Programcsomagok numerikus módszerekben

A relaxált Jacobi módszer

Márföldi Péter Bence
(ZXD9SV)

2014. május 3.

1. Feladat

17. *Relaxált Jacobi módszer*
Készítsen olyan függvényt, amely a Jacobi-féle iterációt alkalmazza, választható ω relaxációs paraméter(ek) mellett az $A\underline{x} = \underline{b}$ egyenletrendszer megoldására.

2. Matematikai háttér

2.1. Iterációs módszerek

Célunk az $A\underline{x} = \underline{b}$ egyenletrendszer megoldása. Iterációs módszereket akkor érdemes alkalmazni, ha az A mátrix úgynevezett ritka mátrix (a nem nulla elemek száma maximum $\sigma(n)$) és nagyméretű. A fentebb említett egyenletrendszert átírva a vele ekvivalens $\underline{x} = B\underline{x} + \underline{c}$ alakra visszavezethetjük az egyenletrendszer megoldását az $f(\underline{x}) = B\underline{x} + \underline{c}$ függvény fixpontjának a keresésére.

1. Tétel. *(Banach-féle fixponttétel $\mathbb{R}^n$-re)*
Legyen $f : \mathbb{R}^n \to \mathbb{R}^n$ függvény kontrakció, azaz $\exists q \in [0, 1)$:

$$\forall \underline{x}, \underline{y} \in \mathbb{R}^n - re : \|f(\underline{x}) - f(\underline{y})\| \leq q \cdot \|\underline{x} - \underline{y}\| \tag{1}$$

Ekkor:

1. *Az f függvénynek egyértelműen létezik fixpontja ($\underline{x}^*$, melyre $\underline{x}^* = f(\underline{x}^*)$)*
2. *$\forall \underline{x}^{(0)} \in \mathbb{R}^n$-re az $\underline{x}^{(k+1)} = f(\underline{x}^{(k)})$ iterációs sorozat konvergál az $\underline{x}^*$-hoz, azaz $\lim\limits_{k\to\infty} \underline{x}^{(k)} = \underline{x}^*$*

3. Továbbá érvényesek a következő hibabecslések:

$$\|\underline{x}^{(k)} - \underline{x}^*\| \leq q \cdot \|\underline{x}^{(k+1)} - \underline{x}^*\| \tag{2}$$

$$\|\underline{x}^{(k)} - \underline{x}^*\| \leq \frac{q}{1-q} \cdot \|\underline{x}^{(k)} - \underline{x}^{(k-1)}\| \leq \frac{q^k}{1-q} \cdot \|\underline{x}^{(1)} - \underline{x}^{(0)}\| \tag{3}$$

2. Tétel. *(Elégséges feltétel a konvergenciára)*
Ha az iterációs módszer B átmenetmátrixára igaz, hogy $\|B\| \leq 1$, akkor tetszőleges $\underline{x}^{(0)}$ kezdőértékre az $\underline{x}^{(k+1)} = B\underline{x}^{(k)} + \underline{c}$ iteráció konvergál az $A\underline{x} = \underline{b}$ egyenletrendszer megoldáshoz.

3. Tétel. *(Szükséges feltétel a konvergenciára)*
Tetszőleges $\underline{x}^{(0)}$-ból indított $\underline{x}^{(k+1)} = B\underline{x}^{(k)} + \underline{c}$ iteráció konvergál az $A\underline{x} = \underline{b}$ egyenletrendszer megoldáshoz $\iff \varrho(B) < 1$.

2.1.1. Az iterációs módszerek előnyei

A lineáris egyenletrendszerek megoldásánál direkt módszereket alkalmazva a megoldást véges sok művelet segítségével állítjuk elő. Ha a részeredményeink minden egyes lépésben pontosak, akkor végül a pontos megoldáshoz jutunk. Azonban felmerül a kérdése annak, hogy feltétlenül szükségünk van-e a pontos eredményre, hiszen a gyakorlatban sokszor már a bemeneti adatok is hibásak lehetnek. A direkt módszerek egyik fő hátránya, hogy rendkívül sok számolással járnak, vegyük például a Gauss-eliminációt, melynek műveletigénye: $\frac{2}{3}n^3 + \sigma(n^2)$. Ezzel szemben egy iterációs lépés mindössze egy mátrix-vektor szorzásból és egy vektorösszeadásból áll, aminek a műveletigénye $2n^2$. Ha nem akarjuk meghaladni a Gauss-elimináció műveletigényét, akkor az egyenletrendszer megoldása során maximum $\frac{n}{3}$ iterációs lépést hajthatunk végre. Ez egy 100×100-as mátrix esetén 33 lépést jelent. Mivel iterációs módszereket főleg ritka mátrixokra alkalmazunk a mátrix-vektor szorzást nagyban megkönnyíti a sok nullelem.

2.2. A Jacobi módszer

Tekintsük az $A \in \mathbb{R}^{n \times n}$ mátrix $L + D + U$ felbontását, ahol L a mátrix szigorú alsó része, D a diagonális része, U pedig a szigorú alsó része. Tehát $l_{ij} = a_{ij}$ (ha i<j), $d_{ii} = a_{ii}$ és $u_{ij} = a_{ij}$ (ha i>j), a többi elem nulla. Ezen mátrixok segítségével konstruáljuk meg a következő átalakítást:

$$A\underline{x} = \underline{b} \iff (L + D + U)\underline{x} = \underline{b} \iff D\underline{x} = -(L + U)\underline{x} + \underline{b} \tag{4}$$

$$D\underline{x} = -(L + U)\underline{x} + \underline{b} \iff -D^{-1}(L + U)\underline{x} + D^{-1}\underline{b} \tag{5}$$

A Jacobi iteráció átmenetmátrixa tehát $B_j = -D^{-1}(L+U)$, az iteráció pedig a következő:

$$\underline{x}^{(k+1)} = -D^{-1}(L+U)\underline{x}^{(k)} + D^{-1}\underline{b} \tag{6}$$

Koordinátás alakban az iteráció felírva:

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} - b_i \right] (i = 1 \dots n) \tag{7}$$

1. Definíció. *(Szigorúan diagonális dominancia)*
Az $A \in \mathbb{R}^{n \times n}$ mátrixot szigorúan diagonálisan dominánsnak nevezzük, ha:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \, (\forall i = 1 \dots n - re) \tag{8}$$

4. Tétel. *Ha az A mátrix szigorúan diagonálisan domináns $\Rightarrow \|B_j\|_\infty < 1$. ($\|B_j\|_1 < 1$)*

2.2.1. A Jacobi módszer előnye

Fentebb már beláttuk, hogy a direkt módszerekhez képest bizonyos esetekben mennyivel előnyösebb iterációs módszereket alkalmazni. A Jacobi módszer legjelentősebb előnye a Gauss-Seidel iterációval szemben az, hogy a koordinátákat akár párhuzamosan is számíthatjuk. Ez leginkább a két módszer koordinátás alakjából látszik:

1. Gauss-Seidel:

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \left[\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k+1)} - b_i \right] (i = 1 \dots n) \tag{9}$$

2. Jacobi:

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} - b_i \right] (i = 1 \dots n) \tag{10}$$

Jól látszik, hogy a Gauss-Seidel iterációnál a $(k+1)$-edik közelítővektor i-edik koordinátájának a kiszámításakor figyelembe vesszük a már kiszámított $1, 2, \dots, (i-1)$ koordinátákat és azokat felhasználjuk. Emiatt a Gauss-Seidel módszernél az új közelítővektor koordinátáit csak meghatározott sorrendben számíthatjuk. Ezzel ellentétben a Jacobi iterációnál ilyen megkötés nincs, az egyes koordinátákat akár párhuzamosan is kiszámíthatjuk.

2.3. A relaxált Jacobi módszer

A módszer alapját az előbb leírt Jacobi iteráció adja, azzal a különbséggel, hogy egy w paraméter bevezetésével próbáljuk finomítani a közelítést. Ha $0 < \omega < 1$, akkor alulrelaxálásról, ha pedig $1 < \omega$, akkor túlrelaxálásról beszélünk. Tekintsük a $D\underline{x} = -(L+U)\underline{x} + \underline{b}$ egyenletet valamint a $D\underline{x} = D\underline{x}$ egyenletet. Szorozzuk meg őket rendre ω illetve $(1-\omega)$ értékekkel, majd adjuk össze a két egyenletet:

$$D\underline{x} = (1-\omega)D\underline{x} - \omega(L+U)\underline{x} + \omega\underline{b} \tag{11}$$

Ezt követően D^{-1}-el szorozva kapjuk a csillapított Jacobi iteráció átmenetmátrixát:

$$B_j(\omega) = (1-w)I - wD^{-1}(L+U) \tag{12}$$

$\omega = 1$-et helyettesítve éppen a Jacobi iterációt kapjuk vissza. A koordinátás alak a következőképp írható fel:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} - \frac{\omega}{a_{ii}}\left[\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)} - b_i\right] (i = 1 \dots n) \tag{13}$$

5. Tétel. *Ha a Jacobi módszer konvergens, akkor $\omega \in (0,1]$-re a relaxált Jacobi módszer is konvergens.*

3. Megvalósítás

A program a relaxált Jacobi iterációt valósítja meg. Elsőként ellenőrzi a kapott paramétereket $(A, \underline{b}, k)$, majd a felhasználótól vár egy ω értéket. Egy felugró ablakban felajánlja lehetőségként a részeredmények kiíratását. Ezt követően elvégzi a Jacobi iterációt, visszaadva az egyenletrendszer megoldását valamint azt, hogy hány lépésből sikerült meghatározni a megoldást. A fentebb említett k paraméter opcionális, az iterációszámot lehet vele meghatározni. A program akkor is működik, ha $\underline{b}$ nem oszlopvektor, hanem mátrix. Végül megjeleníti az egyes lépések hibáit.

3.1. Paraméterek ellenőrzése

A program ellenőrzi a bemeneti paramétereket. Amennyiben az A mátrix nem négyzetes vagy a $\underline{b}$ vektor/mátrix hossza nem azonos az A mátrix hosszával, hibát dob. Ezt követően a felhasználótól bekéri az ω értékét. Elkészíti az A mátrix LDU felbontását, itt ellenőrzi, hogy az A főátlójában nem szerepel-e nulla érték, mivel ekkor a D mátrix szinguláris. Továbbá ellenőrzi, hogy a megadott ω értékre az iteráció konvergens lesz-e:

$$|\lambda_i| < 1 \ (\forall i = 1 \dots n) \Rightarrow J(\omega) \text{ konvergens.}$$

Ahol λ_i a $B_j(\omega)$ i-edik sajátértéke.

3.2. Az iteráció

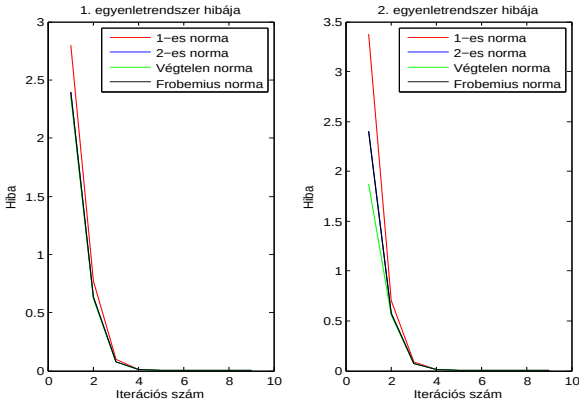
A program a fentebb leírt Jacobi relaxáció koordinátás alakját használja az $\underline{x}$ meghatározásához. Amennyiben három paraméterrel történ a hívás, akkor a k. iteráció után áll le a program futása. Ellenkező esetben a következő feltételig végzi a lépéseket:

$$e^{(k+1)} = \|\underline{x}^{(k+1)} - \underline{x}^{(k)}\| < \epsilon \tag{14}$$

Az iteráció addig fut, amíg az e értéke nem lesz stabil, tehát amíg két egymást követő iterációból származó $\underline{x}$-ek nem lesznek egymáshoz közeliek. Az implementáláskor lehetett volna a következő feltételt is alkalmazni a maradékvektor segítségével:

$$\frac{\|\underline{r}^{(k)}\|}{\|\underline{b}\|} = \frac{\|A\underline{x}^{(k)} - \underline{b}\|}{\|\underline{b}\|} < \epsilon \tag{15}$$

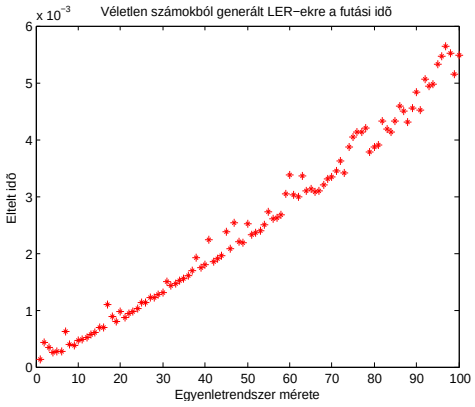
A megvalósításkor az ϵ értékét 10^{-6}-nak választottam. Továbbá a program futás közben grafikonon ábrázolja az aktuális közelítővektor hibáját. ($\|A\underline{x} - \underline{b}\|_p$-t, ahol $p \in \{1, 2, \infty, Frobemius\}$):



1. ábra. A $\begin{bmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 1 & 5 \\ 9 & 6 \end{bmatrix}$ lineáris egyenletrendszerek megoldásakor az iterációs lépések hibái (k paraméter nélkül).

4. Tesztesetek

Véletlen számokból generált egyenletrendszerekre az iteráció futási idejét a következő grafikon szemlélteti:



2. ábra. Az iteráció futási ideje $A \in \mathbb{R}^{i \times i}$ és $\underline{b}, \underline{x} \in \mathbb{R}^i$ egyenletrendszerekre $(i = 1 \ldots 100)$

4.1. Helyes tesztesetek

4.1.1. Numerikus módszerek példatár 4.1.4 fejezet 21. feladat

```
>> A = [4,-1,0;-1,4,-1;0,-1,4];
>> b = [1;1;1];
>> x = JacobiRelax(A,b)
Add meg a választott "w" értéket: 1
```

Eredmény:

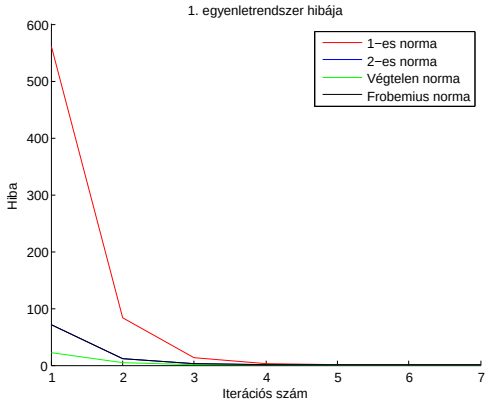
```
Elapsed time is 0.429422 seconds.

x =

    0.3571
    0.4286
    0.3571
```

4.1.2. $A \in \mathbb{R}^{100 \times 100}$ szigorúan diagonálisan domináns mátrixra

```
>> A = round(100*randn(100));
>> A = A+diag(sum(abs(A),2));
>> b = round(100*rand(100,1));
>> JacobiRelax(A,b);
Add meg a választott "w" értéket: 0.9
```

Eredmény:

```
Elapsed time is 0.125359 seconds.
```



3. ábra. Nagyméretű szigorúan diagonálisan domináns A mátrixra az egyes iterációs lépések hibái

4.1.3. $A \in \mathbb{R}^{5\times5}$ tridiagonális mátrixra

```
>> A = diag(rand(5,1))+diag(rand(4,1),1)+diag(rand(4,1),-1);
>> A = A+diag(sum(abs(A),2));
>> b = rand(5,1);
>> x = JacobiRelax(A,b);
Add meg a választott "w" értéket: 1.2
```

Eredmény:

```
Elapsed time is 0.129316 seconds.

x =

    1.1083
    0.2972
   -0.0160
    0.1329
    0.2874
```

4.1.4. $A \in \mathbb{R}^{5\times5}$ szimmetrikus A mátrixra

```
>> A = round(rand(5,1)*100);
>> A = toeplitz(A);
>> b = round(rand(5,1)*100);
```

```
>> x = JacobiRelax(A,b);
Add meg a választott "w" értéket: 1
```

Eredmény:

```
Elapsed time is 0.121552 seconds.

x =

    0.0611
    0.0409
    0.1705
    0.1426
   -0.0010
```

4.2. Hibás tesztesetek

4.2.1. Az A mátrix nem négyzetes

```
>> A = [1,2,3;4,5,6];
>> b = [1;2];
>> JacobiRelax(A,b);
```

Eredmény:

```
??? Error using ==> JacobiRelax at 6
A megadott "A" mátrix nem négyzetes!
```

4.2.2. Az A és a b vektor hossza nem azonos

```
>> A = [1,2,3;4,5,6;7,8,9];
>> b = [1;2];
>> JacobiRelax(A,b);
```

Eredmény:

```
??? Error using ==> JacobiRelax at 10
A megadott "A" mátrix és "b" vektor/mátrix hossza nem azonos!
```

4.2.3. A választott ω paraméterre az iteráció nem konvergens

```
>> A = [1,4,5;2,1,9;-2,2,1];
>> b = [1;2;3];
>> JacobiRelax(A,b);
Add meg a választott "w" értéket: 2.5
```

Eredmény:

```
??? Error using ==> JacobiRelax at 31
Az iteráció a megadott "w"-re nem konvergens!
```

5. Felhasznált irodalom

Hivatkozások

[1] **Dr. Krebsz Anna, Bozsik József,** Numerikus módszerek példatár, `http://www.inf.elte.hu/karunkrol/digitkonyv/Jegyzetek2010/Numerikus_modszerek_peldatar.pdf`

[2] **Sövegjártó András,** Numerikus analízis I., `http://numanal.inf.elte.hu/ soveg/oktanyagok/numanal1.pdf`

[3] **Jegyzet Dr. Krebsz Anna előadása alapján**

A. Függelék

Algorithm 1 Jacobi relaxáció

function [$\underline{x}$,itnum]=JacobiRelax(A,$\underline{b}$,ω)
tolerance=10^{-6};
error=∞;
itnum=0;
while error>tolerance **do**
  $\underline{x}^{old} = \underline{x}$;
  for i=1 **to** size(A) **do**

$$x_i = (1-\omega)x_i - \frac{\omega}{a_{ii}} \left[\sum_{\substack{j=1 \\ j \neq i}}^{size(A)} a_{ij}x_j - b_i \right];$$

  end for
  itnum=itnum+1;
  error=$\|\underline{x} - \underline{x}^{old}\|$;
end while
return [$\underline{x}$, itnum];