Numerikus számítások II. zárthelyi dolgozat, 2008/09. II. félév, Minta

Minden feladat 5 pontot ér, így összesen 40 pont szerezhető a feladatsorral. Sikeres zárthelyihez legalább 16 pont szükséges. Az 1. feladatot csak írásban kell megoldani. Beadandó a feladatlap, a 2., 3. és 6. feladatokat megoldó m-fájl ill. az 5. feladatban készített ábra,

Az internet kivételével minden tárgyi eszköz használható a zh-hoz.

1. Feladat. Adjuk meg az

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

mátrix LU- és Cholesky felbontását! A számolást a feladatlap hátulján írásban végezzük el! (MATLAB-bal lehet ellenőrizni.)

Az $\mathbf{A}$ mátrix első sorának $l_{21} = 2/3$-szeresét vonjuk ki a második sorból, majd az $l_{31} = 1/3$-szeresét a harmadikból, végül a második sor $l_{32} = 4/5$-szeresét vonjuk ki a harmadikból (azaz végezzük el a Gauss-eliminációt). Ezzel előáll az $\mathbf{U}$ mátrix.

$$\mathbf{U} = \begin{bmatrix} 3 & 2 & 1 \\ 0 & 5/3 & 4/3 \\ 0 & 0 & 8/5 \end{bmatrix}$$

Az $\mathbf{L}$ mátrix főátlójában 1-esek vannak, a fenti l_{21}, l_{31}, l_{32} elemeket írjuk be a megfelelő helyekre, a többi elem legyen nulla. Tehát

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2/3 & 1 & 0 \\ 1/3 & 4/5 & 1 \end{bmatrix}$$

A Cholesky-felbontásnál olyan $\mathbf{L}$ alsó háromszögmátrixot keresünk, melyre $\mathbf{L}\mathbf{L}^T = A$. Az $\mathbf{L}$ mátrix elemei a mátrixszorzás szabályát figyelembe véve balról jobbra és fentről lefelé határozhatók meg. Így kapjuk, hogy

$$\begin{bmatrix} \sqrt{3} & 0 & 0 \\ 2/3\sqrt{3} & 1/3\sqrt{15} & 0 \\ 1/3\sqrt{3} & \frac{4}{15}\sqrt{15} & 2/5\sqrt{10} \end{bmatrix}.$$

2. Feladat. Keressük meg a help-ben a `tic` és `toc` parancsokat és nézzük meg, hogy hogy kell őket használni. Hasonlítsuk össze a segítségükkel a MATLAB egyenletrendszer-megoldó függvényének futási idejét és az $\overline{\mathbf{x}} = \mathbf{B}^{-1}\overline{\mathbf{b}}$ képlet kiszámításának idejét, ahol $\mathbf{B}$ egy 1000×1000-es mátrix, melyben minden főátlóbeli elem 2000 és az összes főátlón kívüli elem 1, továbbá $\overline{\mathbf{b}} = [1, 2, \ldots, 1000]^T$.

MATLAB megoldás ideje (s):...0.25s....., $\overline{\mathbf{x}} = \mathbf{B}^{-1}\overline{\mathbf{b}}$ képlet ideje: ...1s....

A `tic` és `toc` parancsok stopperként viselkednek. Azaz a `tic` parancs nullázza a stoppert a `toc` pedig leolvassa. A következő parancsokat írjuk a készenléti jelhez (vagy egy m-fájlba):

```
B=1999*eye(1000)+1;
b=[1:1000]';
tic; B\b; toc
tic; inv(B)*b; toc
```

A megoldási idő függ a használt számítógép sebességétől, de a két idő közti arány az kb. ugyanaz.

3. Feladat. Oldjuk meg az előző feladat egyenletrendszerét iterációs módszerrel! Végezzünk annyi iterációt, hogy kb. 4 tizedesjegyre pontos megoldást kapjunk!

A megoldásvektor 300.: .....0.0666..... és 600.: ...0.2167..... eleme.

A következő parancsokat írjuk a készenléti jelhez (vagy egy m-fájlba):

```
B=(B-diag(diag(B)))/2000;
b=[1:1000]'/2000;
x=zeros(1000,1);
for i=1:50 x=-B*x+b; end
x(300)
x(600)
```

Látható, hogy ha az 50-es értéket növeljük, az első négy zizedesjegy változatlan marad.

4. Feladat. A `math.bme.hu/~rhorvath/zh2mintfel.m` m-fájl egy olyan függvény, amely intervallumfelezéssel old meg egy egyenletet. A kezdeti lépésben az a pontban a függvényérték negatív, a b-ben pedig pozitív. Próbáljuk ki a program futtatását az

```
[x,hiba]=zh2mintfel('x^2-2',1,2,20)
```

paranccsal, amely láthatóan nem ad helyes eredményt. Keressük meg a hibás sort és javítsuk ki! A helyes sor:

$$x = a + (b - a)/2;$$

Adjuk meg a program segítségével a $2x^2 = \sin x$ egyenlet azon megoldását 10^{-10}-nél kisebb hibával, amely 0.2 és 0.8 közé esik : ...5.456746166032644e-013.....

Az intervallum felezőpontját kell minden lépésben maghatározni, így a helyes képlet $x = a + (b - a)/2$, azaz nem $-$ jel, hanem $+$ szerepel.

A következő parancsot gépeljük a készenléti jelhez:

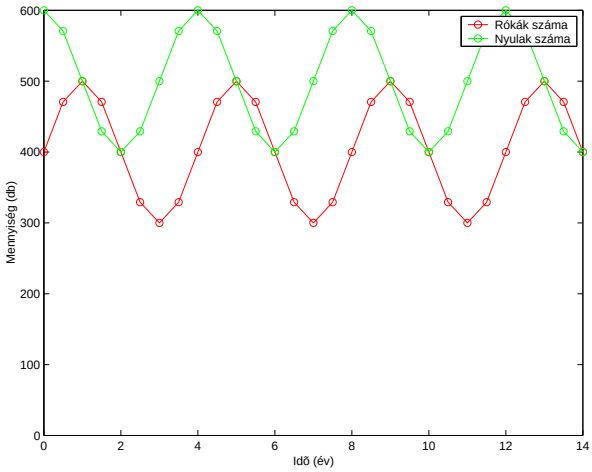
```
format long
[x,hiba]=zh2mintfel('2*x^2-sin(x)',0.2,0.8,30)
```

Mivel a program a hibát is kiírja láthatjuk, hogy a hiba a kívánt értéknél kisebb.

5. Feladat. Az $r(t) = 100(\sin(t\pi/2) + 4)$ függvény adja meg egy szigeten a rókák számának (db) közelítését az idő (év) függvényében, az $n(t) = 100(\cos(t\pi/2) + 5)$ pedig legyen egy hasonló függvény a nyulak számára vonatkozóan. Ábrázoljuk a két függvényt azonos koordinátarendszerben a $[0, 14]$ időintervallumon úgy, hogy csak félévenként számítjuk ki az állatok számát. A rókák grafikonja legyen piros, a nyulaké zöld és ezt az információt az ábrán is tüntessük fel! Adjunk címet a grafikonnak és felíratozzuk a tengelyeket! Az y-koordinátánál 0-tól 600-ig skálázzunk! Mentsük el az ábrát jpg kiterjesztéssel `rokanyul.jpg` néven!

A következő parancsokat gépeljük a készenléti jelhez, majd az ábrát exportáljuk jpg formátumba:

```
t=[0:0.5:14];
plot(t,100*(sin(t*pi/2)+4),'ro-',t,100*(cos(t*pi/2)+5),'go-')
axis([0,14,0,600])
legend('Rókák száma','Nyulak száma')
xlabel('Ido (év)')
ylabel('Mennyiség (db)')
```



6. Feladat. Készítsünk egy olyan MATLAB függvényt, `sormanipulacio.m` néven, amelynek bemenő adata egy mátrix (**A**), két sorindex (i,j) ill. egy valós szám (a), és kimenete az a mátrix, amely az **A** mátrixból úgy jön létre, hogy az i-edik sor a-szorosát kivonjuk a j-edik sorból!

A következő m-fájlt kell elkészíteni:

```
function A=sormanipulacio(A,i,j,a);
A(j,:)=A(j,:)-a*A(i,:);
```

7. Feladat. Illesszünk az (1,1), (3,13), (5,81), (7,253), (9,577) és (11,1101) pontokra legalább elsőfokú, másodfokú ill. harmadfokú polinomokat!

Elsőfokú: $105.2x - 2.9353$, másodfokú: $16x^2 - 86.8x + 95.8$ ....., harmadfokú: $\ldots x^3 - 2x^2 + x + 1 \ldots$

Adjuk meg a polinomok értékeit az $x = 4$ pontban!

Elsőfokú: $\ldots 127.266 \ldots$, másodfokú: $\ldots 4.6 \ldots$, harmadfokú: $\ldots 37 \ldots$

A következő parancsokat kell a készenléti jelhez gépelni (vagy egy m-fájlba beírni):

```
x=[1:2:11]
y=[1,13,81,253,577,1101]
z1=polyfit(x,y,1)
z2=polyfit(x,y,2)
z3=polyfit(x,y,3)
polyval(z1,4)
polyval(z2,4)
polyval(z3,4)
```

8. Feladat. Adjuk meg az $y' = (y^2+y)/x$, $y(1) = 1$ kezdetiértékfeladat megoldásának értékét az Euler-módszer segítségével az $x = 1.5$ pontban a $h = 1/10, 1/100$ és $1/1000$ lépéstávolságok használatával! Oldjuk meg az egyenletet az `ode45` parancs segítségével is!

Euler $y(1.5)$: h=1/10 $\ldots 2.56097028792132$ h=1/100 $\ldots 2.93737979986922$ h=1/1000 $\ldots 2.99344307739539$

ode45 $y(1.5)$: $\ldots 2.99999998641251$

Az Euler-módszert megvalósító program (a gyakorlaton is használt `eulermeth.m`, csak átírjuk úgy, hogy írja ki a függvényértéket az utolsó lépésnek megfelelő pontban, azaz hozzávettük a $yy(kmax + 1)$ sort.)

```
function eulermeth(fstr,x0,y0,h,kmax)
f=inline(fstr,'x','y')
x=x0;
yy(1)=y0;
for k=1:kmax
   y=yy(k);
   z=f(x,y);
   yy(k+1)=yy(k)+h*z;
   x=x+h;
end;
xx=[x0:h:x0+kmax*h];
plot(xx,yy)
yy(kmax+1)
```

Ezt használjuk az

```
eulermeth('(y^2+y)/x',1,1,0.1,5)
eulermeth('(y^2+y)/x',1,1,0.01,50)
eulermeth('(y^2+y)/x',1,1,0.001,500)
```

módon.

Az `ode45` parancsnál nem tudjuk a lépéstávolságot megadni, így csak futtatjuk a módszert és megnézzük milyen eredményt ad.

Ehhez kell egy m-fájl, ami az egyenlet jobb oldalát definiálja

```
function f=mode(x,y)
f=(y.*(y+1))./x;
```

majd ezután az alábbi parancsot futtatjuk

```
[t,x]=ode45('mode',[1,1.5],1)
```

Az x változó utolsó eleme adja a keresett értéket. Ez jóval pontosabb, mint az Euler-módszernél kapott érték.