Question 1 (Repeat example 35.1)

m = 200; A = 2\*eye(m) + 0.5\*randn(m)/sqrt(m);

b = ones(m,1);

[V,~] = eig(A);

plot(V,'k.')

tol = 10^(-12);

maxit = 20;

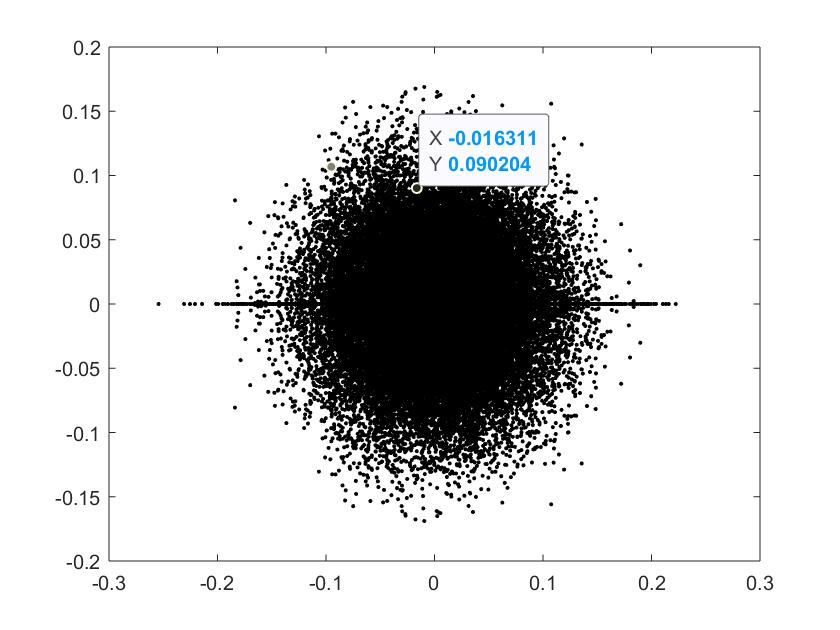
[x, flag, relres, iter, resvec] = gmres(A,b,[],tol,maxit);

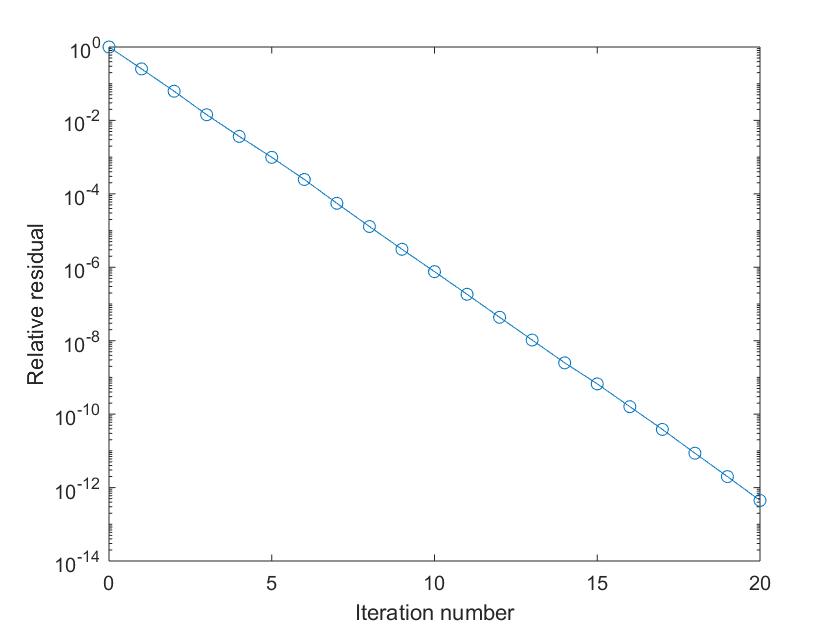
figure

semilogy(0:maxit, resvec/norm(b), '-o');

xlabel('Iteration number');

ylabel('Relative residual');





Question 2

(a)

A = [8 4 2 1; 4 8 4 2; 2 4 8 4; 1 2 4 8];

v = [1 1 1 1]'./2;

n = 10;

lambda1 = []; lambda2 = []; lambda3 = [];

format long

for k = 1:n

w = A \* v;

v = w / norm(w);

lambda1 = [lambda1 v'\*A\*v];

end

v = [1 1 1 1]'./2;

for k = 1:n

w = A \ v;

v = w / norm(w);

lambda2 = [lambda2 v'\*A\*v];

end

v = [1 1 1 1]'./2;

lambda3 = [v'\*A\*v];

for k = 1:n

w = (A - lambda3(k)\*eye(4)) \ v;

v = w / norm(w);

lambda3 = [lambda3 v'\*A\*v];

end

lambda1

lambda2

lambda3

The answer is as follows:

Note that for both power method and Rayleigh quotient iteration, they compute the eigenvalue with the largest absolute value. Besides, Rayleigh quotient converges faster than power method. And for unshifted inverse iteration, it computes the eigenvalue with the smallest absolute value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Power method | 16.672131147540988 | 16.683819628647207 | 16.684602322430148 | 16.684654684513276 | 16.684658187307214 |
| 16.684658421627784 | 16.684658437302733 | 16.684658438351320 | 16.684658438421469 | 16.684658438426155 |
| Unshifted inverse iteration | 14.399999999999999 | 7.135135135135133 | 4.554938956714762 | 4.331664534989951 | 4.316434838256360 |
| 4.315414702695185 | 4.315346454398972 | 4.315341888880736 | 4.315341583468831 | 4.315341563038203 |
| Rayleigh quotient | 16.684615384615384 | 16.684658438426489 | 16.684658438426489 | 16.684658438426492 | 16.684658438426489 |
| 16.684658438426492 | 16.684658438426489 | 16.684658438426492 | 16.684658438426489 | 16.684658438426492 |

(b)

A = [8 4 2 1; 4 8 4 2; 2 4 8 4; 1 2 4 8];

v = [1 1 1 1]'./2;

n = 10;

off\_diag\_max = [];

format long

max = 0;

for k = 1:n

[Q,R] = qr(A);

A = R\*Q;

for i = 1:4

for j = 1:4

if (i ~= j & A(i,j) > max)

max = A(i,j);

end

end

end

off\_diag\_max = [off\_diag\_max max];

end

A

off\_diag\_max

For the answer below, note that the off diagonal elements at each step is truly decreasing to zero. And the matrix of A is convergent to somehow tridiagonal matrix.

A=

16.684650870697748 0.008106965374876 0.000018128314709 0.000000256400930

0.008106965374876 7.999920721769696 0.017888947809786 0.000253015404780

0.000018128314708 0.017888947809787 4.312593875744195 0.060995911929790

0.000000256400932 0.000253015404780 0.060995911929791 3.002834531788368

off\_diag\_max =

4.289906995109921 2.659558449077073 1.363329400421118 0.727713493424352 0.394169759053323 0.252862471417951 0.114259089109530 0.125365640161673 0.033176642774905 0.060995911929791