云南大学数学与统计学实验教学中心

实验报告

课程名称：数值计算方法实验	**学期**：2015—2016 学年第一学期	**成绩**：
指导教师： 李耀堂	**学生姓名**：金洋	**学生学号**：20131910023
实验名称： **追赶法解方程组**		
实验编号：No. 4	**实验日期**： **2015.12.4**	**实验学时**： **3**
学院： 数学与统计学院	**专业**： **信息与计算科学**	**年级**：2013 级

一、实验目的

练习利用高斯消元法解线性方程组；

二、实验内容

编程用追赶法解下列严格对角优势的三对角线方程组

$$\begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -4 \\ & & & -1 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}=\begin{bmatrix} 100 \\ 200 \\ 200 \\ 200 \\ 100 \end{bmatrix}$$

三、实验环境

PC 计算机；

MATLAB R2014a；

四.实验方法：追赶法

追赶法简述：

在微分方程数值解及三次样条函数插值的计算中，常常会归结为 Y 一种特殊的稀疏矩阵，即所谓的三对角线方程组

$$\begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & a_i & b_i & c_i & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & a_n & b_n \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}=\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

简记为 $AX = f$.

此种方程组往往 n 较大，但零元素很多，若用高斯消元法解需要很大内存，零元素都得参加运算，速度也慢，于是考虑用 Crout 分解。A=LU,其中 L 是下三角阵，U 是单位上三角阵。

$$
\begin{bmatrix}
b_1 & c_1 & & & & & \\
a_2 & b_2 & c_2 & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & a_i & b_i & c_i & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & a_{n-1} & b_{n-1} & c_{n-1} \\
 & & & & & a_n & b_n
\end{bmatrix}
=
\begin{bmatrix}
\alpha_1 & & & & & & \\
\gamma_2 & \alpha_2 & & & & & \\
 & \gamma_3 & \alpha_3 & & & & \\
 & & \ddots & \ddots & & & \\
 & & & \gamma_i & \alpha_i & & \\
 & & & & \ddots & \ddots & \\
 & & & & & \gamma_n & \alpha_n
\end{bmatrix}
\cdot
\begin{bmatrix}
1 & \beta_1 & & & & & \\
 & 1 & \beta_2 & & & & \\
 & & 1 & \ddots & & & \\
 & & & \ddots & \beta_i & & \\
 & & & & 1 & & \\
 & & & & & \ddots & \beta_{n-1} \\
 & & & & & & 1
\end{bmatrix}
$$

$$
=\begin{bmatrix}
\alpha_1 & \alpha_1\beta_1 & & & & & \\
\gamma_2 & \gamma_2\beta_1+\alpha_2 & \alpha_2\beta_2 & & & & \\
 & \ddots & \ddots & \ddots & & & \\
 & & \gamma_i & \gamma_i\beta_{i-1}+\alpha_i & \alpha_i\beta_i & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & \gamma_{n-1} & \gamma_{n-1}\beta_{n-2}+\alpha_{n-1} & \alpha_{n-1}\beta_{n-1} \\
 & & & & & \gamma_n & \gamma_n\beta_{n-1}+\alpha_n
\end{bmatrix}
$$

对照矩阵中的元素可得

$$
\begin{aligned}
& a_i = \gamma_i && (i=2,3,\cdots,n) \\
& \alpha_i\beta_i = c_i && (i=1,2,\cdots,n-1) \\
& b_1 = a_1 && \\
& b_i = r_i\beta_{i-1}+a_i && (i=2,3,\cdots,n)
\end{aligned}
$$

计算得

$$
\begin{aligned}
& \gamma_i = a_i, i=2,3,\cdots,n; \\
& \alpha_1 = b_1, \beta_1 = \frac{c_1}{\alpha_1} \\
& \begin{cases} \alpha_i = b_i - \gamma_i\beta_{i-1} \\ \beta_i = \dfrac{c_i}{\alpha_i} \end{cases}, i=2,3,\cdots,n-1 \\
& \alpha_n = b_n - \gamma_n\beta_{n-1}
\end{aligned}
$$

求解 $Ax=f$ 可分为两步（追、赶）来实现，首先由上述公式求出 L、U，再解两个三角形方程组：

1） $Ly=f$,即

$$
\begin{bmatrix}
\alpha_1 & & & & & & \\
\gamma_2 & \alpha_2 & & & & & \\
 & \gamma_3 & \alpha_3 & & & & \\
 & & \ddots & \ddots & & & \\
 & & & \gamma_i & \alpha_i & & \\
 & & & & \ddots & \ddots & \\
 & & & & & \gamma_n & \alpha_n
\end{bmatrix}
\cdot
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
=
\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}
$$

$$y_1=\frac{f_1}{\alpha_1},y_i=(f_i-\gamma_i y_{i-1})/\alpha_i(i=1,2,\cdots,n).$$

2）$Ux=y$，即 $\begin{bmatrix}1 & \beta_1 & & & & & \\ & 1 & \beta_2 & & & & \\ & & 1 & \ddots & & & \\ & & & \ddots & \beta_i & & \\ & & & & 1 & & \\ & & & & & \ddots & \beta_{n-1} \\ & & & & & & 1\end{bmatrix}\cdot\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}=\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix}$

$$x_n=y_n,x_i=y_i-\beta_i x_{i+1} \qquad (i=n-1,n-2,\cdots,2,1).$$

五、实验过程

1 实验步骤

①编程：用 MATLAB 语言编出源程序。

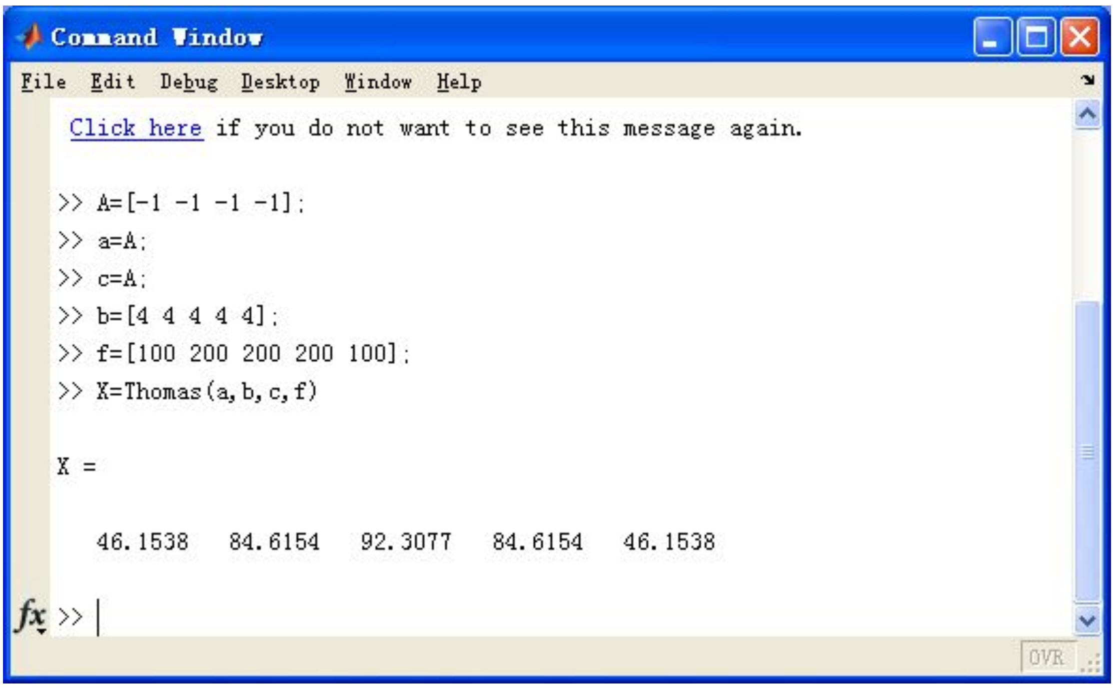
②开机,键入所编程序源代码.

③调试程序，修改错误至能正确运行.

④运行程序并输出计算结果.

⑤尝试改进

运行结果



即 $x(1) = 46.1538$

$x(2) = 84.6154$

$x(3) = 92.3077$

$x(4) = 84.6154$

$x(5) = 46.1538$

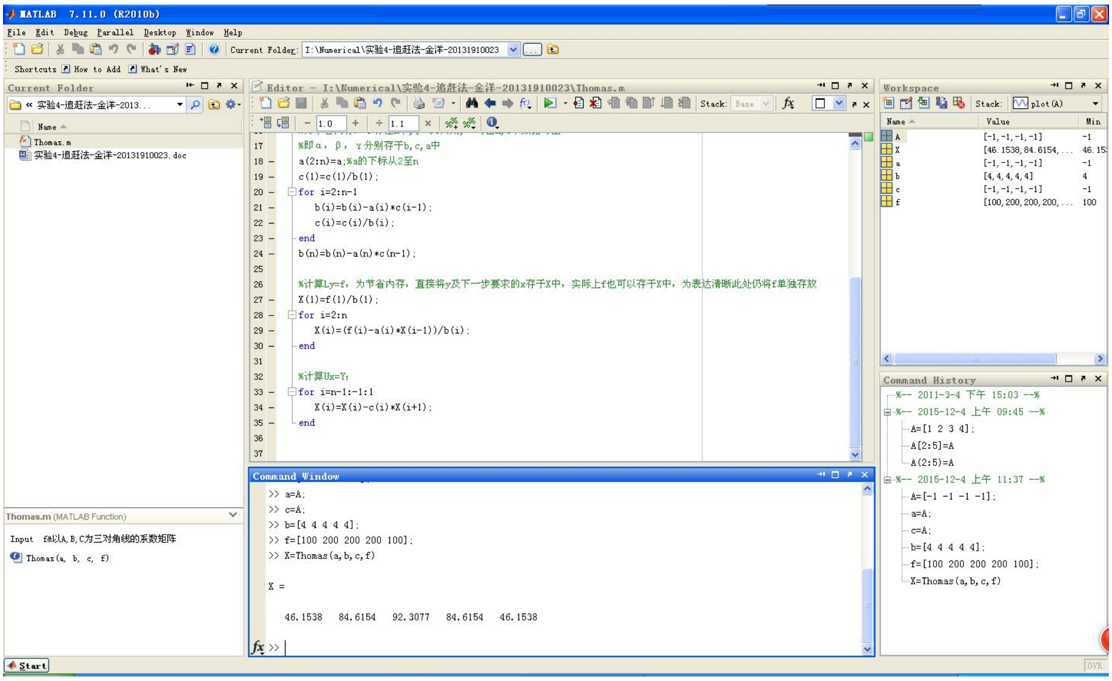
2 关键代码及其解释

```
%计算α， β， γ
%为节省内存， γ存在 A 中，β与 C 共用，α可由 A,C 中数据导出
%即α， β， γ分别存于 b,c,a 中
a(2:n)=a;%a 的下标从 2 至 n
c(1)=c(1)/b(1);
for i=2:n-1
   b(i)=b(i)-a(i)*c(i-1);
   c(i)=c(i)/b(i);
end
b(n)=b(n)-a(n)*c(n-1);

%计算 Ly=f，为节省内存，直接将 y 及下一步要求的 x 存于 X 中，实际上 f 也可以存于 X 中，为表达清晰此处仍将 f 单独存放
X(1)=f(1)/b(1);
for i=2:n
   X(i)=(f(i)-a(i)*X(i-1))/b(i);
end

%计算 Ux=Y;
for i=n-1:-1:1
   X(i)=X(i)-c(i)*X(i+1);
end
```

3 调试过程

一开始为了节省内存，将α， β， γ与 a,b,c 共用同一个空间，但是未搞清前后与谁对应，在编程途中遇阻不少；

后仔细分析对应关系后，程序一次性调试成功；



六、实验总结

1. 遇到的问题及解决过程

一开始为了节省内存，将α，β，γ与 a, b, c 共用同一个空间，但是未 搞清前后与谁对应，在编程途中遇阻不少

2. 产生的错误及原因分析

Crout 分解公式为：

$$\gamma_i = a_i, i = 2,3,\cdots,n;$$

$$\alpha_1 = b_1, \beta_1 = \frac{c_1}{\alpha_1}$$

$$\begin{cases} \alpha_i = b_i - \gamma_i\beta_{i-1} \\ \beta_i = \dfrac{c_i}{\alpha_i} \end{cases}, i = 2,3,\cdots,n-1$$

$$\alpha_n = b_n - \gamma_n\beta_{n-1}$$

追赶公式为：

$$y_1 = \frac{f_1}{\alpha_1}, y_i = (f_i - \gamma_i y_{i-1})/\alpha_i (i = 1,2,\cdots,n).$$

$$x_n = y_n, x_i = y_i - \beta_i x_{i+1} \qquad (i = n-1, n-2, \cdots, 2,1).$$

故为节省内存，γ存在 A 中，β与 C 共用，α可由 A,C 中数据导出，即α，β，γ分别存于 b,c,a 中；

计算 Ly=f，为节省内存，直接将 y 及下一步 Ux=y 要求的 x 存于 X 中，

实际上 f 也可以存于 X 中；

3．体会和收获。

①追赶法的中间运算没有数量级的很大变化，不会有严重的误差积累，所以此方法是比较稳定的。但在计算过程中要求 $\alpha_i(i=1,2,3,\cdots,n)$ 不能为零；

②通过分析算法，发现很多变量所使用的空间都可以共享，γ 存在 A 中，β 与 C 共用，α 可由 A,C 中数据导出，即 α，β，γ 分别存于 b,c,a 中；计算 Ly=f，为节省内存，直接将 y 及下一步 Ux=y 要求的 x 存于 X 中，实际上 f 也可以存于 X 中。这使得追赶法的空间复杂度为 O(n),而高斯消元法的空间复杂度为 $O(n^2)$；

③运行时间上，高斯消元法时间复杂度为 $O(n^2)$，追赶法的时间复杂度是 O(n),故对于具有三对角线方程组使用追赶法具有计算量小、占用内存单元少的特点；

七、程序源代码：

追赶法 Tridiagonal matrix algorithm - TDMA (Thomas algorithm)

Thomas.m

```
function X=Thomas(a,b,c,f)
%Input  f&以 A,B,C 为三对角线的系数矩阵
%       -a 1*(n-1)矩阵
%       -b 1*n 矩阵
%       -c 1*(n-1)矩阵
%       -f 1*n 矩阵
%Output -X 方程组的解

%求维数
n=length(f);

%初始化 X
X=zeros(1,n);

%计算α，β，γ
%为节省内存，γ存在 A 中，β与 C 共用，α可由 A,C 中数据导出
%即α，β，γ分别存于 b,c,a 中
a(2:n)=a;%a 的下标从 2 至 n
c(1)=c(1)/b(1);
for i=2:n-1
   b(i)=b(i)-a(i)*c(i-1);
```

```
    c(i)=c(i)/b(i);
end
b(n)=b(n)-a(n)*c(n-1);

%计算 Ly=f，为节省内存，直接将 y 及下一步要求的 x 存于 X 中，实际上 f 也可以存于 X 中，为表达清晰此处仍将 f 单独存放
X(1)=f(1)/b(1);
for i=2:n
    X(i)=(f(i)-a(i)*X(i-1))/b(i);
end

%计算 Ux=Y;
for i=n-1:-1:1
    X(i)=X(i)-c(i)*X(i+1);
end
```

八、教师评语