Faragó István, Horváth Róbert

ÚTMUTATÓ A NUMERIKUS MÓDSZEREK JEGYZETHEZ KÉSZÜLT MATLAB M-FÁJLOK HASZNÁLATÁHOZ

ÁLTALÁNOS INFORMÁCIÓ

A Numerikus módszerek jegyzetünkhöz készült MATLAB m-fájlok szorosan kapcsolódnak a jegyzetben tárgyalt tananyaghoz, annak szinte elválaszthatatlan részét képezik. Az m-fájlok célja, hogy bemutassák a fontosabb algoritmusok gyakorlati megvalósítását, és lehetőséget biztosítsanak a hallgatóknak a numerikus módszerek kipróbálására és tulajdonságaik vizsgálatára.

Az m-fájlok önmagukban nem futtathatók, azok futtatásához arra van szükség, hogy a számítógépünkön telepítve legyen a MATLAB program. Ha a program telepítve van, akkor le kell töltenünk a honlapról azt az m-fájlt, amit ki szeretnénk próbálni, és el kell mentenünk a számítógépünk egyik mappájába. Ha ez a mappa nincs bent a MATLAB elérési útvonalai között, akkor a mappát a Fájl → Set path ... → Add folder menüben adhatjuk hozzá az elérési útvonalakhoz. Ezek után az m-fájlok úgy futtathatók, hogy az m-fájl nevét (a .m kiterjesztés nélkül) beírjuk a MATLAB fő ablakjában a készenléti jel (») után, majd <enter>-t ütünk.

A programok kódja és részletes leírása megtalálható a Numerikus módszerek jegyzetben is, de az alábbiakban röviden ismertetjük a programok működését.

A MELLÉKELT M-FÁJLOK

A jegyzethez az alábbi m-fájlokat mellékeltük:

- `gaussmodszer.m`: A program az `[U,L,x]=gaussmodszer(A,b)` módon hívható meg. A fájl a Gauss-módszert hajtja végre az Ax=b lineáris egyenletrendszer megoldására. Az A mátrix négyzetes mátrix, a b vektor pedig az A-nak megfelelő méretű oszlopvektor. A program főelemkiválasztást nem csinál. A kimenő paraméterek a Gauss-eljárás során létrejövő L és U mátrixok és az egyenletrendszer x megoldása.

- `hatvanymodszer.m`: A program az `[y,nu,iter]=hatvanymodszer(A,kmax,toll)` módon hívható meg. A fájl a hatványmódszert hajtja végre mátrixok sajátvektorainak és sajátértékeinek meghatározására. A bemenő paraméterek közül A az a mátrix, melynek a sajátértékeit és sajátvektorait keressük. kmax a maximális iterációszám és toll a toleranciaszint. A program futása akkor áll le, ha az iterációszám eléri a kmax értéket vagy két egymás utáni sajátértékközelítés eltérése kisebb, mint toll.

- `osszsimpson.m`: A program az `osszsimpson(a,b,n,fv)` módon hívható meg. A fájl az összetett Simpson-módszerrel közelíti az fv függvény integrálját az [a,b] intervallumon úgy, hogy n egyenlő részre osztja az [a,b] intervallumot. Az fv függvényt x függvényeként kell megadni a MATLAB-ban szokásos elemenkénti műveleti jelek segítségével. Pl. ha az x^2 függvény integrálját szeretnénk közelíteni a [0,1] intervallumon úgy, hogy 20 egyenlő részre osztjuk a [0,1] intervallumot, akkor a

```
osszsimpson(0,1,20,'x.^2')
```

parancsot kell használnunk.

- `dftvalos.m`: A program a `dftvalos(n,fx)` módon hívható meg. Az fx vektorban egy n+1 elemű sorvektort kell megadnunk. n+1 az alappontok száma, ahol n értéke páratlan kell legyen. A program a $(2k\pi/(n+1), fx_k)$ $(k = 0, \ldots, n+1)$ pontokon áthaladó trigonometrikus polinom együtthatóit adja meg.

- `shooting.m` és `bvpsee.m`: Itt a futtatandó fájl a `shooting.m` fájl, amely a belövéses módszerrel oldja meg a

$$\frac{d^2T}{dx^2} + D(Tinf - T) = 0, \quad T(0) = T0, \quad T(L) = T1$$

peremértékfeladatot. A futás után a program kiírja a numerikus megoldást, a maximum-normabeli hibát, és kirajzolja a pontos és közelítő megoldásokat.

A program a `shooting(Nx,zstart,deltaz,Nz)` módon futtatható, ahol Nx a [0,L] intervallum felosztásához szükséges osztásrészek száma, zstart a z(0) kezdeti első értéke (ami az ismeretlen T függvény gradiensét jelenti az x = 0 pontban), deltaz a különböző z(0) értékek meghatározására szolgáló megváltozás, Nz azt adja meg, hogy a kezdeti zstart értékektől jobbra és balra további Nz számú z(0) értékkel számolunk.

Az anyagi és peremfeltételek magában az m-fájlban változtathatók. A program alapértelmezésként a

```
T0 = 300; % bal oldali végponteli peremfeltétel
T1 = 400; % jobb oldali végponteli peremfeltétel
Tinf = 200; % külső hőmérséklet
D = 0.05; % állandó
L = 10; % a rúd hossza
xs = 0; % kezdőpont koordinátája
T = T0; % kezdeti feltétel
```

paraméterekkel számol.

A `shooting.m` minden lépésben meghívja a `[TL,Tvect,xvect] = bvpsee(Nx,z0)` programot, amely az éppen aktuális z(0) közelítéssel megoldja a megfelelő kezdetiérték-feladatot az explicit Euler-módszer segítségével. Az anyagi paramétereket ebben a fájlban is be kell állítani.

- `vdm1.m`, `pontosvdm.m`: A program a

$$T''(x) + c(T_\infty - T(x)) = 0, \; T(0) = T_0, \; T(L) = T_1$$

peremértékfeladatot oldja meg véges differencia módszerrel. T_0 és T_1 adja meg a peremfeltételeket. A program az `[xvect,Tmego]=vdm1(Nx)` módon hívható meg, ahol Nx jelenti a [0,L] intervallumon alkalmazott osztóintervallumok számát. A feladatot jellemző állandókat a programon belül kell megadni. Alapértelmezésben ezek

```
Ta   = 300;     % bal oldali végponteli peremfeltétel
Tb   = 400;     % jobb oldali végponteli peremfeltétel
Tinf = 200;     % külső hőmérséklet
c    = 0.05;    % állandó
L      = 10;    % a rúd hossza
```

A program az xmego vektorban tárolja el a rácspontok koordinátáit, a Tmego vektorban pedig a rácspontokbeli függvényközelítéseket. A futás végén a program kirajzolja a numerikus és a pontos megoldást is, ha ez utóbbit megadtuk a `pontosvdm.m` fájlban.

- `heatexp.m`: Ez a program a

$$\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0, \quad (x,t) \in (0, endx) \times (0, endt)$$

egyenletet oldja meg az

$$u(x,0) = init(x), \quad x \in (0, endx); \quad u(0,t) = bdry(1), \quad u(endx,t) = bdry(2), \quad t \in [0, endt]$$

kiegészítő feltételekkel a véges differenciák módszerével. A program hívása az

```
u = heatexp(endx, endt, Nx,q)
```

módon történik. Itt endx a térbeli egydimenziós tartomány megadására szolgál, a térbeli változó a [0,endx] intervallumon változik. Az endt változó az időbeli tartomány megadására szolgál, az időbeli változó a [0,endt] intervallumon változik. Nx a diszkretizácó során a térbeli osztásrészek száma, q a szokásos τ/h^2 hányados. (Emlékeztetőül: a konvergencia feltétele a $q \leq 0.5$ feltétel.)

A kezdeti és peremfeltételt a programon belül kell megadnunk. Alapesetben ezek az alábbiak.

```
init = sin(x);  % a kezdeti és a peremfeltétel
bdry=[0 0];
```

A program megadja a pontos megoldást közelítő u numerikus megoldást, kirajzolja a numerikus megoldás grafikonját és a pontos és numerikus megoldások eltérését. Ez utóbbihoz arra van szükség, hogy a programon belül megmondjuk a pontos megoldás értékeit a rácspontokban. Ez az adott kezdeti- és peremfeltételek mellett a programban alapesetben az alábbi módon néz ki.

```
upontos=zeros(N,J);
for i=1:N
    for n=1:J
        upontos(i,n)=exp(-t(n))*sin(x(i));
    end
end
```

- `poisson11exp.m`: A program a

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y)$$

egyenletet oldja meg inhomogén (első, avagy Dirichlet-féle) peremfeltétellel az egységnégyzeten a véges differencia módszerrel.

Futtatni az `uapprox = poisson11exp(n)` módon lehet, ahol az egyetlen n paraméter azt adja meg, hogy x és y irányban hány ekvidisztáns osztóintervallumot veszünk fel.

Az f függvény, a peremfeltételek és a pontos megoldás a programon belül adhatók meg. Alapértelmezésben ezek a következők:

```
f = inline('x^2 + y^2');
ux0 = inline('0');
ux1 = inline('0.5*x^2');
u0y = inline('sin(pi*y)');
u1y = inline('exp(pi)*sin(pi*y) + 0.5*y^2');
u = inline('exp(pi*x)*sin(pi*y) + .5*(x^2)*(y^2)'); % pontos megoldás
```

A program megadja a pontos megoldást közelítő uapprox numerikus megoldást, kirajzolja a numerikus megoldás grafikonját és a pontos és numerikus megoldások eltérését.