MATLAB PROJECT 4

Please include this page in your Group file, as a front page. Type in the group number and the names of all members WHO PARTICIPATED in this project.

GROUP # ____5______

FIRST & LAST NAMES (UFID numbers are NOT required):

_1. Manuel Vera

_2. Elton Li

_3. Corey Wolfe

_4. Dany Rashwan

_5. Corey Wolfe

_6. Thomas Pena

_7. Ekaterina Krysova

By including your names above, each of you had confirmed that you did the work and agree with the work submitted.

Part I. Eigenvalues, Eigenvectors, and Diagonalization

Exercise 1

```
type eigen
```

```
function [P,D]=eigen(A)
    %this function finds the eigenvalues of a given n*n matrix, orthonormal
  %bases for the corresponding eigenspaces, and its dimensions. Then, it
  %will check if the matrix is diagonalizable and return the invertible
  %matrix P and the diagonal matrix D.
    format compact
    [~,n]=size(A);

    %part 1. vector L of eigenvalues
    L = eig(A); %column vector of eigenvalues of A
    L = transpose(L); %converts L into a row vector
    L = sort(L); % sorts the entries of L in ascending order

    for i=1:(length(L)-1) %if two eigenvalues are equal within the given range, set them equal to each other
        difL = L(i)-L(i+1);
        if (closetozeroroundoff(difL, 7) == 0)
            L(i+1) = L(i);
        end
    end

    if rank(A) ~= n %checks if matrix A is singular
        for i=1:length(L)
            if (closetozeroroundoff(L(i), 7) == 0) %if an eigenvalue is 0 within the given range, set it to be 0
                L(i) = 0;
            end
        end
    end

    L

    %part 2. orthonormal basis W for each eigenspace
    M = unique(L) %creates a row vector of only the unique eigenvalues of A
    m = zeros(length(M)); %creates a vector of the multiplicity of each unique eigenvalue
    d = zeros(length(M)); %creates a vector for the dimension of the orthonormal basis W

    for i=1:length(M)
        count = 0; %starts the count for the multiplicity
        for j=1:length(L)
            if M(i) == L(j)
                count = count + 1;
            end
        end
        m(i) = count; %assigns the value of the multiplicity to the corresponding entry
        fprintf('Eigenvalue %d has multiplicity %i\n',M(i),m(i));

        nullMat = A - (M(i)*eye(size(A,1)));
        W = null(nullMat); %finds an orthonormal basis for the given eigenvalue
        fprintf('A basis for eigenvalue lambda = %d is:\n',M(i));
            W

        d(i) = rank(W); %determines the dimension of the eigenspace for the given eigenvalue
        fprintf('Dimension of eigenspace for lambda = %d is %i\n',M(i),d(i))
    end

    % part 3. construct diagonalization if possible
    for i=1:length(M)
        if m(i) ~= d(i) %checks if the matrix is not diagonalizable
            disp('The matrix A is not diagonalizable');
```

```
        P=[];
        D=[];
        return; %terminates the program if it isn't diagonalizable
    end
end

disp('The matrix A is diagonalizable');
P = zeros(n, length(L)); %initializes the invertible matrix P
for i=1:length(M)
    nullMat = A - (M(i)*eye(n));
    W = null(nullMat);
    if i == 1
        P = W;
    else
        P = horzcat(P, W); %sets the columns of matrix P to be the bases for each eigenvalue
    end
end
P
D = diag(L) %Creates the diagonal matrix D with the eigenvalues of A on its main diagonal

%now, it checks if the program works properly
AP = A*P;
PD = P*D;
difAPDP = AP-PD;
if closetozeroroundoff(difAPDP, 7) == zeros(size(AP, 1), size(AP, 2)) %checks if AP = DP
    if rank(P) == n %checks if P is invertible
        disp('Great! I got a diagonalization')
    else
        disp('Oops! I got a bug in my code!')
        return; %terminates the program if P is singular
    end
else
   disp('Oops! I got a bug in my code!')
   return; %terminates the program if AP ~= DP
end

%part 4. comparing function outputs with matlab outputs
[U, V] = eig(A); %default matlab function to diagonalize matrices
disp('U =');
disp(U);
disp('V =');
disp(V);

eqCheck = 0; %counter used to check if matrices P and U are equal
for i=1:size(P,2) %loop that iterates through the columns of P and U to see if they're equal
    for j=1:size(U, 2)
        difPU = P(:,i) - U(:, j);
        negPU = P(:,i) + U(:, j);
        if closetozeroroundoff(difPU, 7) == zeros(1, size(U, 2))
            eqCheck = eqCheck + 1; %if two columns match, increases the check counter
        elseif closetozeroroundoff(negPU, 7) == zeros(1, size(U,2))
            eqCheck = eqCheck + 1; %if two columns match to a scalar -1, increases the check counter
        end
    end
end
if eqCheck == size(P,2) %if all columns are equal, displays a message
    disp('Sets of columns of P and U are the same or match up to scalar (-1)');
else %if all columns don't match, displays a message
    disp('There is no specific match among the columns of P and U');
end

diagV = diag(V);
diagV = sort(diagV);
diagD = diag(D);
diagDif = diagV - diagD; %compares the diagonal matrices D and V
```

```
    if closetozeroroundoff(diagDif, 7) == zeros(1, length(diagDif))
        disp('The diagonal elements of D and V match');
    else %displays an error message if the diagonals are not equal
        disp('That cannot be true!');

end
```

```
type closetozeroroundoff
```

```
function B=closetozeroroundoff(A,p)
    A(abs(A)<10^-p)=0;
    B=A;
end
```

```
type jord
```

```
function J=jord(n,r)
    J = [];
 if (mod(n, 1) ~= 0 || n <= 1)
  disp('Jordan Block cannot be built')
  return
    end
 J = diag(diag(eye(n - 1)), 1) + r * eye(n);
end
```

```
format

%(a)
A=[3 3; 0 3]
```

```
A = 2×2
     3     3
     0     3
```

```
eigen(A);
```

```
L = 1×2
     3     3
M = 3
Eigenvalue 3 has multiplicity 2
A basis for eigenvalue lambda = 3 is:
W = 2×1
    -1
     0
Dimension of eigenspace for lambda = 3 is 1
The matrix A is not diagonalizable
```

```
%(b)
A=[4 0 0 0; 1 3 0 0; 0 -1 3 0; 0 -1 5 4]
```

```
A = 4×4
     4     0     0     0
     1     3     0     0
     0    -1     3     0
     0    -1     5     4
```

```
eigen(A);
```

```
L = 1×4
     3     3     4     4
```

```
M = 1×2
     3     4
Eigenvalue 3 has multiplicity 2
A basis for eigenvalue lambda = 3 is:
W = 4×1
         0
         0
   -0.1961
    0.9806
Dimension of eigenspace for lambda = 3 is 1
Eigenvalue 4 has multiplicity 2
A basis for eigenvalue lambda = 4 is:
W = 4×1
     0
     0
     0
     1
Dimension of eigenspace for lambda = 4 is 1
The matrix A is not diagonalizable
```

```
%(c)
A=jord(5,5)
```

```
A = 5×5
     5     1     0     0     0
     0     5     1     0     0
     0     0     5     1     0
     0     0     0     5     1
     0     0     0     0     5
```

```
eigen(A);
```

```
L = 1×5
     5     5     5     5     5
M = 5
Eigenvalue 5 has multiplicity 5
A basis for eigenvalue lambda = 5 is:
W = 5×1
     1
     0
     0
     0
     0
Dimension of eigenspace for lambda = 5 is 1
The matrix A is not diagonalizable
```

```
% (d)
A=diag([3, 3, 3, 2, 2, 1])
```

```
A = 6×6
     3     0     0     0     0     0
     0     3     0     0     0     0
     0     0     3     0     0     0
     0     0     0     2     0     0
     0     0     0     0     2     0
     0     0     0     0     0     1
```

```
eigen(A);
```

```
L = 1×6
     1     2     2     3     3     3
M = 1×3
     1     2     3
```

```
Eigenvalue 1 has multiplicity 1
A basis for eigenvalue lambda = 1 is:
W = 6×1
     0
     0
     0
     0
     0
     1
Dimension of eigenspace for lambda = 1 is 1
Eigenvalue 2 has multiplicity 2
A basis for eigenvalue lambda = 2 is:
W = 6×2
     0     0
     0     0
     0     0
     1     0
     0     1
     0     0
Dimension of eigenspace for lambda = 2 is 2
Eigenvalue 3 has multiplicity 3
A basis for eigenvalue lambda = 3 is:
W = 6×3
     0     0     1
     1     0     0
     0     1     0
     0     0     0
     0     0     0
     0     0     0
Dimension of eigenspace for lambda = 3 is 3
The matrix A is diagonalizable
P = 6×6
     0     0     0     0     0     1
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     1     0     0     0     0     0
D = 6×6
     1     0     0     0     0     0
     0     2     0     0     0     0
     0     0     2     0     0     0
     0     0     0     3     0     0
     0     0     0     0     3     0
     0     0     0     0     0     3
Great! I got a diagonalization
U =
     0     0     0     0     0     1
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     1     0     0     0     0     0
V =
     1     0     0     0     0     0
     0     2     0     0     0     0
     0     0     2     0     0     0
     0     0     0     3     0     0
     0     0     0     0     3     0
     0     0     0     0     0     3
Sets of columns of P and U are the same or match up to scalar (-1)
The diagonal elements of D and V match
```

```
% (e)
```

```
A=magic(4)
```

```
A = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
eigen(A);
```

```
L = 1×4
   -8.9443         0    8.9443   34.0000
M = 1×4
   -8.9443         0    8.9443   34.0000
Eigenvalue -8.944272e+00 has multiplicity 1
A basis for eigenvalue lambda = -8.944272e+00 is:
W = 4×1
   -0.3764
   -0.0236
   -0.4236
    0.8236
Dimension of eigenspace for lambda = -8.944272e+00 is 1
Eigenvalue 0 has multiplicity 1
A basis for eigenvalue lambda = 0 is:
W = 4×1
    0.2236
    0.6708
   -0.6708
   -0.2236
Dimension of eigenspace for lambda = 0 is 1
Eigenvalue 8.944272e+00 has multiplicity 1
A basis for eigenvalue lambda = 8.944272e+00 is:
W = 4×1
    0.8236
   -0.4236
   -0.0236
   -0.3764
Dimension of eigenspace for lambda = 8.944272e+00 is 1
Eigenvalue 3.400000e+01 has multiplicity 1
A basis for eigenvalue lambda = 3.400000e+01 is:
W = 4×1
    0.5000
    0.5000
    0.5000
    0.5000
Dimension of eigenspace for lambda = 3.400000e+01 is 1
The matrix A is diagonalizable
P = 4×4
   -0.3764    0.2236    0.8236    0.5000
   -0.0236    0.6708   -0.4236    0.5000
   -0.4236   -0.6708   -0.0236    0.5000
    0.8236   -0.2236   -0.3764    0.5000
D = 4×4
   -8.9443         0         0         0
         0         0         0         0
         0         0    8.9443         0
         0         0         0   34.0000
Great! I got a diagonalization
U =
   -0.5000   -0.8236    0.3764   -0.2236
   -0.5000    0.4236    0.0236   -0.6708
   -0.5000    0.0236    0.4236    0.6708
   -0.5000    0.3764   -0.8236    0.2236
V =
```

```
   34.0000         0         0         0
         0    8.9443         0         0
         0         0   -8.9443         0
         0         0         0    0.0000
Sets of columns of P and U are the same or match up to scalar (-1)
The diagonal elements of D and V match
```

```
% (f)
A=ones(4)
```

```
A = 4×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
eigen(A);
```

```
L = 1×4
         0         0         0    4.0000
M = 1×2
         0    4.0000
Eigenvalue 0 has multiplicity 3
A basis for eigenvalue lambda = 0 is:
W = 4×3
         0         0    0.8660
   -0.5774   -0.5774   -0.2887
    0.7887   -0.2113   -0.2887
   -0.2113    0.7887   -0.2887
Dimension of eigenspace for lambda = 0 is 3
Eigenvalue 4.000000e+00 has multiplicity 1
A basis for eigenvalue lambda = 4.000000e+00 is:
W = 4×1
   -0.5000
   -0.5000
   -0.5000
   -0.5000
Dimension of eigenspace for lambda = 4.000000e+00 is 1
The matrix A is diagonalizable
P = 4×4
         0         0    0.8660   -0.5000
   -0.5774   -0.5774   -0.2887   -0.5000
    0.7887   -0.2113   -0.2887   -0.5000
   -0.2113    0.7887   -0.2887   -0.5000
D = 4×4
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0    4.0000
Great! I got a diagonalization
U =
    0.0846    0.4928    0.7071    0.5000
    0.0846    0.4928   -0.7071    0.5000
   -0.7815   -0.3732         0    0.5000
    0.6124   -0.6124         0    0.5000
V =
   -0.0000         0         0         0
         0   -0.0000         0         0
         0         0         0         0
         0         0         0    4.0000
There is no specific match among the columns of P and U
The diagonal elements of D and V match
```

```
%(g)
```

```
A=magic(5)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
eigen(A);
```

```
L = 1×5
  -21.2768  -13.1263   13.1263   21.2768   65.0000
M = 1×5
  -21.2768  -13.1263   13.1263   21.2768   65.0000
Eigenvalue -2.127677e+01 has multiplicity 1
A basis for eigenvalue lambda = -2.127677e+01 is:
W = 5×1
   -0.0976
   -0.3525
   -0.5501
    0.3223
    0.6780
Dimension of eigenspace for lambda = -2.127677e+01 is 1
Eigenvalue -1.312628e+01 has multiplicity 1
A basis for eigenvalue lambda = -1.312628e+01 is:
W = 5×1
   -0.6330
    0.5895
   -0.3915
    0.1732
    0.2619
Dimension of eigenspace for lambda = -1.312628e+01 is 1
Eigenvalue 1.312628e+01 has multiplicity 1
A basis for eigenvalue lambda = 1.312628e+01 is:
W = 5×1
    0.2619
    0.1732
   -0.3915
    0.5895
   -0.6330
Dimension of eigenspace for lambda = 1.312628e+01 is 1
Eigenvalue 2.127677e+01 has multiplicity 1
A basis for eigenvalue lambda = 2.127677e+01 is:
W = 5×1
    0.6780
    0.3223
   -0.5501
   -0.3525
   -0.0976
Dimension of eigenspace for lambda = 2.127677e+01 is 1
Eigenvalue 6.500000e+01 has multiplicity 1
A basis for eigenvalue lambda = 6.500000e+01 is:
W = 5×1
   -0.4472
   -0.4472
   -0.4472
   -0.4472
   -0.4472
Dimension of eigenspace for lambda = 6.500000e+01 is 1
The matrix A is diagonalizable
P = 5×5
   -0.0976   -0.6330    0.2619    0.6780   -0.4472
   -0.3525    0.5895    0.1732    0.3223   -0.4472
```

```
   -0.5501   -0.3915   -0.3915   -0.5501   -0.4472
    0.3223    0.1732    0.5895   -0.3525   -0.4472
    0.6780    0.2619   -0.6330   -0.0976   -0.4472
D = 5×5
  -21.2768         0         0         0         0
         0  -13.1263         0         0         0
         0         0   13.1263         0         0
         0         0         0   21.2768         0
         0         0         0         0   65.0000
Great! I got a diagonalization
U =
   -0.4472    0.0976   -0.6330    0.6780   -0.2619
   -0.4472    0.3525    0.5895    0.3223   -0.1732
   -0.4472    0.5501   -0.3915   -0.5501    0.3915
   -0.4472   -0.3223    0.1732   -0.3525   -0.5895
   -0.4472   -0.6780    0.2619   -0.0976    0.6330
V =
   65.0000         0         0         0         0
         0  -21.2768         0         0         0
         0         0  -13.1263         0         0
         0         0         0   21.2768         0
         0         0         0         0   13.1263
Sets of columns of P and U are the same or match up to scalar (-1)
The diagonal elements of D and V match
```

```
%(h)
A=hilb(7)
```

```
A = 7×7
    1.0000    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429
    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250
    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111
    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000
    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909
    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833
    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769
```

```
eigen(A);
```

```
L = 1×7
    0.0000    0.0000    0.0000    0.0010    0.0213    0.2719    1.6609
M = 1×7
    0.0000    0.0000    0.0000    0.0010    0.0213    0.2719    1.6609
Eigenvalue 3.493899e-09 has multiplicity 1
A basis for eigenvalue lambda = 3.493899e-09 is:
W = 7×1
   -0.0002
    0.0098
   -0.0952
    0.3713
   -0.6825
    0.5910
   -0.1944
Dimension of eigenspace for lambda = 3.493899e-09 is 1
Eigenvalue 4.856763e-07 has multiplicity 1
A basis for eigenvalue lambda = 4.856763e-07 is:
W = 7×1
   -0.0025
    0.0618
   -0.3487
    0.6447
   -0.1744
   -0.5436
    0.3647
```

```
Dimension of eigenspace for lambda = 4.856763e-07 is 1
Eigenvalue 2.938637e-05 has multiplicity 1
A basis for eigenvalue lambda = 2.938637e-05 is:
W = 7×1
    0.0160
   -0.2279
    0.6288
   -0.2004
   -0.4970
   -0.1849
    0.4808
Dimension of eigenspace for lambda = 2.938637e-05 is 1
Eigenvalue 1.008588e-03 has multiplicity 1
A basis for eigenvalue lambda = 1.008588e-03 is:
W = 7×1
    0.0752
   -0.5268
    0.4257
    0.4617
    0.1712
   -0.1827
   -0.5098
Dimension of eigenspace for lambda = 1.008588e-03 is 1
Eigenvalue 2.128975e-02 has multiplicity 1
A basis for eigenvalue lambda = 2.128975e-02 is:
W = 7×1
   -0.2608
    0.6706
    0.2953
   -0.0230
   -0.2337
   -0.3679
   -0.4523
Dimension of eigenspace for lambda = 2.128975e-02 is 1
Eigenvalue 2.719202e-01 has multiplicity 1
A basis for eigenvalue lambda = 2.719202e-01 is:
W = 7×1
    0.6232
   -0.1631
   -0.3215
   -0.3574
   -0.3571
   -0.3446
   -0.3281
Dimension of eigenspace for lambda = 2.719202e-01 is 1
Eigenvalue 1.660885e+00 has multiplicity 1
A basis for eigenvalue lambda = 1.660885e+00 is:
W = 7×1
   -0.7332
   -0.4364
   -0.3198
   -0.2549
   -0.2128
   -0.1831
   -0.1609
Dimension of eigenspace for lambda = 1.660885e+00 is 1
The matrix A is diagonalizable
P = 7×7
   -0.0002   -0.0025    0.0160    0.0752   -0.2608    0.6232   -0.7332
    0.0098    0.0618   -0.2279   -0.5268    0.6706   -0.1631   -0.4364
   -0.0952   -0.3487    0.6288    0.4257    0.2953   -0.3215   -0.3198
    0.3713    0.6447   -0.2004    0.4617   -0.0230   -0.3574   -0.2549
   -0.6825   -0.1744   -0.4970    0.1712   -0.2337   -0.3571   -0.2128
    0.5910   -0.5436   -0.1849   -0.1827   -0.3679   -0.3446   -0.1831
   -0.1944    0.3647    0.4808   -0.5098   -0.4523   -0.3281   -0.1609
```

```
D = 7×7
    0.0000         0         0         0         0         0         0
         0    0.0000         0         0         0         0         0
         0         0    0.0000         0         0         0         0
         0         0         0    0.0010         0         0         0
         0         0         0         0    0.0213         0         0
         0         0         0         0         0    0.2719         0
         0         0         0         0         0         0    1.6609
Great! I got a diagonalization
U =
  Columns 1 through 6
    0.0002   -0.0025    0.0160    0.0752    0.2608   -0.6232
   -0.0098    0.0618   -0.2279   -0.5268   -0.6706    0.1631
    0.0952   -0.3487    0.6288    0.4257   -0.2953    0.3215
   -0.3713    0.6447   -0.2004    0.4617    0.0230    0.3574
    0.6825   -0.1744   -0.4970    0.1712    0.2337    0.3571
   -0.5910   -0.5436   -0.1849   -0.1827    0.3679    0.3446
    0.1944    0.3647    0.4808   -0.5098    0.4523    0.3281
  Column 7
    0.7332
    0.4364
    0.3198
    0.2549
    0.2128
    0.1831
    0.1609
V =
  Columns 1 through 6
    0.0000         0         0         0         0         0
         0    0.0000         0         0         0         0
         0         0    0.0000         0         0         0
         0         0         0    0.0010         0         0
         0         0         0         0    0.0213         0
         0         0         0         0         0    0.2719
         0         0         0         0         0         0
  Column 7
         0
         0
         0
         0
         0
         0
    1.6609
Sets of columns of P and U are the same or match up to scalar (-1)
The diagonal elements of D and V match
```

```
%(k)
A=[5 8 -4;8 5 -4;-4 -4 -1]
```

```
A = 3×3
     5     8    -4
     8     5    -4
    -4    -4    -1
```

```
eigen(A);
```

```
L = 1×3
   -3.0000   -3.0000   15.0000
M = 1×2
   -3.0000   15.0000
Eigenvalue -3.000000e+00 has multiplicity 2
A basis for eigenvalue lambda = -3.000000e+00 is:
W = 3×2
   -0.0619    0.7428
```

```
    0.4952   -0.5571
    0.8666    0.3714
Dimension of eigenspace for lambda = -3.000000e+00 is 2
Eigenvalue 1.500000e+01 has multiplicity 1
A basis for eigenvalue lambda = 1.500000e+01 is:
W = 3×1
   -0.6667
   -0.6667
    0.3333
Dimension of eigenspace for lambda = 1.500000e+01 is 1
The matrix A is diagonalizable
P = 3×3
   -0.0619    0.7428   -0.6667
    0.4952   -0.5571   -0.6667
    0.8666    0.3714    0.3333
D = 3×3
   -3.0000         0         0
         0   -3.0000         0
         0         0   15.0000
Great! I got a diagonalization
U =
    0.2902    0.6865    0.6667
    0.1797   -0.7234    0.6667
    0.9399   -0.0737   -0.3333
V =
   -3.0000         0         0
         0   -3.0000         0
         0         0   15.0000
There is no specific match among the columns of P and U
The diagonal elements of D and V match
```

Exercise 2

```
type closetozeroroundoff
```

```
function B=closetozeroroundoff(A,p)
    A(abs(A)<10^-p)=0;
    B=A;
end
```

```
type symmetric
```

```
%Creates the function symmetric
function [] = symmetric(A)

n = size(A,1);      %Creates nxn matrix
p = 7;              %p=7 for closetozeroroundoff function
x = isequal(A,A');  %Checks for symmetry statement

%If not, outputs msg that matrix is not symmetric and terminates program
if x == 0
    fprintf('A is not symmetric')
    return
end

%Constructs an orthogonal diagonalization
[P,D] = eig(A)

%Verifies for orthogonal diagonalization
if closetozeroroundoff(A*P-P*D,p) == zeros(n) & ...
        closetozeroroundoff(inv(P)-P',p) == zeros(n)
    disp('AP = PD and P is orthogonal')
else
    disp('What is wrong?!')
```

```
end
end %end of function
```

```
%(a)
A=[2 -1 1;-1 2 -1;1 -1 2]
```

```
A = 3×3
     2    -1     1
    -1     2    -1
     1    -1     2
```

```
symmetric(A)
```

```
P = 3×3
    0.4082    0.7071   -0.5774
   -0.4082    0.7071    0.5774
   -0.8165         0   -0.5774
D = 3×3
    1.0000         0         0
         0    1.0000         0
         0         0    4.0000
AP = PD and P is orthogonal
```

```
%(b)
A=[2 -1 1;-1 2 -2;1 -1 2]
```

```
A = 3×3
     2    -1     1
    -1     2    -2
     1    -1     2
```

```
symmetric(A)
```

```
A is not symmetric
```

```
%(c)
B=A*A'
```

```
B = 3×3
     6    -6     5
    -6     9    -7
     5    -7     6
```

```
symmetric(B)
```

```
P = 3×3
   -0.0820    0.8577    0.5075
    0.5912    0.4518   -0.6681
    0.8024   -0.2453    0.5441
D = 3×3
    0.3315         0         0
         0    1.4096         0
         0         0   19.2588
AP = PD and P is orthogonal
```

```
%(d)
A=[3 1 1;1 3 1;1 1 3]
```

```
A = 3×3
     3     1     1
     1     3     1
```

```
     1     1     3
```

```
symmetric(A)
```

```
P = 3×3
    0.4082    0.7071    0.5774
    0.4082   -0.7071    0.5774
   -0.8165         0    0.5774
D = 3×3
    2.0000         0         0
         0    2.0000         0
         0         0    5.0000
AP = PD and P is orthogonal
```

```
%(e)
A=[5 8 -4;8 5 -4;-4 -4 -1]
```

```
A = 3×3
     5     8    -4
     8     5    -4
    -4    -4    -1
```

```
symmetric(A)
```

```
P = 3×3
    0.2902    0.6865    0.6667
    0.1797   -0.7234    0.6667
    0.9399   -0.0737   -0.3333
D = 3×3
   -3.0000         0         0
         0   -3.0000         0
         0         0   15.0000
AP = PD and P is orthogonal
```

```
%(f)
A=[4 3 1 1; 3 4 1 1 ; 1 1 4 3; 1 1 3 4]
```

```
A = 4×4
     4     3     1     1
     3     4     1     1
     1     1     4     3
     1     1     3     4
```

```
symmetric(A)
```

```
P = 4×4
   -0.0000    0.7071   -0.5000    0.5000
         0   -0.7071   -0.5000    0.5000
   -0.7071    0.0000    0.5000    0.5000
    0.7071         0    0.5000    0.5000
D = 4×4
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    5.0000         0
         0         0         0    9.0000
AP = PD and P is orthogonal
```

Part II. Orthogonal Projections & Least-Squares Solutions

Exercise 3

```
type closetozeroroundoff.m
```

```
function B=closetozeroroundoff(A,p)
    A(abs(A)<10^-p)=0;
    B=A;
end
```

```
type proj.m
```

```
function [p,z]=proj(A,b)
format compact
A=shrink(A);
m=size(A,1);

if m ~= prod(size(b))
    disp('No solution: dimensions of A and b disagree')
    p = []
    z = []
    return;
end

if rank(A) == rank([A b])
        p = b

        z = 0

        disp('b is in Col A')
        return;
end


a = colspace(sym(A));
t = 0;
for i = 1:size(a,2)
    t = t + abs(dot(a(:,i),b));
end
t = closetozeroroundoff(t, 7);
if t == 0
    z = b
    p = z - b

    disp('b is orthogonal to Col A')
    return;
end


if t ~= 0
    x = pinv(A)*b;
    disp('the least squares solution of the system is')
    x
    x1 = A \ b



    h = isequal(closetozeroroundoff(x - x1,12), zeros(size(x, 1), size(x, 2)));

    if h == 1
        disp('A\b returns the least-squares solution of an inconsistent system Ax = b')
    end
end

p = A * x
z = b - p
```

```
a = colspace(sym(A));
t = 0;
for i = 1:size(a,2)
 t = t + abs(dot(a(:,i),z));
end
t = closetozeroroundoff(t, 7);
if t == 0
    disp('z is orthogonal to Col A! Great Job!')
else
    disp('Oops! Is there a bug in my code?')
end
d = norm(b - p);
fprintf('the distance from b to Col A is %i', d)
end
```

```
type shrink.m
```

```
function B=shrink(A)
format compact
[~,pivot]=rref(A);
B=A(:,pivot);
end
```

```
%(a)
A=magic(4), b=sum(A,2)
```

```
A = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
b = 4×1
    34
    34
    34
    34
```

```
proj(A,b)
```

```
p = 4×1
    34
    34
    34
    34
z = 0
b is in Col A
ans = 4×1
    34
    34
    34
    34
```

```
%(b)
A=magic(4); A=A(:,1:3),b=(1:4)'
```

```
A = 4×3
    16     2     3
     5    11    10
     9     7     6
     4    14    15
b = 4×1
     1
```

```
     2
     3
     4
```

```
proj(A,b)
```

```
the least squares solution of the system is
x = 3×1
    0.0471
    0.1941
    0.0529
x1 = 3×1
    0.0471
    0.1941
    0.0529
A\b returns the least-squares solution of an inconsistent system Ax = b
p = 4×1
    1.3000
    2.9000
    2.1000
    3.7000
z = 4×1
   -0.3000
   -0.9000
    0.9000
    0.3000
z is orthogonal to Col A! Great Job!
the distance from b to Col A is 1.341641e+00
ans = 4×1
    1.3000
    2.9000
    2.1000
    3.7000
```

```
%(c)
A=magic(6), E=eye(6); b=E(:,6)
```

```
A = 6×6
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
b = 6×1
     0
     0
     0
     0
     0
     1
```

```
proj(A,b)
```

```
the least squares solution of the system is
x = 5×1
    0.1328
    0.1420
   -0.0595
   -0.1089
   -0.0973
x1 = 5×1
    0.1328
    0.1420
```

```
   -0.0595
   -0.1089
   -0.0973
A\b returns the least-squares solution of an inconsistent system Ax = b
p = 6×1
   -0.2500
    0.0000
    0.2500
    0.2500
    0.0000
    0.7500
z = 6×1
    0.2500
   -0.0000
   -0.2500
   -0.2500
   -0.0000
    0.2500
z is orthogonal to Col A! Great Job!
the distance from b to Col A is 5.000000e-01
ans = 6×1
   -0.2500
    0.0000
    0.2500
    0.2500
    0.0000
    0.7500
```

```
%(d)
A=magic(6), b=(1:5)'
```

```
A = 6×6
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
b = 5×1
     1
     2
     3
     4
     5
```

```
proj(A,b)
```

```
No solution: dimensions of A and b disagree
p =
     []
z =
     []
ans =
     []
```

```
%(e)
A=magic(5), b = rand(5,1)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
b = 5×1
    0.0975
    0.2785
    0.5469
    0.9575
    0.9649
```

```
proj(A,b)
```

```
p = 5×1
    0.0975
    0.2785
    0.5469
    0.9575
    0.9649
z = 0
b is in Col A
ans = 5×1
    0.0975
    0.2785
    0.5469
    0.9575
    0.9649
```

```
%(f)
A=ones(4); A(:)=1:16, b=[1;0;1;0]
```

```
A = 4×4
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
b = 4×1
     1
     0
     1
     0
```

```
proj(A,b)
```

```
the least squares solution of the system is
x = 2×1
   -0.4500
    0.2500
x1 = 2×1
   -0.4500
    0.2500
A\b returns the least-squares solution of an inconsistent system Ax = b
p = 4×1
    0.8000
    0.6000
    0.4000
    0.2000
z = 4×1
    0.2000
   -0.6000
    0.6000
   -0.2000
z is orthogonal to Col A! Great Job!
the distance from b to Col A is 8.944272e-01
ans = 4×1
    0.8000
    0.6000
    0.4000
```

```
    0.2000
```

```
%(g)
B=ones(4); B(:)=1:16, A=null(B,'r'), b=ones(4,1)
```

```
B = 4×4
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
A = 4×2
     1     2
    -2    -3
     1     0
     0     1
b = 4×1
     1
     1
     1
     1
```

```
proj(A,b)
```

```
z = 4×1
     1
     1
     1
     1
p = 4×1
     0
     0
     0
     0
b is orthogonal to Col A
ans = 4×1
     0
     0
     0
     0
```

```
%In example e you can see that random vector b is in Col A. So why is a
%random vector consistently in Col A? Because the rank of A b is equivalent
%to rank A. They have the same number of linearly independent row
%vectors, which means they have the same rank. Which means b is in Col A.
```

Exercise 4

```
type closetozeroroundoff
```

```
function B=closetozeroroundoff(A,p)
    A(abs(A)<10^-p)=0;
    B=A;
end
```

```
type shrink
```

```
function B=shrink(A)
format compact
[~,pivot]=rref(A);
B=A(:,pivot);
```

```
end
```

```
type solvemore
```

```
function X = solvemore(A,b)
    format compact
    format long
    A=shrink(A);
    [m,n]=size(A);

    if rank([A b]) == rank(A)
        fprintf('The system is consistent - look for the exact soltion\n');
        x1 = A\b;
        if closetozeroroundoff(A'*A-eye(n),7)==0
            if m==n
                fprintf('A is orthogonal\n');
                x2 = A'*b;
            else
                fprintf('A has orthonormal columns but is not orthogonal\n');
                x2 = zeros(n,1);
                for i=1:n
                    x2(i) = dot(b,A(:,i))/dot(A(:,i),A(:,i));
                end
            end
            X = [x1,x2];
            N = norm(x1-x2);
            fprintf('The norm of difference between solutions is \n');
            disp(N);
        else
            fprintf('A does not have orthonormal columns.\n');
            X = x1;
        return
        end
    else
        fprintf('The system is inconsistent: look for the leastsquares solution\n');

        fprintf('The least-squares solution of the system is\n');
        x1 = (A'*A)\(A'*b);
        disp(x1);

        if closetozeroroundoff(A'*A-eye(n),7)==0
            fprintf('A has orthonormal columns: an orthonormal basis for ColA is U=A\n');
            U=A;
        else
            fprintf('A does not have orthonormal columns: orthonormal basis for Col A is\n');
            U=orth(A);
            disp(U);
        end

        fprintf('The projection of b onto Col A is:\n');
        b1 = U*U'*b;
        disp(b1);

        fprintf('The least-squares solution using the projection b1 is\n');
        x2 = A\b1;
        disp(x2);

        fprintf('Error of approximation of b by vector A*x1 of Col A is\n');
        n1 = norm(b-A*x1);
        disp(n1);

        fprintf('Error of approximation of b by vector b1 of Col A is\n');
        n2 = norm(x1-x2);
        disp(n2);
```

```
        fprintf('Error of approximation of b by Ax for a random vector x is\n');
        x = rand(n,1);
        n3 = norm(b-A*x);
        disp(n3);
    end

    if closetozeroroundoff(x2 - x1,12)==0
        fprintf('Solutions x1 and x2 are sufficiently close to each other\n');
        X = [x1, x2];
        disp(X);
    else
        fprintf('Check the code!\n');
        X = [];
        disp(X);
    end
end
```

```
%(a)
A=magic(4); b=A(:,4), A=orth(A)
```

```
b = 4×1
    13
     8
    12
     1
A = 4×3
   -0.5000    0.6708    0.5000
   -0.5000   -0.2236   -0.5000
   -0.5000    0.2236   -0.5000
   -0.5000   -0.6708    0.5000
```

```
X=solvemore(A,b)
```

```
The system is consistent - look for the exact soltion
A has orthonormal columns but is not orthogonal
The norm of difference between solutions is
     6.541842562171600e-15
Solutions x1 and x2 are sufficiently close to each other
 -16.999999999999996 -17.000000000000000
   8.944271909999152   8.944271909999157
  -3.000000000000001  -3.000000000000002
X = 3×2
 -16.999999999999996 -17.000000000000000
   8.944271909999152   8.944271909999157
  -3.000000000000001  -3.000000000000002
```

```
%(b)
A=magic(5); A=orth(A), b=rand(5,1)
```

```
A = 5×5
  -0.447213595499958  -0.545634873129948   0.511667273601714   0.195439507584854 ...
  -0.447213595499958  -0.449758363151205  -0.195439507584838  -0.511667273601691
  -0.447213595499958  -0.000000000000024  -0.632455532033676   0.632455532033676
  -0.447213595499958   0.449758363151189  -0.195439507584872  -0.511667273601694
  -0.447213595499958   0.545634873129987   0.511667273601672   0.195439507584856
b = 5×1
   0.157613081677548
   0.970592781760616
   0.957166948242946
   0.485375648722841
   0.800280468888800
```

```
X=solvemore(A,b)
```

```
The system is consistent - look for the exact soltion
A is orthogonal
The norm of difference between solutions is
     1.133654732222118e-15
Solutions x1 and x2 are sufficiently close to each other
  -1.507569968003384  -1.507569968003385
   0.132431274757778   0.132431274757778
  -0.399796503189899  -0.399796503189899
   0.047604378061833   0.047604378061833
   0.553796420948342   0.553796420948342
X = 5×2
  -1.507569968003384  -1.507569968003385
   0.132431274757778   0.132431274757778
  -0.399796503189899  -0.399796503189899
   0.047604378061833   0.047604378061833
   0.553796420948342   0.553796420948342
```

```
%(c)
A=magic(6); A=shrink(A), b=ones(6,1)
```

```
A = 6×5
    35     1     6    26    19
     3    32     7    21    23
    31     9     2    22    27
     8    28    33    17    10
    30     5    34    12    14
     4    36    29    13    18
b = 6×1
     1
     1
     1
     1
     1
     1
```

```
X=solvemore(A,b)
```

```
The system is consistent - look for the exact soltion
A does not have orthonormal columns.
X = 5×1
  -0.009009009009009
  -0.009009009009009
   0.018018018018018
   0.027027027027027
   0.027027027027027
```

```
%(d)
A=magic(6); A=shrink(A), b=rand(6,1)
```

```
A = 6×5
    35     1     6    26    19
     3    32     7    21    23
    31     9     2    22    27
     8    28    33    17    10
    30     5    34    12    14
     4    36    29    13    18
b = 6×1
   0.141886338627215
   0.421761282626275
   0.915735525189067
   0.792207329559554
```

```
   0.959492426392903
   0.655740699156587
```

```
X=solvemore(A,b)
```

```
The system is inconsistent: look for the leastsquares solution
The least-squares solution of the system is
   0.022840174018124
   0.016496321030778
   0.009099069923706
  -0.035152564553246
   0.021733428423445
A does not have orthonormal columns: orthonormal basis for Col A is
  Columns 1 through 3
  -0.377769194009722   0.565549747316776  -0.073442240091156
  -0.379043550198700  -0.256244279349781  -0.603252569283163
  -0.395564595880258   0.450811783981507  -0.334517803895046
  -0.427525218534308  -0.371500853605915   0.254692519662554
  -0.419591236986113   0.191885425980110   0.673707199810364
  -0.445320620404844  -0.486238816941184  -0.006383044141336
  Columns 4 through 5
  -0.522922149340345  -0.092750107637909
  -0.034288635982761   0.652361466864025
   0.402693937444365  -0.340929343900302
  -0.573273011514828  -0.188941996347896
   0.332305160199707   0.472032908354090
   0.352343075269880  -0.437121232610284
The projection of b onto Col A is:
   0.369465292868420
   0.421761282626274
   0.688156570947863
   0.564628375318350
   0.959492426392903
   0.883319653397792
The least-squares solution using the projection b1 is
   0.022840174018125
   0.016496321030778
   0.009099069923706
  -0.035152564553246
   0.021733428423444
Error of approximation of b by vector A*x1 of Col A is
   0.455157908482410
Error of approximation of b by vector b1 of Col A is
     1.627404550899882e-15
Error of approximation of b by Ax for a random vector x is
     1.501444623728036e+02
Solutions x1 and x2 are sufficiently close to each other
   0.022840174018124   0.022840174018125
   0.016496321030778   0.016496321030778
   0.009099069923706   0.009099069923706
  -0.035152564553246  -0.035152564553246
   0.021733428423445   0.021733428423444
X = 5×2
   0.022840174018124   0.022840174018125
   0.016496321030778   0.016496321030778
   0.009099069923706   0.009099069923706
  -0.035152564553246  -0.035152564553246
   0.021733428423445   0.021733428423444
```

```
%(e)
A=magic(4); A=orth(A), b=rand(4,1)
```

```
A = 4×3
  -0.500000000000000   0.670820393249937   0.500000000000000
```

```
  -0.500000000000000  -0.223606797749979  -0.500000000000000
  -0.500000000000000   0.223606797749979  -0.500000000000000
  -0.500000000000000  -0.670820393249937   0.500000000000000
b = 4×1
   0.743132468124916
   0.392227019534168
   0.655477890177557
   0.171186687811562
```

```
X=solvemore(A,b)
```

```
The system is inconsistent: look for the leastsquares solution
The least-squares solution of the system is
  -0.981012032824101
   0.442537577456909
  -0.066692876887623
A has orthonormal columns: an orthonormal basis for ColA is U=A
The projection of b onto Col A is:
   0.754022809705757
   0.424898044276690
   0.622806865435035
   0.160296346230721
The least-squares solution using the projection b1 is
  -0.981012032824101
   0.442537577456908
  -0.066692876887624
Error of approximation of b by vector A*x1 of Col A is
   0.048703088145904
Error of approximation of b by vector b1 of Col A is
     3.723801229870910e-16
Error of approximation of b by Ax for a random vector x is
   1.770674258734404
Solutions x1 and x2 are sufficiently close to each other
  -0.981012032824101  -0.981012032824101
   0.442537577456909   0.442537577456908
  -0.066692876887623  -0.066692876887624
X = 3×2
  -0.981012032824101  -0.981012032824101
   0.442537577456909   0.442537577456908
  -0.066692876887623  -0.066692876887624
```

```
% The output of approx. of n1 is smaller than that of n2.
% Comparing n1 and n3 shows that using x1 as our least square solution does
% indeed minimize the distance between vector b and vectors Ax of Col A.
```

Part III. Application to Polynomials

Exercise 5

```
type lstsqline.m
```

```
function c=lstsqline(x,y)
hold off
format
format compact
x=x';
y=y';
a=x(1);
m=length(x);
b=x(m);
disp('the design matrix is')
```

```
X=[x,ones(m,1)]
disp('the parameter vector is')
c=lscov(X,y)
disp('the norm of the residual vector is')
N=norm(y-X*c)
plot(x,y,'*'),hold on
polyplot(a,b,c');
fprintf('the least-squares regression line is\n')
P=poly2sym(c)

c1 = (inv(X'*X))*(X'*y);

h = closetozeroroundoff(c - c1, 7);

if (h == zeros(size(c)))
    disp('c is the least-squares solution')
end

hold off
end
```

```
type polyplot.m
```

```
function []=polyplot(a,b,p)
x=(a:(b-a)/50:b)';
y=polyval(p,x);
plot(x,y);
end
```

```
x = [0,2,3,5,6], y = [1,4,3,4,5]
```

```
x = 1×5
     0     2     3     5     6
y = 1×5
     1     4     3     4     5
```

```
c=lstsqline(x,y);
```

```
Warning: MATLAB has disabled some advanced graphics rendering features by switching to software OpenGL. For
 more information, click here.
the design matrix is
X = 5×2
     0     1
     2     1
     3     1
     5     1
     6     1
the parameter vector is
c = 2×1
    0.5526
    1.6316
the norm of the residual vector is
N = 1.4956
the least-squares regression line is
P =
```

$$\frac{21\,x}{38}+\frac{31}{19}$$

```
c is the least-squares solution
```

Exercise 6

```
type lstsqpoly
```

```
function c=lstsqpoly(x,y,n)
hold off
format
format compact
x=x';
y=y';
a=x(1);
m=length(x);
b=x(m);
disp('the design matrix is')
% hold original value of n
n_ = n;
% matrix whose form depends on the on the degree n, this matrix has n+1 columns all conntaining vecvtor x
X = repmat(x,[1 (n+1)]);
% replace last column by a vector of ones
X(:,n+1) = ones(m,1);

for c = 1:n
    X(:,c)= x.^n;
    n = n -1 ;
end
disp(X)
disp('the parameter vector is')
c=lscov(X,y)
disp('the norm of the residual vector is')
N=norm(y-X*c)
plot(x,y,'*'),hold on
polyplot(a,b,c');
fprintf('the polynomial of degree %i of the best least-squares fit is\n',n_)
P=poly2sym(c)
c1 = (inv(X'*X))*(X'*y);
h = closetozeroroundoff(c - c1, 7);
g = closetozeroroundoff(c, 7);
if (h == zeros(size(c)))
 disp('c is the least-squares solution')
end
hold off
end
```

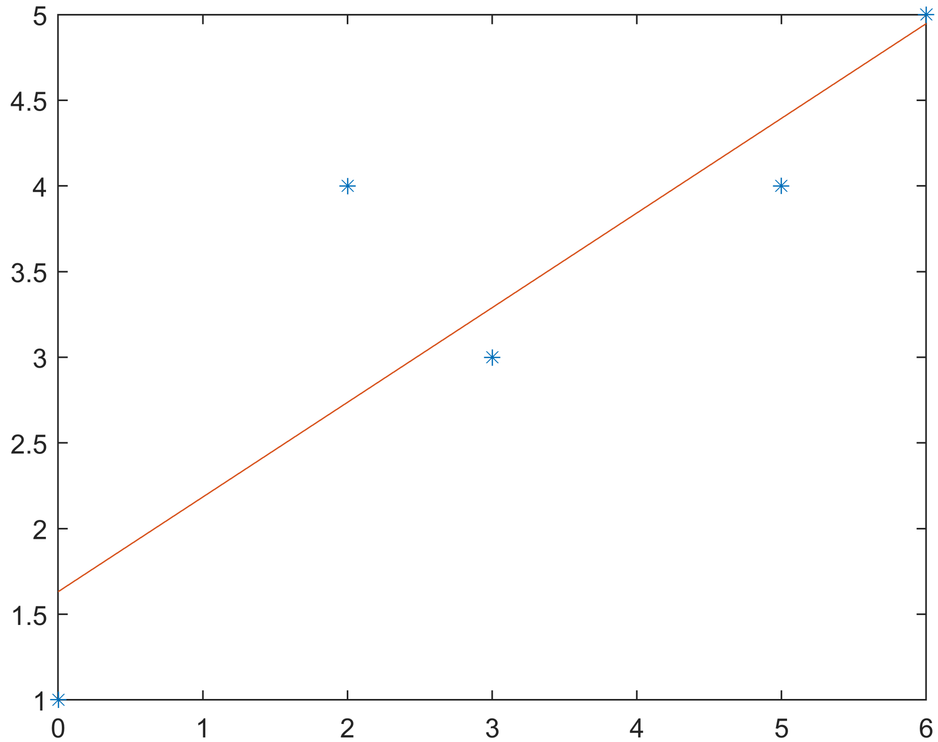
```
type polyplot
```

```
function []=polyplot(a,b,p)
x=(a:(b-a)/50:b)';
y=polyval(p,x);
plot(x,y);
end
```

```
x = [0,2,3,5,6],y = [1,4,3,4,5]
```

```
x = 1×5
     0     2     3     5     6
y = 1×5
     1     4     3     4     5
```

```
% n=1
c=lstsqpoly(x,y,1);
```



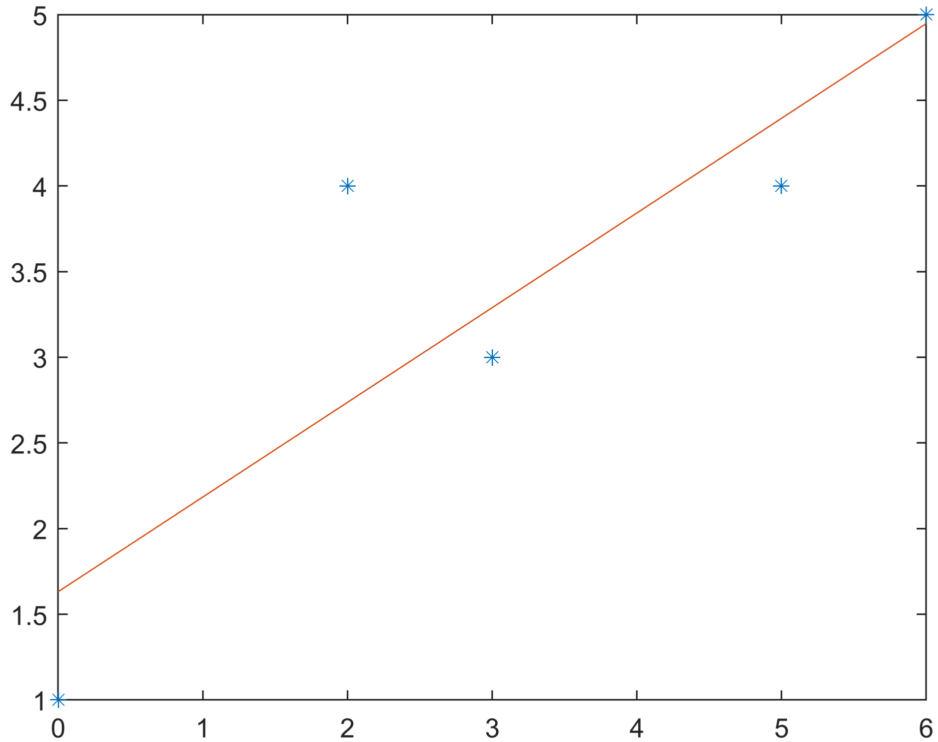

```
the design matrix is
     0     1
     2     1
     3     1
     5     1
     6     1
the parameter vector is
c = 2×1
    0.5526
    1.6316
the norm of the residual vector is
N = 1.4956
the polynomial of degree 1 of the best least-squares fit is
P =
```

$$\frac{21\,x}{38}+\frac{31}{19}$$

```
c is the least-squares solution
```

```
% When n=1 we can see that the code is consistent with Exercise 5, we have
% the same design matrix, the parameter vector, the norm, the polynomial,
% and the plot containing the data points and polynomial p all are the same.

% n=2
c=lstsqpoly(x,y,2);
```

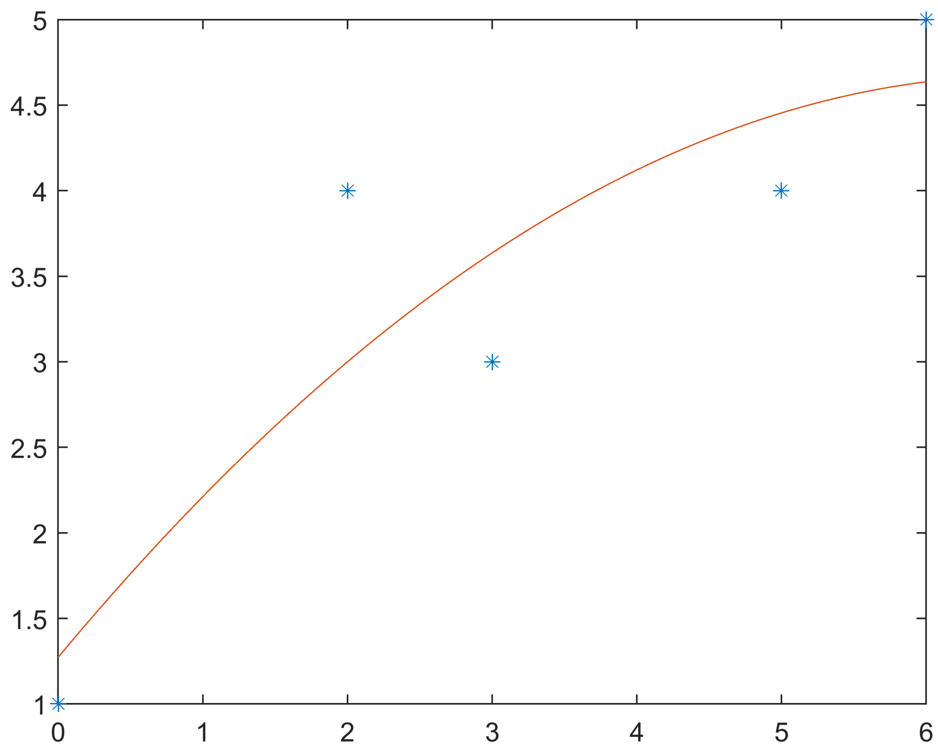



```
the design matrix is
     0     0     1
     4     2     1
     9     3     1
    25     5     1
    36     6     1
the parameter vector is
c = 3×1
   -0.0758
    1.0152
    1.2727
the norm of the residual vector is
N = 1.3484
the polynomial of degree 2 of the best least-squares fit is
P =
```

$$-\frac{5\,x^2}{66}+\frac{67\,x}{66}+\frac{14}{11}$$

```
c is the least-squares solution
```

```
% n = 3
c=lstsqpoly(x,y,3);
```

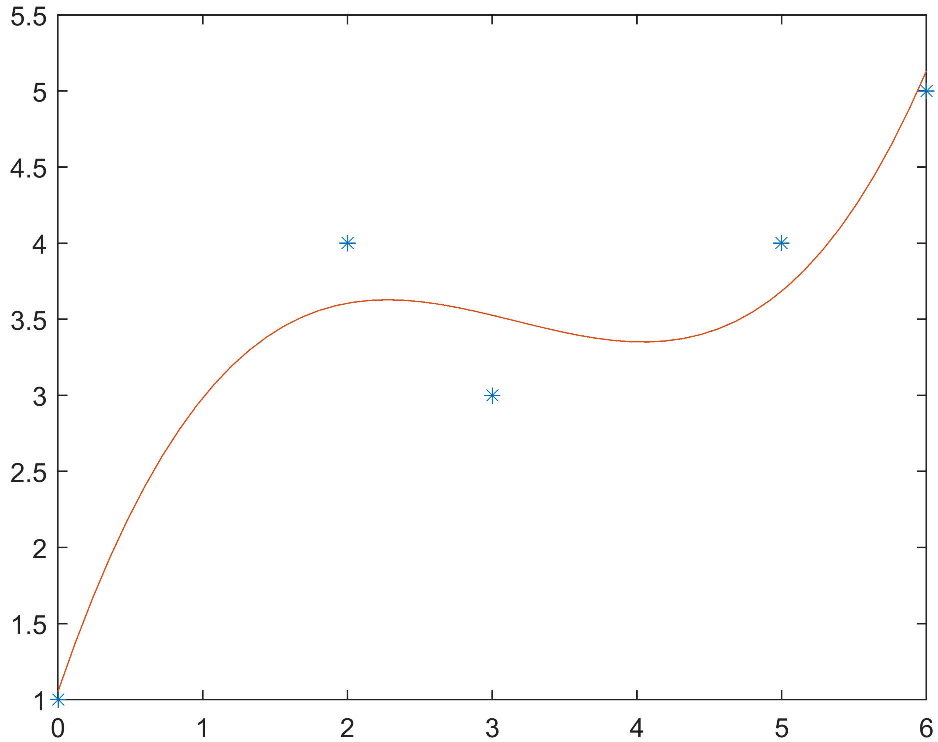



```
the design matrix is
     0     0     0     1
     8     4     2     1
    27     9     3     1
   125    25     5     1
   216    36     6     1
the parameter vector is
c = 4×1
    0.1009
   -0.9561
    2.7851
    1.0526
the norm of the residual vector is
N = 0.7434
the polynomial of degree 3 of the best least-squares fit is
P =
```

$$\frac{23\,x^3}{228} - \frac{109\,x^2}{114} + \frac{635\,x}{228} + \frac{20}{19}$$

```
c is the least-squares solution
```

```
%n=4
c=lstsqpoly(x,y,4);
```

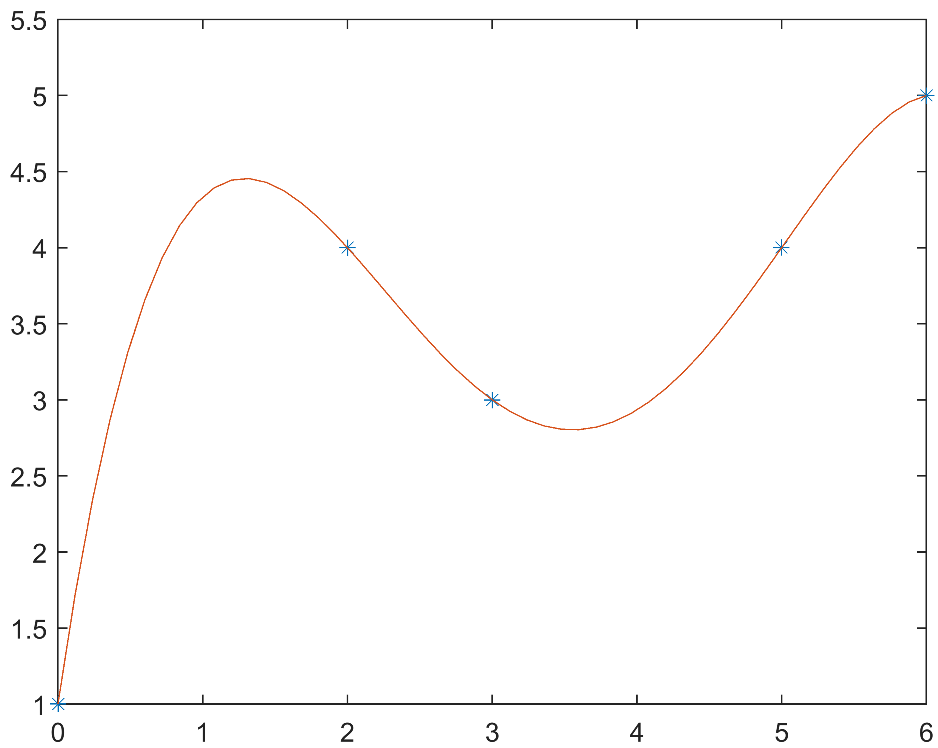



```
the design matrix is
           0            0            0            0            1
          16            8            4            2            1
          81           27            9            3            1
         625          125           25            5            1
        1296          216           36            6            1
the parameter vector is
c = 5×1
   -0.0583
    0.8500
   -3.9750
    6.5167
    1.0000
the norm of the residual vector is
N = 6.5465e-14
the polynomial of degree 4 of the best least-squares fit is
P =
```

$$-\frac{7\,x^4}{120}+\frac{17\,x^3}{20}-\frac{159\,x^2}{40}+\frac{391\,x}{60}+1$$

```
c is the least-squares solution
```

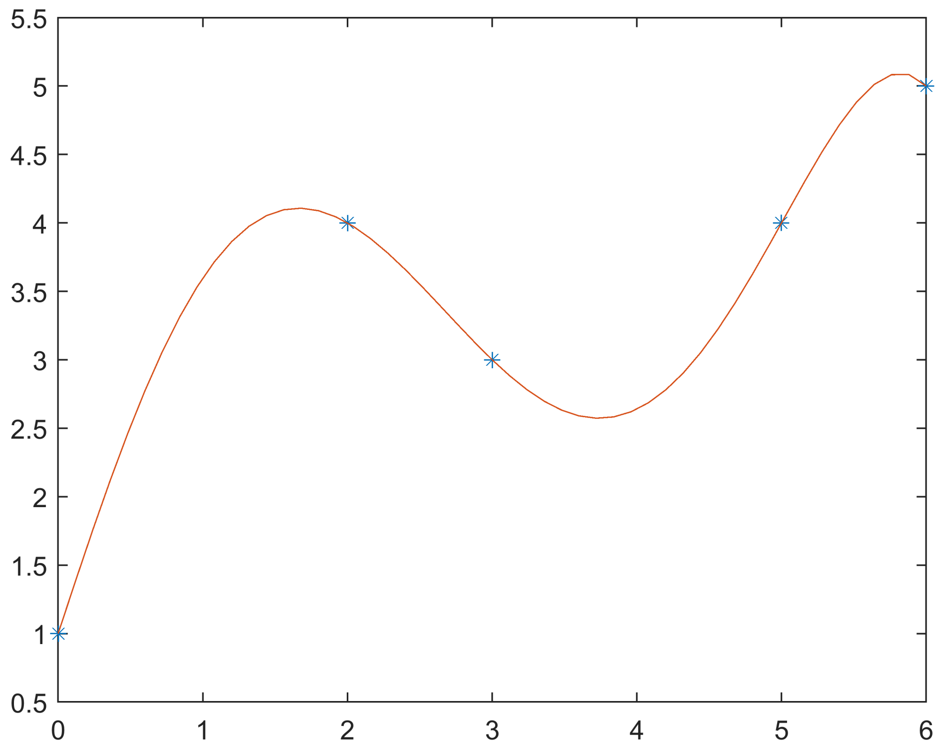
```
% n = 5
c=lstsqpoly(x,y,5);
```




```
the design matrix is
  Columns 1 through 5
         0           0           0           0           0
        32          16           8           4           2
       243          81          27           9           3
      3125         625         125          25           5
      7776        1296         216          36           6
  Column 6
         1
         1
         1
         1
         1
the parameter vector is
Warning: A is rank deficient to within machine precision.
c = 6×1
   -0.0184
    0.2361
   -0.8247
         0
    3.2042
    1.0000
the norm of the residual vector is
N = 7.5137e-14
the polynomial of degree 5 of the best least-squares fit is
P =
```

$$-\frac{53\,x^5}{2880}+\frac{17\,x^4}{72}-\frac{475\,x^3}{576}+\frac{769\,x}{240}+1$$

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  3.659190e-21.
```




```
%{
BONUS:

When n=4, the polynomial interpolates the data points since we can set the Vandermonde
matrix to construct the system: Va = y. And use the Vandermonde determinanat formula
to prove V is nonsingular. The Vandermonde matrix in this case can be considered as
the desgin matrix. Now, recalling that the Vandermonde determinant or the Vadermonde
polynomial can prove if V is nonsingular it can be concluded that since the n+1 points
are different:
The determinant cannot be zero, becasue xi-xj is never zezro. This means the system has
a unique solution and the polynomial interpolates the data points.

In general. if you have any k points on the plane such that no two of them have the
same x-coordinate you can always find a polynomial of degree k-1 whose graph will always
go exactly through these k points.
%}
```

Part III. Application to Dynamical Systems

Exercise 7

```
type trajectory
```

```
function [P]=trajectory(A,X0,N)
    format
    format compact
    L=eig(A);
```

```
if (abs(imag(L(1))) < 1e-7 && abs(imag(L(2))) < 1e-7)
    disp('the eigenvalues of A are real')
    if (L(1) == 0 || L(2) == 0)
        disp('A has a zero eigenvalue')
        return
    else
        % EIGEN ------------------------------------------------

        [~,n]=size(A);

        %part 1. vector L of eigenvalues
        L = eig(A); %column vector of eigenvalues of A
        L = transpose(L); %converts L into a row vector
        L = sort(L); % sorts the entries of L in ascending order

        for i=1:(length(L)-1) %if two eigenvalues are equal within the given range, set them equal to each other
            difL = L(i)-L(i+1);
            if (closetozeroroundoff(difL, 7) == 0)
              L(i+1) = L(i);
            end
        end

        if rank(A) ~= n %checks if matrix A is singular
            for i=1:length(L)
                if (closetozeroroundoff(L(i), 7) == 0) %if an eigenvalue is 0 within the given range, set it to
                    L(i) = 0;
                end
            end
        end

        fprintf('all sorted eigenvalues of A are\n');

        %part 2. orthonormal basis W for each eigenspace
        M = unique(L); %creates a row vector of only the unique eigenvalues of A
        m = zeros(length(M)); %creates a vector of the multiplicity of each unique eigenvalue
        d = zeros(length(M)); %creates a vector for the dimension of the orthonormal basis W

        for i=1:length(M)
            count = 0; %starts the count for the multiplicity
            for j=1:length(L)
                if M(i) == L(j)
                    count = count + 1;
                end
            end
            m(i) = count; %assigns the value of the multiplicity to the corresponding entry
            %fprintf('Eigenvalue %d has multiplicity %i\n',M(i),m(i));

            nullMat = A - (M(i)*eye(size(A,1)));
            W = null(nullMat,'r'); %finds an orthonormal basis for the given eigenvalue
            %fprintf('A basis for eigenvalue lambda = %d is:\n',M(i));
            %W

            d(i) = rank(W); %determines the dimension of the eigenspace for the given eigenvalue
            %fprintf('Dimension of eigenspace for lambda = %d is %i\n',M(i),d(i))
        end

        % part 3. construct diagonalization if possible
        for i=1:length(M)
            if m(i) ~= d(i) %checks if the matrix is not diagonalizable
                disp('A is not diagonalizable: there is no eigenvector basis for R^2');

                P=[];
                D=[];
                return; %terminates the program if it isn't diagonalizable
            end
```

```
     end

     fprintf('A is diagonalizable: there exists an eigenvector basis for R^2\n');
     fprintf('it is formed by V1,V2 corresponding to sorted eigenvalues in L\n');
     P = zeros(n, length(L)); %initializes the invertible matrix P
     for i=1:length(M)
         nullMat = A - (M(i)*eye(n));
         W = null(nullMat,'r');
         if i == 1
             P = W;
         else
             P = horzcat(P, W); %sets the columns of matrix P to be the bases for each eigenvalue
         end
     end

     V1 = P(:,1)
     V2 = P(:,2)


     % EIGEN -------------------------------------------------

     if min(L) < 0
      disp('A has a negative eigenvalue')
      return;
     else
      if L(2) < 1
       disp('the origin is an attractor')
       fprintf('a direction of greatest attraction is through 0 and\n')
       V1
         end
         if L(1) > 1
           disp('the origin is a repeller')
       fprintf('a direction of greatest repulsion is through 0 and\n')
       V2
         end
         if (L(1) < 1 && L(2) > 1)
       disp('the origin is a saddle')
       fprintf('a direction of greatest attraction is through 0 and\n')
       V1
       fprintf('a direction of greatest repulsion is through 0 and\n')
       V2

      end

      if closetozeroroundoff(L(1) - 1,7) == 0 || closetozeroroundoff(L(2) - 1,7) == 0
       disp('A has an eigenvalue 1')
      end
      X0 = [V1,V2,-V1,-V2,X0];
      n=size(X0,2);
X=zeros(2,N+1);

for i = 1:n
 x0 = X0(:,i);
 C = inv([V1, V2]) * x0;
 for j = 1:(N+1)
  X(:,j) = (C(1) * L(1)^(j-1) * V1) + (C(2) * L(2)^(j-1) * V2);
             end

             x=X(1,:);y=X(2,:);
 plot(x,y,'*'), hold on
 plot(x,y)
 if L(2) < 1 || closetozeroroundoff(L(1) - 1,7) == 0 || closetozeroroundoff(L(2) - 1,7) == 0
  v=[-1 1 -1 1];
 else
  v=[-5 5 -5 5];
```

```
        end
        axis(v)
      end
            end
        end
        %end
        %end
    else
        disp('the eigenvalues of A are complex conjugate (non-real) numbers');
        L=eig(A)
     magn=abs(L)
     n=size(X0,2);
     X=zeros(2,N+1);

     for i = 1:n

      x0 = X0(:,i);
      X(:,1) = x0;
      for j = 2:(N+1)

       X(:,j) = A * X(:,j-1);


            end
            x=X(1,:);y=X(2,:);
      plot(x,y,'*'), hold on
      plot(x,y)


     end
    end

    hold off
    %end % MAY be too many end
end
```

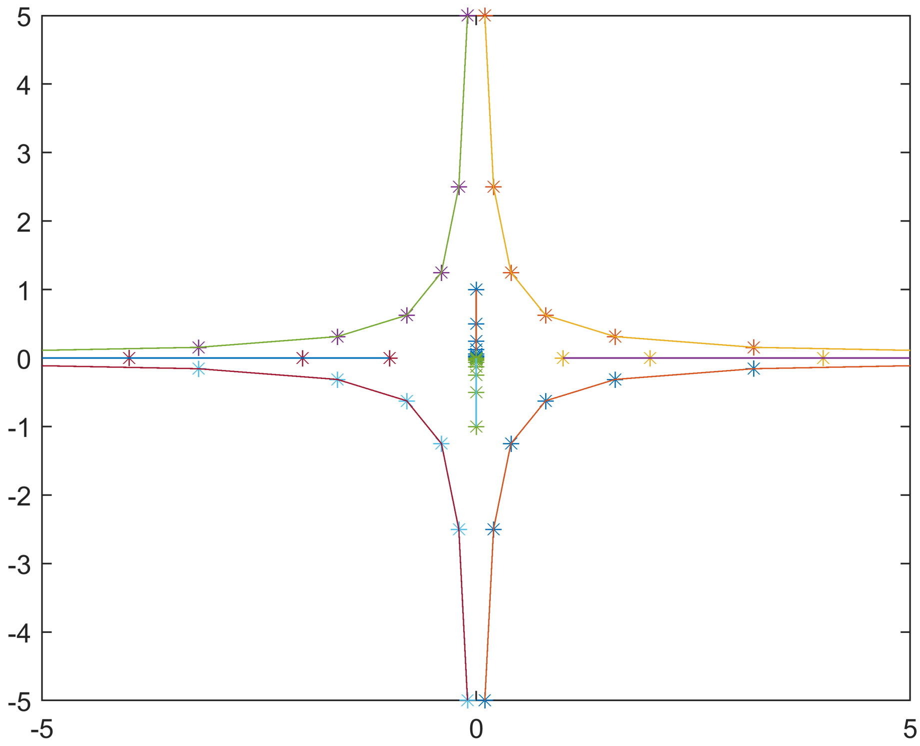
```
%(a)
A=[2 0;0 .5]
```

```
A = 2×2
    2.0000         0
         0    0.5000
```

```
X0=[[.1;5],[-.1;5],[-.1;-5],[.1;-5]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     0
     1
V2 = 2×1
     1
     0
the origin is a saddle
a direction of greatest attraction is through 0 and
V1 = 2×1
     0
     1
a direction of greatest repulsion is through 0 and
```

```
V2 = 2×1
     1
     0
```



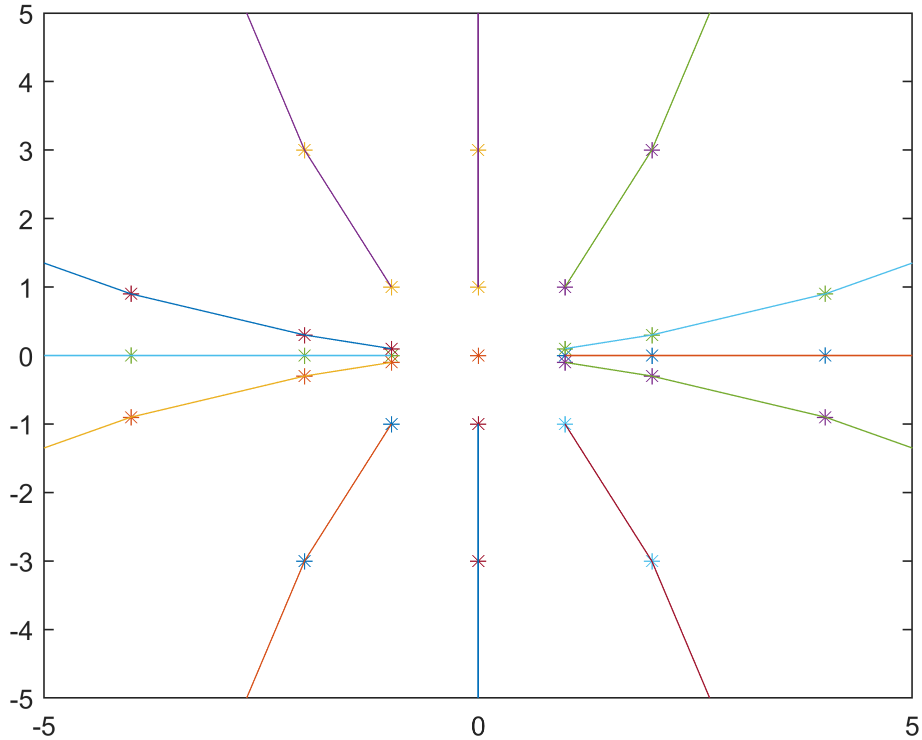

```
ans = 2×2
     0     1
     1     0
```

```
%(b)
A=[2 0; 0 3]
```

```
A = 2×2
     2     0
     0     3
```

```
X0=[[0;0],[1;1],[1;-1],[-1;-1],[-1;1],[1;.1],[-1;.1],[-1;-.1],[1;-.1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     1
     0
V2 = 2×1
     0
     1
the origin is a repeller
a direction of greatest repulsion is through 0 and
V2 = 2×1
     0
     1
```



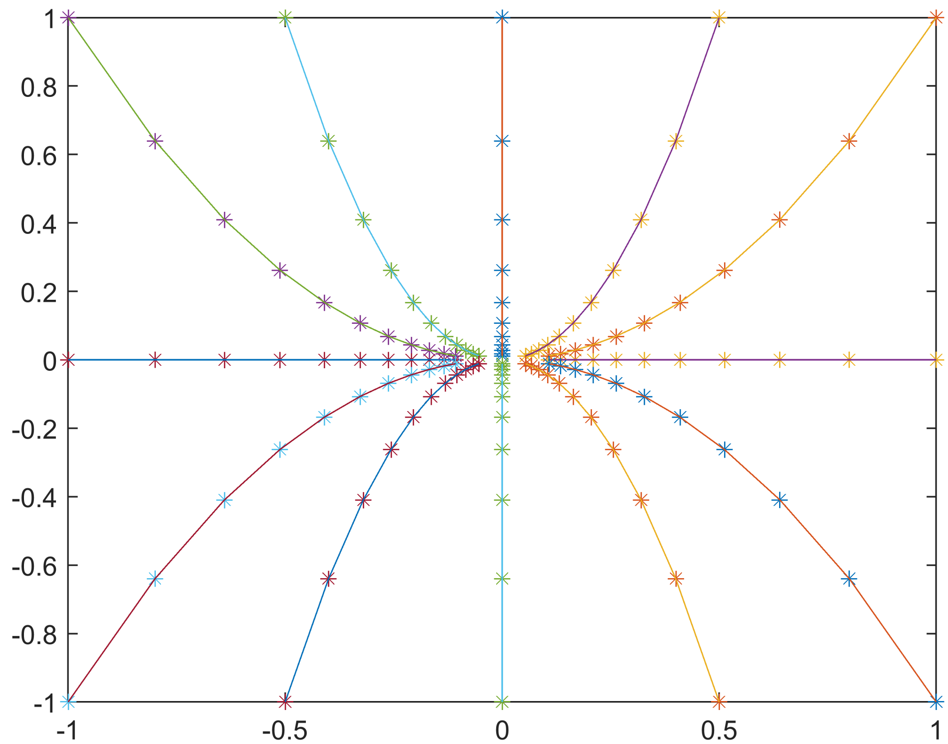

```
ans = 2×2
     1     0
     0     1
```

```
%(c)
A=[.80 0;0 .64]
```

```
A = 2×2
    0.8000         0
         0    0.6400
```

```
X0=[[1;1],[-1;1],[-1;-1],[1;-1],[.5;1],[-.5;1],[-.5;-1],[.5;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     0
     1
V2 = 2×1
     1
     0
the origin is an attractor
a direction of greatest attraction is through 0 and
V1 = 2×1
     0
     1
```



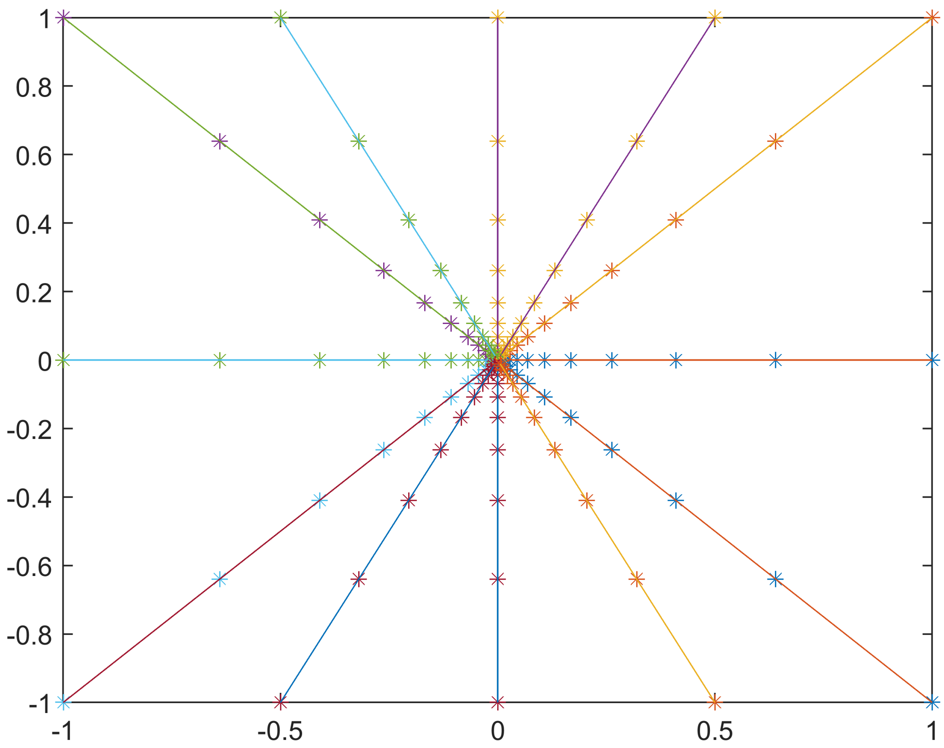

```
ans = 2×2
     0     1
     1     0
```

```
%(d)
A=[.64 0;0 .64]
```

```
A = 2×2
    0.6400         0
         0    0.6400
```

```
X0=[[1;1],[-1;1],[-1;-1],[1;-1],[.5;1],[-.5;1],[-.5;-1],[.5;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     1
     0
V2 = 2×1
     0
     1
the origin is an attractor
a direction of greatest attraction is through 0 and
V1 = 2×1
     1
     0
```




```
ans = 2×2
     1     0
     0     1
```

```
%(e)
A=[5 0; 1 5]
```

```
A = 2×2
     5     0
     1     5
```

```
X0=[[1;1],[-1;1],[-1;-1],[1;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is not diagonalizable: there is no eigenvector basis for R^2
ans =
     []
```

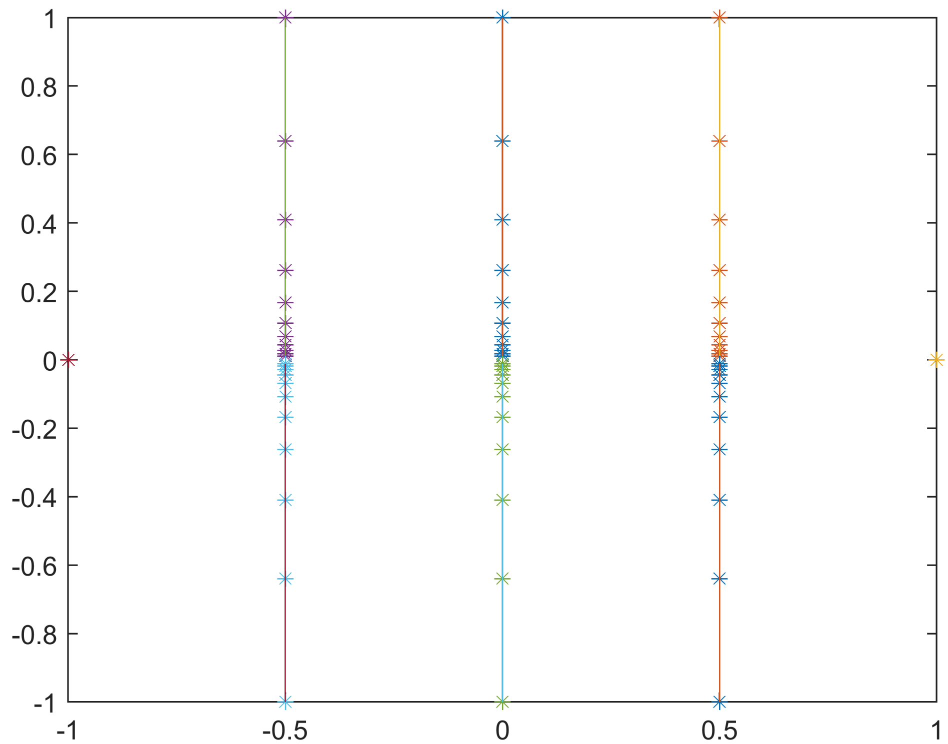
```
%(f)
A=[1 0;0 .64]
```

```
A = 2×2
    1.0000         0
         0    0.6400
```

```
X0=[[.5;1],[-.5;1],[-.5;-1],[.5;-1]];
N=10;
```

```
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     0
     1
V2 = 2×1
     1
     0
A has an eigenvalue 1
```




```
ans = 2×2
     0     1
     1     0
```

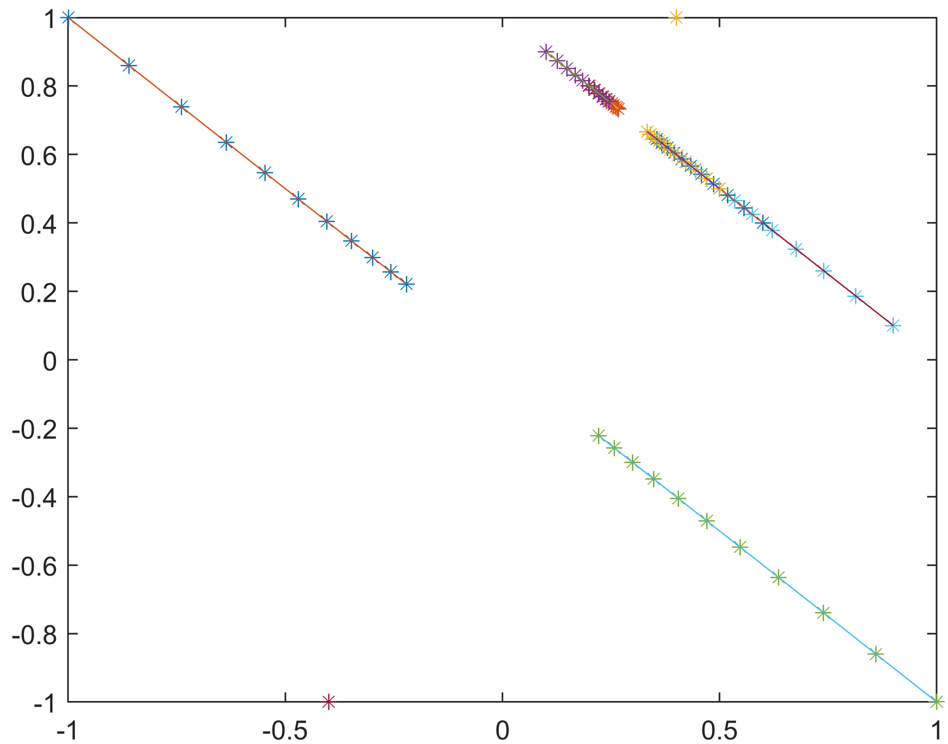
```
%(g)
A=[.90 .04;.10 .96]
```

```
A = 2×2
    0.9000    0.0400
    0.1000    0.9600
```

```
X0=[[.2;.8],[.1;.9],[.9;.1],[.6;.4],[.5;.5]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
```

```
   -1.0000
    1.0000
V2 = 2×1
    0.4000
    1.0000
A has an eigenvalue 1
```



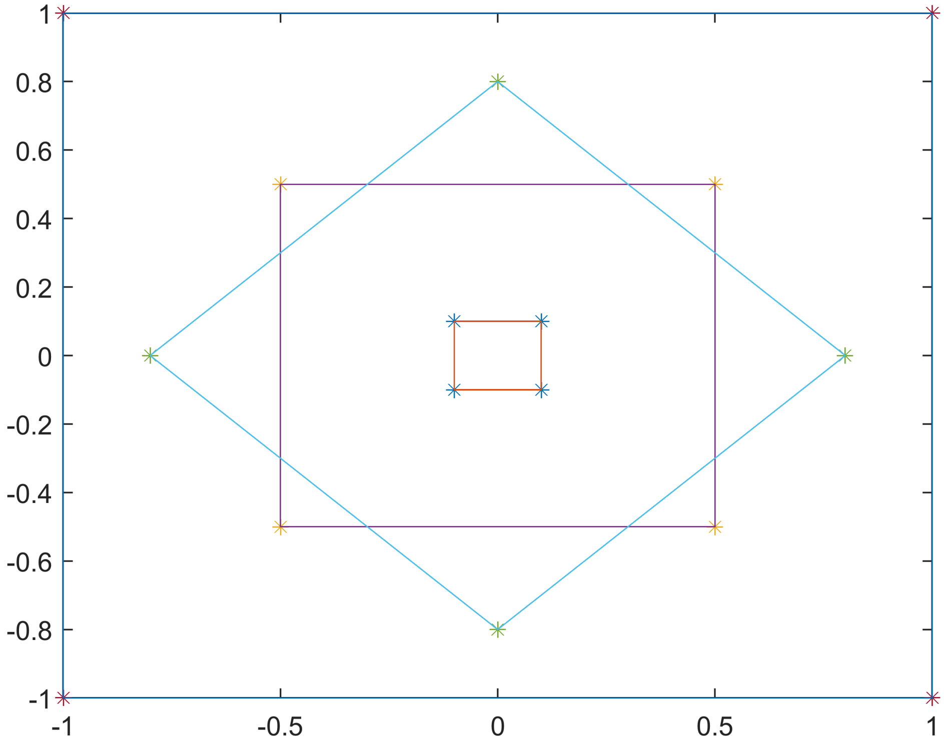

```
ans = 2×2
   -1.0000    0.4000
    1.0000    1.0000
```

```
%(h)
A=[0 -1;1 0]
```

```
A = 2×2
     0    -1
     1     0
```

```
X0=[[.1;.1],[.5;.5],[.8;0],[1;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are complex conjugate (non-real) numbers
L = 2×1
   0.0000
   0.0000
magn = 2×1
     1
     1
```

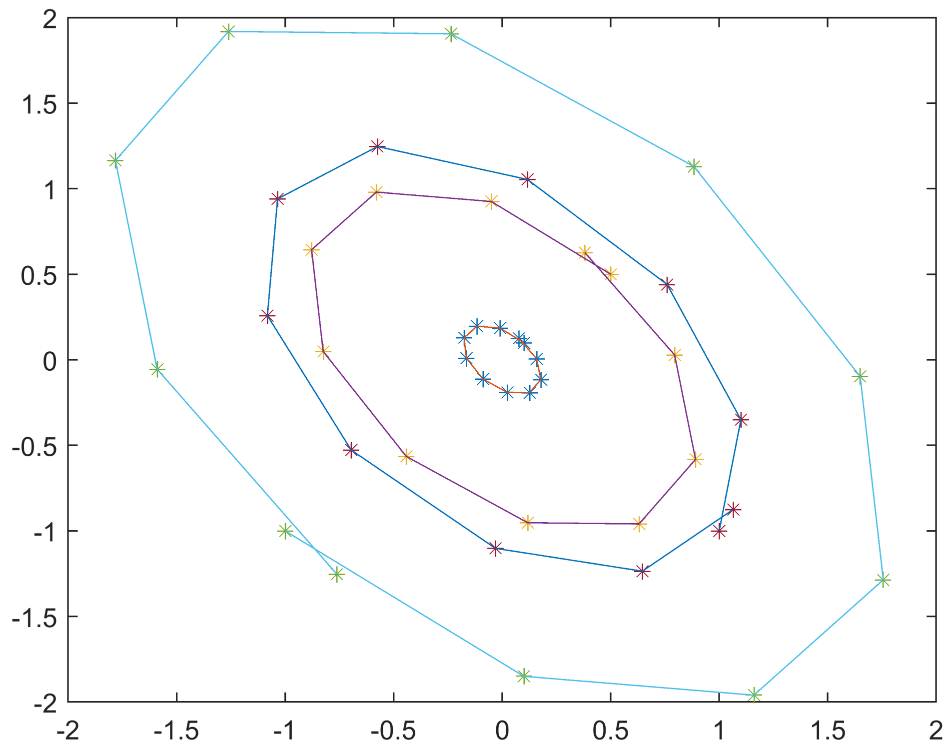



```
%(i)
A=[.5 -.6;.75 1.1]
```

```
A = 2×2
    0.5000   -0.6000
    0.7500    1.1000
```

```
X0=[[.1;.1],[.5;.5],[-1;-1],[1;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are complex conjugate (non-real) numbers
L = 2×1
   0.8000
   0.8000
magn = 2×1
     1
     1
```

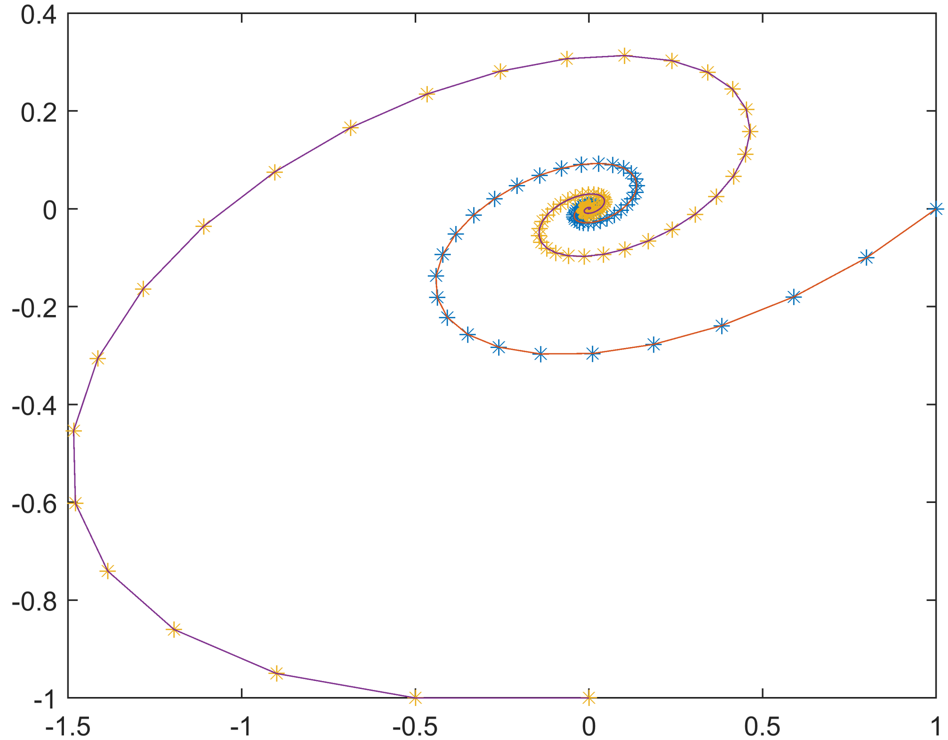



```
%(j)
A=[.8 .5; -.1 1.0]
```

```
A = 2×2
    0.8000    0.5000
   -0.1000    1.0000
```

```
X0=[[1;0],[0;-1]];
N=100;
trajectory(A, X0, N)
```

```
the eigenvalues of A are complex conjugate (non-real) numbers
L = 2×1
   0.9000
   0.9000
magn = 2×1
    0.9220
    0.9220
```

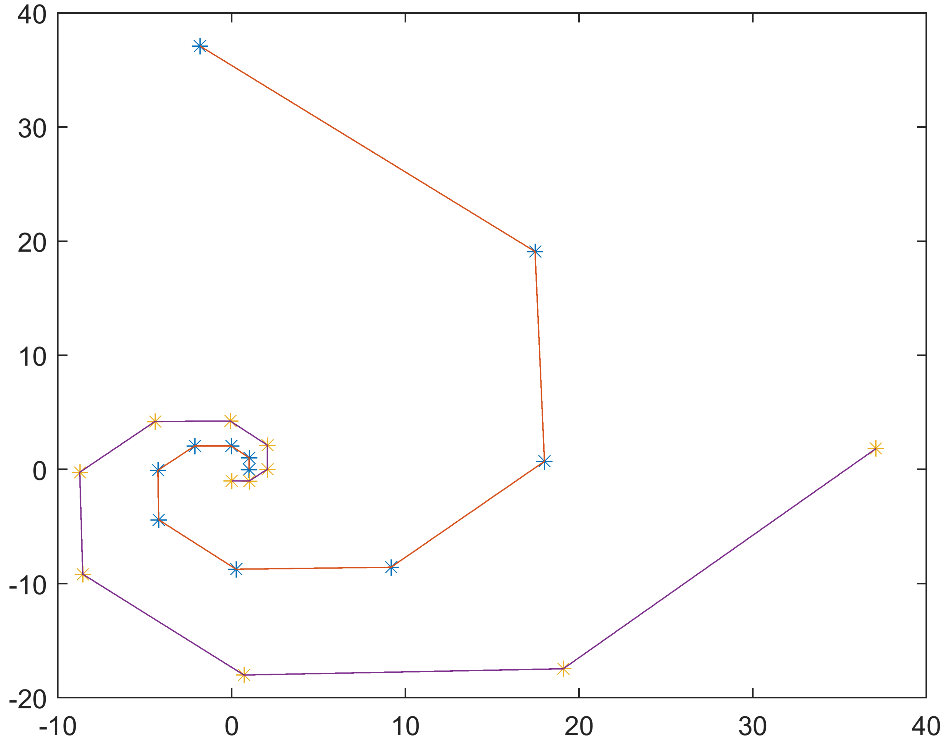



```
%(k)
A=[1.01 -1.02;1.02 1.01]
```

```
A = 2×2
    1.0100   -1.0200
    1.0200    1.0100
```

```
X0=[[1;0],[0;-1]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are complex conjugate (non-real) numbers
L = 2×1
   1.0100
   1.0100
magn = 2×1
    1.4354
    1.4354
```

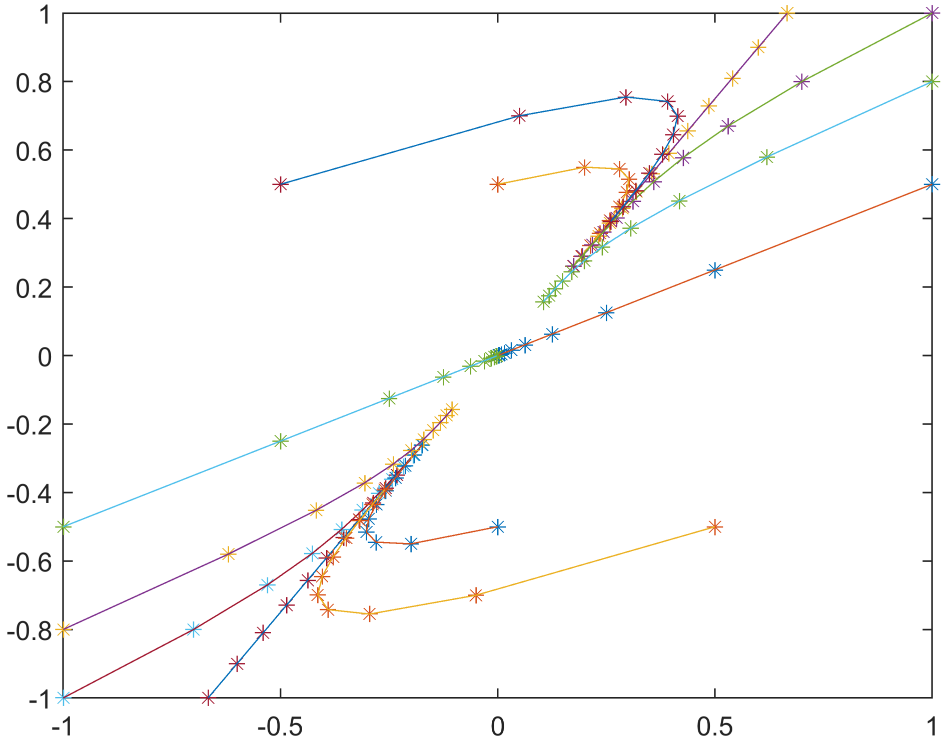



```
%(l)
A=[.3 .4;-.3 1.1]
```

```
A = 2×2
    0.3000    0.4000
   -0.3000    1.1000
```

```
X0=[[0;.5],[1;1],[-1;-1],[0;-.5],[-1;-.8],[1;.8],[-.5;.5],[.5;-.5]];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
    2.0000
    1.0000
V2 = 2×1
    0.6667
    1.0000
the origin is an attractor
a direction of greatest attraction is through 0 and
V1 = 2×1
    2.0000
    1.0000
```




```
ans = 2×2
    2.0000    0.6667
    1.0000    1.0000
```

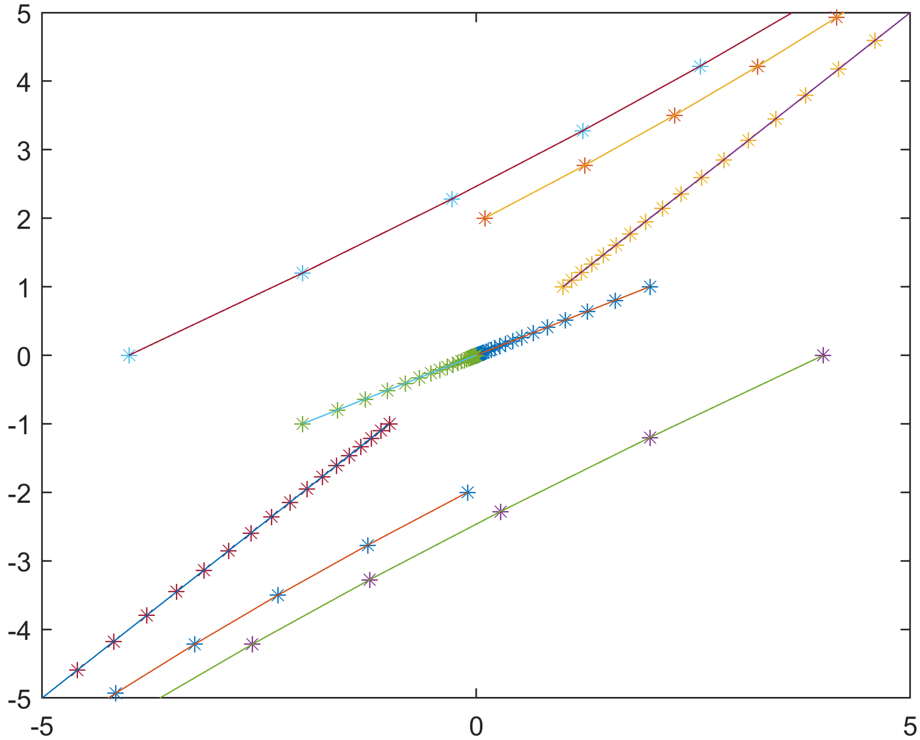
```
%(m)
A=[.5 .6; -.3 1.4]
```

```
A = 2×2
    0.5000    0.6000
   -0.3000    1.4000
```

```
X0=[[.1;2],[4;0],[-4;0],[-.1;-2]];
N=30;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
    2.0000
    1.0000
V2 = 2×1
    1.0000
    1.0000
the origin is a saddle
a direction of greatest attraction is through 0 and
V1 = 2×1
    2.0000
    1.0000
a direction of greatest repulsion is through 0 and
V2 = 2×1
    1.0000
```

```
    1.0000
```



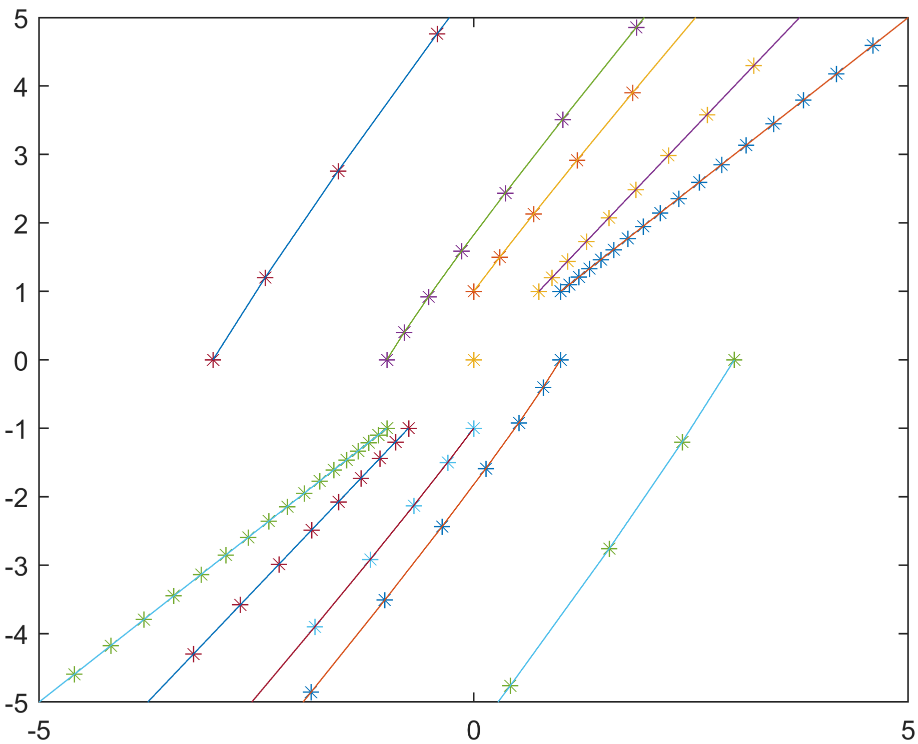

```
ans = 2×2
    2.0000    1.0000
    1.0000    1.0000
```

```
%(n)
A=[.8 .3; -.4 1.5]
```

```
A = 2×2
    0.8000    0.3000
   -0.4000    1.5000
```

```
X0=[[0;1],[-1;0],[0;-1],[1;0],[0;0],[3;0],[-3;0]];
N=50;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
    1.0000
    1.0000
V2 = 2×1
    0.7500
    1.0000
the origin is a repeller
a direction of greatest repulsion is through 0 and
V2 = 2×1
    0.7500
    1.0000
```




```
ans = 2×2
    1.0000    0.7500
    1.0000    1.0000
```

```
%(p)
A=[[1 2;2 4]]
```

```
A = 2×2
     1     2
     2     4
```

```
X0=[];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
A has a zero eigenvalue
```

```
%(q)
A=[-.64 0;0 1.36]
```

```
A = 2×2
   -0.6400         0
         0    1.3600
```

```
X0=[];
N=10;
trajectory(A, X0, N)
```

```
the eigenvalues of A are real
```

```
all sorted eigenvalues of A are
A is diagonalizable: there exists an eigenvector basis for R^2
it is formed by V1,V2 corresponding to sorted eigenvalues in L
V1 = 2×1
     1
     0
V2 = 2×1
     0
     1
A has a negative eigenvalue
ans = 2×2
     1     0
     0     1
```

```
%BONUS 1

% In (d) the eigenvalues of A are both 0.64. So when we use formula (3) we have
%x_k = ((0.64)^k) * (c_1 * v_1 + c_2 * v_2) = ((0.64)^k) * x_0
% so the point on the plane whose coordinates are represented
% in x_i for consecutive i's moves
% along the straight line from x_0 towards the origin.
% At each step the distance from x_i to the origin
% is rescaled by 0.64.


% The pattern obtained in (f) can be explained using formula (3).
% We have
% x_k = c_1 * (0.64^k) * v_1 + c_2 * (1^k)*v_2 =
%  = c_1 * (0.64^k)*v_1 + c_2 * v_2
% since v_2 = [1,0], all the points will not change their first coordinate.
% In the second coordinate the points are being rescaled towards 0


%BONUS 3

% From formula (3), if the magnitudes of the eigenvalues are both greater than 1
% then the consecutive points are being replled from 0
% If the magnitudes of the eigenvalues are both less than 1 then the
% consecutive points are being pulled towards 0.
% If the magnitudes of both eigenvalues are 1, then the consecutive
%points will be "orbiting" the origin.
% That is because we have x_k = c_1 * (lambda_1)^k * v1 + c_2 * (lambda_2)^k * v2
%
```