ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
Mühendislik Mimarlık Fakültesi
İnşaat Mühendisliği Bölümü
E-Posta: ogu.ahmet.topcu@gmail.com
Web: http://mmf2.ogu.edu.tr/atopcu

Bilgisayar Destekli

Nümerik Analiz

Ders notları 2014

Ahmet TOPÇU

m=n

k	x_1	x_2	x_3	x_4
0	0	0	0	0
1	0.2500	0.5000	0	0.2500
2	0.1250	0.3750	-0.1250	0.1250
3	0.1875	0.4375	-0.0625	0.1875
4	0.1563	0.4063	-0.0938	0.1563
5	0.1719	0.4219	-0.0782	0.1719
6	0.1641	0.4141	-0.0860	0.1641
7	0.1680	0.4180	-0.0821	0.1680
8	0.1660	0.4160	-0.0840	0.1660
9	0.1670	0.4170	-0.0830	0.1670
10	0.1665	0.4165	-0.0835	0.1665
11	0.1668	0.4168	-0.0833	0.1667
12	0.1666	0.4166	-0.0834	0.1666
13	0.1667	0.4167	-0.0833	0.1667

Başlangıç değerleri

$Max|x_i^k - x_i^{k-1}| = |x_4^2 - x_4^1| = |0.1250 - 0.2500| = 0.1250 > \varepsilon = 0.0001$ olduğundan **iterasyona devam!**

$|0.1875 - 0.1250| = 0.0625 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1660 - 0.1680| = 0.0020 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1668 - 0.1665| = 0.0003 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1667 - 0.1667| = 0.0001 = \varepsilon = 0.0001$, **iterasyon durduruldu !**

İterasyon no

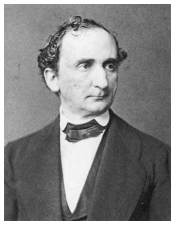
Çözüm

7

DENKLEM SİSTEMİ ÇÖZÜMÜ, İTERASYON YÖNTEMLERİ:

- Jacobi
- Gauss-Seidel
- GG(Conjugate Gradient)

Carl Gustav Jakob **JACOBI** (1804-1851)



Philipp Ludwig von **SEIDEL** (1821–1896)

7. İTERASYON YÖNTEMLERİ: JACOBI, GAUSS-SEIDEL, CG

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ &\cdot \\ &a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n \end{aligned} \quad \rightarrow \quad \underline{A}\,\underline{x}=\underline{b} \qquad (7.1)$$

$\underline{A}\,\underline{x}=\underline{b}$ doğrusal denklem sistemi, n çok büyük ise ve $\underline{A}$ çok seyrek ise iterasyon yöntemleri ile çözülür. Çünkü direkt metodlar çok fazla bellek, dört işlem ve hesap süresi gerektirirler. Biriken yuvarlama hataları çözümü tehlikeye sokar. İterasyon yöntemlerinde $\underline{A}$ nın elemanları değişmez, bu nedenle $\underline{A}$ nın sadece sıfırdan farklı elemanları depolanır, sıfır ile dört işlem yapılmaz. Hem bellek hem hesap süresi hem de yuvarlama hataları önemli miktarda azalır. Uygulamada karşılaşılan $\underline{A}$ matrisi genelde çok seyrektir, sıfırdan farklı eleman oranı yaklaşık %1-5 civarındadır.

İterasyon (kelime anlamı: tekrarlama, yenileme) yöntemleri $\underline{A}\,\underline{x}=\underline{b}$ denklem sisteminin çözümünü aşağıdaki ilkeye göre çözmeye çalışırlar:

Başlangıç: $\underline{x}$ bilinmeyenler vektörü için bir başlangıç değeri tahmin edilir, örneğin: $\underline{x}=\underline{x}^{(0)}=[0\ 0\ \ldots\ 0]^T$

İterasyon:
1 adım: hata vektörü $\underline{h}=\underline{b}-\underline{A}\,\underline{x}$ küçülecek şekilde yeni bir yaklaşım bulunur: $\underline{x}=\underline{x}^{(1)}$
2.adım: hata vektörü $\underline{h}=\underline{b}-\underline{A}\,\underline{x}$ küçülecek şekilde yeni bir yaklaşım bulunur: $\underline{x}=\underline{x}^{(2)}$
…
k.adım: hata vektörü $\underline{h}=\underline{b}-\underline{A}\,\underline{x}$ küçülecek şekilde yeni bir yaklaşım bulunur: $\underline{x}=\underline{x}^{(k)}$
…
İterasyonu durdurma koşulu sağlanıncaya kadar tekrarlanır.

Kaç adım sonunda çözüme varılabileceği, k nın üst sınırı, önceden kestirilemez. İterasyonu durdurma koşuluna gerek vardır. Durdurma koşulu için farklı yaklaşımlar vardır, bazı örnekler:
1. iterasyon sayısının üst sınırı, max k
2. $\underline{h}=\underline{b}-\underline{A}\,\underline{x}^{(k)}$ olmak üzere, $\sqrt{\underline{h}^T\underline{h}} \le \varepsilon$
3. $\underline{h}=\underline{x}^k-\underline{x}^{(k-1)}$ olmak üzere, $\sqrt{\underline{h}^T\underline{h}} \le \varepsilon$
4. $\max|x_i^k - x_i^{k-1}| \le \varepsilon$, i=1, 2, …, n

Burada ε kabul edilebilir hata değeridir, $\underline{x}$ vektörünün fiziksel anlamına bağlı olarak, ε=0.1 - $\varepsilon=10^{-14}$ gibi tipik bir değer olabilir.

Çok sayıda iterasyon metodu vardır. Burada **JACOBI**, **GAUSS-SEIDEL** ve **CG** (Conjugate Gradient) metotlarına yer verilecektir. **JACOBI** ve **GAUSS-SEIDEL** metodları katsayılar matrisi simetrik veya simetrik olmayan denklem sistemlerinde kullanılabilir. **CG** Metodu ise sadece katsayılar matrisi simetrik ve pozitif tanımlı denklem sistemleri için kullanılır.

JACOBI metodu[1]:

İterasyona başlamadan önce, 7.1 denklem sistemi diyagonal elemanları $a_{ii}\neq 0$ olacak şekilde yeniden düzenlenir. Bunun için gerekirse satırların yerleri değiştirilir. 7.1 sisteminden x_1, x_2, …, x_n çekilerek

$$\begin{aligned} x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n) \\ x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2n}x_n) \\ &\cdot \\ x_n &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - a_{n3}x_3 - \ldots) \end{aligned} \qquad (7.2)$$

şeklinde yazılır.

[1] Carl Gustav Jakob **JACOBI** (1804 - 1851), Alman: 1845 de yayınlandı.

İterasyona x_1, x_2, x_3, …, x_n bilinmeyenleri için, fiziksel anlamına göre, bir başlangıç değeri tahmin edilerek başlanır. Herhangi bir tahmin yapılamıyorsa $x_1= x_2= \ldots= x_n=0$ veya $x_1 = \frac{b_1}{a_{11}}, x_2 = \frac{b_2}{a_{22}}, \ldots, x_n = \frac{b_n}{a_{nn}}$ alınabilir. Bu x_i değerleri 7.2 in sağ tarafında yerine konur, soldaki yeni x_i değerleri hesaplanır. İterasyonu durdurma koşulu kontrol edilir, sağlanıyorsa iterasyon durdurulur. Sağlanmıyorsa yeni x_i değerleri 7.2 nin sağ tarafında yerine konur ve soldaki yeni x_i değerleri hesaplanır. İterasyonu durdurma koşulu sağlanıncaya kadar bu işlem tekrarlanır.

Örnek:

$$\underline{A}\underline{x} = \underline{b} \rightarrow \begin{bmatrix} 4 & 1 & 1 & 0 \\ 1 & 4 & 0 & 1 \\ 1 & 0 & 4 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \quad \underline{x} = ?$$

Denklem sisteminin direkt yöntemlerle çözümü $\underline{x}^T$=[0.1667 0.4167 -0.0833 0.1667] dir. Çözümde ondalık sayıdan sonra 4 hane verilmiştir. Aynı denklem sistemini JACOBI iterasyonu ile çözelim. Denklem sistemini

$$x_1 = \frac{1}{4}(1 - x_2 - x_3)$$
$$x_2 = \frac{1}{4}(2 - x_1 - x_4)$$
$$x_3 = \frac{1}{4}(-x_1 - x_4)$$
$$x_4 = \frac{1}{4}(1 - x_2 - x_3)$$

şeklinde yazalım. i. bilinmeyenin k. Ve k-1. adımda hesaplanan iki değerinin farkı $x_i^k - x_i^{k-1}$ olmak üzere, $Max|\, x_i^k - x_i^{k-1}| \le \varepsilon$ koşulu **sağlanınca iterasyonu durduralım**. ε=0.0001 seçelim. Çözümde 4 ondalık hane kullanalım. Başlangıç için $\underline{x} = \underline{x}^{(0)} = [0\ 0\ 0\ 0]^T$ alalım.

k	x_1	x_2	x_3	x_4
0	0	0	0	0
1	0.2500	0.5000	0	0.2500
2	0.1250	0.3750	-0.1250	0.1250
3	0.1875	0.4375	-0.0625	0.1875
4	0.1563	0.4063	-0.0938	0.1563
5	0.1719	0.4219	-0.0782	0.1719
6	0.1641	0.4141	-0.0860	0.1641
7	0.1680	0.4180	-0.0821	0.1680
8	0.1660	0.4160	-0.0840	0.1660
9	0.1670	0.4170	-0.0830	0.1670
10	0.1665	0.4165	-0.0835	0.1665
11	0.1668	0.4168	-0.0833	0.1667
12	0.1666	0.4166	-0.0834	0.1666
13	0.1667	0.4167	-0.0833	0.1667

Başlangıç değerleri

$Max|\, x_i^k - x_i^{k-1}| = |x_4^2 - x_4^1| = |0.1250 - 0.2500| = 0.1250 > \varepsilon = 0.0001$ olduğundan **iterasyona devam!**

$|0.1875 - 0.1250| = 0.0625 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1660 - 0.1680| = 0.0020 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1668 - 0.1665| = 0.0003 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1667 - 0.1666| = 0.0001 = \varepsilon = 0.0001$, **iterasyon durduruldu !**

İterasyon no

13. iterasyon sonunda bulunan çözüm

Çözüm: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.1667 \\ 0.4167 \\ -0.0833 \\ 0.1667 \end{bmatrix}$

GAUSS-SEIDEL metodu[1]:

İterasyona başlamadan önce, 7.1 denklem sistemi diyagonal elemanları $a_{ii}\neq 0$ olacak şekilde düzenlenir. Bunun için gerekirse satırların yerleri değiştirilir. 7.1 sisteminden x_1, x_2, …, x_n çekilerek

$$x_1 = \frac{1}{a_{11}}(b_1\text{-}a_{12}x_2 - a_{13}x_3\text{-}\ldots\text{-}a_{1n}x_n)$$
$$x_2 = \frac{1}{a_{22}}(b_2\text{-}a_{21}x_1 - a_{23}x_3\text{-}\ \ldots - a_{2n}x_n) \qquad (7.3)$$
.
$$x_n = \frac{1}{a_{nn}}(b_n\text{-}a_{n1}x_1 - a_{n2}x_2 - a_{n3}x_3\text{-}\ldots)$$

şeklinde yazılır.

İterasyona x_1, x_2, x_3, …, x_n bilinmeyenleri için, fiziksel anlamına göre, bir başlangıç değeri tahmin edilerek başlanır. Herhangi bir tahmin yapılamıyorsa $x_1= x_2= \ldots= x_n=0$ veya $x_1 = \frac{b_1}{a_{11}}, x_2 = \frac{b_2}{a_{22}}, \ldots, x_n = \frac{b_n}{a_{nn}}$ alınabilir.

x_i değerleri 7.3 ün 1. denkleminin sağ tarafında yerine konur, x_1 in yeni değeri bulunur. x_1 in yeni değeri ve x_3, x_4, …, x_n nin önceki değerleri 2. denklemin sağ tarafında yerine konur, x_2 nin yeni değeri bulunur. x_1 ve x_2 nin yeni değeri ile x_4, …, x_n nin önceki değerleri 3. denklemin sağ tarafında yerine konur, x_2 nin yeni değeri bulunur. … x_1, x_2, …, x_{n-1} in yeni değerleri n. denklemin sağ tarafında yerine konur, x_n nin yeni değeri bulunur.

İterasyonu durdurma koşulu kontrol edilir, sağlanıyorsa iterasyon durdurulur. Sağlanmıyorsa son x_i değerleri ile işlem tekrarlanır.

GAUSS-SEIDEL metodu ile **JACOBI** metodu temelde aynıdır. Tek fark şudur: **GAUSS-SEIDEL** metodunda x_i nin her yeni değeri hemen kullanılır.

Örnek:

$$\underline{A}\underline{x} = \underline{b}, \quad \begin{bmatrix} 4 & 1 & 1 & 0 \\ 1 & 4 & 0 & 1 \\ 1 & 0 & 4 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \quad \underline{x} = ?$$

Denklem sisteminin direkt yöntemlerle çözümü $\underline{x}^T$=[0.1667 0.4167 -0.0833 0.1667] dir. Çözümde ondalık sayıdan sonra 4 hane verilmiştir. Aynı denklem sistemini GAUSS-SEIDEL iterasyonu ile çözelim. Denklem sistemini

$$x_1 = \frac{1}{4}(1 - x_2 - x_3)$$
$$x_2 = \frac{1}{4}(2 - x_1 - x_4)$$
$$x_3 = \frac{1}{4}(-x_1 - x_4)$$
$$x_4 = \frac{1}{4}(1 - x_2 - x_3)$$

şeklinde yazalım. i. bilinmeyenin k. ve k-1. adımda hesaplanan iki değerinin farkı $x_i^k - x_i^{k-1}$ olmak üzere, $Max|\, x_i^k - x_i^{k-1}| \leq \varepsilon$ koşulu **sağlanınca iterasyonu durduralım**. ε=0.0001 seçelim. Çözümde 4 ondalık hane kullanalım. Başlangıç için $\underline{x} = \underline{x}^{(0)} = [0\ 0\ 0\ 0]^T$ alalım.

[1] Philipp Ludwig von **SEIDEL** (1821 – 1896), Alman: 1874 yılında yayınlandı.

k	X_1	X_2	X_3	x_4
0	0	0	0	0
1	0.2500	0.4375	-0.0625	0.1563
2	0.1563	0.4219	-0.0782	0.1641
3	0.1641	0.4180	-0.0821	0.1660
4	0.1660	0.4170	-0.0830	0.1665
5	0.1665	0.4168	-0.0833	0.1666
6	0.1666	0.4167	-0.0833	0.1667
7	0.1667	0.4167	-0.0834	0.1667

Başlangıç değerleri

$Max|x_i^k - x_i^{k-1}| = |x_1^2 - x_1^2| = |0.1563 - 0.2500| = 0.0937 > \varepsilon = 0.0001$ olduğundan **iterasyona devam!**

$|0.1641 - 0.1563| = 0.0078 > \varepsilon = 0.0001$, **iterasyona devam!**

$|0.1667 - 0.1666| = 0.0001 = \varepsilon = 0.0001$, **iterasyonu durdur!**

İterasyon adımları

7. iterasyon sonunda bulunan çözüm

Çözüm: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0.1667 \\ 0.4167 \\ -0.0834 \\ 0.1667 \end{bmatrix}$

Aitken iterasyon hızlandırıcısı

Yukarıdaki örneklerden görüldüğü gibi, iterasyon gerçek çözüme oldukça yavaş yakınsamaktadır. JACOBI ve GAUSS-SEIDEL iterasyonları doğrusal yaklaşım sergilerler. Doğrusal yaklaşımlı iterasyon metotlarında AITKEN[1] hızlandırıcısı kullanılarak iterasyon hızlandırılabilir. Herhangi bir x_i bilinmeyenin birbirini izleyen üç iterasyon adımı sonunda bulunan $x_i^{k-2}, x_i^{k-1}, x_i^k$ değerleri kullanılarak x_i^k nın değeri iyileştirilebilir. **AITKEN**'e göre x_i^k nın iyileştirilmiş değeri $x_i^k \leftarrow x_i^k - \dfrac{(x_i^k - x_i^{k-1})^2}{x_i^k - 2x_i^{k-1} + x_i^{k-2}}$ dir.

Alexander Craig **Aitken**(1895-1967)

Bu formülü kullanarak aşağıdaki örneği JACOBI metodu ile çözelim:

$$\underline{A}\underline{x} = \underline{b}, \quad \begin{bmatrix} 4 & 1 & 1 & 0 \\ 1 & 4 & 0 & 1 \\ 1 & 0 & 4 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \quad \underline{x} = ? \rightarrow$$

$$x_1 = \frac{1}{4}(1 - x_2 - x_3)$$
$$x_2 = \frac{1}{4}(2 - x_1 - x_4)$$
$$x_3 = \frac{1}{4}(-x_2 - x_4)$$
$$x_4 = \frac{1}{4}(1 - x_2 - x_3$$

Başlangıç değerleri

k	X_1	X_2	X_3	x_4
0	0	0	0	0
1	0.2500	0.5000	0	0.2500
2	0.1250	0.3750	-0.1250	0.1250
3	0.1875 **(0.1667)**	0.4375 **(0.4167)**	-0.0625 **(-0.0833)**	0.1875 **(0.1667)**
4	0.1563 **(0.1667)**	0.4063 **(0.4167)**	-0.0938 **(-0.0834)**	0.1563 **(0.1667)**

$Max|x_i^k - x_i^{k-1}| = |0.1250 - 0.2500| = 0.1250 > \varepsilon = 0.0001$ olduğundan **iterasyona devam!**

Aitken

$$0.1875 - \frac{(0.1875 - 0.1250)^2}{0.1875 - 2 \cdot 0.1250 + 0.2500} = 0.1667$$

$|-0.0834 - (-0.0833)| = 0.0001 = \varepsilon = 0.0001$ **iterasyonu durdur!**

İterasyon adımları

4. iterasyon sonunda bulunan çözüm

Parantez içinde koyu yazılmış değerler AITKEN formülü ile iyileştirilmiş değerlerdir. Görüldüğü gibi yakınsama hızlanmış, 13 iterasyon yerine sadece 4 iterasyon yeterli olmuştur.

AITKEN hızlandırıcısı, formülün yapısı gereği, en erken 3. adım sonunda uygulanabilir. Ancak, ilk adımlarda değerler çok kaba olduğundan, büyük denklem sistemlerinde 5.-10. adımdan sonra uygulanması daha uygun olur. İterasyonun son adımlarında da yarar sağlamaz, çünkü sadece son hanelerde çok küçük değişiklikler olmaktadır. $Max|x_i^k - x_i^{k-1}| \leq 10 \cdot \varepsilon$ olduğunda AITKEN hızlandırıcısının devre dışı bırakılması uygun olur.

AITKEN hızlandırıcısının tek sakıncalı yönü, ardaşık üç iterasyon vektörünün bellekte tutulması zorunluluğudur.

[1] A. C. **AITKEN** (1895 - 1967), Yeni Zelandalı matematikçi, ünlü formülü 1926 da yayınladı.

CG (Conjugate Gradient) metodu[1]

$\underline{A}$ **simetrik ve pozitif tanımlı**, yani $\underline{x}^T \underline{Ax} > 0$ olmak üzere

$$\underline{A}\,\underline{x}=\underline{b} \qquad (7.4)$$

denklem sisteminin

$$f(\underline{x})=\frac{1}{2}\underline{x}^T \underline{Ax}-\underline{x}^T \underline{b} \qquad (7.5)$$

fonksiyonunu minimum yapacak $\underline{x}$ çözümü aranır. $f(\underline{x})$ in minimum olma koşulu

$\frac{\partial f(\underline{x})}{\partial \underline{x}}=0$ dır: $\frac{\partial f(\underline{x})}{\partial \underline{x}}=\frac{1}{2}2\underline{Ax}-\underline{b}=0$ dan $\underline{A}\,\underline{x}=\underline{b}$ denklem sistemi olur.

Çözüme $\underline{x}_0$ başlangıç vektörü tahmin edilerek başlanır. Genellikle $\underline{x}_0 = \underline{0}$ alınır. Toplam k iterasyon adımında öyle $\underline{x}_1$, $\underline{x}_2$, $\underline{x}_3$, … , $\underline{x}_k$ vektörleri belirlenir ki son bulunan $\underline{x}_k$ vektörü 7.5 ifadesini minimum yapar. Maksimum iterasyon sayısı teorik olarak bilinmeyen sayısı kadardır: k≤n. Bu nedenle CG metodunu direkt metot olarak gören yazarlar da vardır. Ancak, yuvarlama hataları nedeniyle k>n olabilir.

CG Metodunun iterasyon adımları:

Çözülecek denklem sistemi: $\underline{A}\,\underline{x}=\underline{b}$

Başlangıç vektörü seçilir: $\underline{x}_0=\underline{0}$

Kabul edilebilir hata sınırı ε seçilir.

Hazırlık: $\underline{r}_0=\underline{b}$, $\underline{s}_1=\underline{b}$ alınır (yardımcı başlangıç vektörleri).

1.adımı:	2.adımı:	3.adımı:	……	k.adımı:
$\alpha_1=\frac{\underline{s}_1^T \underline{r}_0}{\underline{s}_1^T \underline{As}_1}$	$\alpha_2=\frac{\underline{s}_2^T \underline{r}_1}{\underline{s}_2^T \underline{As}_2}$	$\alpha_3=\frac{\underline{s}_3^T \underline{r}_2}{\underline{s}_3^T \underline{As}_3}$		$\alpha_k=\frac{\underline{s}_k^T \underline{r}_{k-1}}{\underline{s}_k^T \underline{As}_k}$
$\underline{x}_1=\underline{x}_0+\alpha_1\underline{s}_1$	$\underline{x}_2=\underline{x}_1+\alpha_2\underline{s}_2$	$\underline{x}_3=\underline{x}_2+\alpha_3\underline{s}_3$		$\underline{x}_k=\underline{x}_{k-1}+\alpha_k\underline{s}_k$
$\underline{r}_1=\underline{r}_0-\alpha_1\underline{As}_1$	$\underline{r}_2=\underline{r}_1-\alpha_2\underline{As}_2$	$\underline{r}_3=\underline{r}_2-\alpha_3\underline{As}_3$		$\underline{r}_k=\underline{r}_{k-1}-\alpha_k\underline{As}_k$
$\sqrt{\underline{r}_1^T \underline{r}_2}\le\varepsilon$ ise durdur!	$\sqrt{\underline{r}_2^T \underline{r}_2}\le\varepsilon$ ise durdur!	$\sqrt{\underline{r}_3^T \underline{r}_3}\le\varepsilon$ ise durdur!		$\sqrt{\underline{r}_k^T \underline{r}_k}\le\varepsilon$ ise durdur!
$\beta_1=-\frac{\underline{r}_1^T \underline{As}_1}{\underline{s}_1^T \underline{As}_1}$	$\beta_2=-\frac{\underline{r}_2^T \underline{As}_2}{\underline{s}_2^T \underline{As}_2}$	$\beta_3=-\frac{\underline{r}_3^T \underline{As}_3}{\underline{s}_3^T \underline{As}_3}$		$\beta_k=-\frac{\underline{r}_k^T \underline{As}_k}{\underline{s}_k^T \underline{As}_k}$
$\underline{s}_2=\underline{r}_1+\beta_1\underline{s}_1$	$\underline{s}_3=\underline{r}_2+\beta_2\underline{s}_2$	$\underline{s}_4=\underline{r}_3+\beta_3\underline{s}_3$		$\underline{s}_{k+1}=\underline{r}_k+\beta_k\underline{s}_k$

Not:

- Her adımın formüllerinde 4 kez görülen $\underline{As}_k$ çarpımı her adımda bir kez yapılır.
- Her adımın formüllerinde 2 kez görülen $\underline{s}_k^T \underline{As}_k$ değeri her adımda bir kez hesaplanır.
- $\underline{r}_k$ (kalan) hata vektörüdür. Teorik olara $\underline{r}_k=\underline{0}$ olduğunda iterasyon sona erer. Ancak, sayısal hesaplarda tam sıfır genelde yakalanamaz, hataların karelerinin karekökü kabul edilebilir hata sınırı altında kalınca iterasyonu durdurmak daha iyi bir yoldur: $\sqrt{\underline{r}_k^T \underline{r}_k}\le\varepsilon$. Diğer bir seçenek: $\underline{r}_k$ nin elemanlarından mutlak değerce en büyüğü<ε olunca iterasyon durdurulur.
- $i\neq k$ için $\underline{r}_i$ ve $\underline{r}_k$ ortogonaldır, $\underline{r}_i^T \underline{r}_k = 0$ olmalıdır. Bu koşul yardımıyla iterasyonun, yuvarlama hataları nedeniyle, tehlikeye girip girmediği kontrol edilebilir.

[1] Magnus R. **HESTENES**(1906-1991), Amerikalı ve Eduard L. **STIEFEL**(1909-1978), İsviçreli: 1952 yılında yayınladılar.

Örnek:

$$\underline{\underline{A}}\underline{x}=\underline{b},\quad \begin{bmatrix}4&1&1&0\\1&4&0&1\\1&0&4&1\\0&1&1&4\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}1\\2\\0\\1\end{bmatrix},\ \underline{x}=?$$

Başlangıç vektörü seçilir: $\underline{x}_0=[0\ 0\ 0\ 0]^T$ alalım.

Kabul edilebilir hata sınırı seçilir: $\varepsilon=0.0001$ olsun.

Hazırlık, yardımcı vektörler: $\underline{r}_0=[1\ 2\ 0\ 1]^T$, $\underline{s}_1=[1\ 2\ 0\ 1]^T$

Hesaplarda ondalık sayıdan sonra 4 hane yürütelim.

1. adım:

$$\underline{s}_1^T\underline{r}_0=6,\quad \underline{\underline{A}}\underline{s}_1=\begin{bmatrix}6\\10\\2\\6\end{bmatrix},\quad \underline{s}_1^T\underline{\underline{A}}\underline{s}_1=32$$

$$\alpha_1=\frac{\underline{s}_1^T\underline{r}_0}{\underline{s}_1^T\underline{\underline{A}}\underline{s}_1}=\frac{6}{32}=0.1875$$

$$\underline{x}_1=\underline{x}_0+\alpha_1\underline{s}_1=\begin{bmatrix}0.1875\\0.3750\\0\\0.1875\end{bmatrix},\quad \underline{r}_1=\underline{r}_0-\alpha_1\underline{\underline{A}}\underline{s}_1=\begin{bmatrix}-0.1250\\0.1250\\-0.3750\\-0.1250\end{bmatrix}$$

$\underline{r}_1^T\underline{r}_1=0.1875$, $\sqrt{0.1875}=0.4330>\varepsilon=0.0001$ olduğundan iterasyona devam edilecek!

$$\underline{r}_1^T\underline{\underline{A}}\underline{s}_1=-1,\quad \beta_1=-\frac{\underline{r}_1^T\underline{\underline{A}}\underline{s}_1}{\underline{s}_1^T\underline{\underline{A}}\underline{s}_1}=-\frac{-1}{32}=0.0313,\quad \underline{s}_2=\underline{r}_1+\beta_1\underline{s}_1=\begin{bmatrix}-0.0937\\0.1876\\-0.3750\\-0.0937\end{bmatrix}$$

2. adım:

$$\underline{s}_2^T\underline{r}_1=0.1875,\ \underline{\underline{A}}\underline{s}_2=\begin{bmatrix}-0.5622\\0.5630\\-1.6874\\-0.5622\end{bmatrix},\quad \underline{s}_2^T\underline{\underline{A}}\underline{s}_2=0.8438$$

$$\alpha_2=\frac{\underline{s}_2^T\underline{r}_1}{\underline{s}_2^T\underline{\underline{A}}\underline{s}_2}=\frac{0.1875}{0.8438}=0.2222,\quad \underline{x}_2=\underline{x}_1+\alpha_2\underline{s}_2=\begin{bmatrix}0.1667\\0.4167\\-0.0833\\0.1667\end{bmatrix},\quad \underline{r}_2=\begin{bmatrix}-0.0001\\-0.0001\\-0.0001\\-0.0001\end{bmatrix}$$

$\underline{r}_2^T\underline{r}_2\approx 0.$, $\sqrt{0}=0<\varepsilon=0.0001$ olduğundan iterasyon durduruldu!

Çözüm: $\underline{x}=\begin{bmatrix}0.1667\\0.4167\\-0.0833\\0.1667\end{bmatrix}$

Sadece 2 adımda çözüm bulunmuştur. Aynı örnek için JACOBI 13, GAUSS-SEIDEL 7, AITKEN hızlandırıcılı JACOBI 4 iterasyon adımı gerektirmişti. CG metodunun diğer iterasyon metotlarına göre üstünlüğü bu küçük örnek ile dahi görülmektedir. Ancak, CG metodunun sadece simetrik ve pozitif tanımlı $\underline{A}$ özel durumu için geçerli olduğunu hatırlatalım. Genel olarak, GAUSS-SEIDEL metodu JACOBI metodundan, CG metodu da her ikisinden çok daha çabuk yakınsar. JACOBI metodu uygulamada hemen hiç kullanılmaz.

Not:

- **CG** (Conjugate Gradient) iterasyon metodu 1952 yılında M. R. **HESTENES** ve E. **STIEFEL** tarafından yayınlandı. Bak: **Hestenes**, M. L., **Stiefel**, E., 1952. Methods of Conjugate Gradients for Solving Linear Systems. J. of Research of the National Bureau of Standards 49 (6). http://nvl.nist.gov/pub/nistpubs/jres/049/6/V49.N06.A08.pdf.
- **CG** (Conjugate Gradient) iterasyon metodu sadece $\underline{A}$ katsayılar matrisi simetrik ve pozifif tanımlı sistemlerin çözümünde kullanılabilir. $\underline{A}\ \underline{x}=\underline{b}$ denklem sisteminde $\underline{A}$ simetrik değilse, $\underline{A}^T\underline{A}\ \underline{x}=\underline{A}^T\underline{b}$ dönüşümü yapılarak CG metodu $\underline{A}^T\underline{A}\ \underline{x}=\underline{A}^T\underline{b}$ sistemine uygulanabilir. Çünkü $\underline{A}^T\underline{A}\ \underline{x}=\underline{A}^T\underline{b}$ sisteminde $\underline{A}^T\underline{A}$ katsayılar matrisi simetrik ve pozitif tanımlıdır.
- **CG** (Conjugate Gradient) iterasyon metodu 20. yüzyılın en iyi algoritmalarından biri seçilmiştir. Bak: http://www.cs.duke.edu/courses/fall06/cps258/references/topten.pdf http://www.bilisimdergi.com/20-Yuzyilin-En-iyi-10-Algoritmasi-3-10.html

Magnus R. **HESTENES**(1906-1991)

Eduard L. **STIEFEL**(1909-1978)

İterasyon yöntemlerinin avantajları

- A x=b denklem sistemi direkt metodlar ile çözülemeyecek kadar büyük ise ve A seyrek bir matris ise iterasyon yöntemleri ile çözülebilir. İterasyon yöntemleri ile çözülebilecek denklem sisteminin büyüklüğü günümüzde 1-2 milyara dayanmıştır.
- A seyrek matris ise çok az bellek gerektirir.
- A seyrek matris ise çok az dört işlem, çok daha az hesap süresi gerektirir.
- Optimizasyon gibi tekrarlanan problemlerin çözümüne uygundur. Çünkü bir önceki çözüm bir sonraki çözümün başlangıç vektörü olarak alınabilir, çözüm çok çabuk yakınsar.
- İterasyon metotlarında A x=b denklem sisteminin A katsayılar matrisi değişikliğe uğramaz. Buna karşın; direkt metotlarda A nın elemanları değişir, sıfır elemanlar sıfırdan farklı olurlar. Seyrek matrisler giderek dolu matris olurlar.
- İterasyon metotlarında yuvarlama hataları direkt metotlara nazaran çok daha az sorun yaratır.
- Birkaç iterasyon adımı sonunda oldukça yaklaşık bir ara çözüm oluşur. Kabaca bir çözümün yeterli olduğu problem türlerinde birkaç iterasyon adımı yeterli olur. Direkt metotlarda ara çözüm bulmak mümkün değildir.
- Tümü ana bellekte depolanamayacak kadar büyük denklem sistemlerinde A nın dış bellekte(hard disk veya benzeri) depolanması ve çözümün taksit-taksit yapılması çok daha basittir.

İterasyon yöntemlerinin dezavantajları

- Katsayılar matrisi tam dolu ise, yani seyrek değil ise, iterasyon metotları kullanmak uygun olamaz (çok fazla bellek, çok fazla işlem yükü, biriken yuvarlama hataları).
- Karşı taraf vektörünün iterasyon başlamadan önce bilinmesi zorunludur.
- Birden çok her karşı taraf vektörü için itarasyonun tekrarlanması gerekir.
- İterasyonun kaç adımda sona ereceği, dolayısıyla hesap süresi önceden kestirilemez.
- Bazı özel durumlar hariç, her denklem sisteminde iterasyonun yakınsayacağı garantisi yoktur. Çözüm yakınsayabilirde, ıraksayabilirde. Aynı denklem sisteminin için iterasyon metotlarından biri yakınsarken bir diğeri ıraksayabilir. Ancak; katsayılar matrisi ***kesin diyagonal ağırlıklı*** veya ***simetrik pozitif tanımlı*** ise JACOBI ve GAUSS-SEIDEL metodu mutlaka yakınsar. CG metodu ise sadece ***simetrik ve pozitif tanımlı*** matrisler için daima yakınsar.
- Çözümün yakınsamaması durumunda iterasyon, teorik olarak, sonsuza kadar devam eder. Bu nedenle iterasyon sayısı maksimum iterasyon sayısı ile sınırlandırılmak zorundadır. Maksimum iterasyon sayısının ne olması gerektiğinin net bir cevabı yoktur. Çok küçük tutulursa doğru sonuca ulaşılamaz, çok büyük tutulursa gereksiz yere hesap süresi uzar.
- İterasyonun yakınsamaması durumunda sorunun kullanılan metottan mı yoksa denklem sisteminin yapısından mı kaynaklandığını belirlemek zordur. Örneğin denklem sistemi hatalı kurulmuş ise, katsayılar matrisinin determinantı sıfır ise çözüm yakınsamaz. İterasyon metotlarında determinant hesabı mümkün olmadığından hata kaynağı belirlenemez.

Hatırlatma:

Kesin diyagonal ağırlıklı matris *tanımı için bak: bölüm 1 , sayfa 15.*
Simetrik pozitif tanımlı matris *tanımı için bak : : bölüm 1, sayfa 15 ve bölüm 6 sayfa 78.*

Hangi çözüm yöntemi daha iyi?

n bilinmeyenli A x=b doğrusal denklem sisteminin çözümü için hangi metot daha uygundur? Cevabı oldukça zor bir sorudur bu! Çünkü çok sayıda etken vardır: Denklem sisteminin büyüklüğü, A nın yapısı, çözüm süresi, çözümden beklenen hassasiyet, kullanılacak bilgisayarın özellikleri, paket program var mı veya programlanacak mı? Bir fikir vermek üzere aşağıdaki yorumları yapabiliriz:

- Denklem sistemi küçük ise direkt metotlardan (GAUSS, DOOLITTLE, CROUT, CHOLESKY) birini kullanmak uygundur. Ancak küçük nedir? Sorusu çıkar karşımıza. A tam dolu ise, günümüz bilgisayarları için $n \leq 10000$ bilinmeyen küçük sayılabilir. A bant matris ise $n \leq 20000$, bant ve simetrik ise $n \leq 40000$ bilinmeyen küçük sayılabilir.
- Denklem sistemi büyük ve A seyrek bir matris ise; iterasyon yöntemi kullanılmak zorundadır. Ancak, sayısız denilecek kadar, az ya da çok birbirinden farklı, iterasyon metodu vardır. En uygun iterasyon metodunun seçimi zordur.
- Yuvarlama hatalarının sorun yaratacağı sistemlerde iterasyon yöntemleri uygun olur. A nın simetrik ve pozitif tanımlı olduğu biliniyorsa CG metodu en uygun iterasyonyon yöntemidir.
- İterasyon yöntemlerinden, elden geldiğince, kaçınmak gerekir. Uygun bir iterasyon yöntemi aramak yerine direkt metotlar ile çözümün yapılabileceği uygun bir bilgisayarın aranması belki de daha doğru bir yoldur.