



北京化工大学

BEIJING UNIVERSITY OF CHEMICAL TECHNOLOGY

COMPUTING METHODS

计算方法实验指导书

Cheng Yong

目录

第 1 章 Lagrange 插值方法	1
1.1 实验目的	1
1.2 实验原理	1
1.3 实验环境	1
1.4 实验内容	1
1.5 样例程序	2
1.6 实验报告	3
第 2 章 牛顿插值方法	4
2.1 实验目的	4
2.2 实验原理	4
2.3 实验环境	4
2.4 实验内容	4
2.5 样例程序	4

2.6	实验报告	5
第 3 章	Newton-Cotes 方法	7
3.1	实验目的	7
3.2	实验原理	7
3.3	实验环境	7
3.4	实验内容	7
3.5	样例程序	7
3.6	实验报告	8
3.7	实验分析	9
第 4 章	求非线性方程根的牛顿法	10
4.1	实验目的	10
4.2	实验原理	10
4.3	实验环境	10
4.4	实验内容	10
4.5	样例程序	10
4.6	实验报告	11
第 5 章	解线性方程组的迭代法	12
5.1	实验目的	12
5.2	实验原理	12
5.3	实验环境	12
5.4	实验内容	12
5.5	样例程序	12
5.6	实验报告	14
第 6 章	线性方程组的高斯消元法	15
6.1	实验目的	15
6.2	实验原理	15
6.3	实验环境	15
6.4	实验内容	15
6.5	样例程序	15
6.6	实验报告	16
第 7 章	线性方程组的矩阵分解法	18
7.1	实验目的	18
7.2	实验原理	18

7.3	实验环境	18
7.4	实验内容	18
7.5	样例程序	18
7.6	实验报告	20
第 8 章	常微分方程求解算法	21
8.1	实验目的	21
8.2	实验原理	21
8.3	实验环境	21
8.4	实验内容	21
8.5	样例程序	21
8.6	实验报告	22

创建日期：2019 年 11 月 13 日
 更新日期：2020 年 10 月 30 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

```

1  #include <stdio.h>
2  #include <math.h>
3  #define MAXSIZE 50
4
5  void input(double x[MAXSIZE], double y[MAXSIZE], long n);
6
7  void main(void) {
8      double x[MAXSIZE], y[MAXSIZE], __x, __y, t;
9      long n, i, j;
10     printf("\n请输入插值节点的个数: ");
11     scanf("%ld", &n);
12     input(x, y, n);
13     printf("\n请输入插值点: ");
14     scanf("%lf", &__x);
15     __y = 0;
16
17     for(i = 0; i <= n - 1; i++) {
18         t = 1;
19         for(j = 0; j <= n-1; j++)
20             if(j != i)
21                 t*=(__x-x[j])/(x[i] - x[j]);
22         __y+= t*y[i];
23     }
24     printf("\n插值点(x,y) = (%lf, %lf)。", __x, __y);
25 }
26
27 void input(double x[MAXSIZE], double y[MAXSIZE], long n) {
28     long i;
29     for(i = 0; i <= n-1; i++) {
30         printf("\n请输入插值节点x[%ld], y[%ld]:", i,i);
31         scanf("%lf, %lf", &x[i], &y[i]);
32     }
33 }

```

运行结果如下:

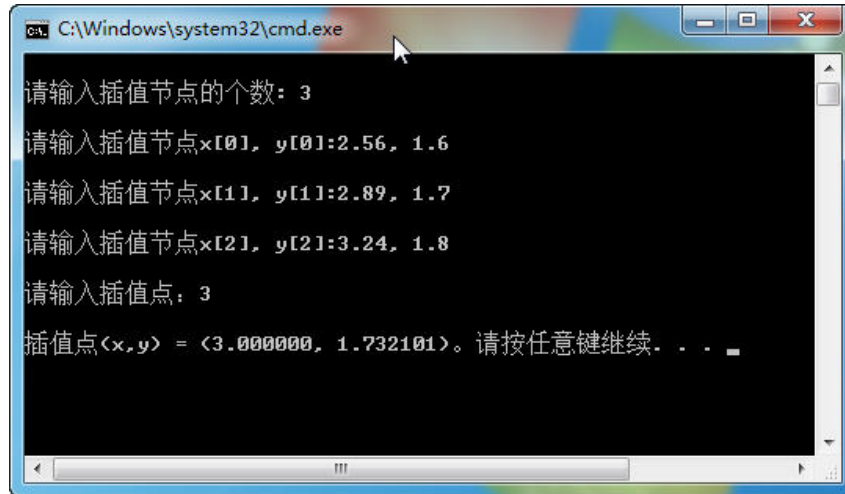


图 1.1: 运行结果

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

```
1  #include<iostream>
2  #define N 4
3
4  using namespace std;
5
6  void difference(float *x, float *y, int n) {
7      float *f;
8      int k, i;
9      f = new float[(n*sizeof(float))];
10     for(k=1; k<=n; k++) {
11         f[0]=y[k];
12         for(i=0; i<k; i++)
```

```
13         f[i+1]=(f[i]-y[i])/(x[k]-x[i]);
14         y[k]=f[k];
15     }
16     cout<<endl;
17     return ;
18 }
19
20 void main() {
21     int i;
22     float varx =0.895,b;
23     float x[N+1] = {0.4,0.55,0.65,0.8,0.9};
24     float y[N+1] = {0.41075,0.57815,0.69675,0.88811,1.02652};
25     difference(x,(float *)y,N);
26     b =y[N];
27     for (i=N-1; i>=0; i--) {
28         b = b*(varx-x[i])+y[i];
29         cout<<b<<endl;
30     }
31     cout<<"Nn("<<varx<<")="<<b<<endl;
32 }
```

运行结果如下：

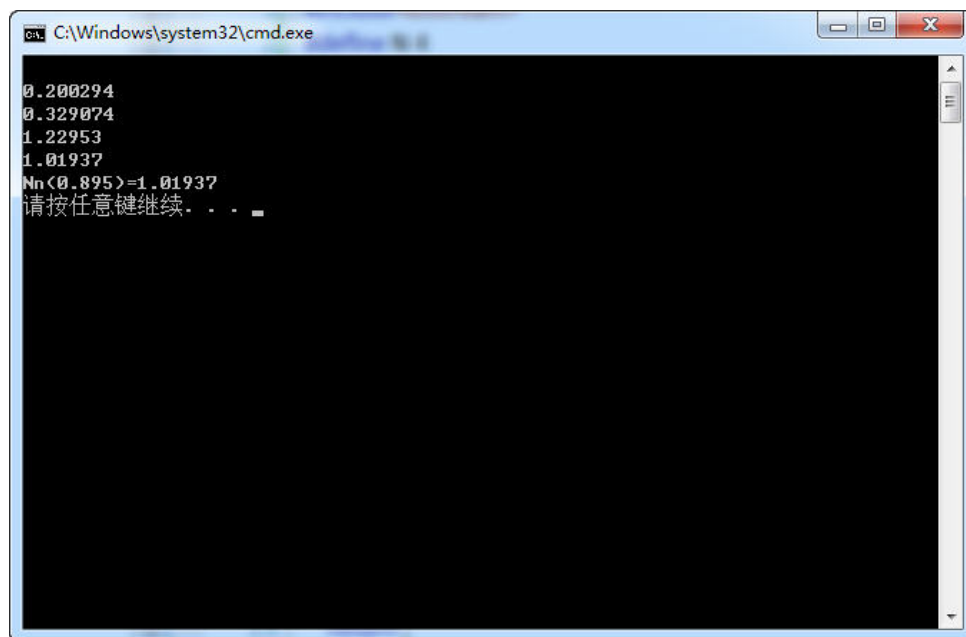


图 2.1: 运行结果

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:

$$I = \int_0^1 \frac{\sin x}{x} dx \quad (f(0) = 1, \quad I \approx 0.9460831) \quad (3.1)$$

$$I = \int_0^1 \frac{e^x}{4+x^2} dx \quad (3.2)$$

$$I = \int_0^1 \frac{\ln(1+x)}{1+x^2} dx \quad (3.3)$$

3.5 样例程序

```
1  #include <stdio.h>
2  #define MAXSIZE 7
3  void input(double f[MAXSIZE+ 1], double a, double b, long n);
4  void main(void) {
5      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}};
6      double a, b, f[MAXSIZE + 1], integral;
7      long n, i;
8      printf("\n请输入积分区间边界a, b:");
9      scanf("%lf, %lf", &a, &b);
10     printf("\n请输入积分节点的个数(2~8):");
```

```

11     scanf("%ld", &n);
12     input(f, a, b, n);
13     integral = 0;
14     for(i = 0; i < n/2; i++)
15         integral += (f[i]+f[n-i-1])*c[n-2][i+1]/c[n-2][0];
16     if(n%2)
17         integral += f[n/2]*c[n-2][n/2+1]/c[n-2][0];
18     integral *= b-a;
19     printf("\n积分值为=%lf", integral);
20 }
21 void input(double f[MAXSIZE+ 1], double a, double b, long n) {
22     long i ;
23     double h;
24     h = (b - a)/ (n - 1);
25     printf("\n请输入求积节点纵坐标: ");
26     for(i = 0; i <= n-1; i++) {
27         printf("nx[%ld] = %lf, f[%ld] =", i, a+i*h, i);
28         scanf("%lf", &f[i]);
29     }
30 }

```

运行结果如下:

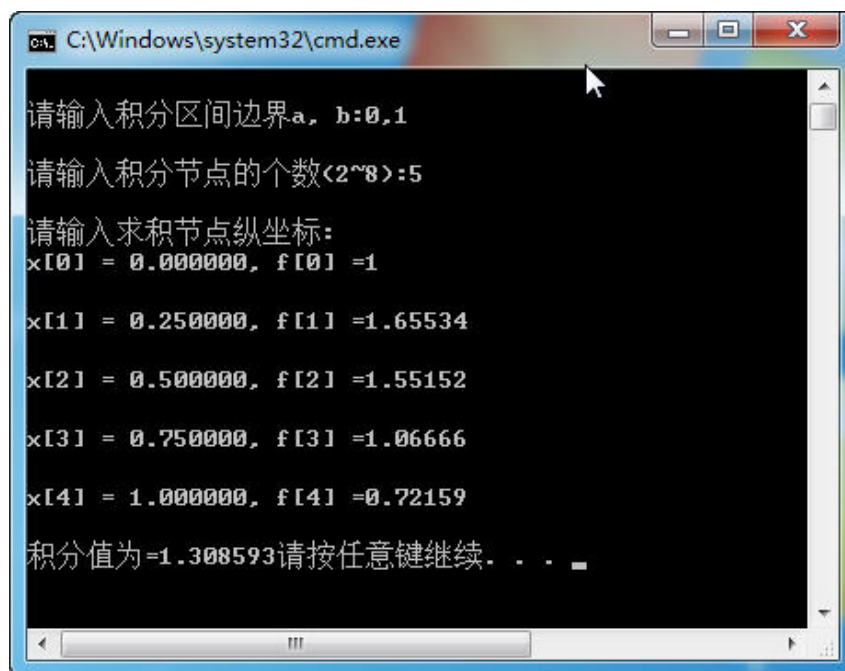


图 3.1: 运行结果

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；

3.7 实验分析

函数 (3.1) 计算过程如下：

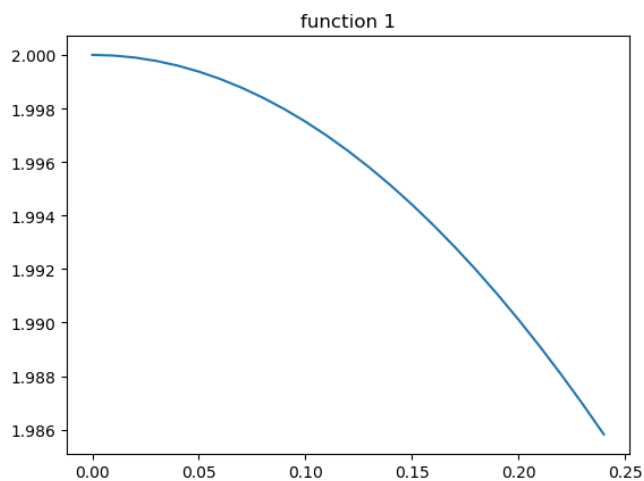


图 3.2: 函数 (3.1) 的图像

按照 Cotes 公式等距结点取样要求，得到函数值如下：

0	0.0625	0.125	0.1875	0.25
2.	1.99902447	1.99611027	1.99129451	1.98463883

表 3.1: 样本值

应用 Newton-Cotes 公式得到近似积分值为：

$$I = 0.4987111175077971 \quad (3.4)$$

积分精确值为 0.4987111175752327，由此可见两者是非常接近的。

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

```
1  #include <stdio.h>
2  #include <math.h>
3  double f(double x);
4  double df(double x);
5
6
7  void main(void) {
8      double epsilon, x0, x1, fx0, dfx0;
9      long i, maxi;
10     printf("\n请输入x的精度要求: ");
11     scanf("%lf", &epsilon);
12     printf("\n请输入迭代初值: ");
13     scanf("%lf", &x1);
14     printf("\n请输入最大迭代次数: ");
15     scanf("%ld", &maxi);
16
17     for(i=0; i<maxi; i++) {
18         x0=x1;
19         fx0=f(x0);
20         dfx0=df(x0);
```

```
21     x1=x0-fx0/dfx0;  
22     if(fabs(x1-x0)<=epsilon)  
23         break;  
24 }  
25  
26 if(i<maxi)  
27     printf("\n方程f(x)=0的根x=%lf。",x1);  
28 else  
29     printf("\n迭代次数已超过上限。");  
30 }  
31  
32  
33 double f(double x) {  
34     return x*exp(x) - 1; /*计算并返回函数值f(x)*/  
35 }  
36 double df(double x) {  
37     return exp(x) + x*exp(x); /*计算并返回函数值f'(x)*/  
38 }
```

运行结果如下：

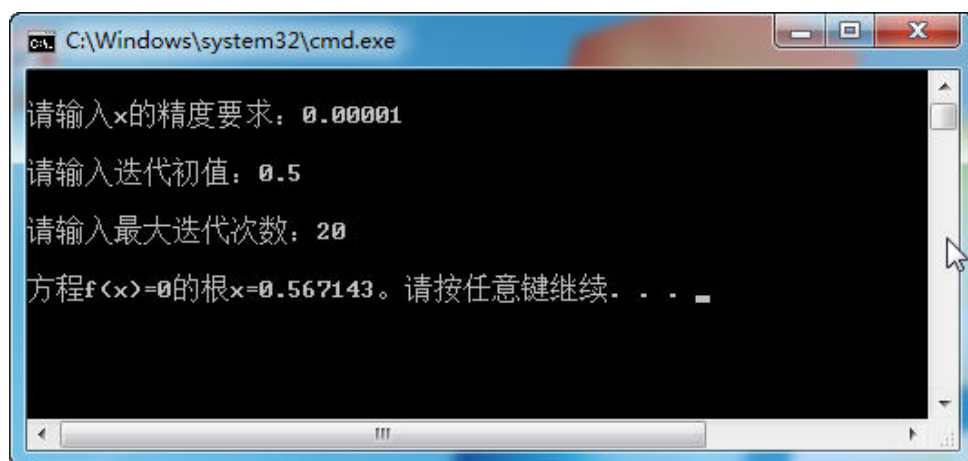


图 4.1: 运行结果

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:

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

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-赛德尔迭代法求解下列方程组:

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

5.5 样例程序

```
1 #include <stdio.h>
2 #include <math.h>
3 #define MAXSIZE 50
4
5 void input(double a[MAXSIZE][MAXSIZE], double b[MAXSIZE], long n);
```

```

6 void output(double x[MAXSIZE],long n);
7
8
9 void main(void) {
10     double a[MAXSIZE][MAXSIZE],b[MAXSIZE],x[MAXSIZE];
11     double epsilon,e,s,oldx;
12     long n,i,j,k,maxk;
13     printf("\n请输入原方程组的阶数: ");
14     scanf("%ld",&n);
15     input(a,b,n);
16     printf("\n请输入迭代初始向量: ");
17     for(i=0; i<=n-1; i++)
18         scanf("%lf",&x[i]);
19     printf("\n请输入最大迭代次数: ");
20     scanf("%ld",&maxk);
21     printf("\n请输入误差上限: ");
22     scanf("%lf",&epsilon);
23     for(k=1; k<=maxk; k++) {
24         e=0;
25         for(i=0; i<=n-1; i++) {
26             oldx=x[i];
27             s=0;
28             for(j=0; j<=n-1; j++)
29                 if(j!=i) s+=a[i][j]*x[j];
30             x[i]=(b[i]-s)/a[i][i];
31             if(e<fabs(oldx-x[i]))
32                 e=fabs(oldx-x[i]);
33         }
34         if(e<epsilon) break;
35     }
36     if(k<=maxk)
37         output(x,n);
38     else
39         printf("\n迭代次数已超过上限。");
40 }
41
42 void input(double a[MAXSIZE][MAXSIZE],double b[MAXSIZE],long n) {
43     long i,j;
44     printf("\n请输入原方程组的增广矩阵: \n");
45     for(i=0; i<=n-1; i++) {
46         for(j=0; j<=n-1; j++)
47             scanf("%lf",&a[i][j]);
48         scanf("%lf",&b[i]);
49     }
50 }
51
52 void output(double x[MAXSIZE],long n) {
53     long i;

```



```
54     printf("\n原方程组的解向量为: \n");
55     for(i=0; i<=n-1; i++)
56         printf(" %lf\n",x[i]);
57 }
```

运行结果如下:

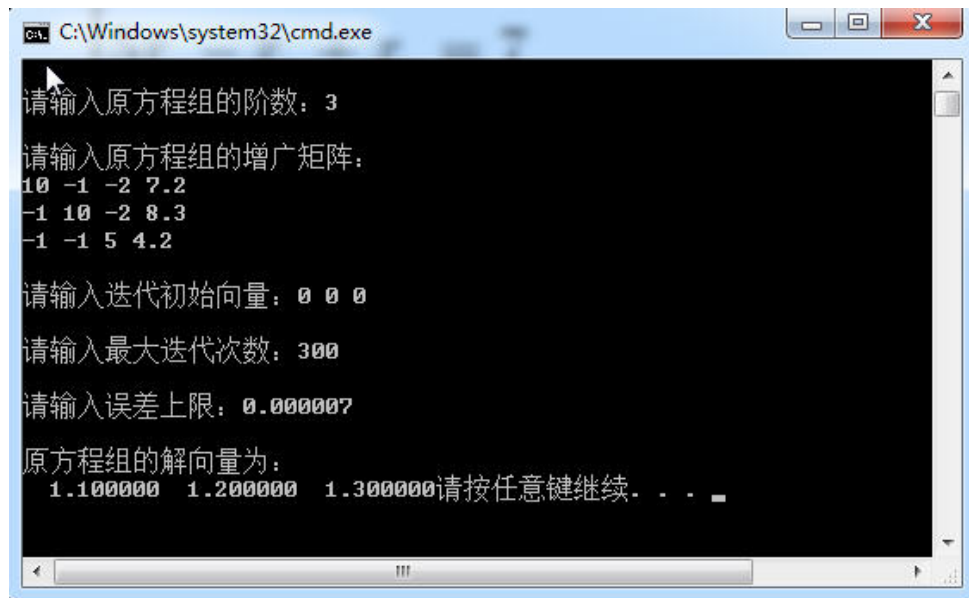


图 5.1: 运行结果

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:

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

6.5 样例程序

```

1  #include <stdio.h>
2  #include <math.h>
3  #define MAXSIZE 50
4  void input(double a[MAXSIZE][MAXSIZE+1], long n);
5  void output(double x[MAXSIZE], long n);
6  void main(void) {
7      double a[MAXSIZE][MAXSIZE+1], x[MAXSIZE], s;
8      long n, i, j, k;
9      printf("\n请输入原方程组的阶数: ");
10     scanf("%ld", &n);
11     input(a, n);
12     for(k=0; k<=n-2; k++)
13         for(i=k+1; i<=n-1; i++) {
14             a[i][k] /= -a[k][k];
15             for(j=k+1; j<=n; j++)
16                 a[i][j] += a[i][k] * a[k][j];

```

```

17     }
18     for(k=n-1; k>=0; k--) {
19         s=0;
20         for(j=k+1; j<=n-1; j++) s+=a[k][j]*x[j];
21         x[k]=(a[k][n]-s)/a[k][k];
22     }
23     output(x,n);
24 }
25
26 void input(double a[MAXSIZE][MAXSIZE+1],long n) {
27     long i,j;
28     printf("\n请输入原方程组的增广矩阵: \n");
29     for(i=1; i<=n; i++)
30         for(j=1; j<=n+1; j++)
31             scanf("%lf",&a[i-1][j-1]);
32 }
33 void output(double x[MAXSIZE],long n) {
34     long k;
35     printf("\n原方程组的解为: \n");
36     for(k=1; k<=n; k++)
37         printf(" %lf",x[k-1]);
38 }

```

运行结果如下:

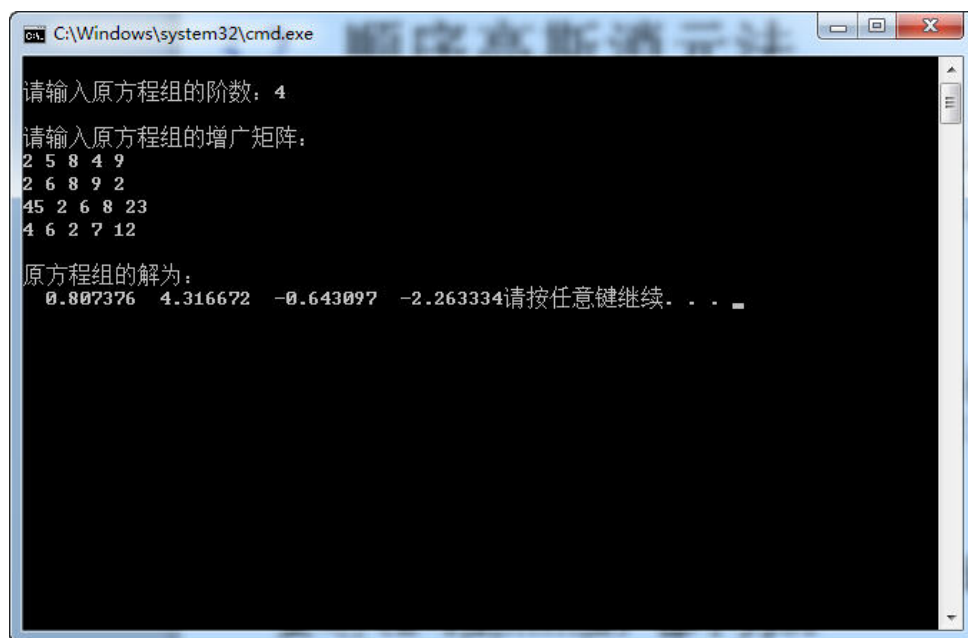


图 6.1: 运行结果

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，而不是某一个具体问题。
- (2) 使用矩阵 Doolittle 分解法求解下列方程组：

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

7.5 样例程序

```

1  #include <stdio.h>
2  #include <iostream>
3  #include <math.h>
4
5  #define MAX_N 20
6
7  int main(int argc, char* argv[]) {
8      int n; // 未知数个数
9      int i, j, k;
10     static double a[MAX_N][MAX_N], b[MAX_N], x[MAX_N], y[MAX_N];
11     static double l[MAX_N][MAX_N], u[MAX_N][MAX_N];
12     printf("\nInput n value(dim of Ax=b): "); //输入系数矩阵维度
13     scanf("%d", &n);
14     if(n > MAX_N) {

```

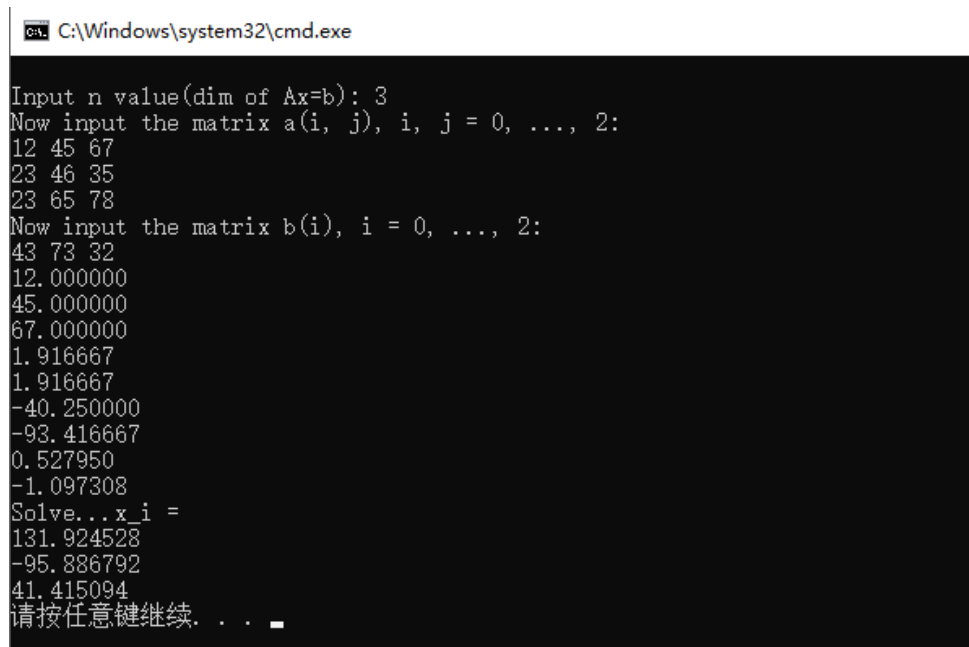
```

15     printf("The input n is larger than MAX_N, please redefine the MAX_N.\n");
16     return 1;
17 }
18 if(n <= 0) {
19     printf("Please input a number between 1 and %d.\n", MAX_N);
20     return 1;
21 }
22 printf("Now input the matrix a(i, j), i, j = 0, ..., %d:\n", n-1); //输入 系数矩阵
23 for (i=0; i<n; i++)
24     for (j=0; j<n; j++)
25         scanf("%lf", &a[i][j]);
26 printf("Now input the matrix b(i), i = 0, ..., %d:\n", n-1); //输入常数项
27 for(i=0; i<n; i++)
28     scanf("%lf", &b[i]);
29 for(i=0; i<n; i++)
30     l[i][i] = 1;
31 for(k=0; k<n; k++) {
32     for(j=k; j<n; j++) { // dolittle分解
33         u[k][j]=a[k][j];
34         for(i=0; i<=k-1; i++)
35             u[k][j]-=(l[k][i]*u[i][j]);
36         printf("%f\n", u[k][j]);
37     }
38     for(i=k+1; i<n; i++) {
39         l[i][k]=a[i][k];
40         for(j=0; j<=k-1; j++)
41             l[i][k]-=(l[i][j]*u[j][k]);
42         l[i][k]/=u[k][k];
43         printf("%f\n", l[i][k]);
44     }
45 }
46
47 for(i=0; i<n; i++) { // 解Ly = b
48     y[i] = b[i];
49     for(j=0; j<=i-1; j++)
50         y[i] -= (l[i][j]*y[j]);
51 }
52
53 for(i=n-1; i>=0; i--){ // 解UX = Y
54     x[i]=y[i];
55     for(j=i+1; j<n; j++)
56         x[i] -= (u[i][j]*x[j]);
57     x[i]/=u[i][i];
58 }
59
60 printf("Solve...x_i = \n"); // 输出结果
61 for(i=0; i<n; i++)
62     printf("%f\n", x[i]);

```

```
63     system("pause");
64     return 0;
65 }
```

运行结果如下：



```
C:\Windows\system32\cmd.exe

Input n value(dim of Ax=b): 3
Now input the matrix a(i, j), i, j = 0, ..., 2:
12 45 67
23 46 35
23 65 78
Now input the matrix b(i), i = 0, ..., 2:
43 73 32
12.000000
45.000000
67.000000
1.916667
1.916667
-40.250000
-93.416667
0.527950
-1.097308
Solve...x_i =
131.924528
-95.886792
41.415094
请按任意键继续. . .
```

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：

$$\begin{cases} y' = 3x - 2y^2 - 12 (0 < x < 5) \\ y(0) = 2 \end{cases} \quad (8.1)$$

8.5 样例程序

```

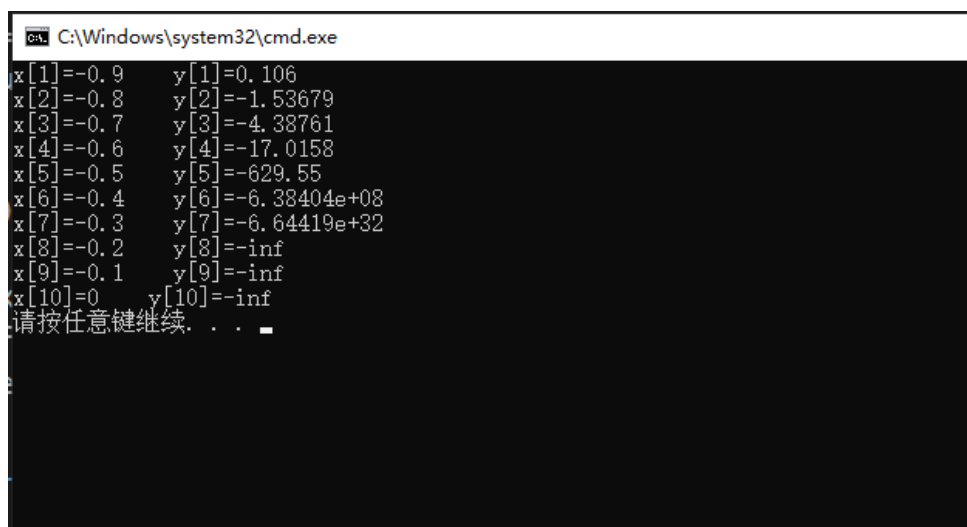
1  #include<iostream>
2  #define N 10
3
4  using namespace std;
5
6  void modEuler(float (*f1)(float, float), float x0, float y0, float xn, int n) {
7      int i;
8      float yp, yc, x=x0, y=y0, h=(xn-x0)/n;
9      // cout<<"x[0]="<<x<<"t'<<"y[0]"<<y<<endl;
10     for(i=1; i<=n; i++) {
11         yp=y+h*f1(x, y);
12         x=x0+i*h;
13         yc=y+h*f1(x, yp);
14         y=(yp+yc)/2.0;
15         cout<<"x["<<i<<"]="<<x<<" y["<<i<<"]="<<y<<endl;
16     }

```



```
17 }  
18 void main() {  
19  
20     float xn=0.0,x0=-1.0,y0=2.0;  
21     float f1(float ,float);  
22     modEuler(f1,x0,y0,xn,N);  
23 }  
24 float f1(float x,float y) {  
25     return 3*x-2*y*y-12;  
26 }
```

运行结果如下：



```
C:\Windows\system32\cmd.exe  
x[1]=-0.9      y[1]=0.106  
x[2]=-0.8      y[2]=-1.53679  
x[3]=-0.7      y[3]=-4.38761  
x[4]=-0.6      y[4]=-17.0158  
x[5]=-0.5      y[5]=-629.55  
x[6]=-0.4      y[6]=-6.38404e+08  
x[7]=-0.3      y[7]=-6.64419e+32  
x[8]=-0.2      y[8]=-inf  
x[9]=-0.1      y[9]=-inf  
x[10]=0        y[10]=-inf  
请按任意键继续 . . .
```

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.