data(:,2) % sűrűség kg/m^3
>
> % köbös elsőrendű Hermite interpoláció
> fsp1 = @(x) interp1(r,ro,x,'pchip') % vagy 'pchip'
> fsp1(3200) % 10361
> figure(1);subplot(1,2,1);
> plot(r,ro,'r*'); hold on;
> fplot(fsp1,[0,max(r)]);
> title('Köbös elsőrendű Hermite interpoláció')
>
> % köbös másodrendű spline interpoláció
> fsp2 = @(x) spline(r,ro,x)
> fsp2(3200) %  11350
> subplot(1,2,2)
> plot(r,ro,'r*'); hold on;
> fplot(fsp2,[0,max(r)]);
> title('Köbös másodrendű spline interpoláció')
```



A két módszer elég nagy eltéréseket adott a 3200 km-nél lévő sűrűség értékekre, az első esetben 10361 kg/m^3 lett az eredmény, a másodikban pedig 11350 kg/m^3.

Az ábra alapján egyértelmű, hogy most jobb illeszkedést kaptunk a harmadfokú elsőrendű spline-ra ('pchip'), mint a harmadfokú másodrendű esetben ('spline'). Azért jobb most az előbbi módszer, mivel az adatokban erős törések, ugrások vannak. Sima függvények esetén a másodrendű 'spline' ad jobb közelítést.

A feladat második felére, hogy mekkora sugárnál lesz a sűrűség értéke pont 4000 kg/m^3 éppen ezért használjuk az elsőre meghatározott spline-t!

```
> % Mekkora sugárhoz tartozik 4000 kg/m^3 sűrűség?
> f = @(x) fsp1(x)-4000
> R4000 = fzero(f,5500) % 5958.9 km
```

Tehát kb. 6000 km-es sugárnál lesz a sűrűség értéke 4000 kg/m^3.

ÚJ FÜGGVÉNYEK A GYAKORLATON

vander	-	Vandermonde mátrix
interp1 (method: linear, nearest, spline, pchip)	-	Egydimenziós interpoláció (módszer: lineáris, legközelebbi szomszéd, köbös másodrendű, köbös elsőrendű Hermite interpoláció)
spline	-	Egydimenziós, köbös másodrendű spline interpoláció