```m
%What I have done here
%Basically Arnoldi process is used to find Km krylov subspace where m is an
%integer
%Here we are using Grahm Shmidtconcept of finding orthonormal vectors

function[V,h,beta]=arnoldi(A,r0,m)
%First we are giving norm value of residue r0 to beta
beta=norm(r0);
n=size(A,1);

% I am preassumin matrix V and matrix h(hessenberg matrix)
V=zeros(n,m+1);
h=zeros(m+1,m);
V(:,1)=r0/beta;
for j=1:m

    % we are using this because krylov subspace is in the form of
    % r,Ar,Ar2,Ar3
    w=A*V(:,j);

    %Grahm shmidt process
    for i=1:j
        h(i,j)=w'*V(:,i);
        w=w-h(i,j)*V(:,i);
    end
    h(j+1,j)=norm(w);

    %Stopping creteria as given in the exam
    if norm(w)<1e-6
        break;
    end

    %We have calculated a new vector which belongs to basis of Km
    V(:,j+1)=w/norm(w);
end

end
```

```
 1 function[]=SelfFOM(A,b)
 2 A = [2 -1 0 0;-1 2 -1 0;0 -1 2 -1;0 0 -1 2];
 3 b = [1;0;0;0];
 4
 5 %I am running this for loop for 4 times because we habe to show for 4
 6 %values of m
 7 for i=1:4
 8
 9 n=size(A,1);
10 x0=zeros(n,1);     %Initial vector as mentioned in exam
11 m=i;
12 r=b;               % Here I am allocating value to r ......r=b-A*x.....But x0=0 ↙
soit will be equal to b
13
14 %Here I am using predefined arnoldi function
15 [V,H,beta]=arnoldi(A,r,m);
16 fprintf('Values for m=%d \n',i);
17     disp("Required H:");
18     H
19     disp("Required V:");
20     V
21     y=H(1:m,:)\(beta*(eye(m,1)));
22
23     x=V(:,1:m)*y;
24     x
25 end
26 end
27
28
29
```

```
>> SelfFOM
Values for m=1
Required H:

H =

     2
     1

Required V:

V =

     1     0
     0    -1
     0     0
     0     0


x =

    0.5000
         0
         0
         0

Values for m=2
Required H:

H =

     2     1
     1     2
     0     1

Required V:

V =

     1     0     0
     0    -1     0
     0     0     1
     0     0     0


x =

    0.6667
    0.3333
```

```
         0
         0

Values for m=3
Required H:

H =

     2     1     0
     1     2     1
     0     1     2
     0     0     1

Required V:

V =

     1     0     0     0
     0    -1     0     0
     0     0     1     0
     0     0     0    -1


x =

    0.7500
    0.5000
    0.2500
         0

Values for m=4
Required H:

H =

     2     1     0     0
     1     2     1     0
     0     1     2     1
     0     0     1     2
     0     0     0     0

Required V:

V =

     1     0     0     0     0
     0    -1     0     0     0
     0     0     1     0     0
     0     0     0    -1     0
```

```
x =

    0.8000
    0.6000
    0.4000
    0.2000

>>
```

```
 1 function[]=SelfCG(A,b)
 2 A= [4 -1 0 0 0 0;-1 4 -1 0 0 0;0 -1 4 -1 0 0;0 0 -1 4 -1 0;0 0 0 -1 4 -1;0 0 0 0 ↙
-1 4];
 3 b = [0;5;0;6;-2;6];
 4
 5 tol=1e-8;
 6 n=size(A,1);
 7 %Here I am assumin initial vector as zero vector
 8 x0=zeros(n,1);
 9 x=x0;
10 iter=1;
11 r=b;    % It will work because x0=0 so r=b
12 p=r;
13 while(iter>0)
14     iter=iter+1;
15     alpha=(r'*r)/(p'*(A*p));
16     x=x+alpha*p;
17     temp=r;
18     r=r-alpha*A*p;
19
20     beta=(r'*r)/(temp'*temp);
21     p=r+beta*p;
22
23     if(norm(r)<tol)
24         break;
25     end
26 end
27 disp("Number of Iteration");
28 iter-1
29 disp("Requires Solution:")
30 x
31
```

```
>> SelfCG
Number of Iteration

ans =

     6

Requires Solution:

x =

    0.3892
    1.5569
    0.8382
    1.7959
    0.3456
    1.5864

>>
```

```
 1
 2 function[]=SelfSQRIter(A,maxNumIter)
 3 A=[17 24 1 8 15;23 5 7 14 16;4 6 13 20 22;10 12 19 21 3;11 18 25 2 8];
 4 maxNumIter=20;
 5
 6 [n,m]=size(A);
 7 for i=1:2
 8 %Initialising spectrum set by zero vector
 9 spset = zeros(n, 1);
10 Ai = A;
11 iter=1;
12 ex=eig(A);
13
14 E=zeros(n,maxNumIter);
15 while iter<=maxNumIter
16     %Here using if condiiton i distincting both the questions mentioned in
17     %the exam
18     if(i==1)
19         mu=0;
20     end
21     if(i==2)
22         mu=Ai(n,n);
23     end
24
25     Ai=Ai - mu * eye(n);
26     [Q, R] = qr(Ai);
27
28
29     Ai = R * Q + mu * eye(m);
30     ei=abs(sort(ex)-sort(diag(Ai)));
31     E(:,iter)=ei;
32     if(iter==maxNumIter)
33         spset = diag(Ai);
34     end
35     iter=iter+1;
36 end
37 %Printing spectrum set and Error matrix
38 spset
39 E
40 end
41 end
42
43
```

```
>> SelfSQRIter

spset =

   64.8024
  -20.4958
   20.1062
  -11.5578
   11.1450


E =

  Columns 1 through 11

   11.5180    6.1315    4.1265    2.9166    3.2018    2.1993    2.9911    1.9550 ↙
2.8396    1.7924    2.6864
   14.9793    5.0475    5.2044    3.3700    3.0403    3.0837    2.1562    2.8517 ↙
1.7056    2.6107    1.4112
    9.9352    4.7740    3.4832    3.0210    2.1995    2.9336    1.8384    2.7996 ↙
1.5883    2.5931    1.3677
    6.1947    5.2289    5.7431    3.2548    4.0415    2.3494    3.3089    2.0071 ↙
2.9569    1.8100    2.7299
   10.3673    1.1761    0.1046    0.0108    0.0011    0.0001    0.0000    0.0000 ↙
0.0000    0.0000    0.0000

  Columns 12 through 20

    1.6511    2.5341    1.5210    2.3869    1.4000    2.2468    1.2875    2.1143 ↙
1.1829
    2.3746    1.1875    2.1535    1.0041    1.9504    0.8484    1.7655    0.7142 ↙
1.5979
    2.3687    1.1713    2.1515    0.9981    1.9497    0.8461    1.7653    0.7133 ↙
1.5978
    1.6570    2.5503    1.5230    2.3929    1.4007    2.2490    1.2877    2.1152 ↙
1.1830
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ↙
0.0000


spset =

   64.8024
  -21.6948
  -13.1397
   21.2892
   12.7429
```

```
E =

  Columns 1 through 11

   17.4536    7.4460    3.7972    1.8738    0.7820    0.2038    0.0664    0.1722 ↙
0.1956    0.1832    0.1573
   14.4466    0.7515    2.1467    1.6804    0.6594    0.1865    0.0811    0.1753 ↙
0.1980    0.1839    0.1578
    5.9888    0.0019    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ↙
0.0000    0.0000    0.0000
    9.9366    2.7569    0.6601    0.0831    0.0373    0.0108    0.0036    0.0011 ↙
0.0004    0.0001    0.0000
   15.9748    3.9358    0.9903    0.2765    0.0852    0.0281    0.0111    0.0042 ↙
0.0020    0.0007    0.0004

  Columns 12 through 20

    0.1291    0.1030    0.0807    0.0625    0.0481    0.0367    0.0280    0.0212 ↙
0.0161
    0.1292    0.1031    0.0808    0.0626    0.0481    0.0367    0.0279    0.0212 ↙
0.0161
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ↙
0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ↙
0.0000
    0.0001    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ↙
0.0000

>>
```