

## 云南大学数学与统计学院实验教学中心 实验报告

课程名称: 数学建模实验		学期: 2016~2017 学年下学期	
指导教师: 李朝迁			
学生: 刘鹏 20151910042 信计		学生: 王泽坤 20151910011 应数	学生: 段奕臣 20151910002 应数
实验名称: 线性代数方程组的数值解法 & MATLAB 相关命令学习			成绩:
实验编号: NO.4		实验日期: 2017 年 6 月 12 日	实验学时: 2
学院: 数学与统计学院		专业: 信息与计算科学	年级: 2015 级

### 一、实验目的

1. 掌握有关线性方程组的数值解法原理;
2. 学会用 MATLAB 进行数值求解。

### 二、实验内容

1. 完成课后有关习题, 用 MATLAB 进行编程;
2. 对线性方程组的 MATLAB 相关函数进行学习掌握。

### 三、实验平台

Windows10 Enterprise 1703 中文版操作系统;  
MATLAB R2017a 中文版。

### 四、实验记录与实验结果分析

#### 1 题

学习使用 Page103 页中的 MATLAB 命令函数。

#### Solution

#### lu

LU matrix factorization

#### Syntax

```
Y = lu(A)
[L,U] = lu(A)
[L,U,P] = lu(A)
[L,U,P,Q] = lu(A)
[L,U,P,Q,R] = lu(A)
[...] = lu(A,'vector')
[...] = lu(A,thresh)
[...] = lu(A,thresh,'vector')
```

#### Description

The `lu` function expresses a matrix  $A$  as the product of two essentially triangular matrices, one of them a permutation of a lower triangular

matrix and the other an upper triangular matrix. The factorization is often called the  $LU$ , or sometimes the  $LR$ , factorization.  $A$  can be rectangular.

$Y = \text{lu}(A)$  returns matrix  $Y$  that contains the strictly lower triangular  $L$ , i.e., without its unit diagonal, and the upper triangular  $U$  as submatrices. That is, if  $[L,U,P] = \text{lu}(A)$ , then  $Y = U+L\text{-eye}(\text{size}(A))$ . The permutation matrix  $P$  is not returned.

$[L, U] = \text{lu}(A)$  returns an upper triangular matrix in  $U$  and a permuted lower triangular matrix in  $L$  such that  $A = L*U$ . Return value  $L$  is a product of lower triangular and permutation matrices.

`[L,U,P] = lu(A)` returns an upper triangular matrix in U, a lower triangular matrix L with a unit diagonal, and a permutation matrix P, such that  $L*U = P*A$ . The statement `lu(A, 'matrix')` returns identical output values.

`[L, U, P, Q] = lu(A)` for sparse nonempty A, returns a unit lower triangular matrix L, an upper triangular matrix U, a row permutation matrix P, and a column reordering matrix Q, so that  $P*A*Q = L*U$ . If A is empty or not sparse, `lu` displays an error message. The statement `lu(A, 'matrix')` returns identical output values.

`[L, U, P, Q, R] = lu(A)` returns unit lower triangular matrix L, upper triangular matrix U, permutation matrices P and Q, and a diagonal scaling matrix R so that  $P*(R\backslash A) * Q = L*U$  for sparse non-empty A. Typically, but not always, the row-scaling leads to a sparser and more stable factorization. The statement `lu(A, 'matrix')` returns identical output values.

`[...] = lu(A, 'vector')` returns the permutation information in two row vectors p and q. You can specify from 1 to 5 outputs. Output p is defined as  $A(p, :) = L*U$ , output q is defined as  $A(p, q) = L*U$ , and output R is defined as  $R(:, p) \backslash A(:, q) = L*U$ .

`[...] = lu(A, thresh)` controls pivoting. This syntax applies to sparse matrices only. The `thresh` input is a one- or two-element vector of type `single` or `double` that defaults to `[0.1, 0.001]`. If A is a square matrix with a mostly symmetric structure and mostly nonzero diagonal, MATLAB® uses a symmetric pivoting strategy. For this strategy, the diagonal where

$$A(i, j) \geq \text{thresh}(2) * \max(\text{abs}(A(j : m, j)))$$

is selected. If the diagonal entry fails this test, a pivot entry below the diagonal is selected, using `thresh(1)`. In this case, L has entries with absolute value  $1/\min(\text{thresh})$  or less.

If A is not as described above, MATLAB uses an asymmetric strategy. In this case, the sparsest row `i` where

$$A(i, j) \geq \text{thresh}(1) * \max(\text{abs}(A(j:m, j)))$$

is selected. A value of 1.0 results in conventional partial pivoting. Entries in L have an absolute value of  $1/\text{thresh}(1)$  or less. The second element of the `thresh` input vector is not used when MATLAB uses an asymmetric strategy.

Smaller values of `thresh(1)` and `thresh(2)` tend to lead to sparser LU factors, but the solution can become inaccurate. Larger values can lead to a more accurate solution (but not always), and usually an increase in the total work and memory usage. The statement `lu(A, thresh, 'matrix')` returns identical output values.

`[...] = lu(A, thresh, 'vector')` controls the pivoting strategy and also returns the permutation information in row vectors, as described above. The `thresh` input must precede `'vector'` in the input argument list.

**Note:** In rare instances, incorrect factorization results in  $P*A*Q \neq L*U$ . Increase `thresh`, to a maximum of 1.0 (regular partial pivoting), and try again.

### Arguments

<b>A</b>	Rectangular matrix to be factored.
<b>thresh</b>	Pivot threshold for sparse matrices. Valid values are in the interval <code>[0,1]</code> . If you specify the fourth output Q, the default is 0.1. Otherwise, the default is 1.0.
<b>L</b>	Factor of A. Depending on the form of the function, L is either a unit lower triangular matrix, or else the product of a unit lower triangular matrix with P'.
<b>U</b>	Upper triangular matrix that is a factor of A.
<b>P</b>	Row permutation matrix satisfying the equation $L*U = P*A$ , or $L*U = P*A*Q$ . Used for numerical stability.
<b>Q</b>	Column permutation matrix satisfying the equation $P*A*Q = L*U$ . Used to reduce fill-in in the sparse case.

R	Row-scaling matrix
---	--------------------

## 2 题

对 Page<sub>99</sub> 中 (51) 式, 借助 (58) 和 (61) 式进行编程, 得到结果如表 5.3 所示。

**Solution:**

先看如下的数字例子:

$$\begin{cases} 9x_1 - x_2 - x_3 = 7 \\ -x_1 + 10x_2 - x_3 \\ -x_1 - x_2 + 15x_3 = 13 \end{cases} \quad (51)$$

线性方程组 (51) 可等价地写为

$$\begin{cases} x_1 = \frac{1}{9}x_2 + \frac{1}{9}x_3 + \frac{7}{9}, \\ x_2 = \frac{1}{10}x_1 + \frac{1}{10}x_3 + \frac{8}{10}, \\ x_3 = \frac{1}{15}x_1 + \frac{1}{15}x_2 + \frac{13}{15}. \end{cases} \quad (52)$$

利用线性方程组 (52) 可以进行如下形式的迭代 (用上标  $(k)$  表示迭代步数):

$$\begin{cases} x_1^{(k+1)} = \frac{1}{9}x_2^{(k)} + \frac{1}{9}x_3^{(k)} + \frac{7}{9}, \\ x_2^{(k+1)} = \frac{1}{10}x_1^{(k)} + \frac{1}{10}x_3^{(k)} + \frac{8}{10}, \\ x_3^{(k+1)} = \frac{1}{15}x_1^{(k)} + \frac{1}{15}x_2^{(k)} + \frac{13}{15}. \end{cases} \quad (53)$$

对选定的初始解  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})^T$ , 可由 (53) 式迭代计算  $\mathbf{x}^{(1)}$ ,  $\mathbf{x}^{(2)}$ ,  $\dots$  如取  $\mathbf{x}^{(0)} = (0, 0, 0)^T$ , 计算至  $k=3$  时可得  $\mathbf{x}^{(4)} = (0.9987, 0.9988, 0.9991)^T$ , 已经与线性方程组 (51) 的精确解  $\mathbf{x} = (1, 1, 1)^T$  非常接近。这就是求解线性方程组的雅可比迭代法。

从 (53) 式的迭代过程很容易发现, 计算  $x_2^{(k+1)}$  时  $x_1^{(k+1)}$  已经知道, 计算  $x_3^{(k+1)}$  时,  $x_1^{(k+1)}$  和  $x_2^{(k+1)}$  均已知道。因此, 如果在计算  $x_2^{(k+1)}$  和  $x_3^{(k+1)}$  时, 用最新的  $x_1^{(k+1)}$  和  $x_2^{(k+1)}$  进行迭代, 则可得知如下的公式:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{9}x_2^{(k)} + \frac{1}{9}x_3^{(k)} + \frac{7}{9}, \\ x_2^{(k+1)} = \frac{1}{10}x_1^{(k+1)} + \frac{1}{10}x_3^{(k)} + \frac{8}{10}, \\ x_3^{(k+1)} = \frac{1}{15}x_1^{(k+1)} + \frac{1}{15}x_2^{(k+1)} + \frac{13}{15}. \end{cases} \quad (54)$$

称为高斯-赛德尔迭代法。

雅克比迭代:  $\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$

高斯-赛德尔迭代:  $\mathbf{x}^{(k+1)} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$

程序代码:

```
1 % filename: Jacobi
2 A = [9 -1 -1;-1 10 -1;-1 -1 15];
3 b = [7;8;13];
4
5 D = diag(diag(A));
6 L = -1 * tril(A,-1);
7 U = -1 * triu(A,1);
8
```

```

9   B = D^(-1) * (L + U);
10  f = D^(-1) * b;
11
12  x = [0 0 0]';
13  for k = 1 : 4
14      x = B * x + f;
15  end
16  fprintf('%1.4f\n',x);

```

程序代码 1

运行结果:

```

命令窗口
>> Jacobi
0.9987
0.9988
0.9991

```

运行结果 1

代码分析:

可以看到, 利用雅可比矩阵迭代四次, 只能精确二到三位。收敛速度不太好。

程序代码:

```

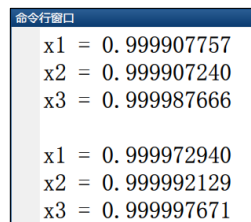
1   % filename: GS
2
3   %- 用高斯-赛德尔方法进行迭代 -%
4
5   clc
6
7   A1 = [1/9,1/9,7/9];
8   A2 = [1/10,1/10,8/10];
9   A3 = [1/15,1/15,13/15];
10
11  x1 = sum(A1 .* [0 0 1]);
12  x2 = sum(A2 .* [0 0 1]);
13  x3 = sum(A3 .* [0 0 1]);
14
15  for k = 1 : 4
16      x1 = sum(A1 .* [x2,x3,1]);
17      x2 = sum(A2 .* [x1,x3,x1]);
18      x3 = sum(A3 .* [x1,x2,1]);
19  end
20  fprintf('x1 = %1.9f\nx2 = %1.9f\nx3 = %1.9f\n\n',x1,x2,x3);
21
22  %- 用公式解决 -%
23
24  A = [9 -1 -1;-1 10 -1;-1 -1 15];
25  b = [7;8;13];
26
27  D = diag(diag(A));
28  L = -1 * tril(A,-1);

```

```
29 U = -1 * triu(A,1);
30
31 B = (D - L)^(-1) * U;
32 f = (D - L)^(-1) * b;
33
34 x = [0 0 0]';
35 for k = 1 : 4
36     x = B * x + f;
37 end
38 fprintf('x1 = %1.9f\nx2 = %1.9f\nx3 = %1.9f\n',x(1),x(2),x(3));
```

程序代码 2

运行结果：



k	x1	x2	x3
1	0.999907757	0.999907240	0.999987666
2	0.999972940	0.999992129	0.999997671

运行结果 2

代码分析：

是采用分数形式，所以结果稍有出入（虽然都是四次迭代）。

## 3 题

已知方程组  $\mathbf{Ax} = \mathbf{b}$ ，其中  $\mathbf{A} \in \mathbb{R}^{20 \times 20}$ ，定义为

$$\mathbf{A} = \begin{bmatrix} 3 & -\frac{1}{2} & -\frac{1}{4} & & & \\ -\frac{1}{2} & 3 & -\frac{1}{2} & -\frac{1}{4} & & \\ -\frac{1}{4} & -\frac{1}{2} & 3 & -\frac{1}{2} & \ddots & \\ & \ddots & \ddots & \ddots & \ddots & -\frac{1}{4} \\ & & -\frac{1}{4} & -\frac{1}{2} & 3 & -\frac{1}{2} \\ & & & -\frac{1}{4} & -\frac{1}{2} & 3 \end{bmatrix}$$

试通过迭代法求解此方程组，认识迭代法收敛的含义以及迭代初值和方程组系数矩阵性质对收敛速度的影响。实验要求

- (1) 选取不同的初始向量  $\mathbf{x}^{(0)}$  和不同的方程组右端向量  $\mathbf{b}$ ，给定迭代误差要求，用雅可比迭代法和高斯-赛德尔迭代法计算，观测得到的迭代向量序列是否均收敛？若收敛，记录迭代次数，分析计算结果并得出你的结论。
- (2) 取定右端向量  $\mathbf{b}$  和初始向量  $\mathbf{x}^{(0)}$ ，将  $\mathbf{A}$  的主对角线元素成倍增长若干次，非主对角线元素不变，每次用雅可比迭代法计算，要求迭代误差满足  $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_{\infty} < 10^{-5}$ ，比较收敛速度，分析现象并得出你的结论。

**Solution:**

(1)

程序代码:

```
1 % filename: iter_test
2
3 clear all;clc
4 %- 矩阵输入 -%
5 n = 20;
6 A1 = sparse(1:n,1:n,3,n,n); % 对角线元素
7 A2 = sparse(1:n-1,2:n,-1/2,n,n);
8 A3 = sparse(1:n-2,3:n,-1/4,n,n);
9 A = A1 + A2 + A2' + A3 + A3';
10
11 D = diag(diag(A));
12 L = tril(A,-1);
13 U = triu(A,1);
14
15 b = [1 : 20]';
16
17 B = (D - L)^(-1) * U;
18 f = (D - L)^(-1) * b;
19
```

```

20 x0 = zeros(20,1);
21 x1 = B * x0 + f;
22 count = 1;
23
24 while sum(abs(x1 - x0)) > 1e-6
25     x0 = x1;
26     x1 = B * x0 + f;
27     count = count + 1;
28     fprintf('第 %1.0f 次  [' ,count)
29     for i = 1 : 20
30         fprintf('%1.4f ',x0(i))
31     end
32     fprintf(']\n\n')
33 end
34
35 fprintf('换一个初值\n')
36 clear all;
37 %- 矩阵输入  -%
38 n = 20;
39 A1 = sparse(1:n,1:n,3,n,n);      % 对角线元素
40 A2 = sparse(1:n-1,2:n,-1/2,n,n);
41 A3 = sparse(1:n-2,3:n,-1/4,n,n);
42 A = A1 + A2 + A2' + A3 + A3';
43
44 D = diag(diag(A));
45 L = tril(A,-1);
46 U = triu(A,1);
47
48 b = [1 : 20]';
49
50 B = (D - L)^(-1) * U;
51 f = (D - L)^(-1) * b;
52
53 x0 = ones(20,1);
54 x1 = B * x0 + f;
55 count = 1;
56
57 while sum(abs(x1 - x0)) > 1e-6
58     x0 = x1;
59     x1 = B * x0 + f;
60     count = count + 1;
61     fprintf('第 %1.0f 次  [' ,count)
62     for i = 1 : 20
63         fprintf('%1.4f ',x0(i))
64     end
65     fprintf(']\n\n')
66
67 end
68

```



```

69 fprintf('换一个b值\n')
70 clear all;
71 %- 矩阵输入 -%
72 n = 20;
73 A1 = sparse(1:n,1:n,3,n,n); % 对角线元素
74 A2 = sparse(1:n-1,2:n,-1/2,n,n);
75 A3 = sparse(1:n-2,3:n,-1/4,n,n);
76 A = A1 + A2 + A2' + A3 + A3';
77
78 D = diag(diag(A));
79 L = tril(A,-1);
80 U = triu(A,1);
81
82 b = [1 : 0.5 : 10.5]';
83
84 B = (D - L)^(-1) * U;
85 f = (D - L)^(-1) * b;
86
87 x0 = zeros(20,1);
88 x1 = B * x0 + f;
89 count = 1;
90
91 while sum(abs(x1 - x0)) > 1e-6
92     x0 = x1;
93     x1 = B * x0 + f;
94     count = count + 1;
95     fprintf('第 %1.0f 次  [' ,count)
96     for i = 1 : 20
97         fprintf('%1.4f ',x0(i))
98     end
99     fprintf(']\n\n')
100
101 end

```

程序代码 3

运行结果:

```

第 2 次 [0.3333 0.6111 0.8704 1.1373 1.4046 1.6711 1.9378 2.2044 2.4711 2.7378 3.0044 3.2711 3.5378 3.8044 4.0711 4.3378 4.6044 4.8711 5.1378 5.4044 ]
第 3 次 [0.1590 0.4003 0.6134 0.8244 1.0382 1.2516 1.4649 1.6782 1.8916 2.1049 2.3182 2.5316 2.7449 2.9582 3.1716 3.3849 3.5982 3.8116 4.4975 5.5995 ]
第 4 次 [0.2155 0.4598 0.6815 0.9041 1.1285 1.3526 1.5765 1.8005 2.0245 2.2485 2.4725 2.6965 2.9205 3.1445 3.3685 3.5925 3.7772 3.8549 4.4428 5.6050 ]
第 5 次 [0.1999 0.4444 0.6645 0.8847 1.1070 1.3290 1.5508 1.7727 1.9945 2.2164 2.4383 2.6601 2.8820 3.1039 3.3290 3.5691 3.7817 3.8647 4.4399 5.6046 ]
第 6 次 [0.2039 0.4482 0.6686 0.8893 1.1120 1.3344 1.5566 1.7789 2.0012 2.2235 2.4458 2.6681 2.8901 3.1104 3.3308 3.5667 3.7805 3.8657 4.4399 5.6045 ]
第 7 次 [0.2029 0.4473 0.6677 0.8882 1.1109 1.3332 1.5553 1.7775 1.9997 2.2219 2.4442 2.6666 2.8893 3.1106 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 8 次 [0.2031 0.4475 0.6679 0.8885 1.1111 1.3334 1.5556 1.7778 2.0001 2.2223 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 9 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5555 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 10 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 11 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 12 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
换一个初值
第 2 次 [0.0833 0.4028 0.6759 0.9371 1.2042 1.4712 1.7378 2.0044 2.2711 2.5378 2.8044 3.0711 3.3378 3.6044 3.8711 4.1378 4.4044 4.6711 5.0211 5.4406 ]
第 3 次 [0.2099 0.4409 0.6525 0.8645 1.0782 1.2916 1.5049 1.7182 1.9316 2.1449 2.3582 2.5716 2.7849 2.9982 3.2116 3.4249 3.6313 3.8191 4.4874 5.6005 ]

```

```

第 4 次 [0.2055 0.4516 0.6737 0.8961 1.1205 1.3446 1.5685 1.7925 2.0165 2.2405 2.4645 2.6885 2.9125 3.1365 3.3611 3.5883 3.7780 3.8657 4.4423 5.6049 ]
第 5 次 [0.2019 0.4461 0.6661 0.8863 1.1086 1.3306 1.5524 1.7743 1.9961 2.2180 2.4399 2.6617 2.8835 3.1051 3.3293 3.5686 3.7815 3.8649 4.4399 5.6046 ]
第 6 次 [0.2035 0.4479 0.6683 0.8890 1.1117 1.3341 1.5563 1.7786 2.0009 2.2232 2.4455 2.6678 2.8900 3.1104 3.3309 3.5667 3.7805 3.8657 4.4399 5.6045 ]
第 7 次 [0.2030 0.4474 0.6677 0.8883 1.1109 1.3332 1.5554 1.7776 1.9998 2.2220 2.4442 2.6666 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 8 次 [0.2031 0.4475 0.6679 0.8885 1.1111 1.3334 1.5556 1.7778 2.0000 2.2223 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 9 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5555 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 10 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 11 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 12 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]

换一个b值
第 2 次 [0.3333 0.4444 0.5648 0.7022 0.8359 0.9688 1.1022 1.2356 1.3689 1.5022 1.6356 1.7689 1.9022 2.0356 2.1689 2.3022 2.4356 2.5689 2.7022 2.8356 ]
第 3 次 [0.2122 0.3120 0.4103 0.5189 0.6260 0.7324 0.8391 0.9458 1.0524 1.1591 1.2658 1.3724 1.4791 1.5858 1.6924 1.7991 1.9058 2.0124 2.3665 2.9379 ]
第 4 次 [0.2471 0.3472 0.4496 0.5641 0.6765 0.7882 0.9002 1.0123 1.1243 1.2363 1.3483 1.4603 1.5723 1.6843 1.7963 1.9083 1.9996 2.0351 2.3379 2.9408 ]
第 5 次 [0.2380 0.3384 0.4400 0.5534 0.6647 0.7754 0.8863 0.9973 1.1082 1.2191 1.3301 1.4410 1.5519 1.6629 1.7755 1.8960 2.0020 2.0403 2.3363 2.9406 ]
第 6 次 [0.2403 0.3405 0.4423 0.5558 0.6674 0.7783 0.8894 1.0006 1.1118 1.2229 1.3340 1.4452 1.5562 1.6663 1.7764 1.8947 2.0014 2.0408 2.3363 2.9405 ]
第 7 次 [0.2397 0.3400 0.4418 0.5553 0.6668 0.7776 0.8887 0.9999 1.1110 1.2221 1.3332 1.4444 1.5557 1.6664 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
第 8 次 [0.2399 0.3401 0.4419 0.5554 0.6669 0.7778 0.8889 1.0000 1.1111 1.2222 1.3333 1.4444 1.5557 1.6663 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
第 9 次 [0.2398 0.3401 0.4419 0.5554 0.6669 0.7777 0.8889 1.0000 1.1111 1.2222 1.3333 1.4445 1.5557 1.6663 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
第 10 次 [0.2398 0.3401 0.4419 0.5554 0.6669 0.7778 0.8889 1.0000 1.1111 1.2222 1.3333 1.4445 1.5557 1.6663 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
第 11 次 [0.2398 0.3401 0.4419 0.5554 0.6669 0.7778 0.8889 1.0000 1.1111 1.2222 1.3333 1.4445 1.5557 1.6663 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
第 12 次 [0.2398 0.3401 0.4419 0.5554 0.6669 0.7778 0.8889 1.0000 1.1111 1.2222 1.3333 1.4445 1.5557 1.6663 1.7767 1.8947 2.0013 2.0408 2.3363 2.9405 ]
>>

```

### 运行结果 3

#### 代码分析：

由于矩阵有点大，所以截图比较麻烦，这里综合地分析一下。这个矩阵与希尔伯特矩阵很像，但是这个矩阵是稀疏矩阵。这里采用 `sparse` 方法进行矩阵生成。可以发现，改变初值与 `b` 矩阵数值，都没能使得这个系数矩阵 `A` 下的 `x` 不收敛，而且基本都在 12 次完成了收敛。

#### (2)

#### 程序代码：

```

1  % filename: iter_test
2
3  clear all;clc
4  %- 矩阵输入 -%
5  n = 20;
6  A1 = sparse(1:n,1:n,3,n,n);    % 对角线元素
7  A2 = sparse(1:n-1,2:n,-1/2,n,n);
8  A3 = sparse(1:n-2,3:n,-1/4,n,n);
9  A = A1 + A2 + A2' + A3 + A3';
10
11 D = diag(diag(A));
12 L = tril(A,-1);
13 U = triu(A,1);
14
15 b = [1 : 20]';
16
17 B = (D - L)^(-1) * U;

```

```

18 f = (D - L)^(-1) * b;
19
20 x0 = zeros(20,1);
21 x1 = B * x0 + f;
22 count = 1;
23
24 while sum(abs(x1 - x0)) > 1e-5
25     x0 = x1;
26     x1 = B * x0 + f;
27     count = count + 1;
28     fprintf('第 %1.0f 次  ',count)
29     for i = 1 : 20
30         fprintf('%1.4f ',x0(i))
31     end
32     fprintf(']\n\n')
33 end
34
35
36 fprintf('3 倍\n\n');
37 clear all
38 n = 20;
39 A1 = sparse(1:n,1:n,3,n,n);      % 对角线元素
40 A2 = sparse(1:n-1,2:n,-1/2,n,n);
41 A3 = sparse(1:n-2,3:n,-1/4,n,n);
42 A = A1 + A2 + A2' + A3 + A3';
43
44 D = diag(diag(A)) * 3;
45 L = tril(A,-1);
46 U = triu(A,1);
47
48 b = [1 : 20]';
49
50 B = (D - L)^(-1) * U;
51 f = (D - L)^(-1) * b;
52
53 x0 = zeros(20,1);
54 x1 = B * x0 + f;
55 count = 1;
56
57 while sum(abs(x1 - x0)) > 1e-5
58     x0 = x1;
59     x1 = B * x0 + f;
60     count = count + 1;
61     fprintf('第 %1.0f 次  ',count)
62     for i = 1 : 20
63         fprintf('%1.4f ',x0(i))
64     end
65     fprintf(']\n\n')
66 end

```

```

67
68 fprintf('90 倍\n\n');
69 clear all
70 n = 20;
71 A1 = sparse(1:n,1:n,3,n,n); % 对角线元素
72 A2 = sparse(1:n-1,2:n,-1/2,n,n);
73 A3 = sparse(1:n-2,3:n,-1/4,n,n);
74 A = A1 + A2 + A2' + A3 + A3';
75
76 D = diag(diag(A)) * 90;
77 L = tril(A,-1);
78 U = triu(A,1);
79
80 b = [1 : 20]';
81
82 B = (D - L)^(-1) * U;
83 f = (D - L)^(-1) * b;
84
85 x0 = zeros(20,1);
86 x1 = B * x0 + f;
87 count = 1;
88
89 while sum(abs(x1 - x0)) > 1e-5
90     x0 = x1;
91     x1 = B * x0 + f;
92     count = count + 1;
93     fprintf('第 %1.0f 次  [' ,count)
94     for i = 1 : 20
95         fprintf('%1.4f ',x0(i))
96     end
97     fprintf(']\n\n')
98 end

```

程序代码 4

运行结果:

```

第 2 次 [0.3333 0.6111 0.8704 1.1373 1.4046 1.6711 1.9378 2.2044 2.4711 2.7378 3.0044 3.2711 3.5378 3.8044 4.0711 4.3378 4.6044 4.8711 5.1378 5.4044 ]
第 3 次 [0.1590 0.4003 0.6134 0.8244 1.0382 1.2516 1.4649 1.6782 1.8916 2.1049 2.3182 2.5316 2.7449 2.9582 3.1716 3.3849 3.5982 3.8116 4.4975 5.5995 ]
第 4 次 [0.2155 0.4598 0.6815 0.9041 1.1285 1.3526 1.5765 1.8005 2.0245 2.2485 2.4725 2.6965 2.9205 3.1445 3.3685 3.5925 3.7772 3.8549 4.4428 5.6050 ]
第 5 次 [0.1999 0.4444 0.6645 0.8847 1.1070 1.3290 1.5508 1.7727 1.9945 2.2164 2.4383 2.6601 2.8820 3.1039 3.3290 3.5691 3.7817 3.8647 4.4399 5.6046 ]
第 6 次 [0.2039 0.4482 0.6686 0.8893 1.1120 1.3344 1.5566 1.7789 2.0012 2.2235 2.4458 2.6681 2.8901 3.1104 3.3308 3.5667 3.7805 3.8657 4.4399 5.6045 ]
第 7 次 [0.2029 0.4473 0.6677 0.8882 1.1109 1.3332 1.5553 1.7775 1.9997 2.2219 2.4442 2.6666 2.8893 3.1106 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 8 次 [0.2031 0.4475 0.6679 0.8885 1.1111 1.3334 1.5556 1.7778 2.0001 2.2223 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 9 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5555 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 10 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
第 11 次 [0.2031 0.4475 0.6678 0.8884 1.1111 1.3334 1.5556 1.7778 2.0000 2.2222 2.4444 2.6667 2.8892 3.1105 3.3313 3.5667 3.7803 3.8657 4.4399 5.6045 ]
3 倍

```

第 2 次	[0.1111 0.2160 0.3182 0.4208 0.5233 0.6259 0.7285 0.8310 0.9336 1.0362 1.1387 1.2413 1.3439 1.4464 1.5490 1.6515 1.7541 1.8567 1.9592 2.0618 ]
第 3 次	[0.0903 0.1878 0.2825 0.3771 0.4718 0.5664 0.6611 0.7558 0.8505 0.9451 1.0398 1.1345 1.2292 1.3238 1.4185 1.5132 1.6078 1.7025 1.8573 2.0717 ]
第 4 次	[0.0928 0.1909 0.2861 0.3813 0.4766 0.5719 0.6672 0.7624 0.8577 0.9530 1.0483 1.1436 1.2388 1.3341 1.4294 1.5247 1.6183 1.7070 1.8562 2.0717 ]
第 5 次	[0.0926 0.1906 0.2858 0.3809 0.4762 0.5714 0.6666 0.7619 0.8571 0.9523 1.0476 1.1428 1.2380 1.3333 1.4285 1.5241 1.6181 1.7071 1.8562 2.0717 ]
第 6 次	[0.0926 0.1906 0.2858 0.3809 0.4762 0.5714 0.6667 0.7619 0.8571 0.9524 1.0476 1.1429 1.2381 1.3333 1.4286 1.5241 1.6181 1.7071 1.8562 2.0717 ]
第 7 次	[0.0926 0.1906 0.2858 0.3809 0.4762 0.5714 0.6667 0.7619 0.8571 0.9524 1.0476 1.1429 1.2381 1.3333 1.4286 1.5241 1.6181 1.7071 1.8562 2.0717 ]
90 倍	
第 2 次	[0.0037 0.0074 0.0111 0.0148 0.0185 0.0222 0.0259 0.0296 0.0333 0.0369 0.0406 0.0443 0.0480 0.0517 0.0554 0.0591 0.0628 0.0665 0.0702 0.0739 ]
第 3 次	[0.0037 0.0074 0.0110 0.0147 0.0184 0.0221 0.0258 0.0295 0.0331 0.0368 0.0405 0.0442 0.0479 0.0516 0.0552 0.0589 0.0626 0.0663 0.0701 0.0739 ]
>>	

运行结果 4

代码分析：

经过我多次尝试，发现只要倍增对角线元素，收敛速度就会加快。

## 4 题

设国民经济由农业、制造业和服务业三个部门构成，已知某年它们之间的投入产出关系、外部关系、初始投入等如表 5.6 所示。

表 5.6 国民经济三个部门之间的投入产出表 单位：亿元

产出 投入	农业	制造业	服务业	外部需求	总产出
农业	15	20	30	35	100
制造业	30	10	45	115	200
服务业	20	60	0	70	150
初始投入	35	110	75		
总投入	100	200	150		

根据表 5.6 回答下列问题：

- (1) 如果今年对农业、制造业和服务业的外部需求分别为 50, 150, 100 亿元，问这三个部门的总产出分别应为多少？
- (2) 如果有三个部门的外部需求分别增加 1 个单位，问它们的总产出应分别增加多少？

## solution

程序代码：

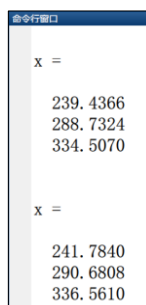
```

1  % filename: Input_and_output
2
3  clc;clear all;
4  A = [15/100 30/100 20/100
5        20/200 10/200 60/200
6        30/150 45/150 0];
7
8  d = [50 150 200]';
9  B = eye(3) - A;
10 x = B\d
11
12 %- 外部需求分别增加 1 -%
13 d = d + 1;
14 x = B\d

```

程序代码 5

运行结果：



```

x =

    239.4366
    288.7324
    334.5070

x =

    241.7840
    290.6808
    336.5610

```

运行结果 5

**代码分析：**

通过已有的模型，直接解方程组就好了。

## 五、实验体会



## 六、参考文献

- [1] 大学数学实验/姜启源, 谢金星, 邢文训, 张立平, 北京: 清华大学出版社, 2010.12
- [2] MATLAB 教程/张志涌, 杨祖樱, 北京: 北京航空航天大学出版社, 2015.1

## 七、教师评语