数学实验 · 实验报告3

计算机系　计01　容逸朗　2020010869

1　5-1 误差

1.1　算法设计

范德蒙矩阵可以利用简单的数学计算和矩阵连接的方式生成，希尔伯特矩阵则可以通过 `hilb` 生成。完成矩阵后可以利用 `sum(A1,2)` 计算矩阵 `A1` 的行和。利用 `cond` 命令可以得到矩阵的条件数。

误差上限则可以通过下面的式子估算：

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

1.2　程序

```
format short e;

conn = [];
colA1 = [];
colA2 = [];
colb1 = [];
colb2 = [];
col = [];

for n = 5:2:11

    x = 1: 0.1: 1 + (n - 1) * 0.1;

    % 生成 n 阶范德蒙矩阵
    A1 = [];
    for i = 0 : n - 1
        A1 = [A1 x'.^i];
    end

    % 生成 n 阶希尔伯特矩阵
    A2 = hilb(n);

    b1 = sum(A1, 2);
    b2 = sum(A2, 2);

    % 解方程组
    solx1 = A1 \ b1;
    solx2 = A2 \ b2;

    % 计算范数
    cond1 = cond(A1);
```

```matlab
    cond2 = cond(A2);

    conn = [conn [cond1; cond2]];
    for eps = [1e-10 1e-8 1e-6]

        % 引入临时变量
        tmpA1 = A1;
        tmpA2 = A2;
        tmpb1 = b1;
        tmpb2 = b2;

        % 增加偏移量
        tmpA1(n, n) = tmpA1(n, n) + eps;
        tmpA2(n, n) = tmpA2(n, n) + eps;
        tmpb1(n) = tmpb1(n) + eps;
        tmpb2(n) = tmpb2(n) + eps;

        % 方程组求解
        solA1 = tmpA1 \ b1;
        solA2 = tmpA2 \ b2;
        solb1 = A1 \ tmpb1;
        solb2 = A2 \ tmpb2;

        % 生成数据
        colA1 = [colA1; solA1' zeros(1, 11 - n)];
        colA2 = [colA2; solA2' zeros(1, 11 - n)];
        colb1 = [colb1; solb1' zeros(1, 11 - n)];
        colb2 = [colb2; solb2' zeros(1, 11 - n)];

        % 估计误差
        estA1 = cond1 * norm(tmpA1 - A1) / norm(A1);
        estA2 = cond2 * norm(tmpA2 - A2) / norm(A2);
        estb1 = cond1 * norm(tmpb1 - b1) / norm(b1);
        estb2 = cond2 * norm(tmpb2 - b2) / norm(b2);

        % 实际误差
        difA1 = norm(solx1 - solA1) / norm(solx1);
        difA2 = norm(solx2 - solA2) / norm(solx2);
        difb1 = norm(solx1 - solb1) / norm(solx1);
        difb2 = norm(solx2 - solb2) / norm(solx2);

        col = [col [estA1 difA1 estb1 difb1...
                    estA2 difA2 estb2 difb2]'];

    end

end
```

1.3 计算结果及分析

首先验证函数的正确性:

$A_1x = b_1$	$A_2x = b_2$
1.000002052839417	0.999999993617449
0.999985494114615	1.000000673331311
1.000045845093373	0.999982417027897
0.999914658778163	1.000197695109112
1.000103627269604	0.998816521722876
0.999914229995649	1.004178442734113
1.000049007729694	0.990868799652636
0.999980910330229	1.012488441600492
1.000004851743605	0.989597036111437
0.999999273412817	1.004825308816259
1.000000048692835	0.999044665957601

从上表可见，$n = 11$ 时的结果与理论解（全为 1）有不少的误差，其中 $A_2x = b_2$ 的误差更大。

得到矩阵后，可以计算得到各矩阵的条件数:

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

由此可见，当 n 增大时，两者的条件数不断增大， 而希尔伯特矩阵的条件数上升得更快。

矩阵增加扰动的情况如下:

1. 对 A_1 进行扰动，此时 $A_1x = b_1$ 的解为:

n	**ε**	**sol**										
	1e-10	1.0000	1.0000	1.0000	1.0000	1.0000						
5	1e-8	1.0000	1.0000	1.0000	1.0000	1.0000						
	1e-6	0.9993	1.0025	0.9967	1.0019	0.9996						
	1e-10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000				
7	1e-8	0.9999	1.0002	0.9995	1.0005	0.9997	1.0001	1.0000				
	1e-6	0.9950	1.0245	0.9503	1.0536	0.9676	1.0104	0.9986				
	1e-10	1.0000	1.0000	1.0000	1.0001	0.9999	1.0000	1.0000	1.0000	1.0000		
9	1e-8	0.9998	1.0015	0.9961	1.0060	0.9944	1.0034	0.9987	1.0003	1.0000		
	1e-6	0.9758	1.1481	0.6062	1.5958	0.4389	1.3367	0.8743	1.0267	0.9975		
	1e-10	1.0000	1.0001	0.9998	1.0003	0.9996	1.0004	0.9998	1.0001	1.0000	1.0000	1.0000
11	1e-8	0.9991	1.0066	0.9787	1.0405	0.9498	1.0425	0.9751	1.0099	0.9974	1.0004	1.0000
	1e-6	0.9079	1.6622	-1.1314	5.0465	-4.0177	5.2467	-1.4845	1.9922	0.7411	1.0398	0.9973

2. 对 b_1 进行扰动，此时 $A_1x = b_1$ 的解为:

n	ε	sol										
	1e-10	1.0000	1.0000	1.0000	1.0000	1.0000						
5	1e-8	1.0000	1.0000	1.0000	1.0000	1.0000						
	1e-6	1.0007	0.9975	1.0033	0.9981	1.0004						
	1e-10	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000				
7	1e-8	1.0001	0.9998	1.0005	0.9995	1.0003	0.9999	1.0000				
	1e-6	1.0050	0.9755	1.0497	0.9464	1.0324	0.9896	1.0014				
	1e-10	1.0000	1.0000	1.0000	0.9999	1.0001	1.0000	1.0000	1.0000	1.0000		
9	1e-8	1.0002	0.9985	1.0039	0.9940	1.0056	0.9966	1.0013	0.9997	1.0000		
	1e-6	1.0243	0.8516	1.3948	0.4027	1.5624	0.6625	1.1260	0.9732	1.0025		
	1e-10	1.0000	0.9999	1.0002	0.9995	1.0006	0.9995	1.0003	0.9999	1.0000	1.0000	1.0000
11	1e-8	1.0009	0.9933	1.0214	0.9593	1.0504	0.9573	1.0250	0.9900	1.0026	0.9996	1.0000
	1e-6	1.0924	0.3360	3.1373	-3.0576	6.0315	-3.2584	3.4914	0.0050	1.2596	0.9600	1.0028

3. 对 A_2 进行扰动，此时 $A_2x = b_2$ 的解为：

n	ε	sol										
	1e-10	1.0000	1.0000	1.0000	1.0000	1.0000						
5	1e-8	1.0000	1.0001	0.9994	1.0009	0.9996						
	1e-6	0.9994	1.0121	0.9457	1.0845	0.9578						
	1e-10	1.0000	1.0001	0.9995	1.0020	0.9962	1.0033	0.9989				
7	1e-8	0.9999	1.0045	0.9546	1.1816	0.6594	1.2997	0.9001				
	1e-6	0.9990	1.0417	0.5830	2.6679	-2.1273	3.7520	0.0827				
	1e-10	1.0000	1.0012	0.9785	1.1577	0.4085	2.2304	-0.4355	1.8789	0.7803		
9	1e-8	0.9999	1.0054	0.9055	1.6933	-1.6000	6.4079	-5.3093	4.8628	0.0343		
	1e-6	0.9999	1.0056	0.9021	1.7177	-1.6914	6.5980	-5.5310	4.9986	0.0004		
	1e-10	1.0000	1.0006	0.9842	1.1832	-0.1219	5.0390	-7.9759	13.4567	-9.5106	5.9310	0.0138
11	1e-8	1.0000	1.0006	0.9839	1.1857	-0.1374	5.0949	-8.1001	13.6291	-9.6560	5.9992	0.0001
	1e-6	1.0000	1.0006	0.9839	1.1857	-0.1376	5.0955	-8.1014	13.6308	-9.6575	5.9999	0.0000

4. 对 b_2 进行扰动，此时 $A_2x = b_2$ 的解为：

n	ε	sol										
	1e-10	1	1	1	0.99999	1						
5	1e-8	1	0.99987	1.0006	0.99912	1.0004						
	1e-6	1.0006	0.9874	1.0567	0.9118	1.0441						
	1e-10	1	0.99995	1.0005	0.99798	1.0038	0.99667	1.0011				
7	1e-8	1.0001	0.99495	1.0505	0.7982	1.3784	0.66703	1.111				
	1e-6	1.012	0.4955	6.045	-19.18	38.838	-32.297	12.099				
	1e-10	1	0.99842	1.0276	0.79784	1.7581	-0.57685	2.8397	-0.12632	1.2816		
9	1e-8	1.0022	0.84247	3.7567	-19.216	76.811	-156.69	184.97	-111.63	29.158		
	1e-6	1.2188	-14.753	276.67	-2020.6	7582.1	-15768	18398	-11262	2816.8		
	1e-10	1.0004	0.95747	2.1484	-12.272	82.297	-291.68	651.44	-901.69	762.67	-356.34	72.47
11	1e-8	1.0387	-3.2531	115.85	-1326.2	8130.8	-29268	65046	-90269	76169	-35733	7148.1
	1e-6	4.8659	-424.31	11486	-132721	812979	-2926887	6504508	-9027032	7616830	-3573437	714708

误差为:

干扰项		A1		b1		A2		b2	
n	ε	估计误差	实际误差	估计误差	实际误差	估计误差	实际误差	估计误差	实际误差
	1e-10	4.2477e-06	2.0749e-07	2.0003e-06	2.0748e-07	3.0414e-05	5.1182e-06	1.5187e-05	5.1182e-06
5	1e-8	4.2477e-04	2.0749e-05	2.0003e-04	2.0749e-05	3.0414e-03	5.1160e-04	1.5187e-03	5.1182e-04
	1e-6	4.2477e-02	2.0740e-03	2.0003e-02	2.0749e-03	3.0414e-01	4.9020e-02	1.5187e-01	5.1182e-02
	1e-10	2.9144e-04	3.1930e-06	1.3691e-04	3.1919e-06	2.8621e-02	2.1009e-03	1.2389e-02	2.1032e-03
7	1e-8	2.9145e-02	3.1931e-04	1.3690e-02	3.1932e-04	2.8621e+00	1.8931e-01	1.2389e+00	2.1032e-01
	1e-6	2.9145e+00	3.1887e-02	1.3690e+00	3.1932e-02	2.8621e+02	1.7383e+00	1.2389e+02	2.1032e+01
	1e-10	1.3177e-02	3.3050e-05	6.8070e-03	3.3042e-05	2.8574e+01	7.2804e-01	1.1116e+01	9.3304e-01
9	1e-8	1.3177e+00	3.3032e-03	6.8079e-01	3.3031e-03	2.8574e+03	3.1999e+00	1.1116e+03	9.3304e+01
	1e-6	1.3177e+02	3.2952e-01	6.8079e+01	3.3034e-01	2.8574e+05	3.3124e+00	1.1116e+05	9.3304e+03
	1e-10	4.4072e-01	2.6474e-04	2.4875e-01	2.3857e-04	2.9412e+04	5.9415e+00	1.0505e+04	4.3099e+02
11	1e-8	4.4052e+01	2.5618e-02	2.4864e+01	2.5614e-02	2.9412e+06	6.0237e+00	1.0505e+06	4.3099e+04
	1e-6	4.4052e+03	2.5549e+00	2.4864e+03	2.5619e+00	2.9412e+08	6.0246e+00	1.0505e+08	4.3099e+06

从表中可见:

- 算法的实际误差均不超过估计误差，因此我们可以利用条件数来估算误差的上界;
- 在 n 相同的情况下，希尔伯特矩阵（A_2）解的误差比范德蒙矩阵（A_1）大，越大的 n 会导致更大的误差;
- 当 n 较小时（$n=5,7$），两个矩阵的病态程度仍可接受。当 n 较大（$n=9,11$），两个矩阵的病态程度已经不能被接受，其解也是完全没有实际意义的;
- 除此之外，解受 b 扰动的影响比 A 扰动的影响为大。

1.4 结论

- 条件数越大，矩阵的病态性越强，这时微小的扰动也会导致极大的误差；
- 在 n 相同的情况下，希尔伯特矩阵的病态程度较范德蒙矩阵高；
- 估计的误差往往会比实际误差值为大，但是估计误差的数量级变化和实际误差的量级变化是相若的。

2 5-3 迭代法

2.1 算法设计

利用雅可比迭代和高斯-塞德尔迭代法解方程，其迭代形式分别为：

$$x^{(k+1)} = B_J x^{(k)} + f_J \quad B_J x^{(k)} = D^{-1}(L+U) \quad f_J = D^{-1} b x^{(k+1)} = B_{G-S} x^{(k)} + f_{G-S} \quad B_{G-S} = (D-L)^{-1}U \quad f_{G-S} = (D-L)^{-1}b$$

又因为矩阵 A 是对角估优的，理论上雅可比迭代和高斯-塞德尔迭代法解方程最终都会收敛。

首先利用 `sparse` 生成稀疏矩阵，然后可以用 `full` 填上剩余的空位（也可不做）。接着利用 `triu` 、 `tril` 和 `diag` 计算矩阵的上下三角阵和对角阵。最后可以利用 `zeros` 和 `ones` 函数生成各类参数。

2.2 程序

```
format short;

n = 20;
eps = 1e-10;

% Task1
% 生成矩阵 A
a1 = sparse(1:n, 1:n, 3, n, n);
a2 = sparse(1:n-1, 2:n, -1/2, n, n);
a3 = sparse(1:n-2, 3:n, -1/4, n, n);

a = a1 + a2 + a2' + a3 + a3';
A = full(a);

% 计算 A 的上三角、下三角和对角阵
U = -triu(A, 1);
L = -tril(A, -1);
D = diag(diag(A));

for b = [sum(A, 2) zeros(n, 1) ones(n, 1)]
    for x = [ones(n, 1) zeros(n, 1) eig(A)]

        % 计算雅可比迭代所需参数
        B1 = D \ (L + U);
        f1 = D \ b;

        % 计算高斯-塞德尔迭代所需参数
        B2 = (D - L) \ U;
        f2 = (D - L) \ b;

        x1 = x;
```

```
        x2 = x;

        cnt1 = 0;
        cnt2 = 0;
        i = 0;

        while cnt1 * cnt2 == 0
            i = i + 1;

            if cnt1 == 0
                tmpx1 = B1 * x1 + f1;
                if (norm(tmpx1 - x1) < eps)
                    cnt1 = i;
                end
                x1 = tmpx1;
            end

            if cnt2 == 0
                tmpx2 = B2 * x2 + f2;
                if (norm(tmpx2 - x2) < eps)
                    cnt2 = i;
                end
                x2 = tmpx2;
            end

        end

    end
end

% Task2
eps = 1e-5;
ans = [];

% 计算 A 的上三角、下三角阵
U = -triu(A, 1);
L = -tril(A, -1);

for k = 0.5:0.01:50
    % 计算 A 的对角阵
    D = diag(diag(A) * k);

    % 重置初值
    b = zeros(n, 1);
    x = ones(n, 1);

    % 计算雅可比迭代所需参数
    B = D \ (L + U);
    f = D \ b;
```

```
    cnt = 0;

    while 1
        cnt = cnt + 1;
        tmpx = B * x + f;
        if (abs(tmpx - x) < eps)
            break
        end
        x = tmpx;
    end
    ans = [ans cnt];
end

hold on
axis([0, 50, 0, 50]);
plot([0.5:0.01:50], ans);
xlabel("对角线元素乘数");
ylabel("迭代次数");
hold off
```

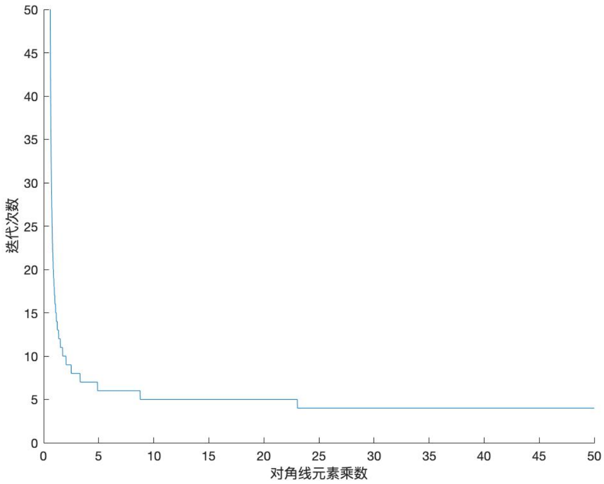
2.3 计算结果与分析

分别取 b 为矩阵 A 的行和、 20 维零向量、 20 维全为 1 向量，再取初值 x 为 20 维的零向量、 20 维全为 1 向量和由 A 的特征根组成的向量。若误差为 $\epsilon = 10^{-10}$ ，有：

	方法	$x = ones(n,1)$	$x = zeros(n,1)$	$x = eig(A)$
$b = sum(A,2)$	Jacobi	1	35	36
	Gauss-Seidel	1	22	23
$b = zeros(n,1)$	Jacobi	35	1	36
	Gauss-Seidel	22	1	23
$b = ones(n,1)$	Jacobi	33	34	36
	Gauss-Seidel	21	21	23

从数据可知，两种方法最终都会收敛，而雅可比迭代法的收敛速度比高斯-塞德尔迭代法要慢。初始值和最终解越接近，迭代次数越少。

取 b 为 20 维的零向量，x 为 20 维全为 1 向量，误差为 $\epsilon = 10^{-5}$ ，并将矩阵 A 的对角线乘上 k 倍（下图横坐标）得到：



从表中可知，对这个乘数越大时，所需的迭代次数越少，大概到 $k = 23$ 的位置时，所需的迭代次数降为 4 。从中我们可知，当一个矩阵的对角优势越大，迭代求解的次数越少、越快捷。

2.4 结论

1. 对角占优矩阵可以利用迭代法求解。
2. 相同条件下（且满足收敛条件时）高斯-塞德尔迭代法的收敛速度比雅可比迭代法要快。
3. 初始值和最终解越接近，求解迭代次数越少。
4. 一个矩阵的对角优势越大，迭代求解的次数越少。

3 5-9 种群

3.1 问题分析与模型建立

由题意知，来年的种群数量应有如下关系：

$$\widetilde{x}_1 = \sum_{k=1}^{n} b_k x_k \widetilde{x}_{k+1} = s_k x_k - h_k, \quad k = 1, 2 \ldots n-1$$

将上式整合得；

$$\widetilde{x} = Ax - h \tag{1}$$

其中 A 是 $n \times n$ 维矩阵，$\widetilde{x}, x, h$ 都是 n 维向量。（如下所示）

$$A = \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_n \\ s_1 & 0 & \ldots & \ldots & 0 \\ 0 & s_2 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \ldots & s_{n-1} & 0 \end{bmatrix}, \quad \widetilde{x} = \begin{bmatrix} \widetilde{x}_1 \\ \widetilde{x}_2 \\ \widetilde{x}_3 \\ \vdots \\ \widetilde{x}_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \quad h = \begin{bmatrix} 0 \\ h_1 \\ h_2 \\ \vdots \\ h_{n-1} \end{bmatrix} \tag{2}$$

要使种群数量不变，应当有 $\widetilde{x} = x$ ，代入式 (1) 有：

$$(A - I)x = h \tag{3}$$

3.2 算法设计

利用 Matlab 的 `diag` 函数生成对角阵，再利用左除的方式求解方程即可。

3.3 程序

```matlab
format short g

% 参数
b = [0 0 5 3 0];
s = [0.4 0.6 0.6 0.4];
h = [0; 500; 400; 200; 100];

% 计算解
A = [b; [diag(s) zeros(4, 1)]];
I = diag(ones(5,1));
x = (A - I) \ h
```

3.4 计算结果与分析

当 $n=5$ ，繁殖率 $b_1=b_2=b_5=0, b_3=5, b_4=3$，自然存活率 $s_1=s_4=0.4, s_2=s_3=0.6$ 以及收获量 $h_1=500, h_2=400,$ $h_3=200, h_4=100, h_5=100$ 时，计算得：

参数	x_1	x_2	x_3	x_4	x_5
值	8481	2892.4	1335.4	601.27	140.51

需要注意的是，(2) 式中的 h 首项为零，末项为 h_{n-1} ，也就是说题目给出的 h_5 是多余的。不过由于计算所得的 $x_5 \geq h_5$ ，因此数据仍然具有实际意义。（即我们也会收获 5 岁的此种群）

保持 b, s 不变，将 h 变为 $h_i=500, i=1,2,\ldots,5$ ，计算得：

参数	x_1	x_2	x_3	x_4	x_5
值	10981	3892.4	1835.4	601.27	-259.49

这里的 x_5 为负数，显然不符合实际意义，也就是说根据题目给出的条件，并不能达到题目所需的条件。

此时我们可以令 $s_1=0.28$ ，其余数字不变，可得：

参数	x_1	x_2	x_3	x_4	x_5
值	48736	13146	7387.6	3932.6	1073

通过降低年轻种群的生存率，使得维持物种的数量需要保有更多各年龄的种群，更多的种群数才可以满足题目所需的收获量。

3.5 结论

题目所给的模型对各项数据都十分敏感，一但种群的生存环境改善（存活率提高）后，由于年轻的种群数量增大，会导致维持种群所需的老年种群数量降低。一种解决方案是降低年轻种群的存活率，但这在实际生产中意义不大；与之相对的是提升存活率和繁殖率，这时可以加大收获量来维持种群的数量平衡。