

Iterative Techniques for Solving Linear Systems

An iterative technique to solve an $n \times n$ linear system $Ax = b$ starts with an initial approximation x_0 to the solution x^* , and generates a sequence of vectors $\{x^{(k)}\}_{k=0}^{\infty}$ that converges to x^* . Most of these iterative techniques involve a process that converts the system $Ax = b$ into an equivalent system of the form $x = Tx + c$ for some $n \times n$ matrix T and vector c . After the initial vector x_0 is selected, the sequence of approximate solution vectors is generated by computing

$$\mathbf{x}^{(k+1)} = \mathbf{T}\mathbf{x}^{(k)} + \mathbf{c}, \quad \mathbf{k} = 1, 2, \dots$$

where contrary to the Gaussian elimination method, neither T nor c depends upon the iteration count k . Clearly iterative methods are fast and efficient. The four main methods are the Jacobi, Gauss-Seidel, successive over-relaxation (SOR), and symmetric successive over-relaxation (SSOR) methods.

Iterative techniques are seldom used for solving linear systems of small dimension since the time required for sufficient accuracy exceeds that required for direct technique such as the Gaussian elimination method. For large systems with a high percentage of zero entries, however, these techniques are efficient in terms of computer storage and time requirements. Systems of this type arise frequently in the numerical solution of boundary-value problems and partial-differential equations.

In all that follows, we assume that $A = D + (L + U)$, where the matrices D , L , and U represent the diagonal, strictly lower triangular, and strictly upper triangular parts of A , respectively.

The Jacobi method is based on solving for every variable locally with respect to the other variables; one iteration corresponds to solving for every variable once. It is easy to understand and implement, but convergence is slow.

The Gauss-Seidel method is similar to the Jacobi method except that it uses updated values as soon as they are available. It generally converges faster than the Jacobi method, although still relatively slowly.

The successive over-relaxation method can be derived from the Gauss-Seidel method by introducing an extrapolation parameter ω . This method can converge faster than Gauss-Seidel by an order of magnitude.

Finally, the symmetric successive over-relaxation method is useful as a pre-conditioner for non-stationary methods. However, it has no advantage over the successive over-relaxation method as a stand-alone iterative method.

Neumann Lemma. *If A is an $n \times n$ matrix with $\rho(A) < 1$, then $(I_n - A)^{-1}$ exists and*

$$(I_n - A)^{-1} = I_n + A + A^2 + \cdots + A^k + \cdots$$

Proof. If $AU = \lambda u$, then $A^k U = \lambda^k u$. Since $\rho(A) < 1$, it follows that

$$\lim_{j \rightarrow \infty} \rho(A^j) = 0 \quad \text{thus} \quad \lim_{k \rightarrow \infty} (A^k) = Z_n ,$$

where Z_n is the $n \times n$ zero matrix. Since $\lim_{k \rightarrow \infty} A^k = Z_n$, the matrix series

$$I_n + A + A^2 + \cdots + A^k + \cdots$$

is convergent. Now by multiplying the matrix $(I_n - A)$ by this series, we obtain

$$(I_n - A)(I_n + A + A^2 + \cdots + A^k + \cdots) = I_n .$$

Thus

$$(I_n - A)^{-1} = \sum_{k=0}^{\infty} A^k = I_n . \quad \blacksquare$$

Perturbation Lemma. *Let A and B be $n \times n$ matrices and assume that A is invertible with $\|A^{-1}\| \leq \alpha$. If $\|A - B\| \leq \beta$ and $\alpha\beta \leq 1$, then B is also invertible, and*

$$\|B^{-1}\| \leq \alpha/(1 - \alpha\beta).$$

Proof. Since

$$\|I_n - A^{-1}B\| = \|A^{-1}(A - B)\| \leq \alpha\beta < 1$$

and

$$A^{-1}B = I_n - (I_n - A^{-1}B),$$

by Neumann Lemma, $A^{-1}B$ is invertible. Hence, B is invertible. Moreover, we conclude that

$$\|B^{-1}\| = \|[I_n - (I_n - A^{-1}B)]^{-1}A^{-1}\| \leq \alpha \sum_{i=0}^{\infty} (\alpha\beta)^i = \alpha/(1 - \alpha\beta). \quad \blacksquare$$

Note. It is important to note that if A is a matrix with $\rho(A) = 0$, then A is not necessarily zero. Consider for example the nonzero matrix $A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ with $\rho(A) = 0$. Although $A \neq Z_2$ but $A^2 = Z_2$. Next we establish a result concerning matrices with zero eigenvalues.

Lemma 1. Let A be an $n \times n$ matrix with zero eigenvalues, then a power of A must be zero.

Proof. According to Schur Triangularization theorem, A is unitarily similar to a triangular matrix T with zeros on the main diagonal. A direct computation shows that for some $k \leq n$, $T^k = Z_n$; hence $A^k = Z_n$. ■

Theorem 1. For any $x^{(0)} \in \mathbb{R}^n$, the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ defined by

$$x^{(k+1)} = Tx^{(k)} + c \quad (k = 1, 2, \dots) \quad c \neq 0,$$

converges to the unique solution of $x = Tx + c$ if and only if $\rho(T) < 1$.

Proof. Note that the system $x = Tx + c$ is equivalent to $(I_n - T)x = c$, and

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

If $\rho(T) < 1$, then

$$\lim_{k \rightarrow \infty} T^kx^{(0)} = Z_nx^{(0)} = \theta \quad \text{and} \quad \lim_{k \rightarrow \infty} (I_n + T + T^2 + \dots + T^{k-1}) = (I_n - T)^{-1}.$$

Thus

$$x^* = \lim_{k \rightarrow \infty} x^{(k)} = (I_n - T)^{-1}c$$

will be the unique solution to $x = Tx + c$.

Conversely, if $\{x^{(k)}\}_{k=0}^{\infty}$ converges to x for any $x^{(0)}$, then by (1),

$$\lim_{k \rightarrow \infty} T^kx^{(0)} = \theta \quad \text{for any } x^{(0)}.$$

This clearly implies that all the eigenvalues of T must be inside of the unit circle. Thus $\rho(T) < 1$. ■

Lemma 2. If T is an $n \times n$, then $\rho(T) \leq \|T\|$ for any natural matrix norm $\|\cdot\|$.

Proof. Let u be a unit eigenvector of T corresponding to the dominant eigenvalue λ (i.e., $\|u\| = 1$ and $Tu = \lambda u$ with $|\lambda| = \rho(T)$). We have

$$\rho(T) = |\lambda| \cdot 1 = |\lambda| \|u\| = \|\lambda u\| = \|Tu\| \leq \max\{Tx : \|x\| = 1\} = \|T\|. \quad \blacksquare$$

The following result is an immediate consequence of Theorem 1 and Lemma 2.

Corollary. *If $\|T\| < 1$, then for any $x_0 \in \mathbb{R}^n$, the sequence $\{x^{(k)}\}_{k=0}^{\infty}$ defined by*

$$x^{(k+1)} = Tx^{(k)} + c \quad (k = 1, 2, \dots) \quad c \neq 0,$$

converges to the unique solution of $x = Tx + c$.

♣ **Jacobi Method.** The Jacobi method is a method of solving a matrix equation on a matrix that has no zeros along its main diagonal. Each diagonal element is solved for, and an approximate value plugged in. The process is then iterated until it converges. The Jacobi method is easily derived by examining each of the n equations in the linear system of equations $Ax = b$ in isolation. If, in the i -th equation

$$\sum_{j=1}^n a_{ij} x_j = b_i, \quad (1)$$

solve for the value of x_i while assuming the other entries of x remain fixed. This gives

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

which is the Jacobi method. Note that the computation of $x_i^{(k+1)}$ requires each component in $x^{(k)}$, except $x_i^{(k)}$. In this method, the order in which the equations are examined is irrelevant, since the Jacobi method treats them independently.

The equation $Ax = b$, which is $(L + D + U)x = b$, can be written as

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

This reduces to

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

Thus $T_j = -D^{-1}(L + U)x$ and $c_j = D^{-1}b$.

Note that if A is a strictly diagonally dominant matrix, then

$$\|T_j\|_{\infty} = \|-D^{-1}(L + U)\|_{\infty} < \|-D^{-1}\|_{\infty} \|(L + U)\|_{\infty} = \|D\|_{\infty}^{-1} \|(L + U)\|_{\infty} = \frac{\|(L + U)\|_{\infty}}{|\max a_{ii}|} < 1.$$

Hence for any $x_0 \in \mathbb{R}^n$, the sequence $\{x_k\}_{k=0}^{\infty}$ defined by

$$x_k = T_j x_{k-1} + c_j \quad (k = 1, 2, \dots) \quad c_j \neq 0,$$

converges to the unique solution of $x = T_j x + c_j$. Therefore Jacobi's iterative technique will always guarantee a solution to the equation $Ax = b$ whenever A is strictly diagonally dominant.

Algorithm. The algorithm for Jacobi method:

Enter a matrix A ;
choose an initial guess x_0 , a maximum number of iterations N , and a tolerance Tol ;
find D , representing the diagonal, L , strictly lower triangular, and U , strictly upper triangular parts of A ;
set $T = -D^{-1}(L + U)$;
set $c = D^{-1}b$;
set $k = 0$;
while $k \leq N$;
 set $x^{(k+1)} = T x^{(k)} + c$;
 if $\|x^{(k+1)} - x^{(k)}\|_\infty \leq Tol$, then $k = N + 1$;
end;
Display $x^{(k+1)}$. ■

MATLAB PROGRAM

```
A = input('Enter the diagonally dominant matrix A : ');
b = input('Enter the constant vector b as a row vector : ');
x0 = input('Enter the initial vector as a row vector x0 : ');
x = x0';
Tol = input('Enter the tolerance : ');
max = input('Enter the maximum number of iterations : ');
disp(' '), disp(' '),
n = length(b);
D = zeros(n); T = zeros(n); c = zeros(n, 1);
for i = 1 : n, D(i, i) = A(i, i); end
T = A - L;
T = -inv(D) * T; c = inv(D) * b';
format short
P = zeros(max, n);
k = 1;
while k < max + 1,
    y = T * x + c;
    dx = y - x;
    x = y; P(k, 1 : n) = x';
    if norm(dx) < Tol, last = k; k = max; end
    k = k + 1;
end
```

```

clc, disp(' '), disp(' '),
disp('The solution to the system [A b] : '), disp(' '), disp([A b'])
disp('by using Jacobi's iterative method with the initial vector x0 = ')
disp(' '), disp(x0)
disp(['and the Tolerance = ', num2str(Tol), ' is :'])
disp(' '), disp(x')
disp('The convergence to the solution is linear, and here is a list of x''s ')
disp(' '), disp(P(1 : last, 1 : n)), disp('')
disp('The accuracy is dx' = ')
disp(dx')
disp(['To obtain the solution with a tolerance = ', num2str(Tol), ','])
disp(['int2str(max), ' iterations was needed. '])

```

Example 1. Use the Jacobi method to find a solution to the linear system defined by:

$$\begin{pmatrix} 5 & 2 & -1 \\ 3 & 7 & 3 \\ 1 & -4 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}.$$

We rewrite this system as:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1/5 & 0 & 0 \\ 0 & 1/7 & 0 \\ 0 & 0 & 1/6 \end{pmatrix} \begin{pmatrix} 0 & 2 & -1 \\ 3 & 0 & 3 \\ 1 & -4 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1/5 & 0 & 0 \\ 0 & 1/7 & 0 \\ 0 & 0 & 1/6 \end{pmatrix} \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}.$$

Thus, if we start with a random vector, say

$$x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ and iterate until } \|x^{(k+1)} - x^{(k)}\|_2 < \epsilon = 0.1.$$

$$x^{(1)} = \begin{pmatrix} 0.4 \\ -0.14286 \\ 0.16667 \end{pmatrix} \quad \|x^{(1)} - x^{(0)}\|_2 = 0.45627,$$

$$x^{(2)} = \begin{pmatrix} 0.49048 \\ -0.38571 \\ 0.00476 \end{pmatrix} \quad \|x^{(2)} - x^{(1)}\|_2 = 0.30558,$$

$$x^{(3)} = \begin{pmatrix} 0.55524 \\ -0.35510 \\ -0.17222 \end{pmatrix} \quad \|x^{(3)} - x^{(2)}\|_2 = 0.19093,$$

$$x^{(4)} = \begin{pmatrix} 0.50760 \\ -0.30701 \\ -0.16261 \end{pmatrix} \quad \|x^{(4)} - x^{(3)}\|_2 = 0.068376 < \epsilon.$$

Thus, after 4 iterations, we have converged. It takes 8 iterations to reduce the step size to less than $\epsilon = 0.01$ and 11 iterations to reduce the step size to less than $\epsilon = 0.001$.

For such a small example, it would be cheaper to use Gaussian elimination and backward substitution, however, the number of multiplications and divisions grows $O(n^3)$ whereas the Jacobi method only

requires one matrix-vector multiplication and is therefore $O(n^2)$. Note that with the Gaussian elimination and backward substitution, we have no restriction on the matrix A and also, the solution in general is exact and not an approximation.

♣ **Gauss-Seidel Method Iterative Method for Linear System.** The method is an improved version of the Jacobi method. It is defined on matrices with non-zero diagonals, but convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and (semi) positive definite.

The equation $Ax = b$, which is $(L + D + U)x = b$, can be written as $(D + L)x = -Ux + b$. This reduces to

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

Thus

$$x^{(k+1)} = -(D + L)^{-1}Ux^{(k)} + (D + L)^{-1}b = \mathbf{T}_G \mathbf{x}^{(k)} + \mathbf{c}_G.$$

This matrix expression is mainly used to analyze the method. When implementing Gauss-Seidel, an explicit entry-by-entry approach is used:

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

Note that the computation of $x_i^{(k+1)}$ uses only those elements of $x^{(k+1)}$ that have already been computed ($x_j^{(k+1)}$ for $j < i$) and only those elements of $x^{(k)}$ that have yet to be advanced to iteration $k + 1$ ($x_j^{(k)}$ for $j \geq i$). This means that no additional storage is required, and the computation can be done in place ($x^{(k+1)}$ replaces $x^{(k)}$). While this might seem like a rather minor concern, for large systems it is unlikely that every iteration can be stored. Thus, unlike the Jacobi method, one does not have to do any vector copying should one want to use only one storage vector. The iteration is generally continued until the changes made by an iteration are below some tolerance.

Theorem 2. *If A is nonsingular, then for any eigenvalue λ of A*

$$\frac{1}{\|A^{-1}\|} \leq |\lambda| \leq \|A\|,$$

where $\|\cdot\|$ is any natural matrix norm.

Proof. To prove the theorem, we use the facts that $\rho(A) \leq \|A\|$ and if λ is an eigenvalue of A , then $1/\lambda$ is an eigenvalue of A^{-1} . ■

If A is a strictly diagonally dominant matrix, then according to the above theorem

$$\|-(D + L)^{-1}U\| \leq \|(D + L)^{-1}\| \|U\| < 1.$$

Thus for any $x_0 \in \mathbb{R}^n$, the sequence $\{x_k\}_{k=0}^{\infty}$ defined by

$$x_k = T_G x_{k-1} + c_G \quad (k = 1, 2, \dots) \quad c_g \neq 0,$$

converges to the unique solution of $x = T_G x + c_G$, i.e., the Gauss-Seidel's iterative technique solves the equation $Ax = b$.

Unlike in the Gauss-Seidel method, in Jacobi method, we can't overwrite $x_i^{(k)}$ with $x_i^{(k+1)}$, as that value will be needed by the rest of the computation. This is the most meaningful difference between the Jacobi and GaussSeidel methods.

Algorithm. The algorithm for Gauss-Siedel method:

```

Enter a matrix A;
choose an initial guess  $x_0$ , a maximum number of iterations  $N$ , and a tolerance  $Tol$ ;
find  $D$ , representing the diagonal,  $L$ , strictly lower triangular, and  $U$ , strictly upper triangular parts
of  $A$ ;
find the inverse of the lower triangular matrix  $(D + L)$ ;
set  $T = -(D + L)^{-1}U$ ;
set  $c = (D + L)^{-1}b$ ;
set  $k = 0$ ;
while  $k \leq N$ ;
    set  $x^{(k+1)} = T x^{(k)} + c$ ;
    if  $\|x^{(k+1)} - x^{(k)}\|_\infty \leq Tol$ , then  $k = N + 1$ ;
end;
Display  $x^{(k+1)}$ .

```

MATLAB PROGRAM

```

A = input('Enter the diagonally dominant matrix A : ');
b = input('Enter the constant vector b as a row vector : ');
x0 = input('Enter the initial vector as a row vector x0 : ');
x = x0';
Tol = input('Enter the tolerance : ');
max = input('Enter the maximum number of iterations : ');
disp(' '), disp(' ');
n = length(b);
D = zeros(n); L = zeros(n); U = zeros(n); T = zeros(n); c = zeros(n, 1);
for i = 1 : n, D(i, i) = A(i, i); end
for i = 2 : n, for j = 1 : i - 1, L(i, j) = -A(i, j); end, end
for i = 1 : n - 1, for j = i + 1 : n, U(i, j) = -A(i, j); end, end
T = inv(D - U) * L; c = inv(D - U) * b';
format short
P = zeros(max, n);

```

```

k = 1;
while k < max + 1,
    y = T * x + c;
    dx = y - x;
    x = y; P(k, 1 : n) = x';
    if norm(dx) < Tol, last = k; k = max; end
    k = k + 1;
end
clc, disp(' '), disp(' '),
disp('The solution to the system [A b] : '), disp(' '), disp([A b])
disp('by using Gauss - Seidel's iterative method with the initial vector x0 = ')
disp(' '), disp(x0)
disp(['and the Tolerance = ', num2str(Tol), ' is :'])
disp(' '), disp(x')
disp('The convergence to the solution is linear, and here is a list of x''s ')
disp(' '), disp(P(1 : last, 1 : n)), disp(' ')
disp('The accuracy is dx' = ')
disp(dx')
disp(['To obtain the solution with a tolerance = ', num2str(Tol), ', '])
disp(['int2str(max), ' iterations was needed.'])

```

Example 2. Use the Gauss-Seidel method to find a solution to the linear system:

$$\begin{cases} E_1 : 4x_1 + x_2 - x_3 = 3 \\ E_2 : 2x_1 + 7x_2 + x_3 = 19 \\ E_3 : x_1 - 3x_2 + 12x_3 = 31 \end{cases} \implies \begin{pmatrix} 4 & 1 & -1 \\ 2 & 7 & 1 \\ 1 & -3 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 19 \\ 31 \end{pmatrix}$$

First we need to find

$$(D - L)^{-1} = \begin{pmatrix} 1/4 & 0 & 0 \\ -1/14 & 1/7 & 0 \\ -13/336 & 1/28 & 1/12 \end{pmatrix} = \begin{pmatrix} 0.2500 & 0 & 0 \\ -0.0714 & 0.1429 & 0 \\ -0.0387 & 0.0357 & 0.0833 \end{pmatrix}.$$

If we start with a random vector, say $x^{(0)} = (0, 0, 0)^t$, then from

$$x^{(k+1)} = \begin{pmatrix} 0.2500 & 0 & 0 \\ -0.0714 & 0.1429 & 0 \\ -0.0387 & 0.0357 & 0.0833 \end{pmatrix} \left[\begin{pmatrix} 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} x^{(k)} + \begin{pmatrix} 3 \\ 19 \\ 31 \end{pmatrix} \right],$$

we obtain

$$x^{(1)} = \begin{pmatrix} 0.7500 \\ 2.5000 \\ 3.1458 \end{pmatrix}, \quad x^{(2)} = \begin{pmatrix} 0.9115 \\ 2.0045 \\ 3.0085 \end{pmatrix}, \quad x^{(3)} = \begin{pmatrix} 1.0010 \\ 1.9985 \\ 2.9995 \end{pmatrix}, \quad x^{(4)} = \begin{pmatrix} 1.0000 \\ 2.0000 \\ 3.0000 \end{pmatrix}.$$

In order to obtain the same result with the Jacobi method, where

$$T_J = D^{-1}(L + U) = \begin{pmatrix} 0 & -0.2500 & 0.2500 \\ -0.2857 & 0 & -0.1429 \\ -0.0833 & 0.2500 & 0 \end{pmatrix} \quad \text{and} \quad c_J = D^{-1}b = \begin{pmatrix} 0.7500 \\ 2.7143 \\ 2.5833 \end{pmatrix},$$

we need nine iterations.

♣ **Successive Over-relaxation Method (SOR).** This method of solving a linear system of equations $Ax = b$ derived by extrapolating the Gauss-Seidel method. This extrapolation takes the form of a weighted average between the previous iterate and the computed Gauss-Seidel iterate successively for each component,

$$x_i^{(k+1)} = \omega \bar{x}_i^{(k+1)} + (1 - \omega)x_i^{(k)}.$$

In matrix terms, the SOR algorithm can be written as:

$$x^{(k+1)} = -(D + \omega L)^{-1} [\omega U + (1 - \omega)D] x^{(k)} + \omega (D + \omega L)^{-1} b,$$

If $\omega = 1$, the SOR method simplifies to the Gauss-Seidel method. A theorem due to Kahan (1958) shows that SOR fails to converge if ω is outside of the interval $(0, 2)$. In general, it is not possible to compute in advance the value of ω that will maximize the rate of convergence of SOR.

Algorithm. The algorithm for Successive Over-relaxation method:

Enter a matrix A ;

choose an initial guess x_0 , a maximum number of iterations N , a value for $\omega \in (0, 2)$, and a tolerance Tol ;

find D , representing the diagonal, L , strictly lower triangular, and U , strictly upper triangular parts of A ;

find the inverse of the lower triangular matrix $(D + \omega L)$;

set $T = -(D + \omega L)^{-1}[\omega U + (1 - \omega)D]$;

set $c = \omega (D + \omega L)^{-1}b$;

set $k = 0$;

while $k \leq N$;

set $x^{(k+1)} = T x^{(k)} + c$;

if $\|x^{(k+1)} - x^{(k)}\|_\infty \leq Tol$, then $k = N + 1$;

end;

Display $x^{(k+1)}$. ■

♣ **Symmetric Successive Over-relaxation Method (SSOR).** If the matrix A is symmetric, then the Symmetric Successive Over-relaxation method, combines two SOR sweeps together in such a way that the resulting iteration matrix is similar to a symmetric matrix. Specifically, the first SOR sweep is carried out as in SOR, but in the second sweep the unknowns are updated in the reverse order. That is,

SSOR is a forward SOR sweep followed by a backward SOR sweep. The similarity of the SSOR iteration matrix to a symmetric matrix permits the application of SSOR as a pre-conditioner for other iterative schemes for symmetric matrices. Indeed, this is the primary motivation for SSOR since its convergence rate, with an optimal value, is usually slower than the convergence rate of SOR with an optimal value.

In matrix terms, the SSOR iteration can be expressed as follows:

$$x^{(k+1)} = T_1 T_2 x^{(k)} + \omega (2 - \omega) [D + \omega U]^{-1} [D + \omega L]^{-1} b,$$

where

$$T_1 = -[D + \omega U]^{-1} [\omega L + (1 - \omega)D]$$

and

$$T_2 = -[D + \omega L]^{-1} [\omega U + (1 - \omega)D].$$

Algorithm. The algorithm for Successive Over-relaxation method:

Enter a matrix A ;

choose an initial guess x_0 , a maximum number of iterations N , a value for $\omega \in (0, 2)$, and a tolerance Tol ;

find D , representing the diagonal, L , strictly lower triangular, and U , strictly upper triangular parts of A ;

find the inverse of the upper triangular matrix $(D + \omega U)$;

find the inverse of the lower triangular matrix $(D + \omega L)$;

set $T_1 = -[D + \omega U]^{-1} [\omega L + (1 - \omega)D]$;

set $T_2 = -[D + \omega L]^{-1} [\omega U + (1 - \omega)D]$;

set $T = T_1 T_2$;

set $c = \omega (2 - \omega) [D + \omega U]^{-1} [D + \omega L]^{-1} b$;

set $k = 0$;

while $k \leq N$;

set $x^{(k+1)} = T x^{(k)} + c$;

if $\|x^{(k+1)} - x^{(k)}\|_\infty \leq Tol$, then $k = N + 1$;

end;

Display $x^{(k+1)}$. ■

♣ **Error Estimates and Perturbation Methods.** It seems intuitively reasonable that if x^* is an approximation to the solution x of $Ax = b$ and the vector $r = Ax^* - b$ which is called the **residual vector** has the property that $\|r\|$ is small, then $\|x - x^*\|$ would be small as well. Although this is often the case, certain special systems which occur quite often in practice, fail to have this property.

The **condition number** $\kappa(A)$ of a nonsingular matrix relative to a natural norm $\|\cdot\|$ is defined to be

$$\kappa(A) = \|A\| \|A^{-1}\|.$$

Since for any nonsingular matrix A

$$1 = \|I_n\| = \|A \cdot A^{-1}\| \leq \|A\| \|A^{-1}\| = \kappa(A),$$

the expectation is that the matrix A will be well-behaved (formally called a **well-conditioned matrix**) if $\kappa(A)$ is close to one and not well-behaved (called **ill-conditioned**), when $\kappa(A)$ is significantly greater than one. Behavior in this instance refers to the relative security that a small residual vector implies a correspondingly accurate approximate solution.

Theorem 3. *If x^* is an approximation to the solution of $Ax = b$ and A is a nonsingular matrix, then for any natural norm,*

$$\|x - x^*\| \leq \|r\| \|A^{-1}\|$$

and if $x \neq 0$ and $b \neq 0$

$$\frac{\|x - x^*\|}{\|x\|} \leq \kappa(A) \frac{\|r\|}{\|b\|},$$

where r is the residual vector for x^* with respect to the system $Ax = b$.

Proof. Since A is nonsingular

$$\|x - x^*\| = \|A^{-1}(b - Ax^*)\| = \|A^{-1}r\| \leq \|r\| \|A^{-1}\|.$$

Moreover, since $b = Ax$, $\|b\| \leq \|A\| \|x\|$; so

$$\frac{\|x - x^*\|}{\|x\|} = \|A\| \frac{\|x - x^*\|}{\|A\| \|x\|} \leq \|A\| \frac{\|x - x^*\|}{\|b\|} = \kappa(A) \frac{\|r\|}{\|b\|}. \quad \blacksquare$$

Example 3. The linear system $Ax = b$ given by

$$\begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix}$$

has the unique solution $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. The approximation to this solution $x^* = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$ has the residual vector

$$r = b - Ax^* = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.0002 \end{pmatrix},$$

so $\|r\|_\infty = 0.0002$. Although the norm of the residual vector is small, the approximation $x^* = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$ is obviously quite poor; in fact, $\|x - x^*\|_\infty = 2$.

The difficulty that arose in this example can be explain by the fact that for $\|\cdot\|_\infty$, the condition number

$$\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = \left\| \begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \right\|_\infty \left\| \begin{pmatrix} -10000 & 10000 \\ 5000.5 & -5000 \end{pmatrix} \right\|_\infty = (3.0001)(20,000) = 60,002$$

is extremely large. The size of the condition number for this example would certainly keep us from making hasty accuracy decision based on the residual of an approximation.

The **spectrum** of an $n \times n$ matrix is the set of all the eigenvalues of A and is denoted by $\sigma(A)$.

Theorem 4. Let A be an $n \times n$ nonsingular matrix and

$$\sigma(A) = \{\lambda_i : 0 \leq |\lambda_1| \leq |\lambda_2| \leq \cdots |\lambda_n| = \rho(A)\}.$$

Then

$$\kappa(A) \geq \left| \frac{\lambda_n}{\lambda_1} \right|.$$

Proof. The proof follows from facts that

$$\|A\| \geq \rho(A) = |\lambda_1| \quad \text{and} \quad \|A^{-1}\| \geq \rho(A^{-1}) = |1/\lambda_1|. \quad \blacksquare$$

Theorem 5. Consider a linear system $Ax = b$, where A is an $n \times n$ nonsingular matrix. Let $\delta \neq 0$, then:

- (i) If x^* is a solution to the system $(A + \delta A)x = (b + \delta b)$, then $(I_n + \delta A^{-1})x^*$ is a solution to the system $Ax = b$.
- (ii) If x^* is a solution to the system $(A + \delta I_n)x = b$, then $(I_n + A^{-1})x^*$ is a solution to the system $Ax = b$.

Proof. The proof of (i) is obvious.

- (ii) If x^* is a solution to the system $(A + \delta I_n)x = b$, then

$$(I_n + A^{-1})x^* = A^{-1}(A + \delta I_n)x^* = A^{-1}b.$$

We complete the proof by using the fact that $A^{-1}b$ is the unique solution to the system $Ax = b$. ■

According to Theorem 4 if the gap between $|\lambda_1|$ and λ_n is large, then $\kappa(A)$ becomes large. Note that if $Au_i = \lambda_i u_i$, then

$$(A + \delta A)u_i = (1 + \delta)\lambda_i u_i \quad \text{and} \quad (A + \delta I)u_i = (1 + \delta)\lambda_i u_i.$$

If the $n \times n$ matrix A in the system $Ax = b$ has a large condition number, then according to Theorem 5, by perturbing A and b or only A , we may solve the system

$$(A + \delta A)x = (b + \delta b) \quad \text{in place of} \quad Ax = b$$

or
$$(A + \delta I_n)x = b \quad \text{in place of} \quad Ax = b.$$