Summary of last session

-Chapter 4. Solving Systems of Linear Equations

- 4.0. Introduction

- 4.1. Matrix Algebra:

  Basic Concepts, Elementary Operation, Equivalent System, Inverse Matrix.

- 4.2. LU Factorization:

  Easy to Solve System, LU Factorization, LDU Factorization, Doolittle Factorization, Crout’s Factorization, Cholesky Factorization.

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

Basic Gaussian Elimination :

Here is a simple system of four equations in four unknowns that will be used to illustrate the Gaussian algorithm :

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 34 \\ 27 \\ -38 \end{bmatrix} \qquad (1)$$

In the first step of the process, we subtract 2 times the first equation from second, and then we subtract 1/2 times the first equation from third. Finally, we subtract -1 times the first equation from the fourth. The numbers 2, 1/2 and -1 are called the multipliers for the first step in the elimination process.

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

The number 6, used as the divisor in the forming each of these multipliers, is called the **pivot element** for this step. After the first step has been completed, the system become

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 21 \\ -26 \end{bmatrix} \qquad (2)$$

Although the first row was used in the process, it was not changed. In this first step, we refer to row 1 as the **pivot row**. In the next step of the process, row 2 is used as the pivot row and -4 is the pivot element. We subtract 3 times the second row from the third, and then $-1/2$ times the second row is subtracted from the fourth.

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

The multipliers for this step are 3 and $-1/2$. The result is

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -21 \end{bmatrix} \qquad (3)$$

The final step consists of subtracting 2 times the third row from the fourth so that 2 is both the multiplier and the pivot element. The resulting system is

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -3 \end{bmatrix} \qquad (4)$$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

This system is upper triangular and **equivalent** to the origional system in the sence that the solutions of the two systems are the same. The final system is easily solved by the fourth row and working backward up the rows. The solution is

$$x = \begin{bmatrix} 1 \\ -3 \\ -2 \\ 1 \end{bmatrix}$$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

The multipliers used in transforming the system can be exhibited in a unit triangular matrix $L = (l_{ij})$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 1/2 & 3 & 1 & 0 \\ -1 & -1/2 & 2 & 1 \end{bmatrix} \tag{5}$$

Notice that each multiplier is written in the location corresponding to the 0 entry in the matrix it was reasonable for creating. The coefficient matrix of the final system is an upper triangular matrix

$$U = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \tag{6}$$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

These two matrices give the **LU**-factorization of **A**, where **A** is the coefficient matrix of the origional system. Thus,

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \qquad (7)$$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

It is not hard to see why this must be true. If we know how **U** was obtained from **A**, then by reversing the process we can get **A** from **U**. If we denote the rows of $\mathbf{A}$, $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\mathbf{A}_4$ and rows of **U**, $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{U}_3$, $\mathbf{U}_4$, then the elimination process gives us, for example,

$\mathbf{U}_2 = \mathbf{A}_2 - 2\mathbf{A}_1.$ Hence, $\mathbf{A}_2 = 2\mathbf{A}_1 + \mathbf{U}_2 = 2\mathbf{U}_1 + \mathbf{U}_2.$

The coefficients 2 and 1 occupy the second row of **L**.

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \\ \mathbf{U}_4 \end{bmatrix}$$

i.e. $l_{21} = 2,\ l_{22} = 1.$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

Similarly, $\mathbf{U}_3 = (\mathbf{A}_3 - 1/2\mathbf{A}_1) - 3\mathbf{U}_2$, and finally we have $\mathbf{A}_3 = 1/2\mathbf{A}_1 + 3\mathbf{U}_2 + \mathbf{U}_3 = 1/2\mathbf{U}_1 + 3\mathbf{U}_2 + \mathbf{U}_3$

The coefficients 1/2, 3, and 1 must therefore occupy the third row of $\mathbf{L}$ i.e. $l_{31} = 1/2$, $l_{32} = 3$, and $l_{33} = 1$, and correspondly the fourth row of $\mathbf{L}$ is $l_{41} = -1$, $l_{42} = -1/2$, $l_{43} = 2$ and $l_{44} = 1$.

To describe formally the progress of the Gaussian algorithm, we interpret it as a succession of $n-1$ major steps resulting in a sequence of matrices as follows:

$$\mathbf{A} = \mathbf{A}^{(1)} \to \mathbf{A}^{(2)} \to \cdots \to \mathbf{A}^{(n)}$$

4.3 Pivoting and Constructing Algorithm

- Basic Gaussian Elimination

At the conclusion of step k-1, the matrix $\mathbf{A}^{(k)}$ will have been constructed; its appearance is shown in the following display in which lines are placed around the kth row and just before the kth column to illustrate the structure produced by the elimination process:

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

4.3 Pivoting and Constructing Algorithm

Next we describe how $\mathbf{A}^{(k+1)}$ is obtained from $\mathbf{A}^{(k)}$. To produce 0's in column k below the pivot element $a_{kk}^{(k)}$, we subtract multiples of row k from the rows beneath it. Rows $1, 2, \cdots, k$ are not alerted. The formula is therefore

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & \text{if } i \le k \\ a_{ij}^{(k)} - \left(a_{ik}^{(k)} / a_{kk}^{(k)}\right) a_{kj}^{(k)} & \text{if } i \ge k+1 \text{ and } j \ge k+1 \\ 0 & \text{if } i \ge k+1 \text{ and } j \le k \end{cases} \quad (8)$$

4.3 Pivoting and Constructing Algorithm

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

$$l_{ik} = \begin{cases} a_{ik}{}^{(\mathrm{k})} \big/ a_{kk}{}^{(\mathrm{k})} & \text{if } i \geq k+1 \\ 1 & \text{if } i = k \\ 0 & \text{if } i \leq k\text{-}1 \end{cases} \qquad (9)$$

Here $\mathbf{A} = \mathbf{LU}$ is the standard Gaussian factorization of matrix $\mathbf{A}$ with $\mathbf{L}$ unit lower triangular and $\mathbf{U}$ upper triangular.

It should be clear from (8) and (9) that this entire elimination process will break down if any of the pivot elements is 0.

4.3 Pivoting and Constructing Algorithm*

- Theorem on Nonzero Pivots

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

Proof : Observe that $a_{ij}^{(k+1)} = a_{ij}^{(k)}$ if $i \le k$ or $j \le k-1$. Next note that $u_{kj} = a_{kj}^{(n)} = a_{kj}^{(k)}$. Finally, note that $l_{ik} = 0$ if $k > i$, and $u_{kj} = 0$, if $k > j$. Now let $i \le j$. Using these facts, we have

$$(\mathbf{LU})_{ij} = \sum_{k=1}^{n} l_{ik} u_{kj} = \sum_{k=1}^{i} l_{ik} u_{kj}^{(k)} = \sum_{k=1}^{i} l_{ik} a_{kj}^{(k)}$$

$$= \sum_{k=1}^{i-1} l_{ik} a_{kj}^{(k)} + l_{ii} a_{ij}^{(i)} = \sum_{k=1}^{i-1} \left(a_{ik}^{(k)} / a_{kk}^{(k)} \right) a_{kj}^{(k)} + a_{ij}^{(i)}$$

$$= \sum_{k=1}^{i-1} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) + a_{ij}^{(i)}$$

$$= a_{ij}^{(1)} - a_{ij}^{(2)} + a_{ij}^{(2)} - a_{ij}^{(3)} + \cdots + a_{ij}^{(i-1)} - a_{ij}^{(i)} + a_{ij}^{(i)}$$

$$= a_{ij}^{(1)} = a_{ij}$$

4.3 Pivoting and Constructing Algorithm*

- Theorem on Nonzero Pivots

Similarly, if $i > j$, then

$$(\mathbf{LU})_{ij} = \sum_{k=1}^{j} l_{ik} u_{kj} = \sum_{k=1}^{j} l_{ik} a_{kj}^{(k)}$$

$$= \sum_{k=1}^{j} (a_{ij}^{(k)} - a_{ij}^{(k+1)}) = a_{ij}^{(1)} - a_{ij}^{(j+1)} = a_{ij}^{(1)} = a_{ij}$$

where $a_{ij}^{(j+1)} = 0$ since $a_{ij}^{(k)} = 0$ if $i \geq j+1$ and $k \geq j+1$.

4.3 Pivoting and Constructing Algorithm

- *LU*-factorization

Example: Using the **LU** factorization to solve the follwoing equation

$$\begin{bmatrix} 2 & 1 & 5 \\ 4 & 1 & 12 \\ -2 & -4 & 5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 27 \\ 12 \end{bmatrix}$$

Solution: Using the formula (8) and (9) we can calculate

$k=1$: The first row of **U** and first column of **L**

$$a_{11}^{(2)} = a_{11}^{(1)} = 2,\ a_{12}^{(2)} = a_{12}^{(1)} = 1,\ a_{13}^{(2)} = a_{13}^{(1)} = 5$$

$$a_{22}^{(2)} = a_{22}^{(1)} - (a_{21}^{(1)} / a_{11}^{(1)})a_{12}^{(1)} = 1-(4/2)\times 1 = -1$$

$$a_{23}^{(2)} = a_{23}^{(1)} - (a_{21}^{(1)} / a_{11}^{(1)})a_{13}^{(1)} = 12-(4/2)\times 5 = 2$$

$$a_{32}^{(2)} = a_{32}^{(1)} - (a_{31}^{(1)} / a_{11}^{(1)})a_{12}^{(1)} = -4-(-2/2)\times 1 = -3$$

$$a_{33}^{(2)} = a_{33}^{(1)} - (a_{31}^{(1)} / a_{11}^{(1)})a_{13}^{(1)} = 5-(-2/2)\times 5 = 10$$

$$l_{11}=1,\ l_{21}= a_{21}^{(1)} / a_{11}^{(1)} = 2,\ l_{31} = a_{31}^{(1)} / a_{11}^{(1)} = -1$$

4.3 Pivoting and Constructing Algorithm

- *LU*-factorization

$k=2$: The second row of **U** and second column of **L**

$$a_{33}^{(3)}=a_{33}^{(2)}-(a_{32}^{(2)}/a_{22}^{(2)})a_{23}^{(2)}=10-(-3/(-1))\times 2=4$$

$$l_{32}=a_{32}^{(2)}/a_{22}^{(2)}=-3/(-1)=3$$

$$\mathbf{A}^{(3)}=\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} \\ 0 & 0 & a_{33}^{(3)} \end{bmatrix}=\begin{bmatrix} 2 & 1 & 5 \\ 0 & -1 & 2 \\ 0 & 0 & 4 \end{bmatrix}$$

$$\mathbf{L}=\begin{bmatrix} 1 & 0 & 0 \\ a_{21}^{(1)}/a_{11}^{(1)} & 1 & 0 \\ a_{31}^{(1)}/a_{11}^{(1)} & a_{32}^{(2)}/a_{22}^{(2)} & 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}$$

$$\mathbf{U}=\mathbf{A}^{(3)}=\begin{bmatrix} 2 & 1 & 5 \\ 0 & -1 & 2 \\ 0 & 0 & 4 \end{bmatrix}$$

4.3 Pivoting and Constructing Algorithm

- *LU*-factorization

Therefore $\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 5 \\ 0 & -1 & 2 \\ 0 & 0 & 4 \end{bmatrix}$

Solve $\mathbf{Lz} = \mathbf{b},$ $\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 27 \\ 12 \end{bmatrix}$

$$z_1 = 11, \quad z_2 = 5, \quad z_3 = 8$$

solve $\mathbf{Ux} = \mathbf{z},$ $\begin{bmatrix} 2 & 1 & 5 \\ 0 & -1 & 2 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ 8 \end{bmatrix}$

$$x_3 = 2, \quad x_2 = -1, \quad x_1 = 1$$

4.3 Pivoting and Constructing Algorithm

- Pivoting

The Gaussian algorithm, in the simple form just described, is not satisfactory since it fails on systems that are in fact easy to solve. To illustrate this, we consider some elementry examples. The first is

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad (10)$$

The simple version of the algorithm fails because there is no way of adding a multiple of the first equation to the second in order to obtain a 0-coefficient of x_1 in the second equation.

4.3 Pivoting and Constructing Algorithm

- Pivoting

The difficulty just encountered will persist for the following system, in which ε is a small number different from zero:

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{11}$$

When applied to (11), the Gaussian algorithm will produce this upper triangular system.

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1-\varepsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2-\varepsilon^{-1} \end{bmatrix} \tag{12}$$

4.3 Pivoting and Constructing Algorithm
- Pivoting

The solution is

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

In the computer, if ε is small enough, $2-\varepsilon^{-1}$ will be computed to be the same as $-\varepsilon^{-1}$. Likewise, the denomenator $1 - \varepsilon^{-1}$ will be computed to be the same as $-\varepsilon^{-1}$. Hence, under these circumstances, x_2 will be computed as 1, and x_1 will be computed as 0. Since the correct solution is

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

the computed solution is accurate for x_2 but is extremely inaccurate for x_1!

4.3 Pivoting and Constructing Algorithm

- Pivoting

The following will show that it is not actually the smallness of the coefficient a_{11} that is causing the trouble. Rather, it is the smallness of a_{11} relative to the other elements in its row. Consider the following system, which is equivalent to System (11):

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2 \end{bmatrix} \qquad (14)$$

The simple Gaussian algorithm produces

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 0 & 1-\varepsilon^{-1} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2-\varepsilon^{-1} \end{bmatrix} \qquad (15)$$

The solution of (15) is

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

4.3 Pivoting and Constructing Algorithm

- Pivoting

Again for small ε, x_2 will be computed as 1 and x_1 will be computed as 0, which is, as before, wrong!

The difficulties in these examples will disappear if the order of the equations is changed.

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

Gaussian elimination applied to this system produces

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

and the solution is

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

4.3 Pivoting and Constructing Algorithm

- Pivot Rows

The conclusion to be drawn from these elementry examples is that good algorithms must incorporate the interchanging of equations in a system when the circumstances require it. For reasons of economy in computing, we prefer not to move the rows of the matrix around the computer's memory. Instead, we simply choose the **pivot rows** in a logical manner. Suppose that instead of using the rows in the order 1, 2, $\cdots$, $n-1$ as pivot rows, we use rows $p_1, p_2, \cdots, p_{n-1}$. Then in the first step, multiples of row p_1 will be subtracted from the other rows. If we introduce p_n so that $(p_1, p_2, \cdots, p_n)$ is a permutation of $(1, 2, \cdots, n)$, then row p_n will not occur as a pivot row, but we can say that multiples of row p_1 will be subtracted from rows $p_2, p_3, \cdots, p_n$. In the next step, multiples of row p_2 will be subtracted from rows $p_3, p_4, \cdots, p_n$; and so on.

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

We now describe an algorithm called **Gaussian elimination with scaled row pivoting**, for solving an $n \times n$ system

$$Ax = b$$

The algorithm consists of two parts: a **factorization phase** (also called **forward elimination**) and a **solution phase** (involving **updating** and **back substitution**). The factorization phase is applied to A only and is designed to produce the LU-decomposition of PA, where P is a permutation matrix derived from the permutation array p. (PA is obtained from A by permuting its rows). The permuted linear system is

$$PAx = Pb$$

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

The factorization $PA = LU$ is obtained from a modified Gaussian elimination algorithm. In the solution phase, we consider two equations $Lz = Pb$ and $Ux = z$. First, the right hand side b is rearranged according to P and the results are stored back in b; that is, $b \leftarrow Pb$. Next $Lz = b$ is solved for z and the results stored back in b; that is $b \leftarrow L^{-1}b$. Since L is unit lower triangular, this amounts to a forward substitution process. This process is called **updating** b. Then back substitution is used to solve $Ux = b$ for $x_n, x_{n-1}, \ldots, x_1$.

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

In the factorization phase, we begin by computing the **scale** of each row. We put

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

and we do not arbitrarily subtract multiples of row 1 from the other rows. Rather, we select the row for which $|a_{i1}|/s_i$ is largest as the **pivot row**. The index thus chosen is denoted by p_1 and becomes the first entry in the permutation array. Thus, $|a_{p_1 1}|/s_{p_1} \ge |a_{i1}|/s_i$ for $1 \le i \le n$. Once p_1 has been determined, we subtract appropriate multiples of row p_1 from the other rows in order to produce 0's in the first column of A. Of course, row p_1 will remain unchanged throughout the remainder of the factorization process.

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

To keep track of the indices p_i that arise, we begin by setting the permutation vector $(p_1, p_2, p_3, K\ , p_n)$ to $(1, 2, K\ , n)$. Then we select an index j for which $\left|a_{p_j 1}\right| / s_{p_j}$ is maximal and interchange p_1 with p_j in the permutation array p. The actual elimination step involves subtracting $\left(a_{p_i 1} / a_{p_1 1}\right)$ times row p_1 from row p_i for $2 \leq i \leq n$.

To describe the genereal process, suppose that we are ready to create 0's in column k. We scan the numbers $\left|a_{p_i k}\right| / s_{p_i}$ for $k \leq i \leq n$, looking for a maximal entry. If j is the index of the first of the largest of these ratios, we interchange p_k with p_j in the array p and then subtract $(a_{p_i k} / a_{p_k k})$ times row p_k from row p_i for $k+1 \leq i \leq n$.

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

Here is how this works on the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix}$$

At the beginning, $p=(1,2,3)$, and $s=(6,8,3)$. To select the first pivot row, we look at the ratio $\{2/6,\ 1/8,\ 3/3\}$. The largest corresponds to $j=3$, and row 3 is to be the first pivot row. So we interchange p_1 with p_3, obtaining $p=(3,2,1)$. Multiples of row 3 are now subtracted from rows 2 and 1 to produce 0's in the first column. The result is

$$\left(A^{(2)} = \begin{bmatrix} 0 & 13/3 & -20/3 \\ 0 & -16/3 & 23/3 \\ 3 & -2 & 1 \end{bmatrix} \right) A^{(2)} = \begin{bmatrix} 2/3 & 13/3 & -20/3 \\ 1/3 & -16/3 & 23/3 \\ 3 & -2 & 1 \end{bmatrix}$$

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

The entries in location a_{11} and a_{21} are the multipliers. In the next step, the selection of a pivot row is made on the basis of the numbers $|a_{p_2 2}|/s_{p_2}$ and $|a_{p_3 2}|/s_{p_3}$. The first of these ratios is (16/3)/ (23/3) and the second is (13/3)/ (20/3). So $j = 2$, and we choose row p_2 as a pivot row. Then a multiple of row p_2 is subtracted from row p_3. The result is $p = (3,2,1)$ and

$$(A^{(3)} = \begin{bmatrix} 0 & 0 & -7/16 \\ 0 & -16/3 & 23/3 \\ 3 & -2 & 1 \end{bmatrix})\ A^{(3)} = \begin{bmatrix} 2/3 & -13/16 & -7/16 \\ 1/3 & -16/3 & 23/3 \\ 3 & -2 & 1 \end{bmatrix}$$

The final multiplier is stored in location a_{12}.

4.3 Pivoting and constructing Algorithm

- Gaussian Elimination with Scaled Row Pivoting

If the rows of the original matrix A were interchanged according to the final permutation array p, then we would have an LU-factorization of A such as

$$PA = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ 2/3 & -13/16 & 1 \end{bmatrix} \begin{bmatrix} 3 & -2 & 1 \\ 0 & -16/3 & 23/3 \\ 0 & 0 & -7/16 \end{bmatrix} = \begin{bmatrix} 3 & -2 & 1 \\ 1 & -6 & 8 \\ 2 & 3 & -6 \end{bmatrix}$$

where

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix}$$

The permutation matrix P is obtained from the permutation array p by setting $(P_{\text{ij}}) = \delta_{p_i j}$. In other words, P is formed by permuting the rows of identity matrix I accoring to the entries in p.

4.3 Pivoting and Constructing Algorithm

- Factorizations *PA=LU*

Factorizations *PA=LU*

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

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

4.3 Pivoting and Constructing Algorithm

- Factorizations *PA=LU*

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 $j \geq i$. Define a unit lower triangular matrix L whose elements are

$l_{ij} = a_{p_i j}{}^{(n)}$ if $j < i$. Then $PA = LU$.

Proof (as an exercise).

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 $Ux = z$.

Proof (as an exercise).

4.3 Pivoting and Constructing Algorithm

- Diagonally Dominant Matrices

Sometimes a system of equations has the property that Gaussian elimination without pivoting can be safely used. One class of matrices for which this is true is diagonally dominant matrices. This property is expressed by the inequility

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad (1 \le i \le n)$$

Please note that the property of diagonally dominant is invariant under Gaussian elimination. An example of a diagonally dominant matrix is

$$\begin{bmatrix} 4 & -1 & 0 & -1 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}$$

Such matrices arise naturally in applications involving the discretization by finite differences of partial differential equations. Also, they are seen in study of splines and many other areas.

4.3 Pivoting and Constructing Algorithm

- Tridiagonal System

In applications, systems of equations often arise in which the coefficient matrix has a special structure. It is usually better to solve these systems using tailor-made algorithms that exploit the special structure. We consider one example of this, the tridiagonal system. A square matrix $A=(a_{ij})$ is said to be tridiagonal if $a_{ij}=0$ for all pairs *(i, j)* that satisfy $|i-j|>1$. Thus, in the *i*th row, only $a_{i,i-1}$, a_{ii}, and $a_{i,i+1}$ can be different from 0. Three vectors can be used to store the nonzero elements, and we arrange the notation such that

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

4.3 Pivoting and Constructing Algorithm

- Tridiagonal System

The tridiagonal matrix is expecially important since the boundary value problem of ordinary differential equations usually ends up with solving linear equation system with a tridiagonal matrix. The most effective method of solving the tridiagonal matrix is to do the Doolittle or the Crout factorization for the matrix $A = LU$. L and U can be written as

$$L = \begin{bmatrix} 1 & & & \\ l_2 & 1 & & \\ & \ddots & \ddots & \\ & & l_n & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} r_1 & c_1 & & \\ & r_2 & c_2 & \\ & & \ddots & c_{n\text{-}1} \\ & & & r_n \end{bmatrix}$$

and the procedure of solution is then as follows

step -1: we carry out Doolittle LU factorization, $r_1 = d_1$,

for $i = 2, \cdots, n$, we then calculate

$$l_i = a_{i-1} / r_{i-1},$$

$$r_i = \ d_i - l_i \times c_{i-1};$$

4.3 Pivoting and Constructing Algorithm

- Tridiagonal System

$step-2:$ we solve $\mathbf{Lz}=\mathbf{b}$

$$z_1 = b_1,$$

for $i = 2, \cdots, n,$ we make calculation

$$z_i = b_i - l_i \times z_{i-1};$$

$Step-3:$ we solve $\mathbf{Ux}=\mathbf{z}$

$$x_n = z_n / r_n,$$

for $i = n-1, \cdots, 1,$ we make calculation

$$x_i = (z_i - c_i \times x_{i+1}) / r_i;$$

It is easy to see that the **x** and **z** can be stored in the same set of memory unit, therefore only $4n$ unit is needed for this calculation, and $5n$ multiplications.

Scientific calculation often needs to deal with linear equation system with very high order, such as several thousands and several handred thousands. fortunately, they often are sparse matrices.

Exercises

$Ex1$. Solve the following linear equation system twice. First, using Gaussian elimination and give the factorization A = LU. Second, use Gaussian elimination with scaled row pivoting and determine the factorization of the form PA = LU.

$$\begin{pmatrix} 1 & 6 & 0 \\ 2 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

$Ex2$. Find the LU factorization of A using the Doolittle factorization method.

$$\mathrm{A} = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 1 & 4 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$