Beijing University of Chemical Technology

Computing Methods

计算方法实验指导书

Cheng Yong

目录

创建日期：2019 年 11 月 13 日
更新日期：2019 年 11 月 14 日

第 1 章　Lagrange 插值方法

1.1　实验目的

(1) 熟悉简单的一阶和二阶 Lagrange 插值方法；

(2) 学会计算 Lagrange 基函数；

(3) 正确构造插值多项式；

(4) 对插值结果进行合理分析；

1.2　实验原理

Lagrange 插值多项式：$L_n(x) = \sum_{i=0}^{n} l_i(x) y_i$，其中 $l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$。

1.3　实验环境

Windows 10 + Visual Studio

1.4　实验内容

设函数为 $f(x) = \sqrt[5]{x}$，已知

x	$f(x)$
24	1.888175
26	1.918645
28	1.947294
30	1.961009

表 1.1: 数据样本表

使用 Lagrange 插值多项式计算 $f(25), f(27), f(29)$，并给出插值多项式。修改程序直至运行成功，查看运行结果，并和如下真实值进行比较。

x	$f(x)$
25	1.903653938715879
27	1.933182044931763
29	1.961009057454548

表 1.2: 数据真实值

1.5 样例程序

```
#include <stdio.h>
#include <math.h>
#define MAXSIZE 50

void input(double x[MAXSIZE], double y[MAXSIZE], long n);

void main(void) {
    double x[MAXSIZE], y[MAXSIZE], _x, _y, t;
    long n, i, j;
    printf("\n请输入插值节点的个数: ");
    scanf("%ld", &n);
    input(x, y, n);
    printf("\n请输入插值点: ");
    scanf("%lf", &_x);
    _y = 0;

    for(i = 0; i <= n - 1; i++) {
        t = 1;
        for(j = 0; j <= n-1; j++)
            if(j != i)
                t*=(_x-x[j])/(x[i] - x[j]);
        _y+= t*y[i];
    }
    printf("\n插值点(x,y) = (%lf, %lf)。",_x, _y);
}

void input(double x[MAXSIZE], double y[MAXSIZE], long n) {
    long i;
    for(i = 0; i <= n-1; i++) {
        printf("\n请输入插值节点x[%ld], y[%ld]:", i,i);
        scanf("%lf, %lf", &x[i], &y[i]);
    }
}
```

运行结果如下：

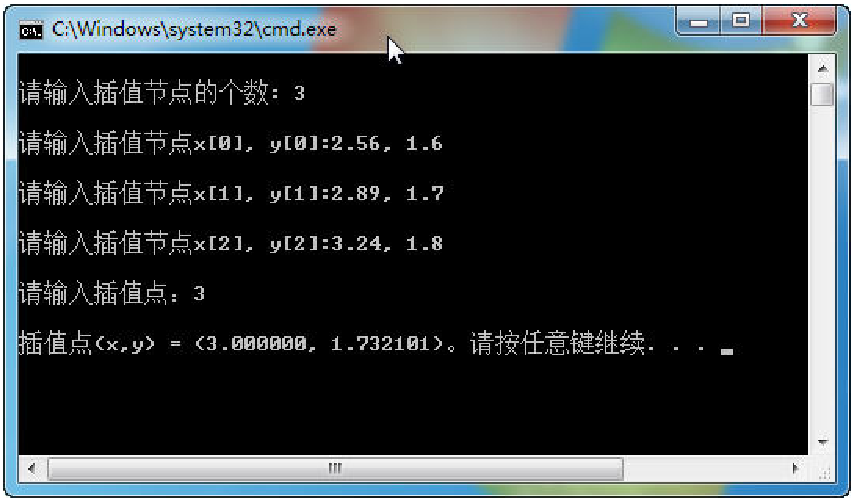

图 1.1: 运行结果

1.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 2 章　牛顿插值方法

2.1　实验目的

(1) 理解牛顿插值方法；

(2) 学会计算差商；

(3) 正确构造插值多项式；

(4) 设计程序并调试得到正确结果；

2.2　实验原理

参考教材第 5 章。

2.3　实验环境

Windows 10 + Visual Studio

2.4　实验内容

样本数据值如下：

x	0.4	0.55	0.65	0.8	0.9
$f(x)$	0.41075	0.57815	0.69675	0.88811	1.02652

表 2.1: 样本值

2.5　样例程序

```
#include<iostream>
#define N 4

using namespace std;

void difference(float *x,float *y,int n) {
    float *f;
    int k,i;
    f =new float[(n*sizeof(float))];
    for(k=1; k<=n; k++) {
        f[0]=y[k];
        for(i=0; i<k; i++)
```

```
            f[i+1]=(f[i]-y[i])/(x[k]-x[i]);
        y[k]=f[k];
    }
    cout<<endl;
    return ;
}

void main() {
    int i;
    float varx =0.895,b;
    float x[N+1] = {0.4,0.55,0.65,0.8,0.9};
    float y[N+1] = {0.41075,0.57815,0.69675,0.88811,1.02652};
    difference(x,(float *)y,N);
    b =y[N];
    for (i=N-1; i>=0; i--) {
        b = b*(varx-x[i])+y[i];
        cout<<b<<endl;
    }
    cout<<”Nn(”<<varx<<”)=”<<b<<endl;
}
```

运行结果如下：

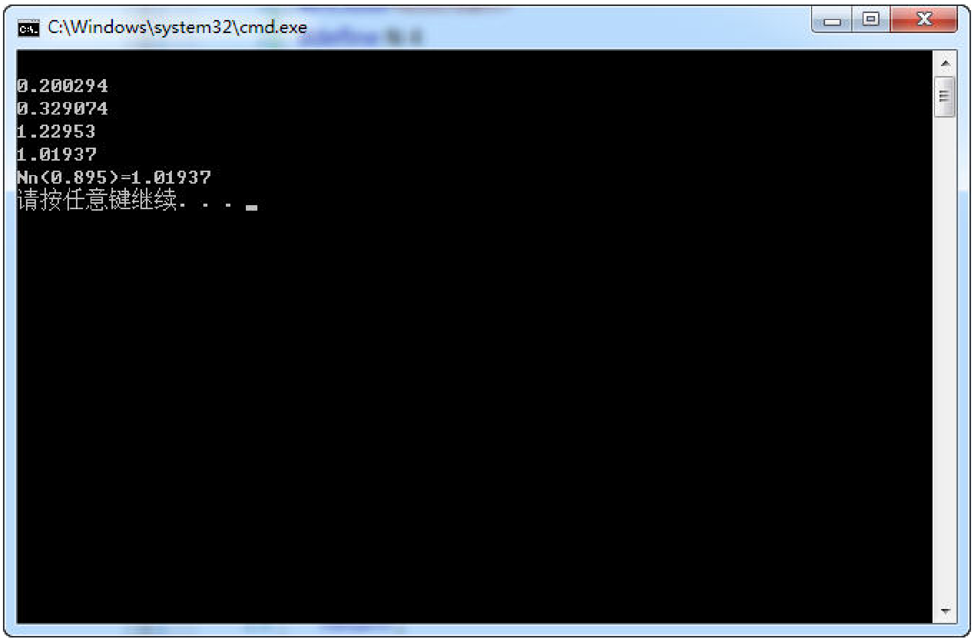

图 2.1: 运行结果

2.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 3 章 Newton-Cotes 方法

3.1 实验目的

(1) 掌握 Newton-Cotes 算法；

(2) 要求程序不断加密对积分区间的等分，自动地控制 Newton-Cotes 算法中的加速收敛过程；

(3) 编写程序，分析实验结果；

3.2 实验原理

常见教材 P164 页。

3.3 实验环境

Windows 10 + Visual Studio

3.4 实验内容

计算以下积分值：

$$\mathrm{I} = \int_0^{1/4} \sqrt{4 - \sin^2 x} dx \quad (I \approx 1.5343916) \tag{3.1}$$

$$\mathrm{I} = \int_0^{1} \frac{\sin x}{x} dx \quad (f(0) = 1, \quad I \approx 0.9460831) \tag{3.2}$$

$$\mathrm{I} = \int_0^{1} \frac{e^x}{4 + x^2} dx \tag{3.3}$$

$$\mathrm{I} = \int_0^{1} \frac{\ln(1 + x)}{1 + x^2} dx \tag{3.4}$$

3.5 样例程序

```
#include <stdio.h>
#define MAXSIZE 7
void input(double f[MAXSIZE+ 1], double a, double b, long n);
void main(void) {
    long c[MAXSIZE][MAXSIZE/2+2] = {{2, 1}, {6, 1, 4}, {8, 1, 3}, {90, 7,
        32, 12}, {288, 19, 75, 50}, {840, 41, 216, 27, 272}, {17280, 751, 3577,
        1323, 2989}};
    double a, b, f[MAXSIZE + 1], integral;
    long n, i;
    printf("\n请输入积分区间边界a, b:");
```

```c
    scanf("%lf, %lf", &a, &b);
    printf("\n请输入积分节点的个数(2~8):");
    scanf("%ld", &n);
    input(f, a, b, n);
    integral = 0;
    for(i = 0; i < n/2; i++)
        integral += (f[i]+f[n-i-1])*c[n-2][i+1]/c[n-2][0];
    if(n%2)
        integral += f[n/2]*c[n-2][n/2+1]/c[n-2][0];
    integral *= b-a;
    printf("\n积分值为=%lf", integral);
}
void input(double f[MAXSIZE+ 1], double a, double b, long n) {
    long i ;
    double h;
    h = (b - a)/ (n - 1);
    printf("\n请输入求积节点纵坐标: ");
    for(i = 0; i <= n-1; i++) {
        printf("\nx[%ld] = %lf, f[%ld] =", i, a+i*h, i);
        scanf("%lf", &f[i]);
    }
}
```

运行结果如下：

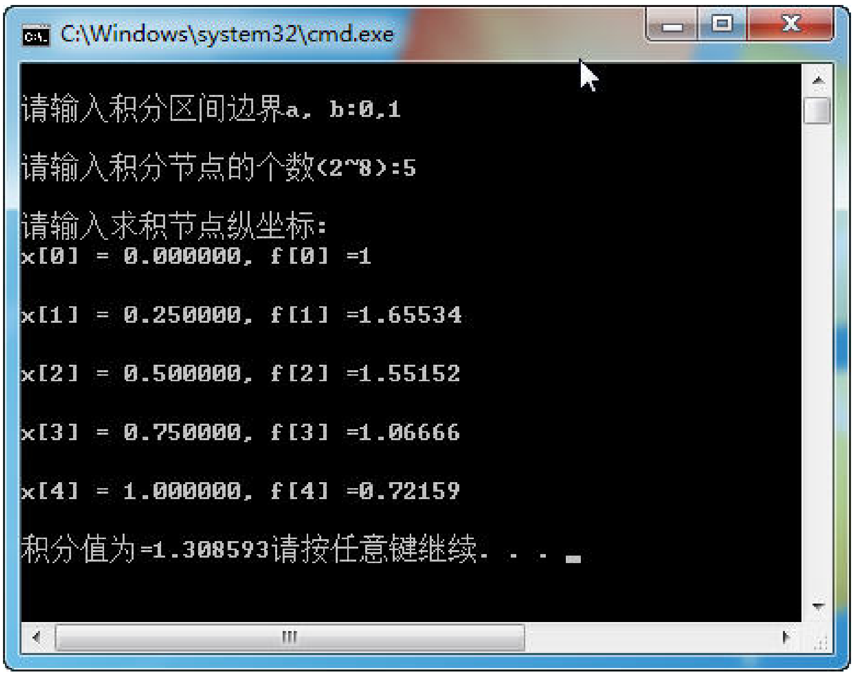

图 3.1: 运行结果

3.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 4 章　求非线性方程根的牛顿法

4.1　实验目的

(1) 掌握求非线性方程根的牛顿法；

(2) 进一步了解牛顿法的改进算法；

(3) 编写程序，分析实验结果；

4.2　实验原理

参考教材第 2 章。

4.3　实验环境

Windows 10 + Visual Studio

4.4　实验内容

用牛顿迭代法求 $xe^x - 1 = 0$ 的根，迭代初始值为 $x_0 = 0.5$。

4.5　样例程序

```c
#include <stdio.h>
#include <math.h>
double f(double x);
double df(double x);


void main(void) {
    double epsilon,x0,x1,fx0,dfx0;
    long i,maxi;
    printf("\n请输入x的精度要求：");
    scanf("%lf",&epsilon);
    printf("\n请输入迭代初值：");
    scanf("%lf",&x1);
    printf("\n请输入最大迭代次数：");
    scanf("%ld",&maxi);

    for(i=0; i<maxi; i++) {
        x0=x1;
        fx0=f(x0);
        dfx0=df(x0);
```

```
        x1=x0-fx0/dfx0;
        if(fabs(x1-x0)<=epsilon)
            break;
    }

    if(i<maxi)
        printf("\n方程f(x)=0的根x=%lf。",x1);
    else
        printf("\n迭代次数已超过上限。");
}


double f(double x) {
    return x*exp(x) - 1; /*计算并返回函数值f(x)*/
}
double df(double x) {
    return exp(x) + x*exp(x); /*计算并返回函数值f' (x)*/
}
```

运行结果如下：

图 4.1: 运行结果

4.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 5 章　解线性方程组的迭代法

5.1　实验目的

(1) 掌握雅可比迭代和 Seidel 迭代来求解方程组；

(2) 掌握常用的几种迭代格式；

(3) 编写程序实现上述迭代方法；

(4) 分析实验结果，并估计误差；

5.2　实验原理

有如下线性方程组 $Ax = b$ 如下：

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{5.1}$$

使用迭代法进行求解，主要迭代方法为雅可比迭代和 Gauss-Seidel 迭代，分别见教材第 4 章。

5.3　实验环境

Windows 10 + Visual Studio

5.4　实验内容

使用高斯-塞德尔迭代法求解下列方程组：

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2 \\ -x_1 + 10x_2 - 2x_3 = 8.3 \\ -x_1 - x_2 + 5x_3 = 4.2 \end{cases} \tag{5.2}$$

5.5　样例程序

```c
#include <stdio.h>
#include <math.h>
#define MAXSIZE 50

void input(double a[MAXSIZE][MAXSIZE],double b[MAXSIZE],long n);
```

```c
void output(double x[MAXSIZE],long n);


void main(void) {
    double a[MAXSIZE][MAXSIZE],b[MAXSIZE],x[MAXSIZE];
    double epsilon,e,s,oldx;
    long n,i,j,k,maxk;
    printf("\n请输入原方程组的阶数：");
    scanf("%ld",&n);
    input(a,b,n);
    printf("\n请输入迭代初始向量：");
    for(i=0; i<=n-1; i++)
        scanf("%lf",&x[i]);
    printf("\n请输入最大迭代次数：");
    scanf("%ld",&maxk);
    printf("\n请输入误差上限：");
    scanf("%lf",&epsilon);
    for(k=1; k<=maxk; k++) {
        e=0;
        for(i=0; i<=n-1; i++) {
            oldx=x[i];
            s=0;
            for(j=0; j<=n-1; j++)
                if(j!=i) s+=a[i][j]*x[j];
            x[i]=(b[i]-s)/a[i][i];
            if(e<fabs(oldx-x[i]))
                e=fabs(oldx-x[i]);
        }
        if(e<epsilon) break;
    }
    if(k<=maxk)
        output(x,n);
    else
        printf("\n迭代次数已超过上限。");
}

void input(double a[MAXSIZE][MAXSIZE],double b[MAXSIZE],long n) {
    long i,j;
    printf("\n请输入原方程组的增广矩阵：\n");
    for(i=0; i<=n-1; i++) {
        for(j=0; j<=n-1; j++)
            scanf("%lf",&a[i][j]);
        scanf("%lf",&b[i]);
    }
}

void output(double x[MAXSIZE],long n) {
    long i;
```

```c
    printf("\n原方程组的解向量为：\n");
    for(i=0; i<=n-1; i++)
        printf("%lf\n",x[i]);
}
```

运行结果如下：

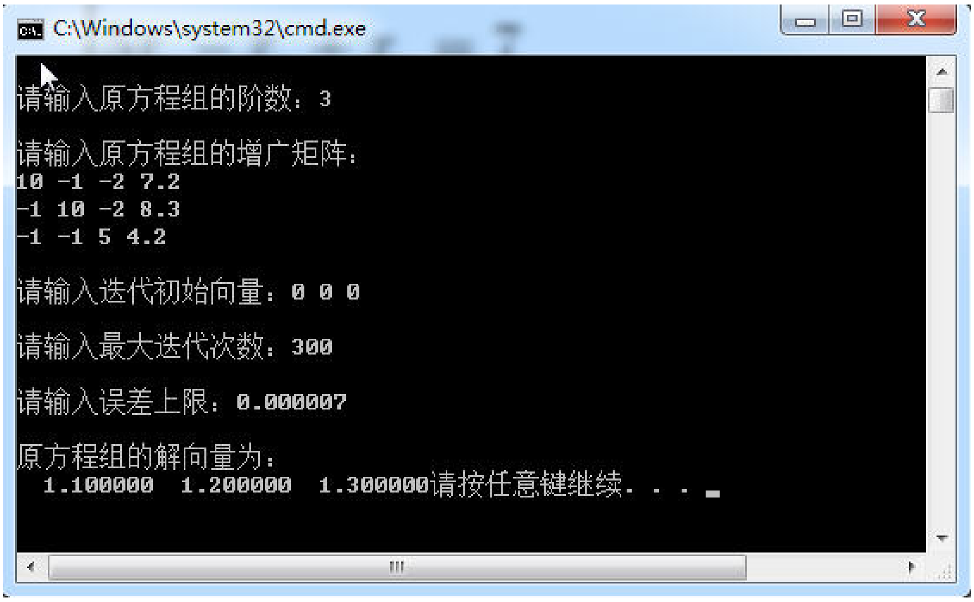

图 5.1: 运行结果

5.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 6 章　线性方程组的高斯消元法

6.1　实验目的

(1) 掌握高斯消元法求解方程组；

(2) 掌握列主元高斯消元法求解方程组；

(3) 分析实验结果，并估计误差；

6.2　实验原理

参考教材 PP52-59 页。

6.3　实验环境

Windows 10 + Visual Studio

6.4　实验内容

使用高斯消元法求解下列方程组：

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2 \\ -x_1 + 10x_2 - 2x_3 = 8.3 \\ -x_1 - x_2 + 5x_3 = 4.2 \end{cases} \tag{6.1}$$

6.5　样例程序

```c
#include <stdio.h>
#include <math.h>
#define MAXSIZE 50
void input(double a[MAXSIZE][MAXSIZE+1],long n);
void output(double x[MAXSIZE],long n);
void main(void) {
    double a[MAXSIZE][MAXSIZE+1],x[MAXSIZE],s;
    long n,i,j,k;
    printf("\n请输入原方程组的阶数：");
    scanf("%ld",&n);
    input(a,n);
    for(k=0; k<=n-2; k++)
        for(i=k+1; i<=n-1; i++) {
            a[i][k]/=-a[k][k];
            for(j=k+1; j<=n; j++)
                a[i][j]+=a[i][k]*a[k][j];
```

```
        }
    for(k=n-1; k>=0; k--) {
        s=0;
        for(j=k+1; j<=n-1; j++) s+=a[k][j]*x[j];
        x[k]=(a[k][n]-s)/a[k][k];
    }
    output(x,n);
}

void input(double a[MAXSIZE][MAXSIZE+1],long n) {
    long i,j;
    printf("\n请输入原方程组的增广矩阵: \n");
    for(i=1; i<=n; i++)
        for(j=1; j<=n+1; j++)
            scanf("%lf",&a[i-1][j-1]);
}
void output(double x[MAXSIZE],long n) {
    long k;
    printf("\n原方程组的解为: \n");
    for(k=1; k<=n; k++)
        printf(" %lf",x[k-1]);
}
```

运行结果如下：

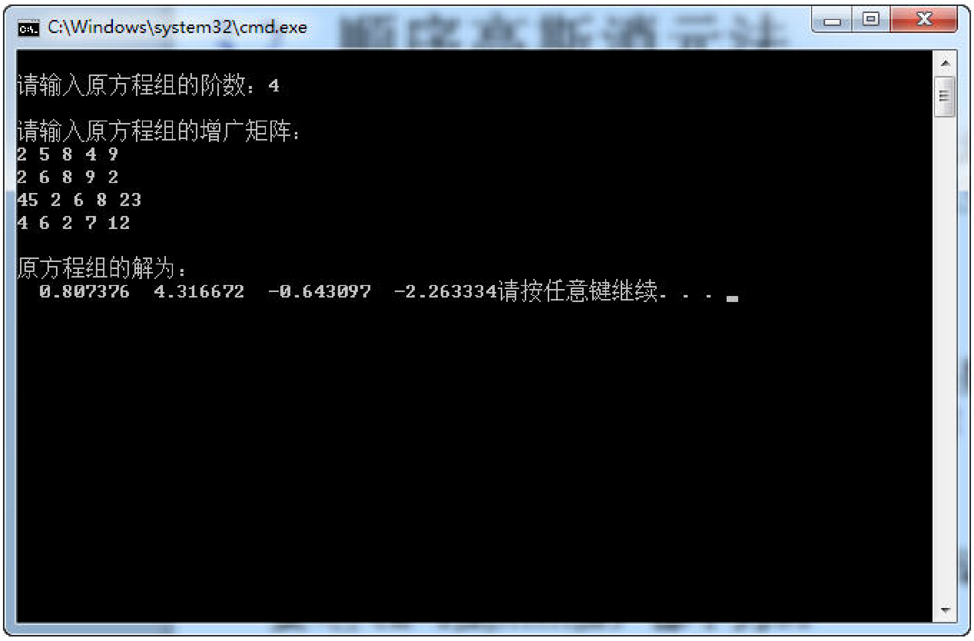

图 6.1: 运行结果

6.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 7 章　线性方程组的矩阵分解法

7.1　实验目的

(1) 掌握采用矩阵 LU 分解方法来求解线性方程组；

(2) 编程实现矩阵 LU 分解算法；

7.2　实验原理

具体参见教材第 3 章。

7.3　实验环境

Windows 10 + Visual Studio

7.4　实验内容

(1) 写出矩阵 LU 分解法解线性方程组算法，编一程序上机调试出结果，要求所编程序适用于任何一解线性方程组问题，即能解决这一类问题，而不是某一个问题。

(2) 使用矩阵 Doolittle 分解法求解下列方程组：

$$\begin{cases} 10x_1 - x_2 - 2x_3 = 7.2 \\ -x_1 + 10x_2 - 2x_3 = 8.3 \\ -x_1 - x_2 + 5x_3 = 4.2 \end{cases} \tag{7.1}$$

7.5　样例程序

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

#define MAX_N 20

int main(int argc, char* argv[]) {
    int n; //未知数个数
    int i, j, k;
    static double a[MAX_N][MAX_N], b[MAX_N], x[MAX_N], y[MAX_N];
    static double l[MAX_N][MAX_N], u[MAX_N][MAX_N];
    printf("\nInput n value(dim of Ax=b): "); //输入系数矩阵维度
    scanf("%d", &n);
    if(n >MAX_N) {
```

```
        printf("The input n is larger than MAX_N, please redefine the MAX_N.\n");
        return 1;
    }
    if(n <= 0) {
        printf("Please input a number between 1 and %d.\n", MAX_N);
        return 1;
    }
    printf("Now input the matrix a(i, j), i, j = 0, ..., %d:\n", n-1); //输入 系数矩阵
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            scanf("%lf", &a[i][j]);
    printf("Now input the matrix b(i), i = 0, ..., %d:\n", n-1); //输入常数项
    for(i=0; i<n; i++)
        scanf("%lf", &b[i]);
    for(i=0; i<n; i++)
        l[i][i] = 1;
    for(k=0; k<n; k++) {
        for(j=k; j<n; j++) { // dolittle分解
            u[k][j]=a[k][j];
            for(i=0; i<=k-1; i++)
                u[k][j]-=(l[k][i]*u[i][j]);
            printf("%f\n", u[k][j]);
        }
        for(i=k+1; i<n; i++) {
            l[i][k]=a[i][k];
            for(j=0; j<=k-1; j++)
                l[i][k]-=(l[i][j]*u[j][k]);
            l[i][k]/=u[k][k];
            printf("%f\n", l[i][k]);
        }
    }

    for(i=0; i<n; i++) { // 解Ly = b
        y[i] = b[i];
        for(j=0; j<=i-1; j++)
            y[i] -= (l[i][j]*y[j]);
    }

    for(i=n-1; i>=0; i--){ // 解UX = Y
        x[i]=y[i];
        for(j=i+1; j<n; j++)
            x[i] -= (u[i][j]*x[j]);
        x[i]/=u[i][i];
    }

    printf("Solve...x_i = \n"); // 输出结果
    for(i=0; i<n; i++)
        printf("%f\n", x[i]);
```

```
    system(”pause”);
    return 0;
}
```

运行结果如下：

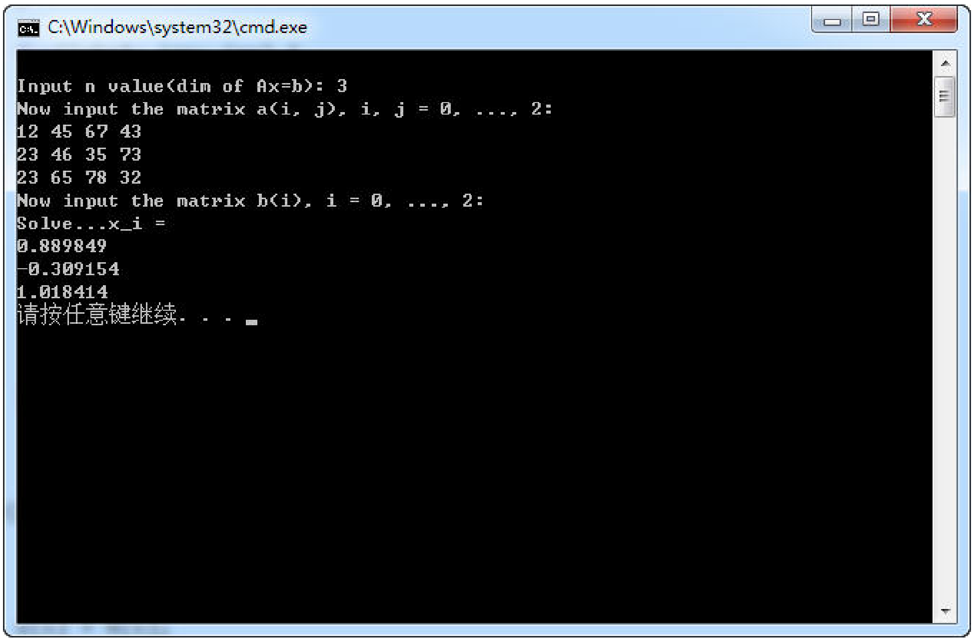

图 7.1: 运行结果

7.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

第 8 章　常微分方程求解算法

8.1　实验目的

(1) 掌握采用欧拉法来求解常微分方程；

(2) 掌握采用改进的欧拉法来求解常微分方程；

(3) 编程实现上述两个算法；

8.2　实验原理

具体参见教材第 8 章。

8.3　实验环境

Windows 10 + Visual Studio

8.4　实验内容

(1) 写出欧拉法或改进的欧拉法来求解常微分方程，编程序上机调试出结果。

(2) 使用常微分方程例子如下：

$$\begin{cases} y' = y - \frac{2x}{y}(0 < x < 1) \\ y(0) = 1 \end{cases} \tag{8.1}$$

8.5　样例程序

```
#include<iostream>
#define N 10

using namespace std;

void modEuler(float (*f1)(float,float),float x0,float y0,float xn,int n) {
    int i;
    float yp,yc,x=x0,y=y0,h=(xn-x0)/n;
    // cout<<"x[0]="<<x<<'\t'<<"y[0]"<<y<<endl;
    for(i=1; i<=n; i++) {
        yp=y+h*f1(x,y);
        x=x0+i*h;
        yc=y+h*f1(x,yp);
        y=(yp+yc)/2.0;
        cout<<"x["<<i<<"]="<<x<<" y["<<i<<"]="<<y<<endl;
```

```
    }
}
void main() {

    float xn=0.0,x0=-1.0,y0=3.0;
    float f1(float ,float);
    modEuler(f1,x0,y0,xn,N);
}
float f1(float x,float y) {
    return 3*x-2*y*y-12;
}
```

运行结果如下：

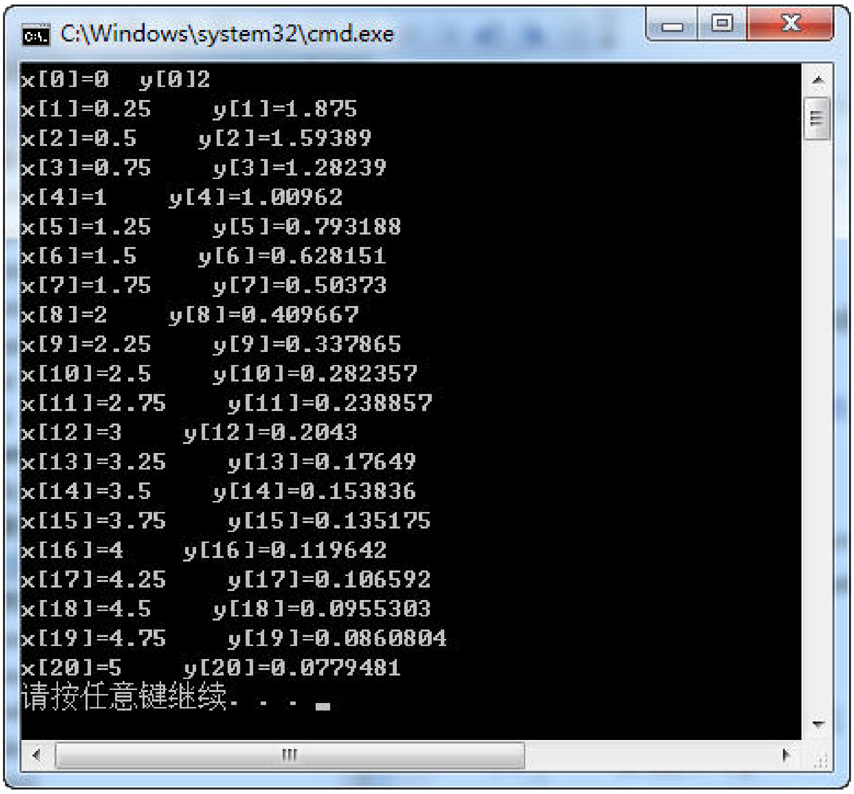

图 8.1: 运行结果

8.6 实验报告

(1) 包括实验主要内容；

(2) 理解算法并编写程序，调试成功；

(3) 撰写实验报告，实验报告纸上写上自己的代码；

参考文献

[1] 靳天飞等. *计算方法 (C 语言版)*. 北京：清华大学出版社，ISBN(9787302221753)，2010.