9. gyakorlat

Ezen a gyakorlaton programozási feladatok szerepelnek.

1. feladat

Az előző gyakorlat feladatának megoldását programozzuk le. A LER-t oldjuk meg a Matlab-bal, rajzoljuk ki a függvényt és a megoldást! Harmadfokú esetben is részintervallumonként rajzoljunk.

$f(x) = \sin\left(\frac{\pi}{2}x\right), \quad x \in [-1;1]$-en interpoláljuk a függvényt a $-1, 0, 1$ alappontokon

a) másodfokú spline-nal $f'(-1) = 0$ peremfeltétellel,

b) harmadfokú spline-nal, Hermite-féle peremfeltétellel: $f'(-1) = f'(1) = 0$.

Megjegyzés

Emlékeztető: Az $a)$ alfeladathoz tartozó lineráis egyenletrendszer a következő volt - most kivételesen másolható Matlab alakban megadva - másolható Matlab alakban:

```
 A = [ 1, -1, 1, 0, 0, 0;
0, 0, 1, 0, 0, 0;
0, 0, 0, 0, 0, 1;
0, 0, 0, 1, 1, 1;
0, 1, 0, 0, -1, 0;
-2, 1, 0, 0, 0, 0];

b = [-1; 0; 0; 1; 0; 0]';
```

Megjegyzés

Emlékeztető: A harmadfokú spline-nál a Hermite-féle peremfeltétel miatt a linerális egyenletrendszer a következő alakban adható meg:

```
A = [ -1, 1, -1, 1, 0, 0, 0, 0;
0, 0, 0, 1, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 1;
0, 0, 0, 0, 1, 1, 1, 1;
0, 0, 1, 0, 0, 0, -1, 0;
0, 1, 0, 0, 0, -1, 0, 0;
3, -2, 1, 0, 0, 0, 0, 0;
0, 0, 0, 0, 3, 2, 1, 0];

b = [-1; 0; 0; 1; 0; 0; 0; 0]';
```

2. feladat

Készítsünk másodfokú *spline*-t, amely interpolál az $\mathbf{x(i)}, \mathbf{y(i)}$ pontokban ($i = 1, \ldots, n$).

- A hiányzó peremfeltétel az $x(1)$ pontbeli derivált értéke, mely adott (m).
- Adjuk meg az $n-1$ db másodfokú polinomot és rajzoljuk fel a *spline*-t!
- A kiértékelésnél használjuk a MATLAB `polyval` utasítását jó paraméterrel.
- Képletek a 7. előadás diái között megtalálhatóak.
- `sp12.m` legyen a neve.

Bemenő paraméterei: $\mathbf{x}$: az alappontok vektora, $\mathbf{y}$: a függvényértékek vektora, m: a baloldali végpontban a derivált értéke

Kimenő paraméter: $\mathbf{P}$: mátrix, mely a soraiban az intervallumonként megadott polinomok együtthatóit tartalmazza az $(x - x_k)$ hatványai szerint.

a) Próbáljuk ki a $f(x) = \sin\left(\frac{\pi}{2}x\right),\ x \in [-1; 1]$ függvényre és $m = 0$-ra.

```
x = [-1,0,1];
y = sin(pi/2*x);
xx = linspace(-1,1,100);
yy = sin(pi/2*xx);
plot(xx,yy,'r')
hold on
spl2(x,y,0);
hold off
```

b) Próbáljuk ki a $f(x) = \sin\left(\frac{\pi}{2}x\right)$ függvényre $x(i) = i$-re, $m = \frac{\pi}{2}$-re. Teszteljük, milyen m-re közelít jobban.

```
x = [0 1 2 3 4];
y = sin(pi/2*x);
xx = linspace(0,4,50);
yy = sin(pi/2*xx);
plot(xx,yy,'r')
hold on
spl2(x,y,pi/2);
hold off
```

3. feladat

A Matlab beépített függvényei

`spline(x,y,xx)`

Köbös `spline`-t állít elő. x-ben vannak az alappontok, y-ban a függvényértékek. Az xx vektorban megadott értékekre számítja ki a `spline` értékét.

- Ha az x és y vektor azonos elemszámú, akkor a *not-a-knot* feltétellel dolgozik, ami azt jelenti, hogy az első részintervallum végén és az utolsó elején az $S''' \in C$ feltételt is megköveteli, ez a két plusz feltétel az egyértelműséghez. Mivel harmadfokú a `spline`, ez azt jelenti, hogy az első két és utolsó két intervallumon egy harmadfokú polinomot adunk meg két intervallumra.
- Ha az y vektor kettővel nagyobb elemszámú, akkor az első és utolsó y-beli elem a két végpontbeli derivált értékeket tartalmazza, azaz *Hermite*-féle feltétellel dolgozik, ez a két plusz feltétel az egyértelműséghez.

`spline(x,y)` Köbös `spline`-t állít elő. x-ben vannak az alappontok, y-ban a függvényértékek. Az intervallumonkénti polinom előállítást adja meg, amit a `ppval` utasítással kezelhetünk. Érdemes megnézni az `mkpp`, `unmkpp` utasításokat, melyek szintén az intervallumonkénti polinomokkal dolgoznak.

`pchip(x,y,xx)` Szakaszonkénti köbös *Hermite* interpolációt állít elő. x-ben vannak az alappontok, y-ban a függvényértékek. Az xx vektorban megadott értékekre számítja ki a görbe értékét. Az S görbe interpolál az x-ben megadott pontokon, $S' \in C$, de a második derivált nem feltétlenül folytonos. Egy részintervallumon a két végpontbeli függvényértékből és a derivált értékekből dolgozik. Megőrzi a monotonitást, ha az adatok ilyenek, akkor S is ilyen lesz. (Ezt a `spline` nem tudja.) Ahol az adatoknak szélsőértéke van, ott S-nek is. Sima (ahol a magasabb derivált is folytonosak) függvények esetén a `spline` ad jobb közelítést.

`pchip(x,y)` Az előző Matlab függvény szakaszosan polinom (piecewise polynomial) alakja.

A következő programokkal a `pchip` és a Matlab által megvalósított kétféle peremfeltételű `spline` közelítést hasonlítjuk össze egy példán.

a) Összehasonlító példa `spline`-ra és `pchip`-re:

```
x1 = 1:4;
y1 = sin(pi/2*x1);
x2 = 0:5;
y2 = [1,y1,0];
xx = linspace(0,5,50);
yy = sin(pi/2*xx);
yy1 = pchip(x2,y2,xx);
yy2 = spline(x2,y2,xx); % 'not-a-knot' spline
plot(x2,y2,'o',xx,yy,xx,yy1,xx,yy2)
legend('adatok','sin','pchip','spline')
```

b) Összehasonlítás `pchip`-re és *Hermite*-féle `spline`-ra:

```
x2 = 0:5;
y2 = sin(pi/2*x2);
xx = linspace(0,5,50);
yy = sin(pi/2*xx);
yy1 = pchip(x2,y2,xx);
yy2 = spline(x2,[pi/2,y2,0],xx); % Hermite peremfeltételű spline
plot(x2,y2,'o',xx,yy,xx,yy1,xx,yy2);
legend('adatok','sin','pchip','spline')
```

c) Összehasonlítás a kétféle peremfeltételre (not-a-knot, Hermite-féle):

```
x2 = 0:5;
y2 = sin(pi/2*x2);
xx = linspace(0,5,50);
yy = sin(pi/2*xx);
yy1 = spline(x2,y2,xx); % not-a-knot peremfeltételű spline
yy2 = spline(x2,[pi/2,y2,0],xx); % Hermite peremfeltételű spline
plot(x2,y2,'o',xx,yy,xx,yy1,xx,yy2);
legend('adatok','sin','spline-not-a-knot','spline-Hermite')
```

1. megoldás

Miután felvettük a mátrixot és a jobb oldalt, a feladat megoldása a következő (is lehet):

a)

```
p = A;
P = [p(1:3)'; p(4:6)'];
spldrawer(x, f, P);
```

b)

```
p = A;
P = [p(1:4)'; p(5:8)'];
spldrawer(x, f, P);
```

2. megoldás

Órai gyakorló feladat.

3. megoldás

Leírás.