计算方法-第二次上机作业

PB19051183 吴承泽

实验源码：

```
#include <stdio.h>
#include <math.h>


//存放矩阵系数的二维数组如下所示
double A_1[5][5] = {
                 1.0 / 9 , 1.0 / 8 , 1.0 / 7 , 1.0 / 6 , 1.0 / 5 ,
                 1.0 / 8 , 1.0 / 7 , 1.0 / 6 , 1.0 / 5 , 1.0 / 4 ,
                 1.0 / 7 , 1.0 / 6 , 1.0 / 5 , 1.0 / 4 , 1.0 / 3 ,
                 1.0 / 6 , 1.0 / 5 , 1.0 / 4 , 1.0 / 3 , 1.0 / 2 ,
                 1.0 / 5 , 1.0 / 4 , 1.0 / 3 , 1.0 / 2 , 1.0 / 1
                };


double b_1[5][1] = { 1 ,
                     1 ,
                     1 ,
                     1 ,
                     1
                    };



double A_2[4][4] = {
                     7.2 , 2.3 , -4.4 , 0.5 ,
                     1.3 , 6.3 , -3.5 , 2.8 ,
                     5.6 , 0.9 , 8.1 , -1.3 ,
                     1.5 , 0.4 , 3.7 , 5.9 ,
};

double b_2[4][1] = {
                    15.1 ,
                    1.8 ,
                    16.6 ,
                    36.9
};
    /*
        用于展示系数矩阵，Display_Matrix_A1用来展示5*5的矩阵，Display_Matrix_A2用来展示4*4的矩阵，Display_Matrix用来展示n维向量
    */
void Display_Matrix_A1(double M[][5])
{
    int i, j;
    printf("\nThe matrix is :\n");
    for(i = 0; i < 5; i++)
    {
        for(j = 0 ; j < 5; j++)
        {
```

```
            printf("%lf ", M[i][j]);
        }
        printf("\n");
    }
}


void Display_Matrix_A2(double M[][4])
{
    int i, j;
    printf("\nThe matrix is :\n");
    for(i = 0; i < 4; i++)
    {
        for(j = 0 ; j < 4; j++)
        {
            printf("%lf ", M[i][j]);
        }
        printf("\n");
    }
}


void Display_Matrix(double b[][1], int n)
{
    int i;
    printf("\nThe Solution is :\n");
    for(i = 0; i < n; i++)
    {
        printf(" %lf ", b[i][0]);
    }
    printf("\n");
}

    /*
      Gauss列主元消元法：输入为系数矩阵A，常数矩阵b,得到列主元消元后的上三角矩阵U，以及输出x矩阵
    */
int GaussEliminationWithPartialPivoting_A1()
{

    int i, j, k;
    double temp;

    for(i = 0; i < 5 ; i++)
    {
        k = i;

        for(j = i + 1; j < 5; j++)
        {
            if(fabs(A_1[k][i]) < fabs(A_1[j][i]))
                k = j;
        }

        for(j = i; j < 5; j++)
        {
            temp = A_1[i][j];
            A_1[i][j] = A_1[k][j];
            A_1[k][j] = temp;
```

```
        }

        temp = b_1[i][0];
        b_1[i][0] = b_1[k][0];
        b_1[k][0] = temp;

        for(j = i + 1; j < 5 ; j++)
        {
            double lamda = A_1[j][i] / A_1[i][i];
            for(k = i; k < 5; k++)
            {
                A_1[j][k] = A_1[j][k] - lamda * A_1[i][k];
            }
            b_1[j][0] = b_1[j][0] - lamda * b_1[i][0];
        }

    }

    Display_Matrix_A1(A_1);
    //此处输出的矩阵值为上三角矩阵的值，下面计算解

    for(i = 4; i >= 0; i--)
    {
        for(j = 4; j > i ; j--)
        {
            b_1[i][0] = b_1[i][0] - A_1[i][j] * b_1[j][0];
        }
        b_1[i][0] = b_1[i][0] / A_1[i][i];
    }

    Display_Matrix(b_1, 5);
    //此处输出为解的值

}

int GaussEliminationWithPartialPivoting_A2()
{
    int i, j, k;
    double temp;

    for(i = 0; i < 4 ; i++)
    {
        k = i;

        for(j = i + 1; j < 4; j++)
        {
            if(fabs(A_2[k][i]) < fabs(A_2[j][i]))
                k = j;
        }

        for(j = i; j < 4; j++)
        {
            temp = A_2[i][j];
            A_2[i][j] = A_2[k][j];
            A_2[k][j] = temp;
        }

        temp = b_2[i][0];
```

```
        b_2[i][0] = b_2[k][0];
        b_2[k][0] = temp;

        for(j = i + 1; j < 4 ; j++)
        {
            double lamda = A_2[j][i] / A_2[i][i];
            for(k = i; k < 4; k++)
            {
                A_2[j][k] = A_2[j][k] - lamda * A_2[i][k];
            }
            b_2[j][0] = b_2[j][0] - lamda * b_2[i][0];
        }

    }

    Display_Matrix_A2(A_2);
    //此处输出的矩阵值为上三角矩阵的值，下面计算解

    for(i = 3; i >= 0; i--)
    {
        for(j = 3; j > i ; j--)
        {
            b_2[i][0] = b_2[i][0] - A_2[i][j] * b_2[j][0];
        }
        b_2[i][0] = b_2[i][0] / A_2[i][i];
    }

    Display_Matrix(b_2, 4);
    //此处输出为解的值
}

    /*
      Doolittle分解
    */


int Doolittle_A1()
{
    int i, j, k, r;
    double temp;
    double U[5][5], L[5][5];

    for(i = 0; i < 5; i++)
    {
        for(j = 0; j < 5; j++)
        {
            L[i][j] = 0;
            U[i][j] = 0;
        }
    }
    for(i = 0; i < 5; i++)
    {
        L[i][i] = 1;
    }

    for(k = 0; k < 5; k++)
    {
        for(j = k; j < 5; j++)
```

```
        {
            temp = 0;
            for(r = 0; r < k; r++)
            {
                temp += L[k][r] * U[r][j];
            }
            U[k][j] = A_1[k][j] - temp;
        }

        for(i = k + 1; i < 5; i++)
        {
            temp = 0;
            for(r = 0; r < k; r++)
            {
                temp += L[i][r] * U[r][k];
            }
            L[i][k] = (A_1[i][k] - temp) / U[k][k];
        }
    }

    //此处计算出L、U矩阵，并打印出来
    printf("L Matrix");
    Display_Matrix_A1(L);
    printf("\n");
    printf("U Matrix");
    Display_Matrix_A1(U);
    printf("\n");

    double y[5][1], x[5][1];

    for(i = 0; i < 5; i++)
    {
        y[i][0] = 0;
        x[i][0] = 0;
    }

    for(i = 0; i < 5; i++)
    {
        temp = 0;
        for(j = 0; j < i; j++)
        {
            temp += L[i][j] * y[j][0];
        }
        y[i][0] = b_1[i][0] - temp;
    }

    for(i = 4; i >= 0; i--)
    {
        temp = 0;
        for(j = i + 1; j < 5; j++)
        {
            temp += U[i][j] * x[j][0];
        }
        x[i][0] = (y[i][0] - temp) / U[i][i];
    }

    Display_Matrix(x, 5);
    //打印答案
```

```
}



int Doolittle_A2()
{
    int i, j, k, r;
    double temp;
    double U[4][4], L[4][4];

    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            L[i][j] = 0;
            U[i][j] = 0;
        }
    }
    for(i = 0; i < 4; i++)
    {
        L[i][i] = 1;
    }

    for(k = 0; k < 4; k++)
    {
        for(j = k; j < 4; j++)
        {
            temp = 0;
            for(r = 0; r < k; r++)
            {
                temp += L[k][r] * U[r][j];
            }
            U[k][j] = A_2[k][j] - temp;
        }

        for(i = k + 1; i < 4; i++)
        {
            temp = 0;
            for(r = 0; r < k; r++)
            {
                temp += L[i][r] * U[r][k];
            }
            L[i][k] = (A_2[i][k] - temp) / U[k][k];
        }
    }

    //此处计算出L、U矩阵，并打印出来
    printf("L Matrix");
    Display_Matrix_A2(L);
    printf("\n");
    printf("U Matrix");
    Display_Matrix_A2(U);
    printf("\n");

    double y[4][1], x[4][1];

    for(i = 0; i < 4; i++)
```

```
    {
        y[i][0] = 0;
        x[i][0] = 0;
    }

    for(i = 0; i < 4; i++)
    {
        temp = 0;
        for(j = 0; j < i; j++)
        {
            temp += L[i][j] * y[j][0];
        }
        y[i][0] = b_2[i][0] - temp;
    }

    for(i = 3; i >= 0; i--)
    {
        temp = 0;
        for(j = i + 1; j < 4; j++)
        {
            temp += U[i][j] * x[j][0];
        }
        x[i][0] = (y[i][0] - temp) / U[i][i];
    }

    Display_Matrix(x, 4);
    //打印答案
}


int main()
{
    //打印Doolittle分解法产生的解
    printf("\n\nDoolittle start\n\n");
    printf("Matrix 1\n");
    Doolittle_A1();
    printf("Matrix 2\n");
    Doolittle_A2();
    printf("\n\nDoolittle over\n\n");

    //打印Gauss列主元法产生的解
    printf("\n\nGauss start\n\n");
    printf("Matrix 1\n");
    GaussEliminationWithPartialPivoting_A1();
    printf("Matrix 2\n");
    GaussEliminationWithPartialPivoting_A2();
    printf("\n\nGauss over\n\n");

    return 0;

}
```

实验结果：

输出说明：Doolittle start至Doolittle over之间是Doolittle分解法的解，其中L Matrix下是L矩阵，U Matrix下是U矩阵，Solution为解。

Gauss start至Gauss over之间是Gauss消元法的解，其中Matrix为列主元完的上三角矩阵，Solution为解。

Matrix 1是代码中A1的输出，Matrix 2是代码中A2的输出，A1、A2分别计算的是第一个与第二个方程组。

```
Doolittle start


Matrix 1
L Matrix
The matrix is :
1.000000 0.000000 0.000000 0.000000 0.000000
1.125000 1.000000 0.000000 0.000000 0.000000
1.285714 2.666667 1.000000 0.000000 0.000000
1.500000 5.600000 5.250000 1.000000 0.000000
1.800000 11.200000 21.000000 12.000000 1.000000

U Matrix
The matrix is :
0.111111 0.125000 0.142857 0.166667 0.200000
0.000000 0.002232 0.005952 0.012500 0.025000
0.000000 0.000000 0.000454 0.002381 0.009524
0.000000 0.000000 0.000000 0.000833 0.010000
0.000000 0.000000 0.000000 0.000000 0.040000


The Solution is :
 630.000000  -1120.000000  630.000000  -120.000000  5.000000

Error is : 0.000000

Matrix 2
L Matrix
The matrix is :
1.000000 0.000000 0.000000 0.000000
0.180556 1.000000 0.000000 0.000000
0.777778 -0.151050 1.000000 0.000000
0.208333 -0.013453 0.412134 1.000000

U Matrix
The matrix is :
7.200000 2.300000 -4.400000 0.500000
0.000000 5.884722 -2.705556 2.709722
0.000000 0.000000 11.113547 -1.279585
0.000000 0.000000 0.000000 6.359647


The Solution is :
 3.000000  -2.000000  1.000000  5.000000

Error is : 0.000000


Doolittle over
```

```
Gauss start


Matrix 1

The matrix is :
0.200000 0.250000 0.333333 0.500000 1.000000
0.000000 -0.013889 -0.042328 -0.111111 -0.355556
0.000000 0.000000 -0.002381 -0.016667 -0.120000
0.000000 0.000000 0.000000 0.000794 0.015238
0.000000 0.000000 0.000000 0.000000 -0.000714

The Solution is :
 630.000000  -1120.000000  630.000000  -120.000000  5.000000

Error is : 0.000000

Matrix 2

The matrix is :
7.200000 2.300000 -4.400000 0.500000
0.000000 5.884722 -2.705556 2.709722
0.000000 0.000000 11.113547 -1.279585
0.000000 0.000000 0.000000 6.359647

The Solution is :
 3.000000  -2.000000  1.000000  5.000000

Error is : 0.000000


Gauss over
```

Gauss列主元法:

1、

得到的上三角矩阵U的系数为:

The matrix is :

0.200000 0.250000 0.333333 0.500000 1.000000
0.000000 -0.013889 -0.042328 -0.111111 -0.355556
0.000000 0.000000 -0.002381 -0.016667 -0.120000
0.000000 0.000000 0.000000 0.000794 0.015238
0.000000 0.000000 0.000000 0.000000 -0.000714

得到的解为:

The Solution is :
630.000000 -1120.000000 630.000000 -120.000000 5.000000

计算得到的误差（2-范数）为：

Error is : 0.000000

2、

得到的上三角矩阵U的系数为：

The matrix is :

7.200000 2.300000 -4.400000 0.500000
0.000000 5.884722 -2.705556 2.709722
0.000000 0.000000 11.113547 -1.279585
0.000000 0.000000 0.000000 6.359647

得到的解为：

The Solution is :
 3.000000 -2.000000 1.000000 5.000000

计算得到的误差（2-范数）为：

Error is : 0.000000

Doolittle直接分解法：

1、

得到的下三角矩阵L的系数为：

The matrix is :
1.000000 0.000000 0.000000 0.000000 0.000000
1.125000 1.000000 0.000000 0.000000 0.000000
1.285714 2.666667 1.000000 0.000000 0.000000
1.500000 5.600000 5.250000 1.000000 0.000000
1.800000 11.200000 21.000000 12.000000 1.000000

得到的上三角矩阵U的系数为：

The matrix is :
0.111111 0.125000 0.142857 0.166667 0.200000
0.000000 0.002232 0.005952 0.012500 0.025000
0.000000 0.000000 0.000454 0.002381 0.009524
0.000000 0.000000 0.000000 0.000833 0.010000
0.000000 0.000000 0.000000 0.000000 0.040000

得到的解为：

The Solution is :
 630.000000 -1120.000000 630.000000 -120.000000 5.000000

计算得到的误差（2-范数）为：

Error is : 0.000000

2、

得到的下三角矩阵L的系数为：

The matrix is :
1.000000 0.000000 0.000000 0.000000
0.180556 1.000000 0.000000 0.000000
0.777778 -0.151050 1.000000 0.000000
0.208333 -0.013453 0.412134 1.000000

得到的上三角矩阵U的系数为：

The matrix is :
7.200000 2.300000 -4.400000 0.500000
0.000000 5.884722 -2.705556 2.709722
0.000000 0.000000 11.113547 -1.279585
0.000000 0.000000 0.000000 6.359647

得到的解为：

The Solution is :
3.000000 -2.000000 1.000000 5.000000

计算得到的误差（2-范数）为：

Error is : 0.000000

两种解法的优劣

可以看出，Doolittle直接分解法和列主元Gauss消元法对这两个方程组的解的误差都为0.000000，从这两个矩阵的分解中中并不能看出孰优孰劣，两种误差的表现相近。