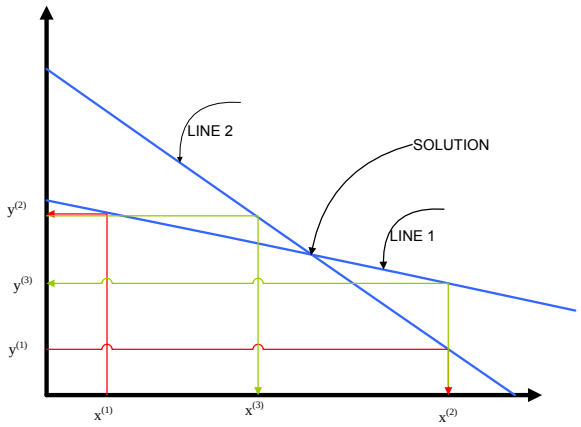
Iterative Techniques For Solving Linear Systems



Gauss elimination and its variants are called direct methods.

An iterative technique to solve the $n \times n$ linear system

$$AX=B$$

Starts with an initial approximation $X^{(0)}$ to the solution X, and generates a sequence of vectors

$$\{X^{(k)}\}_{k=0\ldots\infty}$$

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

$$X^{(k)} = TX^{(k-1)} + C$$

for each k=1,2,3,…

Jacobi Iteration

Suppose that the given linear system is

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = b_2 \\
&\quad . \\
&\quad . \\
&\quad . \\
&a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jj}x_j + \ldots + a_{jn}x_n = b_j \\
&a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nj}x_j + \ldots + a_{nn}x_n = b_n.
\end{aligned}
$$

The iteration formulas use row j of the given system to solve for $x_j^{(k+1)}$ in terms of a linear combination of the previous values $x_1^{(k)}, x_2^{(k)}, x_3^{(k)} \ldots, x_j^{(k)}, \ldots, x_n^{(k)}$:

Jacobi iteration:

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

Illustrating the Jacobi Method Graphically

To visualize the Jacobi iterations, consider the two-by-two system.

$$\begin{aligned} 2x + y &= 6 \\ x + 2y &= 6 \end{aligned}$$

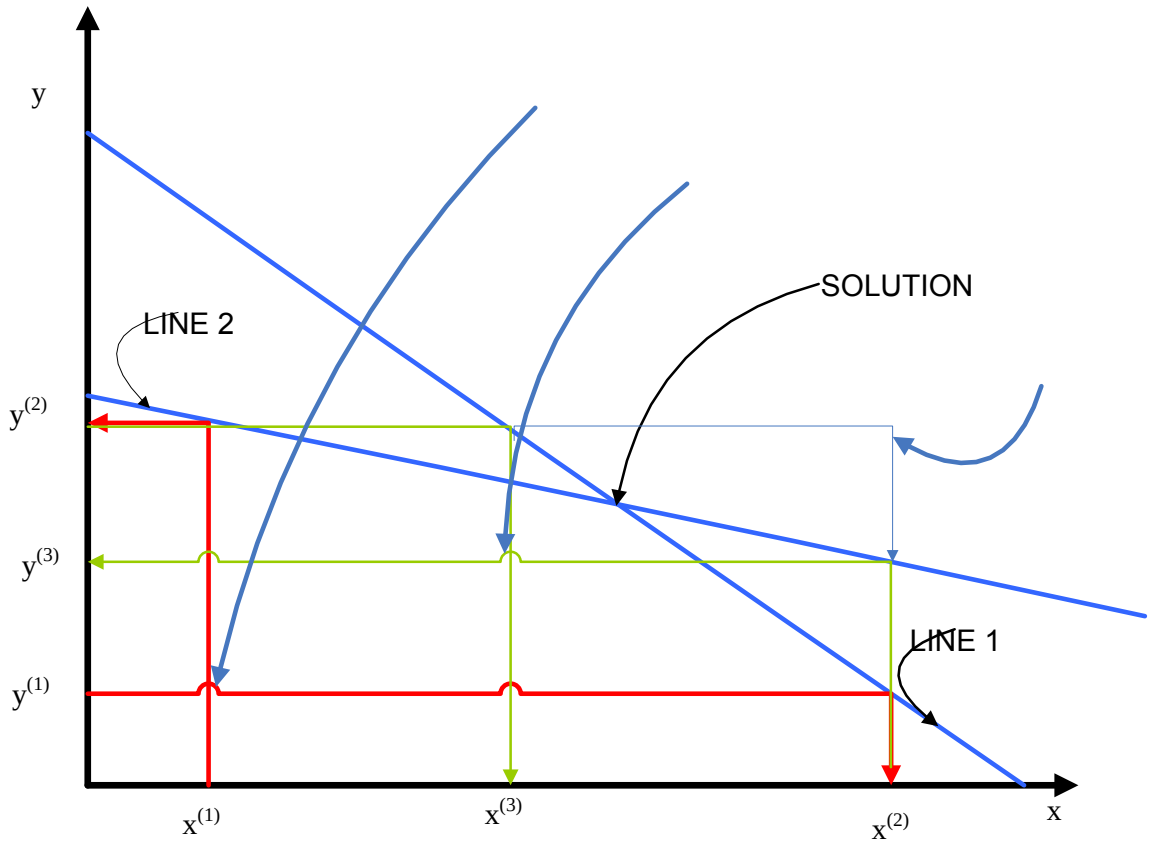
For the Jacobi method, the equations are written as

$$\begin{aligned} x &= -\frac{1}{2}y + 3 \\ y &= -\frac{1}{2}x + 3 \end{aligned}$$

Starting with $x^{(1)} = y^{(1)} = 1/2$ **, the first equation produces the next estimate for** x **using** $y^{(1)}$**, and the second equation gives the next value of** y **using** $x^{(1)}$**.**

$$\begin{aligned} x^{(2)} &= -\frac{1}{2}y^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4}, \\ y^{(2)} &= -\frac{1}{2}x^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4}. \end{aligned}$$

Notice that the new values of the variables <u>are not used until a new iteration step is begun</u>. This is called simultaneous updating.


y
SOLUTION
LINE 2
y(2)
y(3)
LINE 1
y(1)
x(1)
x(3)
x(2)
x

Convergence of the Jacobi Method

The following theorem gives a sufficient condition for Jacobi iteration to converge.

Theorem: (Jacobi Iteration). ***Suppose that A is nonsingular matrix. Then AX=B has a unique solution X=P. Iteration using Jacobi formula will produce a sequence of vectors $\{P_k\}$ that will converge to P for any choice of the starting vector P_0.***

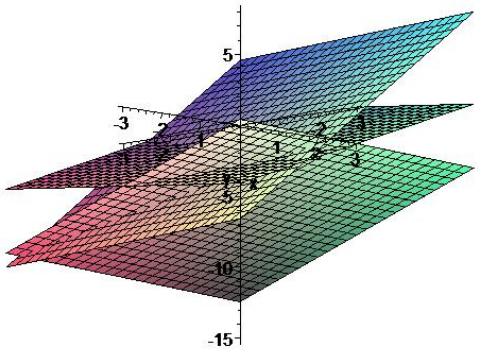
Iterative techniques are seldom used for solving linear systems of small dimension since the time required for sufficient accuracy exceeds that required for direct techniques such as he Gaussian elimination method.

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.

Example: **The linear system AX=B given by**

$$\begin{aligned} 2x_1 - x_2 + x_3 &= -1, \\ x_1 + 2x_2 - x_3 &= 6 \\ x_1 - x_2 + 2x_3 &= -3 \end{aligned}$$

```
> multiple( plot3d, [-2*x+y-1, x=-3..3, y=-3..3], [x+2*y-6, x=-3..3, y=-3..3],[-0.5*x+0.5*y-1.5, x=-3..3,y=-3..3] );
```



These equations are converted to

$$\begin{aligned} x_1 &= 0.5x_2 - 0.5x_3 - 0.5, \\ x_2 &= -0.5x_1 + 0.5x_3 + 3, \\ x_3 &= -0.5x_1 + 0.5x_2 - 1.5. \end{aligned}$$

Original system in matrix notation

$$\begin{bmatrix} 2 & -1 & 1 \\ 1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 6 \\ -3 \end{bmatrix}$$

$$\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ x_3^{(k-1)} \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix}$$

$$X^{(k)} = TX^{(k-1)} + C$$

With $x^{(0)}=(0,0,0)^T$ we find first iteration

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix}$$

For the second iteration, we have

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix} = \begin{bmatrix} 0.0 & 0.5 & -0.5 \\ -0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} + \begin{bmatrix} -0.5 \\ 3.0 \\ -1.5 \end{bmatrix} = \begin{bmatrix} 1.75 \\ 2.50 \\ 0.25 \end{bmatrix}$$

The Jacobi method converges in 13 iterations to the vector

$$X = [1.0002 \quad 2.0001 \quad -0.9997]^T$$

The true solution is

$$X = [1.000 \quad 2.000 \quad -1.0000]^T$$

```
>> true=[1 2 -1]
true =     1     2    -1
>> x=[1.0002 2.0001 -0.9997]
x =    1.0002    2.0001   -0.9997
>> error=true-x
error =  1.0e-003 * -0.2000   -0.1000   -0.3000
>> norm(error)
ans =
 3.7417e-004
```

$$Error = X^{(k)} - X^{(k-1)}$$

Norm (Error) ≤ Tolerance Value (δ) ?

Example: The linear system AX=B given by

$$\begin{aligned} 10x_1 - x_2 + 2x_3 + 0x_4 &= 6 \\ -x_1 - 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 0x_1 + 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

has solution $X=[1 \quad 2 \quad -1 \quad 1]^T$.

To convert AX=B to the form X=TX+C solve equation E_j for x_j for j=1,2,3,4 to obtain

$$\begin{aligned} x_1 &= 1/10x_2 - 1/5x_3 + 3/5, \\ x_2 &= 1/11x_1 + 1/11x_3 - 3/11x_4 + 25/11 \\ x_3 &= -1/5x_1 + 1/10x_2 + 1/10x_4 - 11/10 \\ x_4 &= -3/8x_2 + 1/8x_3 + 15/8. \end{aligned}$$

In this example,

$$T = \begin{bmatrix} 0 & 1/10 & -1/5 & 0 \\ 1/11 & 0 & 1/11 & -3/11 \\ -1/5 & 1/10 & 0 & 1/10 \\ 0 & -3/8 & 1/8 & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 3/5 \\ 25/11 \\ -11/10 \\ 15/8 \end{bmatrix}$$

For an initial approximation let $x^{(0)}=[0\ 0\ 0\ 0\ ]^T$ and generate $x^{(1)}$ by:

$$
\begin{aligned}
x_1^{(1)} &= 1/10x_2^{(0)} - 1/5x_3^{(0)} + 3/5 = 0.60, \\
x_2^{(1)} &= 1/11x_1^{(0)} + 1/11x_3^{(0)} - 3/11x_4^{(0)} + 25/11 = 2.2727 \\
x_3^{(1)} &= -1/5x_1^{(0)} + 1/10x_2^{(0)} + 1/10x_4^{(0)} - 11/10 = -1.100 \\
x_4^{(1)} &= -3/8x_2^{(0)} + 1/8x_3^{(0)} + 15/8 = 1.8750.
\end{aligned}
$$

Additional iterates, are generated in a similar manner.

$$
\begin{aligned}
x_1^{(k+1)} &= 1/10x_2^{(k)} - 1/5x_3^{(k)} + 3/5, \\
x_2^{(k+1)} &= 1/11x_1^{(k)} + 1/11x_3^{(k)} - 3/11x_4^{(k)} + 25/11 \\
x_3^{(k+1)} &= -1/5x_1^{(k)} + 1/10x_2^{(k)} + 1/10x_4^{(k)} - 11/10 \\
x_4^{(k+1)} &= -3/8x_2^{(k)} + 1/8x_3^{(k)} + 15/8
\end{aligned}
$$

$X^{(1)}=[0.60\ ,\ 2.2727\ ,\ -1.100,\ 1.875]^T$

.

.

.

$X^{(10)}=[1.001\ ,\ 1.9998\ ,\ -0.9998,\ 0.9998]^T$

The stopping condition:

The Euclidian of the difference of the solution vectors between two successive iterations be less than tolerance value (δ e.g. 0.001).

Other stopping conditions can also be used.

Parallel Computation:

The Jacobi method is particularly convenient for parallel computation, because each component of the solution can be updated independently of the other components.

Number of iterations:

Each iteration of the Jacobi method requires one matrix-vector multiplication, or $(n-1)^2$ scalar multiplications.

Convergence of the Jacobi Method:

A necessary and sufficient condition for the convergence of the Jacobi method is that the magnitude of the largest eigenvalue of the iteration matrix T be less than 1. (See Section 4.1 Applied Numerical Analysis using Matlab-Fausett)

Algorithm for Jacobi Iteration

We assume that the system **Ax=b** has been rearranged so that the matrix A is diagonally dominant. That is for each row of A:

$$|a_{i,j}| > \sum_{\substack{j=i \\ j\neq i}}^{n} |a_{i,j}|, i = 1,2,\ldots,n$$

This is a sufficient condition for converge both for this method and for the one that we discuss next. We begin with an initial approximation to the solution vector, which we store in the vector: old x.

```
            For i=1 to n

   b[i]=b[i]/a[i,i]
new_x[i]= old|x[i]
a[i,j]=a[i,j]/a[i,i]; j=1…n and i<>j
End For i

Repeat

 For i=1 To n
   old_x[i]=new_x[i]
   new_x[i]=b[i]
End For i

For i=1 To n
  For j=1 To n
 If (j<>i) Then
   new_x[i]= new_x[i]- a[i,j]*old_x[j]
    End For j
End For i
Until new_x and old_x converge to each other.
```

MATLAB M-File(Jacobi Iteration for solving linear system)

```
function X=jacobi(A,B,P,delta, maxit)
% 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
%            - maxit 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:maxit
  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=norm(X'-P);
  relerr=err/(norm(X)+eps);
  P=X';
    if (err<delta)|(relerr<delta)
    break
  end
end
X=X';
k
```

Example: The linear system AX=B given by

$$\begin{aligned} 10x_1 - x_2 + 2x_3 + 0x_4 &= 6 \\ -x_1 - 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 0x_1 + 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

has solution $X=[1\ 2\ -1\ 1]^T$.

For an initial approximation let $P=X^{(0)}=[0\ 0\ 0\ 0\ ]^T$

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1;0 3 -1 8]
A =
    10    -1     2     0
    -1    11    -1     3
     2    -1    10    -1
     0     3    -1     8
>> B=[6 25 -11 15]'
B =
     6
    25
   -11
    15
>>
>> P=[0 0 0 0]';
>> X=jacobi(A,B,P,0.0001,20)
k =
    12
X =
    1.0000
    2.0000
   -1.0000
    1.0000
```

Gauss-Seidel Iteration

Suppose that the given linear system is

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = b_1\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = b_2\\
&\quad\cdot\\
&a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jj}x_j + \ldots + a_{jn}x_n = b_j\\
&\quad\cdot\\
&a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nj}x_j + \ldots + a_{nn}x_n = b_n.
\end{aligned}
$$

The iteration formulas use row j of the given system to solve for $x_j^{(k+1)}$ in terms of a linear combination of the previous values $x_1^{(k)}, x_2^{(k)}, x_3^{(k)} \ldots, x_j^{(k)}, \ldots, x_n^{(k)}$:

Jacobi iteration uses all old coordinates to generate all new coordinates, whereas Gauss-Seidel iteration uses the new coordinates as they become available:

Jacobi Iteration:

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

Gauss-Seidel Iteration:

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

Compare?

Illustrating the Gauss-Seidel Method Graphically

To visualize the Gauss-Seidel iterations, consider the two-by-two system. (Same Problem)

$$\begin{aligned} 2x + y &= 6 \\ x + 2y &= 6 \end{aligned}$$

For the Gauss-Seidel method, the equations are written as

$$\begin{aligned} x &= -\frac{1}{2}y + 3 \\ y &= -\frac{1}{2}x + 3 \end{aligned}$$

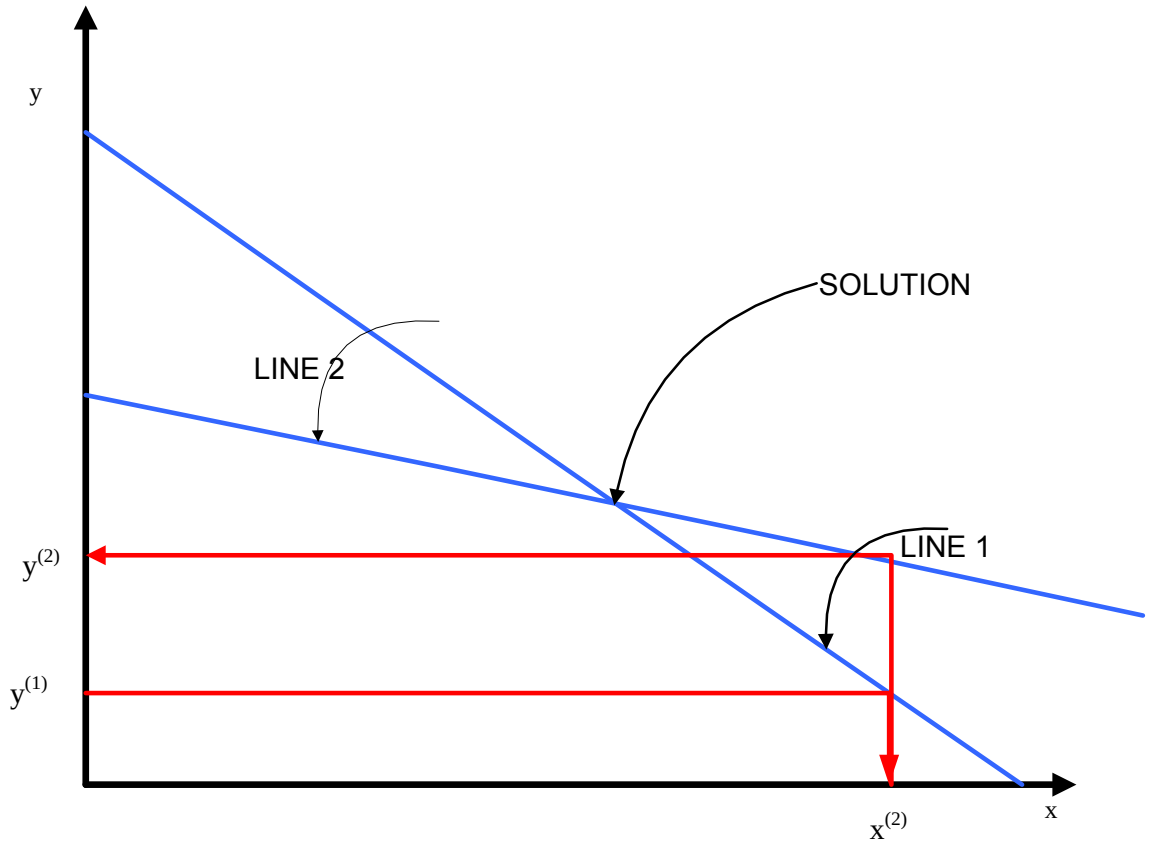
Start with $x^{(1)} = y^{(1)} = 1/2$. **The first equation produces the next estimate for** x **using** $y^{(1)}$**, and the second equation is used to find the next value of** y **(<u>using the newly computed value</u>** $x^{(2)}$ **not** $x^{(1)}$**).**

$$x^{(2)} = -\frac{1}{2}y^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4},$$
$$y^{(2)} = -\frac{1}{2}x^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4}.$$

JACOBI

$$x^{(2)} = -\frac{1}{2}y^{(1)} + 3 = -\frac{1}{4} + 3 = \frac{11}{4},$$
$$y^{(2)} = -\frac{1}{2}x^{(2)} + 3 = -\frac{11}{8} + 3 = \frac{13}{8}.$$

GAUSS-SEIDEL



Four steps for Gauss-Seidel Iteration

i	x	y
1	**2.75**	**1.625**
2	**2.1875**	**1.9062**
3	**2.0469**	**1.9766**
4	**2.0117**	**1.9941**

Example: (Gauss-Seidel) The linear system AX=B given by

$$\begin{aligned}
10x_1 - x_2 + 2x_3 + 0x_4 &= 6\\
-x_1 - 11x_2 - x_3 + 3x_4 &= 25\\
2x_1 - x_2 + 10x_3 - x_4 &= -11\\
0x_1 + 3x_2 - x_3 + 8x_4 &= 15
\end{aligned}$$

has solution $X=[1\ 2\ -1\ 1]^T$.

To convert AX=B to the form X=TX+C solve equation E_j for x_j for j=1,2,3,4 to obtain

$$\begin{aligned}
x_1 &= 1/10x_2 - 1/5x_3 + 3/5,\\
x_2 &= 1/11x_1 + 1/11x_3 - 3/11x_4 + 25/11\\
x_3 &= -1/5x_1 + 1/10x_2 + 1/10x_4 - 11/10\\
x_4 &= -3/8x_2 + 1/8x_3 + 15/8.
\end{aligned}$$

In this example,

$$T = \begin{bmatrix} 0 & 1/10 & -1/5 & 0\\ 1/11 & 0 & 1/11 & -3/11\\ -1/5 & 1/10 & 0 & 1/10\\ 0 & -3/8 & 1/8 & 0 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 3/5\\ 25/11\\ -11/10\\ 15/8 \end{bmatrix}$$

For an initial approximation let $x^{(0)}=[0\ 0\ 0\ 0\ ]^T$ and generate $x^{(1)}$ by:

$$
\begin{aligned}
x_1^{(1)} &= 1/10x_2^{(0)} - 1/5x_3^{(0)} + 3/5 = 0.60, \\
x_2^{(1)} &= 1/11x_1^{(1)} + 1/11x_3^{(0)} - 3/11x_4^{(0)} + 25/11 = 2.3272 \\
x_3^{(1)} &= -1/5x_1^{(1)} + 1/10x_2^{(1)} + 1/10x_4^{(0)} - 11/10 = -0.9873 \\
x_4^{(1)} &= -3/8x_2^{(1)} + 1/8x_3^{(1)} + 15/8 = 0.8789.
\end{aligned}
$$

Additional iterates, are generated in a similar manner.

$$
\begin{aligned}
x_1^{(k+1)} &= 1/10x_2^{(k)} - 1/5x_3^{(k)} + 3/5, \\
x_2^{(k+1)} &= 1/11x_1^{(k+1)} + 1/11x_3^{(k)} - 3/11x_4^{(k)} + 25/11 \\
x_3^{(k+1)} &= -1/5x_1^{(k+1)} + 1/10x_2^{(k+1)} + 1/10x_4^{(k)} - 11/10 \\
x_4^{(k+1)} &= -3/8x_2^{(k+1)} + 1/8x_3^{(k+1)} + 15/8
\end{aligned}
$$

$X^{(1)}=[0.60\ ,\ 2.3272\ ,\ -0.9873,\ 0.8789]^T$

.

.

.

$X^{(10)}=[1.001\ ,\ 2.0000\ ,\ -1.0000,\ 1.0000]^T$

Algorithm for Gauss-Seidel Iteration

We assume as we did in the previous algorithm that the system **Ax=b** has been rearranged so that the matrix A is diagonally dominant. As before, we begin with an initial approximation to the solution vetor, which we store in the vector : x.

```
            For i=1 to n
    b[i]=b[i]/a[i,i]
a[i,j]=a[i,j]/a[i,i]; j=1…n and i<>j
End For i
While Not (yet convergent) Do
 For i=1 To n
      x[i]=b[i]
      For j=1 To n
           If (j<>i) Then
            x[i]= x[i]- a[i,j]*x[j]
    End For j
End For i
End While
```

MATLAB M-File (Gauss-Seidel Iteration for solving linear system)

```
function X=gseid(A,B,P,delta, maxit)
% 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
%            - maxit is the maximum number of iterations
% Output - X is an N x 1 matrix: the gauss-seidel approximation to
%              the solution of AX = B
N = length(B);
for k=1:maxit
  for j=1:N
    if j==1
      X(1)=(B(1)-A(1,2:N)*P(2:N))/A(1,1);
    else if 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=norm(X'-P);
  relerr=err/(norm(X)+eps);
  P=X';
    if (err<delta)|(relerr<delta)
    break
  end
end
X=X';
k
```

Example: The linear system AX=B given by

$$\begin{aligned} 10x_1 - x_2 + 2x_3 + 0x_4 &= 6 \\ -x_1 - 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 0x_1 + 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

has solution $X=[1\ 2\ -1\ 1]^T$.

For an initial approximation let $P=X^{(0)}=[0\ 0\ 0\ 0\ ]^T$

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1;0 3 -1 8]
A =
    10    -1     2     0
    -1    11    -1     3
     2    -1    10    -1
     0     3    -1     8
>> B=[6 25 -11 15]'
B =
     6
    25
   -11
    15
>> P=[0 0 0 0]'
P =
     0
     0
     0
     0
>> X=gseid(A,B,P,0.0001,20)
k =     6        COMPARE WITH JACOBI   ITEARTION
X =
    1.0000
    2.0000
   -1.0000
    1.0000
```