OFFLİNE DERS

2020-2021 BAHAR DÖNEMİ

YMH214
SAYISAL ANALIZ
LAB. DERSİ

7.DERS

Arş. Gör. Alev KAYA

Lineer Denklem Sistemlerinin Çözümü

- Sayısal Yöntemler
- A- Jocobi İterasyon Yöntemi
- B- Gauss Seidel Yöntemi
- LAB: Jacobi yöntemi Matlab örnek programı

A- Jocobi(Basit) İterasyon Yöntemi

- **Ardışık(iteratif-dolaylı) yöntemlerde** izlenen yol; denklem sistemlerine yaklaşık bir çözüm takımı ile başlamak ve belirli bir algoritmayı tekrarlayarak, gerçek çözümü **en az hata ile** hesaplamaktır.

- Böylece hem algoritmalarının **kolayca hesaplanabilir** olmaları hem de **yuvarlama hatalarının en az ve iterasyon sayısı arttıkça hata birikimi olmaması** açısından avantajlıdır.

- Bu **yöntemlerin en önemli problemi**; **YAKINSAMA** PROBLEMİDİR.

- Bazı tip problemlerde Jacobi iterasyonu, bazı tip problemlerde ise Gauss-Seidel daha çabuk yakınsar**(GAUSS-SEİDEL, DAHA HIZLI AYNI PROBLEM İÇİNDE).**

Jacobi iterasyonunun yakınsayabilmesi için A matrisinin ana köşegen üzerindeki elemanlarının mutlak değerlerinin bir **koşulu** (strictly diagonally dominant) yerine getirmesi gerekmektedir; A matrisinin i. satırının köşegen üzerindeki değeri (a_{ii})'nin mutlak değeri A matrisinin i. satırındaki tüm elemanların mutlak değerlerinin toplamından büyük olmalıdır.

Aşağıda verilen lineer denklem sistemi Jacobi iterasyonu ile çözülsün;

$$\begin{aligned} 4x_1 - 2x_2 + x_3 &= 6 \\ 4x_1 - 8x_2 + x_3 &= -20 \\ -x_1 + x_2 + 5x_3 &= 11 \end{aligned} \Rightarrow \begin{bmatrix} 4 & -2 & 1 \\ 4 & -8 & 1 \\ -1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -20 \\ 11 \end{bmatrix} \Rightarrow AX = B$$

Yukarıdaki eşitliklerden;

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

elde edilir. Eğer program $x_1^{(0)} = 1;\ x_2^{(0)} = 2;\ x_3^{(0)} = 1$ başlangıç değerleri ile başlar ise;

$$x_1^{(1)} = \frac{6 + 2*2 - 1}{4} = 2.25 \quad ; \qquad x_2^{(1)} = \frac{20 + 4*1 + 1}{8} = 3.125$$

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

elde edilir. Eğer program $x_1^{(0)} = 1;\ x_2^{(0)} = 2;\ x_3^{(0)} = 1$ başlangıç değerleri ile başlar ise;

$$x_1^{(1)} = \frac{6 + 2*2 - 1}{4} = 2.25 \quad ; \qquad x_2^{(1)} = \frac{20 + 4*1 + 1}{8} = 3.125 \quad ;$$

$$x_3^{(1)} = \frac{11 + 1 - 2}{5} = 2$$

birinci iterasyon sonunda bulunan değerler ile ikinci iterasyona gidilir ve bu işlem bu şekilde devam ettirilir ise Tablo 11.1'de gösterildiği gibi 15. iterasyonda (verilen tolerans için);

$$x_1^{(15)} = 3.1745\ ;\ x_2^{(15)} = 4.3333\ ;\ x_3^{(15)} = 1.9683$$

değerleri X çözüm matrisini belirler. İterasyonu durdurmak için kullanılabilecek

bir kriter j. iterasyon sonunda elde edilen $X^{(j)}$ çözüm matrisi ile bundan bir önceki (j-1). iterasyonda elde edilen $X^{(j-1)}$ çözüm matrisi arasındaki farkın mutlak değer olarak başlangıçta belirlenen epsilon değerinden küçük olmasıdır. Buna benzer bir çok durdurma kriteri geliştirilebilir. Ardışık yaklaşımlar için diğer önemli bir nokta da $X^{(0)}$ başlangıç değerlerinin abartılı olarak seçilmesi durumunda yakınsamanın gecikebileceği gerçeğidir. Ardışık yöntemler için verilen sisteme ilişkin uygun değerler (daha önceden) elde edilebilmiş ise bu değerlerin kullanılması yakınsamanın kolaylaşması açısından çok uygun olur.

Tablo 11.1

İterasyon sayısı	x_1	x_2	x_3
1	2.2500	3.1250	2.0000
2	2.5625	3.8750	2.0250
3	2.9312	4.0344	1.9375
4	3.0328	4.2078	1.9794
5	3.1091	4.2638	1.9650
6	3.1407	4.3002	1.9690
7	3.1578	4.3165	1.9681
8	3.1662	4.3249	1.9683
9	3.1704	4.3291	1.9683
10	3.1725	4.3312	1.9683
11	3.1736	4.3323	1.9683
12	3.1741	4.3328	1.9683
13	3.1743	4.3331	1.9683
14	3.1745	4.3332	1.9683
15	3.1745	4.3333	1.9683

$$|4| > |-2| + |1|$$

$$|-8| > |4| + |1|$$

$$|5| > |-1| + |1|$$

JACOBI_BASIT.m

```
% A.X=B lineer matris eşitliğinin JACOBI iterasyonu yöntemi ile çözümü
% A; matrisi N*N boyutunda tekil olmayan katsayılar matrisidir.
% B; matrisi N*1 boyutundadır.
% X; matrisi 1*N boyutunda çözüm matrisinin evriğidir.
% P; matrisi N*1 boyutunda başlangıç değerler matrisidir.
% delta; iterasyonun son iki adımı arasındaki müsade edilebilen fark değeri
% maxiter; max. iterasyon sayısı.Eğer kullanıcı max. sayısına kadar
% iterasyon yakınsamaz ise programı durdurmak için bu değeri kullanılır.

A=[4 -2 1;4 -8 1;-1 1 5];
B=[6;-20;11];
P=[1;2;1];
N=length(B);
X=zeros(1,N);
maxiter=30;
delta=0.00001;
for k=1:maxiter
    for J=1:N
        X(J)=(B(J)-A(J,[1:J-1,J+1:N])*P([1:J-1,J+1:N]))/A(J,J);
    end
    hata=abs(norm(X'-P));
    hatatek=hata/(norm(X)+eps);
    P=X';
    if(hata<delta)|(hatatek<delta)
        'iterasyon maxiterden önce sona erdi'
        break
    end
end
display('iterasyon sayisi=');
display(k-1);
X=X'
```

Command Window

```
>> JACOBI_BASIT

ans =

    'iterasyon maxiterden önce sona erdi'

iterasyon sayisi=
    15


x =

    3.1746
    4.3333
    1.9683

fx >>
```

B- Gauss Seidel Yöntemi

Jacobi iterasyonunda $X^{(0)}$ başlangıç değerleri sırayla;

$$x_1^{(1)} = \frac{6 + 2x_2^{(0)} - x_3^{(0)}}{4}; \quad x_2^{(1)} = \frac{20 + 4x_1^{(0)} + x_3^{(0)}}{8}; \quad x_3^{(1)} = \frac{11 + x_1^{(0)} - x_2^{(0)}}{5}$$

eşitliklerinde yerlerine konulmuştu. Gauss-Seidel yönteminde ise ilk eşitlikte (x_1'e ilişkin) $x_2^{(0)} = 2;\ x_3^{(0)} = 1$ değerleri yerlerine konur. Elde edilen $x_1^{(1)} = 2.25$ değeri (yine aynı iterasyon içindeki)

$$x_2^{(1)} = \frac{20 + 4x_1^{(1)} + x_3^{(0)}}{8} = \frac{20 + 4 * 2.25 + 1}{8} = 3.75$$

eşitliğinde yerine konulur. Yukarıda elde edilen $x_1^{(1)}$ ve $x_2^{(1)}$ değerleri ise $x_3^{(1)}$ eşitliğinde yerine konulursa;

eşitliğinde yerine konulur. Yukarıda elde edilen $x_1^{(1)}$ ve $x_2^{(1)}$ değerleri ise $x_3^{(1)}$ eşitliğinde yerine konulursa;

$$x_3^{(1)} = \frac{11 + x_1^{(1)} - x_2^{(1)}}{5} = 1.9$$

bulunur. Aynı problem Jacobi iterasyonu ile çözüldüğünde ilk iterasyon sonunda;

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.125 \; ; \; x_3^{(1)} = 2$$

bulunmuştu. Gauss-Seidel yaklaşımında ise

$$x_1^{(1)} = 2.25 \; ; \; x_2^{(1)} = 3.75 \; ; \; x_3^{(1)} = 1.9$$

bulunur. Böyle bir yaklaşım yakınsamayı çabuklaştırarak hesaplamanın daha çabuk bitmesini temin eder. Gauss-Seidel yaklaşımında da Jacobi yaklaşımında olduğu gibi, A matrisinin ana köşegen elemanlarının mutlak değerleri, aynı satır üzerinde yer alan diğer elemanların mutlak değerlerinin toplamından daha büyük olmalıdır. Aksi halde yakınsama sağlanamaz.

Yukarıdaki işlem diğer iterasyonlar içinde yapıldığında elde edilen sonuçlar Tablo 11.2'de gösterilmiştir;

Tablo 11.2

İterasyon sayısı	x_1	x_2	x_3
1	2.2500	3.7500	1.9000
2	2.9000	4.1875	1.9425
3	3.1081	4.2969	1.9622
4	3.1579	4.3242	1.9667
5	3.1704	4.3311	1.9679
6	3.1736	4.3328	1.9682
7	3.1743	4.3332	1.9682
8	3.1745	4.3333	1.9682

Tablo 11.1 ile Tablo 11.2 arasında bir karşılaştırma yapıldığında Gauss-Seidel yaklaşımında çok daha az iterasyon sayısı ile yakınsama sağlandığı görülmektedir. Her iki yaklaşımın aynı probleme aynı ilk koşullar altında uygulandığı da unutulmamalıdır.

Editor - C:\Matlab_Dersler\gauss_seidel.m

JACOBI_BASIT.m | gauss_seidel.m

```
% A.X=B lineer matris eşitliğinin GAUSS_SEİDEL iterasyonu yöntemi ile çözümü
% X; matrisi 1*N boyutunda çözüm matrisidir.
% P; matrisi N*1 boyutunda başlangıç değerler matrisidir.
% delta; iterasyonun son iki adımı arasındaki müsade edilebilen fark değeri
% maxiter; max. iterasyon sayısı.Eğer kullanıcı max. sayısına kadar
% iterasyon yakınsamaz ise programı durdurmak için bu değeri kullanılır.
A=[4 -2 1;4 -8 1;-1 1 5];
B=[6;-20;11];
P=[1;2;1];
N=length(B);
X=zeros(1,N);
maxiter=30;
delta=0.00001;
for k=1:maxiter
    for J=1:N
        if J==1
        X(1)=(B(1)-A(1,2:N)*P(2:N))/A(1,1);
        elseif J==N
            X(N)=(B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
        else
            X(J)=(B(J)-A(J,1:J-1)*X(1:J-1)-A(J,J+1:N)*P(J+1:N))/A(J,J);
        end
```

```
22 -            end
23 -        end
24 -        hata=abs(norm(X'-P));
25 -        hatatek=hata/(norm(X)+eps);
26 -        P=X';
27 -        if(hata<delta)|(hatatek<delta)
28 -            'iterasyon maxiterden önce sona erdi'
29 -            break
30 -        end
31 -        X;
32 -    end
33 -    display('iterasyon sayisi=');
34 -    display(k-1);
35 -    X=X'
36
```

mmand Window

Command Window

```
>> gauss_seidel

ans =

     'iterasyon maxiterden önce sona erdi'

iterasyon sayisi=
       8


X =

    3.1746
    4.3333
    1.9683

fx >>
```