作业一

姓名　　　　　　学号　　　　　　日期

第一题 (a)(b)

Jacobi 迭代矩阵：

$$x^{(k+1)} = D^{-1}(D-A)x^{(k)} + D^{-1}b; \tag{1}$$

gauss_seidel 迭代矩阵：

$$x^{(k+1)} = -D + L^{-1}Ux^{(k)} + D + L^{-1}b; \tag{2}$$

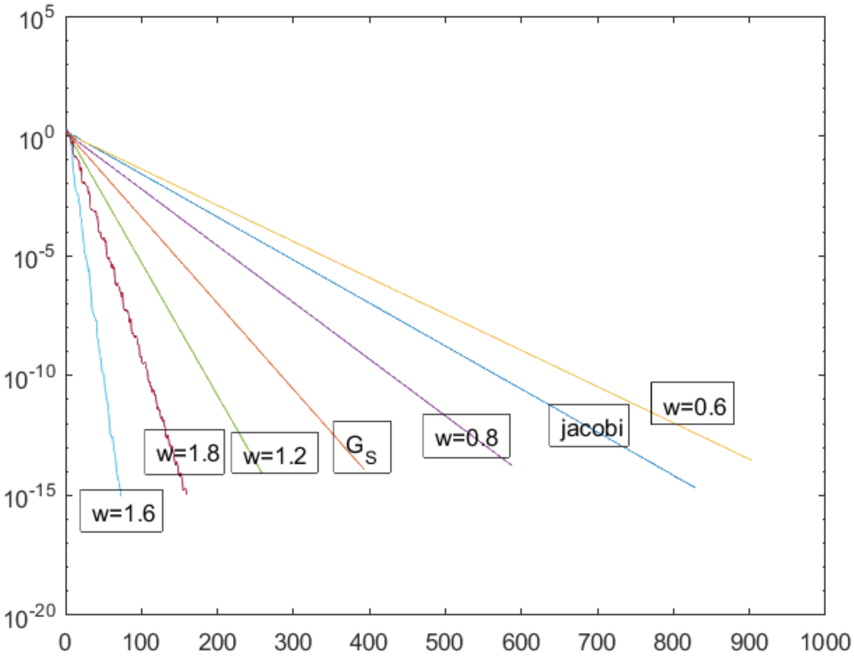
SoR 迭代矩阵：

$$x^{(k+1)} = (I + wD^{-1}L)^{-1}((1-w)I - wD^{-1}U)x^{(k)} + w(I + wD^{-1}L)^{-1}D^{-1}b; \tag{3}$$

利用上述迭代矩阵求解问题，当误差（迭代解与精确解之差的 ∞-范数）满足精度 1^{-15} 时迭代结束，同时记录误差大小-迭代次数函数。

利用精确解将误差大小和迭代次数的关系用 semilogy 图表示出来，可以尝试得出对应于 10—15 的误差目标 w 值在 1.6 左右时收敛速度最快。

（横轴为迭代次数 n，纵轴为迭代解与精确解的差距）



Matlab 程序显示如下:

```
clear, clc;
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];
D=diag(diag(A));
L=tril(A, -1);
U=triu(A, 1);
I=eye(10);
x_real=[1; 0; 1; 0; 0; 0; 0; -1; 0; -1];

%jacobi
```

```
x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
n=0;%迭代次数
while 1
    x1=D\(D-A)*x0+D\b;
    if norm(x1-x0, inf)>1e-15
        x0=x1;
        n=n+1;
        r1(n)=norm(x1-x_real, inf);
    else
        break
    end
end
semilogy(r1)
hold on

%gauss_seidel
x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
n=0;%迭代次数
while 1
    x1=-(D+L)\U*x0+(D+L)\b;
    if norm(x1-x0, inf)>1e-15
        x0=x1;
        n=n+1;
        r2(n)=norm(x1-x_real, inf);
    else
        break
    end
end
semilogy(r2);
hold on

%SoR
SoR(0.6, I, D, L, U, b, x_real);
```

```
SoR(0.8, I, D, L, U, b, x_real);
SoR(1.2, I, D, L, U, b, x_real);
SoR(1.6, I, D, L, U, b, x_real);%接近最优
SoR(1.8, I, D, L, U, b, x_real);
function SoR(w, I, D, L, U, b, x_real)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1=(I+D\L*w)\((1-w)*I-D\U*w)*x0+(I+w*inv(D)*L)\(D\b)*w;
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
            r3(n)=norm(x1-x_real, inf);
        else
            break
        end
    end
    semilogy(r3)
    hold on
end
```

(c)

利用稀疏矩阵的特性，可以先 L-U 分解然后再进行对应矩阵迭代运算。

而本题是一个较为简单的矩阵，我的程序直接在迭代方程中忽略含矩阵中 0 元素的项，即得到了加速算法。各种方法改进前后的 10 次运行时间之和对比如下表 (单位：秒)：

迭代方法	总运行时间/s	
	改进前	改进后
jacobi	0.253018	0.009805
gauss_seidel	0.077205	0.006162
SoR w=0.6	0.624484	0.009901
SoR w=0.8	0.408401	0.007578
SoR w=1.2	0.189481	0.005008
SoR w=1.6	0.051715	0.000912
SoR w=1.8	0.130205	0.002243

表 1: 各种迭代方法改进前后总运行时间对比

MATLAB 程序显示如下：

```
clear, clc;
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];
D=diag(diag(A));
L=tril(A, -1);
U=triu(A, 1);
I=eye(10);
x_real=[1; 0; 1; 0; 0; 0; 0; -1; 0; -1];
```

```
fprintf('%s\n', 'jacobi:')
jacobi(D, A, b);
i=1;
tic
while i<=10
    jacobi(D, A, b);
    i=i+1;
end
toc
jacobi_new(A, b);
i=1;
tic
while i<=10
    jacobi_new(A, b);
    i=i+1;
end
toc

fprintf('%s\n', 'gauss_seidel:')
gauss_seidel(D, L, U, b);
i=1;
tic
while i<=10
    gauss_seidel(D, L, U, b);
    i=i+1;
end
toc
gauss_seidel_new(A, b);
i=1;
tic
while i<=10
    gauss_seidel_new(A, b);
    i=i+1;
end
```

```
toc

fprintf('%s\n', 'SoRw=0.6:')
SoR(0.6, I, D, L, U, b);
i=1;
tic
while i<=10
    SoR(0.6, I, D, L, U, b);
    i=i+1;
end
toc
SoR_new(0.6, A, b);
i=1;
tic
while i<=10
    SoR_new(0.6, A, b);
    i=i+1;
end
toc

fprintf('%s\n', 'SoRw=0.8:')
SoR(0.8, I, D, L, U, b);
i=1;
tic
while i<=10
    SoR(0.8, I, D, L, U, b);
    i=i+1;
end
toc
SoR_new(0.8, A, b);
i=1;
tic
while i<=10
    SoR_new(0.8, A, b);
    i=i+1;
```

```
end
toc

fprintf('%s\n', 'SoRw=1.2:')
SoR(1.2, I, D, L, U, b);
i=1;
tic
while i<=10
    SoR(1.2, I, D, L, U, b);
    i=i+1;
end
toc
SoR_new(1.2, A, b);
i=1;
tic
while i<=10
    SoR_new(1.2, A, b);
    i=i+1;
end
toc

fprintf('%s\n', 'SoRw=1.6:')
SoR(1.6, I, D, L, U, b);
i=1;
tic
while i<=10
    SoR(1.6, I, D, L, U, b);
    i=i+1;
end
toc
SoR_new(1.6, A, b);
i=1;
tic
while i<=10
    SoR_new(1.6, A, b);
```

```
    i=i+1;
end
toc

fprintf('%s\n', 'SoRw=1.8:')
SoR(1.8, I, D, L, U, b);
i=1;
tic
while i<=10
    SoR(1.8, I, D, L, U, b);
    i=i+1;
end
toc
SoR_new(1.8, A, b);
i=1;
tic
while i<=10
    SoR_new(1.8, A, b);
    i=i+1;
end
toc

%jacobi
function jacobi(D, A, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1=D\(D-A)*x0+D\b;
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
```

```
    end
end
%jacobi改
function jacobi_new(A, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1(1)=-A(1,2)/A(1,1)*x0(2)+b(1)/A(1,1);
        x1(2)=-A(2,1)/A(2,2)*x0(1)-A(2,3)/A(2,2)*
        x0(3)+b(2)/A(2,2);
        x1(3)=-A(3,2)/A(3,3)*x0(2)-A(3,4)/A(3,3)*
        x0(4)+b(3)/A(3,3);
        x1(4)=-A(4,3)/A(4,4)*x0(3)-A(4,5)/A(4,4)*
        x0(5)+b(4)/A(4,4);
        x1(5)=-A(5,4)/A(5,5)*x0(4)-A(5,6)/A(5,5)*
        x0(6)+b(5)/A(5,5);
        x1(6)=-A(6,5)/A(6,6)*x0(5)-A(6,7)/A(6,6)*
        x0(7)+b(6)/A(6,6);
        x1(7)=-A(7,6)/A(7,7)*x0(6)-A(7,8)/A(7,7)*
        x0(8)+b(7)/A(7,7);
        x1(8)=-A(8,7)/A(8,8)*x0(7)-A(8,9)/A(8,8)*
        x0(9)+b(8)/A(8,8);
        x1(9)=-A(9,8)/A(9,9)*x0(8)-A(9,10)/A(9,9)*
        x0(10)+b(9)/A(9,9);
        x1(10)=-A(10,9)/A(10,10)*x0(9)+b(10)/A(10,10);
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
    end
end
%gauss_seidel
```

```
function gauss_seidel(D, L, U, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1=-(D+L)\U*x0+(D+L)\b;
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
    end
end
%gauss_seidel改
function gauss_seidel_new(A, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1(1)=-A(1,2)/A(1,1)*x0(2)+b(1)/A(1,1);
        x1(2)=-A(2,1)/A(2,2)*x1(1)-A(2,3)/A(2,2)*
        x0(3)+b(2)/A(2,2);
        x1(3)=-A(3,2)/A(3,3)*x1(2)-A(3,4)/A(3,3)*
        x0(4)+b(3)/A(3,3);
        x1(4)=-A(4,3)/A(4,4)*x1(3)-A(4,5)/A(4,4)*
        x0(5)+b(4)/A(4,4);
        x1(5)=-A(5,4)/A(5,5)*x1(4)-A(5,6)/A(5,5)*
        x0(6)+b(5)/A(5,5);
        x1(6)=-A(6,5)/A(6,6)*x1(5)-A(6,7)/A(6,6)*
        x0(7)+b(6)/A(6,6);
        x1(7)=-A(7,6)/A(7,7)*x1(6)-A(7,8)/A(7,7)*
        x0(8)+b(7)/A(7,7);
        x1(8)=-A(8,7)/A(8,8)*x1(7)-A(8,9)/A(8,8)*
        x0(9)+b(8)/A(8,8);
```

```
        x1(9)=-A(9,8)/A(9,9)*x1(8)-A(9,10)/A(9,9)*
        x0(10)+b(9)/A(9,9);
        x1(10)=-A(10,9)/A(10,10)*x1(9)+b(10)/A(10,10);
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
    end
end
%SoR
function SoR(w, I, D, L, U, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1=(I+D\L*w)\((1-w)*I-D\U*w)*x0+(I+w*inv(D)*L)\(D\b)*w;
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
    end
end
%SoR改
function SoR_new(w, A, b)
    x0=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    x1=[1; 1; 1; 1; 1; 1; 1; 1; 1; 1];
    n=0;%迭代次数
    while 1
        x1(1)=(1-w)*x0(1)+w*(-A(1,2)/A(1,1)*
        x0(2)+b(1)/A(1,1));
        x1(2)=(1-w)*x0(2)+w*(-A(2,1)/A(2,2)*
```

```
        x1(1)-A(2,3)/A(2,2)*x0(3)+b(2)/A(2,2));
        x1(3)=(1-w)*x0(3)+w*(-A(3,2)/A(3,3)*
        x1(2)-A(3,4)/A(3,3)*x0(4)+b(3)/A(3,3));
        x1(4)=(1-w)*x0(4)+w*(-A(4,3)/A(4,4)*
        x1(3)-A(4,5)/A(4,4)*x0(5)+b(4)/A(4,4));
        x1(5)=(1-w)*x0(5)+w*(-A(5,4)/A(5,5)*
        x1(4)-A(5,6)/A(5,5)*x0(6)+b(5)/A(5,5));
        x1(6)=(1-w)*x0(6)+w*(-A(6,5)/A(6,6)*
        x1(5)-A(6,7)/A(6,6)*x0(7)+b(6)/A(6,6));
        x1(7)=(1-w)*x0(7)+w*(-A(7,6)/A(7,7)*
        x1(6)-A(7,8)/A(7,7)*x0(8)+b(7)/A(7,7));
        x1(8)=(1-w)*x0(8)+w*(-A(8,7)/A(8,8)*
        x1(7)-A(8,9)/A(8,8)*x0(9)+b(8)/A(8,8));
        x1(9)=(1-w)*x0(9)+w*(-A(9,8)/A(9,9)*
        x1(8)-A(9,10)/A(9,9)*x0(10)+b(9)/A(9,9));
        x1(10)=(1-w)*x0(10)+w*(-A(10,9)/A(10,10)*
        x1(9)+b(10)/A(10,10));
        if norm(x1-x0, inf)>1e-15
            x0=x1;
            n=n+1;
        else
            break
        end
    end
end
```

第二题 (a)(b)

利用牛顿迭代法求解方程的根，设置精度为 1^{-15}：

$$x_{k+1} = x_k - f(x_k)/h(x_k); \tag{4}$$

x_l 的每次迭代结果依次为-0.8000、-0.7357、-0.7321、-0.7321、-0.7321、-0.7321

x_m 的每次迭代结果依次为 0.5820、1.0590、0.9999、1.0000、1.0000、1.0000

x_r 的每次迭代结果依次为 2.8000、2.7357、2.7321 、2.7321 、2.7321 、2.7321

用最近得到的近似解代替精确解计算收敛阶数：

$$a = log\mid (x_{k+1} - x_k)/(x_k - x_{k-1}) \mid /log\mid (x_k - x_{k-1})/(x_{k-1} - x_{k-2}) \mid; \tag{5}$$

x_l 附近的大概收敛阶数依次为 2.006548、2.000241、Inf

x_m 附近的大概收敛阶数依次为 2.905021、2.999420、Inf

x_r 附近的大概收敛阶数依次为 2.006548、2.000241、1.083436

MATLAB 程序显示如下：

```
clear, clc;
f=@(x) x^3 - 3*x^2 + 2;
h=@(x) 3*x^2 - 6*x;%f(x)一阶导

newton(-0.5, f, h);
newton(1.69, f, h);
newton(2.5, f, h);
function newton(x0, f, h)
    n=1;%迭代次数
    while 1
        x1=x0-f(x0)/h(x0);
        if abs(x0-x1)>1e-15
            x0=x1;
            x(n)=x1;%记录每次迭代的值
        else
            x(n)=x1;%记录每次迭代的值
            break
        end
        if n>3%计算收敛阶数
```

```
            a=log(abs((x(n)-x(n-1))/(x(n-1)-x(n-2))))
            /log(abs((x(n-1)-x(n-2))/(x(n-2)-x(n-3))));
            fprintf('%f  ', a);
        end
        n=n+1;
    end
    if n>3%计算收敛阶数
        a=log(abs((x(n)-x(n-1))/(x(n-1)-x(n-2))))
        /log(abs((x(n-1)-x(n-2))/(x(n-2)-x(n-3))));
        fprintf('%f  ', a);
    end
    x
end
```

(c)

观测到了比二阶收敛更快的现象

在 x_l、x_r 附近大概收敛阶数趋于 2，到了精度特别高的时候，因为 matlab 存在最小精度单位产生了很大误差导致结果为 Inf。

而在 x_m 附近即使在精度允许的情况下，大概收敛阶数也是趋于 3 的，比二阶收敛更快。这可能是因为在 x_m 附近时，近似解的每次迭代结果都是在精确解两侧跳动的，这种情况导致了计算的大概收敛阶数偏高。

第三题 (a)

幂法的伪代码显示如下：

迭代初始向量 $X^{(0)}$=(1, 1, ...)T， $Y^{(0)}=X^{(0)}/||X^{(0)}||\infty$

for k=1：n

$X^{(k)}=A\times Y^{(k-1)}$

$\lambda=||X^{(k)}||\infty$

$Y^{(k)}=X^{(k)}/\lambda$

$if\ Y^{(k)}-Y^{(k-1)}<\varepsilon$

return $\lambda, Y^{(k)}$

$elseif\ Y^{(k)}+Y^{(k-1)}<\varepsilon$

return $-\lambda, Y^{(k)}$

else

$X^{(k+1)}=A\times X^{(k)}$

$\lambda 1=\sqrt{x_1^{(k+1)}/y_1^{(k-1)}},\ \lambda_2=-\lambda_1$

$V_1=X^{(k+1)}+\lambda_1 X^{(k)},\ V_2=X^{(k+1)}-\lambda_1 X^{(k)}$

return $\lambda_1,\lambda_2,V_1,V_2$

(b)

记该矩阵为 A，利用幂法迭代结果为：

$\lambda=8,\ \ V=(-0.3103,1.0000,-0.7931,0.1379)$

其中对于-A，利用幂法迭代结果为：

$\lambda=-8,\ \ V=(0.3103,-1.0000,0.7931,-0.1379)$

(c)

记该矩阵为 B，利用幂法迭代结果为：

$\lambda_1=5,\ \ \lambda_2=-5,$

$V_1=(1.0000,-0.5000,0.1250,0.0000)\ \ V_2=(-1.0000,0.3333,-0.1667,0.1667)$

(b)(c) 使用同一个 matlab 程序

Matlab 程序显示如下：

```
clear, clc;
A=[-148, -105, -83, -67;     %初始化变量
488, 343, 269, 216;
-382, -268, -210, -170;
50, 38, 32, 29];
B=[222, 580, 584, 786;
-82, -211, -208, -288;
37, 98, 101, 132;
-30, -82, -88, -109];
X0=[1; 1; 1; 1];
X1=[0; 0; 0; 0];
X2=[0; 0; 0; 0];

mifa(A, X0, X1, X2);
mifa(-A, X0, X1, X2);
mifa(B, X0, X1, X2);
function mifa(A, X0, X1, X2)
    X0_=X0; X1_=X1; X2_=X2;
    while norm(X2_-X0_, inf)>1e-15    %利用幂法迭代
        X0=A*X0_;
        X0_=X0/norm(X0, inf);
        X1=A*X0_;
        X1_=X1/norm(X1, inf);
        X2=A*X1_;
        X2_=X2/norm(X2, inf);
    end
    if norm(X2_-X1_, inf)<1e-12       %第一种情况
        la1=max(X2)
        X1_
    else if norm(X2_+X1_, inf)<1e-12  %第二种情况
        la1=-norm(X2, inf)
        X1_
        else                          %第三种情况
        X3=A*X2;
        la1=(X3(1)/X1_(1))^0.5
```

```
        la2=-la1
        V1=X3+la1*X2; V1=V1/norm(V1, inf)
        V2=X3-la1*X2; V2=V2/norm(V2, inf)
        end
    end
end
```

(d)

记该矩阵为 A，记 p=0.8-0.6i。根据 p 对 A 进行位移，然后就可以利用反幂法求解 A 最接近 p 的特征值：

$$\overline{A} = A - pI,\ \lambda = p + \frac{1}{\overline{\lambda}} \tag{6}$$

迭代结果依次为：

λ =2.86562710140207-0.74620132610960i　迭代次数：1

λ =0.68343027770884-0.49314325448281i　迭代次数：2

λ =0.85337882460664-0.66347197870053i　迭代次数：3

λ =0.85360308606316-0.66436884055531i　迭代次数：4

λ =0.85467751632035-0.66176160522397i　迭代次数：5

λ =0.85451992565432-0.66218158309518i　迭代次数：6

λ =0.85451358338547-0.66212181498287i　迭代次数：7

λ =0.85452034085363-0.66212081505451i　迭代次数：8

λ =0.85452031280039-0.66212381926834i　迭代次数：9

λ =0.85451980182358-0.66212324755012i　迭代次数：10

λ =0.85451992616717-0.66212325375542i　迭代次数：11

λ =0.85451991955634-0.66212326666457i　迭代次数：12

λ =0.85451991725654-0.66212326560192i　迭代次数：13

λ =0.85451991767500-0.66212326529040i　迭代次数：14

λ =0.85451991767231-0.66212326534597i　迭代次数：15

λ =0.85451991767236-0.66212326534912i　迭代次数：16

λ =0.85451991767024-0.66212326534860i　迭代次数：17

λ =0.85451991767049-0.66212326534821i　迭代次数：18

λ =0.85451991767058-0.66212326534827i　迭代次数：19

λ =0.85451991767056-0.66212326534829i　迭代次数：20

λ =0.85451991767056-0.66212326534828i　迭代次数：21

λ =0.85451991767056-0.66212326534828i　迭代次数：22

λ =0.85451991767056-0.66212326534828i　迭代次数：23

MATLAB 程序显示如下：

```
clear, clc;
s=rng(2);
A=rand(100, 100);
X0=ones(100, 1);
X1=zeros(100, 1);
I=eye(100);
p=0.8-0.6i;

newmifa(A, X0, X1, I, p);
function newmifa(A, X0, X1, I, p)
    X0_=X0;
    n=1; %迭代次数
    while abs(max(X1-X0_))>1e-14
    %利用反幂法针对给定的位移值迭代
        X1=X0_;
        X0=(A-p*I)\X0_;
        X0_=X0/max(X0);
        la1=p+1/max(X0);
        fprintf('%1.14f%.14fi  迭代次数：%d\n\n',
        real(la1), imag(la1), n)
        n=n+1;
    end
end
```