Lab5 LU Factorization Algorithm

Sheng Zhang
2016310200210
2nd class of information and computing science

1 Lab purpose

1. Master the theory of LU factorization algorithm.
2. Implement LU factorization utilizing C language.

2 Lab preparation

Read section 5.3 of the textbook «numerical analysis»

3 Lab requirements

Solve given linear equations by utilzing LU factorization algorithm.

3.1 Definition of function interface

```
bool Direct( int n, double a[][MAX_SIZE], double b[] )
```

In the function interface of Direct, n denotes the dimension of matrix a, MAX-SIZE denotes the max dimension, and b denotes the constant vector. The solution of the equation is required to be stored in b. The return value of Direct function is Boolean type, when it can be solved, it's TRUE. Otherwise, it's FALSE.
The original code is as follow:

```
#include<stdio.h>
#include<math.h>
#define MAX_SIZE 100
#define ZERO 0.000000001
bool Direct( int n, double a[][MAX_SIZE], double b[] );
int main()
{
  int n, i, j;
  double a[MAX_SIZE][MAX_SIZE], b[MAX_SIZE];

  while ( scanf("%d", &n) != EOF ) {
    for ( i=0; i<n; i++ ) {
      for ( j=0; j<n; j++ )
        scanf("%lf", &a[i][j]);
      scanf("%lf", &b[i]);
    }
    if ( Direct(n, a, b) ) {
      printf("Result of direct method:\n");
      for ( j=0; j<n; j++ )
        printf("%.8lf\n", b[j]);
    }
    else
      printf("Doolittle factorization failed.\n");
    printf("\n");
  }
  return 0;
}
```

3.2 Data requirements

Input:

```
6
4 -1 0 -1 0 0 0
-1 4 -1 0 -1 0 5
0 -1 4 0 0 -1 0
-1 0 0 4 -1 0 6
0 -1 0 -1 4 -1 -2
0 0 -1 0 -1 4 6

3
3 -1 0 1
3 6 0 0
3 3 0 4

3
3 -1 3 1
3 6 3 0
3 3 3 4

7
1.0 0.0 2.0 0.0 3.0 0.0 4.0 3
3.0 -1.0 0.5 8.0 2.2 1.6 0.0 8
0.0 0.0 0.0 4.5 3.2 2.0 1.0 2
2.0 3.0 5.0 0.0 0.0 0.0 2.0 4
-2.0 -3.0 1.0 1.0 0.0 0.0 3.3 1
2.5 4.5 0.0 0.0 1.0 0.0 0.0 -2
-0.5 -1.5 3.0 2.0 0.0 1.0 -1.0 5

3
1.00000000 0.50000000 0.33333333 1.0
0.50000000 0.33333333 0.25000000 1.0
0.33333333 0.25000000 0.20000000 1.0

4
1.00000000 0.50000000 0.33333333 0.25000000 1.0
0.50000000 0.33333333 0.25000000 0.20000000 1.0
0.33333333 0.25000000 0.20000000 0.16666667 1.0
0.25000000 0.20000000 0.16666667 0.14285714 1.0
```

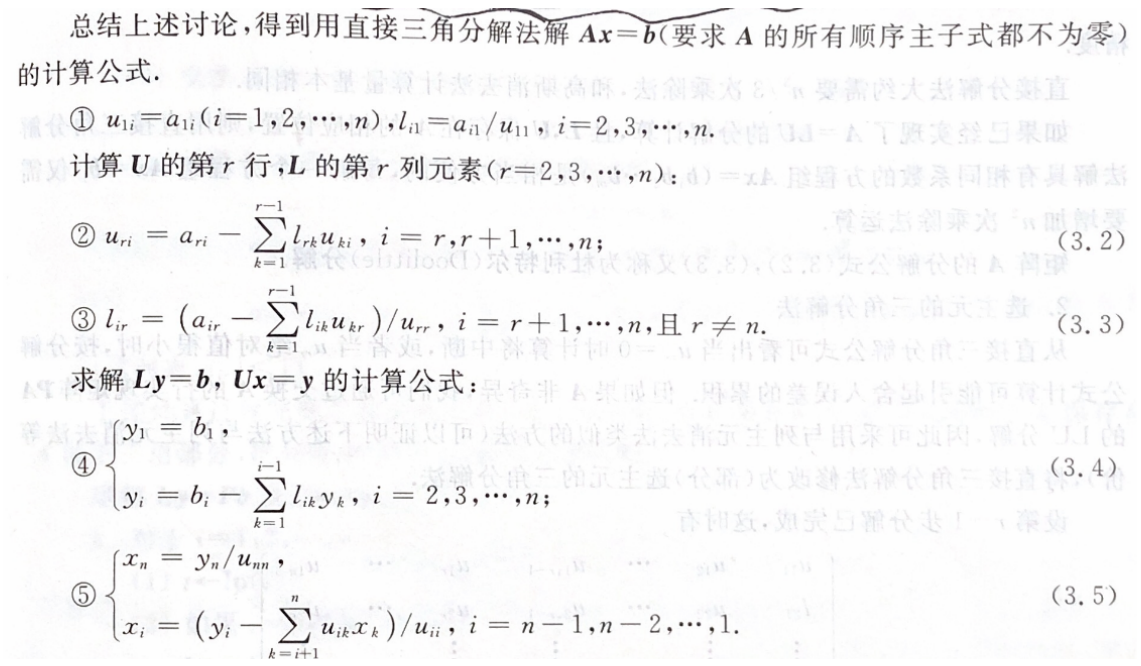
Output:

```
Result of direct method:
1.00000000
2.00000000
1.00000000
2.00000000
1.00000000
2.00000000

Doolittle factorization failed.
Doolittle factorization failed.
Doolittle factorization failed.
Result of direct method:
3.00000408
-24.00002076
30.00001920

Result of direct method:
-4.00028881
60.00328814
-180.00799473
140.00523622
```

4 Detail of LU factorization algorithm



总结上述讨论，得到用直接三角分解法解 $\boldsymbol{Ax}=\boldsymbol{b}$（要求 $\boldsymbol{A}$ 的所有顺序主子式都不为零）的计算公式.

① $u_{1i}=a_{1i}(i=1,2,\cdots,n)$，$l_{i1}=a_{i1}/u_{11}$，$i=2,3,\cdots,n$.

计算 $\boldsymbol{U}$ 的第 r 行，$\boldsymbol{L}$ 的第 r 列元素（$r=2,3,\cdots,n$）：

$$② \; u_{ri}=a_{ri}-\sum_{k=1}^{r-1} l_{rk}u_{ki}, \quad i=r,r+1,\cdots,n; \tag{3.2}$$

$$③ \; l_{ir}=\left(a_{ir}-\sum_{k=1}^{r-1} l_{ik}u_{kr}\right)/u_{rr}, \quad i=r+1,\cdots,n, \text{且 } r\neq n. \tag{3.3}$$

求解 $\boldsymbol{Ly}=\boldsymbol{b}$，$\boldsymbol{Ux}=\boldsymbol{y}$ 的计算公式：

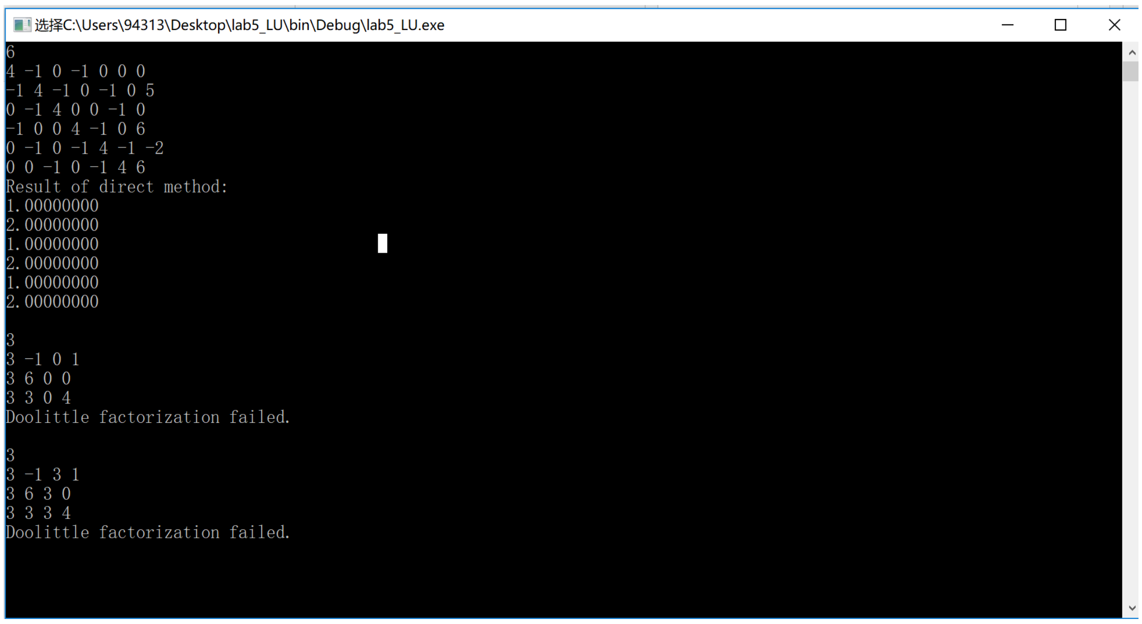
$$④ \begin{cases} y_1=b_1, \\ y_i=b_i-\sum\limits_{k=1}^{i-1} l_{ik}y_k, \quad i=2,3,\cdots,n; \end{cases} \tag{3.4}$$

$$⑤ \begin{cases} x_n=y_n/u_{nn}, \\ x_i=\left(y_i-\sum\limits_{k=i+1}^{n} u_{ik}x_k\right)/u_{ii}, \quad i=n-1,n-2,\cdots,1. \end{cases} \tag{3.5}$$

Figure 1: The algorithm flow.

5 Code analysis and thinking

As far as I am concerned, the most difficult problem in implement process is index problem. In programming, an array index starts at zero, however, a normal matrix representation starts at one. So be careful in programming is important. And the code result is shown in Figure 2.



选择C:\Users\94313\Desktop\lab5_LU\bin\Debug\lab5_LU.exe

```
6
4 -1 0 -1 0 0 0
-1 4 -1 0 -1 0 5
0 -1 4 0 0 -1 0
-1 0 0 4 -1 0 6
0 -1 0 -1 4 -1 -2
0 0 -1 0 -1 4 6
Result of direct method:
1.00000000
2.00000000
1.00000000
2.00000000
1.00000000
2.00000000

3
3 -1 0 1
3 6 0 0
3 3 0 4
Doolittle factorization failed.

3
3 -1 3 1
3 6 3 0
3 3 3 4
Doolittle factorization failed.
```

Figure 2: The result of LU algorithm.

6 Code sharing

```
#include<stdio.h>
#include<math.h>
#define MAX_SIZE 100
#define ZERO 0.000000001
typedef enum {false, true} bool;
bool Direct( int n, double a[][MAX_SIZE], double *b )
{
    double u[n][n], l[n][n];
    for(int i=0;i<n;i++)
        u[0][i] = a[0][i];
    for(int i=1;i<n;i++)
        l[i][0] = a[i][0]/u[0][0];
    for(int r=1;r<n;r++)
    {
        for(int i=r;i<n;i++)
        {
            double sum=0;
            for(int k=0;k<=r-1;k++)
                sum+=l[r][k]*u[k][i];
            u[r][i] = a[r][i] - sum;
        }
        for(int i=r+1;i<n&&r!=n-1;i++)
        {
            double sum=0;
            for(int k=0;k<=r-1;k++)
                sum+=l[i][k]*u[k][r];
            l[i][r]=(a[i][r] - sum)/u[r][r];
        }
    }
    for(int i=0;i<n;i++)
        if(u[i][i]==0)
            return false;
    double y[n],x[n];
    y[0] = b[0];
    for(int i=1;i<n;i++)
    {
        double sum=0;
        for(int k=0;k<=i-1;k++)
            sum += l[i][k]*y[k];
        y[i] = b[i] - sum;
    }
    x[n-1] = y[n-1]/u[n-1][n-1];
    for(int i=n-2;i>-1;i--)
    {
        double sum=0;
        for(int k=i+1;k<n;k++)
            sum += u[i][k]*x[k];
        x[i] = (y[i]-sum)/u[i][i];
    }
    for(int i=0;i<n;i++)
        b[i]=x[i];
    return true;
}
int main()
{
  int n, i, j;
  double a[MAX_SIZE][MAX_SIZE], b[MAX_SIZE];
  while ( scanf("%d", &n) != EOF ) {
    for ( i=0; i<n; i++ ) {
      for ( j=0; j<n; j++ )
        scanf("%lf", &a[i][j]);
      scanf("%lf", &b[i]);
    }
    if ( Direct(n, a, b) ) {
      printf("Result␣of␣direct␣method:\n");
      for ( j=0; j<n; j++ )
        printf("%.8f\n", b[j]);
    }
    else
      printf("Doolittle␣factorization␣failed.\n");
    printf("\n");
  }
  return 0;
}
```