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 # \_\_\_\_\_13\_\_\_\_\_\_

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

\_1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Koby Miller\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ­­­

\_3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Nicholas Miller\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_4.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Sean O’Reilly\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_5.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Carl Simpron\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_6.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Daniel Silva\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

%Exercise 2

type diagonal

function L = diagonal(A)

%This function makes a matrix P from a matrix A whose columns are

%corresponding eigenvectors so that AP=PD. It then puts matrix P in

%reduced Echelon form. From there, it counts how many columns are

%linearly independent. From this, it determins if the original matrix A

%can be diagonalized

n=size(A,1)

[P, D] = eig(A);

E = rref(P);

k = 0;

r = 0; %Used to check for linear independence

for i=1:1:n %Used to keep track of the row number

for j=1:1:n %Used to keep track of the col number

r = 0;

%Once in reduced Echelon form you can easily check for linear independence.

%The matrix should mirror the eye() matrix

if(j==i)

if(E(i,j)==1)

continue

else

r = 1;

end

else

if(E(i,j)==0)

continue

else

r = 1;

end

end

if(r==1)

break

end

end

if(r==0)

k = k+1;

end

end

disp('The number of linearly independent columns of P is k')

k

if(n==k)

disp('A is diagonalizable')

disp('A basis for R^n is: ')

P

else

disp('A is not diagonalizable')

disp('A does not have enough linearly independent eigenvectors to create

a basis for R^n')

end

L=transpose(diag(D));

end

%(a)

A = [2,2;0,2];

L = diagonal(A)

n =

2

The number of linearly independent columns of P is k

k =

0

A is not diagonalizable

A does not have enough linearly independent eigenvectors to create a basis

for R^n

L =

2 2

%(b)

A = [4,0,0,0;1,3,0,0;0,-1,3,0;0,-1,5,4];

L = diagonal(A)

n =

4

The number of linearly independent columns of P is k

k =

0

A is not diagonalizable

A does not have enough linearly independent eigenvectors to create a basis

for R^n

L =

4 3 3 4

%(c)

A = jord(5,3);

L = diagonal(A)

n =

5

The number of linearly independent columns of P is k

k =

0

A is not diagonalizable

A does not have enough linearly independent eigenvectors to create a basis

for R^n

L =

3 3 3 3 3

%(d)

A = diag([3, 3, 3, 2, 2, 1]);

L = diagonal(A)

n =

6

The number of linearly independent columns of P is k

k =

6

A is diagonalizable

A basis for R^n is:

P =

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

L =

1 2 2 3 3 3

%(e)

A = magic(4);

L = diagonal(A)

n =

4

The number of linearly independent columns of P is k

k =

4

A is diagonalizable

A basis for R^n is:

P =

-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

L =

34.0000 8.9443 -8.9443 0.0000

%(f)

A= ones(5);

L = diagonal(A)

n =

5

The number of linearly independent columns of P is k

k =

5

A is diagonalizable

A basis for R^n is:

P =

0.8333 -0.1667 -0.1667 0.2236 0.4472

-0.1667 0.8333 -0.1667 0.2236 0.4472

-0.1667 -0.1667 0.8333 0.2236 0.4472

-0.5000 -0.5000 -0.5000 0.2236 0.4472

0 0 0 -0.8944 0.4472

L =

0 0 0 0 5

%(g)

A = magic(5);

L = diagonal(A)

n =

5

The number of linearly independent columns of P is k

k =

5

A is diagonalizable

A basis for R^n is:

P =

-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

L =

65.0000 -21.2768 -13.1263 21.2768 13.1263

%exercise3

function B = shrink(A)

format compact,

[~, pivot] = rref(A);

B = A(: , pivot);

function [p,z] = proj(A,b)

format compact,

A=shrink(A);

b=transpose(b);

sz = size(A);

szb = size(b);

if szb(1)~=sz(1)

disp('No solution: dimensions of A and b disagree')

p = [ ];

z = [ ];

return

end

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

disp('no computations needed!')

p = transpose(b);

z = transpose(b)-p;

return

end

ortest = 0;

count = sz(2);

if count>0

ortest = ortest + (A(count,1)\*b(count));

count = count - 1;

end

if ortest == 0

p = transpose(b);

z = transpose(b)-p;

disp('b is orthogonal to Col A')

return

end

x = b\A;

p = A\*x;

p = transpose(p);

z = transpose(b)-p;

ortest = 0;

sz = length(p)

count = sz(1);

if count>0

ortest = ortest + (p(count)\*b(count));

count = count - 1;

end

if ortest == 0

disp('Yes, p and z are orthogonal! Great Job!')

return

else

disp('Oops! Is there a bug in my code?')

return

end

format compact

%(a)

A= magic(6); A=A( : , 1 : 4), b = (1 : 6)

A =

35 1 6 26

3 32 7 21

31 9 2 22

8 28 33 17

30 5 34 12

4 36 29 13

b =

1 2 3 4 5 6

[p,z] = proj(A,b)

{\_Error using <a href="matlab:matlab.internal.language.introspective.errorDocCallback('mtimes')" style="font-weight:bold"> \* </a>

Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number

of rows in the second matrix. To perform elementwise multiplication, use '.\*'.

Error in <a href="matlab:matlab.internal.language.introspective.errorDocCallback('proj', 'M:\proj.m', 34)" style="font-weight:bold">proj</a> (<a href="matlab: opentoline('M:\proj.m',34,0)">line 34</a>)

p = A\*x;}\_

%(b)

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

A =

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 =

0 0 0 0 0 1

[p,z] = proj(A,b)

b is orthogonal to Col A

p =

0 0 0 0 0 1

z =

0 0 0 0 0 0

%(c)

A = magic(4), b = (1 : 5)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

b =

1 2 3 4 5

[p,z] = proj(A,b)

No solution: dimensions of A and b disagree

p =

[]

z =

[]

%(d)

A = magic(5), b = rand(1,5)

A =

17 24 1 8 15

23 5 7 14 16

4 6 13 20 22

10 12 19 21 3

11 18 25 2 9

b =

0.1576 0.9706 0.9572 0.4854 0.8003

[p,z] = proj(A,b)

no computations needed!

p =

0.1576 0.9706 0.9572 0.4854 0.8003

z =

0 0 0 0 0

%(e)

A= ones(6); A( : ) = 1 : 36, b = [1,0,1,0,1,0]

A =

1 7 13 19 25 31

2 8 14 20 26 32

3 9 15 21 27 33

4 10 16 22 28 34

5 11 17 23 29 35

6 12 18 24 30 36

b =

1 0 1 0 1 0

[p,z] = proj(A,b)

b is orthogonal to Col A

p =

1 0 1 0 1 0

z =

0 0 0 0 0 0

%(f)

A= ones(6); A( : ) = 1 : 36; A= null(A,'r'), b = ones(1,6)

A =

1 2 3 4

-2 -3 -4 -5

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

b =

1 1 1 1 1 1

[p,z] = proj(A,b)

b is orthogonal to Col A

p =

1 1 1 1 1 1

z =

0 0 0 0 0 0

%Exercise 4

type shrink

function B = shrink(A)

format compact,

[~, pivot] = rref(A);

B = A(: , pivot);

type solvemore

function X = solvemore(A,b)

format long,

A=shrink(A);

[m, n] = size(A);

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

disp('The equation is consistent – look for the exact solution')

B=closetozeroroundoff( A'\*A-eye(n));

if B == zeros(n, n)

disp('A is orthogonal')

x1 = A\b;

x2 = A'\*b;

X = [x1, x2];

N = norm(x1-x2);

disp ('The norm of the difference between two solutions is N = ')

disp (N)

else

disp('A does not have orthonormal columns')

x1 = A\b;

X = x1;

end

else

disp('The system is inconsistent – look for the least-squares solution')

x3 = (A'\*A)\(A'\*b);

disp('The solution of the normal equations is x3:')

disp(x3)

B=closetozeroroundoff( A'\*A-eye(n));

if B == zeros(n, n)

disp('A has orthonormal columns: an orthonormal basis for Col A is U=A')

U=A;

else

disp('An orthonormal basis for Col A is U=')

U = orth(A)

end

disp ('The projection of b onto Col A is')

b1 = U\*U'\*b

disp('The least-squares solution by using the projection onto Col A is x4=')

x4 = A\(b1)

disp('The least-squares error of this approximation is n2=')

n2 = norm(b-A\*x4)

disp('the norm of the difference between two solutions x3 and x4 is n3=')

n3 = norm(x3-x4)

x = rand(n,1);

disp('An error of approximation of b by Ax for a random vector x in R^n is n4=')

n4 = norm(b-A\*x)

X = [x3, x4];

end

%(a)

A=magic(4); b=A( : ,1), A=orth(A)

b =

16

5

9

4

A =

-0.500000000000000 0.670820393249937 0.500000000000000

-0.500000000000000 -0.223606797749979 -0.500000000000000

-0.500000000000000 0.223606797749979 -0.500000000000000

-0.500000000000000 -0.670820393249937 0.500000000000000

X = solvemore(A,b)

The equation is consistent – look for the exact solution

A is orthogonal

The norm of the difference between two solutions is N =

3.580361673049448e-15

X =

-16.999999999999996 -17.000000000000000

8.944271909999159 8.944271909999159

3.000000000000000 2.999999999999999

%(b)

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

A =

-0.447213595499958 -0.545634873129948 0.511667273601714 0.195439507584854 -0.449758363151198

-0.447213595499958 -0.449758363151205 -0.195439507584838 -0.511667273601691 0.545634873129969

-0.447213595499958 -0.000000000000024 -0.632455532033676 0.632455532033676 -0.000000000000002

-0.447213595499958 0.449758363151189 -0.195439507584872 -0.511667273601694 -0.545634873129966

-0.447213595499958 0.545634873129987 0.511667273601672 0.195439507584856 0.449758363151196

b =

0.097540404999410

0.278498218867048

0.546881519204984

0.957506835434298

0.964888535199277

X = solvemore(A,b)

The equation is consistent – look for the exact solution

A is orthogonal

The norm of the difference between two solutions is N =

8.980775854696058e-16

X =

-1.272463781215829 -1.272463781215830

0.778645190933962 0.778645190933962

-0.043832342146252 -0.043832342146252

-0.078904505187779 -0.078904505187779

0.019606294966118 0.019606294966118

%(c)

A= magic(4), b = ones(4,1)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

b =

1

1

1

1

X = solvemore(A,b)

The equation is consistent – look for the exact solution

A does not have orthonormal columns

X =

0.058823529411765

0.117647058823529

-0.058823529411765

%(d)

A = magic(4), b = rand(4, 1)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

b =

0.157613081677548

0.970592781760616

0.957166948242946

0.485375648722841

X = solvemore(A,b)

The system is inconsistent – look for the least-squares solution

The solution of the normal equations is x3:

0.025346205234536

0.384344079553035

-0.334080035952160

An orthonormal basis for Col A is U=

U =

-0.363225569906992 -0.839773278980323 0.335928601456289

-0.511952614823082 0.201051551215513 -0.497476425501395

-0.413098103234259 -0.228706948321817 -0.571876812690966

-0.659789104673461 0.449502219631667 0.559129763025005

The projection of b onto Col A is

b1 =

0.171987335002163

1.013715541734457

0.914044188269102

0.471001395398227

The least-squares solution by using the projection onto Col A is x4=

x4 =

0.025346205234536

0.384344079553033

-0.334080035952159

The least-squares error of this approximation is n2=

n2 =

0.064283615119279

the norm of the difference between two solutions x3 and x4 is n3=

n3 =

2.362089582792836e-15

An error of approximation of b by Ax for a random vector x in R^n is n4=

n4 =

22.275351979395186

X =

0.025346205234536 0.025346205234536

0.384344079553035 0.384344079553033

-0.334080035952160 -0.334080035952159

%(e)

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

A =

-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 =

0.915735525189067

0.792207329559554

0.959492426392903

0.655740699156587

X = solvemore(A,b)

The system is inconsistent – look for the least-squares solution

The solution of the normal equations is x3:

-1.661587990149056

0.211815916256258

-0.090111765803402

A has orthonormal columns: an orthonormal basis for Col A is U=A

The projection of b onto Col A is

b1 =

0.927828548412445

0.828486399229689

0.923213356722768

0.643647675933208

The least-squares solution by using the projection onto Col A is x4=

x4 =

-1.661587990149055

0.211815916256258

-0.090111765803402

The least-squares error of this approximation is n2=

n2 =

0.054081643961915

the norm of the difference between two solutions x3 and x4 is n3=

n3 =

2.914335439641036e-16

An error of approximation of b by Ax for a random vector x in R^n is n4=

n4 =

2.082957134710587

X =

-1.661587990149056 -1.661587990149055

0.211815916256258 0.211815916256258

-0.090111765803402 -0.090111765803402

%Comparing n2 to n4, we can see that the error is much higher using a random vector than using a projection

%(Exercise 5)

type polyplot

function [] = polyplot(a,b,p)

x = (a : (b - a) / 50 : b)';

y = polyval(p,x);

plot(x,y)

end

type lstsqline

function c = lstsqline(x,y)

format rat,

x = x';

y = y';

a = x(1);

m = length(x);

b = x(m);

X = [x,ones(m,1)];

c = lscov(X,y);

%the same result will be obtained if c = (X^T \* X)^(-1) \* (X^T \* y)

%or c = (X^T \* X) \ (X^T \* y)

%and we are verifying it:

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

c2 = (X' \* X) \ (X' \* y)

%the next command calculates the 2-norm of the residual vector

N = norm(y - X\*c)

%plot data points and the least-squares regression line:

plot(x,y,'\*'), hold

polyplot(a,b,c');

%output the polynomial:

P = poly2sym(c)

End

x = [0,2,3,5,6]

x =

0 2 3 5 6

y = [4,3,2,1,0]

y =

4 3 2 1 0

c = lstsqline(x,y)

c1 =

-25/38

78/19

c2 =

-25/38

78/19

N =

514/1417

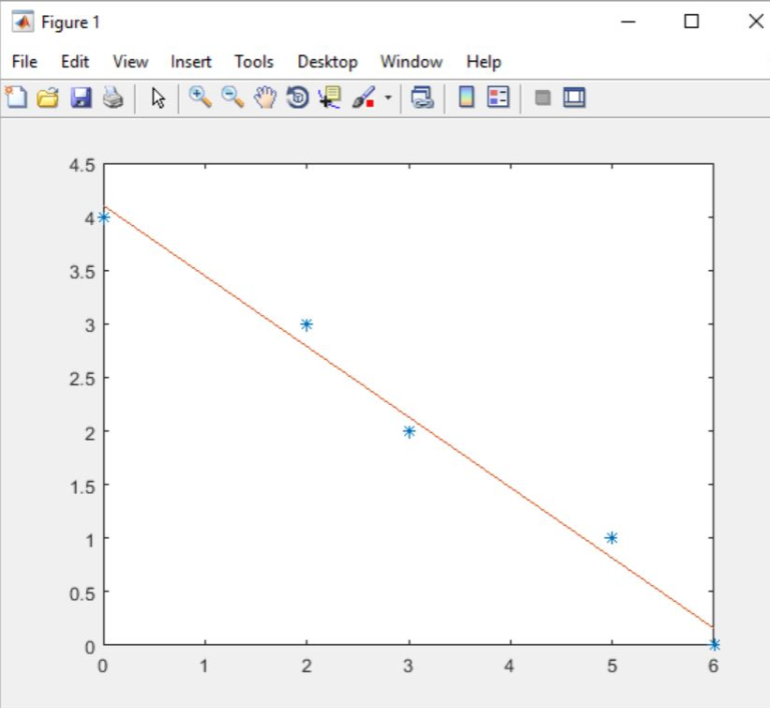
P =

78/19 - (25\*x)/38

c =

-25/38

78/19



>> % Exercise 6

>> type lstsqpoly

function c = lstsqpoly( x, y, n )

%LSTSQLINE generates the least-squares polynomial

% of degree n

x = x';

y = y';

a = x(1);

m = length(x);

b = x(m);

%construct matrix X whose form is defined by the

%degree n of the polynomial

X (:, n+1) = ones (m, 1, class(x));

for j = n:-1:1

X (:, j) = x.\*X (:, j+1);

end

c = lscov(X, y);

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

c2 = (X'\*X)\(X'\*y)

N = norm(y - X\*c)

plot(x, y, ' \* '), hold

polyplot(a, b, c');

P = poly2sym(c)

end

>> x = [0, 2, 3, 5, 6]

x =

    0 2     3 5 6

>> y = [4, 3, 2, 1, 0]

y =

    4 3     2 1 0

% n = 1

>> n = 1;

>> c = lstsqpoly (x, y, n)

c1 =

    -25/38

     78/19

c2 =

    -25/38

     78/19

N =

    514/1417

Current plot held

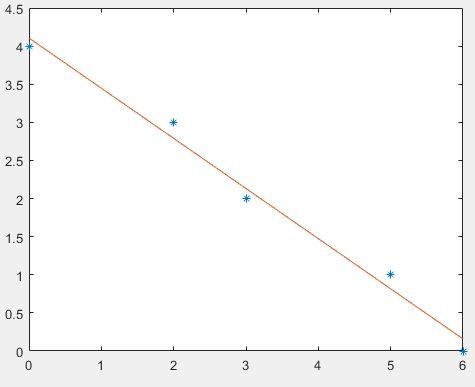
P =

78/19 - (25\*x)/38

c =

    -25/38

     78/19



% When n = 1, all vectors of coefficients c, c1, and c2 match:

>> % c = c1 = c2 = [-25/38; 78/19];

>> n = 2;

>> c = lstsqpoly (x, y, n)

c1 =

     -3/154

    -83/154

    309/77

c2 =

     -3/154

    -83/154

    309/77

N =

    703/2181

Current plot held

P =

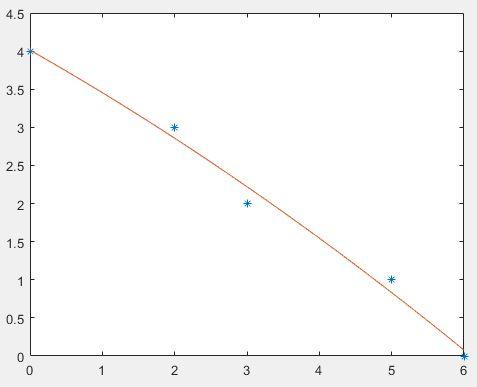
309/77 - (3\*x^2)/154 - (83\*x)/154

c =

     -3/154

    -83/154

    309/77



% When n = 2, all vectors of coefficients c, c1, and c2 match:

>> % c = c1 = c2 =  [-3/154; -83/154; 309/77];

>> n = 3;

>> c = lstsqpoly (x, y, n)

c1 =

     -1/228

      5/266

   -983/1596

    535/133

c2 =

     -1/228

      5/266

   -983/1596

    535/133

N =

    454/1425

Current plot held

P =

- x^3/228 + (5\*x^2)/266 - (983\*x)/1596 + 535/133

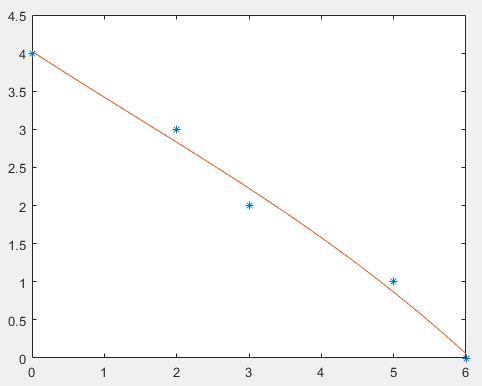
c =

     -1/228

      5/266

   -983/1596

    535/133



% When n = 3, all vectors of coefficients c, c1, and c2 match:

>> % c = c1 = c2 =  [-1/228; 5/266; -983/1596; 535/133];

>> n = 4;

>> c = lstsqpoly (x, y, n)

c1 =

     -1/40

     19/60

    -51/40

     59/60

      4

c2 =

     -1/40

     19/60

    -51/40

     59/60

      4

N =

      1/153929866311199

Current plot held

P =

- x^4/40 + (19\*x^3)/60 - (51\*x^2)/40 + (59\*x)/60 + 4

c =

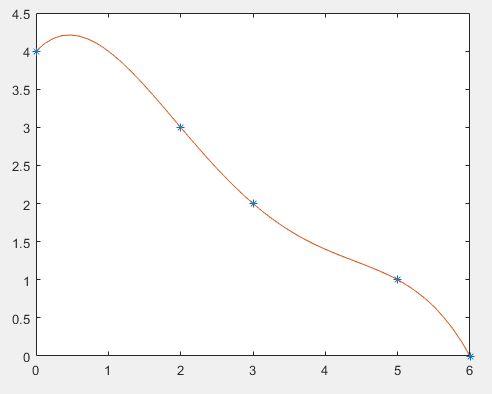
     -1/40

     19/60

    -51/40

     59/60

      4



% When n = 4, all vectors of coefficients c, c1, and c2 match:

>> % c = c1 = c2 =  [-1/40; 19/60; -51/40; 59/40; 4];

>> % When n = 1, the output is the same as it is in Exercise 5,

>> % both functions *lstsqline* and *lstsqpoly* give the same output for n = 1;

>> % Based on the plots and the norms of the residual vectors, polynomials of degree 4

>> % (n = 4) fits them the best.

>> diary off