作业一

姓名　　学号　　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）。

生成的图片如下

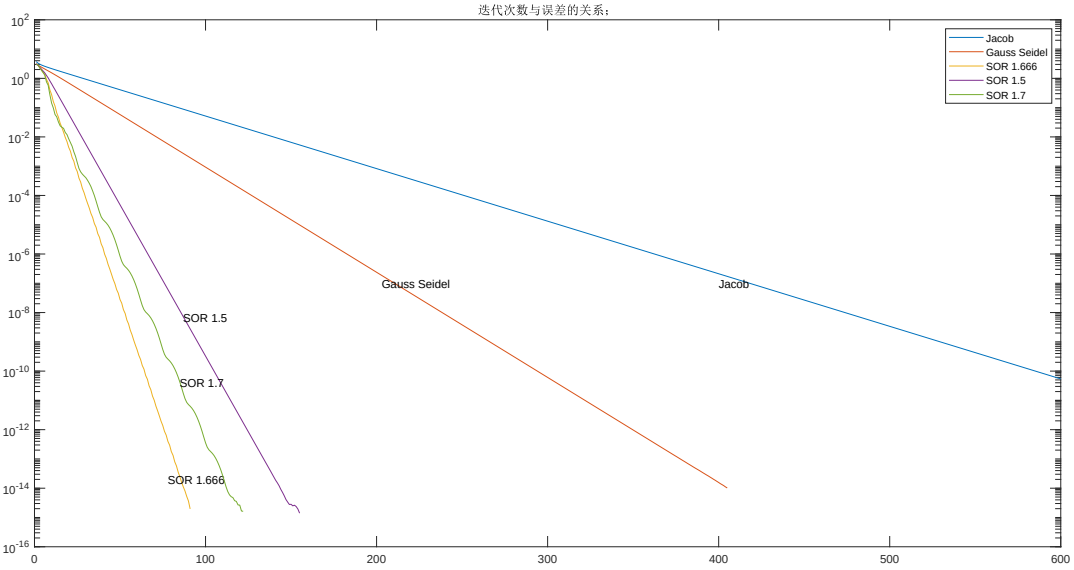

图 1: 误差与迭代次数的关系

(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：

经过反复测试，ω 约等于 1.666 时收敛速度最快，只迭代了 91 次。

(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”。

在优化模式下将三种方法累计运行 20 次，取后 19 次累计用时

优化前与优化后 19 次累计用时（秒）：

Jacob	0.0427	0.0365
Gauss-Seidel	0.2925	0.0254
ω=1.666 的 SOR	0.0671	0.0076

Matlab 程序如下：

```
A = [2 -1 0 0 0 0 0 0 0 0;
    -1 2 -1 0 0 0 0 0 0 0;
    0 -1 2 -1 0 0 0 0 0 0;
    0 0 -1 2 -1 0 0 0 0 0;
    0 0 0 -1 2 -1 0 0 0 0;
    0 0 0 0 -1 2 -1 0 0 0;
    0 0 0 0 0 -1 2 -1 0 0;
    0 0 0 0 0 0 -1 2 -1 0;
    0 0 0 0 0 0 0 -1 2 -1;
    0 0 0 0 0 0 0 0 -1 2 ];
b=[2 -2 2 -1 0 0 1 -2 2 -2]';
superior=600;%迭代次数上限
exact=[1 0 1 0 0 0 0 -1 0 -1]';%精确解
error=1e-15;%误差
xold=[1 1 1 1 1 1 1 1 1 1]';
tag=1;%是否优化:0代表不优化, 1代表优化
[y1,x]=Jacob(A,b,xold,error,exact,superior,tag);
fprintf('jacob迭代结果为: \n');disp(x);
[y2,x]=Gauss_Seidel(A,b,xold,error,exact,superior,tag);
fprintf('Gauss_Seidel迭代结果: \n');disp(x);
[y3,x]=SOR(A,b,1.666,xold,error,exact,superior,tag);
fprintf('松弛因子为1.666的SOR迭代结果:\n');disp(x);
[y4,x]=SOR(A,b,1.5,xold,error,exact,superior,tag);
fprintf('松弛因子为1.5的SOR迭代结果:\n');disp(x);
[y5,x]=SOR(A,b,1.75,xold,error,exact,superior,tag);
fprintf('松弛因子为1.75的SOR迭代结果:\n');disp(x);
C=compare(A,b,xold,error,exact,superior,20);
fprintf('优化前与优化后19次累计用时（秒）: \n');disp(C);

%绘图
n1=1:length(y1);
n2=1:length(y2);
n3=1:length(y3);
n4=1:length(y4);
```

```
n5=1:length(y5);
semilogy(n1,y1);
text(400,1e-7,'Jacob');hold on
semilogy(n2,y2);
text(203,1e-7,'Gauss Seidel');hold on;
semilogy(n3,y3);
text(78,2e-14,'SOR 1.666');hold on;
semilogy(n4,y4);
text(87,7e-9,'SOR 1.5');
semilogy(n5,y5);
text(85,4e-11,'SOR 1.7');
title '迭代次数与误差的关系';
legend('Jacob','Gauss Seidel','SOR 1.666','SOR 1.5','SOR
   1.7');
%Jacob方法:
function [y,xnew]=Jacob(A,b,xold,error,exact,superior,tag)
   %y是存储误差的数组
    xnew=[0 0 0 0 0 0 0 0 0 0]';
    B1= -0.5 .* eye(size(A)) *(A- 2 .* eye(size(A)));%迭代
       矩阵
    d1= 0.5 .* eye(size(A))*b;
    n=1; y=[];
    while 1
        if n<=superior
            y=[y,norm(xold-exact)];
            if tag==0  %判断是否优化
                xnew=B1*xold+d1;
            else
                for i=1:10
                    if i==1
                        xnew(1)=B1(1,2)*xold(2)+d1(i);
                    elseif i==10
                        xnew(10)=B1(10,9)*xold(9)+d1(i);
                    else
                        xnew(i)=B1(i,i-1)*xold(i-1)+B1(i,i
```

```
                        +1)*xold(i+1)+d1(i);
                    end
                end
            end
            if norm(xnew-exact)<error
                break;
            end
            xold=xnew;
            n=n+1;
        else
            fprintf('迭代次数达到上限\n');
            break;
        end
    end
end
%Gauss-Seidel方法
function [y,xnew]=Gauss_Seidel(A,b,xold,error,exact,
   superior,tag)
    n=0;
    y=[];
    xnew=[0 0 0 0 0 0 0 0 0 0]';
    while 1
        if tag==0
            xnew(1)=(b(1)-A(1,2:10)*xold(2:10))/A(1,1);
            for i=2:10
                if i==10
                    xnew(10)=(b(10)-A(10,1:9)*xnew(1:9))/A
                       (10,10);
                else
                    xnew(i)=(b(i)-A(i,1:i-1) * xnew(1:i-1)
                       -A(i,i+1:10) * xold(i+1:10))/A(i,i)
                       ;
                end
            end
        else %优化模式
```

```
            xnew(1)=(b(1)-A(1,2)*xold(2))/A(1,1);
            for i=2:10
                if i==10
                    xnew(10)=(b(10)-A(10,9)*xnew(9))/A
                        (10,10);
                else
                    xnew(i)=(b(i)-A(i,i-1) * xnew(i-1)-A(i
                        ,i+1) * xold(i+1))/A(i,i);
                end
            end
        end
        n=n+1;
        y=[y,norm(xnew-exact)];
        if norm(xnew-xold)<error
            break;
        end
        xold=xnew;%更新解向量
        if n>superior
            fprintf('迭代次数达到上限\n');
            break;
        end
    end
end
% SOR方法
function [y,xnew]=SOR(A,b,w,xold,error,exact,superior,tag)
    n=0;
    y=[];
    xnew=[0 0 0 0 0 0 0 0 0 0]';
    while 1
        if tag==0
            xnew(1)=(b(1)-A(1,2:10) * xold(2:10))/A(1,1);
            for i=2:10
                xnew(i)=(1-w)*xold(i) + w*(b(i)-A(i,1:i-1)
                    *xnew(1:i-1)-A(i,i+1:10)*xold(i+1:10))/
                    A(i,i);
```

```
            end
        else %优化模式
            xnew(1)=(b(1)-A(1,2) * xold(2))/A(1,1);
            for i=2:9
                xnew(i)=(1-w)*xold(i) + w*(b(i)-A(i,i-1)*
                   xnew(i-1)-A(i,i+1)*xold(i+1))/A(i,i);
            end
            xnew(10)=(b(10)-A(10,9)*xold(9))/A(10,10);
        end
        if norm(xnew-exact)<error
            break;
        end
        n=n+1;
        y=[y,norm(xnew-exact)];
        xold=xnew;
        if n>superior
            fprintf('迭代次数达到上限\n');
            break;
        end
    end
end
function diff=compare(A,b,xold,error,exact,superior,N)
    diff=zeros(3,2);
    J0=[];GS0=[];SOR0=[];
    J1=[];GS1=[];SOR1=[];
    tag=0;%关闭优化
    for i=1:N
        tic;
        Jacob(A,b,xold,error,exact,superior,tag);
        toc;
        J0=[J0,toc];
        tic;
        Gauss_Seidel(A,b,xold,error,exact,superior,tag);
        toc;
        GS0=[GS0,toc];
```

```
        tic;
        SOR(A,b,1.65,xold,error,exact,superior,tag);
        toc;
        SOR0=[SOR0,toc];
    end
    tag=1;%启用优化
    for i=1:N
        tic;
        Jacob(A,b,xold,error,exact,superior,tag);
        toc;
        J1=[J1,toc];
        tic;
        Gauss_Seidel(A,b,xold,error,exact,superior,tag);
        toc;
        GS1=[GS1,toc];
        tic;
        SOR(A,b,1.65,xold,error,exact,superior,tag);
        toc;
        SOR1=[SOR1,toc];
    end
    for i=2:N
        diff(1,1)=diff(1,1)+J0(i);
        diff(1,2)=diff(1,2)+J1(i);
        diff(2,1)=diff(2,1)+GS0(i);
        diff(2,2)=diff(2,2)+GS1(i);
        diff(3,1)=diff(3,1)+SOR0(i);
        diff(3,2)=diff(3,2)+SOR1(i);
    end
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。

(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。

(a)(b) 答：

用 d0 和 d1 分别记录第奇数次和偶数次迭代的误差 $|x_n - x_{n-1}|$。当迭代次数 $n \geq 2$ 时，阶数

$$Order = \begin{cases} \frac{lg(d1)}{lg(d0)} & n\text{为偶数} \\ \frac{lg(d0)}{lg(d1)} & n\text{为奇数} \end{cases}$$

运行完毕后 MATLAB 终端显示如下：

```
Present value: -1.25000000000000
Present value: -0.86923076923077   Order:3.356350
Present value: -0.74580982850218   Order:2.166774
Present value: -0.73221177240886   Order:2.054260
Present value: -0.73205083000245   Order:2.032297
Present value: -0.73205080756888   Order:2.016461
Present value: -0.73205080756888   Order:2.007102
The solution of xl is -0.73205080756888
Present value: 0.99444444444444
Present value: 1.00000011431537   Order:3.282433
Present value: 1.00000000000000   Order:3.078086
Present value: 1.00000000000000   Order:2.229574
The solution of xm is 1.00000000000000
Present value: 2.77777777777778
Present value: 2.73375661375661   Order:2.076412
Present value: 2.73205332173038   Order:2.041313
Present value: 2.73205080757435   Order:2.022460
Present value: 2.73205080756888   Order:2.011147
Present value: 2.73205080756888   Order:1.363259
The solution of xr is 2.73205080756888
```

MATLAB 代码如下：

```
error=1e-15;%误差上限
```

```
%三个初始值
x0=-2;
x1=1.2;
x2=3;

xl=newton(x0,error);
fprintf('The solution of xl is %.14f\n',xl);
xm=newton(x1,error);
fprintf('The solution of xm is %.14f\n',xm);
xr=newton(x2,error);
fprintf('The solution of xr is %.14f\n',xr);

function f=f(x)
    f=x^3-3*x^2+2;
end
%f1(x)是f(x)的导数
function f=f1(x)
    f=3*x^2-6*x;
end
function resolve=newton(x0,error)
    N=100;n=0;resolve=inf;
    while 1
        %pause;
        n=n+1;
        x1=x0-f(x0)/f1(x0);
        if mod(n,2)==1
            d0=abs(x1-x0);
        else
            d1=abs(x1-x0);
        end
        fprintf('Present value: %.14f    ',x1);
        if n<2
            fprintf('\n');
        else
            if mod(n,2)==0
```

```
                order=log10(d1)/log10(d0);
            else
                order=log10(d0)/log10(d1);
            end
            fprintf('Order:%.6f\n',order);
        end
        if abs(x1-x0)<error
            resolve=x1;
            break;
        end
        x0=x1;%迭代
        if n>N %超出迭代次数上限，发生异常
            fprintf('ERROR\n');
            break;
        end
    end
end
```

(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：

在求解 x_m 时出现了收敛速度大于二阶的情况（准确来说是三阶收敛），分析如下：

根据 Taylor 展开得

$$f(\alpha)=f(x_k)+(\alpha-x_k)f'(x_k)+\frac{(\alpha-x_k)^2}{2!}f''(x_k)+\frac{(\alpha-x_k)^3}{3!}f'''(\xi),\xi\in[\alpha,x_k]$$

$\because f(\alpha)=0, f'''(\xi)=6$ 不难得到

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

即

$$x_{k+1}-\alpha=\frac{(\alpha-x_k)^2}{2f'(x_k)}f''(x_k)+\frac{(\alpha-x_k)^3}{f'(x_k)}$$

进一步变换

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

当 $k\to\infty$, $x_k, x_{k+1}\to\alpha$, 此时

$$|x_{k+1}-\alpha| = |x_k-\alpha|^3|\frac{3(x_k-1)}{f'(x_k)(\alpha-x_k)}+\frac{1}{f'(x_k)}|$$

对于 x_m, 此时 $\alpha=1$，将其带入上式得

$$|x_{k+1}-\alpha| = |x_k-\alpha|^3|\frac{2}{f'(x_k)}|, x_k\to\alpha$$

$$\because f^{'}(x) = 3x(x-2)\ \forall x\in[1-0,1+0]\ \ f^{'}(x)\neq 0$$

$$\therefore \exists M>0 \quad s.t. \quad \frac{|x_{k+1}-\alpha|}{|x_k-\alpha|^3} = |\frac{2}{f'(x_k)}| \le M, k\to\infty$$

因此函数在 x_m 处三阶收敛，QED

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：

define $\bar{x}$=$x/mynorm(x,\infty)$　　这里的无穷范数指返回模最大的分量，包括负数

$for \quad i=1:m$　　其中迭代次数 m 要足够大

$x^{new} = A*\bar{x}^{old}$

$\lambda = mynorm(x^{new},\infty)$

$x^{new} = x^{new}/\lambda$

$if \|\bar{x}^{new}-\bar{x}^{old}\| < esp$

$return \quad \lambda, \quad \bar{x}^{new}$

如果未能退出，说明有两个模相等，符号相反的特征值

$v = A*\bar{x}^{new}$

$\lambda_1 = \sqrt{v/\bar{x}^{old}}$

$\lambda_2 = -\lambda_1$

$x_1 = \overline{v + \lambda_1 * x^{new}}$
$x_2 = \overline{v - \lambda_2 * x^{new}}$
$return \quad \lambda_1, \quad x_1, \lambda_2, \quad x_2$

(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。

MATLAB 运行结果如下：(执行了两次，分别输入了 A 和-A)

```
有一个最大的特征值
特征值为：
8.0000000000
特征向量为：
   -0.3103
    1.0000
   -0.7931
    0.1379

有一个最大的特征值
特征值为：
-8.0000000000
特征向量为：
   -0.3103
    1.0000
   -0.7931
    0.1379
```

(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）。

MATLAB 运行结果如下：

```
有两个最大的特征值
特征值为：
5.0000000000
-5.0000000000
特征向量分别为：
    1.0000
   -0.5000
    0.1250
   -0.0000

    1.0000
   -0.3333
    0.1667
   -0.1667
```

(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。

MATLAB 运行结果如下：

```
有一个最大的特征值
特征值为：
0.8545199177-0.6621232653i
特征向量为：
   [0.1969 + 0.1324i,-0.7869 + 0.1151i
  -0.3421 - 0.0554i, 0.2603 - 0.1368i
  -0.1977 - 0.1863i,-0.1713 - 0.2228i
```

```
-0.8247 + 0.0383i,-0.0793 - 0.0777i
-0.1729 + 0.1453i,0.1725 - 0.0574i
 0.0974 - 0.2171i,0.1233 + 0.0479i
 0.1125 + 0.5165i,-0.0626 + 0.0498i
 0.0943 - 0.0268i,0.2234 + 0.1151i
 0.3348 - 0.1773i,-0.4789 + 0.0135i
-0.1881 + 0.5186i,0.0740 - 0.4042i
-0.1298 + 0.0702i,0.0865 + 0.1149i
 0.5883 - 0.2917i,0.4423 + 0.1119i
 0.1763 - 0.0375i,0.1373 + 0.4129i
-0.2508 + 0.0329i,-0.1210 + 0.1348i
-0.1467 + 0.2172i,0.1971 - 0.3420i
-0.7111 - 0.4004i,-0.5149 + 0.1636i
-0.2133 + 0.2146i,0.7213 - 0.3292i
-0.3663 - 0.3189i,0.2086 + 0.4165i
 0.0736 + 0.2742i,-0.8720 + 0.1155i
 0.0517 - 0.3382i,0.0269 + 0.1579i
-0.3324 - 0.2147i,-0.0105 - 0.1938i
-0.7123 + 0.2085i,0.3705 - 0.1688i
 0.1169 - 0.1574i,-0.2081 - 0.1869i
-0.0899 + 0.3448i,0.2137 + 0.2433i
-0.4132 + 0.4088i,-0.0393 - 0.2717i
 0.4559 + 0.0441i,-0.0498 + 0.0053i
 0.4925 + 0.0276i,-0.0543 - 0.1920i
-0.4258 + 0.0656i,-0.0837 + 0.3223i
 0.3313 + 0.2727i,1.0000 + 0.0000i
-0.3022 - 0.1933i,0.3136 + 0.1990i
-0.3774 - 0.7111i,-0.2590 - 0.3040i
 0.5435 + 0.1834i,0.4766 + 0.2055i
 0.4003 + 0.4675i,-0.0871 - 0.0173i
-0.0321 - 0.3802i,-0.4076 - 0.1420i
-0.5132 + 0.1043i,-0.1684 - 0.3062i
 0.3777 - 0.3672i,-0.2815 + 0.5662i
 0.4999 + 0.3222i,0.2723 + 0.2813i
-0.7922 + 0.3999i,-0.0495 - 0.3059i
```

```
   0.0695 + 0.0568i,-0.3177 - 0.3403i
  -0.2887 - 0.1428i,0.1954 + 0.2970i
   0.2954 + 0.0078i,0.2988 - 0.1508i
   0.0591 - 0.1484i,-0.1284 - 0.0206i
  -0.0173 + 0.2973i,-0.0665 - 0.0781i
   0.3642 - 0.4142i,-0.3888 + 0.2431i
   0.2920 - 0.1868i,0.1720 + 0.3333i
   0.1823 - 0.3432i,0.2221 + 0.0389i
  -0.0026 - 0.1074i-0.1693 - 0.0071i
   0.3975 - 0.1957i,-0.5745 - 0.2636i
   0.0578 - 0.1252i,0.3992 + 0.2100i
   0.9710 + 0.0458i,-0.1525 - 0.0026i]
```

本题的全部的 MATLAB 代码如下：

```
error=1e-15; %误差上限
x0=[1 0 0 0]';%x0是初始特征向量
%第二问
A=[-148 -105 -83 -67;
    488 343 269 216;
    -382 -268 -210 -170;
    50 38 32 29];
[x1,x2,m1,m2,tag]=eigen(A,x0,error);
show(x1,x2,m1,m2,tag);
[x1,x2,m1,m2,tag]=eigen(-A,x0,error);%测试-A的情况
show(x1,x2,m1,m2,tag);
%第三问
A=[222 580 584 786;
    -82 -211 -208 -288;
    37 98 101 132;
    -30 -82 -88 -109];
[x1,x2,m1,m2,tag]=eigen(A,x0,error);
show(x1,x2,m1,m2,tag);
% 第四问
rng(2);
A=rand(100,100);
x0=ones(100,1);%重新选择初始向量
```

```
[x1,x2,m1,m2,tag]=ipow(A,x0,error,0.8-0.6i);%反幂法
show(x1,x2,m1,m2,tag);

function [x1,x2,m1,m2,tag]=eigen(A,x0,error)%求模最大的特
   征值与特征向量
    m1=0; m2=0;n=0;
    x1=xbar(x0);%无穷范数规范化
    tag=1; %tag=1表示是否有两个模相等大小相反的特征值，默
       认为1
    for i=1:1000
        n=n+1;
        x2=A*x1;
        if norm(xbar(x2)-x1)<error
            m1=myinf(x2);
            m2=m1;
            tag=0;
            break;
        end
        x1=xbar(x2);
    end
    if tag==0
        x1=xbar(x1);
        x2=xbar(x2);
        return;
    end
    x3=A*x2;  %运行到这里说明确实有模一样大的特征值
    m1=sqrt(myinf(x3)/myinf(x1));
    m2=-m1;
    t=x3+m1*x2;
    x2=x3-m1*x2;
    x1=t;
    % 无穷范数规范化
    x1=xbar(x1);
    x2=xbar(x2);
end
```

```
function a=myinf(x) %得到向量x的“无穷范数”（这里规定的无穷范数可以是复数和负数）
    [~,l]=max(abs(x));
    a=x(l);
end
function x=xbar(x) %将向量x规范化（强制要求最大的分量变为1）
    x=x/myinf(x);
end
function [x1,x2,m1,m2,tag]=ipow(A,x0,error,d) %反幂法，d是平移距离
    A = A-d*eye(size(A)); %平移变换
    [x1,x2,m1,m2,tag]=LUeigen(A,x0,error);%利用LU分解迭代
    if tag==0  %将特征值还原
        m1=d+1/m1;
        m2=m1;
    else
        m1=d+1/m1;
        m2=d+1/m2;
    end
end
function show(x1,x2,m1,m2,tag)      %将特征值和特征向量显示出来
    if tag==0
        fprintf('有一个最大的特征值\n');
        fprintf('特征值为:\n');
        printev(m1);
        fprintf('特征向量为: \n');
        disp(x1);
    else
        fprintf('有两个最大的特征值\n');
        fprintf('特征值为:\n');
        printev(m1);
        printev(m2);
        fprintf('特征向量分别为: \n');
```

```
        disp(x1);
        disp(x2);
    end
end
function printev(m)%显示特征值，包括特征值为复数的情况
    if imag(m)==0
        fprintf('%.10f\n',m);
    elseif imag(m)>0
        fprintf('%.10f+%.10fi\n',m,imag(m));
    else
        fprintf('%.10f%.10fi\n',m,imag(m));
    end
end
function [L,U]=myLU(A)%进行LU分解
    n=size(A);n=n(1); %n代表矩阵A的阶数
    L=ones(n,n);      %初始化下三角阵
    for j=1:n-1       %LU分解定义，这里A是形参
        for i=j+1:n
            m=A(i,j)/A(j,j);
            L(i,j)=m;
            for k=j:n
                A(i,k)=A(i,k)-m*A(j,k);
            end
        end
    end
    U=A;
end
function [x1,x2,m1,m2,tag]=LUeigen(A,x0,error)%与函数eigen
   几乎没变，只是迭代部分用LU方法了
    m1=0; m2=0;n=0;
    x1=xbar(x0);%规范化
    tag=1;
    [L,U]=myLU(A);   %LU分解
    for i=1:1000
        n=n+1;
```

```
        x2=iteration(L,U,x1);
        if norm(xbar(x2)-x1)<100*error %此时只有一个模最大
           的特征值;多次测试发现精度只能达到1e-13
            m1=myinf(x2);
            m2=m1;
            tag=0;
            break;
        end
        x1=xbar(x2);
    end
    if tag==0
        x1=xbar(x1);
        x2=xbar(x2);
        return;
    end
    fprintf('!\n');%迭代次数超过上限，有精确度不足的风险
    %有不至一个模最大的特征值
    x3=iteration(L,U,x2);
    m1=sqrt(myinf(x3)/myinf(x1));%获得特征值
    m2=-m1;
    t=x3+m1*x2;
    x2=x3-m1*x2;
    x1=t;
    %无穷范数规范化
    x1=xbar(x1);
    x2=xbar(x2);
end
function x=iteration(L,U,x0)  %利用LU分解迭代，x= A^-1 *
   x0
    y=x0;n=length(x0);
    y(1)=x0(1);
    for i=2:n
        for j=1:i-1
            x0(i)=x0(i)-L(i,j)*y(j);
        end
```

```
        y(i)=x0(i);
    end
    x(n)=y(n)/U(n,n);
    for i=(n-1):-1:1
        for j=n:-1:i+1
            y(i)=y(i)-U(i,j)*x(j);
        end
        x(i)=y(i)/U(i,i);
    end
    x=x-2i.*imag(x);
    x=x';
end
```