Közönséges differenciálegyenletek megoldása

Ebben a fejezetben

$$y' = f(t, y), t \in [0, T]$$
$$y(0) = y_0$$

alakú kezdetiérték-feladatok numerikus megoldásával foglalkozunk. A Matlabban erre vannak beépített függvények, amelyek előírt pontossággal megoldják a feladatot (ezekről később...) Ha azonban mi egy adott numerikus módszert szeretnénk alkalmazni megválasztott diszkretizációs paraméterekkel, akkor erre nekünk kell elkészítenünk a kódot. (Nincs pl. olyan beépített utasítás, amely az explicit Euler-módszerrel oldja meg a feladatot.) A következőkben sorra vesszük egy-egy fontosabb módszertípus programozását, és a félév végén ejtünk majd szót a beépített függvények nyújtotta lehetőségekről.

Először skaláris egyenletet tekintünk, majd rátérünk a rendszerek megoldására.

Amikor közelítőleg oldjuk meg a kezdetiérték-feladatot, akkor a feladat intervallumán definiálunk egy $\{t_0, t_1, \ldots, t_n\}$ rácsot, és ennek pontjaiban határozzuk meg a megoldás közelítését. Jelölés: $y_i \approx y(t_i)$, $h = t_{i+1} - t_i$. (Az egyszerűség kedvéért végig egyenlőközű rácson fogunk dolgozni.)

Az explicit Euler-módszer

Az explicit Euler módszer képlete:

$$t_{i+1} = t_i + h, \quad y_{i+1} = y_i + hf(t_i, y_i).$$

Ezt a következőképpen kódolhatjuk be:

```
function[t,y] = expeuler(diffegy, t0, y0, h, N)
t = zeros(N+1,1);
y = zeros(N+1,1);
t(1) = t0;
y(1) = y0;
for i=1:N
   t(i+1) = t(i) + h;
   y(i+1) = y(i) + h * diffegy(t(i),y(i));
end
```

Az `expeuler.m` függvénynek két kimenő paramétere van: a diszkretizált idővektor és

az ezen pontokban kiszámolt közelítő megoldások vektora. Az első bemenő paraméter egy függvény, aminek jelen esetben `diffegy` a neve, és ebben a függvényben írjuk meg a differenciálegyenlet jobb oldalán álló f függvényt. A második paraméter `t0`, ez a kezdeti időpontot jelöli, a harmadik a `t0` pontbeli `y0` kezdeti értéket jelenti. A következő paraméter a `h` lépéstávolság, majd végül `N` jelöli a megtett időlépések számát. A második és harmadik sorban vesszük fel a `t` és `y` vektorokat, amikben az értékeket először lenullázzuk, és a következő két sorban beállítjuk a kezdőértékeket. Utána következik lényegében a módszer algoritmusa: egy ciklus keretében az explicit Euler-módszer képlete szerint feltöltjük a `t` és `y` vektort.

A fenti programmal még nem tudjuk közvetlenül az adott differenciálegyenlet numerikus megoldását előállítani, ehhez szükségünk van az f függvényt definiáló `diffegy.m` függvény megadására. Ha az

$$y' = -y + t + 1, \quad t \in [0, T]$$
$$y(0) = 1$$

feladat megoldását szeretnénk előállítani, akkor a `diffegy` nevű alfüggvény elkészítéséhez nyissunk egy új m-fájlt, amibe írjuk a következőket:

```
function dydt = diffegy(t,y)
dydt = -y + t + 1;
```

Ha már megírtunk mindkét függvényt, futtassuk le a programunkat $h = 0,1$, $h = 0,01$ és $h = 0,001$ lépéstávolságokkal a [0, 1] intervallumon. A parancsablakba írjuk a következőt:

```
»[tee,yee] = expeuler(@diffegy, 0, 1, 0.1, 10)
```

Enter után megkapjuk a `tee` és `yee` kimenő vektorokat, amelyek a közelítések időbeli helyét és értékét tartalmazzák. (Ezeket természetesen máshogy is elnevezhetjük, csak jelezni akartuk, hogy nem szükséges a t és y változónevet használni, mint a függvény belsejében, de akár azokat is használhatjuk.) Ha ki is szeretnénk rajzoltatni, akkor a `plot(tee,yee)` paranccsal egy külön ablakban megjelenik a függvényünk. (A h = 0,01 és h = 0,001 lépésközökre értelemszerűen a

```
»[tee,yee] = expeuler(@diffegy, 0,1, 0.01, 100)
```

illetve a

```
»[tee,yee] = expeuler(@diffegy, 0, 1, 0.001, 1000)
```

parancsokat kell beírni.)

Ha másik differenciálegyenletet akarunk megoldani, amelynek a jobb oldalát pl. `diffegy1` néven készítjük el, akkor ehhez a fenti expeuler_sajat.m függvényben nem kell semmit átírnunk, a parancsablakból való futtatásnál viszont `@diffegy1`-gyel hívjuk.

Ha az explicit Euler-módszer hibájára vagyunk kíváncsiak, akkor a pontos megoldással szükséges a numerikus megoldásunkat összehasonlítani. A példánk pontos megoldása az

$$u(t) = \exp(-t) + t$$

függvény. Az `expeuler` függvényt egészítsük ki a következő sorokkal:

```
u=exp(-t)+t;
plot(t,u,'b',t,y,'r')
xlabel('t')
ylabel('y')
title('Explicit Euler-modszer')
legend('Pontos megoldas','Numerikus megoldas')
```

Ennek hatására a program ábrát is készít, amelyen összehasonlíthatjuk a pontos és az explicit Euler-módszerrel kapott közelítő megoldás görbéjét. Futtassuk le a módszert mindhárom fenti lépésközzel, és megfigyelhetjük, hogy a numerikus megoldás piros görbéje egyre közelebb kerül a pontos megoldást ábrázoló kék görbéhez. Természetesen ha másik feladatot oldunk meg, akkor az ahhoz tartozó pontos megoldást kell beírnunk, tehát az `u=exp(-t)+t` sort módosítani szükséges.

Az explicit Euler-módszer programozása úgy is megoldható, hogy nem használunk bemenő függvényt (igaz, az általában előnyösebb). Ehhez töröljük diffegy-et a bemenő adatok közül, és írjuk a program végére a következő sorokat:

```
function dydt=diffegy(t,y)
dydt=-y+t+1;
```

Itt tehát függvényen belül definiáltunk függvényt. Megjegyezzük, hogy ilyen függvényde-

finíció csak a függvény legvégén szerepelhet (ott akár egymás után több is), és szkriptben nem lehetséges!