数学实验实验报告

ZhaohengLi 2017050025
cainetatum@foxmail.com
15801206130

2020 年 3 月 24 日

1 实验目的

- 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;
- 通过实例学习用线性代数方程组解决简化的实际问题。

2 CH5-T1 误差

2.1 (1)

在构造 Vandermonde 矩阵时，可以先构造初始行向量 v，再使用 $fliplr(vander(v))$ 函数获得所需矩阵，具体代码如下：

```
%% Functions
function result = generateA1(n)
v = zeros(1,n);
for i = 0:n-1
    v(1,i+1)=1+0.1*i;
end
result = fliplr(vander(v));
end
```

由于 $b1, b2$ 是通过求 $A1$ $A2$ 行和所得到的，因此可以预先知道方程组 $A_1x = b_1, A_2x = b_2$ 的解均为 $[1, 1, 1, 1, 1]^T$。在 MATLAB 中使用左除命令求得两方程的解也得到了相同的结果。

实验的完整代码如下：

```
%% Global Variables
n = 5;
A1 = generateA1(n);
A2 = hilb(n);
b1 = cal(A1);
b2 = cal(A2);

%% Cal
x1 = A1\b1;
x2 = A2\b2;

%% Functions
function result = generateA1(n)
v = zeros(1,n);
for i = 0:n-1
    v(1,i+1)=1+0.1*i;
end
result = fliplr(vander(v));
end

function b = cal(A)
[m,n]=size(A);
b=zeros(m,1);
for i = 1:m
    for j = 1:n
        b(i,1)=b(i,1)+A(i,j);
    end
end
end
```

2.2 (2)

令 $n = 5, 7, 9, 11$ 计算 $A1$ 和 $A2$ 的条件数结果如下：

	n=5	n=7	n=9	n=11
cond A1	3.5740e+05	8.7385e+07	2.2739e+10	6.5185e+12
cond A2	4.7661e+05	4.7537e+08	4.9315e+11	5.2202e+14

可以看出，n 越大，两个矩阵的条件数越大，且 Hilbert 矩阵的增长速度明显大于 Vandermonde 矩阵。

这两个矩阵的条件数都远远大于 1，因此这两矩阵是病态的。

依题目要求，在不同 n 值的情况下，加入不同的扰动，计算出的结果统计如下表所示：

n=5 A 扰动	1e-10	1e-8	1e-6
x1	0.999999928497779 1.00000025109076 0.999999670407408 1.00000019167177 0.999999958332282	0.999992850027616 1.00002510823598 0.999967041794948 1.00001916659179 0.999995833349670	0.999285297790269 1.00250978759627 0.996705539348900 1.00191586839354 0.999583506871030
x2	0.999999937000235 1.00000125999511 0.999994330022462 1.00000881996460 0.999995590017859	0.999993702777033 1.00012594445916 0.999433249934216 1.00088161121299 0.999559194393668	0.999396609520115 1.01206780959752 0.945694856811606 1.08447466718149 0.957762666409413

n=5 b 扰动	1e-10	1e-8	1e-6
x1	1.00000007149698 0.999999748926741 1.00000032957080 0.999999808340230 1.00000004166525	1.00000714999837 0.999974891672003 1.00003295832681 0.999980833336847 1.00000416666596	1.00071499999482 0.997489166684004 1.00329583331167 0.998083333345308 1.00041666666420
x2	1.00000006299996 0.999998740000632 1.00000566999761 0.999991180003263 1.00000440999853	1.00000629999999 0.999874000000103 1.00056699999999 0.999117999999560 1.00044100000038	1.00062999999997 0.987400000000360 1.05669999999884 0.911800000001346 1.04409999999949

n=7 A 扰动	1e-10	1e-8	1e-6
x1	0.999999499473406	0.999949950645062	0.995001941707229
	1.00000244888018	1.00024487184613	1.02445353695199
	0.999995027181337	0.999502750846092	0.950343411528071
	1.00000536486440	1.00053645142822	1.05357142900001
	0.999996756777021	0.999675698619076	0.967614424153913
	1.00000104171942	1.00010416532538	1.01040221922150
	0.999999861104238	0.999986111290040	0.998613037437293
x2	0.999998800124475	0.999891880293464	0.999007197895274
	1.00005039475708	1.00454102765866	1.04169768837525
	0.999496052533493	0.954589723524054	0.583023116411334
	1.00201578955801	1.18164110557645	2.66790753386753
	0.996220395023040	0.659422927516681	-2.12732662529561
	1.00332605206947	1.29970782345517	3.75204742976538
	0.998891316060928	0.900097392271126	0.0826508568793309

n=7 b 扰动	1e-10	1e-8	1e-6
x1	1.00000050048887	1.00005004995820	1.00500499986614
	0.999997551298745	0.999755125206632	0.975512500626693
	1.00000497246815	1.00049725513320	1.04972555434088
	0.999994635498678	0.999463542123549	0.946354167914376
	1.00000324301318	1.00032430527964	1.03243055483908
	0.999998958344739	0.999895833421559	0.989583333551454
	1.00000013888764	1.00001388887722	1.00138888886138
x2	1.00000120119277	1.00012011999382	1.01201200004809
	0.999949549888833	0.994954960245747	0.495495998065503
	1.00050450121579	1.05045039763952	6.04504001872822
	0.997981994829239	0.798198409155802	-19.1801600730510
	1.00378376013885	1.37837798324516	38.8378001342445
	0.996670290768006	0.667027374456577	-32.2972641162102
	1.00110990316130	1.11099087525906	12.0990880382106

n=9 A 扰动	1e-10	1e-8	1e-6
x1	0.999997542501161	0.999756867022327	0.975750116090522
	1.00001500086016	1.00148452538750	1.14806597244386
	0.999960116194742	0.996051868814822	0.606213262215019
	1.00006033074736	1.00597391995029	1.59584159509339
	0.999943208746394	0.994374908415262	0.438948214712039
	1.00003406796918	1.00337538349365	1.33666535155499
	0.999987280938215	0.998739445840160	0.874270042098089
	1.00000270216756	1.00026788514322	1.02671947079277
	0.999999749875228	0.999975195932771	0.997525974999321
x2	0.999982928009930	0.999924964656596	0.999922327471624
	1.00122918254710	1.00540254400284	1.00559242132158
	0.978489314804997	0.905455488703753	0.902132635597343
	1.15774497254363	1.69332636921616	1.71769395890258
	0.408456506662986	-1.59997375048907	-1.69135221270623
	2.23041020859133	6.40794518201983	6.59801238518553
	-0.435478329497796	-5.30926917336949	-5.53101424561936
	1.87886415669299	4.86281775783593	4.99858004802688
	0.780283987719148	0.0342955821333097	0.000355009344432205

n=9 b 扰动	1e-10	1e-8	1e-6
x1	1.00000242855558	1.00024307303559	1.02430997323200
	0.999985171921619	0.998515830428008	0.851567123916966
	1.00003943507414	1.00394721257429	1.39476255212680
	0.999940331486341	0.994027427304686	0.402681860539590
	1.00005618380383	1.00562386382179	1.56244214890294
	0.999966286685977	0.996625328449687	0.662500326264429
	1.00001259034795	1.00126029761427	1.12604154838692
	0.999997324383267	0.999732167383157	0.973214310085535
	1.00000024774129	1.00002479938853	1.00248015654482
x2	1.00002187876656	1.00218789539489	1.21878955730390
	0.998424728059455	0.842471530232240	-14.7528481860260
	1.02756726875922	3.75674825411853	276.674845627009
	0.797839973342318	-19.2161541159797	-2020.61555451928
	1.75810026685910	76.8105788353668	7582.05840370281
	-0.576848838523256	-156.686005701666	-15767.6016254906
	2.83965725338261	184.967008479288	18397.7020534570
	-0.126320909682138	-111.632863345846	-11262.2870594441
	1.28158025787454	29.1582160649736	2816.82178489535

在以上表格中可以看出：

- 在 n（条件数）固定的前提下，对 A, b 加入的扰动越大，计算的数值结果与正确结果偏差越大。
- 在对 A, b 加入的扰动固定的情况下，n（条件数）越大，计算的数值结果与正确结果偏差越大。

2.3 (3)

在 MATLAB 中计算 $\frac{||x-\tilde{x}||}{||x||}$ 统计数据如下：

n=5 A 扰动	1e-10	1e-8	1e-6
deltax1	2.07492014778727e-07	2.07485330326969e-05	0.00207399721487008
deltax2	5.11820124651315e-06	0.000511596562207495	0.0490204170311345

n=5 b 扰动	1e-10	1e-8	1e-6
deltax1	2.07478178099712e-07	2.07486095898347e-05	0.00207486136047175
deltax2	5.11821980291673e-06	0.000511822174406925	0.0511822174188975

n=7 A 扰动	1e-10	1e-8	1e-6
deltax1	3.19332346182143e-06	0.000319311471720422	0.0318872706189846
deltax2	0.00210092251471574	0.189312391051443	1.73834861180689

n=7 b 扰动	1e-10	1e-8	1e-6
deltax1	3.19310441910646e-06	0.000319315310988476	0.0319315574834335
deltax2	0.00210323228755759	0.210324318783057	21.0324328796005

n=9 A 扰动	1e-10	1e-8	1e-6
deltax1	3.33609784027688e-05	0.00330377952328433	0.329521121951531
deltax2	0.728045789880671	3.19993450387683	3.31239914567981

n=9 b 扰动	1e-10	1e-8	1e-6
deltax1	3.29986041841976e-05	0.00330304417304634	0.330337709810490
deltax2	0.933037121444716	93.3043674528254	9330.43736122720

在 MATLAB 中使用如下代码验证，所有数据均满足

$$\frac{||\delta x||}{||x||} \approx cond(A) * \frac{||\delta A||}{||A||}$$

即 A 的条件数越大,(由 A 的扰动引起的)x 的误差越大。

$$\frac{||\delta x||}{||x||} \leq cond(A) * \frac{||\delta b||}{||b||}$$

即 A 的条件数越大,(由 b 的扰动引起的)x 的误差可能越大。

总体来看，x 的 (相对) 误差不超过 b 的 (相对) 误差的 Cond(A) 倍, 也大致上是 A 的 (相对) 误差的 Cond(A) 倍，因此我们可以通过条件数来更加简便的判断计算所得的数值结果的误差范围数量级。

```
%% Global Variables
n = 5;
epsilon = 1e-10;
disturbA = 0; % 为1则A扰动 为0则b扰动

%% Cal

originalA1 = generateA1(n);
originalA2 = hilb(n);

originalb1 = cal(originalA1);
originalb2 = cal(originalA2);

A1 = originalA1;
A2 = originalA2;
b1 = originalb1;
b2 = originalb2;

if disturbA
    A1(n,n) = A1(n,n) + epsilon;
    A2(n,n) = A2(n,n) + epsilon;
else
    b1(n,1) = b1(n,1) + epsilon;
    b2(n,1) = b2(n,1) + epsilon;
end


x1 = A1\b1;
x2 = A2\b2;

originalx = ones(n,1);

condA1 = cond(originalA1)
ΔA1 = norm(originalA1-A1)/norm(originalA1)
Δb1 = norm(originalb1-b1)/norm(originalb1)
Δx1 = norm(originalx-x1)/norm(originalx)

condA2 = cond(originalA2)
ΔA2 = norm(originalA2-A2)/norm(originalA2)
Δb2 = norm(originalb2-b2)/norm(originalb2)
Δx2 = norm(originalx-x2)/norm(originalx)

if disturbA
    check1 = condA1*ΔA1/Δx1
    check2 = condA2*ΔA2/Δx2
else
    check1 = condA1*Δb1/Δx1
    check2 = condA2*Δb2/Δx2
end
```

3 CH5-T3 迭代法

3.1 (1)

通过随机生成右端向量 b 以及初始向量 $x^{(0)}$, 并设定不同的迭代误差要求统计得到如下数据:

迭代方法	随机生成 x0 和 b 的范围	精度要求	迭代步数
Jacob	[0,1]	1e-10	32
GaussSeidel			21
Jacob	[100,1000]	1e-10	42
GaussSeidel			26
Jacob	[0,1]	1e-100	53
GaussSeidel			33
Jacob	[100,1000]	1e-100	55
GaussSeidel			32
Jacob	[-5000,5000]	1e-100	54
GaussSeidel			33

可以看出，两种方法的迭代步数与误差要求程正相关，数据精度要求越高，需要迭代的步数越大，也可以通过对比看出，迭代步数与右端向量 b 以及初始向量 $x^{(0)}$ 的数据范围没有明显关系。

同时，可以看到 GaussSeidel 方法使用的迭代步数要小于 Jacob 方法的，这是因为在 Jacob 的每一步迭代中，采用的全部是上一步的数据，而在 GaussSeidel 方法中，部分数据采用了当前步生成的数据，更加有效地利用了数据，避免了“弯路”，所以 GaussSeidel 方法使用的迭代步数要小于 Jacob 方法。

MATLAB 代码如下:

```
%% Init
clc;
clear;

%% Global Variables
n = 20; % 矩阵阶数
range = [-5000, 5000]; % 随机数范围
epsilon = 1e-100; % 误差要求
M = 200; % 迭代最大步数

%% Cal
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+a3+a2'+a3';

b = range(1) + (range(2)-range(1)).*rand(n,1);
x0 = range(1) + (range(2)-range(1)).*rand(n,1);

[xJ, stepJ, sequenceJ] = Jacobi(A,b,x0,epsilon,M);
[xG, stepG, sequenceG] = GaussSeidel(A,b,x0,epsilon,M);
```

3.2 (2)

MATLAB 代码实现如下：

```
%% Init
clc;
clear;

%% Global Variables
n = 20; % 矩阵阶数
range = [0, 100]; % 随机数范围
epsilon = 1e-5; % 误差要求
M = 200; % 迭代最大步数
cycle = 15;
%% Cal
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);

b = range(1) + (range(2)-range(1)).*rand(n,1);
x0 = range(1) + (range(2)-range(1)).*rand(n,1);

step = zeros(cycle,1);
normB = zeros(cycle, 1);

for i = 1:cycle
A = 2^(i-1).*a1+a2+a3+a2'+a3';
[~, step(i,1), ~,normB(i,1)] = Jacobi(A,b,x0,epsilon,M);
end
```

按照题目要求进行了 15 轮求解，使用 Jacob 方法每轮 step 的数据如下：

I	step	||B||
1	24	0.489306191807942
2	13	0.244653095903971
3	10	0.122326547951986
4	8	0.0611632739759928
5	6	0.0305816369879964
6	6	0.0152908184939982
7	5	0.00764540924699909
8	5	0.00382270462349955
9	4	0.00191135231174977
10	4	0.000955676155874887
11	4	0.000477838077937443
12	4	0.000238919038968722
13	3	0.000119459519484361
14	3	5.97297597421804e-05
15	3	2.98648798710902e-05

可以看出，随着主对角线元素不断增大，达到某一给定的误差精度所需要的迭代步数会不断减少，收敛速度不断增加。

在迭代法中，如果 $||B|| = q < 1$，则迭代公式 $x^{(k+1)} = Bx^{(k)} + f$ 收敛，且 $||x^{(k+1)} - x^*|| \leq \frac{q}{1-q}||x^{(k+1)} - x^{(k)}||$, 即 q 越小，收敛越快。

在本题中，当主对角元素不断增大的时候，使用 Jacob 方法计算出来的 B 矩阵的模如表中所示，可以看出，$||B||$ 在不断减小，因此使得收敛速度不断增加。

4 CH5-T9 种群

4.1 (1)

根据题目意义。如果需要稳定收获，即 $\widetilde{x}_k = x_k$, 则针对 $x_k(k = 1, 2, ..., n)$ 可列出如下线性方程组：

$$\sum_{k=1}^{n} b_k x_k = x_1$$

$$s_k x_k - h_k = x_{k+1}, \quad k = 1, 2..., n-1$$

设：

$$x = (x_1, x_2, ..., x_n)^T$$

$$h = (0, h_1, h_2, ..., h_{n-1})^T$$

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

因此，上述线性方程组模型即可化为所求个年龄的稳定种群数量模型:

$$Sx - h = x$$

4.2 (2)

令 $A = S - I$，则模型可以表示为 $Ax = h$，通过计算得到矩阵 A 的条件数为 87，大于 1，因此稍显病态，说明矩阵 A 以及向量 h 的微小扰动会给 x 的计算结果带来较大的变化。

当 $h_1, \cdots, h_5 = 500, 400, 200, 100, 100$ 时，计算可得到：

$$x = (8481.0, 2892.4, 1335.4, 601.3, 140.5)^T$$

误差为 $-3.7e-13$，说明结果较为准确。

4.3 (3)

当 $h_1, \cdots, h_5 = 500$ 时，计算可得到：

$$x = (11772.1, 4208.8, 2025.3, 715.1, -213.9)^T$$

误差为 $8.5e-13$，说明结果较为准确。

注意到 $x_5 = -213.9 < 0$, 在实际情况中不可能使得种群数量为负数。因此题目中要求的 $h_1, \cdots, h_5 = 500$ 在当前参数设定下不可能达到。

通过人工控制的方式改变物种的死亡率 (例如提高生活环境质量) 可以达到预期的收获率: 假设我们将 s_4 提高为 0.9，可以得到：

$$x = (11772.1, 4208.8, 2025.3, 715.1, 143.6)^T$$

因此，当提高自然存活率 $s_4 = 0.9$，并且种群数量设定为上述数值后，可以使得 $h_1, \cdots, h_5 = 500$。

4.4 MATLAB 代码

```
%% Global Variables
b = [0 0 5 3 0];
s = [0.4 0.6 0.6 0.9];
% h = [0; 500; 400; 200; 100];
h = [500; 500; 500; 500; 500];
%% Cal
S = [b; [diag(s) zeros(length(s), 1)]];
A = S - eye(length(b));
fprintf('Cond(A) = %f\n', cond(A));
x = A\h
err = sum(sum(A * x - h))
```

5 收获和建议

通过这次的实验，我对 MATLAB 中提供线性方程组求解函数理解更加深刻，通过实际编程、画图的方式观察了方程求解的结果，这是书本上无法学到的知识。同时，在做上机实验的过程中，我对 MATLAB 这款软件的使用也更加熟练了。希望在之后的课堂上老师能够当堂进行相关的技巧演示并给出题目的分步解答。