实验 1 解线性方程组的直接方法

201311211914

赵帅帅

2015-3-15

一、 实验问题

给定下列几个不同类型的线性方程组，请用适当的直接法求解。

1. 线性方程组

$$\begin{bmatrix} 4 & 2 & -3 & -1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 8 & 6 & -5 & -3 & 6 & 5 & 0 & 1 & 0 & 0 \\ 4 & 2 & -2 & -1 & 3 & 2 & -1 & 0 & 3 & 1 \\ 0 & -2 & 1 & 5 & -1 & 3 & -1 & 1 & 9 & 4 \\ -4 & 2 & 6 & -1 & 6 & 7 & -3 & 3 & 2 & 3 \\ 8 & 6 & -8 & 5 & 7 & 17 & 2 & 6 & -3 & 5 \\ 0 & 2 & -1 & 3 & -4 & 2 & 5 & 3 & 0 & 1 \\ 16 & 10 & -11 & -9 & 17 & 34 & 2 & -1 & 2 & 2 \\ 4 & 6 & 2 & -7 & 13 & 9 & 2 & 0 & 12 & 4 \\ 0 & 0 & -1 & 8 & -3 & -24 & -8 & 6 & 3 & -1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \\ x10 \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \\ 3 \\ 2 \\ 3 \\ 46 \\ 13 \\ 38 \\ 19 \\ -21 \end{bmatrix}$$

精确解 $x^* = (1，\ -1，\ 0，\ 1，\ 2，\ 0，\ 3，\ 1，\ -1，\ 2)^T$

2. 对称正定线性方程组

$$\begin{bmatrix} 4 & 2 & -4 & 0 & 2 & 4 & 0 & 0 \\ 2 & 2 & -1 & -2 & 1 & 3 & 2 & 0 \\ -4 & -1 & 14 & 1 & -8 & -3 & 5 & 6 \\ 0 & -2 & 1 & 6 & -1 & -4 & -3 & 3 \\ 2 & 1 & -8 & -1 & 22 & 4 & -10 & -3 \\ 4 & 3 & -3 & -4 & 4 & 11 & 1 & -4 \\ 0 & 2 & 5 & -3 & -10 & 1 & 14 & 2 \\ 0 & 0 & 6 & 3 & -3 & -4 & 2 & 19 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \end{bmatrix} = \begin{bmatrix} 0 \\ -6 \\ 6 \\ 23 \\ 11 \\ -22 \\ -15 \\ 45 \end{bmatrix}$$

精确解 $x^* = (1，\ -1，\ 0，\ 2，\ 1，\ -1，\ 0，\ 2)^T$

3. 三对角线性方程组

$$\begin{bmatrix} 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \\ x10 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \\ -13 \\ 2 \\ 6 \\ -12 \\ 14 \\ -4 \\ 5 \\ -5 \end{bmatrix}$$

精确解 $x^* = (2，\ 1，\ -3，\ 0，\ 1，\ -2，\ 3，\ 0，\ 1，\ -1)^T$

二、 实验要求

1. 对上述三个方程组分别利用 Gauss 顺序消去法与 Gauss 列主元消去法；平方根法；追赶法求解。
2. 编出算法通用程序。
3. 在应用 Gauss 消去法时，尽可能利用相应程序输出系数矩阵的三角分解式。

三、 实验目的和意义

1. 通过写程序，掌握模块化结构程序设计方法。
2. 掌握求解各类线性方程组的直接方法，了解各种方法的特点。
3. 体会 Gauss 消去法选主元的必要性。

四、 算法

1. Gauss 顺序消去法

输入：$A=\left(a_{ij}\right)$，$b=\left(b_1, \ldots, b_n\right)^T$，维数 n

输出：方程组解$x_1, \ldots, x_n$，或方程组无解信息

（1） 对于 k=1，2，…，n-1，执行

对于 i=k+1，…，n，计算

$$l_{ik}=\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

$$a_{ij}^{(k+1)}=a_{ij}^{(k)}-l_{ik}a_{kj}^{(k)}\ ,\quad j=k+1,\ \ldots,\ n$$

$$b_i^{(k+1)}=b_i^{(k)}-l_{ik}b_k^{(k)}$$

（2） 回带过程：

$$x_n=\frac{b_n^{(n)}}{a_{nn}^{(n)}}$$

对于 i=n-1，n-2，…，1，计算

$$x_i=\frac{b_i^{(i)}-\sum_{j=i+1}^{n}a_{ij}^{(i)}x_j}{a_{ii}^{(i)}}$$

2. Gauss 列主元消去法

输入：$A=\left(a_{ij}\right)$，$b=\left(b_1, \ldots, b_n\right)^T$，维数 n

输出：方程组解$x_1, \ldots, x_n$，或方程组无解信息

（1） 对于 k = 1，2，…，n-1，循环执行步 2 到步 5；

（2） 按列选主元素 a_{ik}，即确定下标 i 使

$$|a_{ik}| = \max_{k \le j \le n} |a_{jk}|$$

（3） 若 $a_{ik} = 0$，输出“no unique solution”，停止计算；

（4） 若$i \neq k$，换行

$a_{kj} \leftrightarrow a_{ij}$ $j = k$， …， n

$b_k \leftrightarrow b_i$

（5） 消元计算，对于 i = k+1，…，n，计算

$l_{ik} = \dfrac{a_{ik}}{a_{kk}}$

$a_{ij} = a_{ij} - l_{ik} a_{kj}$ $j = k+1$， …， n

$b_i = b_i - l_{ik} b_k$

（6） 若 $a_{nn} = 0$，输出“no unique solution”，停止计算；

（7） 回带求解

$x_n = \frac{b_n}{a_{nn}}$

$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j}{a_{ii}}$ $i = n-1$， …， 2， 1

3. 平方根法

求解对称正定方程组 Ax = b。首先，计算分解 $A = GG^T$，G 的元素 $g_{ik}(i \ge j)$存放在 A 的下三角部分，然后求解方程组 Gy = b 和 $G^T x = y$。

输入：$A = (a_{ij})$，$i \ge j$，$b = (b_1, \ldots, b_n)^T$，维数 n

输出：方程组的解 $x = (x_1, \ldots, x_n)^T$

（1） 对于 k = 1，2，…，n，循环执行步 2 到步 4；

（2） $a_{kk} \Leftarrow g_{kk} = \left(a_{kk} - \sum_{m=1}^{k-1} a_{km}^2\right)^{\frac{1}{2}}$

（3） 对于 i = k + 1，…，n，计算

$a_{ik} \Leftarrow g_{ik} = \dfrac{a_{ik} - \sum_{m=1}^{k-1} a_{im} a_{km}}{a_{kk}}$

（4） $y_k \Leftarrow \frac{b_k - \sum_{m=1}^{k-1} a_{km} y_m}{a_{kk}}$

（5） $x_n \Leftarrow \frac{y_n}{a_{nn}}$

$$x_k \Leftarrow \frac{y_k - \sum_{m=k+1}^{n} a_{mk} x_m}{a_{kk}} \quad k = n-1，\ …，\ 2，\ 1$$

（6） 输出 x = (x_1， …， x_n)^T

4. 追赶法

求解三对角方程组 Ax = b，用四个一维数组存放方程组数据$\{a_i，c_i，d_i，b_i\}$。

输入：方程组数据$\{a_i，c_i，d_i，b_i，n\}$

输出：方程组解 x = (x_1， …， x_n)^T

（1） $\alpha_1 \Leftarrow a_1$

（2） 对于 i = 1，2，…，n-1，计算

$$\beta_i \Leftarrow \frac{c_i}{\alpha_i}$$

$$\alpha_{i+1} \Leftarrow a_{i+1} - d_{i+1}\beta_i$$

（3） $y_i \Leftarrow \frac{b_1}{\alpha_i}$

$$y_i \Leftarrow \frac{b_1 - d_i y_{i-1}}{\alpha_i}$$

（4） $x_n \Leftarrow y_n$

$$x_i \Leftarrow y_i - \beta_i x_{i+1}，\ \mathrm{i = n-1}，\ …，\ 2，\ 1$$

（5） 输出 x = (x_1， …， x_n)^T

五、 实验代码

Gauss 顺序消去法：

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

const int N = 10;
double a[N][N] =
{
    {4, 2, -3, -1, 2, 1, 0, 0, 0, 0},
    {8, 6, -5, -3, 6, 5, 0, 1, 0, 0},
    {4, 2, -2, -1, 3, 2, -1, 0, 3, 1},
    {0, -2, 1, 5, -1, 3, -1, 1, 9, 4},
    {-4, 2, 6, -1, 6, 7, -3, 3, 2, 3},
    {8, 6, -8, 5, 7, 17, 2, 6, -3, 5},
    {0, 2, -1, 3, -4, 2, 5, 3, 0, 1},
    {16, 10, -11, -9, 17, 34, 2, -1, 2, 2},
```

```
    {4, 6, 2, -7, 13, 9, 2, 0, 12, 4},
    {0, 0, -1, 8, -3, -24, -8, 6, 3, -1}
};
double b[N] = {5, 12, 3, 2, 3, 46, 13, 38, 19, -21};
bool Gauss(double a[N][N], double b[N], double x[N])
{
    for(int k = 0; k < N - 1; k ++)
    {
        //处理增广矩阵
        for(int i = k + 1; i <= N - 1; i ++)
        {
            if(a[k][k] == 0)
            {
                printf("Divided by zero!");
                return false;
            }
            double l = a[i][k] / a[k][k];
            for(int j = k + 1; j <= N - 1; j++)
            {
                a[i][j] = a[i][j] - l * a[k][j];
            }
            b[i] = b[i] - l * b[k];
        }
    }
    //回代求解
    x[N - 1] = b[N - 1] / a[N - 1][N - 1];
    for(int i = N - 2; i >= 0; i --)
    {
        double sum = 0;
        for(int j = i + 1; j <= N - 1; j ++)
        {
            sum += a[i][j] * x[j];
        }
        x[i] = (b[i] - sum ) / a[i][i];
    }
    return true;
}

int main(void)
{
    double x[N] = {0};
    if(Gauss(a, b, x))
    {
        //输出处理后的增广矩阵
```

```
        for(int i = 0; i < N; i++)
        {
            for(int j = 0; j < i; j++)
            {
                printf("      ");
            }
            for(int j = i ; j < N; j ++)
            {
                printf("%6.2lf", a[i][j]);
            }
            printf("%6.2lf\n", b[i]);

        }
        //输出解
        for(int i = 0; i < N; i ++)
        {
            printf("%10.6lf\n", x[i]);
        }
    }

    return 0;
}
```

Gauss 列主元消去法：

```
#include<stdio.h>
#include<stdlib.h>
#include<cmath>
const int N = 10;
double a[N][N] =
{
    {4, 2, -3, -1, 2, 1, 0, 0, 0, 0},
    {8, 6, -5, -3, 6, 5, 0, 1, 0, 0},
    {4, 2, -2, -1, 3, 2, -1, 0, 3, 1},
    {0, -2, 1, 5, -1, 3, -1, 1, 9, 4},
    {-4, 2, 6, -1, 6, 7, -3, 3, 2, 3},
    {8, 6, -8, 5, 7, 17, 2, 6, -3, 5},
    {0, 2, -1, 3, -4, 2, 5, 3, 0, 1},
    {16, 10, -11, -9, 17, 34, 2, -1, 2, 2},
    {4, 6, 2, -7, 13, 9, 2, 0, 12, 4},
    {0, 0, -1, 8, -3, -24, -8, 6, 3, -1}
};
double b[N] = {5, 12, 3, 2, 3, 46, 13, 38, 19, -21};
bool Gauss_update(double a[N][N], double b[N], double x[N])
{
    for(int k = 0; k < N - 1; k ++)
```

```
{
    //选列主元
    int max_index = k;
    double max_ele = a[k][k];
    for(int j = k; j <= N - 1; j++)
    {
        if(abs(a[j][k]) > abs(max_ele))
        {
            max_index = j;
            max_ele = a[j][k];
        }
    }
    //交换两行
    if(max_index != k)
    {
        double temp = 0;
        for(int j = k; j <= N-1; j++)
        {
            temp = a[k][j];
            a[k][j] = a[max_index][j];
            a[max_index][j] = temp;
        }
        temp = b[k];
        b[k] = b[max_index];
        b[max_index] = temp;
    }
    //处理增广矩阵
    for(int i = k + 1; i <= N - 1; i ++)
    {
        if(a[k][k] == 0)
        {
            printf("Divided by zero!");
            return false;
        }
        double l = a[i][k] / a[k][k];
        for(int j = k + 1; j <= N - 1; j++)
        {
            a[i][j] = a[i][j] - l * a[k][j];
        }
        b[i] = b[i] - l * b[k];
    }
}
//回代求解
x[N - 1] = b[N - 1] / a[N - 1][N - 1];
```

```
    for(int i = N - 2; i >= 0; i --)
    {
        double sum = 0;
        for(int j = i + 1; j <= N - 1; j ++)
        {
            sum += a[i][j] * x[j];
        }
        x[i] = (b[i] - sum ) / a[i][i];
    }
    return true;
}

int main(void)
{
    double x[N] = {0};
    if(Gauss_update(a, b, x))
    {
        //输出处理后的增广矩阵
        for(int i = 0; i < N; i++)
        {
            for(int j = 0; j < i; j++)
            {
                printf("      ");
            }
            for(int j = i ; j < N; j ++)
            {
                printf("%6.2lf", a[i][j]);
            }
            printf("%6.2lf\n", b[i]);

        }
        //输出解
        for(int i = 0; i < N; i ++)
        {
            printf("%10.6lf\n", x[i]);
        }
    }
    return 0;
}
```

平方根法:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
const int N = 8;
double a[N][N] =
{
    {4, 2, -4, 0, 2, 4, 0, 0},
    {2, 2, -1, -2, 1, 3, 2, 0},
    {-4, -1, 14, 1, -8, -3, 5, 6},
    {0, -2, 1, 6, -1, -4, -3, 3},
    {2, 1, -8, -1, 22, 4, -10, -3},
    {4, 3, -3, -4, 4, 11, 1, -4},
    {0, 2, 5, -3, -10, 1, 14, 2},
    {0, 0, 6, 3, -3, -4, 2, 19}
};
double b[N] = {0,-6,6,23,11,-22,-15,45};

bool Choleskey(double a[N][N], double b[N], double x[N])
{
    double sum = 0;
    double y[N] = {0};
    for(int k = 0; k <= N - 1; k ++)
    {
        //求对角线的元素
        sum = 0;
        for(int m = 0; m <= k - 1; m ++)
        {
            sum += a[k][m] * a[k][m];
        }
        a[k][k] = sqrt(a[k][k] - sum);
        if(a[k][k] == 0)
        {
            printf("Divided by zero!");
            return false;
        }
        //求非对角线元素
        for(int i = k + 1; i <= N - 1; i ++)
        {
            sum = 0;
            for(int m = 0; m <= k - 1; m ++)
            {
                sum += a[i][m] * a[k][m];
            }
```

```
            a[i][k] = (a[i][k] - sum) / a[k][k];
        }
        //解G y = b
        sum = 0;
        for(int m = 0; m <= k - 1; m ++)
        {
            sum += a[k][m] * y[m];
        }
        y[k] = (b[k] - sum) / a[k][k];
    }
    //解GT x = y
    x[N - 1] = y[N - 1] / a[N - 1][N - 1];
    for(int k = N -2; k >= 0; k --)
    {
        sum = 0;
        for(int m = k + 1; m <= N - 1; m ++)
        {
            sum += a[m][k] * x[m];
        }
        x[k] = (y[k] - sum) / a[k][k];
    }
    return true;
}

int main(void)
{
    double x[N] = {0};
    if(Choleskey(a, b, x))
    {
        //输出分解后的系数矩阵G
        for(int i = 0; i < N; i++)
        {
            for(int j = 0 ; j <= i; j ++)
            {
                printf("%6.2lf", a[i][j]);
            }
            printf("\n");
        }
        //输出解
        for(int i = 0; i < N; i ++)
        {
            printf("%10.6lf\n", x[i]);
        }
    }
```

```
    return 0;
}
```

追赶法：

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

const int N = 10;
double a[N] = {4, 4, 4, 4, 4, 4, 4, 4, 4, 4};
double c[N - 1] = {-1, -1, -1, -1, -1, -1, -1, -1, -1};
double d[N] = {0, -1, -1, -1, -1, -1, -1, -1, -1, -1};
double b[N] = {7, 5, -13, 2, 6, -12, 14, -4, 5, -5};
bool Crout(double a[N],double c[N], double d[N], double b[N],
double x[N])
{
    //分解矩阵
    for(int i = 0; i <= N - 2; i ++)
    {
        c[i] = c[i] / a[i];
        a[i + 1] = a[i + 1] - d[i + 1] * c[i];
    }
    //分析知 实际上并不需要中间变量y[N]
    //计算逻辑上的y[N]
    x[0] = b[0] / a[0];
    for(int i = 1; i <= N - 1; i ++)
    {
        x[i] = (b[i] - d[i] * x[i - 1]) / a[i];
    }
    //求出x[N];
    for(int i = N - 2; i >=0; i --)
    {
        x[i] = x[i] - c[i] * x[i + 1];
    }
    return true;
}

int main(void)
{
    //A = TM
    double x[N] = {0};
    if(Crout(a, c, d, b, x))
    {
        //打印T
        printf("%6.2lf\n", a[0]);
        for(int i = 1; i <= N-1; i++)
```

```
        {
            for(int k = 0; k <= i - 2; k ++)
            {
                printf("      ");
            }
            printf("%6.2lf%6.2lf\n", d[i], a[i]);
        }
        //打印M
        for(int i = 0; i <= N-2; i++)
        {
            for(int k = 0; k <= i - 1; k ++)
            {
                printf("      ");
            }
            printf("  1.00%6.2lf\n", a[i]);
        }
        for(int k = 0; k <= N - 2; k ++)
        {
            printf("      ");
        }
        printf("  1.00\n");
        //输出解
        for(int i = 0; i < N; i ++)
        {
            printf("%10.6lf\n", x[i]);
        }
    }

    return 0;
}
```

六、 实验结果

操作系统：Windows 10 企业版

GCC Version：4.7.1

G++ Version：4.7.1

Gauss 顺序消去法：

```
 4.00  2.00 -3.00 -1.00  2.00  1.00  0.00  0.00  0.00  0.00  5.00
       2.00  1.00 -1.00  2.00  3.00  0.00  1.00  0.00  0.00  2.00
             1.00  0.00  1.00  1.00 -1.00  0.00  3.00  1.00 -2.00
                   4.00 -1.00  4.00  1.00  2.00  3.00  2.00  8.00
                         3.00  1.00 -2.00  1.00 -1.00  2.00  6.00
                               5.00  1.00 -1.00  2.00  0.00  0.00
                                     0.40  0.60  2.80  3.00  5.00
                                           2.00-13.00 -9.00 -3.00
                                              -125.00-110.00-95.00
                                                    -12.14-24.28
 1.000000
-1.000000
-0.000000
 1.000000
 2.000000
 0.000000
 3.000000
 1.000000
-1.000000
 2.000000
```

Gauss 列主元消去法：

```
16.00 10.00-11.00 -9.00 17.00 34.00  2.00 -1.00  2.00  2.00 38.00
       4.50  3.25 -3.25 10.25 15.50 -2.50  2.75  2.50  3.50 12.50
            -3.22 10.22 -3.78 -3.44  1.56  5.89 -4.56  3.22 24.22
                  11.31  0.69  7.28 -0.93  6.69  6.66  8.00 25.93
                        -5.49 -0.15  4.66 -0.73  4.29 -0.66 -3.34
                             -25.98 -9.89  1.60 -0.09 -5.16-38.29
                                     9.58 -1.45  1.97  3.21 31.73
                                          -0.52 -6.05  0.08  5.68
                                                -1.23  0.32  1.88
                                                       0.25  0.51
 1.000000
-1.000000
 0.000000
 1.000000
 2.000000
-0.000000
 3.000000
 1.000000
-1.000000
 2.000000
```

平方根法：

```
  2.00
  1.00  1.00
 -2.00  1.00  3.00
  0.00 -2.00  1.00  1.00
  1.00  0.00 -2.00  1.00  4.00
  2.00  1.00  0.00 -2.00  1.00  1.00
  0.00  2.00  1.00  0.00 -2.00  1.00  2.00
  0.00  0.00  2.00  1.00  0.00 -2.00  1.00  3.00
 1.000000
-1.000000
 0.000000
 2.000000
 1.000000
-1.000000
 0.000000
 2.000000
```

追赶法：

```
  4.00
 -1.00  3.75
       -1.00  3.73
             -1.00  3.73
                   -1.00  3.73
                         -1.00  3.73
                               -1.00  3.73
                                     -1.00  3.73
                                           -1.00  3.73
                                                 -1.00  3.73
  1.00  4.00
        1.00  3.75
              1.00  3.73
                    1.00  3.73
                          1.00  3.73
                                1.00  3.73
                                      1.00  3.73
                                            1.00  3.73
                                                  1.00  3.73
                                                        1.00
 2.000000
 1.000000
-3.000000
 0.000000
 1.000000
-2.000000
 3.000000
-0.000000
 1.000000
-1.000000
```

七、 实验感受

1. 因为算法都有，所以编程没有什么难处。
2. 写完程序之后，感觉自己对线性方程组的直接解法有了理解性的记忆。