作业一

姓名 和泳毅　　学号 PB19010450 日期

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 (boundary value problem)

$$-u''(x) = f(x) \quad (0 < x < 1) \text{ 使得 } u(0) = T_0 \quad \text{and} \quad u(1) = T_1$$

时产生的离散化线性系统

$$Ax = b$$

的求解问题。适当选取离散化的步长后我们会得到一个 10×10 的系统：

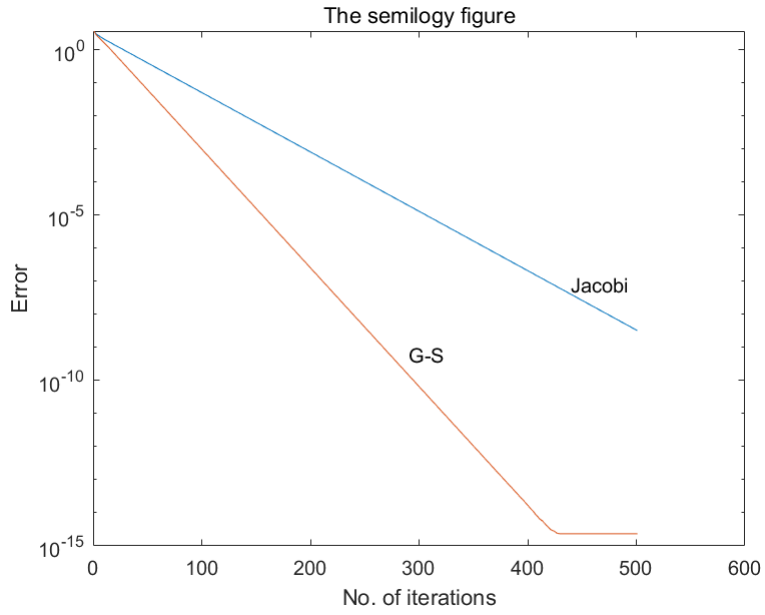
$$A = \begin{bmatrix} 2 & -1 & & & & & & & & \\ -1 & 2 & -1 & & & & & & & \\ & -1 & 2 & \ddots & & & & & & \\ & & \ddots & \ddots & -1 & & & & & \\ & & & -1 & 2 & -1 & & & & \\ & & & & -1 & 2 & & & & \\ & & & & & & & & & \end{bmatrix}, \quad b = \begin{bmatrix} 2 & -2 & 2 & -1 & 0 & 0 & 1 & -2 & 2 & -2 \end{bmatrix}^T$$

此处，A 中空白部分的元素皆为 0。我们容易验证上述线性系统的精确解为

$$x_{exact} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix}^T \tag{1}$$

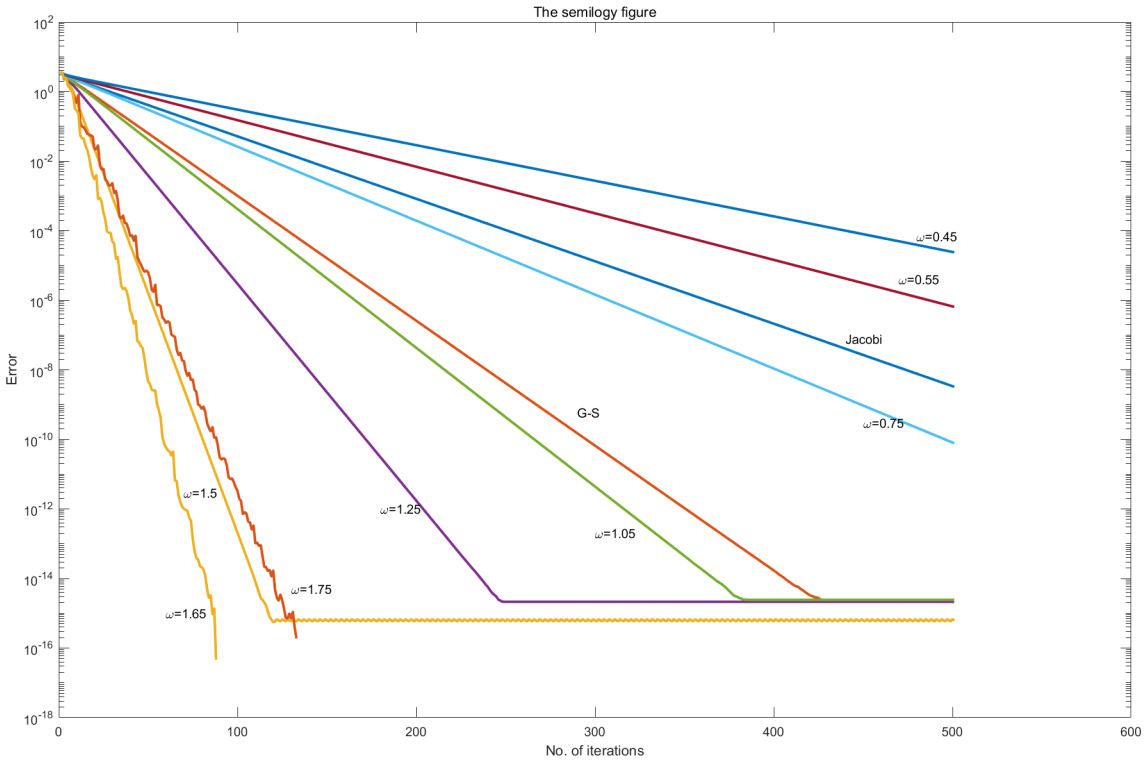
(a)（20 分）分别使用 Jacobi 和 Gauss-Siedel 方法求解上述问题。利用精确解 (1) 将误差大小和迭代次数的关系用 **semilogy** 图表示出来（横轴为迭代次数 n，纵轴为迭代解与精确解的差距）。

答：代码见题末。**semilogy** 图如下：



(b)（10 分）选取若干不同的松弛因子 ω 使用 SOR 方法解上述问题，并在收敛结果画在上一问的图中。请在图上相应的收敛线旁标示出这些 ω 的值。以迭代次数做为判断标准，指出对应于 10^{-15} 的误差目标哪个大概的 ω 值收敛速度最快。

答：代码见题末。对应于 10^{-15} 的误差目标，当 ω 值在 1.65 左右时，收敛速度最快 **semilogy** 图如下：



(c)（15 分）注意到题目中的矩阵 A 是一个稀疏矩阵 (sparse matrix)，即有大量元素

为 0 的矩阵。更改你的程序，省略那些和零元素相关的运算，使得你的程序得到加速。使用 MATLAB 中的 tic 和 toc 命令统计上述三种方法得到较为精确的解的时候的计算用时，并和改进后的程序的（在使用相同迭代次数的情况下的）耗时列表做对比（左边一列为未加速的程序的计算时间，右边一列为加速后的时间）。注意你需要将每种方法反复运行 N 次（比如 10 次）然后忽略第一次的运行时间，求后面 $N-1$ 次运行时间的平均值或者总和。这是由于 MATLAB 需要在第一次运算时对程序进行编译并分配存储空间。这类花费被统称为 overhead，中文有时会勉强地将其译为“额外开销”。

答：

	$Oldtime$	$Newtime$
$Jacobi:$	$0.001402s$	$0.000169s$
$G-S:$	$0.001217s$	$0.000176s$
$SOR_{w=1.65}:$	$0.000558s$	$0.000037s$
$SOR_{w=1.05}:$	$0.003614s$	$0.000342s$
$SOR_{w=1.75}:$	$0.001107s$	$0.000057s$
$SOR_{w=1.25}:$	$0.003431s$	$0.000229s$
$SOR_{w=0.75}:$	$0.003359s$	$0.000198s$
$SOR_{w=0.45}:$	$0.003397s$	$0.000183s$

完整代码如下：

```
clear,clc;
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]';
x_exact=[1 0 1 0 0 0 0 -1 0 -1]';
x0=[ones(1,size(b,1))]';
%%
%误差图
[a1,b1,~]=Jacobi(A,b,x0,x_exact);
[a2,b2,~]=G_S(A,b,x0,x_exact);
[a3,b3,~]=SOR(A,b,1.5,x0,x_exact);
[a4,b4,~]=SOR(A,b,1.25,x0,x_exact);
[a5,b5,~]=SOR(A,b,1.05,x0,x_exact);
```

```
[a6,b6,~]=SOR(A,b,0.75,x0,x_exact);
[a7,b7,~]=SOR(A,b,0.55,x0,x_exact);
[a8,b8,~]=SOR(A,b,0.45,x0,x_exact);
[a9,b9,~]=SOR(A,b,1.75,x0,x_exact);
[a10,b10,~]=SOR(A,b,1.65,x0,x_exact);

n1=1:1:a1;
n2=1:1:a2;
n3=1:1:a3;
n4=1:1:a4;
n5=1:1:a5;
n6=1:1:a6;
n7=1:1:a7;
n8=1:1:a8;
n9=1:1:a9;
n10=1:1:a10;

figure
semilogy(n1,b1,n2,b2,n3,b3,n4,b4,n5,b5, ...
    n6,b6,n7,b7,n8,b8,n9,b9,n10,b10);
title('The semilogy figure');
xlabel('No. of iterations');
ylabel('Error');
text(440,8e-8,'Jacobi');
text(290,6e-10,'G-S');
text(70,3e-12,'\omega=1.5');
text(180,1e-12,'\omega=1.25')
text(300,2e-13,'\omega=1.05')
text(450,3e-10,'\omega=0.75')
text(470,4e-6,'\omega=0.55')
text(480,7e-5,'\omega=0.45')
text(130,5e-15,'\omega=1.75')
text(60,1e-15,'\omega=1.65')
%%
%稀疏矩阵改进与用时比较
```

```
for i=1:10
    [~,~,t1]=Jacobi(A,b,x0,x_exact);
    t(i)=t1;
end
T(1) = sum(t(2:10))/9;
for i=1:10
    t2=Jacobi_new(A,b,x0,x_exact);
    t(i)=t2;
end
T(2) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=G_S(A,b,x0,x_exact);
    t(i)=t1;
end
T(3) = sum(t(2:10))/9;
for i=1:10
    t2=G_S_new(A,b,x0,x_exact);
    t(i)=t2;
end
T(4) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=SOR(A,b,1.65,x0,x_exact);
    t(i)=t1;
end
T(5) = sum(t(2:10))/9;
for i=1:10
    t2=SOR_new(A,b,1.65,x0,x_exact);
    t(i)=t2;
end
T(6) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=SOR(A,b,1.05,x0,x_exact);
    t(i)=t1;
end
T(7) = sum(t(2:10))/9;
```

```
for i=1:10
    t2=SOR_new(A,b,1.05,x0,x_exact);
    t(i)=t2;
end
T(8) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=SOR(A,b,1.75,x0,x_exact);
    t(i)=t1;
end
T(9) = sum(t(2:10))/9;
for i=1:10
    t2=SOR_new(A,b,1.75,x0,x_exact);
    t(i)=t2;
end
T(10) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=SOR(A,b,1.25,x0,x_exact);
    t(i)=t1;
end
T(11) = sum(t(2:10))/9;
for i=1:10
    t2=SOR_new(A,b,1.25,x0,x_exact);
    t(i)=t2;
end
T(12) = sum(t(2:10))/9;
for i=1:10
    [~,~,t1]=SOR(A,b,0.75,x0,x_exact);
    t(i)=t1;
end
T(13) = sum(t(2:10))/9;
for i=1:10
    t2=SOR_new(A,b,0.75,x0,x_exact);
    t(i)=t2;
end
T(14) = sum(t(2:10))/9;
```

```
for i=1:10
    [~,~,t1]=SOR(A,b,0.45,x0,x_exact);
    t(i)=t1;
end
T(15) = sum(t(2:10))/9;
for i=1:10
    t2=SOR_new(A,b,0.45,x0,x_exact);
    t(i)=t2;
end
T(16) = sum(t(2:10))/9;

dashString = repmat('-', 1, 35);
fprintf('               Oldtime     Newtime\n');
fprintf('%s\n', dashString);
fprintf('Jacobi:');
fprintf('      %fs   %fs\n',T(1),T(2));
fprintf('G-S:');
fprintf('          %fs   %fs\n',T(3),T(4));
fprintf('SOR w=1.65:');
fprintf('  %fs   %fs\n',T(5),T(6));
fprintf('SOR w=1.05:');
fprintf('  %fs   %fs\n',T(7),T(8));
fprintf('SOR w=1.75:');
fprintf('  %fs   %fs\n',T(9),T(10));
fprintf('SOR w=1.25:');
fprintf('  %fs   %fs\n',T(11),T(12));
fprintf('SOR w=0.75:');
fprintf('  %fs   %fs\n',T(13),T(14));
fprintf('SOR w=0.45:');
fprintf('  %fs   %fs\n',T(15),T(16));
fprintf('%s\n', dashString);
%%
%Jacobi
function [k,y,t]=Jacobi(A,b,x0,x_exact)
    tic;
```

```
    D = diag(diag(A));
    x_old = x0;
    x_new = [zeros(1,size(b,1))]';
    y = [norm(x_old-x_exact)];%初始误差
    k = 1;
    m = 500;%次数上限
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new = D\((D-A)*x_old+b);%迭代
            y = [y,norm(x_new-x_exact)];
            k = k+1;
            x_old = x_new;
        else
            break;
        end
    end
    t = toc;
%   fprintf('The result of the Jacobi method is:\n');
%   showvector(x_new);
end

%G-S
function [k,y,t]=G_S(A,b,x0,x_exact)
    tic;
    L = tril(A,-1);
    U = triu(A,1);
    D = diag(diag(A));
    x_old = x0;
    x_new = [zeros(1,size(b,1))]';
    y = [norm(x_old-x_exact)];%初始误差
    k = 1;
    m=500; %次数上限
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new = (D+L)\(b-U*x_old);%迭代
```

```
            y = [y,norm(x_new-x_exact)];
            x_old = x_new;
            k = k+1;
        else
            break;
        end
    end
    t = toc;
%   fprintf('The result of the Gauss-Seidel method is:\n');
%   showvector(x_new);
end

%SOR
function [k,y,t]=SOR(A,b,w,x0,x_exact)
    tic;
    L = tril(A,-1);
    U = triu(A,1);
    D = diag(diag(A));
    I=eye(size(A));
    x_old = x0;
    x_new = [zeros(1,size(b,1))]';
    y = [norm(x_old-x_exact)];%初始误差
    k = 1;
    m = 500;%次数上限
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new=(I+D\L*w)\ ...
                ((I-w*(D\U+I))*x_old+D\b*w);%迭代
            y = [y,norm(x_new-x_exact)];
            x_old = x_new;
            k = k+1;
        else
            break;
        end
    end
```

```
    t = toc;
%   fprintf ...
%   ('The result of the SOR method with w=%.2f is: \n',w);
%   showvector(x_new);
end

%Jacobi_new
function t=Jacobi_new(A,b,x0,x_exact)
    tic;
    D = diag(diag(A));
    x_old = x0;
    n = size(b,1);
    x_new = [zeros(1,n)]';
    k = 1;
    m = 500;%次数上限
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new(1)=(b(1)-A(1,2)*x_old(2))/D(1,1);
            for i = 2:n-1
                x_new(i)=(b(i)-A(i,i-1)*x_old(i-1) ...
                    -A(i,i+1)*x_old(i+1))/D(i,i);
            end
            x_new(n)=(b(n)-A(n,n-1)*x_old(n-1))/D(n,n);
            k = k+1;
            x_old = x_new;
        else
            break;
        end
    end
    t = toc;
   %fprintf('The result of the Jacobi method is:\n');
   %showvector(x_new);
end

%G-S_new
```

```
function t=G_S_new(A,b,x0,x_exact)
    tic;
    L = tril(A,-1);
    U = triu(A,1);
    D = diag(diag(A));
    n = size(b,1);
    x_old = x0;
    x_new = [zeros(1,n)]';
    k = 1;
    m = 500; %次数上限
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new(1)=(b(1)-A(1,2)*x_old(2))/D(1,1);
            for i=2:n-1
                x_new(i)=(b(i)-A(i,i-1)*x_new(i-1) ...
                    -A(i,i+1)*x_old(i+1))/D(i,i);
            end
            x_new(n)=(b(n)-A(n,n-1)*x_new(n-1))/D(n,n);
            x_old = x_new;
            k = k+1;
        else
            break;
        end
    end
    t = toc;
   %fprintf('The result of the Gauss-Seidel method is:\n');
   %showvector(x_new);
end

%SOR_new
function t=SOR_new(A,b,w,x0,x_exact)
    tic;
    L = tril(A,-1);
    U = triu(A,1);
    D = diag(diag(A));
```

```
    I = eye(size(A));
    x_old=x0;
    x_new=[zeros(1,size(b,1))]';
    n = size(b,1);
    k = 1;
    m = 500;
    while norm(x_new-x_exact) > eps
        if k <= m
            x_new(1)=w*(b(1)-A(1,2)*x_old(2)) ...
                /D(1,1)+(1-w)*x_old(1);
            for i = 2:n-1
                x_bar=(b(i)-A(i,i-1)*x_new(i-1) ...
                    -A(i,i+1)*x_old(i+1))/D(i,i);
                x_new(i)=w*x_bar+(1-w)*x_old(i);
            end
            x_new(n)=w*(b(n)-A(n,n-1)*x_new(n-1)) ...
                /D(n,n)+(1-w)*x_old(n);
            x_old = x_new;
            k = k+1;
        else
            break;
        end
    end
    t = toc;
  %fprintf ...
  % ('The result of the SOR method with w=%.2f is: \n',w);
  %showvector(x_new);
end

function showvector(v)%打印向量
    [n,~] = size(v);
    fprintf('(');
    fprintf('%.3f',v(1));
    for k = 2:n
        fprintf(',%.3f',v(k));
```

```
    end
    fprintf(')^T\n\n');
end
```

第二题 本题将利用求解方程

$$x^3 - 3x^2 + 2 = 0 \tag{2}$$

的根来深入我们关于 Newton 方法的收敛速度的讨论。容易验证 (2) 的三个根分别位于 $[-3, 0]$、$[0, 2]$、$[2, 4]$ 三个区间内。我们依次从左向右的顺序分别称这三个根为 x_l、x_m、x_r。

(a)（10 分）适当选取迭代的初始点，写程序用 Newton 法求解这三个根，并将每一步迭代的新的近似值打印出来。

答：完整代码见题末。迭代过程如下：

The left one:

iter.	x	$log(\|err\|)$	*order*
1	−0.984126984126984	−0.66189464	
2	−0.773163127830252	−1.55606846	
3	−0.733437807950018	−3.22576651	2.073
4	−0.732052470495501	−6.58181152	2.040
5	−0.732050807571272	−13.30693292	2.022
6	−0.732050807568877	−26.75770624	2.011

The solution is: −0.732050807568877

The middle one:

iter.	x	$log(\|err\|)$	*order*
1	1.111111111111111	−0.49247649	
2	0.999074074074074	−2.18892577	
3	1.000000000529222	−6.98471575	3.191
4	1.000000000000000	−21.35961321	3.058

The solution is: 1.000000000000000

The right one:

iter.	x	$log(\|err\|)$	*order*
1	2.777777777777778	−1.50407740	
2	2.733756613756614	−3.12308476	
3	2.732053321730378	−6.37519241	2.041
4	2.732050807574351	−12.89357339	2.022
5	2.732050807568877	−25.93098684	2.011

The solution is: 2.732050807568877

(b)（10 分）设计一个估计收敛阶数的方法，在上一问求解的过程中同时求出大概的收敛阶数。

答：由收敛阶数定义我们需要估计出：

$$\lim_{k\to+\infty}\frac{\epsilon_{k+1}}{{\epsilon_k}^q}=\mu,\quad(\mu>0,\quad q>1)$$

中的 q，由于我们事先无法得知精确解，用两次迭代结果差的绝对值来近似误差：

$$q\approx\frac{log(\epsilon_{k+1})}{log(\epsilon_k)}\approx\frac{log(|x_{k+2}-x_{k+1}|)}{log(|x_{k+1}-x_k|)}$$

求得 Newton 迭代的收敛阶数约为 2。

(c)（10 分）Newton 是一个二阶收敛的方法。上一问中你是否观测到了比二阶收敛更快的现象？如果有，请尽可能详细地解释其原因。

答：在求 x_m 的迭代过程中观测到比二阶收敛更快的现象。

原因思考：回顾 Newton 迭代的几何意义，是从初值开始与 $f(x)$ 的图像不断地做切线。收敛的速度与根（不动点）附近曲线的斜率和初值的选取有关。

分析 $f(x)$，

$$f^{'}(x)=3x^2-6x\quad f^{''}(x)=6x-6\quad f^{'''}(x)=6$$

$x=1$ 是 $f(x)$ 在 $[0,2]$ 区间内的斜率最大值（绝对值意义上），且从 $x=1$ 开始，斜率绝对值向两边对称地减小。

而在 (b) 中，我们用两次迭代结果的差来近似误差 ϵ。于是当我们在 x_m 的邻域里迭代，由于 $x=1$ 在此邻域内的斜率绝对值最大且向两侧对称地减小，使得每一

次新的迭代结果对应的斜率绝对值比上一次的要大，每两次迭代结果差的绝对值越来越小，则有可能观测到的迭代收敛阶数比二阶更快的情况。
回归到定理本身，对 $f(x)$ 在不动点 x_m 处做 $Taylor$ 展开：

$$0 = f(x_m) = f(x_k) + (x_m - x_k)f^{'}(x_k) + \frac{(x_m - x_k)^2}{2}f^{''}(\eta_k), \quad \eta_k \in [x_m, x_k]$$

$$x_m - x_k + \frac{f(x_k)}{f^{'}(x_k)} = -\frac{(x_m - x_k)^2 f^{''}(\eta_k)}{2f^{'}(x_k)}, \quad f^{'}(x_k) \neq 0$$

$$|x_m - x_{k+1}| = |\frac{(x_m - x_k)^2 f^{''}(\eta_k)}{2f^{'}(x_k)}|$$

$$k \to \infty, \quad x_k \to x_m \quad |x_m - x_k| \to \epsilon_k \quad |x_m - x_{k+1}| \to \epsilon_{k+1}$$

而因为

$$f^{'}(x) = 3x^2 - 6x \quad f^{''}(x) = 6x - 6$$

$$f^{'}(1) \neq 0 \quad f^{''}(1) = 0$$

所以

$$k \to \infty, \quad x_k \to 1 \quad f^{''}(\eta_k) \to 0 \quad \frac{\epsilon_{k+1}}{{\epsilon_k}^2} \to 0$$

则可能出现迭代收敛阶数比二阶更快的情况，进一步的：

$$0 = f(x_k) + (x_m - x_k)f^{'}(x_k) + \frac{(x_m - x_k)^2}{2}f^{''}(x_k) + \frac{(x_m - x_k)^3}{3!}f^{'''}(\eta_k), \quad \eta_k \in [x_m, x_k]$$

$$x_m - x_k + \frac{f(x_k)}{f^{'}(x_k)} = -\frac{(x_m - x_k)^2 f^{''}(x_k)}{2f^{'}(x_k)} - \frac{(x_m - x_k)^3 f^{'''}(\eta_k)}{3!f^{'}(x_k)}$$

$$|x_m - x_{k+1}| = |x_m - x_k|^3 |\frac{f^{''}(x_k)}{2f^{'}(x_k)(x_m - x_k)} + \frac{f^{'''}(\eta_k)}{3!f^{'}(x_k)}|$$

$$\begin{aligned}\frac{\epsilon_{k+1}}{{\epsilon_k}^3} &= |\frac{f^{''}(x_k)}{2f^{'}(x_k)(x_m - x_k)} + \frac{f^{'''}(\eta_k)}{3!f^{'}(x_k)}| = |\frac{f^{''}(x_k) - f^{''}(x_m)}{2f^{'}(x_k)(x_m - x_k)} + \frac{f^{'''}(\eta_k)}{3!f^{'}(x_k)}| \\ &= |\frac{f^{'''}(\lambda_k)(x_k - x_m)}{2f^{'}(x_k)(x_m - x_k)} + \frac{f^{'''}(\eta_k)}{3!f^{'}(x_k)}| = |\frac{-3f^{'''}(\lambda_k) + f^{'''}(\eta_k)}{3!f^{'}(x_k)}|\end{aligned}$$

其中 $\eta_k \in [x_m, x_k]$，$\lambda_k \in [x_m, x_k]$，

$$k \to \infty, \quad x_k \to x_m, \quad -3f^{'''}(\lambda_k) + f^{'''}(\eta_k) \neq 0$$

综上，可以解释当不动点处函数二阶导为 0 时，可能出现迭代收敛阶数比二阶更快的情况，并且可以解释结果中出现三阶收敛情况的原因。
完整代码：

```
syms x;
f(x)=x^3-3*x^2+2;
df(x)=diff(f(x));
fprintf('\nThe left one:\n');
NewtonIteration(f,df,-1.5);
fprintf('\nThe middle one:\n');
NewtonIteration(f,df,0.5);
fprintf('\nThe right one:\n');
NewtonIteration(f,df,3);
%牛顿迭代法
function NewtonIteration(fun,dfun,x0)
    f = fun;
    df = dfun;
    fprintf ...
     ('iter.        x              log(|err|)      order\n')
    dashString = repmat('-', 1, 50);
    fprintf('%s\n', dashString);
    x1 = x0 - f(x0)/df(x0);
    d1 = norm(x1-x0);
    k = 1;
    tol = 100;%迭代次数上限
    while d1 > eps
        if k < tol
            if k > 2 %第三次迭代开始估计收敛阶数
                fprintf ...
                    ('%d    %.15f    %.8f    %.3f\n' ...
                    ,k,x1,log(d1),log(d1)/log(d2));
            else
                fprintf ...
                    ('%d    %.15f    %.8f\n',k,x1,log(d1));
            end
            x0 = x1;
            x1 = x0 - f(x0)/df(x0);%迭代序列
            d2 = d1;
            d1 = norm(x1-x0);%两次迭代相差
```

```
            k = k + 1;
        else
            break;
        end
    end
    fprintf('%s\n', dashString);
    fprintf('The solutiion is: %.15f\n',x1);
end
```

第三题 我们已经学习了使用幂法求解特征值问题。

(a)（15 分）设计一个能够求解问题存在一个（绝对值意义下的）最大特征值和存在最大的两个特征值，大小相同但符号相反的情况的算法并仿照课堂上所介绍的伪代码的格式写出一个清晰易懂的伪代码。

答：

$q_0 = (1, 1, ..., 1)^T$

for k = 1:m

$\bar{q}_0 = q_0/||q_0||_\infty$

$q_1 = A * \bar{q}_0$

$\bar{q}_1 = q_1/||q_1||_\infty$

$q_2 = A * \bar{q}_1$

$\bar{q}_2 = q_2/||q_2||_\infty$

if $(||\bar{q}_2 - \bar{q}_0||_\infty < \epsilon_1)$

if $(||\bar{q}_1 - \bar{q}_0||_\infty < \epsilon_2)$

$\lambda = ||q_1||_\infty$

return λ,$\bar{q}_1$，exit

elseif $(||\bar{q}_1 + \bar{q}_0||_\infty < \epsilon_2)$

$\lambda = ||q_1||_\infty$

return λ,$\bar{q}_1$，exit

else

$\lambda_1 = \sqrt{\frac{||A*A*\bar{q}_0||_\infty}{||\bar{q}_0||_\infty}}$

$\lambda_2 = -\lambda_1$

$v_1 = \lambda_1 * \bar{q}_0 + q_1$

$v_2 = \lambda_1 * \bar{q}_0 - q_1$

return λ_1,λ_2,v_1,v_2，exit

end

end

$q_0 = q_1$

end

但是在实际操作中发现，存在 $||\bar{q}_2 - \bar{q}_0||_\infty$ 的收敛速度比 $||\bar{q}_1 - \bar{q}_0||_\infty$ 更快的情况，需要略微放松 ϵ_2 的限制。如果要保证精度，还可以独立地循环两次：

$q_0 = (1, 1, ..., 1)^T$

for k = 1:m

$\bar{q}_0 = q_0/||q_0||_\infty$

$q_1 = A * \bar{q}_0$

$\bar{q}_1 = q_1/||q_1||_\infty$

if $(||\bar{q}_1 - \bar{q}_0||_\infty < \epsilon)$

$\lambda = ||q_1||_\infty$

return λ,$\bar{q}_1$， exit

elseif $(||\bar{q}_1 + \bar{q}_0||_\infty < \epsilon)$

$\lambda = ||q_1||_\infty$

return λ,$\bar{q}_1$， exit

end

$q_0 = q_1$

end

$q_0 = (1, 1, ..., 1)^T$

for k = 1:m

$\bar{q}_0 = q_0/||q_0||_\infty$

$q_1 = A * \bar{q}_0$

$\bar{q}_1 = q_1/||q_1||_\infty$

$q_2 = A * \bar{q}_1$

$\bar{q}_2 = q_2/||q_2||_\infty$

if $(||\bar{q}_2 - \bar{q}_0||_\infty < \epsilon)$

$\lambda_1 = \sqrt{\frac{||A*A*\bar{q}_0||_\infty}{||\bar{q}_0||_\infty}}$

$\lambda_2 = -\lambda_1$

$v_1 = \lambda_1 * \bar{q}_0 + q_1$

$v_2 = \lambda_1 * \bar{q}_0 - q_1$

return λ_1,λ_2,v_1,v_2， exit

end

$q_0 = q_1$

end

(b)（10 分）用程序实现上一问中你设计的算法，用于求解

$$A = \begin{bmatrix} -148 & -105 & -83 & -67 \\ 488 & 343 & 269 & 216 \\ -382 & -268 & -210 & -170 \\ 50 & 38 & 32 & 29 \end{bmatrix}$$

的模最大的特征值和特征向量（你提供的特征向量的 ∞-范数应为 1）。如果用你的程序求 $-A$ 的模最大的特征值和特征向量呢？你需要保证你的程序对负值的特征值也有效。

答：代码见题末。

对 A：

$$\begin{aligned} &\text{The eigenvalue is:} \quad 8.0000000000002132 \\ &\text{The eigenvector is:} \quad (-0.23498, 0.75715, -0.6005, 0.10443)^T \end{aligned}$$

对 $-A$：

$$\begin{aligned} &\text{The eigenvalue is:} \quad -8.0000000000002132 \\ &\text{The eigenvector is:} \quad (-0.23498, 0.75715, -0.6005, 0.10443)^T \end{aligned}$$

(c)（10 分）用程序实现第一问中你设计的算法，用于求解

$$\begin{bmatrix} 222 & 508 & 584 & 786 \\ -82 & -211 & -208 & -288 \\ 37 & 98 & 101 & 132 \\ -30 & -82 & -88 & -109 \end{bmatrix}$$

的模最大的特征值和特征向量（你提供的特征向量的 ∞-范数应为 1）。

答：

$$\begin{aligned} \text{The eigenvalues are:} \quad & 4.9999999999997415 \quad \text{and} \quad -4.9999999999997415 \\ \text{The eigenvectors are:} \quad & (-0.88889, 0.44444, -0.11111, -5.2916e-14)^T \quad \text{and} \\ & (0.92582, -0.30861, 0.1543, -0.1543)^T \end{aligned}$$

(d)（10 分）在 MATLAB 中设定随机数种子为 **rng(2)** 使用 **rand** 命令生成一个 100×100 的随机矩阵。用你的程序求解离 $0.8 - 0.6i$ 最近的特征值和特征向量（你提供的特征向量的 ∞-范数应为 1）。

注意：在以上三问中，使用尽可能小的误差上限，使得你得到的特征值和特征向量尽可能地精确。

答：

$$\text{The eigenvalue is:} \quad 0.8545199176705471 - 0.6621232653482716i$$

The eigenvector is:

$$(0.044508-0.029931i, -0.17791-0.026017i, -0.077337+0.012533i, 0.058857+0.030941i,$$

$$-0.04469+0.042112i, -0.038734+0.050385i, -0.18646-0.0086692i, -0.017932+0.01756i,$$

$$-0.039082-0.032854i, 0.039008+0.012988i, 0.022029+0.049096i, 0.027878-0.010826i,$$

$$0.025441-0.11678i, -0.014153-0.011267i, 0.021319+0.0060482i, 0.050508-0.026017i,$$

$$0.075692+0.040088i, -0.10828-0.0030475i, -0.042537-0.11726i, 0.016729+0.091386i,$$

$$-0.029346 - 0.01588i, 0.019568 - 0.025973i, 0.13302 + 0.06595i, 0.1 - 0.025296i,$$

$$0.03986+0.0084844i, 0.03104-0.093359i, -0.056697-0.0074393i, -0.027351-0.030473i,$$

$$-0.033174-0.049112i, 0.044566+0.077323i, -0.16077+0.090539i, -0.11642-0.036983i,$$

$$-0.048215-0.048511i, 0.16309+0.074442i, -0.08282+0.072107i, 0.047161-0.094174i,$$

$$0.016631-0.061995i, -0.19717-0.026104i, 0.011691+0.076457i, 0.0060856-0.035701i,$$

$$-0.075164+0.048549i, -0.0023781+0.043827i, -0.16106-0.047144i, 0.083771+0.038164i,$$

$$0.026438+0.035592i, -0.047056+0.042266i, -0.020319-0.077957i, 0.048322-0.05501i,$$

$$-0.093426-0.092435i, -0.0088827+0.06144i, 0.10308-0.0099666i, -0.01126-0.0011967i,$$

$$0.11135-0.0062411i, -0.012285+0.043416i, -0.096263-0.014823i, -0.018928-0.072863i,$$

$$0.074906 - 0.061656i, 0.2261 + 0i, -0.068316 + 0.043696i, 0.070895 - 0.045002i,$$

$$-0.085324+0.16078i, -0.058553+0.068741i, 0.12288-0.041464i, 0.10775-0.046466i,$$

$$0.090504-0.1057i, -0.019689+0.0039033i, -0.0072496+0.085959i, -0.092156+0.032109i,$$

$$-0.11602-0.02359i, -0.038066+0.069228i, 0.085403+0.083028i, -0.063649-0.12802i,$$

$$0.11302-0.072857i, 0.061558-0.063598i, -0.1791-0.090414i, -0.011196+0.069164i,$$

$$0.015724-0.012852i, -0.071842+0.076944i, -0.065266+0.032298i, 0.044177-0.067161i,$$

$$0.066792-0.0017617i, 0.067547+0.034093i, 0.013354+0.033543i, -0.029022+0.0046504i,$$

$-0.0039195-0.067228i, -0.015028+0.017657i, 0.082337+0.093639i, -0.087903-0.054953i,$
$0.06602+0.042245i, 0.038881-0.075356i, 0.041218+0.077602i, 0.05021-0.0087981i,$
$-0.00058155+0.024292i, -0.038272+0.0016038i, 0.089879+0.044252i, -0.12989+0.059604i,$
$0.013063+0.028316i, 0.090256-0.047488i, 0.21955-0.010349i, -0.034483+0.00059128i)^T$

完整代码：

```
fprintf('\n第二问\n');
A=[-148 -105 -83 -67;488 343 269 216; ...
    -382 -268 -210 -170;50 38 32 29];
[a1,a2,v1,v2] = PM(A);
show(a1,a2,v1,v2)
[a1,a2,v1,v2] = PM(-A);
show(a1,a2,v1,v2)
fprintf('\n第三问\n');
A=[222 580 584 786;-82 -211 -208 -288; ...
    37 98 101 132;-30 -82 -88 -109];
[a1,a2,v1,v2] = PM(A);
show(a1,a2,v1,v2)
fprintf('\n第四问\n');
rng(2);
A=rand(100,100);
[a1,a2,v1,v2] = IPM(A,0.8-0.6i);
show(a1,a2,v1,v2)
%为了保证精度，采用两个独立for循环分别处理只有
%一个模最大的特征值和两个互为相反数的特征值的情
%况，所以预先认为题目为可解类型
%幂法
function [a1,a2,v1,v2] = PM(A)
    q0 = ones(size(A, 1), 1);
    tol = 1e-15;
    %一个正值或负值
    for k = 1:1000
        q0_bar = q0/defnorm(q0);%规范化
        q1 = A*q0_bar;
        q1_bar = q1/defnorm(q1);
```

```
        %defnorm返回模最大的分量值
        if(norm(q1_bar-q0_bar,inf) < tol)|| ...
                (norm(q1_bar+q0_bar,inf)<tol)
            a1 = defnorm(q1); a2 = a1;
            v1 = q1_bar;
            v1 = v1/norm(v1,2); v2 = v1;
            return;
        end
        q0 = q1;
    end
    %互为相反数
    q0 = ones(size(A,1),1);
    for k = 1:1000
        q0_bar = q0/defnorm(q0);
        q1 = A*q0_bar;
        q1_bar = q1/defnorm(q1);
        q2 = A*q1_bar;
        q2_bar = q2/defnorm(q2);
        if (norm(q2_bar-q0_bar,inf) < tol)
            a1 = sqrt(defnorm(A*A*q0_bar) ...
                /defnorm(q0_bar));
            a2 = -a1;
            v1 = a1*q0_bar+q1; v1 = v1/norm(v1,2);
            v2 = a1*q0_bar-q1; v2 = v2/norm(v2,2);
            break
        end
        q0 = q1;
    end
    fprintf('超过最大迭代次数\n');
end

%反幂法
function [a1,a2,v1,v2] = IPM(A,p)
    I=eye(size(A));
    B=A-p*I;
```

```
q0 = ones(size(B, 1), 1);
tol = 1e-13;
%一个正值或负值
for k = 1:100
    q0_bar = q0/defnorm(q0);
    q1 = LU(B,q0_bar);
    q1_bar = q1/defnorm(q1);
    %defnorm返回模最大的分量值
    if(norm(q1_bar-q0_bar,inf) < tol)|| ...
            (norm(q1_bar+q0_bar,inf)<tol)
        a1 = defnorm(q1);
        a1 = p+1/a1;%反幂法求得原特征值
        a2 = a1;
        v1 = q1_bar;
        v1 = v1/norm(v1,2); v2 = v1;
        return;
    end
    q0 = q1;
end
 %互为相反数
q0 = ones(size(A, 1), 1);
for k = 1:100
    q0_bar = q0/defnorm(q0);
    q1 = LU(B,q0_bar);
    q1_bar = q1/defnorm(q1);
    q2 = LU(B,q1_bar);
    q2_bar = q2/defnorm(q2);
    if (norm(q2_bar - q0_bar, inf) < tol)
        a1 = sqrt(defnorm(LU(B*B,q0_bar)) ...
            /defnorm(q0_bar));
        a1 = p+1/a1;%反幂法求得原特征值
        a2 = -a1;
        v1 = a1*q0_bar+q1; v1 = v1/norm(v1,2);
        v2 = a1*q0_bar-q1; v2 = v2/norm(v2,2);
        break
```

```
        end
        q0 = q1;
    end
    fprintf('超过最大迭代次数\n');
end

%考虑复数的带符号的模最大分量
function x=defnorm(v)
    [~,n]=max(abs(v));
    x=v(n);
end

%LU分解法解方程
function [x] = LU(A,b)
    [n,m] = size(A);
    for j = 1:m
        U(1,j) = A(1,j);
    end
    L(1,1) = 1;
    for i = 2:n
        L(i,1) = A(i,1)/A(1,1);
    end
    for i = 2:n
        for j = i:m
            sum = 0;
            for k = 1:i-1
                sum = sum+L(i,k)*U(k,j);
            end
            U(i,j) = A(i,j) - sum;
            sum = 0;
            for p = 1:i-1
                sum = sum+L(j,p)*U(p,i);
            end
            L(j,i) = (A(j,i)-sum)/U(i,i);
        end
```

```
    end
    %解Ly=b
    x = zeros(n,1);
    y(1,1) = b(1,1);
    for i = 2:n
        sum = 0;
        for j = 1:i-1
            sum = sum+L(i,j)*y(j,1);
        end
        y(i,1) = b(i) - sum;
    end
    %解Ux=y
    x(n,1) = y(n,1)/U(n,m);
    for i=n-1:-1:1
        sum = 0;
        for j = i+1:n
            sum = sum+U(i,j)*x(j,1);
        end
        x(i,1) = (y(i,1)-sum)/U(i,i);
    end
end

%考虑复数的格式化结果打印
function show(a1,a2,v1,v2)
    if(a1==a2)
        fprintf('The eigenvalue is:\n');
        if(imag(a1)==0)
            fprintf('%18.16f\n\n',a1);
        elseif(imag(a1)>0)
            fprintf('%18.16f+%18.16fi\n\n' ...
                ,real(a1),imag(a1));
        else
            fprintf('%18.16f%18.16fi\n\n' ...
                ,real(a1),imag(a1));
        end
```

```
        fprintf('The eigenvector is:\n ');
        v1 = regexprep(num2str(v1'),'\s*',',');
        disp(['(',v1,')^T']);
    else
        fprintf('The eigenvalues are:\n');
        if(imag(a1)==0)
            fprintf('%18.16f',a1);
            fprintf('  and  %18.16f\n\n',a2);
        elseif(imag(a1)>0)
            fprintf('%18.16f+%18.16fi',real(a1),imag(a1));
            fprintf('  and  %18.16f%18.16fi\n\n' ...
                ,real(a2),imag(a2));
        else
            fprintf('%18.16f%18.16fi',real(a1),imag(a1));
            fprintf('  and  %18.16f+%18.16fi\n\n' ...
                ,real(a2),imag(a2));
        end
        fprintf('The eigenvectors are:\n')
        v1 = regexprep(num2str(v1'),'\s*',',');
        v2 = regexprep(num2str(v2'),'\s*',',');
        disp(['(',v1,')^T']);
        disp(['(',v2,')^T']);
    end
end
```