西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

西北工业大学航空学院

《计算方法》课程实习报告

2019 年 4 月 18 日

指导教师：陶亮老师

学生姓名：冯铮浩

学院专业：航空学院 飞行器设计与工程

所在年级：2017 级

所在班级：01011704 班

学生学号：2017300281

联系电话：13815233646

摘 要

电子数字计算机的高速发展，为现代工程领域众多复杂科学技术问题的求解提供了强有力的条件。本文结合西北工业大学航空学院《计算方法》课程中所学的各种计算理论与算法思想，主要采用基于 C 语言的基础编程方式，设计不同的求解算法，较好地实现四个典型计算方法问题的求解与分析。同时，本文提供各问题的完整 C 程序源代码与注释及使用简明说明书，方便相关人员参考使用。

针对问题一，编程实现利用最小二乘法以双曲线函数与指数函数形式分别拟合函数曲线，并利用 MATLAB 软件绘制拟合曲线。首先对经验公式进行变形与变量代换，将经验公式转化为含待定常数的线性形式，然后通过求解矛盾方程组，得到待求拟合函数中的未知参数，最后输出拟合结果与均方误差以及最大误差，并绘制含原始数据点与拟合曲线的图像。由理论分析与程序计算的结果，可以观察到，指数函数拟合效果明显好于采用双曲线函数拟合。

针对问题二，编程实现利用建立在简单迭代公式基础上的 Seidel 迭代法求解三阶非齐次线性方程组。其求解收敛速度更快，计算效率更高。首先对该问题的 Seidel 迭代法的理论求解方式进行讨论，并设计可适用于不同非齐次线性矩阵方程求解的通用 C 语言程序，采用交互式界面。程序算法核心是使用 **While** 循环判断迭代过程是否应该退出，以此保证程序正确稳定地执行。

针对问题三，编程实现利用直接三角分解法中的 Doolittle 三角分解法求解三阶非齐次线性方程组。首先结合题目数据讨论直接三角分解法的理论求解，并设计可适用于任何非齐次线性矩阵方程求解的通用 C 语言程序。将数学矩阵符号用数组表示，并根据 Doolittle 三角分解理论推导公式以及回带公式，列出边界控制条件，对待求的分解矩阵中各行、列元素值进行循环累加计算，得到方程组解集以及三角分解的单位下三角矩阵 $\boldsymbol{L}$ 与上三角矩阵 $\boldsymbol{U}$。

针对问题四，编程实现使用古典显示格式求解边值问题的数值解。本文首先结合题目给出的具体混合边值问题，进行理论求解。设计可适用于不同矩形网格划分条件的古典显示差分求数值解的通用 C 语言程序，完成边界条件的初始化，并通过递推公式求解 0~n 层上每个单元节点的值，获得结果与理论值较为吻合。

关键词：最小二乘法、Seidel 迭代法、Doolittle 三角分解法、古典显示格式差分

目录

1 实习问题一：利用最小二乘法拟合函数曲线

1.1 问题叙述

某化学反应，根据试验所得生物的浓度与时间关系如表 1 所示，求浓度 y 与时间 t 的拟合曲线。（分别用双曲线函数和指数函数拟合曲线，并进行误差计算）

表 1 试验所得生物的浓度与时间关系

t(分)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$y\times10^{-3}$	4.00	6.40	8.00	8.80	9.22	9.50	9.70	9.86	10.00	10.20	10.32	10.42	10.50	10.55	10.58	10.60

1.2 问题分析

问题一给出一组试验得到的离散数据，要求用不同的常见函数进行曲线拟合。因此，任务即为寻求函数的一个近似表达式 $y=\varphi(x)$，使函数曲线能够连续地反映离散数据点的变化趋势，而不一定过全部的点 (x_i,y_i)。这类问题，通常采用**最小二乘法**解矛盾方程组来求解未知函数中的参数。需要注意的是，题目中给出的经验公式（即双曲线函数与指数函数）待定系数是非线性形式，如果按多项式拟合的方法，将得到难以求解的非线性方程组。因此，本文首先对经验公式进行变形与变量代换，将经验公式转化为待定常数的线性形式，再使用最小二乘法拟合。

1.3 理论求解

将数据点标于坐标纸上，可以观察到离散数据点近似符合双曲线函数与指数函数形式。下面仅分别对两种拟合函数的求解进行讨论（不包括误差计算）。

1.3.1 情形 1：双曲线函数拟合理论求解[1]

设双曲线函数形式为：$\frac{1}{y}=a+\frac{b}{t}$，即有

$$y=\frac{t}{at+b} \tag{1}$$

进行函数变形。不妨假设

$$\bar{y}=\frac{1}{y},\ x=\frac{1}{t} \tag{2}$$

又由数据表 t，y 可得到 x，$\bar{y}$，则设变形后的函数为

$$S_1(x)=a+bx \tag{3}$$

利用该函数拟合数据 $(x_i,\overline{y_i})$ $(\mathrm{i}=1,2,\cdots,16)$。带入数据，可得矛盾方程组，并进一步写出正则方程组，即

$$\begin{cases} 16a+3.38073b=1.8372\times10^3 \\ 3.38073a+1.58435b=0.52886\times10^3 \end{cases} \tag{4}$$

求解得到：$a=80.6621, b=161.6822$，

带入待求双曲线函数可得拟合函数为：

$$\frac{1}{y}=80.6621+\frac{161.6822}{t} \tag{5}$$

$$F^{(1)}(t)=\frac{t}{80.6621t+161.6822}. \tag{6}$$

1.3.2 情形 2：指数函数拟合理论求解

设指数函数形式为：$y=ae^{\frac{b}{t}}$，

进行函数变形。对两边同取对数，可得：

$$\ln y=\ln a+\frac{b}{t} \tag{7}$$

不妨假设

$$\hat{y}=\ln y\ ,\ x=\frac{1}{t}, A=\ln a \tag{8}$$

又由数据表 t，y 可得到 x，$\hat{y}$，则设变形后的函数为

$$S_2(x)=A+bx \tag{9}$$

同情形一，求解矛盾方程组得到：$A=-4.48072, b=-1.0567$，

带入待求指数函数可得拟合函数为：

$$F^{(2)}(t)=11.3253\times10^{-3}e^{\frac{-1.0567}{t}}. \tag{10}$$

1.4 程序设计

1.4.1 C 语言程序算法设计

本问题利用计算机编程实现最小二乘法拟合函数曲线的关键点如下：

- **对原始数据进行数量级变化等预处理工作；**
- **有效保存经验公式中已变形的 *x*，*y* 变量；**
- **利用直接三角分解（Doolittle）方法求解矛盾方程组；**
- **友好的界面设计，采用文件输入输出；**

具体算法设计如下：

1) 数据预处理

将题设所给的 *t*，*y* 数据点保存在“Problem_1.dat”文件中，采用文件读取输入方式，进行变量、数组定义及初始化。注意到，*y* 变量的实际数量级为10^{-3}，故须进行数据放缩处理。程序中浮点数及数组均使用 Double 双精度类型。

2) 经验公式变形化参数计算

结合理论推导得到的公式变形方式，在读取原始数据的过程中，直接进行参数变形，并将数据保存于 *xx*，*yy* 两数组内，供后续程序处理运算。

3) 计算系数与常数矩阵

首先通过循环数据导入，计算矛盾方程组 ***Ax=b*** 的 ***A***，***b*** 矩阵，并利用数组进行保存。同时通过矩阵转置，求解正则方程组中的系数矩阵 $\boldsymbol{A^TA}$ 与常数矩阵 $\boldsymbol{A^Tb}$。

4) 利用 Doolittle 三角分解方法求解正则方程组；

Doolittle 三角分解公式如下：

$$\left.\begin{aligned} u_{1j} &= a_{1j} \quad (j=1,2,\cdots,n) \\ l_{i1} &= \frac{a_{i1}}{u_{11}} \quad (i=1,2,\cdots,n) \\ u_{kj} &= a_{kj} - \sum_{m=1}^{k-1} l_{km}u_{mj} \quad (j=k,k+1,\cdots,n) \\ l_{ik} &= \frac{(a_{ik} - \sum_{m=1}^{k-1} l_{im}u_{mk})}{u_{kk}} \quad (i=k+1,k+2,\cdots,n) \end{aligned}\right\} \tag{11}$$

其中 $k=2,3,\cdots,n$；

由回带公式求解三角方程组，有

$$\left.\begin{aligned} y_1 &= \frac{b_1}{l_{11}} \\ y_k &= \frac{(b_k - \sum_{j=1}^{k-1} l_{kj} y_j)}{l_{kk}} \quad (k = 2,3,\cdots,n) \end{aligned}\right\} \tag{12}$$

$$\left.\begin{aligned} x_n &= \frac{y_n}{u_{nn}} \\ x_k &= \frac{(y_k - \sum_{j=k+1}^{n} u_{kj} x_j)}{u_{kk}} \quad (k = n-1, n-2,\cdots,1) \end{aligned}\right\} \tag{13}$$

根据三角分解公式与回带公式，求解正则方程组，得到未知参数。

5) 误差计算

利用误差公式：

$$\text{均方误差 } \delta_1 = \sqrt{\sum_{i=1}^{N} [\varphi(x_i) - y_i]^2} \tag{14}$$

$$\text{最大偏差 } \delta_2 = \max_{1 \le i \le N} |\varphi(x_i) - y_i| \tag{15}$$

分别通过累加与“擂台法”比较，求解两种误差数值。

6) 结果输出

利用 C 语言设计用户友好的输出界面。输出内容包括双曲线及指数函数曲线的拟合结果，各拟合方式的均方误差与最大误差与原始数据点拟合结果。

7) 图像绘制

将 C 程序计算得到的数据导入 MATLAB R2017a 数学计算软件，绘制包含原始数据的坐标点与拟合曲线的可视化图像，并进行部分文字注释。

1.4.2 程序流程图

问题一程序实现流程图如图 1 所示。

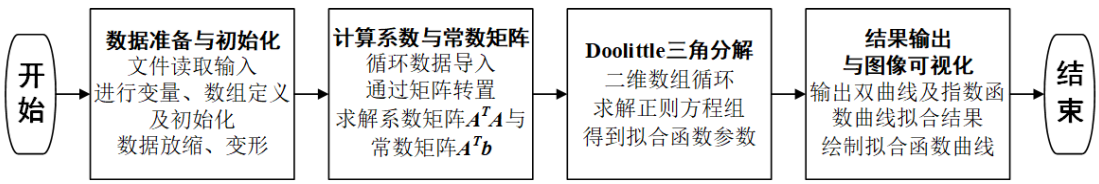

图 1　问题一程序实现流程图

1.4.3 C 语言程序实现

1.4.3.1 问题一：C 程序源代码

问题一中双曲线与指数函数拟合的程序源代码分别保存在“Problem_1_1.c”与“Problem_1_2.c”中。以下为利用双曲线函数拟合数据点的 C 程序源代码。

```
// Problem_1_1.c
// 问题 1_1：求浓度 y 与时间 t 的拟合曲线（双曲线函数拟合）
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
    // MAX 代表数组下标最大值，n 代表实际坐标个数
#define MAX 21
#define n 16
int main()
{
    int t[MAX],i,j,nn,k,m;
    double sum1,sum2,sum3,sum4,temp,ma,sumt;
    // 题设数据导入（文件输入输出）
    double y0[MAX];
    double xx[MAX],yy[MAX],det[MAX],res[MAX];
    double a[5][5],u[5][5],l[5][5],x[5],y[5],b[5];
    FILE *fp;
    fp=fopen("Problem_1_data.txt","r");
    for (i=0;i<=n;i++) fscanf(fp,"%d",&t[i]);
    for (i=0;i<=n;i++) fscanf(fp,"%lf",&y0[i]);
   // 条件初始化
    printf("问题 1_1：求浓度 y 与时间 t 的拟合曲线（双曲线函数拟合）\n");
    printf("\n");
    for (i=1;i<=n;i++) y0[i]*=0.001;
    for (i=0;i<5;i++)
    {
        x[i]=0;y[i]=0;b[i]=0;
        for (j=0;j<5;j++)
        {
            a[i][j]=0;
            u[i][j]=0;
            l[i][j]=0;
        }
    }
    for (i=0;i<MAX;i++)
    {
        xx[i]=0;
```

```
        yy[i]=0;
        det[i]=0;
    }
    // 双曲线拟合数据预处理
    for (i=1;i<=n;i++) t[i]=i;
    for (i=1;i<=n;i++) xx[i]=1.0/t[i];
    for (i=1;i<=n;i++) yy[i]=1.0/(y0[i]);
    a[1][1]=n;
    for (i=1;i<=n;i++)
    {
        a[1][2]+=xx[i];
        a[2][2]+=(xx[i]*xx[i]);
    }
    a[2][1]=a[1][2];
    for (i=1;i<=n;i++)
    {
        b[1]+=yy[i];
        b[2]+=(yy[i]*xx[i]);
    }
    // 采用 Doolittle 三角分解法求解矩阵方程
    // 三角分解
    nn=2;
    for (j=1;j<=nn;j++) u[1][j]=a[1][j];
    for (i=2;i<=nn;i++) l[i][1]=(1.0*a[i][1])/(u[1][1]);
    for (k=2;k<=nn;k++)
    {
        for (j=k;j<=nn;j++)
        {
            sum1=0;
            for (m=1;m<=k-1;m++) sum1+=(l[k][m]*u[m][j]);
            u[k][j]=a[k][j]-sum1;
        }
        for (i=k+1;i<=nn;i++)
        {
            sum2=0;
            for (m=1;m<=k-1;m++) sum2+=(l[i][m]*u[m][k]);
            l[i][k]=a[i][k]-sum2;
        }
    }
    for (k=1;k<=nn;k++) l[k][k]=1;
    // 由回带公式求解未知参数
    y[1]=(1.0*b[1])/l[1][1];
    for (k=2;k<=nn;k++)
    {
```

```
            sum3=0;
            for (j=1;j<=k-1;j++) sum3+=(l[k][j]*y[j]);
            y[k]=(1.0*(b[k]-sum3))/l[k][k];
        }
        x[nn]=(1.0*y[nn])/u[nn][nn];
        for (k=nn-1;k>=1;k--)
        {
            sum4=0;
            for (j=k+1;j<=nn;j++) sum4+=(u[k][j]*x[j]);
            x[k]=(1.0*(y[k]-sum4))/u[k][k];
        }
        // 误差计算
        ma=0;sumt=0;
        for (i=1;i<=n;i++)
        {
            temp=(1.0*t[i])/(x[1]*t[i]+x[2]);
            res[i]=temp;
            det[i]=fabs(y0[i]-temp);
            sumt+=(det[i]*det[i]);
            if (det[i]>ma) ma=det[i];
        }
        // 结果输出
        printf("情形一：\n");
        printf("\n");
        printf("双曲线函数拟合曲线方程为：y=t/(%.6ft+%.6f);\n",x[1],x[2]);
        printf("\n");
        printf("双曲线函数拟合方式最大偏差为：det=%.8f;\n",ma);
        printf("\n");
        printf("双曲线函数拟合方式均方误差为：det_mean=%.8f;\n",sqrt(sumt));
        printf("\n");
        printf("双曲线函数拟合结果如下：\n");
        printf("\n");
        printf("时间 t：  ");
        for (i=1;i<=n;i++) printf("%-5d ",t[i]);
        printf("\n");
        printf("浓度 y：");
        for (i=1;i<=n;i++) printf("%-5.2f ",res[i]*1000);
        printf("\n");
        return 0;
}
```

1.4.3.2 问题一：C 语言输出界面

问题一中编写的 C 程序输出界面如图 2，3 所示。

```
问题1_1：求浓度y与时间t的拟合曲线（双曲线函数拟合）

情形一：

双曲线函数拟合曲线方程为：y=t/(80.174460t+162.722545)；

双曲线函数拟合方式最大偏差为：det=0.00056037；

双曲线函数拟合方式均方误差为：det_mean=0.00124984；

双曲线函数拟合结果如下：

时间t：  1     2     3     4     5     6     7     8     9     10    11    12    13    14    15    16
浓度y：4.12  6.19  7.44  8.27  8.87  9.32  9.67  9.95  10.18 10.37 10.53 10.67 10.79 10.89 10.99 11.07

Process returned 0 (0x0)   execution time : 0.085 s
Press any key to continue.
```

图 2 问题一：C 程序输出界面（双曲线函数拟合）

```
问题1_2：求浓度y与时间t的拟合曲线（指数函数拟合）

情形二：

指数函数拟合曲线方程为：y=0.011325*exp(-1.056684/t)；

指数函数拟合方式最大偏差为：det=0.00027715；

指数函数拟合方式均方误差为：det_mean=0.00034101；

指数函数拟合结果如下：

时间t：  1     2     3     4     5     6     7     8     9     10    11    12    13    14    15    16
浓度y：3.94  6.68  7.96  8.70  9.17  9.50  9.74  9.92  10.07 10.19 10.29 10.37 10.44 10.50 10.55 10.60

Process returned 0 (0x0)   execution time : 0.287 s
Press any key to continue.
```

图 3 问题一：C 程序输出界面（指数函数拟合）

1.4.3.3 问题一：MATLAB 拟合函数曲线绘制

利用 MATLAB 软件绘制函数曲线图像，分别如图 4，5 所示。**（MATLAB 源程序见附录）**可以清晰看出，指数函数拟合效果明显好于双曲线函数拟合。

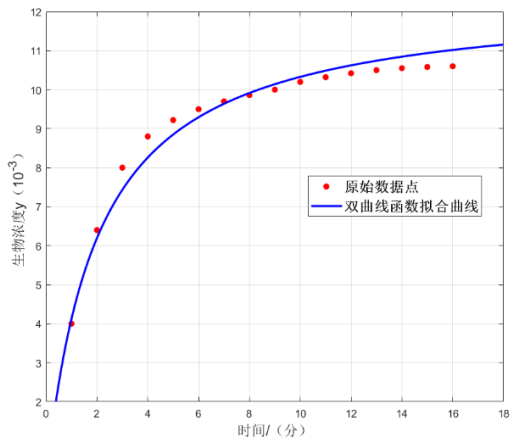

图 4 双曲线函数拟合图像

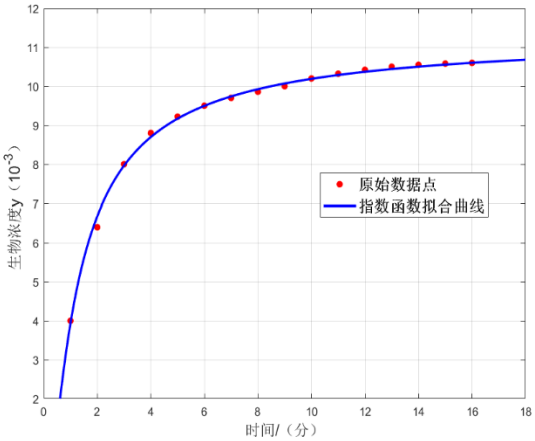

图 5 指数函数拟合图像

1.5 程序使用说明书

问题一为利用最小二乘法拟合函数曲线问题。求解问题的 C 语言程序已保存在“Problem_1_1.c”与“Problem_1_2.c”中。使用说明如下：

- 使用 Code:Blocks、Dev-C++等 C 程序编译软件打开.c 文件；
- 在项目路径目录下找到“Problem_1_data.txt”，打开，进行 *t*,*y* 变量修改；
- 点击编译运行程序，输出程序结果，并在屏幕上显示；
- 记录拟合曲线数据结果，绘制函数图像，完成问题求解。

2 实习问题二：利用 Seidel 迭代法求解矩阵方程组

2.1 问题叙述

用 Seidel 迭代法求解下列方程组。取初始向量 $x^{(0)}=(0,0,0)^T$，要求满足条件 $\max\limits_{1\le i\le 3}\left|x_i^{(k+1)}-x_i^{(k)}\right|\le 10^{-3}$ 时迭代终止。

$$\begin{bmatrix}10 & -2 & -1\\ -2 & 10 & -1\\ -1 & -2 & 5\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}3\\ 15\\ 10\end{bmatrix} \tag{16}$$

2.2 问题分析

问题二要求利用 Seidel 迭代法求解三阶非齐次线性方程组。求解此类方程组的方式多样，主要的迭代法有**简单迭代法（Jacobi 迭代法）、Seidel 迭代法、逐次超松弛迭代法（SOR 方法）**等。Seidel 迭代法是建立在简单迭代公式基础上的改进方法，其求解收敛速度更快，计算效率更高。本文首先对该问题的 Seidel 迭代法的理论求解方式进行讨论，并设计可适用于不同阶数的非齐次线性矩阵方程求解的通用 C 语言程序的编写工作，拓展程序功能。

2.3 理论求解

在简单迭代公式的基础上可提出如下迭代公式：

Seidel 迭代公式：

$$\left.\begin{aligned} x_1^{(k+1)} &= b_{11}x_1^{(k)} + b_{12}x_2^{(k)} + \cdots + b_{1n}x_n^{(k)} + g_1 \\ x_2^{(k+1)} &= b_{21}x_1^{(k)} + b_{22}x_2^{(k)} + \cdots + b_{2n}x_n^{(k)} + g_2 \\ x_3^{(k+1)} &= b_{31}x_1^{(k)} + b_{32}x_2^{(k)} + \cdots + b_{3n}x_n^{(k)} + g_3 \\ &\vdots \\ x_n^{(k+1)} &= b_{n1}x_1^{(k)} + b_{n2}x_2^{(k)} + \cdots + b_{nn}x_n^{(k)} + g_n \end{aligned}\right\} \tag{17}$$

令

$$\boldsymbol{B_1} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ b_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & 0 \end{bmatrix}, \quad \boldsymbol{B}_2 = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ 0 & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & b_{nn} \end{bmatrix} \tag{18}$$

其矩阵形式为

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{B_1 x}^{(k+1)} + \boldsymbol{B_2 x}^{(k)} + \boldsymbol{g} \tag{19}$$

这里不再详细介绍 Seidel 迭代公式的收敛判断定理。

对于问题二，设系数矩阵 $\boldsymbol{A}$，解集向量 $\boldsymbol{x}$，常数矩阵 $\boldsymbol{b}$ 分别如下，

$$\boldsymbol{A} = \begin{bmatrix} 10 & -2 & -1 \\ -2 & 10 & -1 \\ -1 & -2 & 5 \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 3 \\ 15 \\ 10 \end{bmatrix}, \tag{20}$$

因为方程组的系数矩阵 $\boldsymbol{A}$ 是**严格对角占优矩阵**，所以，可判断得与 Jacobi 迭代法对应的 Seidel 迭代法收敛。Seidel 迭代公式为

$$\begin{cases} x_1^{(k+1)} = & 0.2x_2^{(k)} + 0.1x_3^{(k)} + 0.3 \\ x_2^{(k+1)} = 0.2x_1^{(k)} + & 0.1x_3^{(k)} + 1.5 \\ x_3^{(k+1)} = 0.2x_1^{(k)} + 0.4x_2^{(k)} & + 2 \end{cases}, \ (k = 0,1,2\cdots) \tag{21}$$

计算结果如表 2 所示，

表 2 Seidel 迭代法计算结果

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$
0	0	0	0
1	0.300 00	1.560 00	2.684 00
2	0.880 40	1.944 48	2.953 87
3	0.984 28	1.992 24	2.993 75
4	0.997 82	1.998 94	2.999 14
5	0.999 70	1.999 85	2.999 88
6	0.999 96	1.999 98	2.999 98

因为$\left|x_i^{(6)}-x_i^{(5)}\right|\le 10^{-3}\ (i=1,2,3)$，所以$\boldsymbol{x}^{(6)}$可作为方程组的近似解向量。

2.4 程序设计

2.4.1 C 语言程序算法设计

本问题编程实现利用 Seidel 迭代法求解非齐次线性矩阵的关键点如下：

- **设计通用输入程序模块，包括系数矩阵、常数矩阵等信息；**
- **灵活利用数组保存计算得到的各矩阵数据；**
- **使用 While 循环判断迭代过程是否应该退出；**
- **友好的界面设计，拥有清晰的输出格式；**

具体算法设计如下：

1) 数据预处理

将题设所给各矩阵化为数组形式，进行变量、数组定义及初始化，为避免计算混淆，数组下标与原矩阵下标保持一致。采用交互式通用输入方式，提示输入矩阵的行数、列数与系数和常数矩阵。程序中浮点数及数组均使用 Double 双精度类型。

2) 计算迭代公式

通过循环方式，根据导入矩阵数据，计算 Seidel 迭代法的系数矩阵。即分别求解迭代公式$\boldsymbol{x}^{(k+1)}=\boldsymbol{B}_1\boldsymbol{x}^{(k+1)}+\boldsymbol{B}_2\boldsymbol{x}^{(k)}+\boldsymbol{g}$中的各系数矩阵，保存在 **c** 数组中。

3) 利用循环实现 Seidel 迭代过程，用 While 循环判断迭代是否结束；

由题意，本程序设置迭代过程中的相邻解集的元素绝对值之差的最大值为$det_max=10^{-3}$,利用迭代递推公式，结合数组下标循环，计算各组解向量。取初始向量$\boldsymbol{x}^{(0)}=(0,0,0)^{\mathrm{T}}$，并设置 0-1 数值标记*flag*，每次迭代计算完成一组解向量，进行一次相邻解集的元素绝对值之差与结束条件的判断。当 *flag*=1 时，迭代继续；当*flag*=0 时，迭代结束。本程序利用 While 不固定次数循环进行迭代控制，确保程序不发生循环溢出或者死循环现象。

4) 结果输出

利用 C 语言设计用户友好的输出界面。输出内容包括当程序停止迭代递推时，Seidel 迭代方式的迭代次数以及最终求得的解向量。

2.4.2 程序流程图

问题二程序实现流程图如图 6 所示。

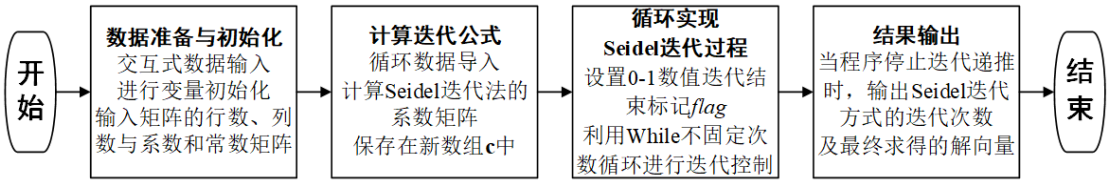

图 6 问题二程序实现流程图

2.4.3 C 语言程序实现

2.4.3.1 问题二：C 程序源代码

问题二中利用 Seidel 迭代法求解非齐次线性矩阵的程序源代码保存在“Problem_2.c”中。C 程序源代码如下。

```
// Problem_2.c
// 问题 2：利用 Seidel 迭代法求解方程组
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX1 8
#define MAX2 100

int main()
{
    int n,m,i,j,k,t,flag;
    double a[MAX1][MAX1],c[MAX1][MAX1],b[MAX1],x[MAX2][MAX1+1];
    double temp,det,det_max;
    // 数据初始化
    for (i=0;i<MAX1;i++)
        for (j=0;j<MAX1;j++)
    {
        a[i][j]=0;
        c[i][j]=0;
    }
    for (i=0;i<MAX1;i++) b[i]=0;
    for (i=0;i<MAX2;i++)
        for (j=0;j<MAX1;j++) x[i][j]=0;
    // 通用数据导入
    printf("问题 2：利用 Seidel 迭代法求解方程组\n");
    printf("\n");
```

```
printf("请输入矩阵的行数：\n");
scanf("%d",&n);
printf("请输入矩阵的列数：\n");
scanf("%d",&m);
printf("请输入%d*%d 的系数矩阵：\n",n,m);
for (i=1;i<=n;i++)
    for (j=1;j<=m;j++) scanf("%lf",&a[i][j]);
printf("\n");
printf("请输入 1*%d 的常数矩阵：\n",n);
for (i=1;i<=n;i++) scanf("%lf",&b[i]);
// 迭代矩阵计算
for (i=1;i<=n;i++)
{
    temp=a[i][i];
    c[i][m+1]=(1.0*b[i])/temp;
    for (j=1;j<=m;j++)
    {
        if (j==i) c[i][j]=0;
        else c[i][j]=(-1.0*a[i][j])/temp;
    }
}
// 设置初始向量
x[0][1]=0;x[0][2]=0;x[0][3]=0;
// 迭代公式计算
flag=1; // 数值标记初始化
det_max=0.001;
k=0; //计数器启动
// 当 det_max<=1e-3 时，结束迭代
while (flag)
{
    //一次完整迭代
    for (t=1;t<=n;t++)
    {
        for (i=1;i<=t-1;i++) x[k+1][t]+=(c[t][i]*x[k+1][i]);
        for (i=t;i<=m;i++) x[k+1][t]+=(c[t][i]*x[k][i]);
        x[k+1][t]+=c[t][m+1];
    }
    //计算误差
    det=-1;
    for (i=1;i<=n;i++)
        if ((x[k+1][i]-x[k][i])>det) det=(x[k+1][i]-x[k][i]);
    if (det<det_max) flag=0;
    k++;
}
```

```
    // 结果输出
    printf("\n");
    printf("求解矩阵方程结果如下：\n");
    printf("\n");
    printf("迭代过程在 k=%d 时结束\n",k);
    for (i=1;i<=n;i++) printf("x%d=%.5f ",i,x[k][i]);
    printf("\n");
    return 0;
}
```

2.4.3.2 问题二：C 语言输出界面

问题二中编写的 C 程序输出界面如图 7 所示。

```
问题2：利用Seidel迭代法求解方程组

请输入矩阵的行数：3
请输入矩阵的列数：3
请输入3*3的系数矩阵：
10 -2 -1
-2 10 -1
-1 -2 5

请输入1*3的常数矩阵：
3 15 10

求解矩阵方程结果如下：

迭代过程在k=6时结束
x1=0.99996 x2=1.99998 x3=2.99998

Process returned 0 (0x0)   execution time : 13.883 s
Press any key to continue.
```

图 7 问题二：C 程序输出界面（Seidel 迭代法求解矩阵方程）

2.5 程序使用说明书

问题二为利用 Seidel 迭代法求解非齐次线性矩阵方程组问题。求解问题的 C 语言程序已保存在“Problem_2.c”中。使用说明如下：

- 使用 Code:Blocks、Dev-C++等 C 程序编译软件打开.c 文件；
- 点击编译运行程序，在屏幕上显示**“请输入矩阵的行数：”**等提示，根据提示输入需要计算得矩阵方程组的系数矩阵及其行数、列数和常数矩阵等数据，每一步完成后按**“Enter”**键继续进行输入步骤；
- 完成所有数据输入，进行程序求解。记录屏幕上显示的 Seidel 迭代法求解矩阵方程的迭代次数与最终解向量等结果，完成问题求解。

3 实习问题三：利用直接三角分解（Doolittle）法求解方程组

3.1 问题叙述

用直接三角分解法求解下面方程组，并写出 Doolittle 分解的单位下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$ 的形式。

$$\begin{cases} 2x_1 + x_2 + 2x_3 = 6 \\ 4x_1 + 5x_2 + 4x_3 = 18 \\ 6x_1 - 3x_2 + 5x_3 = 5 \end{cases} \tag{22}$$

3.2 问题分析

问题三要求利用**直接三角分解法**中的 **Doolittle 三角分解法**求解三阶非齐次线性方程组。直接三角分解法是解线性方程组的直接法的一种常用方法。直接三角分解法直接对方程组的系数矩阵或者推广矩阵进行三角分解变换与回带公式求解，与其他求解线性方程组的直接法，如高斯（Guass）消去法、列主元素消去法、总体选主元素消去法等相比，具有理论公式简练，算法空间利用率高，计算效率高等优势。本文首先结合题目对直接三角分解法中的 Doolittle 三角分解法的理论求解方式进行讨论，并设计可适用于任何非齐次线性矩阵方程求解的通用 C 语言程序，供类似题目求解使用。

3.3 理论求解

由于在本文第一部分，即**利用最小二乘法拟合函数曲线的问题**理论求解过程中已经对 Doolittle 三角分解法进行介绍，详见公式（11）~（13），因此，该部分仅结合本题具体数据对直接三角分解方法进行更加详细的阐述。

对于问题三，设系数矩阵 $\boldsymbol{A}$，解集向量 $\boldsymbol{x}$，常数矩阵 $\boldsymbol{b}$ 分别如下，

$$\boldsymbol{A}=\begin{bmatrix} 2 & 1 & 2 \\ 4 & 5 & 4 \\ 6 & -3 & 5 \end{bmatrix},\quad \boldsymbol{x}=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 6 \\ 18 \\ 5 \end{bmatrix}, \tag{23}$$

设方程组推广矩阵为

$$\boldsymbol{B}=\boldsymbol{(A,b)}=\left[\begin{array}{ccc:c} 2 & 1 & 2 & 6 \\ 4 & 5 & 4 & 18 \\ 6 & -3 & 5 & 5 \end{array}\right] \tag{24}$$

由 Doolittle 分解方法可直接得到推广矩阵 $\boldsymbol{B}$ 的三角分解形式：

$$\boldsymbol{B}=\left[\begin{array}{ccc:c} 2 & 1 & 2 & 6 \\ 4 & 5 & 4 & 18 \\ 6 & -3 & 5 & 5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc:c} 2 & 1 & 2 & 6 \\ 2 & 3 & 0 & 6 \\ 3 & -2 & -1 & -1 \end{array}\right] \tag{25}$$

经过三角分解后的推广矩阵中，折线右上方（竖虚线以左）的元素即为上三角阵 $\boldsymbol{U}$ 的元素，竖虚线右端的向量由回带公式（12）（13）所得的向量 $\boldsymbol{y}$，于是原方程组的解为

$$\begin{aligned} &x_3=\frac{-1}{-1}=1\,,\ x_2=(6-0\times x_3)/3=2, \\ &x_1=(6-2x_3-x_2)/2=1 \end{aligned} \tag{26}$$

系数矩阵 $\boldsymbol{A}$ 的 Doolittle 三角分解结果如下，

$$\boldsymbol{A}=\left[\begin{array}{ccc} 2 & 1 & 2 \\ 4 & 5 & 4 \\ 6 & -3 & 5 \end{array}\right]=\boldsymbol{LU} \tag{27}$$

其中

$$\boldsymbol{L}=\left[\begin{array}{ccc} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -2 & 1 \end{array}\right],\ \boldsymbol{U}=\left[\begin{array}{ccc} 2 & 1 & 2 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{array}\right]. \tag{28}$$

3.4 程序设计

3.4.1 C 语言程序算法设计

本问题编程实现利用直接三角分解法中的Doolittle三角分解法求解三阶非齐次线性方程组，关键点如下：

- **设计通用输入程序模块，包括系数矩阵、常数矩阵等信息；**
- **灵活利用数组保存计算得到的各矩阵数据；**
- **将数学矩阵符号用数组表示，并根据 Doolittle 三角分解理论推导公式以及回带公式，列出边界控制条件，对待求的分解矩阵中各行、列元素值进行循环累加计算；**
- **友好的界面设计，拥有清晰的输出格式，在屏幕上显示方程组解集以及三角分解得到的单位下三角矩阵 $\boldsymbol{L}$ 与上三角矩阵 $\boldsymbol{U}$；**

具体算法设计如下：

1) 数据预处理

问题三的数据导入与预处理模块与问题二类似。将题设所给各矩阵化为数组形式，进行变量、数组定义及初始化，为避免计算混淆，数组下标与原矩阵下标保持一致。采用交互式通用输入方式，提示输入矩阵的行数、列数与系数和常数矩阵。程序中浮点数及数组均使用 Double 双精度类型。

2) 根据 Doolittle 三角分解公式（11）计算 *L* 与上三角 *U* 矩阵

通过循环方式，根据导入矩阵数据，计算单位下三角矩阵 ***L*** 与上三角矩阵 ***U***。***U*** 的元素按行求，***L*** 的元素按列求；先求 ***U*** 的第 *k* 行元素，然后求 ***L*** 的第 *k* 列元素，***U*** 与 ***L*** 一行一行交叉计算，通过边界条件初始化与数组循环方式，进行如图 8 所示的逐框计算，并将 ***L*** 与上三角 ***U*** 矩阵的计算结果保存在相应数组中。

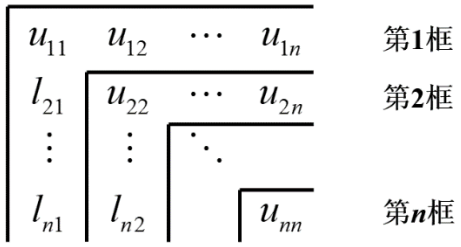

图 8 利用 Doolittle 分解法逐框计算 ***L*** 与 ***U*** 矩阵示意图

3) 根据回带公式（12）（13）求矩阵方程组的解集

完成对系数矩阵 ***A*** 的三角分解后，由以下等价关系

$$\boldsymbol{Ax = b} \Leftrightarrow \boldsymbol{(LU)x = b} \Leftrightarrow \begin{cases} \boldsymbol{Ly = b} \\ \boldsymbol{Ux = y} \end{cases}$$

结合题设给出的常数矩阵 ***b***，通过数组循环实现回带过程，分别求解 ***y*** 与 ***x*** 矩阵，求得原方程组解集，并保存在相应数组中。

4) 结果输出

利用 C 语言设计用户友好的输出界面。输出内容包括最终求得的矩阵方程解向量以及通过 Doolittle 三角分解方法得到满足 ***A=LU*** 的单位下三角矩阵 ***L*** 与上三角矩阵 ***U***，采用书写形式打印。

3.4.2 程序流程图

问题三程序实现流程图如图 9 所示。

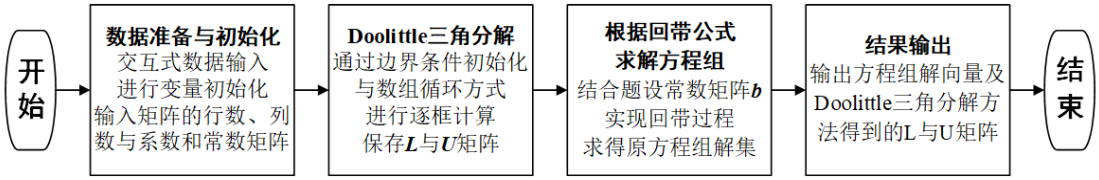

图 9　问题三程序实现流程图

3.4.3 C 语言程序实现

3.4.3.1　问题三：C 程序源代码

问题三利用直接三角分解法（Doolittle 三角分解法）求解三阶非齐次线性方程组，程序源代码保存在“Problem_3.c”中。C 程序源代码如下。

```
// Problem_3.c
/*问题 3：用直接三角分解法求解方程组，并写出
        Doolittle 分解的单位下三角矩阵 L 和上三角矩阵 U 的形式*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX 8

int main()
{
    int n,m,nn,i,j,k;
    double sum1,sum2,sum3,sum4;
    double a[MAX][MAX],u[MAX][MAX],l[MAX][MAX],x[MAX],y[MAX],b[MAX];
    // 数据初始化
    for (i=0;i<MAX;i++)
        for (j=0;j<MAX;j++)
        {
            a[i][j]=0;
            u[i][j]=0;
            l[i][j]=0;
        }
    for (i=0;i<MAX;i++)
    {
        b[i]=0;
        x[i]=0;
        y[i]=0;
    }
    //通用数据导入
    printf("问题 3：用直接三角分解法求解方程组，\n");
```

```
printf("并写出 Doolittle 分解的单位下三角矩阵 L 和上三角矩阵 U 的形式\n");
printf("\n");
printf("请输入矩阵的行数：");
scanf("%d",&n);
printf("请输入矩阵的列数：");
scanf("%d",&m);
printf("请输入%d*%d 的系数矩阵：\n",n,m);
for (i=1;i<=n;i++)
    for (j=1;j<=m;j++) scanf("%lf",&a[i][j]);
printf("请输入 1*%d 的常数矩阵：\n",n);
for (i=1;i<=n;i++) scanf("%lf",&b[i]);
// 采用 Doolittle 三角分解法求解矩阵方程
// 三角分解
nn=n;
for (j=1;j<=nn;j++) u[1][j]=a[1][j];
for (i=2;i<=nn;i++) l[i][1]=(1.0*a[i][1])/(u[1][1]);
for (k=2;k<=nn;k++)
{
    for (j=k;j<=nn;j++)
    {
        sum1=0; //需注意累加变量初始化位置
        for (m=1;m<=k-1;m++) sum1+=(l[k][m]*u[m][j]);
        u[k][j]=a[k][j]-sum1;
    }
    for (i=k+1;i<=nn;i++)
    {
        sum2=0;
        for (m=1;m<=k-1;m++) sum2+=(l[i][m]*u[m][k]);
        l[i][k]=(1.0*(a[i][k]-sum2))/u[k][k];
    }
}
for (k=1;k<=nn;k++) l[k][k]=1;
// 由回带公式求解未知参数
y[1]=(1.0*b[1])/l[1][1];
for (k=2;k<=nn;k++)
{
    sum3=0;
    for (j=1;j<=k-1;j++) sum3+=(l[k][j]*y[j]);
    y[k]=(1.0*(b[k]-sum3))/l[k][k];
}
x[nn]=(1.0*y[nn])/u[nn][nn];
for (k=nn-1;k>=1;k--)
{
    sum4=0;
```

```
            for (j=k+1;j<=nn;j++) sum4+=(u[k][j]*x[j]);
            x[k]=(1.0*(y[k]-sum4))/u[k][k];
        }
        // 结果输出
        printf("\n");
        printf("方程组的解为：\n");
        for (i=1;i<=n;i++) printf("x%d=%.2f ",i,x[i]);
        printf("\n");
        printf("\n");
        printf("Doolittle 三角分解结果如下：\n");
        printf("\n");
        printf("单位下三角矩阵 L 为：\n");
        for (i=1;i<=n;i++)
        {
            for (j=1;j<=m;j++) printf("%6.2f",l[i][j]);
            printf("\n");
        }
        printf("\n");
        printf("上三角矩阵 U 为：\n");
        for (i=1;i<=n;i++)
        {
            for (j=1;j<=m;j++) printf("%6.2f",u[i][j]);
            printf("\n");
        }
        return 0;
}
```

3.4.3.2 问题三：C 语言输出界面

问题三中编写的 C 程序输出界面如图 10 所示。

```
问题3：用直接三角分解法求解方程组，
并写出Doolittle分解的单位下三角矩阵L和上三角矩阵U的形式

请输入矩阵的行数：3
请输入矩阵的列数：3
请输入3*3的系数矩阵：
2 1 2
4 5 4
6 -3 5
请输入1*3的常数矩阵：
6 18 5

方程组的解为：
x1=1.00 x2=2.00 x3=1.00

Doolittle三角分解结果如下：

单位下三角矩阵L为：
  1.00   0.00   0.00
  2.00   1.00   0.00
  3.00  -2.00   1.00

上三角矩阵U为：
  2.00   1.00   2.00
  0.00   3.00   0.00
  0.00   0.00  -1.00

Process returned 0 (0x0)   execution time : 13.480 s
Press any key to continue.
```

图 10 问题三：C 程序输出界面（利用 Doolittle 三角分解法求解三阶非齐次线性方程组）

3.5 程序使用说明书

问题三为利用直接三角分解法（Doolittle 三角分解法）求解三阶非齐次线性方程组问题。求解问题的 C 语言程序已保存在“Problem_3.c”中。使用说明如下：

- 使用 Code:Blocks、Dev-C++等 C 程序编译软件打开.c 文件；
- 点击编译运行程序，在屏幕上显示**“请输入矩阵的行数：”**等提示，根据提示输入需要计算得矩阵方程组的系数矩阵及其行数、列数和常数矩阵等数据，每一步完成后按**“Enter”**键继续进行输入步骤；
- 完成所有数据输入，进行程序求解。记录屏幕上显示的利用直接三角分解法（Doolittle 三角分解法）求解矩阵方程的最终解向量以及单位下三角矩阵 ***L*** 与上三角矩阵 ***U*** 等结果，完成问题求解。

4 实习问题四：使用古典显示格式求解边值问题的数值解

4.1 问题叙述

使用古典显示格式求解下述边值问题的数值解，其中 $h=0.2,\lambda=\frac{1}{6}$，计算每个单元节点的值。（请保留六位小数）

$$\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}\quad(0<x<1,0<t) \tag{29}$$

$$\begin{cases} u(0,t)=u(1,t)=0\quad(0\le t) \\ u(x,0)=4x(1-x)\quad(0\le x\le 1) \end{cases} \tag{30}$$

4.2 问题分析

问题四要求使用古典显示格式求解边值问题的数值解。在偏微分方程混合边值问题求解中，古典显式差分格式是一种常用的差分格式，适用于已知边值条件与偏微分方程形式的情况，可以较好地应用于如一维热传导方程的混合边值等问题的求解过程中。本文首先结合题目给出的具体混合边值问题，对使用古典显示格式的理论求解方式进行讨论，并设计可适用于不同矩形网格划分条件的古典显示差分求数值解的通用 C 语言程序，供类似网格差分题目求解使用。

4.3 理论求解

由问题四题设条件，可得 $m=\frac{1}{h}=5,\ \lambda=\frac{\tau}{h^2}=\frac{1}{6}$，不妨首先假设 n=36, 因为 $u(0,0)=u(1,0)=0$, 从定解条件，解函数 $u(x,t)$ 关于直线 $x=\frac{1}{2}$ 对称，如图 11 所示，

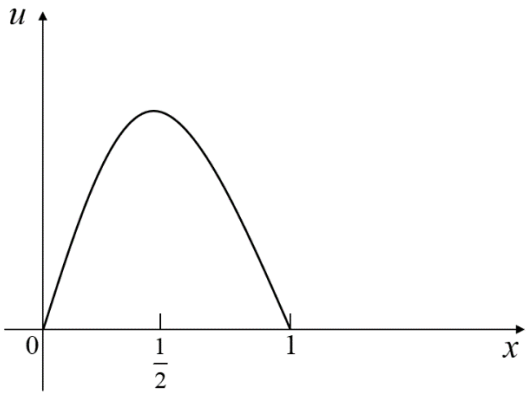

图 11 解函数$u(x,t)$的对称性示意图

故必有

$$u_{1,j}=u_{4,j}\ ,\ u_{2,j}=u_{3,j} \tag{31}$$

因此，只要计算$u_{i,j}$ $(i=1,2;\ j=0,1,\cdots,36)$。差分格式为

$$u_{i,j+1}=\frac{1}{6}u_{i+1,j}+\frac{2}{3}u_{i,j}+\frac{1}{6}u_{i-1,j}\quad (i=1,2;\ j=0,1,2,\cdots,36) \tag{32}$$

$$\begin{cases}u_{i,0}=0.8i(1-0.2i)\quad (i=1,2)\\ u_{0,j}=0\qquad (j=0,1,2,\cdots,36)\end{cases} \tag{33}$$

1) 计算第 0 层上的值：$u_{0,0}=0$

$u_{1,0}=0.8(1-0.2)=0.64$

$u_{2,0}=1.6(1-0.4)=0.96$

2) 计算第 1 层上的值：$u_{0,1}=0$

$$u_{1,1}=\frac{1}{6}u_{2,0}+\frac{2}{3}u_{1,0}+\frac{1}{6}u_{0,0}=0.586667$$

$$u_{2,1}=\frac{1}{6}u_{3,0}+\frac{2}{3}u_{2,0}+\frac{1}{6}u_{1,0}=0.906667$$

3) 完整计算结果如表 3 所示。

表 3 使用古典显示格式求解边值问题的数值解

i / *j*	0	1	2	3	4	5
0	0	0.64	0.96	0.96	0.64	0

1	0	0.586667	0.906667	0.906667	0.586667	0
2	0	0.542222	0.853333	0.853333	0.542222	0
3	0	0.503704	0.801481	0.801481	0.503704	0
…	…	…	…	…	…	…
34	0	0.064767	0.104795	0.104795	0.064767	0
35	0	0.060644	0.098124	0.098124	0.060644	0
36	0	0.056783	0.091877	0.091877	0.056783	0

4.4 程序设计

4.4.1 C 语言程序算法设计

本问题利用 C 语言编程实现使用古典显示格式求解混合边值问题的数值解，该程序编写的关键点如下：

- **设计通用输入程序模块，包括 x 方向步长、网比 λ 、求解层数 n 等信息；**
- **差值边界条件的初始化；**
- **递推求解 0~n 层上每个单元节点的值；**
- **友好的界面设计，清晰的输出格式，屏幕上显示各层节点单元数值；**

具体算法设计如下：

1) 数据预处理

进行变量、数组定义及初始化。采用交互式通用输入方式，提示输入 x 方向步长、网比 λ 、求解层数 n。程序中浮点数及数组均使用 Double 双精度类型。

2) 边界条件初始化

由输入信息计算使用古典显式差分格式求解的各参数。进行边界条件初始化，首先由边值函数计算第 0 层上各单元的数值解。

3) 利用递推公式逐层计算各单元数值解

程序采用 u 数组来保存各层单元的数值，其中 u 数组的行号 i 表示层数，列号 j 表示差分网格纵向第 j 层。利用差分格式给出的递推公式，利用数组循环与下标变换进行递推，直到计算至第 n 层节点单元终止。

4) 结果输出

利用 C 语言设计用户友好的输出界面。输出利用古典显式差分格式求解得到的 0~n 层中各单元的数值解，采用书写形式打印，共计 $n\times(\left[\frac{1}{h}\right]+1)$ 个数据，保留

6 位小数。

4.4.2 程序流程图

问题四程序实现流程图如图 12 所示。

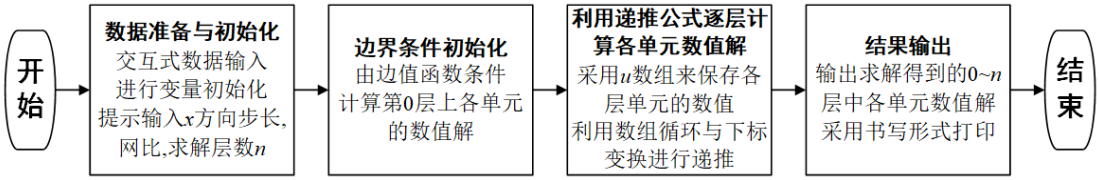

图 12 问题四程序实现流程图

4.4.3 C 语言程序实现

4.4.3.1 问题四：C 程序源代码

问题四实现使用古典显示格式求解混合边值问题的数值解，程序源代码保存在“Problem_4.c”中。C 程序源代码如下。

```
// Problem_4.c
// 问题 4：使用古典显示格式求解边值问题的数值解
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

    // 节点最大层数
#define MAX 100

int main()
{
    int n,m,i,j;
    double h,l,u[MAX][MAX];
    // 数值导入与初始化
    for (i=0;i<MAX;i++)
        for (j=0;j<MAX;j++) u[i][j]=0;
    printf("问题 4：使用古典显示格式求解边值问题的数值解\n");
    printf("\n");
    printf("请输入 x 方向的步长 h：");
    scanf("%lf",&h);
    m=(int)(1.0/h);
    printf("请输入网比 Lambda：");
    scanf("%lf",&l);
    printf("请输入求解层数 n：");
    scanf("%d",&n);
```

```
    // 边界条件初始化 1
    // 计算第 0 层上的值
    for (i=0;i<=m;i++) u[i][0]=(4*h*i)*(1-(h*i));
    // 边界条件初始化 2
    for (j=0;j<=n;j++) u[0][j]=0;
    // 递推求解 0~n 层上各单元节点的值
    for (j=0;j<=n;j++)
    {
        for (i=1;i<=m;i++)
            u[i][j+1]=(l*u[i+1][j])+((1-2*l)*u[i][j])+(l*u[i-1][j]);
    }
    // 计算结果输出
    printf("\n");
    printf("0~%d 层各层单元节点的值列表如下：\n",n);
    printf("\n");
    for (j=0;j<=n;j++)
    {
        for (i=0;i<=m+1;i++) printf("%-8.6f ",u[i][j]);
        printf("\n");
    }
    return 0;
}
```

4.4.3.2 问题四：C 语言输出界面

问题四中编写的 C 程序输出界面如图 13 所示。

```
问题4：使用古典显示格式求解边值问题的数值解

请输入x方向的步长h：0.2
请输入网比Lambda：0.166666667
请输入求解层数n：36

0~36层各层单元节点的值列表如下：

0.000000 0.640000 0.960000 0.960000 0.640000 0.000000
0.000000 0.586667 0.906667 0.906667 0.586667 0.000000
0.000000 0.542222 0.853333 0.853333 0.542222 0.000000
0.000000 0.503704 0.801481 0.801481 0.503704 0.000000
0.000000 0.469383 0.751852 0.751852 0.469383 0.000000
0.000000 0.438230 0.704774 0.704774 0.438230 0.000000
0.000000 0.409616 0.660350 0.660350 0.409616 0.000000
0.000000 0.383136 0.618561 0.618561 0.383136 0.000000
0.000000 0.358517 0.579323 0.579323 0.358517 0.000000
0.000000 0.335565 0.542522 0.542522 0.335565 0.000000
0.000000 0.314131 0.508029 0.508029 0.314131 0.000000
0.000000 0.294092 0.475713 0.475713 0.294092 0.000000
0.000000 0.275347 0.445443 0.445443 0.275347 0.000000
0.000000 0.257805 0.417093 0.417093 0.257805 0.000000
0.000000 0.241386 0.390545 0.390545 0.241386 0.000000
0.000000 0.226015 0.365685 0.365685 0.226015 0.000000
0.000000 0.211624 0.342407 0.342407 0.211624 0.000000
0.000000 0.198150 0.320610 0.320610 0.198150 0.000000
0.000000 0.185535 0.300200 0.300200 0.185535 0.000000
0.000000 0.173724 0.281089 0.281089 0.173724 0.000000
0.000000 0.162664 0.263195 0.263195 0.162664 0.000000
0.000000 0.152308 0.246440 0.246440 0.152308 0.000000
0.000000 0.142612 0.230751 0.230751 0.142612 0.000000
0.000000 0.133533 0.216061 0.216061 0.133533 0.000000
0.000000 0.125032 0.202307 0.202307 0.125032 0.000000
0.000000 0.117073 0.189428 0.189428 0.117073 0.000000
0.000000 0.109620 0.177368 0.177368 0.109620 0.000000
0.000000 0.102641 0.166077 0.166077 0.102641 0.000000
0.000000 0.096107 0.155504 0.155504 0.096107 0.000000
0.000000 0.089989 0.145605 0.145605 0.089989 0.000000
0.000000 0.084260 0.136335 0.136335 0.084260 0.000000
0.000000 0.078896 0.127656 0.127656 0.078896 0.000000
0.000000 0.073873 0.119530 0.119530 0.073873 0.000000
0.000000 0.069170 0.111920 0.111920 0.069170 0.000000
0.000000 0.064767 0.104795 0.104795 0.064767 0.000000
0.000000 0.060644 0.098124 0.098124 0.060644 0.000000
0.000000 0.056783 0.091877 0.091877 0.056783 0.000000

Process returned 0 (0x0)   execution time : 9.568 s
Press any key to continue.
```

图 13 问题四：C 程序输出界面（使用古典显示格式求解混合边值问题的数值解，n=36）

4.5 程序使用说明书

问题四为使用古典显示格式求解混合边值问题的数值解问题。求解问题的 C 语言程序已保存在“Problem_4.c”中。使用说明如下：

- 使用 Code:Blocks、Dev-C++等 C 程序编译软件打开.c 文件；
- 点击编译运行程序，在屏幕上显示**“请输入 x 方向的步长 h：”**等提示，根据提示输入 x 方向步长、网比 λ、求解层数 n 等数据，每一步完成后按**“Enter”**键继续进行输入步骤；
- 完成所有数据输入，进行程序求解。记录屏幕上显示的 0~n 层中各单元的数值解数据，实现问题求解。

5 实习感悟

在航空学院《计算方法》的课程学习中，前人众多精妙而严谨的计算理论进一步提升了我探索算法的热情。虽然自己曾经接触过一些关于数学建模与常用算法的知识，但是过去我的状态基本是“会用但不明其理”。通过本课程的课堂理论学习与自主实习环节，我充分感受到理论与实践相结合迸发出的强大力量。最小二乘法拟合曲线，Seidel 迭代法计算矩阵方程，Doolittle 三角分解法，古典显示格式差分求混合边值问题……这些凝聚着无数智慧的计算方法，有坚实的理论基础，更有广泛的工程应用前景。通过用最基础的 C 语言自我理解理论并编写程序进行问题求解，我认识问题，理性分析问题以及自主解决问题的综合能力都得到了显著的提升。同时，我对算法的理解从停留在“会”层面上升到“懂”层面，这对于今后我在科学研究与工程设计等方面的发展都大有裨益。但是，实习内容细节还不够充分，方法还不够多元。在今后的学习生活中，我也会更加注重将课堂学习与亲身实验相结合，培养能力与兴趣，为未来的科研之路打下更加坚实的基础！

最后，非常感谢陶亮老师在课堂上的教诲，启发我们运用编程工具来进行理论基础上的亲身实践，感受更多有关计算方法的精彩。希望老师批评指正！

6 参考文献

[1] 李信真等. 计算方法.第 2 版[M]. 西北工业大学出版社, 2010.

附录

问题一：利用 MATLAB 软件绘制拟合函数曲线.m 源程序（指数函数拟合）

```
%% 绘制拟合函数曲线
clear all
clf
clc
load Problem_1_data.txt;
t=Problem_1_data(1,:);
x=Problem_1_data(2,:);
n=length(t);

% 拟合函数数据计算
gap=0.02;
cnt=0;
for i=0:gap:18
    cnt=cnt+1;
    xx(cnt)=i;
    yy(cnt)=11.3253*exp(-1.0567/i);
end

% 绘制原始数据点
plot(t,x,'r.','MarkerSize',16);

% 绘制拟合曲线
hold on
plot(xx,yy,'b-','LineWidth',2);
xlabel('时间/（分）');
ylabel('生物浓度y（10^{-3}）');
legend('原始数据点','指数函数拟合曲线');
axis([0,18,2,12]);
grid on;
box on;
```