Numerical Analysis

Mathematics of Scientific Computing

主讲人　邱欣欣
幻灯片制作　邱欣欣

中国海洋大学　信息科学与工程学院

2013 年 10 月 18 日

Solving Systems of Linear Equations

Contents

Contents

Basic Gaussian Elimination

- $$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$Ax = b$$

Basic Gaussian Elimination

- For the process, there are $n-1$ principal steps. In the first step, we refer to the first equation as the first **pivot equation** and to a_{11} as the first **pivot element**. For the remaining equations $(2 \leqslant i \leqslant n)$,

$$
\begin{cases}
a_{ij} \leftarrow a_{ij} - (\dfrac{a_{i1}}{a_{11}})a_{1j} & (1 \leqq j \leqq n) \\
b_i \leftarrow b_i - (\dfrac{a_{i1}}{a_{11}})b_1
\end{cases}
$$

Note that the quantities (a_{i1}/a_{11}) are the **multipliers**.

Basic Gaussian Elimination

- Just prior to the kth step in the forward elimination, the system will appear as follows:

$$
\begin{bmatrix}
a_{11} & a_{12} & a_{13} & \cdots & & \cdots & & \cdots & a_{1n} \\
0 & a_{22} & a_{23} & \cdots & & \cdots & & \cdots & a_{2n} \\
0 & 0 & a_{33} & \cdots & & \cdots & & \cdots & a_{3n} \\
\vdots & \vdots & \vdots & \ddots & & & & & \vdots \\
0 & 0 & 0 & \cdots & a_{kk} & \cdots & a_{kj} & \cdots & a_{kn} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & a_{ik} & \cdots & a_{ij} & \cdots & a_{in} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\
0 & 0 & 0 & \cdots & a_{nk} & \cdots & a_{nj} & \cdots & a_{nn}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_k \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_k \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}
$$

Basic Gaussian Elimination

- We compute for each remaining equation $(k+1 \leqslant i \leqslant n)$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - (\dfrac{a_{ik}}{a_{kk}})a_{kj} & \quad (k \leqq j \leqq n) \\ b_i \leftarrow b_i - (\dfrac{a_{ik}}{a_{kk}})b_k & \end{cases}$$

- Obviously, we must assume that all the divisors in this algorithm are nonzero.

Pseudocode

input $n, (a_{ij}), (b_i)$
for $k = 1$ to $n - 1$ do
for $i = k + 1$ to n do
$z \leftarrow a_{ik}/a_{kk}$
$a_{ik} \leftarrow 0$
for $j = k + 1$ to n do
$a_{ij} \leftarrow a_{ij} - z a_{kj}$
end do
$b_i \leftarrow b_i - z b_k$
end do
end do
output $(a_{ij}), (b_i)$

Basic Gaussian Elimination

```
function a=Gaussian(a)
n=length(a);
for k=1:n-1
    for i=k+1:n
        z=a(i,k)/a(k,k);
        a(i,k)=0;
        for j=k+1:n
            a(i,j)=a(i,j)-z*a(k,j);
        end
    end
end
```

```
function x=GaussianBacksub(a,b)
n=length(a);
for k=1:n-1
    for i=k+1:n
        z=a(i,k)/a(k,k);a(i,k)=0;
        for j=k+1:n
            a(i,j)=a(i,j)-z*a(k,j);
        end
        b(i)=b(i)-z*b(k);
    end
end
for i=n:-1:1
    s=b(i);
    for j=i+1:1:n
        s=s-a(i,j)*x(j);
    end
    x(i)=s/a(i,i);
end
```

Basic Gaussian Elimination

- $A = A^{(1)} \to A^{(2)} \to \ldots \to A^{(n)}$

$$\begin{bmatrix} a_{11}^{(k)} & \ldots & a_{1,k-1}^{(k)} & a_{1k}^{(k)} & \ldots & a_{1j}^{(k)} & \ldots & a_{1n}^{(k)} \\ \vdots & \ddots & & \vdots & & \vdots & & \vdots \\ 0 & \ldots & a_{k-1,k-1}^{(k)} & a_{k-1,k}^{(k)} & \ldots & a_{k-1,j}^{(k)} & \ldots & a_{k-1,n}^{(k)} \\ 0 & \ldots & 0 & a_{kk}^{(k)} & \ldots & a_{kj}^{(k)} & \ldots & a_{kn}^{(k)} \\ 0 & \ldots & 0 & a_{k+1,k}^{(k)} & \ldots & a_{k+1,j}^{(k)} & \ldots & a_{k+1,n}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ 0 & \ldots & 0 & a_{ik}^{(k)} & \ldots & a_{ij}^{(k)} & \ldots & a_{in}^{(k)} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ 0 & \ldots & 0 & a_{nk}^{(k)} & \ldots & a_{nj}^{(k)} & \ldots & a_{nn}^{(k)} \end{bmatrix}$$

Basic Gaussian Elimination

- Our task is to describe how $A^{(k+1)}$ is obtained from $A^{(k)}$. The formula is

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & \text{if } i \leqslant k \\ a_{ij}^{(k)} - (a_{ik}^{(k)}/a_{kk}^{(k)})a_{kj}^{(k)} & \text{if } i \geqslant k+1 and\, j \geqslant k+1 \\ 0 & \text{if } i \geqslant k+1 and\, j \leqslant k \end{cases}$$

Then we set $U = A^{(n)}$ and define L by

$$l_{ik} = \begin{cases} a_{ik}^{(k)}/a_{kk}^{(k)} & \text{if } i \geqslant k+1 \\ 1 & \text{if } i = k \\ 0 & \text{if } i \leqslant k-1 \end{cases}$$

Basic Gaussian Elimination

THEOREM 1 (Theorem on Nonzero Privots)

If all the pivot elements $a_{kk}^{(k)}$ are nonzero in the process just described, then $A = LU$.

Proof.

Observe that $a_{ij}^{(k+1)} = a_{ij}^{(k)}$ if $i \leqslant k$ or $j \leqslant k-1$. Note $u_{kj} = a_{kj}^{(n)} = a_{kj}^{(k)}$, $l_{ik} = 0$ if $k > i$, and $u_{kj} = 0$ if $k > j$. Now let $i \leqslant j$. We have

$$\begin{aligned}(LU)_{ij} &= \sum_{k=1}^{n} l_{ik} u_{kj} = \sum_{k=1}^{i} l_{ik} u_{kj}^{(k)} = \sum_{k=1}^{i} l_{ik} a_{kj}^{(k)} \\ &= \sum_{k=1}^{i-1} l_{ik} a_{kj}^{(k)} + l_{ii} a_{ij}^{(i)}\end{aligned}$$

Basic Gaussian Elimination

Proof.

$$
\begin{aligned}
&= \sum_{k=1}^{i-1} (a_{ik}^{(k)} / a_{kk}^{(k)}) a_{kj}^{(k)} + a_{ij}^{(i)} \\
&= \sum_{k=1}^{i-1} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) + a_{ij}^{(i)} \\
&= a_{ij}^{(1)} = a_{ij}
\end{aligned}
$$

□

Basic Gaussian Elimination

Proof.

If $i > j$, then

$$\begin{aligned}
(LU)_{ij} &= \sum_{k=1}^{n} l_{ik}u_{kj} = \sum_{k=1}^{j} l_{ik}a_{kj}^{(k)} \\
&= \sum_{k=1}^{j}(a_{ij}^{(k)} - a_{ij}^{(k+1)}) \\
&= a_{ij}^{(1)} - a_{ij}^{j+1} \\
&= a_{ij}^{(1)} = a_{ij}
\end{aligned}$$

Since $a_{ij}^{(k)} = 0$ if $i \geqslant j+1$ and $k \geqslant j+1$. □

Contents

Pivoting

- The first example

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The algorithm fails because $a_{11} = 0$.

Pivoting

- Another example, which ε is a small number different from 0.

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The solution is

$$\begin{cases} x_2 = (2 - \varepsilon^{-1})/(1 - \varepsilon^{-1}) \approx 1 \\ x_1 = (1 - x_2)\varepsilon^{-1} \approx 0 \end{cases}$$

But the correct solution is

$$\begin{cases} x_1 = 1/(1 - \varepsilon) \approx 1 \\ x_2 = (1 - 2\varepsilon)/(1 - \varepsilon) \approx 1 \end{cases}$$

Pivoting

- Interchanging the two equaions in example 1

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

The solution is

$$\begin{cases} x_2 = (1 - 2\varepsilon)/(1 - \varepsilon) \approx 1 \\ x_1 = (2 - x_1) \approx 1 \end{cases}$$

Contents

Gaussian Elimination with Scaled Row Pivoting

$$Ax = b$$

The algorithm has two parts: a factorization phase and a solution phase.

- The factorization phase is designed to produce the LU-decomposition of PA, the permuted linear system is $PAx = Pb$. P is a permutation matrix.
  - The factorization is obtained from a modified Gaussian elimination algorithm to be explained below.
- In the solution phase, we consider two equations $Lz = Pb$ and $Ux = z$.
  - b is rearranged according to P and $b \leftarrow Pb$.
  - $Lz = b$ is solved for z and $b \leftarrow L^{-1}b$.
  - Then back substitution is used to solve $Ux = b$ for $x_n, x_{n-1}, \ldots, x_1$.

Gaussian Elimination with Scaled Row Pivoting

- Computing the **scale** of each row. We put

$$s_i = \max_{1 \leqslant j \leqslant n} |a_{ij}| = \max\{|a_{i1}|, |a_{i2}|, \ldots, |a_{in}|\} \qquad (1 \leqslant i \leqslant n)$$

  These n numbers are recorded in the scale vector $s = [s_1, s_2, \ldots, s_n]$.
- Firstly, define the index vector p to be $[p_1, p_2, \ldots, p_n] = [1, 2, \ldots, n]$. Selecting an index j for which $|a_{i1}|/s_i$ is largest. Interchanging p_j with p_1 in the index vector. Next, use multipliers $(a_{p_i 1}/a_{p_1 1})$ times row 1 , and subtract from equations p_i for $2 \leqslant i \leqslant n$.
- At step k, select j to be the first index corresponding to the largest of the ratios, $\{|a_{p_i k}|/s_{p_i} \quad k \leqslant i \leqslant n\}$, interchanging p_j with p_k in p.

Pseudocode for Factorization phase

- input $n, (a_{ij})$
  for $i = 1$ to n do
  $p_i \leftarrow i$
  $smax \leftarrow 0$
  for $j = 1$ to n do
  $smax \leftarrow max(smax, |a_{ij}|)$
  end do
  $s_i \leftarrow smax$
  end do

- for $k = 1$ to $n-1$ do
  $rmax \leftarrow 0$
  for $i = k$ to n do
  $r \leftarrow |a_{p_i k}/s_{p_i}|$
  if $(r > rmax)$ then
  $rmax \leftarrow r \quad j \leftarrow i$
  end if
  end do
  $p_j \leftrightarrow p_k$
  for $i = k+1$ to n do
  $z \leftarrow a_{p_i k}/a_{p_k k}; a_{p_i k} \leftarrow z$
  for $j = k+1$ to n do
  $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
  end do
  end do
  end do
  output $(a_{ij}), (p_i)$

Pseudocode for Solution phase

- for $k = 1$ to $n-1$ do
  for $i = k+1$ to n do
  $b_{p_i} \leftarrow b_{p_i} - a_{p_i k} b_{p_k}$
  end do
  end do
  for $i = n$ to 1 step -1 do
  $x_i \leftarrow (b_{p_i} - \sum_{j=i+1}^{n} a_{p_i j} x_j)/a_{p_i i}$
  end do
  output (x_i)

```
function x=ScaleGaussian(a,b)
n=length(a);
for i=1:n
    p(i)=i;
    smax=0;
    for j=1:n
    smax=max(smax, abs(a(i,j)));
    end
    s(i)=smax;
end
for k=1:n-1
    format rat rmax=0;
    for i=k:n
        r=abs(a(p(i),k))/s(p(i))
        if r>rmax
            rmax=r;j=i;
        end
    end
```

```
            c=p(k);p(k)=p(j);p(j)=c;
        for i=k+1:n
            z=a(p(i),k)/a(p(k),k);a(p(i),k)=z;
            for j=k+1:n
                a(p(i),j)=a(p(i),j)-z*a(p(k),j);
            end
        end
        for i=k+1:n
            b(p(i))=b(p(i))-a(p(i),k)*b(p(k));
        end
end
for i=n:-1:1
    q=b(p(i));
    for j=i+1:1:n
        q=q-a(p(i),j)*x(j);
    end
    x(i)=q/a(p(i),i);
end
```

Contents

Factorizations $PA = LU$

- Let $p_1, p_2, \ldots, p_n$ be the indices of the rows in the order in which they become pivoting rows. Let $A^{(1)} = A$, and define $A^{(2)}, A^{(3)}, \ldots, A^{(n)}$ recursively by the formula

$$a_{p_i j}^{(k+1)} = \begin{cases} a_{p_i j}^{(k)} & \text{if } i \leqslant k \text{ or } i > k > j \\ a_{p_i j}^{(k)} - (a_{p_i k}^{(k)} / a_{p_k k}^{(k)}) a_{p_k j}^{(k)} & \text{if } i > k \text{ and } j > k \\ a_{p_i k}^{(k)} / a_{p_k k}^{(k)} & \text{if } i > k \text{ and } j = k \end{cases}$$

THEOREM 2 (Theorem on LU Factorization of PA)

Define a permutation matrix P whose elements are $P_{ij} = \delta_{p_i j}$. Define an upper triangular matrix U whose elements are $u_{ij} = a_{p_i j}^{(n)}$ if $j \geqslant i$. Define a unit lower triangular matrix L whose elements are $l_{ij} = a_{p_i j}^{(n)}$ if $j < i$. Then $PA = LU$.

Factorizations $PA = LU$

Proof.

From the recursive formula, we have

$$\begin{aligned} u_{kj} &= a_{p_k j}^{(n)} = a_{p_k j}^{(k)} \qquad (j \geqslant k) \\ l_{ik} &= a_{p_i k}^{(n)} = a_{p_i k}^{(k+1)} = a_{p_i k}^{(k)} / a_{p_k k}^{(k)} \qquad (i \geqslant k) \end{aligned}$$

Because the row p_k in $A^{(n)}$ became fixed in step k, and column k in $A^{(n)}$ became fixed in step $k+1$. Thus

$$a_{p_k j}^{(n)} = a_{p_k j}^{(k)} \qquad a_{p_i k}^{(n)} = a_{p_i k}^{(k+1)}$$

$\square$

Factorizations $PA = LU$

Proof.

Suppose that $i \leqslant j$.

$$
\begin{aligned}
(LU)_{ij} &= \sum_{k=1}^{i} l_{ik} u_{kj} \\
&= \sum_{k=1}^{i-1} (a_{p_i k}^{(k)} / a_{p_k k}^{(k)}) a_{p_k j}^{(k)} + l_{ii} a_{p_i j}^{(i)} \\
&= \sum_{k=1}^{i-1} (a_{p_i j}^{(k)} - a_{p_i j}^{(k+1)}) + a_{p_i j}^{(i)} \\
&= a_{p_i j}^{(1)} = a_{p_i j}
\end{aligned}
$$

□

Factorizations $PA = LU$

Proof.

If $i > j$, then

$$
\begin{aligned}
(LU)_{ij} &= \sum_{k=1}^{j} l_{ik} u_{kj} \\
&= \sum_{k=1}^{j-1} (a_{p_i k}^{(k)} / a_{p_k k}^{(k)}) a_{p_k j}^{(k)}) + (a_{p_i j}^{(j)} / a_{p_j j}^{(j)}) \\
&= \sum_{k=1}^{j-1} (a_{p_i j}^{(k)} - a_{p_i j}^{(k+1)}) + a_{p_i j}^{(j)} \\
&= a_{p_i j}^{(1)} = a_{p_i j}
\end{aligned}
$$

$\square$

Factorizations $PA = LU$

Proof.

And,

$$(PA)_{ij} = \sum_{k=1}^{n} P_{ik}a_{kj} = \sum_{k=1}^{n} \delta_{p_i k}a_{kj} = a_{p_i j}$$

So for all pairs (i, j), that

$$(PA)_{ij} = (LU)_{ij}$$

$\square$

Factorizations $PA = LU$

THEOREM 3 (Theorem on Solving $PA = LU$)

If the factorization $PA = LU$ is produced from the Gaussian algorithm with scaled row pivoting, then the solution of $Ax = b$ is obtained by first solving $Lz = Pb$ and then solving $Ux = z$. Similarly, the solution of $y^T A = c^T$ is obtained by solving $U^T z = c$ and then $L^T P y = z$.

Pseudocode in terms of L and U

- input $n, (l_{ij}), (u_{ij}), (b_i), (p_i)$
  for $i = 1$ to n do
  $z_i \leftarrow b_{p_i} - \sum_{j=1}^{i-1} l_{ij} z_j$
  end do
  for $i = n$ to 1 step -1 do
  $x_i \leftarrow (z_i - \sum_{j=i+1}^{n} u_{ij} x_j) / u_{ii}$
  end do
  output (x_i)

Pseudocode in terms of L and U

- input $n, (a_{ij}), (b_i), (p_i)$
  for $i = 1$ to n do
  $z_i \leftarrow b_{p_i} - \sum_{j=1}^{i-1} a_{p_i j} z_j$
  end do
  for $i = n$ to 1 step -1 do
  $x_i \leftarrow (z_i - \sum_{j=i+1}^{n} a_{p_i j} x_j)/a_{p_i i}$
  end do
  output (x_i)

Pseudocode for $y^T A = c^T$

- input $n, (a_{ij}), (c_i), (p_i)$
  for $j = 1$ to n do
  $z_j \leftarrow (c_i - \sum_{i=1}^{j-1} a_{p_i j} z_i) / a_{p_j j}$
  end do
  for $j = n$ to 1 step -1 do
  $y_{p_j} \leftarrow z_j - \sum_{i=j+1}^{n} a_{p_i j} y_{p_i}$
  end do
  output (x_i)

Contents

Operation Counts

THEOREM 4 (Theorem on Long Operations)

If Gaussian elimination is used with scaled row pivoting, then the solution of the system $Ax = b$, with fixed A, and m different vectors b, involves approximately

$$\frac{1}{3}n^3 + (\frac{1}{2} + m)n^2$$

long operations (multiplications or divisions).

Factorizations $PA = LU$

Proof.

Consider the first major step.

- Define p_1 involves n divisions (n ops).
- For each of the $n-1$ rows, a multiplier is computed (1 op), then a multiple of row p_1 is subtracted from p_i for $2 \leqslant i \leqslant n$. So the multiplier and the elimination process consume n ops per row.
- Since $n-1$ rows are processed, we have $n(n-1)$ ops. So the total is n^2 ops.

For the entire calculation, the factorization requires

$$n^2 + (n-1)^2 + \ldots + 2^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n - 1 \approx \frac{1}{3}n^3 + \frac{1}{2}n^2$$

long operations. □

Operation Counts

Proof.

- In updating the b, there are $n-1$ steps. In the first, there are $n-1$ long operations. In the second, there are $n-2$, and so on. The total is

$$(n-1)+(n-2)+\ldots+1=\frac{1}{2}n^2-\frac{1}{2}n$$

- In the back substitution, there is one long operation in the first step (computing x_n). Then there are successively 2, 3,..., n long operations. The total is

$$1+2+\ldots+n=\frac{1}{2}n^2+\frac{1}{2}n$$

The grand total is n^2.

Contents

Diagonally Dominant Matrices

The property of **diagonally dominant matrices** is expressed by the inequality $|a_{ii}| > \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}| \qquad (1 \leqslant i \leqslant n)$.

It has the property that Gaussian elimination without pivoting can be safely used.

THEOREM 5 (Theorem on Preserving Diagonal Dominance)

Gaussian elimination without pivoting preserves the diagonal dominance of a matrix.

Proof.

It suffices to consider the first step in Gaussian elimination, so we have to prove that for $i = 2, 3, ..., n$,
$|a_{ii}^{(2)}| > \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}^{(2)}|$
In terms of A, this means $|a_{ii} - (a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij} - (a_{i1}/a_{11})a_{1j}|$
It suffices to prove the stronger inequality
$|a_{ii}| - |(a_{i1}/a_{11})a_{1i}| > \sum_{\substack{j=2 \\ j\neq i}}^{n} \{|a_{ij}| + |(a_{i1}/a_{11})a_{1j}|\}$
An equivalent inequality is $|a_{ii}| - \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}| > \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|$
Form the diagonal dominance in the ith row, we know that
$|a_{ii}| - \sum_{\substack{j=2 \\ j\neq i}}^{n} |a_{ij}| > |a_{i1}|$
Hence, it suffices to prove that $|a_{i1}| \geqslant \sum_{j=2}^{n} |(a_{i1}/a_{11})a_{1j}|$
This is true because of the diagonal dominance in row 1:
$|a_{11}| > \sum_{j=2}^{n} |a_{1j}| \Longrightarrow 1 > \sum_{j=2}^{n} |a_{1j}/a_{11}|$ □

Diagonally Dominant Matrices

COROLLARY 1 (First Corollary on Diagonally Dominant Matrix)

Every diagonally dominant matrix is nonsingular and has an LU-factorization.

COROLLARY 2 (Second Corollary on Diagonally Dominant Matrix)

If the scaled row pivoting version of Gaussian elimination recomputes the scale array after each major step and is applied to a diagonally dominant matrix, then the pivots will be the natural ones: $1, 2, \ldots, n$*. Hence, the work of choosing the pivots can be omitted in this case.*

Diagonally Dominant Matrices

Proof.

By Theorem 5, we only need to prove that the first pivot chosen in the algorithm is 1. So we should prove $|a_{11}|/s_1 > |a_{i1}|/s_i \quad (2 \leqslant i \leqslant n)$.
By the diagonal dominance, $|a_{ii}| = max_j |a_{ij}| = s_i$ for all i.
Hence, $|a_{11}|/s_1 = 1$.
For $i \geqslant 2$, We have $|a_{i1}| \leqslant \sum_{\substack{j=2 \\ j \neq i}}^{n} |a_{ij}| < |a_{ii}| = s_i$
Thus, $|a_{i1}|/s_i < 1$. $\square$

Contents

Tridiagonal System

- A square matrix $A = (a_{ij})$ is said to be tridiagonal if $a_{ij} = 0$ for all pairs (i, j) that satisfy $|i - j| > 1$.

$$\begin{bmatrix} d_1 & c_1 & & & & & \\ a_1 & d_2 & c_2 & & & & \\ & a_2 & d_3 & c_3 & & & \\ & & a_3 & d_4 & c_4 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-2} & d_{n-1} & c_{n-1} \\ & & & & & a_{n-1} & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Tridiagonal System

- Step 1 consists of these replacements:

$$d_2 \leftarrow d_2 - (a_1/d_1)c_1$$
$$b_2 \leftarrow b_2 - (a_1/d_1)b_1$$

In the back substitution phase, the first step is

$$x_n \leftarrow b_n/d_n$$

The next step is

$$x_{n-1} \leftarrow (b_{n-1} - c_{n-1}x_n)/d_{n-1}$$

Tridiagonal System

input $n, (a_i), (b_i), (c_i), (d_i)$
for $i = 2$ to n do
  $d_i \leftarrow d_i - (a_{i-1}/d_{i-1})c_{i-1}$
  $b_i \leftarrow b_i - (a_{i-1}/d_{i-1})b_{i-1}$
end do
$x_n \leftarrow b_n/d_n$
for $i = n - 1$ to 1 step -1 do
  $x_i \leftarrow (b_i - c_i x_{i+1})/d_i$
end do
output(x_i)

Tridiagonal System

```
function x=tri(a,b,c,d)
n=length(d);
for i=2:n
    d(i)=d(i)-(a(i-1)/d(i-1))*c(i-1);
    b(i)=b(i)-(a(i-1)/d(i-1))*b(i-1);
end
x(n)=b(n)/d(n);
for i=n-1:-1:1
    x(i)=(b(i)-c(i)*x(i+1))/d(i);
end
```