作业一

姓名：马宇骁　学号：**PB19151769**　日期：**2021.5.4**

第一题 本题考虑使用有限差分方法 **(finite difference method)** 解决两点边值问题 **(boundaryvalue problem)**

MATLAB 程序显示如下：

```
clear,clc
% 第一题
A = diag(repmat([2],1,10))+diag(repmat([-1],1,9),1)
+diag(repmat([-1],1,9),-1);
b = [2 -2 2 -1 0 0 1 -2 2 -2];
b = b';
xexact = [1 0 1 0 0 0 0 -1 0 -1];
xexact = xexact';
esp = 1e-15;

% 1
function jacobi(A,b,xexact,esp)
% 其中，A 为线性方程组的系数矩阵，b 为常数项，eps为精度要求
x1 = zeros(10,1);
i = 0;
D = diag(diag(A));
while norm(x1 - xexact,inf) > esp
    i = i+1;
```

```
    x1 = D\((D-A)*x1+b);
    document(i,1) = i;
    document(i,2) = norm(x1 - xexact,inf);
end
semilogy(document(:,1),document(:,2),'DisplayName',
'Jacobi迭代法');
%记录横轴纵轴的数据画图
xlabel('迭代次数');
ylabel('误差大小');
end

function GS(A,b,xexact,esp)
x2 = zeros(10,1);
i = 0;
D = diag(diag(A));
L = tril(A,-1);
U = triu(A,1);
while norm(x2 - xexact,inf) > esp
    i = i+1;
    x2 = (D+L)\(b-U*x2);
    document(i,1) = i;
    document(i,2) = norm(x2 - xexact,inf);
end
semilogy(document(:,1),document(:,2),'DisplayName',
'Gauss-Seidel迭代法');
%记录横轴纵轴的数据画图
end

% 2
function SOR(A,b,xexact,esp,w)
x3 = zeros(10,1);
i = 0;
D = diag(diag(A));
L = tril(A,-1);
U = triu(A,1);
```

```
I = eye(10);
while (norm(x3 - xexact,inf) > esp) && (i<2500)
    i = i+1;
    x3 = (I+D\L*w)\((I-w*(D\U+I))*x3+D\b*w);
    document(i,1) = i;
    document(i,2) = norm(x3 - xexact,inf);
end
%记录横轴纵轴的数据画图
xlabel('迭代次数');
ylabel('误差大小');
name1 = ['SOR w=',num2str(w)];      %数字字母拼接
semilogy(document(:,1),document(:,2),'DisplayName',name1);
name2 = [' \leftarrow ',name1];     %w的数字指向线
text(75,document(75,2),name2);
legend
end

% 3
function newjacobi(A,b,xexact,esp)
% 其中, A 为线性方程组的系数矩阵, b 为常数项, eps为精度要求
x0 = zeros(10,1);
i = 0;
x = x0;
while norm(x - xexact,inf) > esp
    i = i+1;
    x0 = x;
    x(1,1) = (b(1)-(A(1,2)*x0(2,1)))/A(1,1);
    for j = 2:9
        x(j,1) = (b(j)-(A(j,j-1)*x0(j-1,1) +
        A(j,j+1)*x0(j+1,1)))/A(j,j);
    end
    x(10,1) = (b(10)-(A(10,9)*x0(9,1)))/A(10,10);
    document(i,1) = i;
    document(i,2) = norm(x - xexact,inf);
end
```

```
semilogy(document(:,1),document(:,2),'DisplayName',
'新Jacobi迭代法');
%记录横轴纵轴的数据画图
xlabel('迭代次数');
ylabel('误差大小');
end

function xgs = newGS(A,b,xexact,esp)
x0 = zeros(10,1);
i = 0;
x = x0;
while norm(x - xexact,inf) > esp
    i = i+1;
    x0 = x;
    x(1,1) = (b(1)-(A(1,2)*x0(2,1)))/A(1,1);
    for j = 2:9
        x(j,1) = (b(j)-(A(j,j-1)*x(j-1,1) +
        A(j,j+1)*x0(j+1,1)))/A(j,j);
    end
    x(10,1) = (b(10)-(A(10,9)*x(9,1)))/A(10,10);
    document(i,1) = i;
    document(i,2) = norm(x - xexact,inf);
end
xgs = x;
end

function newSOR(A,b,xexact,esp,w)
xGS = newGS(A,b,xexact,esp);
x0 = zeros(10,1);
i = 0;
x = x0;
while norm(x - xexact,inf) > esp && (i<2500)
    i = i+1;
    x0 = x;
    for j = 1:10
```

```
        x(j,1) = w*xGS(j,1) + (1-w)*x0(j,1);
    end
end
end
```

MATLAB 第一题最终演示代码显示如下：

```
jacobi(A,b,xexact,esp);
hold on
GS(A,b,xexact,esp);
w = 0.5;
while w < 2
    SOR(A,b,xexact,esp,w);
    w = w+0.5;
end
w = 1.618;
SOR(A,b,xexact,esp,w);

jacobi(A,b,xexact,esp);
fprintf('jacobi改进前的时间');
tic
for i = 1:10
    jacobi(A,b,xexact,esp);
end
toc
newjacobi(A,b,xexact,esp);
fprintf('jacobi改进后的时间');
tic
for i = 1:10
    newjacobi(A,b,xexact,esp);
end
toc

GS(A,b,xexact,esp)
fprintf('GS改进前的时间');
tic
```

```
for i = 1:10
    GS(A,b,xexact,esp);
end
toc
newGS(A,b,xexact,esp);
fprintf('GS改进后的时间');
tic
for i = 1:10
    newGS(A,b,xexact,esp);
end
toc

w = 0.5;
while w<2
    SOR(A,b,xexact,esp,w);
    fprintf('SOR改进前的时间');
    tic
    for i = 1:10
        SOR(A,b,xexact,esp,w);
    end
    toc
    newSOR(A,b,xexact,esp,w);
    fprintf('SOR改进后的时间');
    tic
    for i = 1:10
        newSOR(A,b,xexact,esp,w);
    end
    toc
    w = w+0.4;
end
```

MATLAB 第一题最终结果显示如下:

```
jacobi改进前的时间历时 0.369507 秒。
jacobi改进后的时间历时 0.229690 秒。
GS改进前的时间历时 0.017519 秒。
GS改进后的时间历时 0.008679 秒。
```

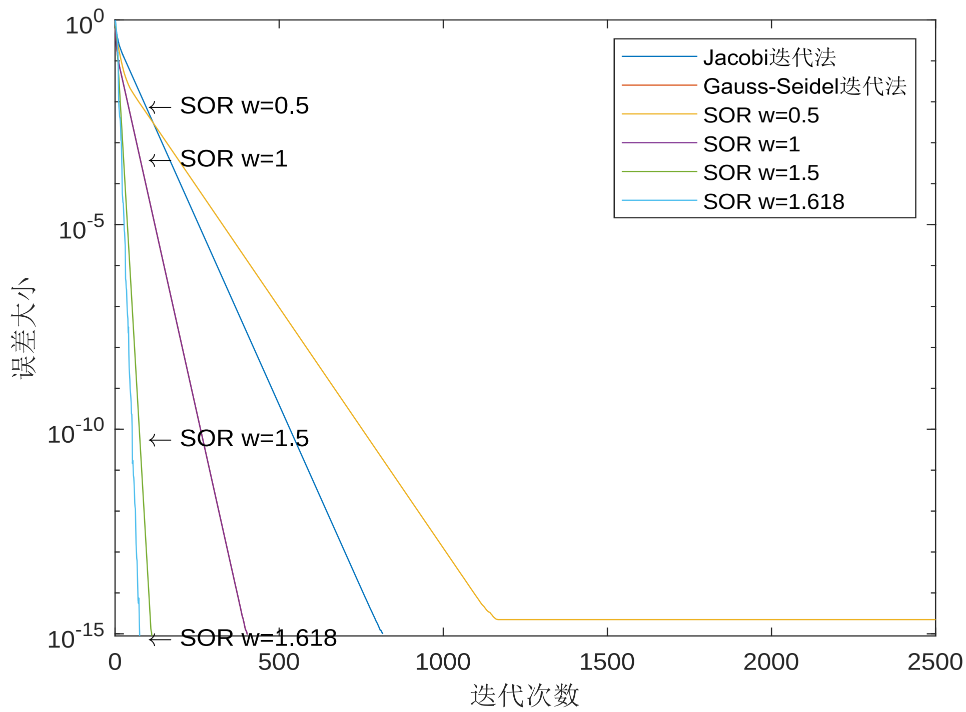

图 1: yes

```
SOR w=0.5改进前的时间历时 0.658485 秒。
SOR w=0.5改进后的时间历时 0.006349 秒。
SOR w=1改进前的时间历时 0.527591 秒。
SOR w=1改进后的时间历时 0.003656 秒。
SOR w=1.5改进前的时间历时 0.577475 秒。
SOR w=1.5改进后的时间历时 0.003579 秒。
SOR w=1.618改进前的时间历时 0.583528 秒。
SOR w=1.618改进后的时间历时 0.004871 秒。
>>
```

从图像当中也可以看出 **1.618** 左右时的 **SOR** 收敛速度快于其他的 ω 的值的收敛速度。

第二题 本题将利用求解方程 **x3−3x2+ 2 = 0** 的根来深入我们关于 **Newton** 方法的收敛速度的讨论。

MATLAB 程序显示如下:

```
clear , clc

%[-3,0]、[0,2]、[2,4]
NewTon(-1.7);
i = 0.1;
while i<2
    NewTon(i);
    i=i+.5;
end
NewTon(2.5);

function NewTon(a)
syms x;
% 方程为f(x) = 0
f(x) = x^3 - 3*x^2 + 2;
%这是求迭代次数与迭代值
df(x) = diff(f(x),1);
h(x) = f(x)/df(x);
x0 = a;
%这是求迭代次数与迭代值
for i =1:100
    x1 = x0 - h(x0);
    if norm(x1-x0)<1e-15
        document(i,1) = i;
        document(i,2) = x1;
        break;
    else
        document(i,1) = i;
        document(i,2) = x1;
        x0 = x1;
    end
    fprintf('%d     %.15f\n',document(i,1),document(i,2));
```

```
end
g(x) = x - f(x)/df(x);
for p = 1:5
    g(x) = diff(g(x));
    t = double(g(x1));          %转成浮点数
    if roundn(t,-15) ~= 0          %保留小数点后15位
        break;
    end
end
fprintf('阶数估计p = %d\n',p);
fprintf('\n');
end
```

MATLAB 第二题最终结果显示如下：

```
1      -1.086168521462639
2      -0.805676955223314
3      -0.736322130064943
4      -0.732066517116726
5      -0.732050807782599
6      -0.732050807568877
阶数估计p = 2


1      3.557894736842105
2      3.012915484777833
3      2.781661532896955
4      2.734048844457692
5      2.732054255592772
6      2.732050807579173
7      2.732050807568877
阶数估计p = 2


1      1.050793650793651
2      0.999912409107207
3      1.000000000000448
4      1.000000000000000
阶数估计p = 3
```

```
1       0.999326599326599
2       1.000000000203577
3       1.000000000000000
阶数估计p = 3

1       0.775000000000000
2       1.007998683344306
3       0.999999658813342
4       1.000000000000000
阶数估计p = 3

1       2.800000000000000
2       2.735714285714286
3       2.732062373480407
4       2.732050807684724
5       2.732050807568877
阶数估计p = 2
```

（c）关于 c 小问，通过第二问简单粗暴的整型判断，发现在判定 1 附近的根时最后的收敛阶数出现大于 2 的情况。根据上课所听讲的内容：电脑（MATLAB）在处理求根问题的时候，若当前点距离精确值较远会采取“作弊”，即调用比 Newton 法局部更快速收敛的方法进行优化，从而使得可能在局部出现比二阶收敛更快的现象。而且牛顿法的二阶也只是整体近似，局部出现微小偏差也情有可原。

第三题 我们已经学习了使用幂法求解特征值问题。

(a) 关于 **a** 小问，算法思想伪代码如下：根据课堂中的思想改进后的算法：

$$q^{old} = (1, 1, \ldots, 1)^T \tag{1}$$

$$\hat{q}^{old} = q^{old}/\|q^{old}\|_\infty m, \varepsilon \tag{2}$$

$$for\ k = 1 : m \tag{3}$$

$$q^{new} = A\hat{q}^{old} \tag{4}$$

$$\lambda = \|q^{new}\|_\infty \tag{5}$$

$$\hat{q}^{new} = q^{new}/\lambda \tag{6}$$

$$if(\left\|\hat{q}^{old} - \hat{q}^{new}\right\|_\infty < \varepsilon) \tag{7}$$

$$\hat{q}^{old} = \hat{q}^{new} \tag{8}$$

$$return\ \lambda, hatq^{new}, break \tag{9}$$

$$elseif(\left\|\hat{q}^{old} + \hat{q}^{new}\right\|_\infty < \varepsilon) \tag{10}$$

$$return\ -\lambda, hatq^{new}, break \tag{11}$$

$$\hat{q}^{old} = \hat{q}^{new} \tag{12}$$

$$elseif(\left\|\hat{q}^{old} - \hat{q}^{newnew}\right\|_\infty < \varepsilon) \tag{13}$$

$$return\ \lambda^2, hatq^{newnew}, break \tag{14}$$

$$\hat{q}^{old} = \hat{q}^{new} \tag{15}$$

end

end

Matlab 程序显示如下：

```
clear, clc

A = [-148 -105 -83 -67;
    488 343 269 216;
    -382 -268 -210 -170;
    50 38 32 29]
power(A);

```

```
B = [222 580 584 786;
    -82 -211 -208 -288;
    37 98 101 132;
    -30 -82 -88 -109]

power(B);

function power(A)
lam2 = -1;
% lam2最后通过判断是否存在两个最大特征值
%幂法法二
qold = ones(4,1);
q_old = qold/norm(qold,inf);
% 归一化, 除以无穷范数
for j = 1:1000
    qnew = A * q_old;
    lam = norm(qnew,inf);
    %  作为特征值
    q_new = qnew/lam;
    %归一化
    % 出现俩绝对值相等的最大特征值时:
    qnewn = A * A * q_old;
    qnn = A * q_new;
    q_nn = qnn/norm(qnn,inf);
    if norm(q_old - q_new,inf) < 1e-15
        %特征值为正时
        break;
    elseif norm(q_old + q_new,inf) < 1e-15
        %特征值为负时
        lambda = -lambda;
        break;
    elseif norm(q_old - q_nn,inf) < 1e-15
        %不满足前两个条件, 返回特征值的平方
        lam2 = norm(qnewn,inf) / norm(q_old,inf);
        %得到特征值的平方
```

```
        break;
    end
    q_old = q_new;
end
fprintf('迭代次数：%d\n',j);

if lam2 > 0
    %俩绝对值相等的最大特征值时
    lam_1 = sqrt(lam2);       %正特征值
    lam_2 = -lam_1;           %负特征值
    fprintf('模最大的特征值：\n');
    fprintf('%.15f\n',lam_1);
    fprintf('%.15f\n',lam_2);
    fprintf('模最大的特征向量\n');
    disp(q_nn);
    disp(q_new);
else
    %只有一个绝对值最大时
    fprintf('模最大的特征值：');
    fprintf('%.16f\n',lam);
    fprintf('模最大的特征向量：\n');
    disp(q_new);
end
end
```

MATLAB 第三题前三问最终结果显示如下：

```
A =

  -148  -105   -83   -67
   488   343   269   216
  -382  -268  -210  -170
    50    38    32    29

迭代次数：41
模最大的特征值：8.0000000000002132
模最大的特征向量：
```

```
   -0.3103
    1.0000
   -0.7931
    0.1379

B =


   222   580   584   786
   -82  -211  -208  -288
    37    98   101   132
   -30   -82   -88  -109

迭代次数: 85
模最大的特征值:
4.999999999999742
-4.999999999999742
模最大的特征向量
   -1.0000
    0.2857
   -0.1786
    0.2143

    1.0000
   -0.3636
    0.1591
   -0.1364


>>
```

Matlab 第三题最后一问由反幂法和平移最终结果显示如下:

```
clear,clc

p = 0.8-0.6*1i;
rng(2);
A = 2*rand(100)-1;

```

```
InversePower(A,p);

function InversePower(A,p)
I = eye(100,100);
u0 = ones(100,1);
v = (A - p * I) \ u0;
u = v / norm(v, inf);
i = 0;
while norm(u - u0, inf)>1e-15 && i<10000
    u0 = u;
    v = (A - p * I) \ u0;
    u = v / norm(v, inf);
    i = i + 1;
end
fprintf('迭代次数：%d\n',i);
fprintf('特征向量：');
disp(u);
lam = p + 1/norm(v,inf);
fprintf('最接近的特征值：%d\n',lam);
end
```