Iterative Techniques in Matrix Algebra

Jacobi & Gauss-Seidel Iterative Techniques I

Numerical Analysis (9th Edition)
R L Burden & J D Faires

Beamer Presentation Slides
prepared by
John Carroll
Dublin City University

© 2011 Brooks/Cole, Cengage Learning
Outline

1. Introducing Iterative Techniques for Linear Systems
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
4. The Jacobi Iterative Algorithm
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
4. The Jacobi Iterative Algorithm
Introduction

We will now describe the Jacobi and the Gauss-Seidel iterative methods, classic methods that date to the late eighteenth century.
The Jacobi & Gauss-Seidel Methods

Introduction

We will now describe the Jacobi and the Gauss-Seidel iterative methods, classic methods that date to the late eighteenth century.

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 Gaussian elimination.
The Jacobi & Gauss-Seidel Methods

Introduction

We will now describe the Jacobi and the Gauss-Seidel iterative methods, classic methods that date to the late eighteenth century.

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 Gaussian elimination.

For large systems with a high percentage of 0 entries, however, these techniques are efficient in terms of both computer storage and computation.
The Jacobi & Gauss-Seidel Methods

Iterative Technique

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

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

$$Ax = b$$

starts with an initial approximation

$$x^{(0)}$$

to the solution $x$
Iterative Technique

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

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

that converges to $x$. 
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
4. The Jacobi Iterative Algorithm
The **Jacobi iterative method** is obtained by solving the $i$th equation in $Ax = b$ for $x_i$ to obtain (provided $a_{ii} \neq 0$)

$$x_i = \sum_{j=1, j \neq i}^{n} \left( -\frac{a_{ij}x_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}}, \quad \text{for } i = 1, 2, \ldots, n$$
The Jacobi iterative method is obtained by solving the $i$th equation in $Ax = b$ for $x_i$ to obtain (provided $a_{ii} \neq 0$)

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

For each $k \geq 1$, generate the components $x_i^{(k)}$ of $x^{(k)}$ from the components of $x^{(k-1)}$ by

$$
x_i^{(k)} = \frac{1}{a_{ii}} \left[ \sum_{\substack{j=1 \atop j \neq i}}^{n} \left( -a_{ij} x_j^{(k-1)} \right) + b_i \right], \quad \text{for } i = 1, 2, \ldots, n
$$
Jacobi’s Method

Example

The linear system $Ax = b$ given by

\[
\begin{align*}
E_1 & : ~ 10x_1 - x_2 + 2x_3 = 6 \\
E_2 & : ~ -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\
E_3 & : ~ 2x_1 - x_2 + 10x_3 - x_4 = -11, \\
E_4 & : ~ 3x_2 - x_3 + 8x_4 = 15
\end{align*}
\]

has the unique solution $x = (1, 2, -1, 1)^t$. 
Example

The linear system $Ax = b$ given by

\[
E_1 : \quad 10x_1 - x_2 + 2x_3 = 6 \\
E_2 : \quad -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\
E_3 : \quad 2x_1 - x_2 + 10x_3 - x_4 = -11, \\
E_4 : \quad 3x_2 - x_3 + 8x_4 = 15,
\]

has the unique solution $x = (1, 2, -1, 1)^t$. Use Jacobi’s iterative technique to find approximations $x^{(k)}$ to $x$ starting with $x^{(0)} = (0, 0, 0, 0)^t$ until

\[
\frac{\|x^{(k)} - x^{(k-1)}\|_\infty}{\|x^{(k)}\|_\infty} < 10^{-3}
\]
Jacobi’s Method: Example

Solution (1/4)

We first solve equation $E_i$ for $x_i$, for each $i = 1, 2, 3, 4$, to obtain

\[
\begin{align*}
    x_1 &= \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5} \\
    x_2 &= \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11} \\
    x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10} \\
    x_4 &= -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8}
\end{align*}
\]
Jacobi’s Method: Example

Solution (2/4)

From the initial approximation \( x^{(0)} = (0, 0, 0, 0)^t \) we have \( x^{(1)} \) given by

\[
\begin{align*}
x_1^{(1)} &= \frac{1}{10} x_2^{(0)} - \frac{1}{5} x_3^{(0)} + \frac{3}{5} = 0.6000 \\
x_2^{(1)} &= \frac{1}{11} x_1^{(0)} + \frac{1}{11} x_3^{(0)} - \frac{3}{11} x_4^{(0)} + \frac{25}{11} = 2.2727 \\
x_3^{(1)} &= -\frac{1}{5} x_1^{(0)} + \frac{1}{10} x_2^{(0)} + \frac{1}{10} x_4^{(0)} - \frac{11}{10} = -1.1000 \\
x_4^{(1)} &= -\frac{3}{8} x_2^{(0)} + \frac{1}{8} x_3^{(0)} + \frac{15}{8} = 1.8750
\end{align*}
\]
### Solution (3/4)

Additional iterates, \( x^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)})^t \), are generated in a similar manner and are summarized as follows:

<table>
<thead>
<tr>
<th>( k )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1^{(k)} )</td>
<td>0.0</td>
<td>0.6000</td>
<td>1.0473</td>
<td>0.9326</td>
<td>1.0152</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( x_2^{(k)} )</td>
<td>0.0</td>
<td>2.2727</td>
<td>1.7159</td>
<td>2.053</td>
<td>1.9537</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( x_3^{(k)} )</td>
<td>0.0</td>
<td>-1.1000</td>
<td>-0.8052</td>
<td>-1.0493</td>
<td>-0.9681</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( x_4^{(k)} )</td>
<td>0.0</td>
<td>1.8750</td>
<td>0.8852</td>
<td>1.1309</td>
<td>0.9739</td>
<td>( \cdots )</td>
</tr>
</tbody>
</table>
The process was stopped after 10 iterations because

\[
\frac{\|x^{(10)} - x^{(9)}\|_\infty}{\|x^{(10)}\|_\infty} = \frac{8.0 \times 10^{-4}}{1.9998} < 10^{-3}
\]

In fact, \(\|x^{(10)} - x\|_\infty = 0.0002\).
Outline

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
4. The Jacobi Iterative Algorithm
In general, iterative techniques for solving linear systems involve a process that converts the system $Ax = b$ into an equivalent system of the form

$$x = Tx + c$$

for some fixed matrix $T$ and vector $c$. 
Jacobi’s Method in the form $x^{(k)} = T x^{(k-1)} + c$

A More General Representation

- In general, iterative techniques for solving linear systems involve a process that converts the system $A x = b$ into an equivalent system of the form
  
  $$x = T x + c$$

  for some fixed matrix $T$ and vector $c$.

- After the initial vector $x^{(0)}$ is selected, the sequence of approximate solution vectors is generated by computing
  
  $$x^{(k)} = T x^{(k-1)} + c$$

  for each $k = 1, 2, 3, \ldots$ (reminiscent of the fixed-point iteration for solving nonlinear equations).
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

- The Jacobi method can be written in the form

\[
\mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c}
\]

by splitting \( A \) into its diagonal and off-diagonal parts.
Jacobi’s Method in the form $\mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c}$

A More General Representation (Cont’d)

- The Jacobi method can be written in the form

$$\mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c}$$

by splitting $A$ into its diagonal and off-diagonal parts.

To see this, let $\mathbf{D}$ be the diagonal matrix whose diagonal entries are those of $A$, $-\mathbf{L}$ be the strictly lower-triangular part of $A$, and $-\mathbf{U}$ be the strictly upper-triangular part of $A$ where

$$A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}$$
Jacobi’s Method in the form $\mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c}$

A More General Representation (Cont’d)

We then write $A = D - L - U$
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

We then write \( A = D - L - U \) where

\[
D = \begin{bmatrix}
 a_{11} & 0 & \cdots & 0 \\
 0 & a_{22} & \ddots & \vdots \\
 \vdots & \ddots & \ddots & 0 \\
 0 & \cdots & 0 & a_{nn}
\end{bmatrix}
\]
Jacobi’s Method in the form $x^{(k)} = Tx^{(k-1)} + c$

A More General Representation (Cont’d)

We then write $A = D - L - U$ where

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix}$$
Jacobi’s Method in the form $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$

**A More General Representation (Cont’d)**

We then write $A = D - L - U$ where

$$
D = \begin{bmatrix}
    a_{11} & 0 & \cdots & 0 \\
    0 & a_{22} & \ddots & \vdots \\
    \vdots & \ddots & \ddots & 0 \\
    0 & \cdots & 0 & a_{nn}
\end{bmatrix}
$$

$$
L = \begin{bmatrix}
    0 & \cdots & \cdots & \cdots & 0 \\
    -a_{21} & \ddots & \ddots & \ddots & \vdots \\
    \vdots & \ddots & \ddots & \ddots & \vdots \\
    -a_{n1} & \cdots & -a_{n,n-1} & 0 & \cdots
\end{bmatrix}
$$

and

$$
U = \begin{bmatrix}
    0 & -a_{12} & \cdots & -a_{1n} \\
    \vdots & \ddots & \ddots & \ddots \\
    \vdots & \ddots & \ddots & \ddots \\
    0 & \cdots & -a_{n-1,n} & 0
\end{bmatrix}
$$
Jacobi’s Method in the form $x^{(k)} = T x^{(k-1)} + c$

A More General Representation (Cont’d)

The equation $Ax = b$, or $(D - L - U)x = b$, is then transformed into

$$Dx = (L + U)x + b$$
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

The equation \( \mathbf{A} \mathbf{x} = \mathbf{b} \), or \( (D - L - U)\mathbf{x} = \mathbf{b} \), is then transformed into

\[
D\mathbf{x} = (L + U)\mathbf{x} + \mathbf{b}
\]

and, if \( D^{-1} \) exists, that is, if \( a_{ii} \neq 0 \) for each \( i \), then

\[
\mathbf{x} = D^{-1}(L + U)\mathbf{x} + D^{-1}\mathbf{b}
\]
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

The equation \( A \mathbf{x} = \mathbf{b} \), or \( (D - L - U) \mathbf{x} = \mathbf{b} \), is then transformed into

\[
D \mathbf{x} = (L + U) \mathbf{x} + \mathbf{b}
\]

and, if \( D^{-1} \) exists, that is, if \( a_{ii} \neq 0 \) for each \( i \), then

\[
\mathbf{x} = D^{-1}(L + U) \mathbf{x} + D^{-1} \mathbf{b}
\]

This results in the matrix form of the Jacobi iterative technique:

\[
\mathbf{x}^{(k)} = D^{-1}(L + U) \mathbf{x}^{(k-1)} + D^{-1} \mathbf{b}, \quad k = 1, 2, \ldots
\]
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

Introducing the notation \( T_j = D^{-1}(L + U) \) and \( c_j = D^{-1}b \)
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

Introducing the notation \( T_j = D^{-1}(L + U) \) and \( c_j = D^{-1}b \) gives the Jacobi technique the form

\[
\mathbf{x}^{(k)} = T_j \mathbf{x}^{(k-1)} + \mathbf{c}_j
\]
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

A More General Representation (Cont’d)

Introducing the notation \( T_j = D^{-1}(L + U) \) and \( c_j = D^{-1}b \) gives the Jacobi technique the form

\[
\mathbf{x}^{(k)} = T_j \mathbf{x}^{(k-1)} + \mathbf{c}_j
\]

In practice, this form is only used for theoretical purposes while

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

is used in computation.
Jacobi’s Method in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \)

Example

Express the Jacobi iteration method for the linear system \( A\mathbf{x} = \mathbf{b} \) given by

\[
E_1 : \quad 10x_1 - x_2 + 2x_3 = 6 \\
E_2 : \quad -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\
E_3 : \quad 2x_1 - x_2 + 10x_3 - x_4 = -11 \\
E_4 : \quad 3x_2 - x_3 + 8x_4 = 15
\]

in the form \( \mathbf{x}^{(k)} = T \mathbf{x}^{(k-1)} + \mathbf{c} \).
Jacobi’s Method in the form $x^{(k)} = Tx^{(k-1)} + c$

Solution (1/2)

We saw earlier that the Jacobi method for this system has the form

\[
\begin{align*}
    x_1 &= \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5} \\
    x_2 &= \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11} \\
    x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10} \\
    x_4 &= -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8}
\end{align*}
\]
Jacobi’s Method in the form $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$

Solution (2/2)

Hence, we have

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

1. Introducing Iterative Techniques for Linear Systems
2. The Jacobi Iterative Method
3. Converting $Ax = b$ into an Equivalent System
4. The Jacobi Iterative Algorithm
The Jacobi Iterative Algorithm (1/2)

To solve $Ax = b$ given an initial approximation $x^{(0)}$: 

INPUT
- the number of equations and unknowns $n$;
- the entries $a_{ij}$, $1 \leq i, j \leq n$ of the matrix $A$;
- the entries $b_i$, $1 \leq i \leq n$ of $b$;
- the entries $x_0^i$, $1 \leq i \leq n$ of $X_0 = x^{(0)}$;
- tolerance $TOL$;
- maximum number of iterations $N$.

OUTPUT
- the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.
The Jacobi Iterative Algorithm (1/2)

To solve $Ax = b$ given an initial approximation $x^{(0)}$:

**INPUT**  
the number of equations and unknowns $n$;  
the entries $a_{ij}, 1 \leq i, j \leq n$ of the matrix $A$;  
the entries $b_i, 1 \leq i \leq n$ of $b$;  
the entries $XO_i, 1 \leq i \leq n$ of $XO = x^{(0)}$;  
tolerance $TOL$;  
maximum number of iterations $N$. 

**OUTPUT**  
the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.
The Jacobi Iterative Algorithm (1/2)

To solve $Ax = b$ given an initial approximation $x^{(0)}$:

**INPUT**
- the number of equations and unknowns $n$;
- the entries $a_{ij}$, $1 \leq i, j \leq n$ of the matrix $A$;
- the entries $b_i$, $1 \leq i \leq n$ of $b$;
- the entries $XO_i$, $1 \leq i \leq n$ of $XO = x^{(0)}$;
- tolerance $TOL$;
- maximum number of iterations $N$.

**OUTPUT**
- the approximate solution $x_1, \ldots, x_n$ or a message that the number of iterations was exceeded.
The Jacobi Iterative Algorithm (2/2)

Step 1 Set $k = 1$
Step 2 While $(k \leq N)$ do Steps 3–6
Step 1  Set \( k = 1 \)

Step 2  While \((k \leq N)\) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)
Step 1  Set \( k = 1 \)
Step 2  While \((k \leq N)\) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)

\[
\text{set } x_i = \frac{1}{a_{ii}} \left[ - \sum_{j=1}^{n} (a_{ij}XO_j) + b_i \right]
\]
The Jacobi Iterative Algorithm (2/2)

Step 1  Set \( k = 1 \)

Step 2  While \( (k \leq N) \) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)

set \( x_i = \frac{1}{a_{ii}} \left[ -\sum_{j=1, j\neq i}^{n} (a_{ij}XO_j) + b_j \right] \)

Step 4  If \( \|x - XO\| < TOL \) then OUTPUT \((x_1, \ldots, x_n)\)
The Jacobi Iterative Algorithm (2/2)

Step 1  Set $k = 1$
Step 2  While $(k \leq N)$ do Steps 3–6

Step 3  For $i = 1, \ldots, n$
        
        set $x_i = \frac{1}{a_{ii}} \left[ - \sum_{j=1}^{n} (a_{ij}XO_j) + b_i \right]$

Step 4  If $\|x - XO\| < TOL$ then OUTPUT $(x_1, \ldots, x_n)$
        $(The \ procedure \ was \ successful)$
        STOP

$(The \ procedure \ was \ successful)$
STOP
The Jacobi Iterative Algorithm (2/2)

Step 1  Set \( k = 1 \)
Step 2  While \(( k \leq N)\) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)
set \( x_i = \frac{1}{a_{ii}} \left[ -\sum_{j=1}^{n} (a_{ij}XO_j) + b_i \right] \)

Step 4  If \( \|x - XO\| < TOL \) then OUTPUT \((x_1, \ldots, x_n)\)
\( (The \ procedure \ was \ successful) \)
STOP

Step 5  Set \( k = k + 1 \)
**The Jacobi Iterative Algorithm (2/2)**

Step 1  Set \( k = 1 \)

Step 2  While \((k \leq N)\) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)

\[
\text{set } x_i = \frac{1}{a_{ii}} \left[ - \sum_{j=1}^{n} a_{ij}XO_j + b_i \right]
\]

Step 4  If \( \|x - XO\| < TOL \) then OUTPUT \((x_1, \ldots, x_n)\)

\((The\ procedure\ was\ successful)\)

STOP

Step 5  Set \( k = k + 1 \)

Step 6  For \( i = 1, \ldots, n \) set \( XO_i = x_i \)
The Jacobi Iterative Algorithm (2/2)

Step 1  Set \( k = 1 \)
Step 2  While \((k \leq N)\) do Steps 3–6

Step 3  For \( i = 1, \ldots, n \)
        set \( x_i = \frac{1}{a_{ii}} \left[ -\sum_{j=1}^{n} (a_{ij}XO_j) + b_i \right] \)

Step 4  If \( \|x - XO\| < TOL \) then OUTPUT \((x_1, \ldots, x_n)\)
        \((The\ procedure\ was\ successful)\)
        STOP

Step 5  Set \( k = k + 1 \)

Step 6  For \( i = 1, \ldots, n \) set \( XO_i = x_i \)

Step 7  OUTPUT \(\text{‘Maximum number of iterations exceeded’}\)
        \((The\ procedure\ was\ successful)\)
        STOP
Step 3 of the algorithm requires that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$. If one of the $a_{ii}$ entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no $a_{ii} = 0$. To speed convergence, the equations should be arranged so that $a_{ii}$ is as large as possible. Another possible stopping criterion in Step 4 is to iterate until $\|x(k) - x(k-1)\| \|x(k)\|$ is smaller than some prescribed tolerance. For this purpose, any convenient norm can be used, the usual being the $l_\infty$ norm.
Step 3 of the algorithm requires that \( a_{ii} \neq 0 \), for each \( i = 1, 2, \ldots, n \). If one of the \( a_{ii} \) entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no \( a_{ii} = 0 \).
Step 3 of the algorithm requires that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$. If one of the $a_{ii}$ entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no $a_{ii} = 0$.

To speed convergence, the equations should be arranged so that $a_{ii}$ is as large as possible.
Step 3 of the algorithm requires that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$. If one of the $a_{ii}$ entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no $a_{ii} = 0$.

To speed convergence, the equations should be arranged so that $a_{ii}$ is as large as possible.

Another possible stopping criterion in Step 4 is to iterate until

$$\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|}$$

is smaller than some prescribed tolerance.
Comments on the Algorithm

- Step 3 of the algorithm requires that $a_{ii} \neq 0$, for each $i = 1, 2, \ldots, n$. If one of the $a_{ii}$ entries is 0 and the system is nonsingular, a reordering of the equations can be performed so that no $a_{ii} = 0$.
- To speed convergence, the equations should be arranged so that $a_{ii}$ is as large as possible.
- Another possible stopping criterion in Step 4 is to iterate until
  \[
  \frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|}
  \]
  is smaller than some prescribed tolerance.
- For this purpose, any convenient norm can be used, the usual being the $l_{\infty}$ norm.
Questions?