Brian KYANJO
Homework 6
MATH 566

2. Companion Matrix.

a) Write down the Companion Matrix A for $p(z) = z^2 + c_1 z + c_0$ and show that $p(z) = \det(zI - A)$ for general Coefficients.

$$p(z) = z^2 + c_1 z + c_0$$

$$A = \begin{bmatrix} 0 & 1 \\ -c_0 & -c_1 \end{bmatrix}$$

- Since the roots of $p(z)$ can be Computed by solving for the eigen values of its companion matrix.

- Suppose z is a root of $p(z) = 0$, then we can show that ~~the~~ z, is an eigen value of A with eigen vector $\begin{pmatrix} 1 \\ z \end{pmatrix}$

So

$$A\begin{pmatrix} 1 \\ z \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -c_0 & -c_1 \end{pmatrix}\begin{pmatrix} 1 \\ z \end{pmatrix}$$

$$= \begin{pmatrix} z \\ -c_0 - c_1 z \end{pmatrix}$$

In the last row of $A\begin{pmatrix} 1 \\ z \end{pmatrix}$, taking $p(z) = 0$

~~in~~

$$A\begin{pmatrix} 1 \\ z \end{pmatrix} = \begin{pmatrix} z \\ z^2 \end{pmatrix} = z\begin{pmatrix} 1 \\ z \end{pmatrix}$$

$$A\begin{pmatrix} 1 \\ z \end{pmatrix} = z\begin{pmatrix} 1 \\ z \end{pmatrix}$$

Let $\frac{1}{z} = x$

$$Ax = zx \Rightarrow (A - zI)x = 0$$

Since matrix $(A - zI)$ is singular for an eigen value z, then the determinant of $(A - zI)$ is zero.

$$\det(zI - A) = 0$$

$$p(z) = \det(zI - A)$$

where $z^2 = -c_0 - c_1 z$

5) Sylvester equation.

(a) Show how the Jordan normal forms can be used to solve (2)

$$AX - XB = C$$

$A = S_A^{-1} \Lambda_A S_A$, and

$B = S_B^{-1} \Lambda_B S_B$

$$S_A^{-1} \Lambda_A S_A X - X S_B^{-1} \Lambda_B S_B = C$$

introduce S_A from the ~~right~~ left

$$S_A S_A^{-1} \Lambda_A S_A X - S_A X S_B^{-1} \Lambda_B S_B = S_A C$$

$$\Lambda_A S_A X - S_A X S_B^{-1} \Lambda_B S_B = S_A C$$

Introduce S_B^{-1} from the right

$$\Lambda_A S_A X S_B^{-1} - S_A X S_B^{-1} \Lambda_B S_B S_B^{-1} = S_A C S_B^{-1}$$

$$\Lambda_A S_A X S_B^{-1} - S_A X S_B^{-1} \Lambda_B = S_A C S_B^{-1}$$

take $\hat{C} = S_A C S_B^{-1}$

$\hat{X} = S_A X S_B^{-1}$

$$\Lambda_A \hat{X} - \hat{X} \Lambda_B = \hat{C} \quad \text{——} (1)$$

We need to obtain j^{th} column entries of $\Lambda_A \hat{X}_j$, $\Lambda_B(j,j)$ and $\hat{C} = S_A C S_{Bj}^{-1}$

where I is an identity matrix.

Therefore equation (1) becomes

$$\Lambda_A \hat{X}_j - \Lambda_B(j,j) I \hat{X}_j = S_A C S_{Bj}^{-1}$$

$$(\Lambda_A - \Lambda_B(j,j) I) \hat{X}_j = S_A C S_{Bj}^{-1}$$

where $\hat{X}_j = S_A X S_{Bj}^{-1} \Rightarrow X = S_A^{-1} \hat{X} S_{Bj}$

the solution exists iff $\Lambda_A \neq \Lambda_B(j,j) I$

Hence $X_j = S_A^{-1} \hat{X} S_{Bj}$ is the solution to the Sylvester Equation.

N0.1

```
% Generalize Minimum Residual (GMRES) method for solving Ax = b
clear
close all

%dimensions of A
m = 200;
%matrix A
A = 2*eye(m) + 0.5*randn(m)/sqrt(m);
%vector b
b = ones(m,1);
%tolerance
tol = 1e-10;
X = GMRES(A,b,tol);

%x = gmres(A,b);

function X = GMRES(A,b,tol)
    m = length(A);
    q = zeros(m,m);
    h = zeros(m,m);
    nb = norm(b,2);
    q(:,1)= b/nb;

    for n = 1:m
        v = A*q(:,n);
        for j = 1:n
           h(j,n) = q(:,j)'*v;
           v = v - h(j,n)*q(:,j);
        end
        h(n+1,n) = norm(v,2);
        q(:,n+1) = v/h(n+1,n);
        H = h(1:n+1,1:n);

        b1=nb*speye(n+1,1);
        y = H\b1;
        xn = q(:,1:n)*y;

        %calculated the residual
        r = A*xn - b;

       %relative residual
        R = norm(r,2)/nb;

        iter = n;
        if (norm(r,2) <nb * tol)
            break
        end

    end
    fprintf("The code converged at %d iterations to solution with relative residual %e\n",iter,R);

    X = xn;
end
```

```
The code converged at 17 iterations to solution with relative residual 2.324489e-11
```

N0.2

```
%Companion matrix
% Compute the roots of the sixth degree Legendre polynomial
clc
close all

fprintf("No.2\n\n");

%functions
q = @(z) (1/16)*(231*z.^6 - 315*z.^4 + 105*z.^2 -5);
%monic polynomial
p = @(z) (16/231)*q(z);

%companion matrix
syms z
A = companion(p(z));

%eigen values
format long
roots = eig(A);
fprintf("Roots in ascending order:\n");
roots = sort(roots);
disp(roots);
fprintf("Check if they are actual roots\n");
q(roots)
fprintf("Hence the roots are actual\n");

function A = companion(p)
    %coeffcients of poly
    C = coeffs(p,'all');
    cof = fliplr(C);
    %degree of poly
    n = polynomialDegree(p);
    cof = (cof(1:n));
    cof = double(cof);
    I = eye(n-1,n-1);
    A = [zeros(n-1,1) I];
    A = [A; -cof];

end
```

```
No.2

Roots in ascending order:
  -0.932469514203151
  -0.661209386466264
  -0.238619186083197
   0.238619186083197
   0.661209386466265
   0.932469514203153

Check if they are actual roots

ans =

   1.0e-13 *

  -0.106581410364015
```

```
    0.008881784197001
   -0.006106226635438
   -0.003885780586188
   -0.004440892098501
    0.044408920985006
```

```
 Hence the roots are actual
```

Published with MATLAB® R2020b

N0.3

```
% Rayleigh quotient iteration
%===========================================================================
% function rqi is a powerful method for finding an eigenvalue—eigenvector
% pairs of certain matrices (especially symmetric tridiagonal ones!
% input: A - matrix
%        x0 - intial starting
%        ep - tolerance
% output: v and lam are the eigenvalue-eigenvector pair that the algorithm
%         converged to.
%===========================================================================

function [v,lam] = rqi(A,x0,ep)
    [m,n] = size(A);
    v = x0;
    lam = v'*A*v;
    I = eye(n);
    kmax = 100;
    for k = 1:kmax
        u = A - lam*I;
        w = u\v;
        v2 = w/norm(w);
        lam2 = v2'*A*v2;

        iter = k;

        %convergence
        if norm(A*v2 - lam2*v2) < ep
            break
        end
        v = v2;
        lam = lam2;

    end
    fprintf("The code converged at %d iterations to solution\n",iter);


end
```

```
Not enough input arguments.

Error in rqi (line 13)
    [m,n] = size(A);
```

```
fprintf("Compute an eigenvalue—eigenvector pair of the matrix\n\n");
clear
close all;
%initial vector
x0 = [1 1 1 1 1 1]'./(6^0.5);
%matrix A
m = 6; n = 6;
A = matrix(m,n)

%tolerance
ep = 1e-10;

%fuction rqi
[v,lam] = rqi(A,x0,ep)
fprintf("Eigen value and its corresponding eigen vector to which the code converges are v and lam.\n\n");

%verification
[V,D] = eig(A);

fprintf("The error in the approximate eigen value.\n\n");
error_lam = abs(D(4,4) - lam)
fprintf("The error in the approximate eigen value.\n\n");
error_v = abs(V(:,4) - (-v))
fprintf("Hence v and lam are well aprroximated since the error is too small.\n\n");

function A = matrix(m,n)
    A = zeros(m,n);
    for i = 1:n
       for j = 1:m
          if i ==j
             A(i,j) = -2;
          elseif i == j+1
             A(i,j) = 1;
          elseif i == j-1
             A(i,j) = 1;
          end
          A(1,2) = 2;
       end

    end
end
```

```
Compute an eigenvalue—eigenvector pair of the matrix


A =

    -2     2     0     0     0     0
     1    -2     1     0     0     0
     0     1    -2     1     0     0
     0     0     1    -2     1     0
     0     0     0     1    -2     1
     0     0     0     0     1    -2

The code converged at 4 iterations to solution

v =

  -0.534522431006303
  -0.516309036260995
```

```
  -0.462910056395315
  -0.377964522630757
  -0.267261309845902
  -0.138344647190983


lam =

  -0.068148339178898
```

Eigen value and its corresponding eigen vector to which the code converges are v and lam.

The error in the approximate eigen value.

```
error_lam =

     8.242965809923675e-09
```

The error in the approximate eigen value.

```
error_v =

   1.0e-07 *

   0.528185452042251
   0.355976077504039
   0.065090395295897
   0.496215292189461
   0.679334780095964
   0.483416076335619
```

Hence v and lam are well aprroximated since the error is too small.

Published with MATLAB® R2020b

N0.4

```
% Gerschgorin's theorem
clc
close all
fprintf("Find regions in the complex plane where the eigenvalues are located.\n\n");

%matrix A
A = [3 -1 -1 1; -1 2 -1/4 1 ; -1 -1 -3 1/2 ...
    ;-1/2 -1/4 0 -7];

%eigen values of A
format long
%centers
c1 = A(1,1); c2 = A(2,2);
c3 = A(3,3); c4 = A(4,4);
%radius
r1 = 1 + 1+ 1; r2 = 1 + 1/4 +1;
r3 = 1 +1 + 1/2; r4 = 1/2 +1/4;

%verification
fprintf("The actual eigen values are:\n\n");
lam = eig(A)

fprintf("The matrix A has eigenvalues in the union of the Gerschgorin disks;\n")
fprintf("|lambda - 3|<3, \n|lambda - 2|<2.25, \n|lambda + 3|<2.5, \n|lambda + 7|<0.75, \n")
fprintf("So comparing the actual eigen values with the plot, its clear that two \n of the eigen values are located in the first two disks, and the \n


%create the disc
figure(1)
p = nsidedpoly(1000, 'Center', [c1 0], 'Radius', r1);
plot(p, 'FaceColor', 'r')
axis equal
grid on
hold on
plot(c1,0,'r.','MarkerSize',20)
hold on
p2 = nsidedpoly(1000, 'Center', [c2 0], 'Radius', r2);
plot(p2, 'FaceColor', 'b')
axis equal
hold on
plot(c2,0,'r.','MarkerSize',20)
hold on
p3 = nsidedpoly(1000, 'Center', [c3 0], 'Radius', r3);
plot(p3, 'FaceColor', 'g')
axis equal
hold on
plot(c3,0,'r.','MarkerSize',20)
hold on
p4 = nsidedpoly(1000, 'Center', [c4 0], 'Radius', r4);
plot(p4, 'FaceColor', 'y')
axis equal
hold on
plot(c4,0,'r.','MarkerSize',20)
xlabel("Re(\lambda)"); ylabel("Im(\lambda)");
title("Gerschgorin disks for the matrix A")
```

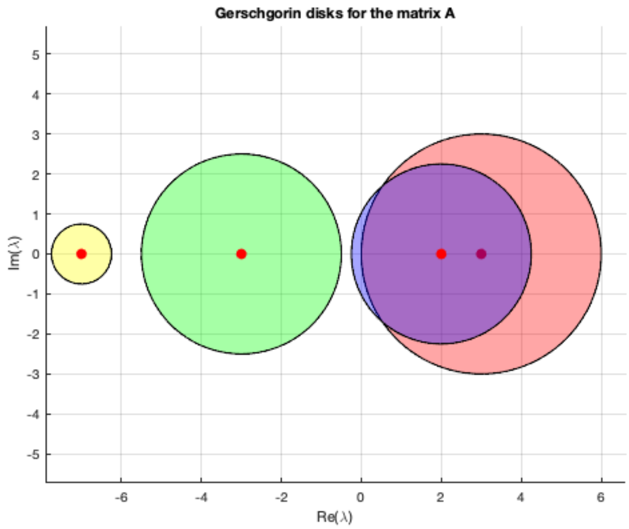
```
Find regions in the complex plane where the eigenvalues are located.

The actual eigen values are:


lam =

   3.645698826149831
   1.533195532698929
  -3.279302662649521
  -6.899591696199242

The matrix A has eigenvalues in the union of the Gerschgorin disks;
|lambda - 3|<3,
|lambda - 2|<2.25,
|lambda + 3|<2.5,
|lambda + 7|<0.75,
So comparing the actual eigen values with the plot, its clear that two
 of the eigen values are located in the first two disks, and the
 remaining two in the union of the other two disks.
This is because the two disks are disjoint from the other 2.
 Hence from Gerschgorin's theorem the results is verified.
```



Published with MATLAB® R2020b

N0.5

```
%Sylvester equations
clc
close all

SA = [4 7 -6 10 9; 4 -6 4 9 5; -2 4 6 10 3; -4 6 3 -3 7;...
       -1 8 0 6 2];
va = [  4, 1,3,9,10]; VA = diag(va);

SB = [6 6 -1 5;8 7 -6 -6; -3 3 -5 10; -6 -6 -9 -7];

vb = [-7, -4, -3, -5]; VB = diag(vb);

C = [-9 10 6 -7; -8 -2 -5 3; -7 0 -6 5;-8 9 0 8;-4 -9 -4 5];


%format long
[m,n] = size(C);

%identity matrix
e = ones(m,1); I = diag(e);

%inverse of SB and SA
invSB = inv(SB); invSA = inv(SA);

xhat = zeros(m,n); x = zeros(m,n);

for i = 1:n
   A = VA - VB(i,i)*I;
   Chat = SA*C*invSB(:,i);
   xhat(:,i) = A\Chat;
end

J = 1:n;
fprintf("The solution in matrix form is:\n\n")
x(:,J) = invSA*xhat*SB(:,J)
```

```
The solution in matrix form is:


x =

  Columns 1 through 3

   7.164257638512381  10.901274644160008  -0.358273182187597
   2.277357282833662   2.038219188954683   0.346773493076263
   3.164640265518672   5.588511648918812  -1.286321203219335
  -2.473528353258581  -0.290393814006962  -1.064736389326263
  -1.340482562043509  -0.388265945321809  -0.949257360228531

  Column 4

   2.112026215055822
  -0.755989882624112
   2.534960382610660
   2.264702503984485
   1.772782710005899
```