数学实验第三次实验报告

计 76　张翔　2017011568

2020 年 3 月 21 日

1　实验目的

1) 学会用 MATLAB 软件数值求解线性代数方程组，对迭代法的收敛性和解的稳定性作初步分析；
2) 通过实例学习用线性代数方程组解决简化的实际问题。

2　计算题

2.1　Ch5-P1 误差

2.1.1　理论计算

题目要求计算形如 $\mathbf{Ax} = \mathbf{b}$ 形式的线性方程组的解，由于 $\mathbf{b}$ 为矩阵 $\mathbf{A}$ 的行和，显然有 $x = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^{\mathrm{T}}$。

2.1.2　主要算法

Vandermonde 矩阵可以使用 `vand` 命令生成，并使用 `fliplr` 将其左右翻转成题目中要求的形式；而 Hilbert 矩阵可以使用 `hilb` 命令生成。构造完矩阵后可以计算行和，并计算最终方程的解。矩阵条件数可使用 `cond` 命令计算。

而扰动 $\mathbf{A}$ 或者 $\mathbf{b}$ 造成的误差上限可以由下面的式子来估算

$$\frac{||\delta \mathbf{x}||}{||\mathbf{x}||} \leq Cond(\mathbf{A}) \cdot \frac{||\delta \mathbf{b}||}{||\mathbf{b}||}$$

$$\frac{||\delta \mathbf{x}||}{||\mathbf{x}||} \leq \frac{Cond(\mathbf{A})}{1 - Cond(\mathbf{A}) \cdot \frac{||\delta \mathbf{A}||}{||\mathbf{A}||}} \cdot \frac{||\delta \mathbf{A}||}{||\mathbf{A}||}$$

2.1.3　Matlab 程序

```
% MathExp 5, P1
n_list = [5, 7, 9, 11];
eps_list = [1e-10, 1e-8, 1e-6];

res_len = length(n_list) * length(eps_list);
original_sol_list = cell(1,length(n_list));
cond_number_list = cell(1,length(n_list));
error_list = cell(1,res_len);
error_est = cell(1,res_len);
perturb_sol_list = cell(1,res_len);
```

```
11
12  for i = 1:length(n_list)
13      n = n_list(i);
14
15      x = 1:0.1:1+0.1*(n-1);
16      %% construct vandermonde and hilbert matrix
17      A1 = fliplr(vander(x));
18      A2 = hilb(n);
19
20      %% Calc row sum
21      b1 = sum(A1, 2);
22      b2 = sum(A2, 2);
23
24      fprintf("%d-order matrix\n", n);
25      %% obtain original solution
26      x1 = A1 \ b1;
27      x2 = A2 \ b2;
28      original_sol_list(i) = {[x1; x2]};
29
30
31      %% calc condition number
32      a1_cond = cond(A1);
33      a2_cond = cond(A2);
34      cond_number_list(i) = {[a1_cond; a2_cond]};
35
36      %% pertubate
37      for j = 1:length(eps_list)
38          eps = eps_list(j);
39          idx = (i - 1) * length(eps_list) + j;
40          A1_pert = A1;
41          A1_pert(n, n) = A1_pert(n, n) + eps;
42          A2_pert = A2;
43          A2_pert(n, n) = A2_pert(n, n) + eps;
44          b1_pert = b1;
45          b1_pert(n) = b1_pert(n) + eps;
46          b2_pert = b2;
47          b2_pert(n) = b2_pert(n) + eps;
48
49          %% estimate err by cond number
50          a1_relative_perturb = norm(A1_pert - A1) / norm(A1);
51          a2_relative_perturb = norm(A2_pert - A2) / norm(A2);
52          a1_perturb_err_est = a1_cond * a1_relative_perturb / ...
53              (1 - a1_cond * a1_relative_perturb);
54          a2_perturb_err_est = a2_cond * a2_relative_perturb / ...
55              (1 - a2_cond * a2_relative_perturb);
```

```
56
57          b1_relative_perturb = norm(b1_pert - b1) / norm(b1);
58          b2_relative_perturb = norm(b2_pert - b2) / norm(b2);
59          b1_perturb_err_est = a1_cond * b1_relative_perturb;
60          b2_perturb_err_est = a2_cond * b2_relative_perturb;
61          error_est(idx) = {[a1_perturb_err_est; a2_perturb_err_est;...
62              b1_perturb_err_est; b2_perturb_err_est]};
63
64          real_sol = ones([n 1]);
65          a1_p_sol = A1_pert \ b1;
66          a2_p_sol = A2_pert \ b2;
67          b1_p_sol = A1 \ b1_pert;
68          b2_p_sol = A2 \ b2_pert;
69
70          perturb_sol_list(idx) = {[a1_p_sol; a2_p_sol; b1_p_sol; b2_p_sol
               ]};
71
72          a1_p_err = norm(a1_p_sol - real_sol) / norm(real_sol);
73          a2_p_err = norm(a2_p_sol - real_sol) / norm(real_sol);
74          b1_p_err = norm(b1_p_sol - real_sol) / norm(real_sol);
75          b2_p_err = norm(b2_p_sol - real_sol) / norm(real_sol);
76
77          error_list(idx) = {[a1_p_err; a2_p_err; b1_p_err; b2_p_err]};
78      end
79  end
```

2.1.4 计算结果及分析

使用 Matlab 自带左乘计算解时，$n \leq 9$ 时得到的结果为元素均为 1.0000 的向量，说明程序是正确的。$n = 11$ 时结果如下

表 1: $n = 11$ 时方程的解

$\mathbf{A_1x = b_1}$	1.0000	1.0000	1.0001	0.9999	1.0002	0.9999	1.0001	1.0000	1.0000	1.0000	1.0000
$\mathbf{A_2x = b_2}$	1.0000	1.0000	1.0000	1.0002	0.9988	1.0042	0.9909	1.0125	0.9896	1.0048	0.9990

可以看到数值解与理论解相比，出现了比较明显的偏差，且 $\mathbf{A_2x = b_2}$ 的偏差更大。

计算得到不同 n 下矩阵的条件数如下：

表 2: 不同 n 值下，两个矩阵的条件数

n	5	7	9	11
$Cond(\mathbf{A_1})$	3.5740×10^5	8.7385×10^7	2.2739×10^{10}	6.5185×10^{12}
$Cond(\mathbf{A_2})$	4.7661×10^5	4.7537×10^8	4.9315×10^{11}	5.2202×10^{14}

可以看出，当 n 增加时，Hilbert 矩阵的条件数相比 Vandermonde 矩阵增长更快，当 $n = 11$ 时已经相差近 100 倍。

对矩阵实施扰动，得到的解如下

表 3: 扰动 $\epsilon = 10^{-10}$ 时的解

$\mathbf{A_1 = b_1}$	$n=5$	扰动 A	1.0000	1.0000	1.0000	1.0000	1.0000						
		扰动 b	1.0000	1.0000	1.0000	1.0000	1.0000						
	$n=7$	扰动 A	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000				
		扰动 b	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000				
	$n=9$	扰动 A	1.0000	1.0000	1.0000	1.0001	0.9999	1.0000	1.0000	1.0000	1.0000		
		扰动 b	1.0000	1.0000	1.0000	0.9999	1.0001	1.0000	1.0000	1.0000	1.0000		
	$n=11$	扰动 A	1.0000	1.0000	0.9999	1.0003	0.9997	1.0003	0.9998	1.0001	1.0000	1.0000	1.0000
		扰动 b	1.0000	0.9999	1.0003	0.9995	1.0006	0.9995	1.0003	0.9999	1.0000	1.0000	1.0000
$\mathbf{A_2 x = b_2}$	$n=5$	扰动 A	1.0000	1.0000	1.0000	1.0000	1.0000						
		扰动 b	1.0000	1.0000	1.0000	1.0000	1.0000						
	$n=7$	扰动 A	1.0000	1.0001	0.9995	1.0020	0.9962	1.0033	0.9989				
		扰动 b	1.0000	0.9999	1.0005	0.9980	1.0038	0.9967	1.0011				
	$n=9$	扰动 A	1.0000	1.0012	0.9785	1.1577	0.4085	2.2304	-0.4355	1.8789	0.7803		
		扰动 b	1.0000	0.9984	1.0276	0.7978	1.7581	-0.5768	2.8397	-0.1263	1.2816		
	$n=11$	扰动 A	1.0000	1.0006	0.9842	1.1832	-0.1219	5.0390	-7.9759	13.4567	-9.5106	5.9310	0.0138
		扰动 b	1.0004	0.9575	2.1484	-12.2720	82.2967	-291.6847	651.4417	-901.6909	762.6726	-356.3390	72.4698

表 4: 扰动 $\epsilon = 10^{-8}$ 时的解

$\mathbf{A_1 = b_1}$	$n=5$	扰动 A	1.0000	1.0000	1.0000	1.0000	1.0000						
		扰动 b	1.0000	1.0000	1.0000	1.0000	1.0000						
	$n=7$	扰动 A	0.9999	1.0002	0.9995	1.0005	0.9997	1.0001	1.0000				
		扰动 b	1.0001	0.9998	1.0005	0.9995	1.0003	0.9999	1.0000				
	$n=9$	扰动 A	0.9998	1.0015	0.9961	1.0060	0.9944	1.0034	0.9987	1.0003	1.0000		
		扰动 b	1.0002	0.9985	1.0039	0.9940	1.0056	0.9966	1.0013	0.9997	1.0000		
	$n=11$	扰动 A	0.9991	1.0066	0.9787	1.0404	0.9499	1.0424	0.9752	1.0099	0.9974	1.0004	1.0000
		扰动 b	1.0009	0.9933	1.0214	0.9593	1.0505	0.9573	1.0250	0.9900	1.0026	0.9996	1.0000
$\mathbf{A_2 x = b_2}$	$n=5$	扰动 A	1.0000	1.0001	0.9994	1.0009	0.9996						
		扰动 b	1.0000	0.9999	1.0006	0.9991	1.0004						
	$n=7$	扰动 A	0.9999	1.0045	0.9546	1.1816	0.6594	1.2997	0.9001				
		扰动 b	1.0001	0.9950	1.0505	0.7982	1.3784	0.6670	1.1110				
	$n=9$	扰动 A	0.9999	1.0054	0.9055	1.6933	-1.6000	6.4079	-5.3093	4.8628	0.0343		
		扰动 b	1.0022	0.8425	3.7567	-19.2162	76.8106	-156.6860	184.9670	-111.6329	29.1582		
	$n=11$	扰动 A	1.0000	1.0006	0.9839	1.1857	-0.1374	5.0949	-8.1001	13.6291	-9.6560	5.9992	0.0001
		扰动 b	1.0387	-3.2531	115.8461	-1326.2215	8130.7850	-29267.8817	65046.0690	-90269.3222	76169.2854	-35733.3781	7148.0712

表 5: 扰动 $\epsilon = 10^{-6}$ 时的解

$\mathbf{A_1 = b_1}$	$n=5$	扰动 A	0.9993	1.0025	0.9967	1.0019	0.9996						
		扰动 b	1.0007	0.9975	1.0033	0.9981	1.0004						
	$n=7$	扰动 A	0.9950	1.0245	0.9503	1.0536	0.9676	1.0104	0.9986				
		扰动 b	1.0050	0.9755	1.0497	0.9464	1.0324	0.9896	1.0014				
	$n=9$	扰动 A	0.9758	1.1481	0.6062	1.5958	0.4389	1.3367	0.8743	1.0267	0.9975		
		扰动 b	1.0243	0.8516	1.3948	0.4027	1.5624	0.6625	1.1260	0.9732	1.0025		
	$n=11$	扰动 A	0.9079	1.6621	-1.1314	5.0464	-4.0176	5.2467	-1.4845	1.9922	0.7411	1.0398	0.9973
		扰动 b	1.0924	0.3360	3.1373	-3.0577	6.0316	-3.2585	3.4914	0.0050	1.2596	0.9600	1.0028
$\mathbf{A_2 x = b_2}$	$n=5$	扰动 A	0.9994	1.0121	0.9457	1.0845	0.9578						
		扰动 b	1.0006	0.9874	1.0567	0.9118	1.0441						
	$n=7$	扰动 A	0.9990	1.0417	0.5830	2.6679	-2.1273	3.7520	0.0827				
		扰动 b	1.0120	0.4955	6.0450	-19.1802	38.8378	-32.2973	12.0991				
	$n=9$	扰动 A	0.9999	1.0056	0.9021	1.7177	-1.6914	6.5980	-5.5310	4.9986	0.0004		
		扰动 b	1.2188	-14.7528	276.6748	-2020.6156	7582.0584	-15767.6016	18397.7021	-11262.2871	2816.8218		
	$n=11$	扰动 A	1.0000	1.0006	0.9839	1.1857	-0.1376	5.0955	-8.1014	13.6308	-9.6575	5.9999	0.0000
		扰动 b	4.8659	-424.3116	11485.6151	-132721.1736	812979.6095	-2926887.5706	6504508.7718	-9027032.4101	7616830.5350	-3573437.2706	714708.2093

上述解的实际误差与用条件数估计的误差如下（注意扰动 A 时用条件数估计误差时，要求 $||\mathbf{A}^{-1}|| \cdot ||\delta\mathbf{A}|| \leq 1$，部分情况下该条件不满足，则使用 $Cond(\mathbf{A}) \cdot \frac{||\delta\mathbf{A}||}{||\mathbf{A}||}$ 来估计上界，这部分在表中用加粗字体表示）：

表 6: 扰动后解的误差

方程	n	扰动项	$\epsilon = 10^{-10}$		$\epsilon = 10^{-8}$		$\epsilon = 10^{-6}$	
			实际误差	估计误差	实际误差	估计误差	实际误差	估计误差
$\mathbf{A_1x = b_1}$	5	**A**	2.0748E-07	4.2477E-06	2.0749E-05	4.2495E-04	2.0740E-03	4.4361E-02
		b	2.0749E-07	2.0003E-06	2.0749E-05	2.0003E-04	2.0749E-03	2.0003E-02
	7	**A**	3.1950E-06	2.9153E-04	3.1931E-04	3.0020E-02	3.1887E-02	**2.9145E+00**
		b	3.1918E-06	1.3691E-04	3.1931E-04	1.3690E-02	3.1932E-02	1.3690E+00
	9	**A**	3.3382E-05	1.3353E-02	3.3038E-03	**1.3177E+00**	3.2952E-01	**1.3177E+02**
		b	3.2978E-05	6.8070E-03	3.3030E-03	6.8079E-01	3.3034E-01	6.8079E+01
	11	**A**	1.7457E-04	7.8800E-01	2.5527E-02	**4.4052E+01**	2.5548E+00	**4.4052E+03**
		b	3.2208E-04	2.4875E-01	2.5701E-02	2.4864E+01	2.5620E+00	2.4864E+03
$\mathbf{A_2x = b_2}$	5	**A**	5.1182E-06	3.0415E-05	5.1160E-04	3.0507E-03	4.9020E-02	4.3708E-01
		b	5.1182E-06	1.5187E-05	5.1182E-04	1.5187E-03	5.1182E-02	1.5187E-01
	7	**A**	2.1009E-03	2.9465E-02	1.8931E-01	**2.8621E+00**	1.7383E+00	**2.8621E+02**
		b	2.1032E-03	1.2389E-02	2.1032E-01	1.2389E+00	2.1032E+01	1.2389E+02
	9	**A**	7.2805E-01	**2.8574E+01**	3.1999E+00	**2.8574E+03**	3.3124E+00	**2.8574E+05**
		b	9.3304E-01	1.1116E+01	9.3304E+01	1.1116E+03	9.3304E+03	1.1116E+05
	11	**A**	5.9475E+00	**2.9412E+04**	6.0298E+00	**2.9412E+06**	6.0306E+00	**2.9412E+08**
		b	4.3099E+02	1.0505E+04	4.3100E+04	1.0505E+06	4.3100E+06	1.0505E+08

从上述表格可以看出，表中所有有效的误差中，实际误差不超过估计误差，因此使用条件数能够正确地估计误差的上界。

当 n 相同时，Hilbert 矩阵的病态程度要高于 Vandermonde 矩阵（从条件数及解的误差均能看出）。当 $n = 5, 7$ 时，二矩阵病态程度有限，可以看出实际误差与扰动 ϵ 有比较好的线性关系，这与误差的理论上界关系式是吻合的。

$n = 9, 11$ 时，二矩阵均呈现较强的病态性，且 Hilbert 矩阵病态性要远强于 Vandermonde 矩阵，使得其解几乎不可用。此时，扰动 $\mathbf{b}$ 相比扰动 $\mathbf{A}$，对解的影响更大。

2.1.5 结论

1. 使用条件数可以方便地刻画矩阵的病态性，当条件数过大时，矩阵病态性强，微小的扰动对解会带来较大影响；

2. Hilbert 矩阵与 Vandermonde 矩阵在 n 较大时均有较强的病态性，且相同维度的情况下，前者病态性更强；

3. 使用条件数估计出的解在扰动下的误差是实际误差的上界，它可能比实际误差大得多，但在不同扰动下，它的数量级变化与实际误差是相同的，从而可以反映实际误差的变化情况。

2.2 Ch5-P3 迭代法

2.2.1 主要算法

题目要求使用 Jacobi 和 Gauss-Seidel 方法迭代求解线性方程组 $\mathbf{Ax = b}$。这两种算法的主要思路是分解系数矩阵为 $\mathbf{A = D - L - U}$，分别为 $\mathbf{A}$ 的对角、下三角、上三角 (不含对角) 元素。上述两种迭代方法可以写成

$$\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}$$

$$\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{U}\mathbf{x}^{(k)}) + \mathbf{D}^{-1}\mathbf{b}$$

使用迭代法的通用形式 $\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{f}$，上述方法中有

$$\mathbf{B}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}),\ \mathbf{f}_J = \mathbf{D}^{-1}\mathbf{b}$$

$$\mathbf{B}_{G-S} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U},\ \mathbf{f}_{G-S} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$$

注意到题给矩阵 A 是严格对角占优的，因此在上述两种迭代方法中一定收敛。

2.2.2 Matlab 程序

如下，我实现了一个迭代法的通用框架，通过 `get_iter_params` 函数可以方便地增加更多的迭代方法。迭代函数可以设置迭代次数上限，当超出上限而误差没有缩小到指定范围内时，说明算法不收敛。

```
%% construct matrix
n = 20;
A = sparse(1:n, 1:n, 3, n, n);
A1 = sparse(1:n-1, 2:n, -1/2, n, n);
A2 = sparse(1:n-2, 3:n, -1/4, n, n);
A = A + A1 + A2 + A1' + A2';

%% initial vals
b = ones(n, 1);
% b = [1: n];
x0 = zeros(n, 1);
% x0 = ones(n, 1);
% x0 = [1: n]';
err = 1e-7;
[x_1, k_1] = iter_solve(A, b, x0, err, 'jacobi', 1000);
[x_2, k_2] = iter_solve(A, b, x0, err, 'gauss_seidel', 1000);
x_3 = A \ b;


%% common iterative method
function [x, k] = iter_solve(A, b, x0, err, method, max_step)
    x = x0; k = 0;
    [B, f] = get_iter_params(A, b, method);
    while 1
        k = k + 1;
        x_tmp = B * x + f;
        cur_err = norm(x_tmp - x);
        x = x_tmp;
        if cur_err < err
            break;
        elseif k >= max_step
            error('Failed to converge, final err: %f', cur_err)
        end
    end
end
```

```
36
37 %% get parameter for iterative method
38 function [B, f] = get_iter_params(A, b, method)
39     D = diag(diag(A));
40     L = -tril(A, -1);
41     U = -triu(A, 1);
42     switch method
43         case 'jacobi'
44             B = D \ (L + U);
45             f = D \ b;
46         case 'gauss_seidel'
47             B = (D - L) \ U;
48             f = (D - L) \ b;
49         otherwise
50             error('Unsupported iter method: %s', method)
51     end
52 end
```

2.2.3 计算结果及分析

不同情况下的迭代次数 程序中使用 Matlab 的左除运算来验证迭代法程序的正确性，容易得到，在给定误差下，使用迭代法和 Matlab 自带的运算得到的结果是相同的。

分别取 $\mathbf{b}$ 为全 0 向量、全 1 向量、顺序向量 $(1, 2, \cdots, 20)^{\mathrm{T}}$，取初值 $\mathbf{x_0}$ 为全 0 向量、全 1 向量、顺序向量 $(1, 2, \cdots, 20)^{\mathrm{T}}$，在迭代误差设定为 10^{-7} 的情况下，可以得到如下结果

表 7: Jacobi 方法的迭代次数

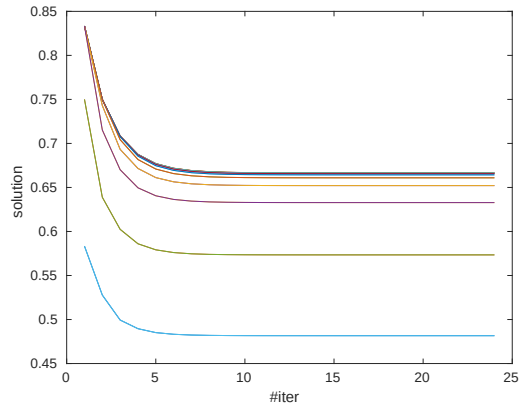
迭代次数	$\mathbf{x_0} = \mathrm{zeros}(20, 1)$	$\mathbf{x_0} = \mathrm{ones}(20, 1)$	$\mathbf{x_0} = [1:20]$	$\mathbf{x_0} = 1000 \times \mathrm{ones}(20, 1)$
$\mathbf{b} = \mathrm{zeros}(20, 1)$	1	25	28	35
$\mathbf{b} = \mathrm{ones}(20, 1)$	25	24	28	35
$\mathbf{b} = [1:20]$	28	28	27	35

表 8: Gauss-Seidel 方法的迭代次数

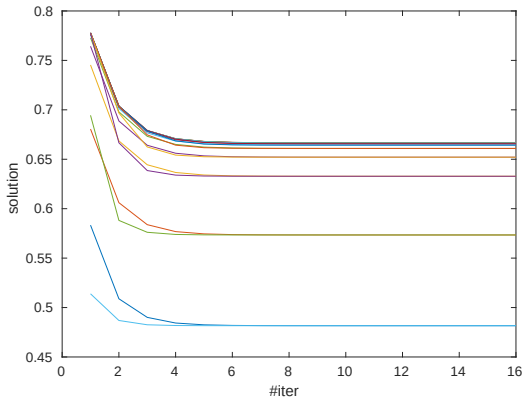
迭代次数	$\mathbf{x_0} = \mathrm{zeros}(20, 1)$	$\mathbf{x_0} = \mathrm{ones}(20, 1)$	$\mathbf{x_0} = [1:20]$	$\mathbf{x_0} = 1000 \times \mathrm{ones}(20, 1)$
$\mathbf{b} = \mathrm{zeros}(20, 1)$	1	16	19	22
$\mathbf{b} = \mathrm{ones}(20, 1)$	16	16	19	22
$\mathbf{b} = [1:20]$	19	18	18	22

收敛过程如下（$\mathbf{x_0} = \mathbf{b} = \mathrm{ones}(20, 1)$ 的情况，其他情况类似，故略去）

图 1: 迭代法求解过程



(a) Jacobi 方法



(b) Gauss-Seidel 方法

很显然，最后的结果是收敛的，这符合理论基础，即对角占优矩阵使用 Jacobi 迭代或 Gauss-Seidel 迭代应均能收敛。并且，从上面的图表可以看出，其他条件相同时，Gauss-Seidel 迭代的收敛速度要快于 Jacobi 迭代；给定初始值距离最终解越近，迭代次数越少。

迭代次数与对角占优的关系 固定迭代误差上限为 10^{-5}，右端向量 $\mathbf{b} = \text{ones}(20, 1)$ 与初始值 $\mathbf{x_0} = \text{zeros}(20, 1)$，改变 $\mathbf{A}$ 矩阵对角线元素，使其为原来的 n 倍，使用 Jacobi 迭代算法，可以得到下列结果

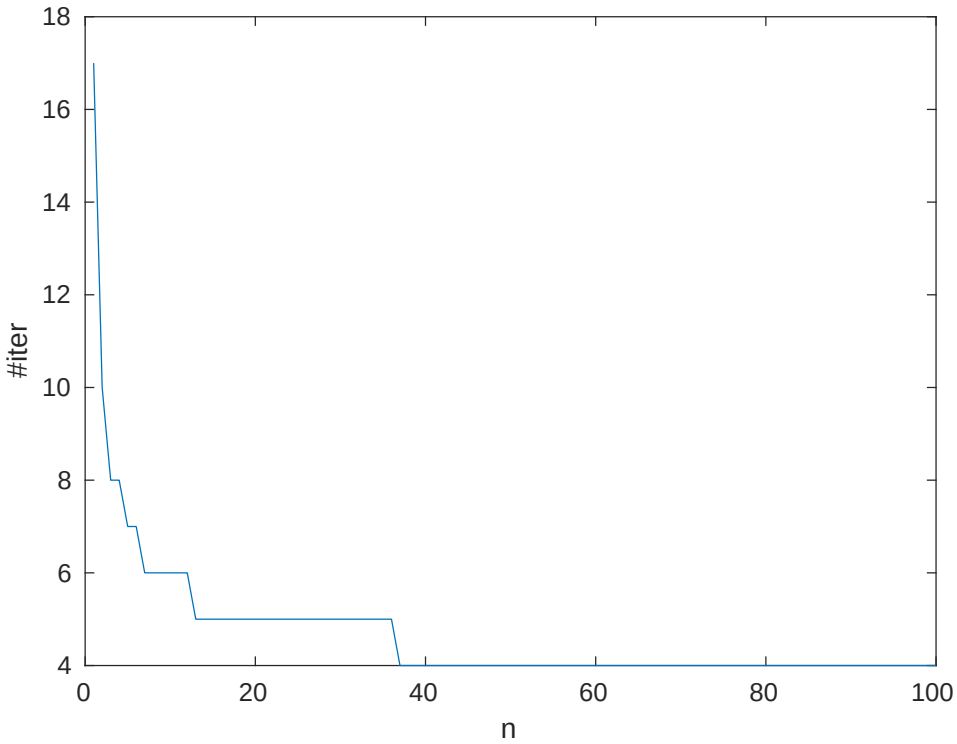


图 2: Jacobi 算法下，迭代次数相对于 n 的变化图线

初始时，矩阵 $\mathbf{A}$ 使用 Jacobi 方法需要 17 次迭代，随着 n 的增大，迭代次数单调减少，在 $n \geq 40$

时，稳定在 4 次。由此可以看出，当系数矩阵 $\mathbf{A}$ 的对角元优势越大时，使用迭代法求解的速度越快，迭代次数越少。当对角元增大到一定程度后，迭代次数趋于一个较小的稳定值。

2.2.4 结论

1. 对角占优矩阵使用 Jacobi 与 Gauss-Seidel 迭代法均能收敛；

2. 相同条件下，Jacobi 方法收敛比 Gauss-Seidel 方法慢；

3. 给定初始值越接近最终解，迭代次数越少；

4. 对角占优矩阵的对角元优势越强，使用迭代法求解速度越快，且迭代次数趋于一个较小的稳定值。

3 应用题

3.1 Ch5-P9 种群

3.1.1 问题分析与模型建立

根据题意，来年年龄为 k 的种群数量 $\widetilde{x}_k$ 与当年的种群数量有如下关系

$$\begin{cases} \widetilde{x}_1 = \sum_{k=1}^{n} b_k x_k \\ \widetilde{x}_{k+1} = s_k x_k - h_k,\ k = 1, 2, \cdots, n-1 \end{cases}$$

其中 h_k 代表收获量，s_k 为一年的自然生存率，b_k 为出生率。记 $\mathbf{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}, \mathbf{h} = (0, h_1, \cdots, h_{n-1})^{\mathrm{T}}$，可以列出如下线性方程组

$$\mathbf{S}\mathbf{x} - \mathbf{h} = \mathbf{x}$$

其中系数矩阵 $\mathbf{S}$ 为

$$\mathbf{S} = \begin{bmatrix} b_1 & b_2 & \cdots & \cdots & b_n \\ s_1 & 0 & \cdots & \cdots & 0 \\ 0 & s_2 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & s_{n-1} & 0 \end{bmatrix}$$

值得注意的是，题目在 $n = 5$ 时给出了 h_5，我认为这是不妥当的，因为加入 h_5 约束后上述方程组可能成为矛盾方程，此时只能求它的最小二乘解。

3.1.2 算法设计

上述方程可以化为

$$(\mathbf{S} - \mathbf{I})\mathbf{x} = \mathbf{h} = \mathbf{A}\mathbf{x}$$

求解时，可以使用 `cond` 计算系数 $\mathbf{A}$ 的条件数，判断问题的病态性。解 $\mathbf{x}$ 可以使用 Matlab 自带的左除直接求得。

3.1.3 Matlab 程序

```
%% input params
b = [0 0 5 3 0];
s = [0.4 0.6 0.6 0.4];
h = [0 500 400 200 100]';

%% construct linear eqn
S = [b; [diag(s) zeros(length(s), 1)]];
A = S - eye(length(b));
fprintf( 'Cond(A)=%f\n ', cond(A));

%% solve eqn
x = A \ h;
err = norm(A * x - h);
```

3.1.4 计算结果与分析

矩阵 $\mathbf{A}$ 的条件数为 87.19，说明问题具有一定的病态性。

当 $h_1 \sim h_5$ 为 500, 400, 200, 100, 100 时，计算可得

$$\mathbf{x} = (8481.0,\ 2892.4,\ 1335.4,\ 601.3,\ 140.5)^{\mathrm{T}} \approx (8481,\ 2892,\ 1335,\ 601,\ 141)^{\mathrm{T}}$$

误差 $||\mathbf{Ax} - \mathbf{h}|| = 1.4 \times 10^{-12}$，注意到 $x_5 > h_5$，说明如果将 h_5 约束加入原方程组确实会产生矛盾，但满足 $x_5 \geq h_5$ 说明最年长种群的数量是满足收获需求的，具有实际意义。

当 $h_1 \sim h5$ 均为 500 时，直接求解可得

$$\mathbf{x} = (10981.0,\ 3892.4,\ 1835.4,\ 601.3,\ -259.4)^{\mathrm{T}}$$

误差 $||\mathbf{Ax} - \mathbf{h}|| = 1.1 \times 10^{-12}$，说明解在数学上是合法的。但此时 $x_5 < 0$，不符合实际意义。因此在题给条件下，是不可能达到 $h_1 \sim h5$ 均为 500 这一目标的。

由于该种群为人工饲养，可以通过人工控制的方法改变生存率（提高生活环境质量）、出生率（人工培育），从而达到最终的收获量目标。例如将自然生存率改为 $\mathbf{s} = (0.4,\ 0.5,\ 0.8,\ 0.8)^{\mathrm{T}}$，可以得到如下解

$$\mathbf{x} = (14687.5,\ 5375.0,\ 2187.5,\ 1250.0,\ 500.0\ )^{\mathrm{T}} \approx (14688,\ 5375,\ 2188,\ 1250,\ 500)^{\mathrm{T}}$$

误差 $||\mathbf{Ax} - \mathbf{h}|| = 5.4 \times 10^{-13}$。此时 $x_5 \geq h_5$，满足最年长种群的收获需求。

从上面也可以看出，对于系数矩阵或右端项的微小扰动，对结果有较大影响，也验证了通过条件数判断的问题的病态性。

3.1.5 结论

题给的种群繁殖模型对出生量、生存率等参数较为敏感，如果要达到题中的高收获量的目标，可以通过提升生存率来实现。

4 收获与建议

通过本次实验，我对线性方程组的求解方法有了更深刻的理解，并通过 Matlab 编程加以实践，巩固了所学知识。一个小建议是修改本章第 9 题种群数量的表述（如是否应该对第 5 年收获量进行约束），增加模型的合理性。