计算方法-第三次上机作业

PB19051183 吴承泽

实验源码：

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

#define INF -65535

//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 A_2[4][4] = {
                      4 , -1 , 1 , 3 ,
                      16 , -2 , -2 , 5 ,
                      16 , -3 , -1 , 7 ,
                      6 , -4 , 2 , 9 ,
};


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

//获取向量的无穷范数
double Get_Max(double X[][1], int n)
{
    int i;
    double max = INF;
    for(i = 0; i < n; i++)
    {
        if(fabs(X[i][0]) > max)
            max = fabs(X[i][0]);
    }
    return max;
}
```

```
//行列数为5的Doolittle矩阵分解函数

int Doolittle_A1(double A_1[][5], double Y[][1], double X[][1])
{
    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LU矩阵分解完毕
    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] = Y[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];
    }

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



//通过反幂法迭代得到特征值与特征向量

double Get_eig_A1(double A_1[][5],double Feature_Vector[][1])
{
    int i,k = 0;
    double X[5][1] = { 1 ,
                       1 ,
                       1 ,
                       1 ,
                       1
                     };
    double X_next[5][1] = { 1 ,
                       1 ,
                       1 ,
                       1 ,
                       1
                     };
    double Y[5][1];
    //暂且初始化为-65535
    double eigenvalue, eigenvalue_next = INF;

    do
    {
        eigenvalue = eigenvalue_next;
        for(i = 0; i < 5; i++)
        {
            X[i][0] = X_next[i][0];
        }
        for(i = 0; i < 5; i++)
        {
            Y[i][0] = X[i][0] / Get_Max(X,5);
        }
        Doolittle_A1(A_1,Y,X_next);
        eigenvalue_next = Get_Max(X_next,5);

        printf("X(%d) is:\n",k);
```

```
        Display_Matrix(X,5);
        printf("Y(%d) is:\n",k);
        Display_Matrix(Y,5);
        printf("eigenvalue is : %lf\n\n", eigenvalue_next);

        k++;
    }while(fabs(eigenvalue_next - eigenvalue) >= 1e-5);


    for(i = 0; i < 5; i++)
    {
        Feature_Vector[i][0] = Y[i][0];
    }
    return 1 / eigenvalue;
}

//行列数为4的Doolittle矩阵分解函数

int Doolittle_A2(double A_2[][4], double Y[][1], double X[][1])
{
    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];
        }
```

```
    }


    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] = Y[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];
    }
    for(i = 0; i < 4; i++)
    {
        X[i][0] = x[i][0];
    }
}

double Get_eig_A2(double A_2[][4],double Feature_Vector[][1])
{
    int i, k = 0;
    double X[4][1] = { 1 ,
                       1 ,
                       1 ,
                       1
                     };
    double X_next[4][1] = { 1 ,
                       1 ,
                       1 ,
                       1
                     };
    double Y[4][1];
    //暂且初始化为-65535
    double eigenvalue, eigenvalue_next = INF;

    do
    {
        eigenvalue = eigenvalue_next;
        for(i = 0; i < 4; i++)
        {
```

```
            X[i][0] = X_next[i][0];
        }
        for(i = 0; i < 4; i++)
        {
            Y[i][0] = X[i][0] / Get_Max(X,4);
        }
        Doolittle_A2(A_2,Y,X_next);
        eigenvalue_next = Get_Max(X_next,4);

        printf("X(%d) is:\n",k);
        Display_Matrix(X,4);
        printf("Y(%d) is:\n",k);
        Display_Matrix(Y,4);
        printf("eigenvalue is : %lf\n\n", eigenvalue_next);

        k++;
    }while(fabs(eigenvalue_next - eigenvalue) >= 1e-5);


    for(i = 0; i < 4; i++)
    {
        Feature_Vector[i][0] = Y[i][0];
    }
    return 1 / eigenvalue;
}

int main()
{
    int i;
    double Feature_Vector_A1[5][1];
    double min_eigenvalue_A1 = Get_eig_A1(A_1,Feature_Vector_A1);
    printf("\n\n\n The final minimum eigenvalue of A1 is :%lf\n
",min_eigenvalue_A1);
    printf("The Feature vector of A1 is:");
    Display_Matrix(Feature_Vector_A1,5);

    printf("\n\n\n");

    double Feature_Vector_A2[4][1];
    double min_eigenvalue_A2 = Get_eig_A2(A_2,Feature_Vector_A2);
    printf("\n\n\n The final minimum eigenvalue of A2 is :%lf\n
",min_eigenvalue_A2);
    printf("The Feature vector of A2 is:");
    Display_Matrix(Feature_Vector_A1,4);



    return 0;
}
```

实验结果：

X(k),Y(k)表示的是第k次迭代中向量的值，其中eigenvalue是当时通过||X(k)||∞得到的特征值，在递归出口会打印得到的特征值和特征向量

```
PS C:\VscodeWorkspace\计算方法> cd "c:\VscodeWorkspace\计算方法\" ; if ($?) { gcc
HW3-PB19051183-吴承泽.c -o HW3-PB19051183-吴承泽 } ; if ($?) { .\HW3-PB19051183-吴
承泽 }

X(0) is:
 1.000000  1.000000  1.000000  1.000000  1.000000
Y(0) is: 1.000000  1.000000  1.000000  1.000000  1.000000
eigenvalue is : 1120.000000

X(1) is:
 630.000000  -1120.000000  630.000000  -120.000000  5.000000
Y(1) is:
 0.562500  -1.000000  0.562500  -0.107143  0.004464
eigenvalue is : 297848.750000

X(2) is:
 146252.812500  -297848.750000  196174.687500  -45114.375000  2377.254464
Y(2) is:
 0.491030  -1.000000  0.658639  -0.151467  0.007981
eigenvalue is : 304047.406171

X(3) is:
 149112.770686  -304047.406171  200595.275176  -46244.691668  2446.559496
Y(3) is:
 0.490426  -1.000000  0.659750  -0.152097  0.008047
eigenvalue is : 304141.809927

X(4) is:
 149157.105244  -304141.809927  200661.194040  -46261.122745  2447.536172
Y(4) is:
 0.490420  -1.000000  0.659762  -0.152104  0.008047
eigenvalue is : 304142.830586

X(5) is:
 149157.584694  -304142.830586  200661.906514  -46261.300272  2447.546720
Y(5) is:
 0.490420  -1.000000  0.659762  -0.152104  0.008047
eigenvalue is : 304142.841558

X(6) is:
 149157.589849  -304142.841558  200661.914173  -46261.302180  2447.546833
Y(6) is:
 0.490420  -1.000000  0.659762  -0.152104  0.008047
eigenvalue is : 304142.841676

X(7) is:
 149157.589904  -304142.841676  200661.914255  -46261.302201  2447.546834
Y(7) is:
 0.490420  -1.000000  0.659762  -0.152104  0.008047
eigenvalue is : 304142.841677




 The final minimum eigenvalue of A1 is :0.000003
 The Feature vector of A1 is: 0.490420  -1.000000  0.659762  -0.152104  0.008047
```

```
X(0) is:
 1.000000  1.000000  1.000000  1.000000
Y(0) is:
 1.000000  1.000000  1.000000  1.000000
eigenvalue is : 2.000000

X(1) is:
 0.000000  2.000000  -0.000000  1.000000
Y(1) is:
 0.000000  1.000000  -0.000000  0.500000
eigenvalue is : 5.625000

X(2) is:
 -0.625000  5.625000  -2.375000  3.500000
Y(2) is:
 -0.111111  1.000000  -0.422222  0.622222
eigenvalue is : 8.077778

X(3) is:
 -0.933333  8.077778  -3.433333  5.044444
Y(3) is:
 -0.115543  1.000000  -0.425034  0.624484
eigenvalue is : 8.089924

X(4) is:
 -0.936210  8.089924  -3.443776  5.054333
Y(4) is:
 -0.115725  1.000000  -0.425687  0.624769
eigenvalue is : 8.093818

X(5) is:
 -0.936712  8.093818  -3.445490  5.056811
Y(5) is:
 -0.115732  1.000000  -0.425694  0.624774
eigenvalue is : 8.093856

X(6) is:
 -0.936719  8.093856  -3.445513  5.056838
Y(6) is:
 -0.115732  1.000000  -0.425695  0.624775
eigenvalue is : 8.093861




 The final minimum eigenvalue of A2 is :0.123551
 The Feature vector of A2 is: 0.490420  -1.000000  0.659762  -0.152104
```

结果分析：

表格如下：

A_1

k	Y_0^k	Y_1^k	Y_2^k	Y_3^k	Y_4^k
0	1	1	1	1	1
1	0.5625	-1	0.5625	-0.107143	0.004464
2	0.491030	-1	0.658639	-0.151467	0.007981
3	0.490426	-1	0.659750	-0.152097	0.008047
4	0.490420	-1	0.659762	-0.152104	0.008047
5	0.490420	-1	0.659762	-0.152104	0.008047
6	0.490420	-1	0.659762	-0.152104	0.008047
7	0.490420	-1	0.659762	-0.152104	0.008047

k	X_0^k	X_1^k	X_2^k	X_3^k	X_4^k
0	630	-1120	630	-120	5
1	146252.8125	-297848.75	196174.6875	-45114.375	2377.254464
2	149112.770686	-304047.406171	200595.275176	-46244.691668	2446.559496
3	149157.105244	-304141.809927	200661.194040	-46261.122745	2447.536172
4	149157.584694	-304142.830585	200661.906514	-46261.300272	2447.546720
5	149157.589848	-304142.841558	200661.914173	-46261.302180	2447.546833
6	149157.589904	-304142.841675	200661.914255	-46261.302201	2447.546834
7	149157.589904	-304142.841676	200661.914255	-46261.302201	2447.546834

递归结束时计算得到的特征值为0.000003，特征向量为$(0.490420 ,-1.000000 ,0.659762 ,-0.152104 ,0.008047)^T$

A_2

k	Y_0^k	Y_1^k	Y_2^k	Y_3^k
0	1	1	1	1
1	0.000000	1.000000	-0.000000	0.500000
2	-0.111111	1.000000	-0.422222	0.622222
3	-0.115543	1.000000	-0.425034	0.624484
4	-0.115725	1.000000	-0.425687	0.624769
5	-0.115732	1.000000	-0.425694	0.624774
6	-0.115732	1.000000	-0.425695	0.624775

k	X_0^k	X_1^k	X_2^k	X_3^k
0	0.000000	2.000000	-0.000000	1.000000
1	-0.625000	5.625000	-2.375000	3.500000
2	-0.933333	8.077778	-3.433333	5.044444
3	-0.936210	8.089924	-3.443776	5.054333
4	-0.936712	8.093818	-3.445490	5.056811
5	-0.936719	8.093856	-3.445513	5.056838
6	-0.936720	8.093861	-3.445515	5.056841

递归结束时计算得到的特征值为0.123551，特征向量为$(0.490420 ,-1.000000 ,0.659762 ,-0.152104)^T$

(a) 对比两个迭代过程的迭代次数，分析是否有“A的按模最小特征值越接近于0，收敛越快”。

并没有这种说法，第一个矩阵的按模最小特征值比第二个矩阵的按模最小特征值更接近于0，但是第一个矩阵进行了25次迭代，第二个矩阵仅进行了11次迭代，因此两者不具有题目中所说的关系。

(b) “估计每次迭代的特征值”中是否遇到问题，是如何解决的（提示：如Xk+1/Y k时是否有数值问题）。

在这次多次迭代中并没有遇到这种数值问题（Xk+1/Yk=0）,当出现某个Yk=0时，并不会影响计算机本身求解特征值，因为当Yk中某个元素为0时，通过LU分解计算得到了Xk+1时的新向量则不一定会含有0这个元素，在迭代后数值问题会消失。