

Math221 Homework # 8 Solutions

Instructor: James W. Demmel

TA: Jiawang Nie

November 10, 2004

Solution to Problem 4.16

Math. Formulation Plug $x_i = g_i(t)$ ($i = 1, 2, 3$) in polynomial matrix $F(t) = F(g_1, g_2, g_3)$. Now the problem becomes to find all the values of t such that $\det F(t) = 0$. Write $F(t)$ in matrix polynomial form

$$F(t) = t^m A_m + t^{m-1} A_{m-1} + \cdots + t A_1 + A_0.$$

Use the technique in Section 4.5.3 to this nonlinear eigenvalue problem to the standard generalized eigenvalue problem $A - \lambda B$. If B is well-conditioned, find the eigenvalues of $B^{-1}A$; otherwise use Matlab command `eig(B,A)` to find all the eigenvalues.

Error Bounds Here we only discuss the error bounds when B is well-conditioned. At first find the error bounds in computing $B^{-1}A$ by using the techniques in solving linear systems. And then use Theorem 4.4 to get an error bound for the eigenvalues. Please see the codes as below for more details.

The Algorithm The following algorithm defines the input for each subproblem.

```
if pick == -1,

% Test -1
NumTerms = [2];
Sterms = [[1 2 ]; ...
          [1 2 ]; ...
          [1 2 ]; ...
          [1 2 ]];
tC = [1 1 1];
Curve = [ 0 1 1 1 2 1 ];

elseif pick == 0

%
% Test 0
NumTerms = [2];
```

```

Sterms = [ [ 1 0 ] ; ...
           [ 1 1 ] ; ...
           [ 1 0 ] ; ...
           [ 2 -6] ];
tC = [1 1 1];
Curve = [ 1 1 1 2 1 3 ];

elseif pick == 1

%
% Problem 4.6.1
NumTerms = [[ 3 0 ];[ 0 4 ]];
Sterms = [ [ 1 0 0 1 ]' , ...
           [ 0 1 0 1 ]' , ...
           [ 0 0 1 1 ]' , ...
           [ 1 0 0 3 ]' , ...
           [ 0 1 0 5 ]' , ...
           [ 0 0 1 -7 ]' , ...
           [ 0 0 0 10 ]' ];
tC = [2 2 2];
Curve = [ 0 0 1 1 0 1 2 0 1 ];

% Plot Curve and Surface
t = (-1:.1:1);
Cx = polyval( fliplr(Curve( 1 : tC(1)+1 )) , t );
Cy = polyval( fliplr(Curve( tC(1)+2 : tC(1)+tC(2)+2 )) , t );
Cz = polyval( fliplr(Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 )) , t );
density = 10;
Cxrange = ( min(Cx)-1 : (max(Cx)-min(Cx))/density : max(Cx)+1 );
Cyrange = ( min(Cy)-1 : (max(Cy)-min(Cy))/density : max(Cy)+1 );
[ Cxmat, Cymat ] = meshgrid( Cxrange, Cyrange );
Surface11 = -Cxmat - Cymat;
Surface22 = (3/7)*Cxmat + (5/7)*Cymat + (10/7);
figure(1)
hold off, clf
mesh( Cxrange , Cyrange , Surface11 );
hold on
mesh( Cxrange , Cyrange , Surface22 );
plot3(Cx,Cy,Cz,'w')
xlabel('X'); ylabel('Y'); zlabel('Z')
grid
pause
view([1 -1 -.075])

elseif pick == 2

```

```

%
% Problem 4.6.2
NumTerms = [[ 3 0 ];[ 0 4 ]];
Sterms = [ [ 1 0 0 1 ]' , ...
           [ 0 1 0 1 ]' , ...
           [ 0 0 1 1 ]' , ...
           [ 1 0 0 3 ]' , ...
           [ 0 1 0 5 ]' , ...
           [ 0 0 1 -7 ]' , ...
           [ 0 0 0 10 ]' ];

tC = [3 3 3];
Curve = [ 0 0 0 1 1 0 0 1 2 0 0 1 ];

% Plot Curve and Surface
t = (-2:.1:2);
Cx = polyval( fliplr(Curve( 1 : tC(1)+1 )) , t );
Cy = polyval( fliplr(Curve( tC(1)+2 : tC(1)+tC(2)+2 )) , t );
Cz = polyval( fliplr(Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 )) , t );
density = 20;
Cxrange = ( min(Cx)-1 : (max(Cx)-min(Cx))/density : max(Cx)+1 );
Cyrange = ( min(Cy)-1 : (max(Cy)-min(Cy))/density : max(Cy)+1 );
[ Cxmat, Cymat ] = meshgrid( Cxrange, Cyrange );
Surface11 = -Cxmat - Cymat;
Surface22 = (3/7)*Cxmat + (5/7)*Cymat + (10/7);
figure(1)
hold off, clg
mesh( Cxrange , Cyrange , Surface11 );
hold on
mesh( Cxrange , Cyrange , Surface22 );
plot3(Cx,Cy,Cz,'w')
xlabel('X'); ylabel('Y'); zlabel('Z')
grid
pause
view([1 -1 -.075])

elseif pick == 3

%
% Problem 4.6.3
NumTerms = [[ 3 0 ];[ 0 4 ]];
Sterms = [ [ 1 0 0 1 ]' , ...
           [ 0 1 0 1 ]' , ...
           [ 0 0 1 1 ]' , ...
           [ 1 0 0 3 ]' , ...

```

```

                [ 0 1 0 5 ]' , ...
                [ 0 0 1 -7 ]' , ...
                [ 0 0 0 10 ]' ];
tC = [1 1 1];
Curve = [ 0 1 1 1 2 1 ];

% Plot Curve and Surface
t = (-4:.1:4);
Cx = polyval( fliplr(Curve( 1 : tC(1)+1 )) , t );
Cy = polyval( fliplr(Curve( tC(1)+2 : tC(1)+tC(2)+2 )) , t );
Cz = polyval( fliplr(Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 )) , t );
density = 20;
Cxrange = ( min(Cx)-1 : (max(Cx)-min(Cx))/density : max(Cx)+1 );
Cyrange = ( min(Cy)-1 : (max(Cy)-min(Cy))/density : max(Cy)+1 );
[ Cxmat, Cymat ] = meshgrid( Cxrange, Cyrange );
Surface11 = -Cxmat - Cymat;
Surface22 = (3/7)*Cxmat + (5/7)*Cymat + (10/7);
figure(1)
hold off, clg
mesh( Cxrange , Cyrange , Surface11 );
hold on
mesh( Cxrange , Cyrange , Surface22 );
plot3(Cx,Cy,Cz,'w')
xlabel('X'); ylabel('Y'); zlabel('Z')
grid
pause
view([1 -1 -.3])

elseif pick == 4

%
% Problem 4.6.4
NumTerms = [[ 1 0 ];[ 0 4 ]];
Sterms = [ [ 0 0 0 1 ]' , ...
            [ 1 0 0 3 ]' , ...
            [ 0 1 0 5 ]' , ...
            [ 0 0 1 -7 ]' , ...
            [ 0 0 0 9 ]' ];
tC = [2 2 2];
Curve = [ 0 0 1 1 0 1 2 0 1 ];

elseif pick == 5

%
% Problem 4.6.5

```

```

NumTerms = [[ 3 0 ];[ 0 4 ]];
Sterms = [ [ 1 0 0 1 ]', ...
            [ 0 1 0 1 ]', ...
            [ 0 0 1 1 ]', ...
            [ 1 0 0 3 ]', ...
            [ 0 1 0 5 ]', ...
            [ 0 0 1 -7 ]', ...
            [ 0 0 0 8 ]' ];
tC = [2 2 2];
Curve = [ 0 0 1 1 0 1 2 0 1 ];

elseif pick == 6

%
% Problem 4.6.6
NumTerms = [[ 3 1 ];[ 1 4 ]];
Sterms = [ [ 1 0 0 1 ]', ...
            [ 0 1 0 1 ]', ...
            [ 0 0 1 1 ]', ...
            [ 0 0 1 1 ]', ...
            [ 1 0 0 1 ]', ...
            [ 1 0 0 3 ]', ...
            [ 0 1 0 5 ]', ...
            [ 0 0 1 -7 ]', ...
            [ 0 0 0 10 ]' ];
tC = [2 2 2];
Curve = [ 0 0 1 1 0 1 2 0 1 ];

elseif pick == 7

%
% Problem 4.6.7
NumTerms = [[ 2 2 7 ];[ 2 3 4 ];[ 1 4 3 ]];
Sterms = [ [ 1 1 0 1 ]', ...
            [ 0 0 5 1 ]', ...
            [ 0 1 0 1 ]', ...
            [ 0 0 5 -7 ]', ...
            [ 0 0 0 2 ]', ...
            [ 0 0 0 3 ]', ...
            [ 0 2 0 -1 ]', ...
            [ 0 0 0 1 ]', ...
            [ 2 0 0 -1 ]', ...
            [ 1 1 2 1 ]', ...
            [ 1 0 0 3 ]', ...
            [ 0 1 0 5 ]', ...

```

```

        [ 0 0 1 -7 ]' , ...
        [ 0 0 0 8 ]' , ...
        [ 0 0 0 5 ]' , ...
        [ 1 0 0 1 ]' , ...
        [ 0 1 0 1 ]' , ...
        [ 0 0 1 1 ]' , ...
        [ 1 1 0 1 ]' , ...
        [ 1 0 1 1 ]' , ...
        [ 0 1 1 1 ]' , ...
        [ 0 0 0 3 ]' , ...
        [ 1 0 0 1 ]' , ...
        [ 0 0 1 3 ]' , ...
        [ 0 1 1 -9 ]' , ...
        [ 3 0 0 1 ]' , ...
        [ 0 4 0 -1 ]' , ...
        [ 0 0 5 4 ]' ];
tC = [5 5 5];
Curve = [ 7 -3 0 0 0 1 ...
          1  0 1 0 0 1 ...
          2  0 1 0 0 1 ];

end

```

The following algorithm find the eigenvalues and intersection points.

```

% Solution to Problem 4.6
%
% Inputs:
% ( d = dimension of matrix F = min(size(NumTerms)) )
% NumTerms(d,d), where
% NumTerms(i,j) = number of polynomials terms in F(i,j) defining Surface S
% ( tS = total number of polynomials terms in all entries of F
% = sum(sum(NumTerms)) )
% Sterms(4,tS)
% All terms c*x1^e1*x2^e2*x3^e3 stored sequentially, columnwise for F
% Sterms(1,k) = exponent e1 for term k
% Sterms(2,k) = exponent e2 for term k
% Sterms(3,k) = exponent e3 for term k
% Sterms(4,k) = coefficient c for term k
% tC(3) = degree of polynomials defining components x1, x2, x3 of Curve C
% Curve(sum(tC)+3) = coefficients of polynomials, x1 then x2 then x3,
% from constant term up
%

```

```

% Example - Problem 4.6.1
% NumTerms = [[ 3 0 ];[ 0 4 ]];
% Sterms = [ [ 1 0 0 1 ]' , ...
%           [ 0 1 0 1 ]' , ...
%           [ 0 0 1 1 ]' , ...
%           [ 1 0 0 3 ]' , ...
%           [ 0 1 0 5 ]' , ...
%           [ 0 0 1 -7 ]' , ...
%           [ 0 0 0 10 ]' ];
% tC = [2 2 2];
% Curve = [ 0 0 1 1 0 1 2 0 1 ];
%
d = min(size(NumTerms));
max_degree = max(tC*Sterms(1:3,:))
% Initialize array [A_0, A_1, ... , A_d] of Matrix Polynomial Coefficients
Matrix_Poly_Coeffs = zeros(d,(max_degree+1)*d);
% Initialize array of Matrix Polynomial Coefficients error bounds
dMatrix_Poly_Coeffs = zeros(d,(max_degree+1)*d);
k=1;
% For each entry of F
for j=1:d,
    for i=1:d
%       Find polynomial terms in Sterms
        terms = (k : k + NumTerms(i,j) -1);
%       Compute degree of F(i,j)
        degij = max( tC * Sterms(1:3,terms) );
%       Initialize vector of coefficients of F(i,j)
        coeffij = zeros( 1, degij + 1 );
%       Initialize vector of coefficient error bounds for F(i,j)
        dcoeffij = zeros( 1, degij + 1 );
%       For each term in polynomial p defining F(i,j)
        for r = terms,
%           Compute term and add to coeffij
%           Compute error bound on term and add to dcoeffij
            p = Sterms(4,r);
            dp = 0;
%           p = p * x1^e1
            x1 = Curve( 1 : tC(1)+1 );
            for t = 1 : Sterms(1,r),
%               For straightforward "dot-product" implementation of conv
                dp = conv(dp, abs(x1) ) + eps*conv( abs(p), abs(x1) );
%               For FFT implementation of conv:
                dp = eps*ones(1,length(p)+length(x1)-1)*(norm(p)+norm(dp))*norm(x1) ;
                p = conv( p, x1 );
            end
        end
    end
end

```

```

%      p = p * x2^e2
x2 = Curve( tC(1)+2 : tC(1)+tC(2)+2 );
for t = 1 : Sterms(2,r),
%      For straightforward "dot-product" implementation of conv
      dp = conv(dp, abs(x2) ) + eps*conv( abs(p), abs(x2) );
%      For FFT implementation of conv:
%      dp = eps*ones(1,length(p)+length(x2)-1)*(norm(p)+norm(dp))*norm(x2) ;
      p = conv( p, x2 );
end
%      p = p * x3^e3
x3 = Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 );
for t = 1 : Sterms(3,r),
%      For straightforward "dot-product" implementation of conv
      dp = conv(dp, abs(x3) ) + eps*conv( abs(p), abs(x3) );
%      For FFT implementation of conv:
%      dp = eps*ones(1,length(p)+length(x3)-1)*(norm(p)+norm(dp))*norm(x3) ;
      p = conv(p, x3 );
end
%      Accumulate p into array of coefficients of F(i,j)
coeffij = coeffij + [ p , zeros( 1, length(coeffij)-length(p) ) ];
%      Accumulate dp into array of error bounds on coefficients of F(i,j)
dcoeffij = dcoeffij + ...
          [ dp , zeros( 1, length(coeffij)-length(p) ) ] + ...
          eps*abs(coeffij);
end
%      Store F(i,j) in Matrix_Poly_Coeffs
loc = ( j : d : degij*d + j );
Matrix_Poly_Coeffs( i , loc ) = coeffij;
%      Store error bound on F(i,j) in dMatrix_Poly_Coeffs
dMatrix_Poly_Coeffs( i , loc ) = dcoeffij;
k = k + NumTerms(i,j);
end
end
Matrix_Poly_Coeffs      % display matrix polynomial coefficients
disp('pause (hit enter to continue)'), pause
dMatrix_Poly_Coeffs    % display error bounds on Matrix_Poly_Coeffs
disp('pause (hit enter to continue)'), pause
dim = d*max_degree;
%
% Build matrix pencil A - lambda B whose eigenvalues we want
%
B = eye(dim);
B(1:d,1:d) = Matrix_Poly_Coeffs( : , dim+1 : dim+d );
A = zeros(dim,dim);
A(d+1:dim,1:dim-d) = eye(dim-d);

```

```

for i = 1:max_degree
    A(1:d,(i-1)*d+1:i*d) = ...
        -Matrix_Poly_Coeffs( 1:d , dim - i*d + 1 : dim+ (1-i)*d);
end
% Display pencil
A
disp('pause (hit enter to continue)'), pause
B
disp('pause (hit enter to continue)'), pause
%
% If B(1:d,1:d) is well conditioned, convert to standard eigenproblem
% and compute error bounds
%
rcB = rcond(B(1:d,1:d));
if (rcB > 1e-8)
%   Convert to standard eigenproblem
    disp(['1/cond(A_d) = ', num2str(rcB), ' so convert to ' ...
        'standard eigenproblem for'])
    A(1:d,:) = (B(1:d,1:d)\A(1:d,:));
    A
    disp('pause (hit enter to continue)'), pause
    %f1 = flops;
    D = eig(A)
    %f2 = flops - f1;
    %disp(['cost( D=eig(A) ) = ', num2str(f2/dim^3), ...
        %   ' n^3 flops, with n = ',int2str(dim)])
%   Compute error bounds for computed eigenvalues
%   Get right and left eigenvectors (Vr and Vl)
    %f1 = flops;
    [Vr,Dr] = eig(A);
    [Vl,Dl] = eig(A');
    %f2 = flops - f1;
    %disp(['cost( left,right evecors ) = ', num2str(f2/dim^3), ...
        %   ' n^3 flops, with n = ',int2str(dim)])
%   Reorder eigenvalues, eigenvectors so they match
    [sDr,sIDr] = sort(diag(Dr));
    Vr = Vr(:,sIDr);
    D = sDr;
    [sDl,sIDl] = sort(diag(Dl));
    Vl = Vl(:,sIDl);
%   Compute eigenvalue condition numbers
    Econd = 1 ./ abs(diag(conj(Vl')*Vr));
%   Compute eigenvalue error bounds
    n1 = norm( dMatrix_Poly_Coeffs( : , d+1:d+dim ), 1 );
    n2 = norm( dMatrix_Poly_Coeffs( : , 1:d ), 1 );

```

```

n3 = norm( A( 1:d, : ), 1 );
Ebnd = Econd .* ( n1/rcB + n2/rcB * n3 );
disp(' Eigenvalue          Error Bound' )
[D, Ebnd]
disp('pause (hit enter to continue)'), pause
% Display intersection points
% Get coeffs of polynomials and derivatives of polynomials
polyx1 = fliplr(Curve( 1 : tC(1)+1 ));
dpolyx1 = (tC(1):-1:1) .* polyx1(1:tC(1));
polyx2 = fliplr(Curve( tC(1)+2 : tC(1)+tC(2)+2 ));
dpolyx2 = (tC(2):-1:1) .* polyx2(1:tC(2));
polyx3 = fliplr(Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 ));
dpolyx3 = (tC(3):-1:1) .* polyx3(1:tC(3));
% Evaluate polynomials and error bounds
Cx1 = polyval( polyx1 , D );
Cx2 = polyval( polyx2 , D );
Cx3 = polyval( polyx3 , D );
dCx1 = abs(polyval( dpolyx1 , D )) .* Ebnd + ...
        eps*polyval( abs(polyx1), abs(D) );
dCx2 = abs(polyval( dpolyx2 , D )) .* Ebnd + ...
        eps*polyval( abs(polyx2), abs(D) );
dCx3 = abs(polyval( dpolyx3 , D )) .* Ebnd + ...
        eps*polyval( abs(polyx3), abs(D) );
% Display results for all eigenvalues, including complex and infinite
disp('all eigenvalues, x1 coords of intersections, error bounds')
[D,Cx1,dCx1]
disp('all eigenvalues, x2 coords of intersections, error bounds')
[D,Cx2,dCx2]
disp('all eigenvalues, x3 coords of intersections, error bounds')
[D,Cx3,dCx3]
disp('pause (hit enter to continue)'), pause
% Display results for real eigenvalues only
Dreal = find(abs(imag(D)) < 1e-11*abs(real(D)));
disp('real eigenvalues, x1 coords of intersections, error bounds')
[D(Dreal),Cx1(Dreal),dCx1(Dreal)]
disp('real eigenvalues, x2 coords of intersections, error bounds')
[D(Dreal),Cx2(Dreal),dCx2(Dreal)]
disp('real eigenvalues, x3 coords of intersections, error bounds')
[D(Dreal),Cx3(Dreal),dCx3(Dreal)]
else
% Solve as generalized eigenproblem (no error bound computed)
disp(['1/cond(A_d) = ', num2str(rcB), ' so solve as ', ...
'generalized eigenproblem'])
% Display flop count
f1 = flops;

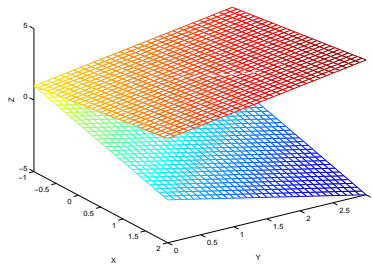
```

```

D=eig(B,A);
f2 = flops - f1;
D = sort(D);
disp(['cost( D=eig(B,A) ) = ', num2str(f2/dim^3), ...
      ' n^3 flops, with n = ',int2str(dim)])
disp('pause (hit enter to continue)'), pause
% Display intersection points
polyx1 = fliplr(Curve( 1 : tC(1)+1 ));
polyx2 = fliplr(Curve( tC(1)+2 : tC(1)+tC(2)+2 ));
polyx3 = fliplr(Curve( tC(1)+tC(2)+3 : tC(1)+tC(2)+tC(3)+3 ));
Cx1 = polyval( polyx1 , D );
Cx2 = polyval( polyx2 , D );
Cx3 = polyval( polyx3 , D );
% Display results for all eigenvalues, including complex and infinite
disp('all eigenvalues, x1 coords of intersections')
[D,Cx1]
disp('all eigenvalues, x2 coords of intersections')
[D,Cx2]
disp('all eigenvalues, x3 coords of intersections')
[D,Cx3]
disp('pause (hit enter to continue)'), pause
% Display results for real eigenvalues only
Dreal = find(abs(imag(D)) <= 1e-11*abs(real(D)));
disp('real eigenvalues, all coords of intersections')
[D(Dreal),Cx1(Dreal),Cx2(Dreal),Cx3(Dreal)]
end

```

Example 1 The plot is in figure.



\small

Matrix_Poly_Coeffs =

```
3 0 0 0 3 0
0 1 0 0 0 1
```

pause (hit enter to continue)

dMatrix_Poly_Coeffs =

1.0e-014 *

```
0.1554 0 0 0 0.1998 0
0 0.7550 0 0 0 0.6217
```

pause (hit enter to continue)

A =

```
0 0 -3 0
0 0 0 -1
1 0 0 0
0 1 0 0
```

pause (hit enter to continue)

B =

```
3 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

pause (hit enter to continue)

1/cond(A_d) = 0.33333 so convert to standard eigenproblem for

A =

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

pause (hit enter to continue)

D =

0 + 1.0000i
0 - 1.0000i
0 + 1.0000i
0 - 1.0000i

Eigenvalue Error Bound

ans =

0 - 1.0000i 0.0000
0 - 1.0000i 0.0000
0 + 1.0000i 0.0000
0 + 1.0000i 0.0000

pause (hit enter to continue)

all eigenvalues, x1 coords of intersections, error bounds

ans =

0 - 1.0000i -1.0000 0.0000
0 - 1.0000i -1.0000 0.0000
0 + 1.0000i -1.0000 0.0000
0 + 1.0000i -1.0000 0.0000

all eigenvalues, x2 coords of intersections, error bounds

ans =

0 - 1.0000i 0 0.0000
0 - 1.0000i 0 0.0000
0 + 1.0000i 0 0.0000
0 + 1.0000i 0 0.0000

all eigenvalues, x3 coords of intersections, error bounds

ans =

0 - 1.0000i 1.0000 0.0000
0 - 1.0000i 1.0000 0.0000
0 + 1.0000i 1.0000 0.0000
0 + 1.0000i 1.0000 0.0000

pause (hit enter to continue)

real eigenvalues, x1 coords of intersections, error bounds

```
ans =
```

```
Empty matrix: 0-by-3
```

```
real eigenvalues, x2 coords of intersections, error bounds
```

```
ans =
```

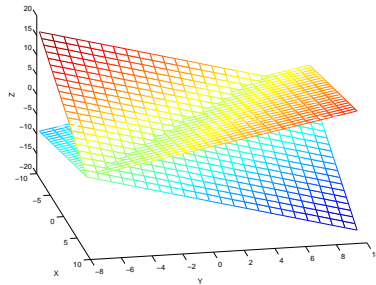
```
Empty matrix: 0-by-3
```

```
real eigenvalues, x3 coords of intersections, error bounds
```

```
ans =
```

```
Empty matrix: 0-by-3
```

Example 2 The plot is in figure.



```
\small
```

```
Matrix_Poly_Coeffs =
```

```
    3    0    0    0    0    0    3    0
    0    1    0    0    0    0    0    1
```

```
pause (hit enter to continue)
```

```
dMatrix_Poly_Coeffs =
```

```
1.0e-014 *
```

```
    0.1554    0    0    0    0    0    0.1998    0
```

0 0.7550 0 0 0 0 0 0.6217

pause (hit enter to continue)

A =

0	0	0	0	-3	0
0	0	0	0	0	-1
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0

pause (hit enter to continue)

B =

3	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

pause (hit enter to continue)

1/cond(A_d) = 0.33333 so convert to standard eigenproblem for

A =

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

pause (hit enter to continue)

D =

-1.0000
0.5000 + 0.8660i
0.5000 - 0.8660i
-1.0000
0.5000 + 0.8660i

0.5000 - 0.8660i

Eigenvalue Error Bound

ans =

-1.0000	0.0000
-1.0000	0.0000
0.5000 - 0.8660i	0.0000
0.5000 - 0.8660i	0.0000
0.5000 + 0.8660i	0.0000
0.5000 + 0.8660i	0.0000

pause (hit enter to continue)

all eigenvalues, x1 coords of intersections, error bounds

ans =

-1.0000	-1.0000	0.0000
-1.0000	-1.0000	0.0000
0.5000 - 0.8660i	-1.0000 + 0.0000i	0.0000
0.5000 - 0.8660i	-1.0000 + 0.0000i	0.0000
0.5000 + 0.8660i	-1.0000 - 0.0000i	0.0000
0.5000 + 0.8660i	-1.0000 - 0.0000i	0.0000

all eigenvalues, x2 coords of intersections, error bounds

ans =

-1.0000	0.0000	0.0000
-1.0000	0.0000	0.0000
0.5000 - 0.8660i	-0.0000 + 0.0000i	0.0000
0.5000 - 0.8660i	-0.0000 + 0.0000i	0.0000
0.5000 + 0.8660i	-0.0000 - 0.0000i	0.0000
0.5000 + 0.8660i	-0.0000 - 0.0000i	0.0000

all eigenvalues, x3 coords of intersections, error bounds

ans =

-1.0000	1.0000	0.0000
-1.0000	1.0000	0.0000
0.5000 - 0.8660i	1.0000 + 0.0000i	0.0000
0.5000 - 0.8660i	1.0000 + 0.0000i	0.0000
0.5000 + 0.8660i	1.0000 - 0.0000i	0.0000
0.5000 + 0.8660i	1.0000 - 0.0000i	0.0000

```
0.5000 + 0.8660i    1.0000 - 0.0000i    0.0000
```

```
pause (hit enter to continue)
```

```
real eigenvalues, x1 coords of intersections, error bounds
```

```
ans =
```

```
-1.0000    -1.0000    0.0000  
-1.0000    -1.0000    0.0000
```

```
real eigenvalues, x2 coords of intersections, error bounds
```

```
ans =
```

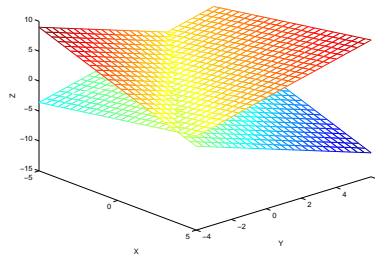
```
-1.0000    0.0000    0.0000  
-1.0000    0.0000    0.0000
```

```
real eigenvalues, x3 coords of intersections, error bounds
```

```
ans =
```

```
-1.0000    1.0000    0.0000  
-1.0000    1.0000    0.0000
```

Example 3 The plot is in figure.



```
\small
```

```
Matrix_Poly_Coeffs =
```

```
3    0    3    0  
0    1    0    1
```

```
pause (hit enter to continue)
```

```

dMatrix_Poly_Coeffs =

1.0e-014 *

    0.1554      0    0.1998      0
         0    0.7550      0    0.6217

pause (hit enter to continue)

A =

    -3     0
     0    -1

pause (hit enter to continue)

B =

     3     0
     0     1

pause (hit enter to continue)
1/cond(A_d) = 0.33333 so convert to standard eigenproblem for

A =

    -1     0
     0    -1

pause (hit enter to continue)

D =

    -1
    -1

Eigenvalue          Error Bound

ans =

    -1.0000    0.0000
    -1.0000    0.0000

pause (hit enter to continue)
all eigenvalues, x1 coords of intersections, error bounds

```

ans =

-1.0000	-1.0000	0.0000
-1.0000	-1.0000	0.0000

all eigenvalues, x2 coords of intersections, error bounds

ans =

-1.0000	0	0.0000
-1.0000	0	0.0000

all eigenvalues, x3 coords of intersections, error bounds

ans =

-1.0000	1.0000	0.0000
-1.0000	1.0000	0.0000

pause (hit enter to continue)

real eigenvalues, x1 coords of intersections, error bounds

ans =

-1.0000	-1.0000	0.0000
-1.0000	-1.0000	0.0000

real eigenvalues, x2 coords of intersections, error bounds

ans =

-1.0000	0	0.0000
-1.0000	0	0.0000

real eigenvalues, x3 coords of intersections, error bounds

ans =

-1.0000	1.0000	0.0000
-1.0000	1.0000	0.0000

Example 4 \small

Matrix_Poly_Coeffs =

1	0	0	0	0	0
0	0	0	0	0	1

pause (hit enter to continue)

dMatrix_Poly_Coeffs =

1.0e-014 *

0.0222	0	0	0	0	0
0	0.7327	0	0	0	0.6217

pause (hit enter to continue)

A =

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

pause (hit enter to continue)

B =

0	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

pause (hit enter to continue)

all eigenvalues, x1 coords of intersections

ans =

-Inf	Inf
0	0
0	0
Inf	Inf

all eigenvalues, x2 coords of intersections

ans =

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

all eigenvalues, x3 coords of intersections

ans =

```
-Inf  Inf
  0    2
  0    2
 Inf  Inf
```

pause (hit enter to continue)

real eigenvalues, all coords of intersections

ans =

```
-Inf  Inf  Inf  Inf
  0    0   1   2
  0    0   1   2
 Inf  Inf  Inf  Inf
```

Example 5 \small

Matrix_Poly_Coeffs =

```
  3    0    0    0    3    0
  0   -1    0    0    0    1
```

pause (hit enter to continue)

dMatrix_Poly_Coeffs =

1.0e-014 *

```
  0.1554    0    0    0    0.1998    0
    0    0.7550    0    0    0    0.6217
```

pause (hit enter to continue)

A =

```
0  0  -3  0
0  0   0  1
1  0   0  0
0  1   0  0
```

pause (hit enter to continue)

B =

```
3  0  0  0
0  1  0  0
0  0  1  0
0  0  0  1
```

pause (hit enter to continue)

1/cond(A_d) = 0.33333 so convert to standard eigenproblem for

A =

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

pause (hit enter to continue)

D =

```
0 + 1.0000i
0 - 1.0000i
1.0000
-1.0000
```

Eigenvalue

Error Bound

ans =

```
0 - 1.0000i  0.0000
1.0000      0.0000
0 + 1.0000i  0.0000
-1.0000     0.0000
```

pause (hit enter to continue)

all eigenvalues, x1 coords of intersections, error bounds

ans =

0 - 1.0000i	-1.0000	0.0000
1.0000	1.0000	0.0000
0 + 1.0000i	-1.0000	0.0000
-1.0000	1.0000	0.0000

all eigenvalues, x2 coords of intersections, error bounds

ans =

0 - 1.0000i	0	0.0000
1.0000	2.0000	0.0000
0 + 1.0000i	0	0.0000
-1.0000	2.0000	0.0000

all eigenvalues, x3 coords of intersections, error bounds

ans =

0 - 1.0000i	1.0000	0.0000
1.0000	3.0000	0.0000
0 + 1.0000i	1.0000	0.0000
-1.0000	3.0000	0.0000

pause (hit enter to continue)

real eigenvalues, x1 coords of intersections, error bounds

ans =

1.0000	1.0000	0.0000
-1.0000	1.0000	0.0000

real eigenvalues, x2 coords of intersections, error bounds

ans =

1.0000	2.0000	0.0000
-1.0000	2.0000	0.0000

real eigenvalues, x3 coords of intersections, error bounds

ans =

```
1.0000 3.0000 0.0000
-1.0000 3.0000 0.0000
```

Example 6 \small

Matrix_Poly_Coeffs =

```
3 0 0 0 3 1
2 1 0 0 1 1
```

pause (hit enter to continue)

dMatrix_Poly_Coeffs =

1.0e-014 *

```
0.1554 0 0 0 0.1998 0.0444
0.0888 0.7550 0 0 0.0444 0.6217
```

pause (hit enter to continue)

A =

```
0 0 -3 0
0 0 -2 -1
1 0 0 0
0 1 0 0
```

pause (hit enter to continue)

B =

```
3 1 0 0
1 1 0 0
0 0 1 0
0 0 0 1
```

pause (hit enter to continue)

1/cond(A_d) = 0.125 so convert to standard eigenproblem for

A =

```
0 0 -0.5000 0.5000
```

0	0	-1.5000	-1.5000
1.0000	0	0	0
0	1.0000	0	0

pause (hit enter to continue)

D =

-0.3352 + 1.0547i
-0.3352 - 1.0547i
0.3352 + 1.0547i
0.3352 - 1.0547i

Eigenvalue	Error Bound
------------	-------------

ans =

0.3352 - 1.0547i	0.0000
0.3352 + 1.0547i	0.0000
-0.3352 - 1.0547i	0.0000
-0.3352 + 1.0547i	0.0000

pause (hit enter to continue)

all eigenvalues, x1 coords of intersections, error bounds

ans =

0.3352 - 1.0547i	-1.0000 - 0.7071i	0.0000
0.3352 + 1.0547i	-1.0000 + 0.7071i	0.0000
-0.3352 - 1.0547i	-1.0000 + 0.7071i	0.0000
-0.3352 + 1.0547i	-1.0000 - 0.7071i	0.0000

all eigenvalues, x2 coords of intersections, error bounds

ans =

0.3352 - 1.0547i	0.0000 - 0.7071i	0.0000
0.3352 + 1.0547i	0.0000 + 0.7071i	0.0000
-0.3352 - 1.0547i	-0.0000 + 0.7071i	0.0000
-0.3352 + 1.0547i	-0.0000 - 0.7071i	0.0000

all eigenvalues, x3 coords of intersections, error bounds

ans =

```

0.3352 - 1.0547i    1.0000 - 0.7071i    0.0000
0.3352 + 1.0547i    1.0000 + 0.7071i    0.0000
-0.3352 - 1.0547i    1.0000 + 0.7071i    0.0000
-0.3352 + 1.0547i    1.0000 - 0.7071i    0.0000

```

pause (hit enter to continue)

real eigenvalues, x1 coords of intersections, error bounds

ans =

Empty matrix: 0-by-3

real eigenvalues, x2 coords of intersections, error bounds

ans =

Empty matrix: 0-by-3

real eigenvalues, x3 coords of intersections, error bounds

ans =

Empty matrix: 0-by-3

Example 7 The whole output is too much. Only the real eigenvalues and the intersection points are listed.

```

\small
real eigenvalues, all coords of intersections

```

ans =

```

      0      7.0000      1.0000      2.0000
      0      7.0000      1.0000      2.0000
      0      7.0000      1.0000      2.0000
      0      7.0000      1.0000      2.0000
    -0.0000      7.0000      1.0000      2.0000
    -0.5993      8.7206      1.2819      2.2819
    -0.7030      8.9372      1.3225      2.3225
    -0.8322      9.0975      1.2934      2.2934
    2.5359    104.2557    112.2938    113.2938

```