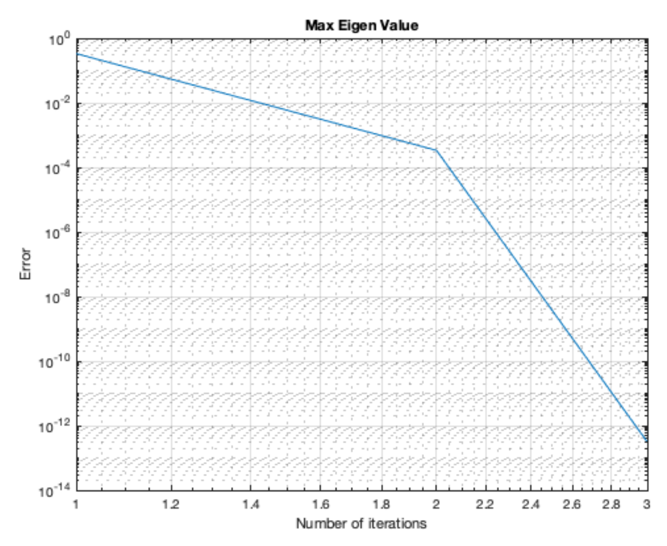
Contents

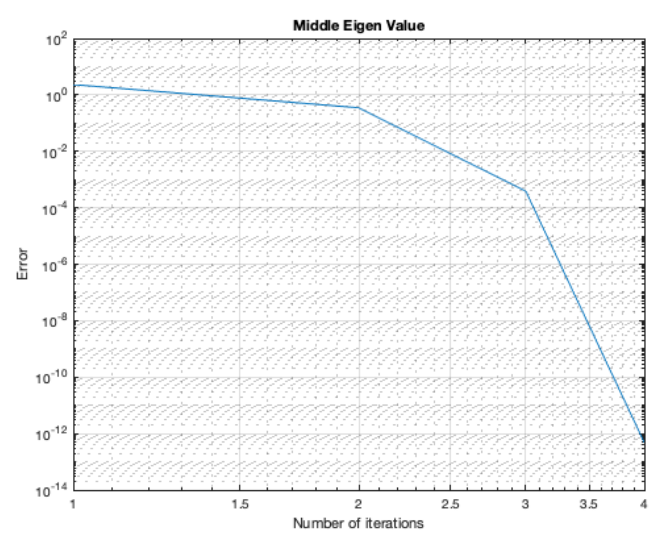

```
clc
clear all
```

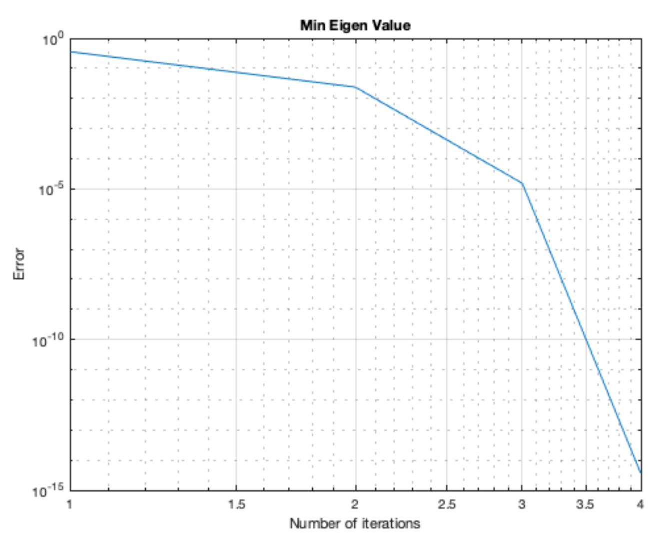
Problem 4

```
A = [4,3,4;3,1,3;4,3,4];

v0 = zeros(size(A,1),1);

v0(1,1) = 1/2;
v0(2,1) = 1/2;
v0(3,1) = 1/2;

% Max Guess
v0 = v0/norm(v0);
v0_max = v0

lambda0 = v0'*A*v0;

Lambda(1,1) = lambda0;
V(:,1) = v0;

del = 1.0;
tol = 0.000001;

I = zeros(size(A));

I(1,1) = 1;
I(2,2) = 1;
I(3,3) = 1;

lambda = lambda0;
v = v0;

a = 0;
while del > tol

    a = a + 1;
    lambda1 = lambda;
    U = A - lambda*I;
    v = U\v;
    v = v/norm(v);
    lambda = v'*A*v;
    del = abs(lambda - lambda1);
    error(a,1) = del;

end

k = zeros(a,1);

for i = 1:a
   k(i,1) = i;
end

figure(1)
loglog(k,error)
grid on
xlabel('Number of iterations')
ylabel('Error')
title('Max Eigen Value')


v0 = zeros(size(A,1),1);

v0(1,1) = 1/2;
v0(2,1) = -1/2;
v0(3,1) = 1/2;

% Middle Guess
v0 = v0/norm(v0);
v0_middle = v0

lambda0 = v0'*A*v0;

Lambda(1,1) = lambda0;
V(:,1) = v0;

del = 1.0;
tol = 0.000001;

I = zeros(size(A));

I(1,1) = 1;
I(2,2) = 1;
I(3,3) = 1;

lambda = lambda0;
v = v0;

a = 0;
while del > tol

    a = a + 1;
    lambda1 = lambda;
    U = A - lambda*I;
    v = U\v;
    v = v/norm(v);
    lambda = v'*A*v;
    del = abs(lambda - lambda1);
    error(a,1) = del;

end

k = zeros(a,1);

for i = 1:a
   k(i,1) = i;
end

figure(2)
loglog(k,error)
grid on
xlabel('Number of iterations')
ylabel('Error')
title('Middle Eigen Value')


v0 = zeros(size(A,1),1);

v0(1,1) = 1/2;
v0(2,1) = -1/2;
v0(3,1) = -1/2;

% Min Guess
v0 = v0/norm(v0);
v0_min = v0

lambda0 = v0'*A*v0;

Lambda(1,1) = lambda0;
V(:,1) = v0;

del = 1.0;
tol = 0.000001;

I = zeros(size(A));

I(1,1) = 1;
I(2,2) = 1;
I(3,3) = 1;

lambda = lambda0;
v = v0;

a = 0;
while del > tol

    a = a + 1;
    lambda1 = lambda;
    U = A - lambda*I;
    v = U\v;
    v = v/norm(v);
    lambda = v'*A*v;
    del = abs(lambda - lambda1);
    error(a,1) = del;

end

k = zeros(a,1);

for i = 1:a
   k(i,1) = i;
end

figure(3)
loglog(k,error)
grid on
xlabel('Number of iterations')
ylabel('Error')
title('Min Eigen Value')



% The best and most robust way is to use the Gram-Schmidt process to
% orthogonalize the vectors at each step. However for this prblems size it
% was easiest just to start here.
```

```
v0_max =

    0.5774
    0.5774
    0.5774


v0_middle =

    0.5774
   -0.5774
    0.5774


v0_min =

    0.5774
   -0.5774
   -0.5774
```








```
D = azelaxes(10,10);
E = azelaxes(20,20);
F = azelaxes(30,30);
G = azelaxes(40,40);
H = azelaxes(50,50);


for i = 1:3
   r1 = D(:,i)'*A*D(:,i);
   r1 = r1/(D(:,i)'*D(:,i));
   W(i) = r1;
end

for i = 1:3
   r1 = E(:,i)'*A*E(:,i);
   r1 = r1/(E(:,i)'*E(:,i));
   W(i+3) = r1;
end

for i = 1:3
   r1 = F(:,i)'*A*F(:,i);
   r1 = r1/(F(:,i)'*F(:,i));
   W(i+2*3) = r1;
end

for i = 1:3
   r1 = G(:,i)'*A*G(:,i);
   r1 = r1/(G(:,i)'*G(:,i));
   W(i+3*3) = r1;
end

for i = 1:3
   r1 = H(:,i)'*A*H(:,i);
   r1 = r1/(H(:,i)'*H(:,i));
   W(i+4*3) = r1;
end

% The r values for the different unit vectors on the unit sphere I cannat
% get the mesh to work at all.
```

Problem 5

```
A = [4,3,4;3,1,3;4,3,4];

b = eig(A);

B = zeros(3,3);

B(1,1) = max(b);

B(3,3) = b(2,1);

B(2,2) = min(b);



AT(:,:,1) = A;

fac = 1000;


kmax = size(A,1)*fac; %Arbitrarily large kmax >> n where A is nxn symmetric matrix

for i = 2:kmax

    [Q,R] = qr(AT(:,:,i-1));

    AT(:,:,i) = R*Q;

end

Error = zeros(3,3,kmax);


for i = 2:kmax

    Error(:,:,i) = abs(AT(:,:,i) - AT(:,:,i-1));


end

Error = abs(Error);

total_error = zeros(kmax,1);


for i = 1:kmax

    total_error(i,1) = sum(diag(Error(:,:,i)));

end

k = zeros(kmax,1);

for i = 1:kmax
   k(i,1) = i;
end


figure(4)
loglog(k,total_error)
grid on
xlabel('Iteration Step (logk)')
ylabel('Summed Errors for Iterations log')
title('Problem 5')
```