T.C.
MUĜLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

SPARSE LİNEER SİSTEMLERİN İTERASYON METODLARI İLE ÇÖZÜMÜ VE METODLARIN KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

SİBEL DEMİRYAKAN SARI

HAZİRAN - 2006
MUĜLA

T.C.
MUĜLA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATEMATİK ANABİLİM DALI

SPARSE LİNEER SİSTEMLERİN İTERASYON METODLARI İLE ÇÖZÜMÜ VE METODLARIN KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

SİBEL DEMİRYAKAN SARI

MUĜLA 2006

ONAY SAYFASI

Yrd. Doç. Dr. Niyazi ŞAHİN danışmanlığında Sibel DEMİRYAKAN SARI tarafından hazırlanan bu çalışma 27/06/2006 tarihinde aşağıdaki jüri tarafından Matematik Ana Bilim Dalı'nda yüksek lisans tezi olarak oybirliği/oyçokluğu ile kabul edilmiştir.

Başkan : Yrd. Doç. Dr. Metin KANTAR İMZA :

Danışman : Yrd. Doç. Dr. Niyazi ŞAHİN İMZA :

Üye : Yrd. Doç. Dr. Murat ATMACA İMZA :

ÖNSÖZ

Bu çalışmanın hazırlanmasında yardımlarını ve desteĝini esirgemeyen deĝerli danışmanım Yrd. Doç. Dr. Niyazi ŞAHİN 'e , kalplerinin her an benimle olduĝunu bildiĝim sevgili aileme ve her zaman her konuda yanımda olan eşime çok teşekkür ederim.

Ayrıca bu tezin Latex'te yazılmasında bana çok yardımcı olan deĝerli hocam Yrd. Doç. Dr. Murat Atmaca'ya teşekkürü borç bilirim.

Sibel Demiryakan SARI

MUĜLA 2006

İÇİNDEKİLER

Sayfa No:

"SPARSE LİNEER SİSTEMLERİN İTERASYON METODLARI İLE ÇÖZÜMÜ VE METODLARIN KARŞILAŞTIRILMASI"

(Yüksek lisans Tezi)

Sibel Demiryakan SARI

MUĜLA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

2006

ÖZET

Bu çalışmada, sparse lineer denklem sistemlerinin bazı iterasyon metodları ile çözümleri ve bu metodların bir karşılaştırılması yapılmıştır. Ayrıca, metodlar arasındaki iterasyon sayısı ve hata deĝişimi de incelenmiştir.

İlaveten, bu sistemlerin çözümü için gerekli Matlab programları da eklenmiştir.

Türkçe Anahtar Kelimeler :Sparse, İterasyon Yöntemleri, Lineer Denklem Sistemleri, Matlab

Sayfa Adedi :72

Tez Yöneticisi :Yrd. Doç. Dr. Niyazi ŞAHİN

"THE SOLUTIONS OF SPARSE LINEAR EQUATIONS USING ITERATIVE METHODS AND COMPARISONS OF METHODS"

(M. Sc. Thesis)

MUĜLA UNIVERSITY

INSTITUTE of SCIENCE and TECHNOLOGY

2006

ABSTRACT

In this study, the solution of sparse systems of linear equations using some iterative solution methods are studied and these methods have been compared with each other in terms of iteration numbers and errors.

In addition, Matlab codes to solve the systems are presented.

Key Words :Sparse, Linear Equations, Iterative Methods, Matlab

Page Number :72

Adviser :Asst. Prof. Dr. Niyazi ŞAHİN

SEMBOLLER ve KISALTMALAR DİZİNİ

ŞEKİLLER DİZİNİ

TABLOLAR/ÇİZELGELER DİZİNİ

1. GİRİŞ

$Ax = b$ şeklindeki denklemlerinin çözümünde Gauss-Eliminasyon gibi yöntemler kullanılması çok zaman gerektirir. Tam çözüm ü sonlu sayıda adımla hesaplayan Gauss-Eliminasyon gibi metodlar "direkt metod" olarak adlandırılırlar. Direkt metoda karşın iterasyon metodları genellikle sonlu sayıda adımdan sonra tam çöz ümü vermez ama her adımda hatayı küçült ür. İteratif metodlar küçük boyutlu lineer sistemleri çözmede nadiren kullanılır. Çünkü yeterli kesinliĝe ulaşmak için zaman gereklidir, bu da Gauss-Eliminasyon gibi direkt metodları kullanmayı gerektirir. "0" girişi yüksek miktarda olan büyü k sistemlerde bu teknikler hem bilgisayar depolamasında, hem de hesaplamada etkilidirler. nxn boyutlu Ax=b lineer sistemini çözmek için bir iteratif teknik, x'i çö zmek için ilk başlaki tahmini $x^{(0)}$ deĝeriyle başlar ve x'e yaklaşan $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ vektör dizisini meydana getirir. İteratif teknikler $Ax = b$ sistemini $x = Tx + c$ formundaki T matrisine ve c vektörüne baĝlı eşitlik sistemine dönü ştürür. İlk başlaki $x^{(0)}$ seçildikten sonra yaklaşık çözümün vektör dizisi $x^{(k)} = Tx^{(k-1)} + c$ 'nin hesaplanmasıyla oluşturulur. Sonuçlaki hata, metodların ve lineer sistemlerin ö zelliklerinde oldu ĝu gibi kaç iterasyon yapıldığına baĝlıdır. Amacımız metodları geliştirmek ve hatayı büyük miktarda azaltıp, mümkü n olduĝunca az işlem yapmaktır.

Bu alandaki tüm çalışmalar, iteratif metodlar daha iyi düzenlenmesi için lineer sistemlerdeki artışları veren, kolayca görülmeyen ama önemli olan matematikle ve fizikle ilgili problemlerden faydalanmayı içerir. Bu önemli problemler sonlu farklılıklar ve fiziksel sistemdeki sonlu eleman modelleridir ve genellikle diferansiyel denklemleri içerir. Bir kare matris şeklindeki "Poisson Denklemleri" ve ona yakın denklemler, "Laplace Denklemi" birçok uygulamalardan doĝar ve elektromagnetizmayı, akışkanlar mekaniĝini, sıcaklık akışını, difüzyonu ve quantum mekani ĝini içerir.

2. KAYNAK ÖZETLERİ

2.1. Jacobi Metodu

Sparse lineer sistemlerin çözümünde kullanılan klasik lineer iterasyon metodlarından ilki Jacobi metodudur. Jacobi metodunda, verilen A matrisi, $A = D - \widetilde{L} - \widetilde{U} = D(I - L - U)$ şeklinde ayrıştırılır.Böylece,

$R_j = D^{-1}(\widetilde{L} + \widetilde{U}) = L + U$ ve $c_j = D^{-1}b$.

Böylece Jacobi metodda $x_{m+1} = R_j x_m + c_j$ gibi bir adım yazılabilir. Bu formülün bizim Jacobi metodla ilgili ilk tanımlamamızla ilişkisini görmek i çin;

$Dx_{m+1} = (\widetilde{L} + \widetilde{U})x_m + b$ veya $a_{jj}x_{m+1,j} = -\sum_{k \neq j} a_{jk}x_{m,k} + b_j$ veya

$a_{jj}x_{m+1,j} + \sum_{k \neq j} a_{jk}x_{m,k} = b_j$.

Bu x_i için bulunan $(a_{ii} \neq 0)$ $Ax = b$ 'deki i'nci denklemi çözmekten ibarettir.

$$x_i = \sum_{j=1, j \neq i}^{n} (-\tfrac{a_{ij}x_j}{a_{ii}}) + \tfrac{b_i}{a_{ii}},\ i = 1, 2, ..., n$$

ve $x^{(k-1)}$'in elemanından her $x_i^{(k)}$'yı oluş turmak$(k \geq 1)$:

$$x_i^{(k)} = \frac{\sum_{j=1, j \neq i}^{n} (-a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}}\ , i = 1, 2, ..., n \tag{2.1}$$

Metod, A'yı diagonallerine ve diagonal bölümlerine ayırarak, $x^{(k)} = Tx^{(k-1)} + c$ formunda yazılır.

D bir diagonal matris, $-L$ alt üçgen matris ve $-U$ üstüçgen matris olmak üzere

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & \cdots & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

matrisi aşagıdaki şekilde ayrıştırılır:

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ & \cdots & & \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & 0 & \cdots & 0 \\ & \cdots & & \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & -a_{2n} \\ & \cdots & & \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Böylece $A = D - L - U$ dur.

$Ax = b$ denklemi veya $(D - L - U)x = b$, daha sonra şu hale dönüştürülür:

$Dx = (L + U)x + b$

Eğer D^{-1} varsa ve a_{ii} her i için sıfırdan farklı ise;

$x = D^{-1}(L + U)x + D^{-1}b$

Bu sonucun Jacobi iterasyon yöntemindeki matris formu şöyledir:

$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b,\ k = 1, 2, ...$

$T_j = D^{-1}(L + U)$ ve $c_j = D^{-1}b$ tanımlanırsa, Jacobi Metod şu formu alır:

$x^{(k)} = T_j x^{(k-1)} + c_j$

Pratikte;

$$x_i^{(k)} = \frac{\sum_{j=1, j\neq i}^{n} (-a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}} \quad i = 1, 2, ..., n$$

Teorikte;

$x^{(k)} = T_j x^{(k-1)} + c_j$ kullanılır.

2.2. **Gauss-Seidel Metodu:**

Gauss-Seidel Metod Jacobi Metod' dan farklıdır, farkı ise, $x_j^{(k)}$' nın k. adımdaki mevcut değerlerinin çözümün güncellenmesinde kullanılıyor olmasıdır.

$$x_i^{(k)} = \frac{\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}} \tag{2.2}$$

Gauss-Seidel Metod'u matris formunda yazarak 2.2 eşitliginin her iki tarafını a_{ii} ile çarpalım. Bütün k. iteratif terimleri bir araya toplayalım:

$a_{i1}x_1^{(k)} + a_{i2}x^{(k)} + ... + a_{ii}x_i^{(k)} = -a_{i,i+1}x_{i+1}^{(k-1)} - ... - a_{in}x_n^{(k-1)} + b_i\ ,\ \forall i = 1, 2, ..., n$

Bütün n denklemlerini yazmak şunu verir:

$$\begin{aligned}
a_{11}x_1^{(k)} &= -a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots - a_{1n}x_n^{(k-1)} + b_1 \\
a_{21}x_1^{(k)} + a_{22}x_2^{(k)} &= -a_{23}x_3^{(k-1)} - \cdots - a_{2n}x_n^{(k-1)} + b_2 \\
&\cdots \\
a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots + a_{nn}x_n^{(k)} &= b_n
\end{aligned}$$

Yukarıda tanımlanan D, L, U ile Gauss-Seidel metodun aşağıdaki yazılışını elde ederiz.

$(D - L)x^{(k)} = Ux^{(k-1)} + b$ veya

$x^{(k)} = (D - L)^{-1}Ux^{(k-1)} + (D - L)^{-1}b\ k = 1, 2, ...$

$T_g = (D-L)^{-1}U$ ve $c_g = (D-L)^{-1}b$ alarak, Gauss-Seidel teknigi şu biçime sahip olur:

$x^{(k)} = T_g x^{(k-1)} + c_g$

Alt üçgensel matris $D-L$'nin tekil olmayan bir matris olması için her $i = 1, 2, ..., n$ için $a_{ii} \neq 0$ olması gerekli ve yeterlidir. Bu hemen hemen her zaman doğrudur, ama Jacobi Metod'un çözüme yaklaştıgı, Gauss-Seidel Metod'un yaklaşmadığı lineer sistemler de vardır.

$k = 1, 2, ...$ için, $x^{(0)}$ keyfi oldugunda genel iterasyon tekniklerinde çalışırken aşa ğıdaki formülü göz önünde tutacagız:

$x^{(k)} = Tx^{(k-1)} + c$ her

Tanım 2..1 *R'nin tüm λ özdeğerlerinin maksimum olarak alındığı durumda R'nin spektral yarıçapı;*

$\rho(R) = \max|\lambda|$ *'dır.*

Yardımcı Teorem 2..1 *Eğer spektral yarıçap $\rho(T), \rho(T) < 1$ koşulunu yerine getiriyorsa, $(I-T)^{-1}$ vardır ve*

$$(I-T)^{-1} = I + T + T^2 + ... = \sum_{j=0}^{\infty} T^j$$

İspat : $(I-T)x = (1-\lambda)x$ olduğunda $Tx = \lambda$ kesinlikle doğrudur. $(1-\lambda)$, $(I-T)$' nin bir özdeğeri olduğu zaman λ'nın, kesinlikle T 'nin bir özdeğeri olduğunu elde ederiz. Ama $|\lambda| \leq \rho(T) < 1$ idi. Bu yüzden $\lambda = 1$, T'nin bir özdeğeri değildir ve 0(sıfır) $(I-T)$'nin bir özdeğeri olamaz. Bu sebepten ötürü, $(I-T)^{-1}$ vardır.

$S_m = I + T + T^2 + ... + T^m$ alalım. Sonra $(I-T)S_m = (1 + T + T^2 + ... + T^m) - (T + T^2 + ... + T^{m+1}) = I - T^{m+1}$ T yaklaşan olduğu için;

$$\lim_{m\to\infty}(I-T)S_m = \lim_{m\to\infty}(I - T^{m+1}) = I$$

Böylece,

$(I-T)^{-1} = \lim_{m\to\infty} S_m = I + T + T^2 + ... = \sum_{j=0}^{\infty} T^j$ olur.

Teorem 2..1 *$x^{(0)} \in R^n$ için $x^{(k)} = Tx^{(k-1)} + c$ ($k \geq 1$ için) diye tanımlanan $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ dizisi ancak ve ancak $\rho(T) < 1$ iken $x = Tx + c$'nin çözümüne tam yaklaşır.*

İspat: İlk başta $\rho(T) < 1$ olduğunu farzedelim. O zaman

$$x^{(k)} \quad = \quad Tx^{(k-1)} + c$$

$$\begin{aligned} &= T(Tx^{(k-2)} + c) + c \\ &= T^2x^{(k-2)} + (T+I)c \\ &\dots \\ &= T^kx^{(0)} + (T^{k-1} + ... + T + I)c \end{aligned}$$

$\rho(T) < 1$ olduĝunda T matrisi yaklaşandır ve $\lim_{k\to\infty} T^kx^{(0)} = 0$. Yardımcı teoremden,

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} T^kx^{(0)} + (\sum_{j=0}^{\infty} T^j)c = 0 + (I-T)^{-1}c = (I-T)^{-1}c$$

Bundan dolayı $\left\{x^{(k)}\right\}$ dizisi $x \equiv (I-T)^{-1}$ vektörüne ve $x = Tx+c$'ye yaklaşır. Aksini ispatlamak için, her $z \in R^n$ iç in göstererecegiz ve $\lim_{k\to\infty} T^kz = 0$'ı elde edecegiz. Bu $\rho(T) < 1$'e eşittir. Keyfi bir z vektörü alalım ve x, $x = Tx+c$'nin tam çözümü olsun.

$x^{(0)} = x - z$ olsun ve $k \geq 1$ için

$x^{(k)} = Tx^{(k-1)} + c$ 'dir.

Daha sonra $\left\{x^{(k)}\right\}$ x'e yaklaşır. Bununla beraber

$$x - x^{(k)} = (Tx + c) - (Tx^{(k-1)} + c) = T(x - x^{(k-1)})$$
$$x - x^{(k)} = T(x - x^{(k-1)}) = T^2(x - x^{(k-2)}) = ... = T^k(x - x^{(0)}) = T^kz$$

Bu nedenle,

$$\lim_{k\to\infty} T^kz = \lim_{k\to\infty} T^k(x - x^{(0)}) = \lim_{k\to\infty} (x - x^{(k)}) = 0\text{' dır.}$$

$z \in R^n$ keyfi olduĝundan, bu T'nin yakınsak(yaklaşan) olduĝunu ve $\rho(T) < 1$ olduĝunu belirtir.

Sonuç: Eĝer her yapıdaki matris normu için $\|T\| < 1$ ise ve c verilen vektör ise, daha sonra $x^{(k)} = T^{(k-1)}$ ile tanımlanan $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ dizisi her $x^{(0)} \in R^n$ için $x \in R^n$ vektörüne yakınsar. İzleyen sınır hata hükmü,

$i)$ $\left\|x - x^{(0)}\right\| \leq \|T\|^k \left\|x^{(0)} - x\right\|$;

$ii)$ $\left\|x - x^{(0)}\right\| \leq \frac{\|T\|^k}{1-\|T\|} \left\|x^{(1)} - x^{(0)}\right\|$.

Jacobi ve Gauss-Seidel Metodlarının şu şekilde yazılabildiĝini görüyoruz

$x^{(k)} = T_jx^{(k-1)} + c_j$ ve $x(k) = T_gx^{(k-1)} + c_g$ matrisleri kullanarak;

$T_j = D^{-1}(L+U)$ ve $T_g = (D-L)^{-1}U$

Eger $\rho(T_j)$ ve $\rho(T_g)$ 1'den küçükse, uygun $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ dizisi $Ax = b$'nin tam çözümüne yakınsayacaktır. Örneğin Jacobi tasarısı $x^{(k)} = D^{-1}(L+U)x^{(k-1)} + D^{-1}b$ 'e sahip olacaktır ve eğ er $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ x'e yakınsarsa

$x = D^{-1}(L+U)x + D^{-1}b$

Bu şunu belirtir:

$Dx = (L+U)x + b$ ve $(D-L-U)x = b$

Çünkü, $(D-L-U) = A$'dır, x çözümü $Ax = b$'yi sağlar. Şimdi Jacobi ve Gauss-Seidel Metod'un yaklaşması için yeterli şartları kolayca verebiliriz.

Teorem 2..2 *Eğer A kesin diagonal dominant ise, herhangi $x^{(0)}$ seçimi için , Jacobi ve Gauss-Seidel Metodları $Ax = b$'nin tam çözümüne yakınsayan $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ dizisini verir.*

Teorem 2..3 *(Stein-Rosenberg): Eğer, $a_{ij} \leq 0$ $(i \neq j)$ için ve $a_{ij} > 0$ $(i = 1, 2, ..., n)$ ise;*

1. $0 \leq \rho(T_g) < \rho(T_j) < 1$
2. $1 < \rho(T_j) < \rho(T_g)$
3. $\rho(T_j) = \rho(T_g) = 0$
4. $\rho(T_j) = \rho(T_g) = 1$

Tanım 2..2 *$\tilde{x} \in R^n$'in $Ax = b$ diye tanımlanan lineer sistemin yaklaşık çözümü olduğunu farzedelim. $\tilde{x}$ için fark vektörü $r = b - A\,\tilde{x}$ 'dır.*

Jacobi ve Gauss-Seidel gibi metodların işlemlerinde, fark vektörü çözüm vektörünün yakınsak elemanlarının her bir hesaplamasıyla birleşen bir vektördür. Amaç 0(sıfır)'a hızlı yaklaşan fark vektörüne sebep olacak yakınsak bir dizi meydana getirmektir.

$r_i^{(k)} = (r_{1i}^{(k)}, r_{2i}^{(k)}, ..., r_{ni}^{(k)})^t$ aldığımızı farzedelim.

Gauss-Seidel Metod'un yaklşık çözüm vektörü şöyle tanımlanır:

$x_i^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_{i-1}^{(k)}, x_i^{(k-1)}, ..., x_n^{(k-1)})^t$.

$r_i^{(k)}$, nın $m.$ elemanı;

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj}x_j^{(k)} - \sum a_{mj}x_j^{(k-1)} \tag{2.3}$$

veya

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj}x_j^{(k)} - \sum_{j=i+1}^{n} a_{mj}x_j^{(k-1)} - a_{mi}x_i^{(k-1)}$$

Her $m = 1, 2, ..., n$ için ; $r_i^{(k)}$'nın $i.$ elemanı
$r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k-1)}$
Böylece;

$$a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} \tag{2.4}$$

Gauss-Seidel Metod'da $x_i^{(k)}$'nın şu şekilde seçildiĝini hatırlayalım:

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} \right] \tag{2.5}$$

Bundan dolayı 2.4 eşitliĝi yeniden şöyle yazılabilir :
$a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = a_{ii}x_i^{(k)}$

Sonuç olarak, Gauss-Seidel Metod koşulları saĝlayan $x_i^{(k)}$ seçerek şu şekilde karakterize edilebilir.

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}} \tag{2.6}$$

Fark vektörleri ile Gauss-Seidel Metodu arasındaki başka bir ilgiyi çıkarabiliriz. $x_{i+1}^{(k)} = (x_1^{(k)}, ..., x_i^{(k)}, x_{i-1}^{(k-1)}, ..., x_n^{(k-1)})^t$ vektörüyle birleşen fark vektörü $r_{i+1}^{(k)}$ 'yı düşünelim. $r_{i+1}^{(k)}$ 'nın $i.$ elemanı:

$$\begin{aligned} r_{i,i+1}^{(k)} &= b_i - \sum_{j=1}^{i} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} \\ &= b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k)} \end{aligned}$$

2.5 eşitligi $r_{i,i+1}^{(k)} = 0$ olduĝunu belirtir.Gauss-Seidel metodu $r_{i+1}^{(k)}$'nın $i.$ elemanı 0(sıfır) olduĝu zaman $x_{i+1}^{(k)}$ seçerek karakterize edilebilir. Fark vektörünün bir koordinatı 0(sıfır) olduĝundan, bununla beraber, $r_{i+1}^{(k)}$ vektörünün normunu azaltmak en etkili yol deĝildir. Eĝer Gauss-Seidel işlemini düzenlersek şu verilir :

$$x_i^{(k)} = x_i^{(k-1)} + w\frac{r_{ii}^{(k)}}{a_{ii}} \tag{2.7}$$

w' nin pozitif seçimleri için fark vektörünü n normunu küçültebiliriz ve hızlı bir yaklaş ma elde ederiz. $x_i^{(k)} = x_i^{(k-1)} + w\frac{r_{ii}^{(k)}}{a_{ii}}$ denklemini içeren metodlar "relaxation metodları" diye adlandırılır.

Teorem 2..4 *Eĝer A kesin diagonal dominant satır matris ise Jacobi ve Gauss-Seidel metodlarının her ikisi de yakınsar. Gerçekte,*

$\|R_{GS}\|_\infty \leq \|R_j\|_\infty < 1$

İspat: $A = D - \tilde{L} - \tilde{U} = D(I - L - U)$ denklemini kullanarak $R_j = L + U$ ve $R_{GS} = (I - L)^{-1}U$ yazarız. $e = [1, ..., 1]^T$ olmak üzere,

$\|R_{GS}\|_\infty = \||R_{GS}|\, e\|_\infty \leq \||R_j|\, e\|_\infty = \|R_j\|_\infty$

$|(I - L)^{-1}U|\, .e = |R_{GS}|\, .e = (|L| + |U|).e$

$|(I - L)^{-1}U|\, .e \leq |(I - L)^{-1}|\, .\, |U|\, .e$ Üçgen eşitsizliği ile;

$= \left|\sum_{i=0}^{n-1} L^i\right| .\, |U|\, .e\ L^n = 0$

$\leq \sum_{i=0}^{n-1} |L|^i\, .\, |U|\, .e$ Üçgen eşitsizliği ile;

$= (I - |L|)^{-1}.\, |U|\, .e\ |L^n| = 0$

$(I - |L|)^{-1}.\, |U|\, .e \leq (|L| + |U|).e$

Çünkü $(I - |L|)^{-1} = \sum_{i=0}^{n-1} |L|^i$ 'nin tüm girişleri negatif değil. Eğer aşağıdaki ifadeyi ispatlayabilirsek eşitsizlik doğru olacak;

$|U|\, .e \leq (I - |L|).(|L| + |U|).e = (|L| + |U| - |L|^2 - |L|\, .\, |U|).e$ veya

$0 \leq (|L| - |L|^2 - |L|\, .\, |U|).e = |L|\, .(I - |L| - |U|).e$

$|L|$'nin tüm tüm girişleri negatif.Aşağıdaki ifadeyi ispatlayabilirsek eşitsizlik doğru olacak:

$0 \leq (I - |L| - |U|).e$ veya $|R_j|\, .e = (|L| + |U|)e \leq e$

Sonuç olarak yukarıdaki eşitsizlik,$\||R_j|\, .e\|_\infty = \|R_j\|_\infty = \rho < 1$ varsayımından dolayı doğru olur.

2.3. **Successive Overrelaxation Metod (SOR)**

$0 < w < 1$ şeklindeki w seçimleri için, işlemler "under-relaxation metodları" diye adlandırılırlar ve Gauss-Seidel Metod'la yakınsak olmayan bazı sistemlerde kullanılabiliyorlar. $1 < w$ şeklindeki, w seçimleri için , işlemler "over-relaxation metodları" diye adlandırılırlar. Bu metod Gauss-Seidel Metodun'da yakınsak olan sistemlerdeki yakınsamayı hızlandırır. Bu metodlar SOR diye kısaltılırlar (Successive over-relaxation). Bu metodlar kesin parçalı diferansiyel denklemlerin nümerik çözümlerinde kullanışlıdırlar.

SOR metodun avantajlarını göstermeden önce;

$a_{ii}x_i^{(k-1)} + r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}$ denklemini kullanarak;

$x_i^{(k)} = x_i^{(k-1)} + w\frac{r_{ii}^{(k)}}{a_{ii}}$ hesaplama amacı için yeniden formüle edilebilir:

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

SOR metodun matrisini belirtmek için şu şekilde tekrar yazabiliriz:

$a_{ii}x_i^{(k)} + w\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} = (1-w)a_{ii}x_i^{(k-1)} - w\sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} + wb_i$ vektör formunda olduğundan şunu elde ederiz: $(D - wL)x^{(k)} = [(1-w)D + wU]\,x^{(k-1)} + wb$ veya $x^{(k)} = (D - wL)^{-1}\,[(1-w)D + wU]\,x^{(k-1)} + w(D - wL)^{-1}b$

Eger;

$T_w = (D - wL)^{-1}\,[(1-w)D + wU]$ ve $c_w = w(D - wL)^{-1}b$ alırsak SOR metod şu formu alır:

$x^{(k)} = T_w x^{(k-1)} + c_w$

Teorem 2..5 *(Kahan) Eğer* $a_{ii} \neq 0$, $i = 1, 2, ..., n$, $\rho(T_w) \geq |w-1|$ *ise bu SOR metodun sadece* $0 < w < 2$ *aralığında yaklaştığını belirtir.*

Teorem 2..6 *(Ostrowski-Reich): Eğer A pozitif tanımlı bir matris ise ve* $0 < w < 2$ *ise; SOR metod ilk baştaki herhangi yaklaşık* $x^{(0)}$ *seçimi için yaklaşır.*

Teorem 2..7 *Eğer A pozitif tanımlı tridiagonal,* $\rho(T_g) = [\rho(T_j)]^2 < 1$ *ise SOR metod için uygun w seçimi* $w = \dfrac{2}{1 + \sqrt{1 - [\rho(T_j)]^2}}$ *dir. Bu seçim ile,* $\rho(T_w) = w - 1$ *'i elde ederiz.*

2.4. **Symmetric Successive Overrelaxation Metod (SSOR)**

A bir simetrik katsayı matrisi olduğunda SSOR simetrik matrise benzer bir iterasyon sonuç matrisiyle birleşir. SSOR ' un yaklaşma oranı (w' nun optimal değeri ile) genellikle SOR ' un w optimal değeri ile yaklaşma oranından daha yavaştır. (Young [217 , page 462])

SSOR iterasyonu ;

$B_1 = (D - wU)^{-1}(wL + (1-w)D)$,

$B_2 = (D - wU)^{-1}(wU + (1-w)D)$, olmak üzere,

$x^{(k)} = B_1 B_2 x^{(k-1)} + w(2-w)(D - wU)^{-1}D(D - wL)^{-1}b$

2.5. **Gradient Metodu**

$Ax = b$ denklem sistemini çözmek demek $x \in R^n$ olmak üzere $\Phi(y) = \frac{1}{2}y^T Ay - y^T b$ quadratic formunu minimize etmektir. Gerçekten Φ 'nin gradyenti şöyle verilir:

$\Delta\ \Phi(y) = \frac{1}{2}(A^T + A)y - b = Ay - b$

Sonuç olarak; eger $\Delta\ \Phi(x) = 0$ ise x orjinal sistemin bir çözümüdür.Aksine, x bir çözüm ise; $\Phi(y) = \Phi(x + (y - x)) = \Phi(x) + \frac{1}{2}(y-x)^T A(y-x)$

Ve böylece; $\Phi(y) > \Phi(x), \quad y \neq x$ olmak üzere

x foksiyonelinin minimizeri $\|\|_A$, A norm veya enerji norm olduĝunda eşitlik;
$\frac{1}{2}\|y-x\|_A^2 = \Phi(y) - \Phi(x)$ olur.

Problem böylece $x^{(0)} \in R^n$ noktasından başlayarak Φ 'nin minimizer x 'ini saptama, bu sebeple de x çözümüne mümkün oldugunca yaklaşan uygun seçimi yapma halini alır. x çözüm noktasına $x^{(0)}$ noktasından başlayarak katılan apaçık bilinmeyen optimal yön önceliklidir. Sonuçta, $d^{(0)}$ yönü boyunca $x^{(0)}$ 'dan bir adım kullanmak zorundayız, sonra yeni $x^{(1)}$ noktası için, daha sonra da yaklaşma gerçekleşene kadar iterasyonlara devam edecegiz.Böylece k adımda, $x^{(k+1)}$ şöyle hesaplanır:

$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ ($\alpha_k, d^{(k)}$ adımı boyunca sabitlenen deĝer)

Buradaki en doĝal fikir, Gradient Metodu veya "steepest descent" metodunun verdiĝi sonuç en büyük eĝim $\Delta\Phi(x^{(k)})$ ' nın alçalma yönünü almaktadır.

Başka bir deyişle; $\Delta\Phi(y) = \frac{1}{2}(A^T + A)y - b = Ay - b$ denklemine raĝmen;
$\Delta\Phi(x^{(k)}) = Ax^{(k)} - b = -r^{(k)}$' dır.

Bu nedenle Gradient 'deki Φ ' nin yönü ile kalanının yönü aynı anda rastlarlar, ve geçerli iterasyon kullanılarak hemen hesaplanabilir. Bu Richardson Metod ' u kadar iyi olan Gradient Metod ' un $d^{(k)} = r^{(k)}$ yönü boyunca her adımda ilerler.

α_k parametresini hesaplamak için parametresi α olan $\Phi(x^{(k+1)})$ fonksiyonunu yazalım.

$\Phi(x^{(k+1)}) = \frac{1}{2}(x^{(k)} + \alpha r^{(k)})^T A(x^{(k)} + \alpha r^{(k)}) - (x^{(k)} + \alpha r^{(k)})^{(T)} b$

Özetle,

$x^{(0)} \in R^n$ verilir.$k = 0, 1, ...$ yakınsama gerçekleşene kadar;

$r^{(k)} = b - Ax^{(k)}$ $\alpha^{(k)} = \frac{r^{(k)^T} r^{(k)}}{r^{(k)^T} A r^{(k)}}$ $x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}$.

Teorem 2..8 *Simetrik, pozitif tanımlı bir A matrisi alınırsa, başlangıç deĝeri $x^{(0)}$ ' ın herhangi bir seçimi için Gradient Metod yakınsaktır ve $\|.\|_A$ enerji normu olmak üzere*

$\left\|e^{(k+1)}\right\| \leq \frac{K_2(A)-1}{K_2(A)+1} \left\|e^{(k)}\right\|_A$ $k = 0, 1, ...,$ *' dır.*

2.6. **Conjugate Gradient Metod (CG)**

Hestenes ve Stiefel [HS] 'nin conjugate gradient metodu, *nxn* pozitif tanımlı lineer sistemi çözmek için aslında direkt metod olarak geliştirilmiştir. Direkt metod olarak Hestenes ve Stiefel 'in Conjugate Gradient metodu pivotlu Gauss-Eliminasyona göre genellikle daha az ö nemlidir. Çünkü her iki metod bir çözümü elde etmek için n adım gerektirir ve Conjugate Gradient metodun adımları hesapsal olarak Gauss-Eliminasyona göre daha pahalıdır. Bununla beraber tahmin edilebilir modellerdeki büyük sparse sistemleri çözmek için kullanılan iterative yaklaşma metodlarında çok

kullanılışlıdır. Bu problemler sıkça (genellikle) sınır deĝer problemlerinden doĝarlar. Matris hesaplamayı daha etkili yapmak için önceden koş ullandırıldığında iyi sonuçlar aşagı yukarı $\sqrt{n}$ adımda elde edilir. Bu yol kullanıldığında, metod Gauss-Eliminasyon'a ve önceden tartış ılan metodlara tercih edilir. Bundan sonra, A matrisinin pozitif tanımlı olduĝunu farzedeceĝiz.

x ve y $n-$boyutlu olduĝunda;

$\langle x, y\rangle = x^t \cdot y$ iç çarpımı kullanacagız.

A pozitif tanımlı olduĝu zaman; $x = 0$ olmadıkça

$\langle x, Ax\rangle = x^t Ax > 0$ Ayrıca A simetrik olduĝundan;

$x^t Ay = x^t A^t y = (Ax)^t y$ 'yi elde ederiz.Sonuçlara ek olarak her x, y için şunu elde ederiz.

$\langle x, Ay\rangle = \langle Ax, y\rangle$

Teorem 2..9 *$r^{(k)}$ fark vektörü $(k = 1, 2, ..., n)$ olduĝunda Conjugate Direction metod için , aşagıdaki denklemdeki koşulları yerine getirir.*

$\left\langle r^{(k)}, v^{(j)}\right\rangle = 0\ ,\ \ j = 1, 2, ..., k$

Hestenes ve Stiefel ' in Conjugate Gradient metodu iterative işlem (yöntem) boyunca arama yönü $\left\{v^{(k)}\right\}$ ' yı seçer. Çünkü $\left\{r^{(k)}\right\}$ fark vektörleri karşılıklı ortogonaldir. Yön vektörleri $\left\{v^{(1)}, v^{(2)}, ...\right\}$ ve yaklaşık $\left\{x^{(1)}, x^{(2)}, ...\right\}$ 'yi oluşturmak için , ilk baştaki yaklaşık $x^{(0)}$ ile başlarız ve ilk arama yönü $v^{(1)}$ gibi $r^{(0)} = b - Ax^{(0)}$ kullanırız. $v^{(1)}, ..., v^{(k-1)}$ ve $x^{(1)}, ..., x^{(k-1)}$ ' in $x^{(k-1)} = x^{(k-2)} + t_{k-1}v^{(k-1)}$ ile hesaplandığını farzedelim;

($\left\langle v^{(i)}, Av^{(j)}\right\rangle = 0$ *ve* $\left\langle r^{(i)}, r^{(j)}\right\rangle = 0$ $i \neq j$) olduĝunda,

Eĝer $x^{(k-1)}$ $Ax = b$ ' nin çözümü ise yapılır. Aksi halde; $r^{(k-1)} = b - Ax^{(k-1)} \neq 0$ 'dır.

$v^{(k)} = r^{(k-1)} + s_{k-1}v^{(k-1)}$ s_{k-1} seçmek isteriz, çünkü;

$\left\langle v^{(k-1)}, Av^{(k)}\right\rangle = 0$, $Av^{(k)} = Ar^{(k-1)} + s_{k-1}Av^{(k-1)}$ ve

$\left\langle v^{(k-1)}, Av^{(k)}\right\rangle = \left\langle v^{(k-1)}, Ar^{(k-1)}\right\rangle + s_{k-1}\left\langle v^{(k-1)}, Av^{(k-1)}\right\rangle$,

Böylece; $s_{k-1} = \frac{\left\langle v^{(k-1)}, Ar^{(k-1)}\right\rangle}{\left\langle v^{(k-1)}, Av^{(k-1)}\right\rangle}$ olduĝunda şunu elde ederiz:

$\left\langle v^{(k-1)}, Av^{(k)}\right\rangle = 0$

$t_k = \frac{\left\langle v^{(k)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle} = \frac{\left\langle r^{(k-1)} + s_{k-1}v^{k-1}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle} = \frac{\left\langle r^{(k-1)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle} + s_{k-1}\frac{\left\langle v^{(k-1)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle}$

Teoremde $\left\langle v^{(k-1)}, Av^{(k)}\right\rangle = 0$ ' dır. Bu nedenle;

$t_k = \frac{\left\langle r^{(k-1)}, r^{(k-1)}\right\rangle}{\left\langle v^{(k)}, Av^{(k)}\right\rangle}$ olur.Böylece ; $x^{(k)} = x^{(k-1)} + t_k v^{(k)}$

$r^{(k)}$'yı hesaplamak için A ile çarpıp b 'yi çıkarırız ve;

$Ax^{(k)} - b = Ax^{(k-1)} - b + t_k Av^{(k)}$ veya

$r^{(k)} = r^{(k-1)} - t_k Av^{(k)}$ böylece

$\left\langle r^{(k)}, r^{(k)} \right\rangle = \left\langle r^{(k-1)}, r^{(k)} \right\rangle - t_k \left\langle Av^{(k)}, r^{(k)} \right\rangle = -t_k \left\langle r^{(k)}, Av^{(k)} \right\rangle$

$\left\langle r^{(k-1)}, r^{(k-1)} \right\rangle = t_k \left\langle v^{(k)}, Av^{(k)} \right\rangle$

$s_k = -\frac{\left\langle v^{(k)}, Ar^{(k)} \right\rangle}{\left\langle v^{(k)}, Av^{(k)} \right\rangle} = -\frac{\left\langle r^{(k)}, Av^{(k)} \right\rangle}{\left\langle v^{(k)}, Av^{(k)} \right\rangle} = \frac{(1/t_k)\left\langle r^{(k)}, r^{(k)} \right\rangle}{(1/t_k)\left\langle r^{(k-1)}, r^{(k-1)} \right\rangle} = \frac{\left\langle r^{(k)}, r^{(k)} \right\rangle}{\left\langle r^{(k-1)}, r^{(k-1)} \right\rangle}$

Kısaca; $r^{(0)} = b - Ax^{(0)}$; $v^{(1)} = r^{(0)}$ ve $k = 1, 2, ..., n$

$t_k = \frac{\left\langle r^{(k-1)}, r^{(k-1)} \right\rangle}{\left\langle v^{(k)}, Av^{(k)} \right\rangle}$ $\quad x^{(k)} = x^{(k-1)} + t_k v^{(k)}$ $r^{(k)} = r^{(k-1)} - t_k Av^{(k)}$

$s_k = \frac{\left\langle r^{(k)}, r^{(k)} \right\rangle}{\left\langle r^{(k-1)}, r^{(k-1)} \right\rangle}$ $\quad v^{(k+1)} = r^{(k)} + s_k v^{(k)}$

2.7. **Preconditioned Conjugate Gradient Metod (PCG)**

Yukarıda Conjugate Gradient Metod'unda verilen formülleri geliştirerek ; "preconditioning(önkoşul)" u içeren Preconditioned Conjugate Gradient Metod elde edilir. Eger A matrisi iyi koşullanmış bir matris ise Conjugate Gradient metod hatalar için yüksek hassasiyet gösterir. Bu nedenle, kesin çözümün n adımda bulunması gerekmesine raĝmen bu durum çoĝunlukla olmaz. Direkt metod gibi, Conjugate Gradient metod Gauss-Eliminasyon kadar iyi deĝildir. Conjugate Gradient metodun temel kullanımı iyi koşullanmış sistemlerdedir. Bu durumda, kabul edilebilir yaklaşık sonuç genellikle $\sqrt{n}$ adımda bulunur. Bu metodu iyi koşullanmış sisteme uygulamak için tekil olmayan bir C matrisi seçmek isteriz, çünkü ;

$\tilde{A} = C^{-1}A(C^{-1})^t$ iyi koşullanmıştır.

Notasyonu kolaylaştırmak için; C^{-t} matrisini kullanacagız ($(C^{-1})^t$ matrisini içerdigi için)

$\tilde{A}\tilde{x} = \tilde{b}$ lineer sistemini düşünelim ($\tilde{x} = C^t x$ ve $\tilde{b} = C^{-1}b$ oldugu durumlarda)

Daha sonra;

$\tilde{A}\tilde{x} = (C^{-1}AC^{-t})(C^t x) = C^{-1}Ax$

Böylece $\tilde{A}\tilde{x} = \tilde{b}$ denklemini $\tilde{x}$ için çözebiliriz , daha sonra C^{-t} ile çarparak x 'i bulabiliriz. Bununla beraber ; $v^{(k+1)} = r^{(k)} + s_k v^{(k)}$ denklemini , $\tilde{r}^{(k)}$, $\tilde{v}^{(k)}$, $\tilde{t}^{(k)}$, $\tilde{x}^{(k)}$ ve $\tilde{s}^{(k)}$ 'yı kullanarak yeniden yazmak yerine preconditioning ile birleştiririz. Çünkü ;

$\tilde{x}^{(k)} = C^t x^{(k)}$ 'dır.

Şunu elde ederiz:

$\tilde{r}^{(k)} = \tilde{b} - \tilde{A}\tilde{x}^{(k)} = C^{-1}b - (C^{-1}AC^{-t})C^t x^{(k)} = C^{-1}(b - Ax^{(k)}) = C^{-1}r^{(k)}$

$\tilde{v}^{(k)} = C^t v^{(k)}$ ve $w^{(k)} = C^{-1}r^{(k)}$ alalım.Sonra

$\tilde{s}_k = \frac{\left\langle \tilde{r}^{(k)}, \tilde{r}^{(k)} \right\rangle}{\left\langle \tilde{r}^{(k-1)}, \tilde{r}^{(k-1)} \right\rangle} = \frac{\left\langle C^{-1}\tilde{r}^{(k)}, C^{-1}\tilde{r}^{(k)} \right\rangle}{\left\langle C^{-1}\tilde{r}^{(k-1)}, C^{-1}\tilde{r}^{(k-1)} \right\rangle}$,

$\tilde{s}_k = \frac{\left\langle w^{(k)}, w^{(k)} \right\rangle}{\left\langle w^{(k-1)}, w^{(k-1)} \right\rangle}$

Böylece;

$\tilde{t}_k = \frac{\left\langle \tilde{r}^{(k-1)}, \tilde{r}^{(k-1)} \right\rangle}{\left\langle \tilde{v}^{(k)}, \tilde{A}\tilde{v}^{(k)} \right\rangle} = \frac{\left\langle C^{-1}\tilde{r}^{(k-1)}, C^{-1}\tilde{r}^{(k-1)} \right\rangle}{\left\langle C^t v^{(k)}, C^{-1}AC^{-t}C^t v^{(k)} \right\rangle} = \frac{\left\langle w^{(k-1)}, w^{(k-1)} \right\rangle}{\left\langle C^t v^{(k)}, C^{-1}Av^{(k)} \right\rangle}$ ve

$\tilde{t}_k = \frac{\langle w^{(k-1)}, w^{(k-1)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$

Ayrıca;

$\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \tilde{t}_k \tilde{v}^{(k)}$ böylece; $C^t x^{(k)} = C^t x^{(k-1)} + \tilde{t}_k C^t v^{(k)}$ Ve;

$x^{(k)} = x^{(k-1)} + \tilde{t}_k v^{(k)}$

Devam edilirse;

$\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \tilde{t}_k \tilde{A} \tilde{v}^{(k)}$

$C^{-1} r^{(k)} = C^{-1} r^{(k-1)} - \tilde{t}_k C^{-1} A C^{-t} \tilde{v}^{(k)}$, $r^{(k)} = r^{(k-1)} - \tilde{t}_k A C^{-t} C^t v^{(k)}$

Ve,

$r^{(k)} = r^{(k-1)} - \tilde{t}_k A v^{(k)}$ .Sonuç olarak;

$\tilde{v}^{(k+1)} = \tilde{r}^{(k)} + \tilde{s}_k \tilde{v}^{(k)}$ ve $C^t v^{(k+1)} = C^{-1} r^{(k)} + \tilde{s}_k C^t v^{(k)}$

$v^{(k+1)} = C^{-t} C^{-1} r^{(k)} + \tilde{s}_k v^{(k)} = C^{-t} w^{(k)} + \tilde{s}_k v^{(k)}$

2.8. **BiConjugate Gradient Metod (BiCG)**

Conjugate Gradient Metod simetrik olmayan sistemler için uygun değildir. Çünkü fark vektörlri kısa tekrarlamalarla ortogonal yapılamaz. Biconjugate Gradient Metod karşılıklı iki ortogonal dizi ile residünün ortogonal dizisinin yerini alarak yeni bir yaklaşım getirir.

Conjugate Gradient Metod' daki residüler için güncelleme ilişkileri, Biconjugate Metod' daki benzer olan ama A yerine A^T' ye dayandırılan ilişkiler genişletilir. Böylece residülerin iki dizisini güncelleriz.

$r^{(k)} = r^{(k-1)} - \alpha_i A p^{(k)}, \tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_i A^T \tilde{p}^{(k)},$

araştırma yönünün iki dizisi;

$p^{(k)} = r^{(k+1)} + \beta_{i-1} p^{(k-1)}, \tilde{p}^{(k)} = \tilde{r}^{(k-1)} + \beta_{i-1} \tilde{p}^{(k-1)}$

Seçimler;

$\alpha_i = \frac{\tilde{r}^{(k-1)^T} r^{(k-1)}}{\tilde{p}^{(k)^T} A p^{(k)}}$, $\beta_i = \frac{\tilde{r}^{(k)^T} r^{(k)}}{\tilde{r}^{(k-1)^T} r^{(k-1)}}$

$\tilde{r}^{(k)^T} r^{(j)} = \tilde{p}^{(k)^T} A p^{(j)} = 0$ $\quad k \neq j$ bi-ortogonalleştirmeyi sağlar.

Pozitif tanımlı simetrik sistemler için bu metod Conjugate Gradient Metod' la aynı sonuçları verir ama iki kat fazla iterasyon gerektirir. Simetrik olmayan matrisler için, residual' in normunda önemli bir azalma olduğunda her evrede işleme tabi tutulduğu gösterilir.

Biortogonalizasyon diye tanımlanan metodu kullanmanın bir yolu, özdeğerleri hesaplamaktır($n \to \infty$). T_n'nin bazı özdeğerleri, A'nın özdeğerlerine hızla yaklaşabilir. Diğer bir uygulama; tersi bulunan $Ax = b$ denklem sistemlerinin çözümüdür. Bu tipteki klasik algoritma "Biconjugate Gradients" veya "BiCG" diye bilinir.

BiCG: $\mathbf{r}_n \perp \langle w_1 A^* w_1, ..., (A^*)^{n-1} w_1 \rangle$

Burada $w_1 \in C^m$ $w_1^* v_1 = 1$ 'i sağlayan keyfi bir vektördür. Uygulamada bazen $w_1 = v_1 / \|v_1\|_2$ alınır. Bu seçim $\|r_n\|_2$ 'yi küçültmez ve bu, iterasyon sayısını mümkün olduğunca azaltmak bakış açısına uygun değildir.

3. MATERYAL VE YÖNTEM

JACOBI

Choose an initial guess $x^{(0)}$ to the solution x.
for $k = 1, 2, ...$
  for $i = 1, 2, ..., n$
    $\overline{x}^{(i)} = 0$
    for $j = 1, 2, ..., i-1, i+1, ..., n$
      $\overline{x}^{(i)} = \overline{x}^{(i)} a_{i,j} x_j^{(k-1)}$
    end
    $\overline{x}^{(i)} = (b_i - \overline{x}^{(i)})/a_{i,i}$
  end
  $x^{(k)} = \overline{x}$
  check convergence; continue if neccesary
end

Şekil 3.1. Jacobi algoritması

GAUSS-SEIDEL

Choose an initial guess $x^{(0)}$ to the solution x.
for $k = 1, 2, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
      $\sigma = \sigma + a_{i,j} x_j^{(k)}$
    for $j = i+1, ..., n$
      $\sigma = \sigma + a_{i,j} x_j^{(k-1)}$
    end
    $x_i^{(k)} = (b_i - \sigma)/a_{i,i}$
  end
  check convergence; continue if neccesary
end

Şekil 3.2. Gauss-Seidel algoritması

SOR

Choose an initial guess $x^{(0)}$ to the solution x.
for $k = 1, 2, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
    $\sigma = \sigma + a_{i,j} x_j^{(k)}$
    end
    for $j = i+1, ..., n$
    $\sigma = \sigma + a_{i,j} x_j^{(k-1)}$
    end
    $\sigma = (b_i - \sigma)/a_{i,i}$
    $x_i^{(k)} = x_i^{(k-1)} + w(\sigma - x_i^{(k-1)})$
  end
  check convergence; continue if neccesary
end

Şekil 3.3. SOR algoritması

SSOR
Choose an initial guess $x^{(0)}$ to the solution x.
for $k = 1, 2, ...$
  for $i = 1, 2, ..., n$
    $\sigma = 0$
    for $j = 1, 2, ..., i-1$
      $\sigma = \sigma + a_{i,j} x_j^{(k-\frac{1}{2})}$
    end
    for $j = i+1, ..., n$
    $\sigma = \sigma + a_{i,j} x_j^{(k-1)}$
    end
    $\sigma = (b_i - \sigma)/a_{i,i}$
    $x_i^{(k-\frac{1}{2})} = x_i^{(k-1)} + w(\sigma - x_i^{(k-1)})$
    end
    for $i = n, n-1, ..., 1$
      $\sigma = 0$
      for $j = 1, 2, ..., i-1$
        $\sigma = \sigma + a_{i,j} x_j^{(k-\frac{1}{2})}$
      end
      for $j = i+1, ..., n$
      $\sigma = \sigma + a_{i,j} x_j^{(k)}$
      end
      $x_i^{(k)} = x_i^{(k-\frac{1}{2})} + w(\sigma - x_i^{(k-\frac{1}{2})})$
      end
      check convergence; continue if neccesary
end

Şekil 3.4. SSOR algoritması

$x_0 =$ initial guess
$r_0 = b - Ax_0$
$k = 0$
while $r_k \neq 0$
    $k = k + 1$
    $\alpha_k = r_{k-1}^T r_{k-1} / r_{k-1}^T A r_k$ 1
    $x_k = x_{k-1} + \alpha_k r_{k-1}$
    $r_k = b - Ax_k$
end

Şekil 3.5. Gradient algoritması

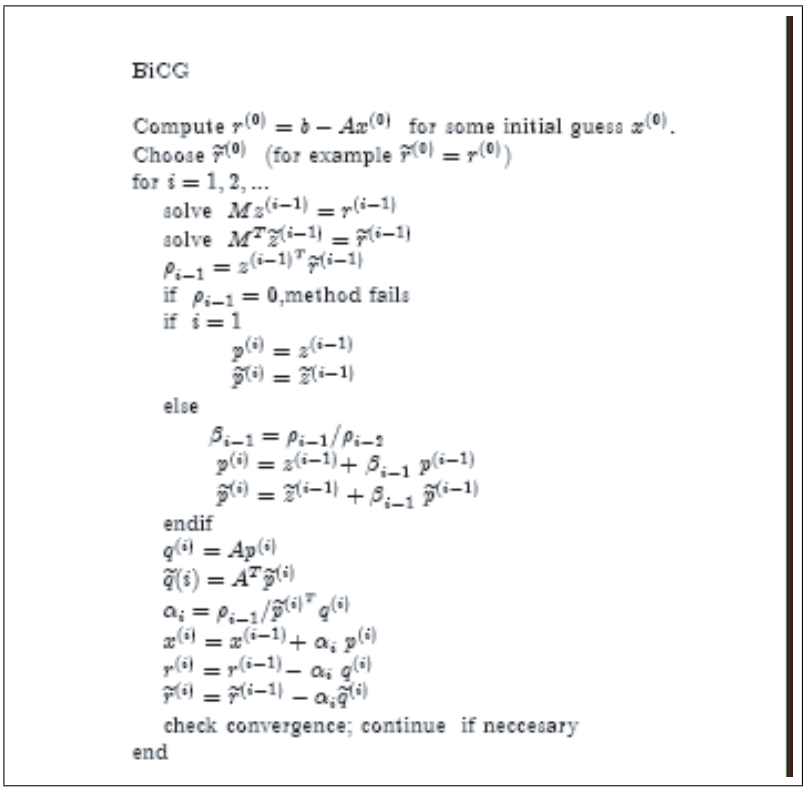
CG
$x^{(0)}$ is initial guess, $r^{(0)} = b - Ax^{(0)}$
for $i = 1, 2, \ldots$
    $\rho_{i-1} = r_{i-1}^T r_{i-1}$
    if $i = 1$
        $p_i = r_{i-1}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p_i = r_{i-1} + \beta_{i-1} \rho_{i-1}$
    endif
    $q_i = Ap_i$
    $\alpha_i = \rho_{i-1} / p_i^T q_i$
    $x_i = x_{i-1} + \alpha_i p_i$
    $r_i = r_{i-1} - \alpha_i q_i$
    if x_i accurate enough then quit
end

Şekil 3.6. CG algoritması

PCG

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
    solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
    if $i = 1$
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1} / p^{(i)^T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    check convergence; continue if neccesary
end

Şekil 3.7. PCG algoritması



BiCG

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$.
Choose $\tilde{r}^{(0)}$ (for example $\tilde{r}^{(0)} = r^{(0)}$)
for $i = 1, 2, \ldots$
solve $Mz^{(i-1)} = r^{(i-1)}$
solve $M^T\tilde{z}^{(i-1)} = \tilde{r}^{(i-1)}$
$\rho_{i-1} = z^{(i-1)^T}\tilde{r}^{(i-1)}$
if $\rho_{i-1} = 0$,method fails
if $i = 1$
$p^{(i)} = z^{(i-1)}$
$\tilde{p}^{(i)} = \tilde{z}^{(i-1)}$
else
$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
$p^{(i)} = z^{(i-1)} + \beta_{i-1}\, p^{(i-1)}$
$\tilde{p}^{(i)} = \tilde{z}^{(i-1)} + \beta_{i-1}\, \tilde{p}^{(i-1)}$
endif
$q^{(i)} = Ap^{(i)}$
$\tilde{q}^{(i)} = A^T\tilde{p}^{(i)}$
$\alpha_i = \rho_{i-1}/\tilde{p}^{(i)^T} q^{(i)}$
$x^{(i)} = x^{(i-1)} + \alpha_i\, p^{(i)}$
$r^{(i)} = r^{(i-1)} - \alpha_i\, q^{(i)}$
$\tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i \tilde{q}^{(i)}$
check convergence; continue if neccesary
end

Şekil 3.8. BiCG algoritması

4. ARAŞTIRMA BULGULARI

Bu bölümde, önceki bölümde verilen iterasyon metodlarının sparse lineer sistemlerine uygulamasını vereceğiz. Bu örnekler, matematik ve mühendislik alanında karşılaşılan sparse lineer sistemler olup, kullanılan sistemler için aşağıdaki yöntemler izlenmiştir:

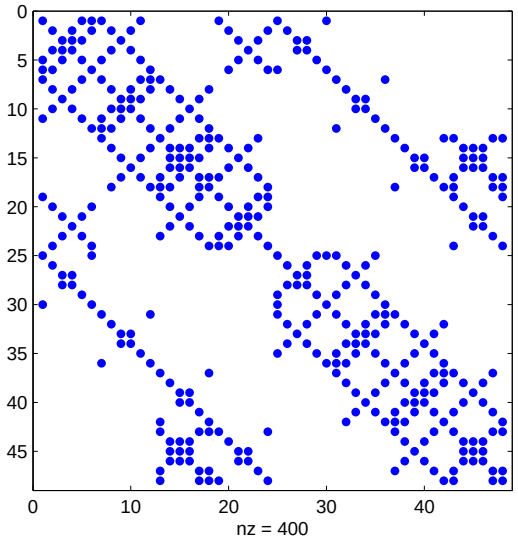
* $A_m \in R^{mxm}$ matrisi, $[-1,1]^2$ bölgesinde diferansiyel Laplace operatörünün ayrıştırılması ile elde edilmiştir. Bu matrisler, MATLAB'de $m = (n-2)^2$ olduğunda $G = numgrid('S', n')$ komutuyla $A = delsq(G)$ olur. 64×64, 324×324 ve 1444×1444 boyutlu matrisler bu şekilde elde edilmiştir.

* 48×48, 112×112, 132×132, 1074×1074 ve 2003×2003 boyutlu matrisler John Lewis' in (Boeing Computer Services) 1982 yılında yaptığı Yapısal Mühendislikteki Dinamik Analiz alanındaki çalışmalarından alınmıştır. (http://math.nist.gov/Matrix Market/)

* Tüm bilgisayar hesaplamalarında tolerans 10^{-6} dır.

* Başlangıç noktası olarak alınan $x^{(0)}$, tüm hesaplamalarda 0(sıfır) vektörü alınmıştır.

* SOR metod için yapılan hesaplamalarda $w = 1.5$ dir.



Şekil 4.1. 48x48 boyutlu sparse matrisi

Tablo 4.1. 48x48 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	50	24.2949
gauss-seidel	50	0.8986
sor	50	1.4552
ssor	-	-
gradient	50	0.9385
CG	50	2.2044
PCG	25	5.7656e-008
BiCG	50	2.2044

Tablo 4.2. 48x48 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	100	3.0392e+003
gauss-seidel	100	0.6761
sor	100	0.7647
ssor	-	-
gradient	100	0.9302
CG	100	2.8742
PCG	25	5.7656e-008
BiCG	100	2.8742

Tablo 4.3. 48x48 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	200	4.7789e+007
gauss-seidel	200	0.4158
sor	200	0.2832
ssor	-	-
gradient	200	0.9154
CG	136	4.9499e-007
PCG	25	5.7656e-008
BiCG	136	4.9499e-007

Tablo 4.4. 48x48 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata değişimi tablosu

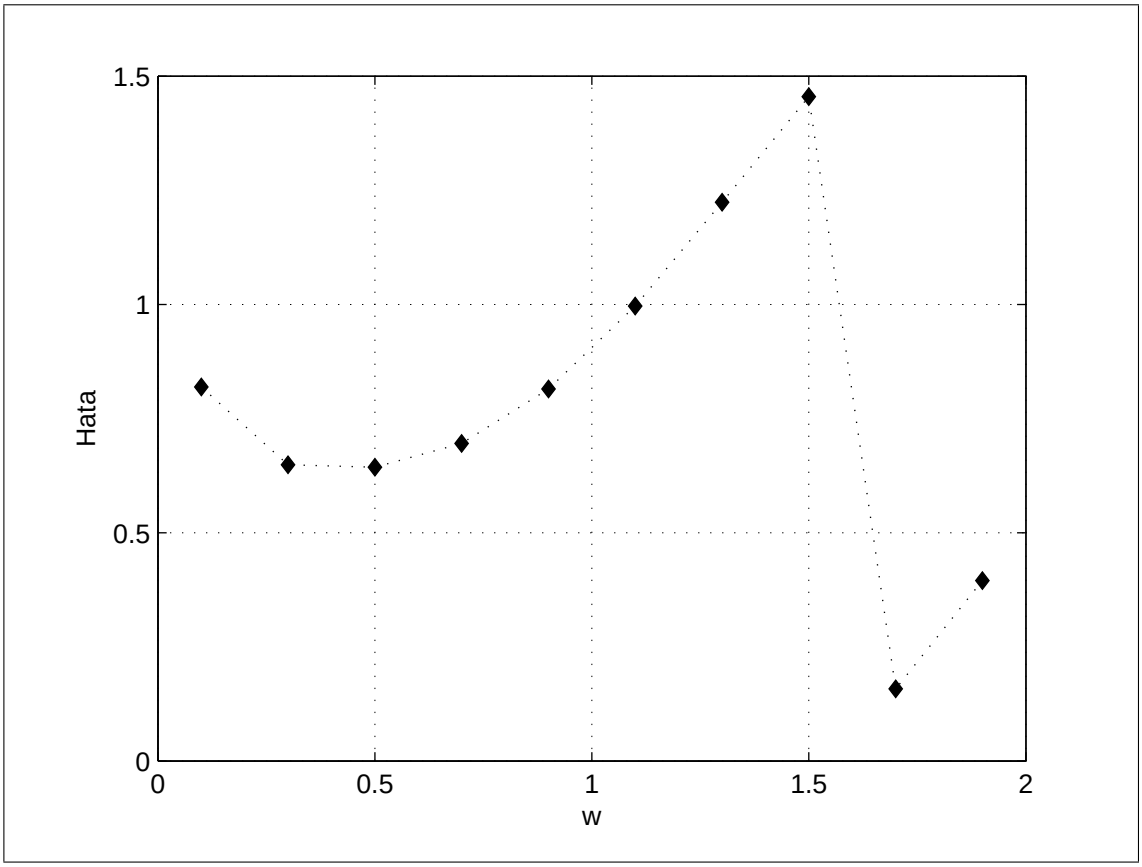
METOD	**İTERASYON SAYISI**	**HATA**
jacobi	400	1.1816e+016
gauss-seidel	400	0.3822
sor	400	0.0436
ssor	-	-
gradient	400	0.8911
CG	136	4.9499e-007
PCG	25	5.7656e-008
BiCG	136	4.9499e-007

Tablo 4.5. 48x48 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata değişimi tablosu

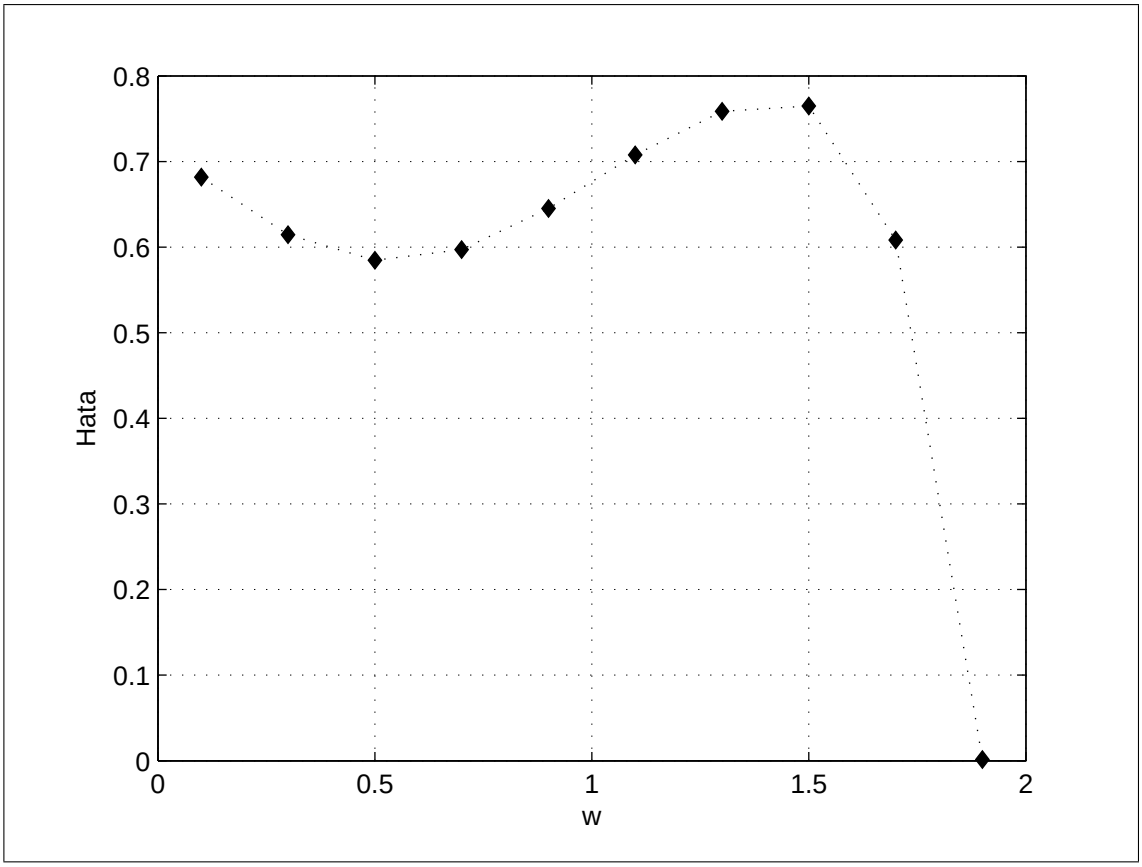
METOD	**İTERASYON SAYISI**	**HATA**
jacobi	600	$2.9214e{+}024$
gauss-seidel	600	0.3822
sor	600	0.0067
ssor	-	-
gradient	600	0.8713
CG	136	4.9499e-007
PCG	25	5.7656e-008
BiCG	136	4.9499e-007

Tablo 4.6. SSOR için 48x48 boyutlu sparse matrisin w, iterasyon sayısı ve hata değişimi tablosu

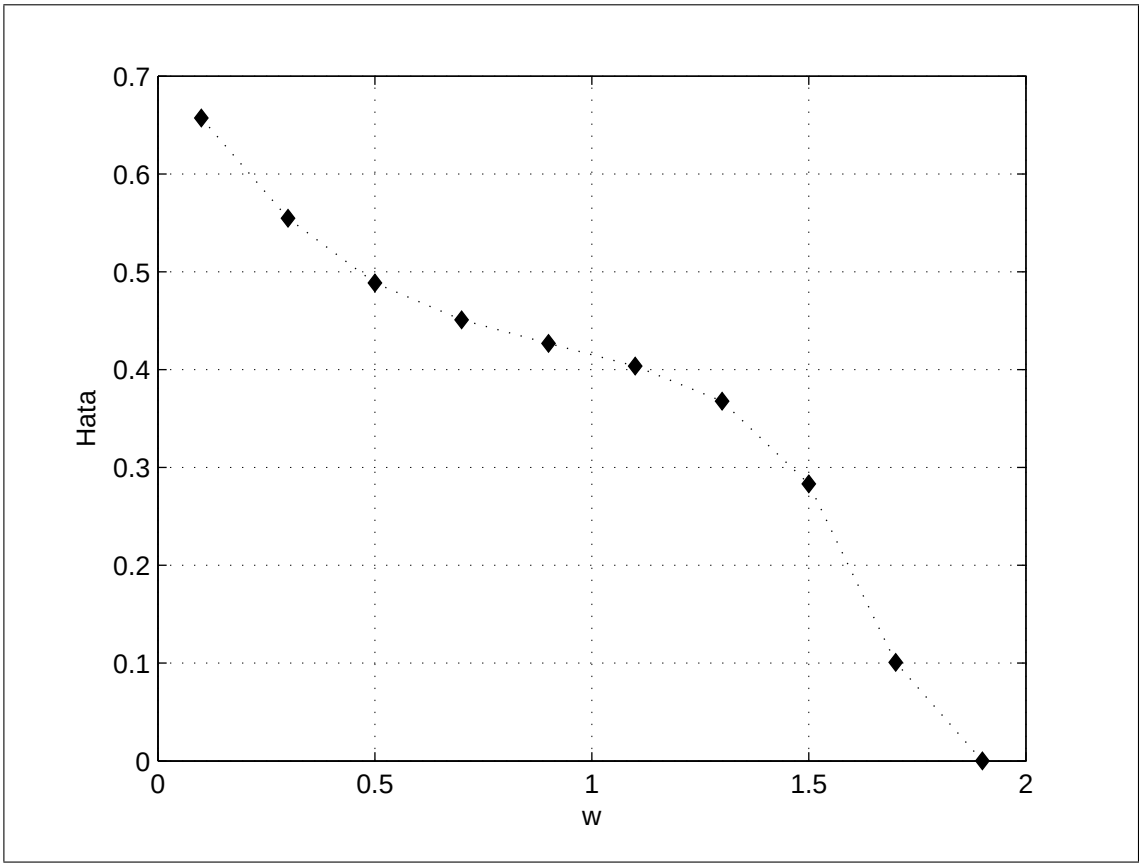
w	**İTERASYON SAYISI**	**HATA**
1.9	52311	3.90117e-008
1.7	16025	3.9137e-008
1.5	9113	3.9579e-008
1.3	6510	4.0351e-008
1.1	5494	4.1232e-008
0.9	5454	3.9353e-008
0.7	6387	3.5600e-008
0.5	8870	3.4505e-008
0.3	15529	3.4967e-008
0.1	50617	3.5313e-008



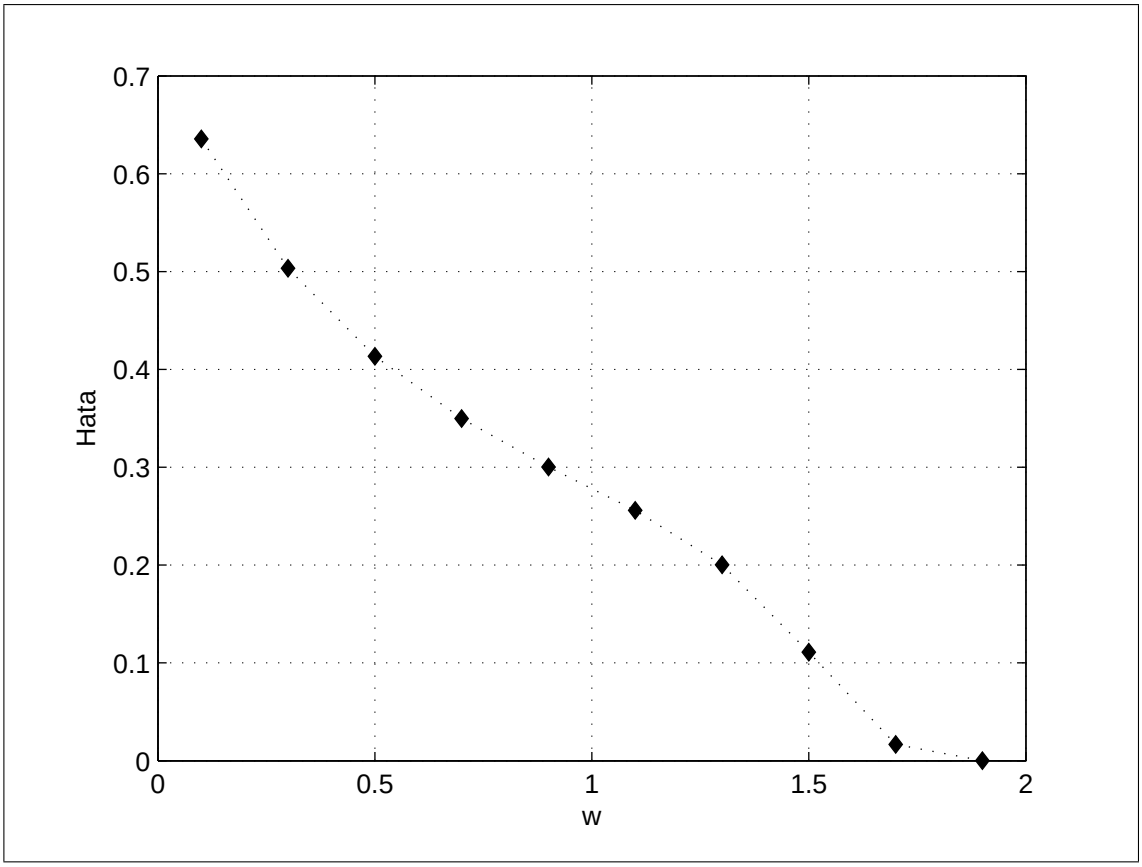
Şekil 4.2. SOR metodunun 50 iterasyon için w 'ya göre hata değişimi



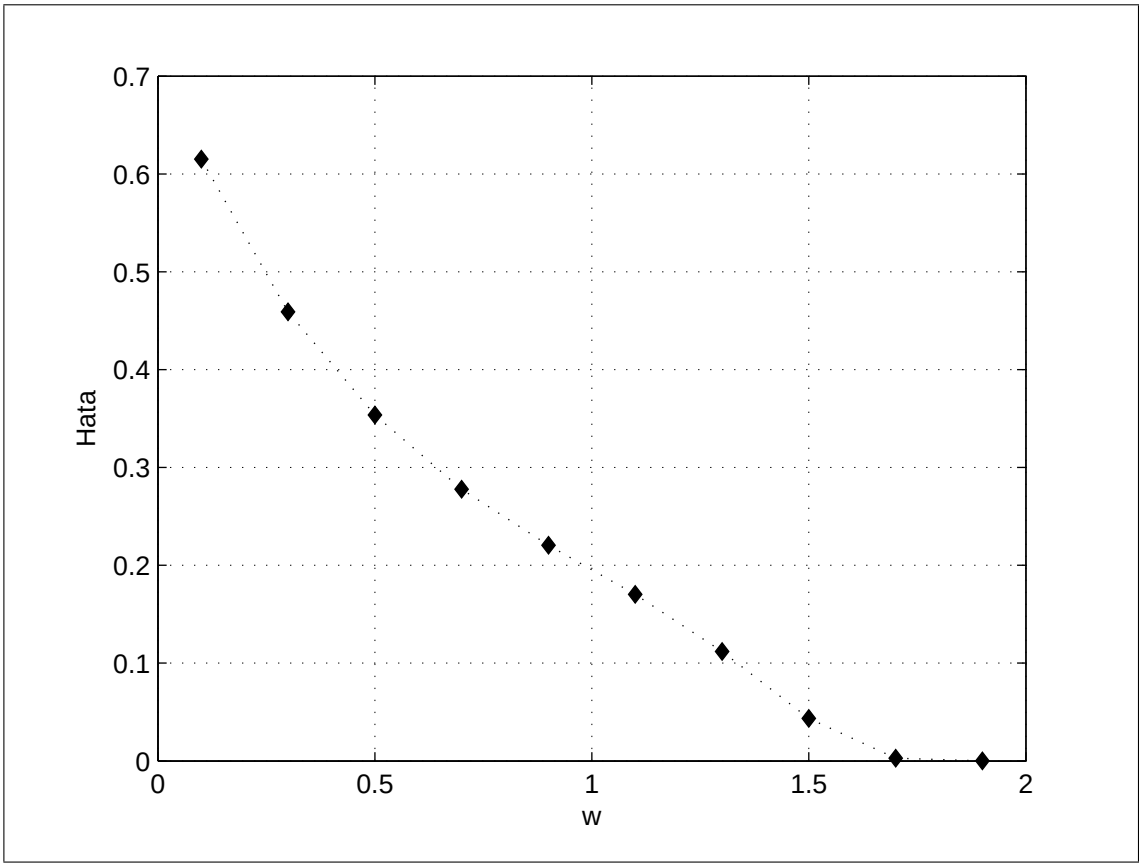
Şekil 4.3. SOR metodunun 100 iterasyon için w 'ya göre hata değişimi



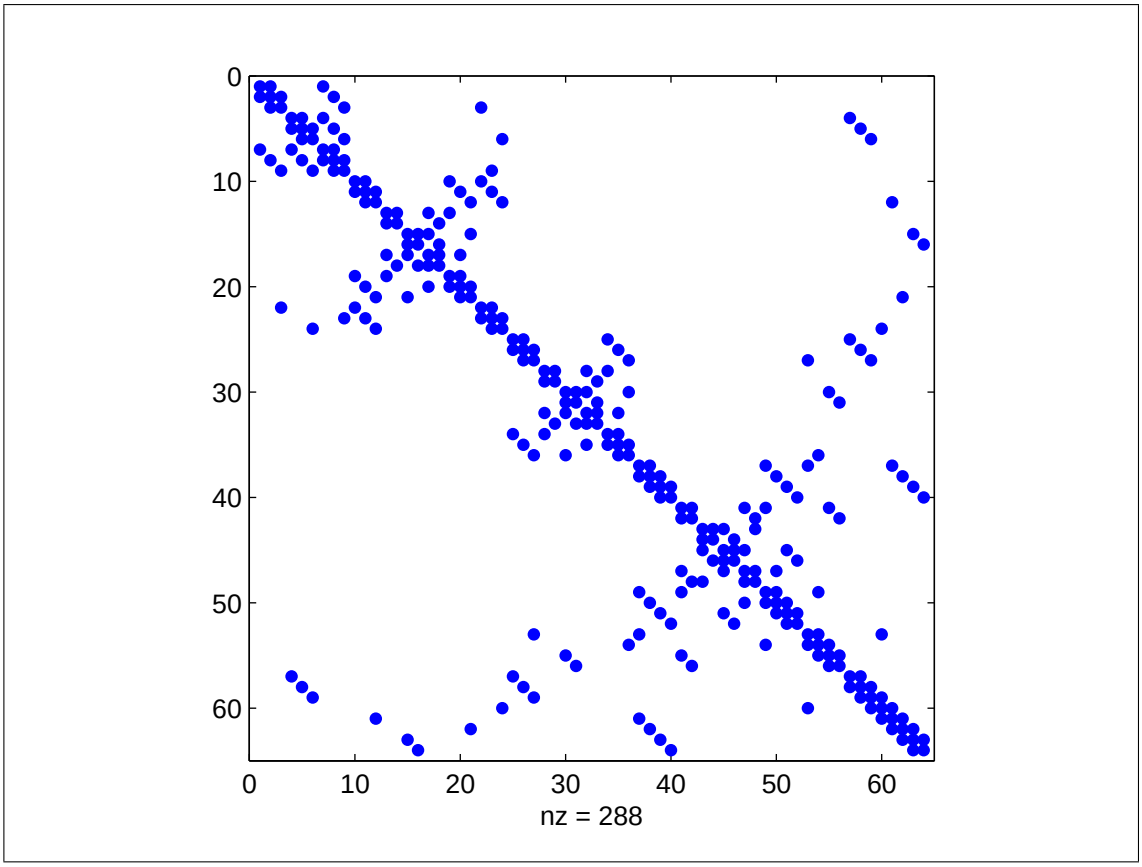
Şekil 4.4. SOR metodunun 200 iterasyon için w 'ya göre hata değişimi



Şekil 4.5. SOR metodunun 300 iterasyon için w 'ya göre hata değişimi



Şekil 4.6. SOR metodunun 400 iterasyon için w 'ya göre hata değişimi



Şekil 4.7. 64x64 boyutlu sparse matrisi

Tablo 4.7. 64x64 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	50	0.0398
gauss-seidel	50	0.0022
sor	28	9.6866e-007
ssor	-	-
gradient	50	0.0203
CG	10	1.6060e-015
PCG	11	4.2503e-007
BiCG	10	1.1291e-016

Tablo 4.8. 64x64 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	100	0.0018
gauss-seidel	100	7.3620e-006
sor	28	9.6866e-007
ssor	-	-
gradient	100	6.6191e-004
CG	10	1.6060e-015
PCG	11	4.2503e-007
BiCG	10	1.1291e-016

Tablo 4.9. 64x64 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

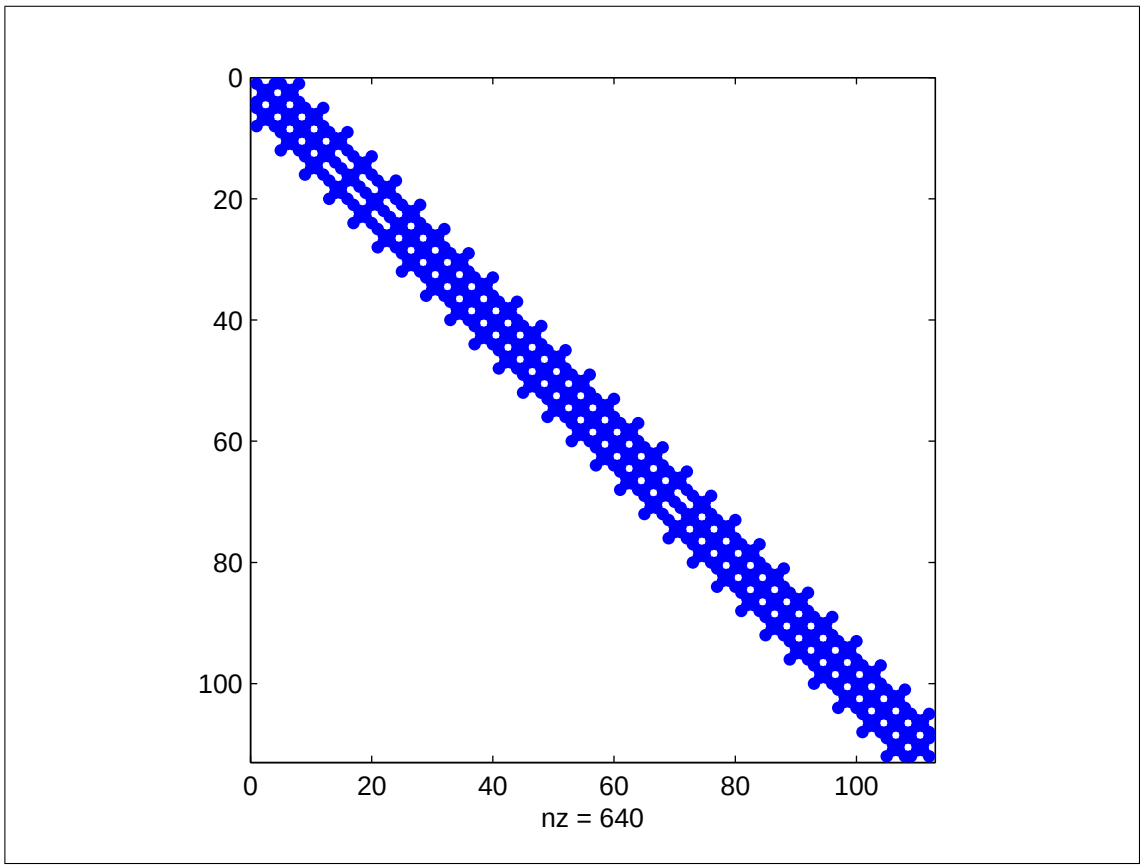
METOD	İTERASYON SAYISI	HATA
jacobi	200	1.3885e-005
gauss-seidel	200	7.3620e-006
sor	28	9.6866e-007
ssor	-	-
gradient	196	9.2888e-007
CG	10	1.6060e-015
PCG	11	4.2503e-007
BiCG	10	1.1291e-016

Tablo 4.10. 64x64 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	400	1.3885e-005
gauss-seidel	400	7.3620e-006
sor	28	9.6866e-007
ssor	-	-
gradient	196	9.2888e-007
CG	10	1.6060e-015
PCG	11	4.2503e-007
BiCG	10	1.1291e-016

Tablo 4.11. 64x64 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	600	1.3885e-005
gauss-seidel	600	7.3620e-006
sor	28	9.6866e-007
ssor	-	-
gradient	196	9.2888e-007
CG	10	1.6060e-015
PCG	11	4.2503e-007
BiCG	10	1.1291e-016



Şekil 4.8. 112x112 boyutlu sparse matrisi

Tablo 4.12. 112x112 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	50	2.8152
Gauss-seidel	50	6.8575
SOR	50	14.3623
SSOR	-	-
Gradient	50	0.9259
CG	50	5.8838
PCG	50	3.2599
BiCG	50	5.8838

Tablo 4.13. 112x112 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	2.8152
Gauss-seidel	100	6.8575
SOR	100	11.8018
SSOR		
Gradient	100	0.9181
CG	100	2.0944
PCG	74	6.1466e-007
BiCG	100	2.0944

Tablo 4.14. 112x112 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

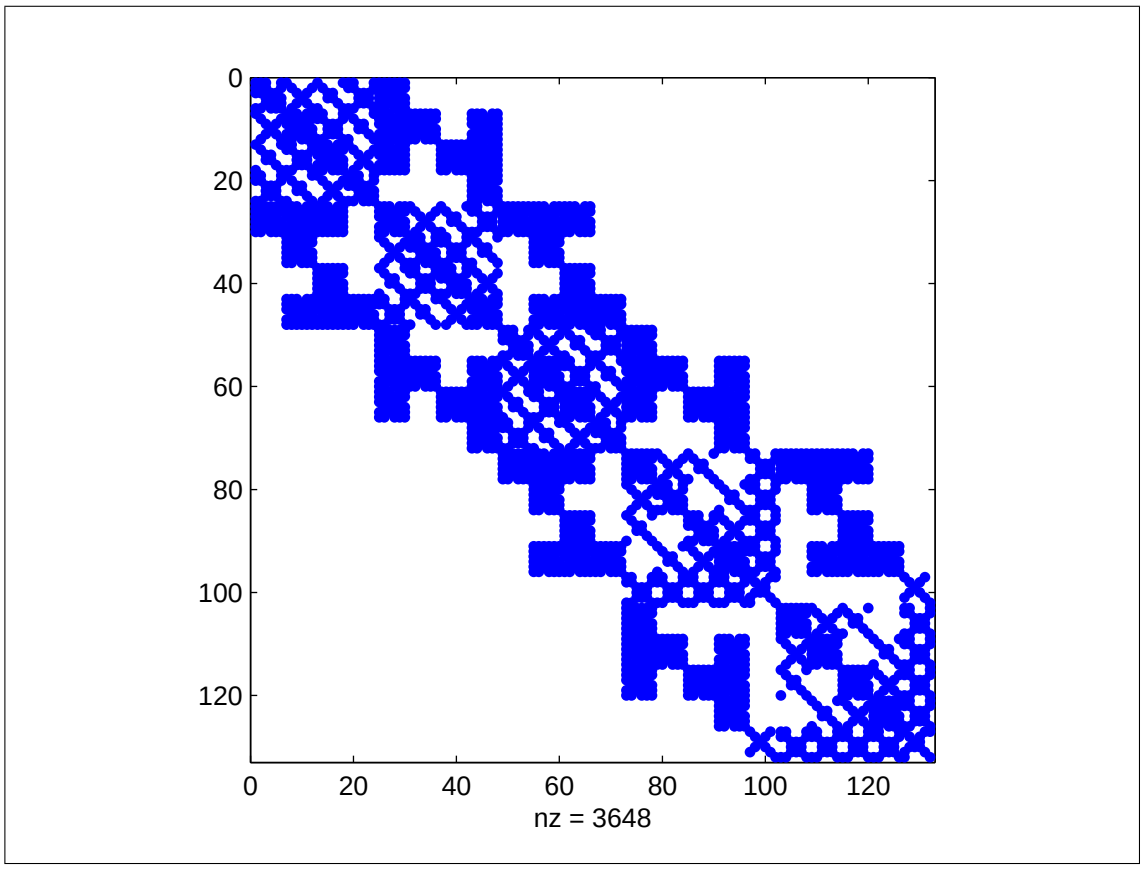
METOD	İTERASYON SAYISI	HATA
Jacobi	200	2.8152
Gauss-seidel	200	6.8575
SOR	200	9.3592
SSOR	-	-
Gradient	200	0.9134
CG	200	0.3816
PCG	74	6.1466e-007
BiCG	200	0.3816

Tablo 4.15. 112x112 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	400	2.8152
Gauss-seidel	400	6.8575
SOR	400	6.3243
SSOR	-	-
Gradient	400	0.9096
CG	400	0.0039
PCG	74	6.1466e-007
BiCG	400	0.0039

Tablo 4.16. 112x112 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	600	2.8152
Gauss-seidel	600	6.8575
SOR	600	4.4997
SSOR	-	-
Gradient	600	0.9068
CG	583	6.0542e-007
PCG	74	6.1466e-007
BiCG	583	6.0542e-007



Şekil 4.9. 132x132 boyutlu sparse matrisi

Tablo 4.17. 132x132 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	50	9.8167e+006
Gauss-seidel	50	2.0982
SOR	50	4.6269
SSOR	-	-
Gradient	50	0.8598
CG	50	3.5156
PCG	37	7.1126e-007
BiCG	50	3.5156

Tablo 4.18. 132x132 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	2.8696e+015
Gauss-seidel	100	1.8299
sor	100	3.0611
ssor	-	-
gradient	100	0.8577
CG	100	13.8631
PCG	37	7.1126e-007
BiCG	100	13.8631

Tablo 4.19. 132x132 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

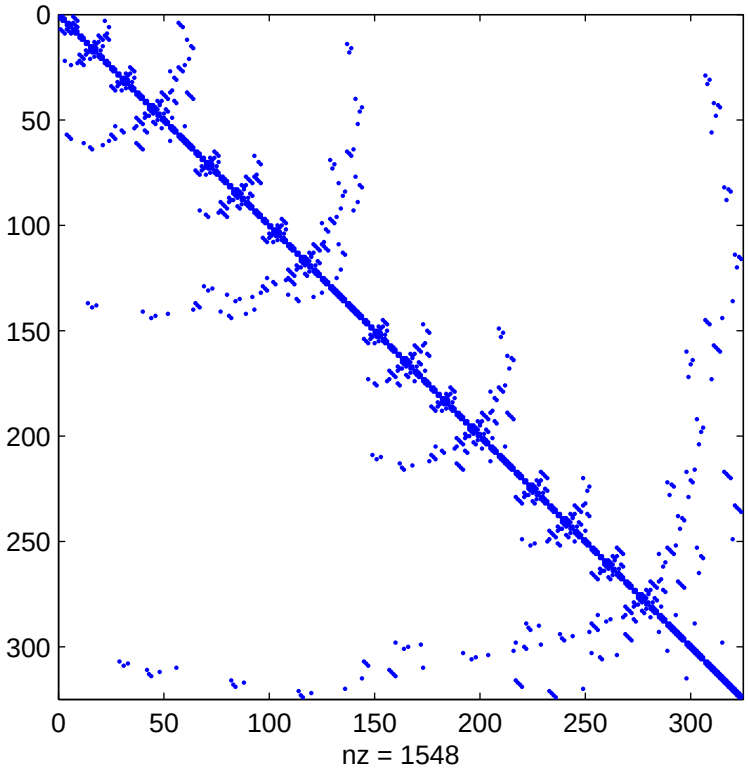
METOD	İTERASYON SAYISI	HATA
jacobi	200	2.4537e+032
gauss-seidel	200	1.3930
sor	200	1.3382
ssor	-	-
gradient	200	0.8537
CG	200	0.9251
PCG	37	7.1126e-007
BiCG	200	0.9251

Tablo 4.20. 132x132 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	400	1.7940e+066
gauss-seidel	400	0.8071
sor	400	0.2557
ssor	-	-
gradient	400	0.8481
CG	400	0.0121
PCG	37	7.1126e-007
BiCG	400	0.0121

Tablo 4.21. 132x132 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	600	1.3117e+100
gauss-seidel	600	0.4677
sor	600	0.0489
ssor	-	-
gradient	600	0.8427
CG	543	9.2454e-007
PCG	37	7.1126e-007
BiCG	543	9.2454e-007



Şekil 4.10. 324x324 boyutlu sparse matrisi

Tablo 4.22. 324x324 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	50	0.4248
Gauss-Seidel	50	0.2492
SOR	50	0.0246
SSOR	-	-
Gradient	50	0.4347
CG	28	8.8871e-007
PCG	20	7.7703e-007
BiCG	28	8.8871e-007

Tablo 4.23. 324x324 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	0.2157
Gauss-Seidel	100	0.0633
SOR	100	3.2810e-004
SSOR	-	-
Gradient	100	0.2157
CG	28	8.8871e-007
PCG	20	7.7703e-007
BiCG	28	8.8871e-007

Tablo 4.24. 324x324 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

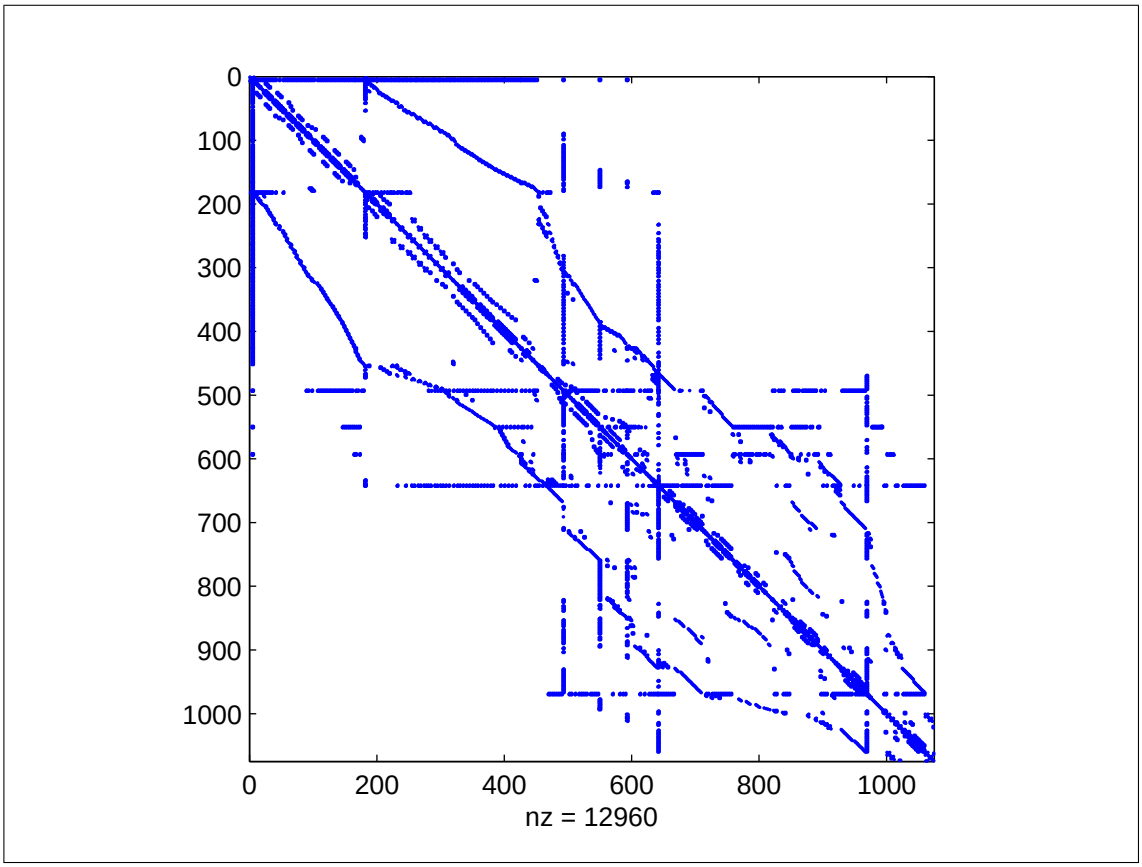
METOD	İTERASYON SAYISI	HATA
Jacobi	200	0.0546
Gauss-Seidel	200	0.0041
SOR	168	9.2236e-007
SSOR	-	-
Gradient	200	0.0531
CG	28	8.8871e-007
PCG	20	7.7703e-007
BiCG	28	8.8871e-007

Tablo 4.25. 324x324 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	400	0.0035
Gauss-Seidel	400	3.4442e-005
SOR	168	9.2236e-007
SSOR	-	-
Gradient	400	0.0032
CG	28	8.8871e-007
PCG	20	7.7703e-007
BiCG	28	8.8871e-007

Tablo 4.26. 324x324 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	600	2.2486e-004
Gauss-Seidel	600	3.4442e-005
SOR	168	9.2236e-007
SSOR	-	-
Gradient	600	1.9471e-004
CG	28	8.8871e-007
PCG	20	7.7703e-007
BiCG	28	8.8871e-007



Şekil 4.11. 1074x1074 boyutlu sparse matrisi

Tablo 4.27. 1074x1074 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	50	5.4125e+013
Gauss-Seidel	50	1.4258
SOR	50	2.8732
SSOR	-	-
Gradient	50	1.2396
CG	50	12.3325
PCG	50	7.5733e-004
BiCG	50	12.3325

Tablo 4.28. 1074x1074 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	8.4736e+026
Gauss-Seidel	100	1.3573
SOR	100	1.2465
SSOR	-	-
Gradient	100	1.2301
CG	100	5.8217
PCG	71	9.8406e-007
BiCG	100	5.8217

Tablo 4.29. 1074x1074 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata değişimi tablosu

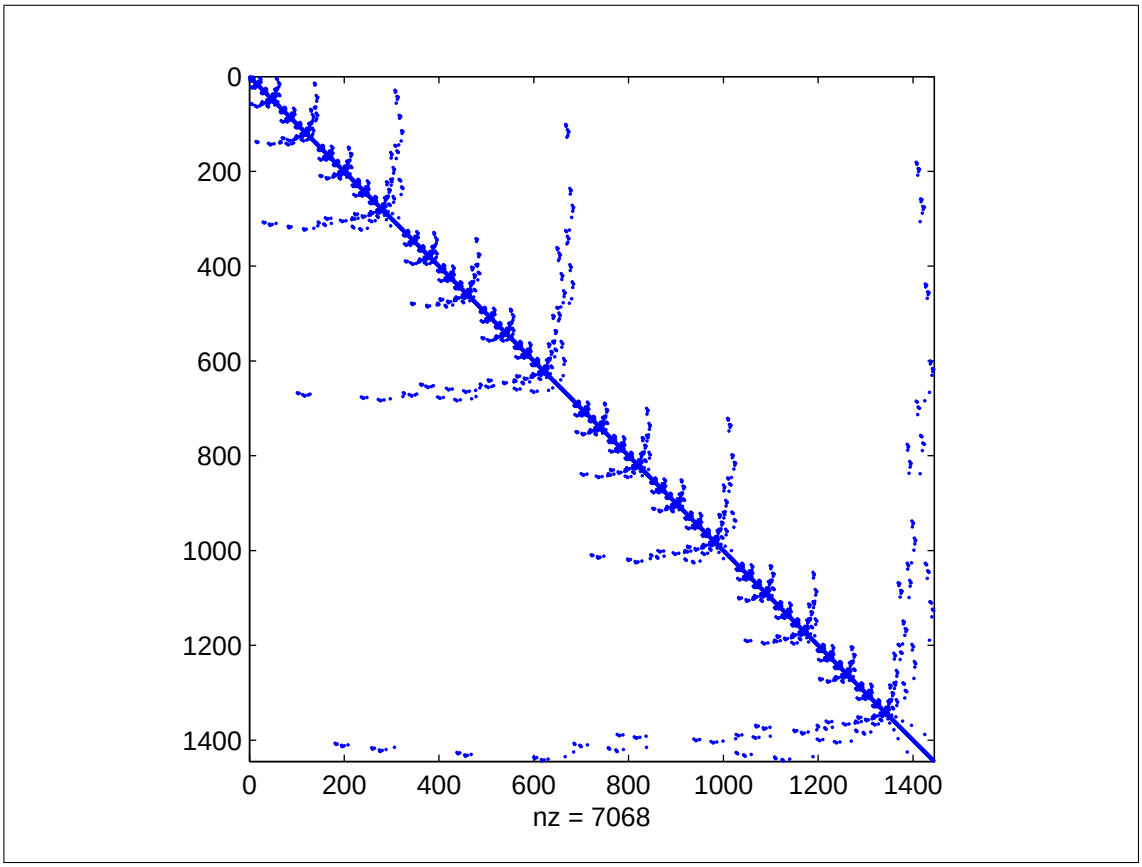
METOD	İTERASYON SAYISI	HATA
Jacobi	200	2.0769e+053
Gauss-Seidel	200	1.3496
SOR	200	1.1910
SSOR	-	-
Gradient	200	1.2237
CG	200	8.6720
PCG	71	9.8406e-007
BiCG	200	8.6720

Tablo 4.30. 1074x1074 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata değişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	400	1.2477e+106
Gauss-Seidel	400	1.3496
SOR	400	0.9563
SSOR	-	-
Gradient	400	1.2146
CG	400	5.1180
PCG	71	9.8406e-007
BiCG	400	5.1180

Tablo 4.31. 1074x1074 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	600	7.4951e+158
Gauss-Seidel	600	1.3496
SOR	600	0.7486
SSOR	-	-
Gradient	600	1.2072
CG	600	6.1515
PCG	71	9.8406e-007
BiCG	600	6.1515



Şekil 4.12. 1444x1444 boyutlu sparse matrisi

Tablo 4.32. 1444x1444 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	50	0.7281
Gauss-Seidel	50	0.6920
SOR	50	0.6113
SSOR	-	-
Gradient	50	0.8739
CG	50	9.4206e-005
PCG	38	7.2106e-007
BiCG	50	9.4206e-005

Tablo 4.33. 1444x1444 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	0.6054
Gauss-Seidel	100	0.4964
SOR	100	0.5371
SSOR	-	-
Gradient	100	0.7386
CG	60	8.9021e-007
PCG	38	7.2106e-007
BiCG	60	8.9021e-007

Tablo 4.34. 1444x1444 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata deĝişimi tablosu

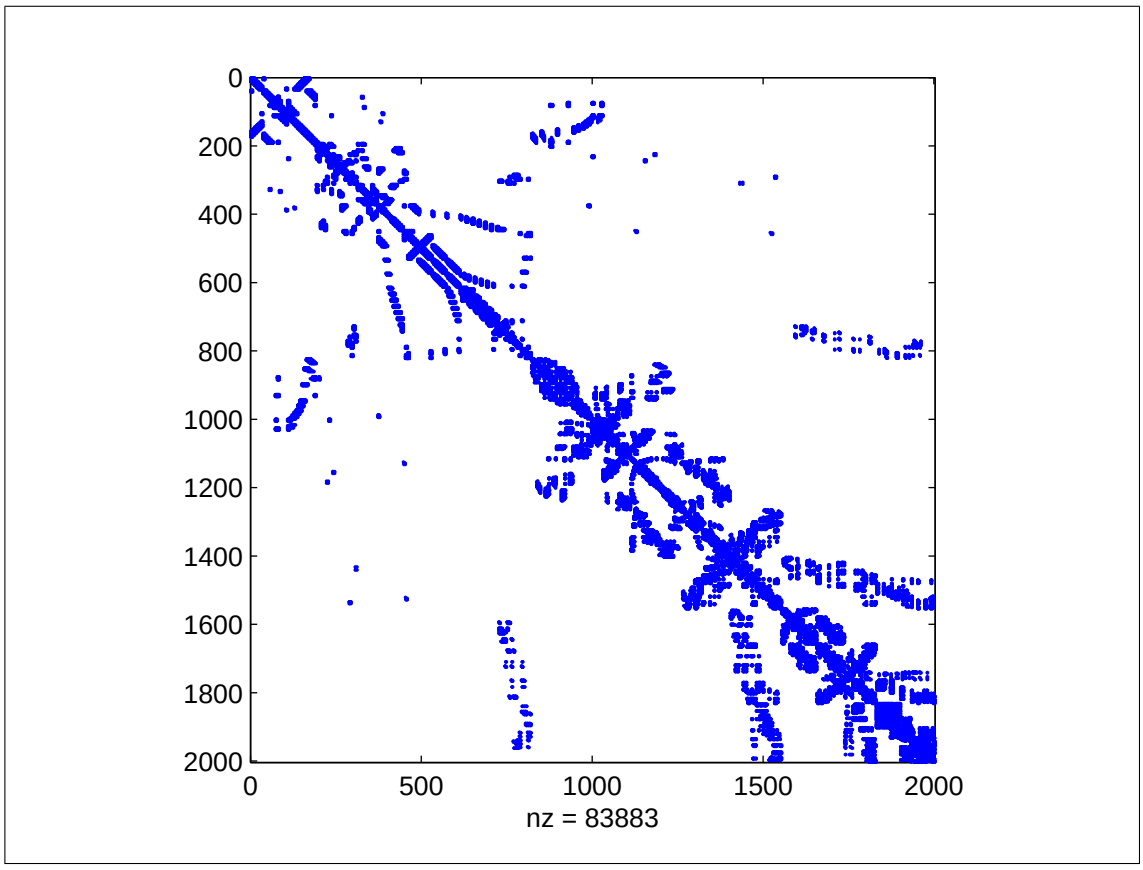
METOD	İTERASYON SAYISI	HATA
Jacobi	200	0.4343
Gauss-Seidel	200	0.2592
SOR	200	0.3153
SSOR	-	-
Gradient	200	0.5322
CG	60	8.9021e-007
PCG	38	7.2106e-007
BiCG	60	8.9021e-007

Tablo 4.35. 1444x1444 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
jacobi	400	0.2267
Gauss-Seidel	400	0.0707
SOR	400	0.1090
SSOR	-	-
Gradient	400	0.2770
CG	60	8.9021e-007
PCG	38	7.2106e-007
BiCG	60	8.9021e-007

Tablo 4.36. 1444x1444 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	600	0.1184
Gauss-Seidel	600	0.0193
SOR	600	0.0377
SSOR	-	-
Gradient	600	0.1442
CG	60	8.9021e-007
PCG	38	7.2106e-007
BiCG	60	8.9021e-007



Şekil 4.13. 2003x2003 boyutlu sparse matrisi

Tablo 4.37. 2003x2003 boyutlu sparse matrisin 50 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	50	4.7232e+025
Gauss-Seidel	50	6.9245
SOR	50	10.1942
SSOR	-	-
Gradient	50	1.1969
CG	50	23.3989
PCG	50	30.2423
BiCG	50	23.3989

Tablo 4.38. 2003x2003 boyutlu sparse matrisin 100 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	100	2.5756e+052
Gauss-Seidel	100	4.3183
SOR	100	8.3603
SSOR	-	-
Gradient	100	1.1830
CG	100	39.7353
PCG	100	26.5914
BiCG	100	39.7353

Tablo 4.39. 2003x2003 boyutlu sparse matrisin 200 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	200	7.6571e+105
Gauss-Seidel	200	3.4858
SOR	200	7.5365
SSOR	-	-
Gradient	200	1.1726
CG	200	139.5930
PCG	200	10.4287
BiCG	200	139.5930

Tablo 4.40. 2003x2003 boyutlu sparse matrisin 400 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	İTERASYON SAYISI	HATA
Jacobi	400	6.7671e+212
Gauss-Seidel	400	3.3571
SOR	400	6.6820
SSOR	-	-
Gradient	400	1.1584
CG	400	96.9058
PCG	400	0.9049
BiCG	400	96.9058

Tablo 4.41. 2003x2003 boyutlu sparse matrisin 600 iterasyon için metodlara göre hata deĝişimi tablosu

METOD	**İTERASYON SAYISI**	**HATA**
Jacobi	600	NaN
Gauss-Seidel	600	2.218e-004
SOR	600	6.0737
SSOR	-	-
Gradient	600	1.1481
CG	600	31.6617
PCG	600	8.7231e-007
BiCG	600	31.6617

5. SONUÇLAR VE TARTIŞMA

Jacobi Metod:

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* Metodun kullanımı kolaydır fakat matris kuvvetli dominant olmadıkça bu metod iterasyon metodlarına bir giriş veya duragan olmayan metodlardaki ön koşul olarak kullanılır.

Gauss-Seidel Metod:

* Bu metod Jacobi'den daha hızlı yakınsar ama duragan metodlarla rekabete giremez.

* Kesin olarak uygun diagonal dominant veya simetrik pozitif tanımlı matrislerde uygulanabilir.

* SOR metodun $w = 1$ seçilerek bulunabilen özel bir halidir.

Successive Overrelaxation Metod (SOR):

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* Sor metodu $w > 1$ için(over-relaxation) yakınsaklığını hızlandırır.Gauss-Seidel'in yakınsamadığı $0 < w < 1$ aralığında SOR yakınsar.

* Yakınsaklık hızı w 'ya bağlıdır. w 'nin en iyi değeri belirli şartlar altında Jacobi iterasyon matrisinin spektral yarıçapından tahmin edilebilir.

Symmetric Successive Overrelaxation Metod (SSOR):

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* SSOR metodu $w > 1$ için(over-relaxation) yakınsaklığını hızlandırır. Gauss-Seidel'in yakınsaklığını $0 < w < 1$ aralığında SSOR yakınsar.

* Yakınsaklık hızı w 'ya bağlıdır. w 'nin en iyi değeri belirli şartlar altında Jacobi iterasyon matrisinin spektral yarıçapından tahmin edilebilir.

* SSOR iyi sonuçlar vermesine rağmen küçük boyutlu matrislerin çözümünde bile çok fazla iterasyon sayısı gerektirmektedir.

Gradient Metod:

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* Yakınsama için fazla iterasyon gerektirir.

Conjugate Gradient Metod(CG):

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* Yakınsamanın hızı koşul sayısına bağlıdır. (Koşul sayısı $K(A) = \|A\| \, . \, \|A^{-1}\|$)

* $n \times n$ boyutlu bir matriste en fazla n iterasyon gerektirir.

Preconditioned Conjugate Gradient Metod (PCG):

* Simetrik ve pozitif tanımlı matrislerde uygulanabilir.

* Conjugate Gradient Metod'un bir preconditioner (ön koşul matrisi) ile birleştirilerek genişletilmiş halidir.

* Yakınsama için diĝer metodlara göre daha az sayıda iterasyon gerektirir.

* Özellikle büyük boyutlu sistemlerde daha iyi sonuç verir.

BiConjugate Gradient Metod (BiCG):

* Simetrik olmayan matrislerde de uygulanabilir.

* Genellikle az sayıda iterasyonla iyi sonuçlar verir. Büyük boyutlu sistemlerde fazla iterasyon gerektirir.

KAYNAKLAR

Burden,R.L.2001.Numerical Analysis , Broks/Cole , Pacific Grove -CA , 841p

Barret, R. Berry, M. Chan, T.F. Demmel, J. Donato, J.M. Dongarra, J. Eijkhout, V. Pozo, R. Romine, C. Vorst, H.V. 1994.Templates For The Solution of Linear Systems:Building Blocks For Iterative Methods, Siam , Philadelphia , 124p

Demmel, J.W.1997. Applied Numerical Linear Algebra , Siam , Philadelphia ,419p

Golub, G.H. Van Loan, C.F.1993.Matrix Computations , John Hopkins , Baltimore-MD , 642p

Saad,Y. 2003. Iterative Methods For Sparse Linear Systems, Siam , Philadelphia , 528p

Quarteroni, A. Sacco,R. Saleri, F.2000. Numerical Mathematics , Springer , Newyork , 654p

Watkins, D.S. 2002. Fundamentals of Matrix Computations , Wiley , Newyork ,618p

Kincaid,D. Cheney,W. 1996. Numerical Analysis , Broks/Cole,Pacific Grove -CA , 804p

Trefethen, L.N. Bau, D. 1997. Numerical Linear Algebra , Siam , Philadelphia , 361p

Manteuffel,T. Faber V.1984. Neccesary and Sufficient Conditions For The Existence Of A Conjugate Gradient Method , Siam J Numer. Anal. , pp. 315-339

Freund, R. 1992. Conjugate Gradient-Type Methods For The Linear Systems With Complex Symmetric Coefficient Matrices , Siam J Sci. Statist. Comput. , pp.425-448

Hageman, L. Young, D.1981. Applied Iterative Methods , Academic Press ,Newyork

Hestenes, M. Stiefel, E.1952.Methods Of Conjugate Gradients For Solving Linear Systems , J. Res. Nat. Bur. Stand. , pp.409-436

Paige, C. Saunders, M.1975.Solution Of Sparse Indefinite Systems Of Linear Equations , Siam J. Numer. Anal. , pp.617-629

Reid, J. 1971.On The Method Of Conjugate Gradients For The Solution Of Large Sparse Systems Of Linear Equqtions , Academic Press , London ,pp.231-254

Vorst, V.D.H.1981. Iterative Solution Methods For Certain Sparse Lineer Systems With A Non-Symmetric Matrix Arising From PDE-Problems , J. Comput. Phys. , pp.45-54

Varga, R. 1962. Matrix Iterative Analysis , Prentice-Hall Inc. , Englewood Cliffs

Young, D. 1971. Iterative Solution Of Large Linear Systems , Academic Press , Newyork

Jacobi için Matlab Programı

```
function [X,max1,relerr]=jacobi(A,b,P,delta, max1)

% Input  - A is an N x N nonsingular matrix
%        - b is an N x 1 matrix
%        - P is an N x 1 matrix; the initial guess
%        - delta is the tolerance for P
%        - max1 is the maximum number of iterations
% Output - X is an N x 1 matrix: the jacobi approximation to
%           the solution of AX = b

N = length(b);

for k=1:max1
   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
   err=abs(norm(X'-P));
   relerr=err/(norm(X)+eps);
   P=X';
      if (err<delta)|(relerr<delta)
     break
   end
end

X=X';
```

Gauss-Seidel için Matlab Programı

```
function [X,max1,relerr]=gauss(A,b,P,delta, max1)

% Input  - A is an N x N nonsingular matrix
%        - b is an N x 1 matrix
%        - P is an N x 1 matrix; the initial guess
%        - delta is the tolerance for P
%        - max1 is the maximum number of iterations
% Output - X is an N x 1 matrix: the gauss-seidel approximation to
%      the solution of AX = b
tic N = length(b);

for k=1:max1
   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 contains the kth approximations and P the (k-1)st
        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
   end
   err=abs(norm(X'-P));
   relerr=err/(norm(X)+eps);
   P=X';
      if (err<delta)|(relerr<delta)
     break
   end
end X=X';
```

SOR için Matlab Programı

```
function [x,iter,err]=sor(A,b,x0,nmax,tol,omega)

% Input  - A is an N x N nonsingular matrix
%        - b is an N x 1 matrix
%        - x0 is an N x 1 matrix; the initial guess
%        - tol is the tolerance for P
%        - nmax is the maximum number of iterations
%        - omega 0 ile 2 arasında bir değer
% Output - x is an N x 1 matrix: the gauss-seidel approximation to
%      the solution of Ax = b

[n,n]=size(A); iter = 0; r = b - A * x0; r0 = norm (r); err = norm
(r); xold = x0; while err > tol & iter < nmax
 iter = iter + 1;
 for i=1:n
   s = 0;
   for j = 1:i-1
      s = s + A(i,j) * x(j);
   end
   for j = i+1:n
     s = s + A(i,j) * xold (j);
   end
   x (i) = omega * ( b(i) - s) / A(i,i) + (1 - omega) * xold (i);
 end
 x = x(:); xold = x;
 r = b - A*x;
 err = norm (r) / r0;
end err=norm(b-A*x)/norm(b) x1=A\b; hata=norm(x-x1)/norm(x1) return
```

SSOR için Matlab Programı

```
function [x,iter] =  ssor(A,b,tol,w)

% Input  - A is an N x N nonsingular matrix
%        - b is an N x 1 matrix
%        - tol is the tolerance for P
%        - w 0 ile 2 aras?nda bir de?er
% Output - X is an N x 1 matrix: the gauss-seidel approximation to
%      the solution of AX = b

n = size(A,1);

    % Choose an initial guess x^(0) to the solution x.
for i=1:n
    x(i,1) = b(i,1)/A(i,i);
end

% Let x^(1/2) = x^(0).
% set values.
iter=0; while (check(A,x,b)>tol)
    for i=1:n
        sig=0;
         for j=1:i-1
              % sig = sig + aij xj^(k-1/2)
             sig=sig+A(i,j)*x(j,1);
         end
         for j=i+1:n
             % sig = sig + aij xj^(k-1)
             sig=sig+A(i,j)*x(j,1);
         end
         % sig = (bi - sig)/aii
          sig=(b(i,1)-sig)/A(i,i);
           % xi^(k-1/2) = xi^(k-1) + w (sig - xi^(k-1))
           x(i,1)=x(i,1)+w*(sig-x(i,1));
        end
        for i=n:-1:1
```

```
            sig=0;
            for j=1:i-1
                % sig = sig + aij xj^(k-1/2)
                sig=sig+A(i,j)*x(j,1);
            end
            for j=i+1:n
                % sig = sig + aij xj^(k)
                sig=sig+A(i,j)*x(j,1);
            end
            % sig = (bi - sig)/aii
            sig=(b(i,1)-sig)/A(i,i);
            % xi^(k) = xi^(k-1/2) + w (sig - xi^(k-1/2))
            x(i,1)=x(i,1)+w*(sig-x(i,1));
        end
        iter=iter+1;
        % check convergence; continue if necessary.
end
```

Gradient için Matlab Programı

```
function [x, error, niter, flag] = gradient(A, x, b, M, maxit, tol)

% This code solves Gradients method with dynamic parameter.
%
%    Input parameters:
%           A : Symmetric, positive definite NxN matrix
%           b : Right-hand side Nx1 column vector
%           x : Nx1 start vector (the initial guess)
%           M : Dynamic parameter (preconditioner)
%         tol : relative residual error tolerance for break
%               condition
%       maxit : Maximum number of iterations to perform
%
%    Output parameters:
%           x : Nx1 solution vector
%       niter : Number of iterations performed
%        flag : 1 if convergence criteria specified by TOL could
%               not be fulfilled within the specified maximum
%               number of iterations, 0 otherwise (= iteration
%               successful).

flag = 0;    niter = 0;   bnrm2 = norm( b ); if  ( bnrm2 == 0.0 ),
bnrm2 = 1.0; end r = b - A*x;  error = norm( r ) / bnrm2; if ( error
< tol ) return, end for niter = 1:maxit
    z  = M \ r;  rho = (r'*z);
    q = A*z;               alpha = rho / (z'*q );
    x = x + alpha * z;     r = r - alpha*q;
    error = norm( r ) / bnrm2;
    if ( error <= tol ), break, end
end if ( error > tol ) flag = 1; end return
```

CG için Matlab Programı

```
function [u, niter,flag] = solveCG(A, b, x0, tol, maxiter)
% This code solves  Conjugate Gradients method.
%
%    Input parameters:
%           A : Symmetric, positive definite NxN matrix
%           b : Right-hand side Nx1 column vector
%           x0 : Nx1 start vector (the initial guess)
%         tol : relative residual error tolerance for break
%               condition
%     maxiter : Maximum number of iterations to perform
%
%    Output parameters:
%           u : Nx1 solution vector
%       niter : Number of iterations performed
%        flag : 1 if convergence criteria specified by TOL could
%               not be fulfilled within the specified maximum
%               number of iterations, 0 otherwise (= iteration
%               successful).

u = x0;         % Set u_0 to the start vector s
r = b - A*x0;   % Compute first residuum
p = r; rho = r'*r;
niter = 0;     % Init counter for number of iterations
flag = 0;      % Init break flag

% Compute norm of right-hand side to take relative residuum as
% break condition.
normb = norm(b);
if normb < eps  % if the norm is very close to zero, take the
                % absolute residuum instead as break condition
                % ( norm(r) > tol ), since the relative
                % residuum will not work (division by zero).
  warning(['norm(b) is very close to zero, taking absolute residuum' ...
                    ' as break condition.']);
    normb = 1;
```

```
end

while (norm(r)/normb > tol)   % Test break condition
    a = A*p;
    alpha = rho/(a'*p);
    u = u + alpha*p;
    r = r - alpha*a;
    rho_new = r'*r;
    p = r + rho_new/rho * p;
    rho = rho_new;
    niter = niter + 1;
    %error = norm( r ) / bnrm2;
    if (niter == maxiter)         % if max. number of iterations
        flag = 1;                   % is reached, break.
        break
    end
end
```

PCG için Matlab Programı

```
function [u,m] = solvePCG(A, b, x0, C1, C2, tol, m_max)
% SOLVEPCG   Preconditioned Conjugate Gradients method for solving
%    a system of linear equations Au = b.
%
%    Input parameters:
%       A    : symmentric, positive definite NxN matrix
%       b    : right-hand side Nx1 column vector
%       x0   : Nx1 start vector (initial guess)
%       C    : Preconditioner, split into C1 and C2. Splitting C
%              in C1 und C2 offers computational advantages.
%                Example: Symmetric Gauss-Seidel preconditioner:
%                C = (D+L)*inv(D)*(D+L')
%                D = diag(diag(A))
%                C1 = tril(A)
%                C2 = inv(D)*triu(A)
%       tol  : Break condition for the relative residuum
%       m_max: Maximum number of iterations to perform
%
%    Output parameters:
%       m    : Number of actually performed iterations
%       u    : Solution vector


u = x0; r = b - A * u; p = C2 \ (C1 \ r); norm_b = norm(b);

m = 0; while( (norm(r)/norm_b > tol) & (m < m_max))
  a = A * p;
  a_dot_p = a' * p;
  lambda = (r' * p) / a_dot_p;
  u = u + lambda * p;
  r = r - lambda * a;
  inv_C_times_r = C2 \ (C1 \ r);
  p = inv_C_times_r - ((inv_C_times_r' * a) / a_dot_p) * p;
  m=m+1;
end
```

BiCG için Matlab Programı

```
function [x, error, iter, flag] = bicg(A, x, b, M, max_it, tol)

%  -- Iterative template routine --
%  [x, error, iter, flag] = bicg(A, x, b, M, max_it, tol)
%
% bicg.m solves the linear system Ax=b using the
% BiConjugate Gradient Method with preconditioning.
%
% input   A        REAL matrix
%         M        REAL preconditioner matrix
%         x        REAL initial guess vector
%         b        REAL right hand side vector
%         max_it   INTEGER maximum number of iterations
%         tol      REAL error tolerance
%
% output  x        REAL solution vector
%         error    REAL error norm
%         iter     INTEGER number of iterations performed
%         flag     INTEGER: 0 = solution found to tolerance
%                           1 = no convergence given max_it
%                          -1 = breakdown

  iter = 0;                                % initialization
  flag = 0;

  bnrm2 = norm( b );
  if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end

  r = b - A*x;
  error = norm( r ) / bnrm2;
  if ( error < tol ) return, end

  r_tld  = r;

  for iter = 1:max_it                      % begin iteration
```

```
   z = M \ r;
   z_tld = M' \ r_tld;
   rho   = ( z'*r_tld );
   if ( rho == 0.0 ),
      break
   end

   if ( iter > 1 ),                    % direction vectors
      beta = rho / rho_1;
      p    = z  + beta*p;
      p_tld = z_tld + beta*p_tld;
   else
      p = z;
      p_tld = z_tld;
   end

   q = A*p;                            % compute residual pair
   q_tld = A'*p_tld;
   alpha = rho / (p_tld'*q );

   x = x + alpha*p;                    % update approximation
   r = r - alpha*q;
   r_tld = r_tld - alpha*q_tld;

   error = norm( r ) / bnrm2;          % check convergence
   if ( error <= tol ), break, end

   rho_1 = rho;

end

if ( error <= tol ),                   % converged
   flag =  0;
elseif ( rho == 0.0 ),                 % breakdown
   flag = -1;
```

```
else
   flag = 1;                              % no convergence
end
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı soyadı :Sibel Demiryakan SARI

Doĝum Yeri :Konya

Doĝum Tarihi :1981

Medeni Hali :Evli

EĜİTİM ve AKADEMİK BİLGİLER

Lise :Konya Lisesi

Lisans :Selçuk Üniversitesi Eĝitim Fakültesi İlköĝretim Matematik Öĝretmenliĝi

Yabancı Dil :İngilizce

MESLEKİ BİLGİLER

Milli Eĝitim Bakalığına Baĝlı Okullarda Öĝretmenlik

Konya (2002-2003)

Muĝla (2003-)