Group 22

1. Kevin Lemos (Leader)
2. Johnny Li
3. Matt Leonard

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

```
Diary on

format compact

type eigen


function [  ] = eigen( A )

L = transpose(eig(A));

L = sort(L);

L = closetozeroroundoff(L);

disp("All the eigenvalues:")

disp(L);

disp("All the unique eigenvalues:")

M = unique(L);

disp(M);

Q = length(A);

disp("The sum of multiplicities of the eigenvalues is Q = " );

disp(Q);

disp("The following is a list of eigenvalues with a basis for the coresponding eigenspace:")

P = [];

for i = 1:length(M)

    disp(M(i));

    disp(" ");

    temp = A - (eye(Q)*M(i));
```

```
    P = [P null(temp, 'r')];
    disp(null(temp, 'r'));
    disp(" ");
end
disp("The total sum of the dimensions of the eigenspaces is N = ");
N = length(P(1, :));
disp(N);
if Q ~= N
    disp("No, matrix A is not diagonalizable since N does not equal Q")
else
    disp("Yes, matrix A is diagonalizable since N=Q");
    D = diag(L);
    F = closetozeroroundoff(A*P - P*D);
    if F == 0
        disp("Great! I got a diagonalization!");
    else
        disp("Oops! I got a bug in my code.");
    end

end
end


type closetozeroroundoff
```

```
function B=closetozeroroundoff(A)
[m,n]=size(A);
for i=1:m
for j=1:n
if abs(A(i,j))<10^(-7)
A(i,j) = 0;
end
end
end
B=A;
type jord

function J = jord (n, r)
A = ones(n);
J = tril(triu(A),1);
for i = 1: n
J(i, i) = r;
end
A = [2 2; 0 2];
eigen(A)
All the eigenvalues:
    2    2
All the unique eigenvalues:
    2
```

The sum of multiplicities of the eigenvalues is Q =

2

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

2

1

0

The total sum of the dimensions of the eigenspaces is N =

1

No, matrix A is not diagonalizable since N does not equal Q

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

eigen(A)

All the eigenvalues:

3 3 4 4

All the unique eigenvalues:

3 4

The sum of multiplicities of the eigenvalues is Q =

4

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

3

0

0

-0.2000

1.0000

4

0

0

0

1

The total sum of the dimensions of the eigenspaces is N =

2

No, matrix A is not diagonalizable since N does not equal Q

A = jord(5,3);

eigen(A)

All the eigenvalues:

3 3 3 3 3

All the unique eigenvalues:

3

The sum of multiplicities of the eigenvalues is Q =

5

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

3

1

0

0

0

0

The total sum of the dimensions of the eigenspaces is N =

1

No, matrix A is not diagonalizable since N does not equal Q

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

eigen(A)

All the eigenvalues:

1 2 2 3 3 3

All the unique eigenvalues:

1 2 3

The sum of multiplicities of the eigenvalues is Q =

6

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

1

0

0

0

0

0

1

2

| | |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 1 | 0 |
| 0 | 1 |
| 0 | 0 |

3

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

The total sum of the dimensions of the eigenspaces is N =

6

Yes, matrix A is diagonalizable since N=Q

Great! I got a diagonalization!

A = magic(4);

eigen(A)

All the eigenvalues:

-8.9443 0 8.9443 34.0000

All the unique eigenvalues:

-8.9443 0 8.9443 34.0000

The sum of multiplicities of the eigenvalues is Q =

4

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

-8.9443

-0.4570

-0.0287

-0.5143

1.0000

0

-1

-3

3

1

```
   8.9443


  -2.1882

   1.1254

   0.0627

   1.0000


  34.0000


   1.0000

   1.0000

   1.0000

   1.0000
```

The total sum of the dimensions of the eigenspaces is N =

```
    4
```

Yes, matrix A is diagonalizable since N=Q

Great! I got a diagonalization!

```
A= ones(5);

eigen(A)
```

All the eigenvalues:

```
    0     0     0     0     5
```

All the unique eigenvalues:

```
    0     5
```

The sum of multiplicities of the eigenvalues is Q =

5

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

0

| | | | |
|---|---|---|---|
| -1 | -1 | -1 | -1 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

5

1

1

1

1

1

The total sum of the dimensions of the eigenspaces is N =

5

Yes, matrix A is diagonalizable since N=Q

Great! I got a diagonalization!

A = magic(5);

eigen(A)

All the eigenvalues:

-21.2768 -13.1263 13.1263 21.2768 65.0000

All the unique eigenvalues:

-21.2768 -13.1263 13.1263 21.2768 65.0000

The sum of multiplicities of the eigenvalues is Q =

5

The following is a list of eigenvalues with a basis for the coresponding eigenspace:

-21.2768

-0.1440

-0.5200

-0.8114

0.4753

1.0000

-13.1263

-2.4172

2.2511

-1.4952

0.6613

1.0000

13.1263

-0.4137

-0.2736

0.6186

-0.9313

1.0000

21.2768

65.0000

The total sum of the dimensions of the eigenspaces is N =

3

No, matrix A is not diagonalizable since N does not equal Q

%this does not make sense as a magic square should be diagonizable

type eigen_1

function [  ] = eigen( A )

L = transpose(eig(A));

L = sort(L);

disp("All the eigenvalues:")

```
disp(L);

disp("All the unique eigenvalues:")

M = unique(L);

disp(M);

Q = length(A);

disp("The sum of multiplicities of the eigenvalues is Q = " );

disp(Q);

disp("The following is a list of eigenvalues with a basis for the coresponding
eigenspace:")

P = [];

for i = 1:length(M)

   disp(M(i));

   disp(" ");

   temp = A - (eye(Q)*M(i));

   P = [P null(temp)];

   disp(null(temp, 'r'));

   disp(" ");

end

disp("The total sum of the dimensions of the eigenspaces is N = ");

N = length(P(1, :));

disp(N);

if Q ~= N

   disp("No, matrix A is not diagonalizable since N does not equal Q")

else
```

```
    disp("Yes, matrix A is diagonalizable since N=Q");
    D = diag(L);
    F = closetozeroroundoff(A*P - P*D);
    if F == 0
        disp("Great! I got a diagonalization!");
    else
        disp("Oops! I got a bug in my code.");
    end

end
end
```

```
eigen_1(A)
All the eigenvalues:
  -21.2768  -13.1263   13.1263   21.2768   65.0000
All the unique eigenvalues:
  -21.2768  -13.1263   13.1263   21.2768   65.0000
The sum of multiplicities of the eigenvalues is Q =
     5
The following is a list of eigenvalues with a basis for the coresponding eigenspace:
  -21.2768


   -0.1440
   -0.5200
```

-0.8114

0.4753

1.0000

-13.1263

-2.4172

2.2511

-1.4952

0.6613

1.0000

13.1263

-0.4137

-0.2736

0.6186

-0.9313

1.0000

21.2768

65.0000

The total sum of the dimensions of the eigenspaces is N =

5

Yes, matrix A is diagonalizable since N=Q

Great! I got a diagonalization!

%all i had to do was find the null space, not necissailty a ration basis of of, of A - I

diary off

%excerise 2

diary on

format compact

A = [2 2; 0 2];

diagonal(A)

The number of linearly independent columns in P is k =

1

A is not diagonizable.

A does not have enough linearly independent eigenvectors to create a basis for R^n

ans =

2 2

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

A =

4 0 0 0

1 3 0 0

0 -1 3 0

0 -1 5 4

diagonal(A)

The number of linearly independent columns in P is k =

2

A is not diagonizable.

A does not have enough linearly independent eigenvectors to create a basis for R^n

ans =

4 3 3 4

A = jord(5,3);

{ Undefined function or variable 'jord'.}

A = jord(5,3);

diagonal(A)

The number of linearly independent columns in P is k =

1

A is not diagonizable.

A does not have enough linearly independent eigenvectors to create a basis for R^n

ans =

3 3 3 3 3

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

diagonal(A);

The number of linearly independent columns in P is k =

6

A is diagonalizable.

A basis for R^n is:

```
   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
```

A = magic(4);

diagonal(A);

The number of linearly independent columns in P is k =

```
    4
```

A is diagonalizable.

A basis for R^n is:

```
  -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
```

A = ones(5);

diagonal(A);

The number of linearly independent columns in P is k =

```
    5
```

A is diagonalizable.

A basis for R^n is:

```
    0.4082    0.7071    0.2236    0.2887    0.4472

    0.4082   -0.7071    0.2236    0.2887    0.4472
```

```
  -0.8165   -0.0000    0.2236    0.2887    0.4472
        0   -0.0000    0.2236   -0.8660    0.4472
        0         0   -0.8944         0    0.4472
A = magic(5);
diagonal(A)
The number of linearly independent columns in P is k =
     5
A is diagonalizable.
A basis for R^n is:
  -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
ans =
  65.0000  -21.2768  -13.1263   21.2768   13.1263
diary off
%Exercise 3
diary on
format compact
type shrink
function B = shrink(A)
%Function given.
[~,pivot] = rref(A);
```

```
B = A(:,pivot);

end

type proj

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

%The function computes the projection of b onto the Column Space of

%an matrix A. The program allow a possibility that the

%columns of A are not linearly independent. The algorithm create a

%basis for Col A.

%Required. Code Given

format compact,

A=shrink(A);

b=transpose(b);

%Set the dimension of matrix A. (m=columns, n=rows)due to shrink function.

[m,n]=size(A);

%Remove Negative zeros.

tol = 1.e-7;

%Check if the input vector b has exactly m entries.

%If the porgram doesn’t have m entries, the program breaks with a

%message and returns empty vectors p and z.

if(size(b)~= m)

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

 p = [ ];

 z = [ ];

 return
```

```
else
%If b has exactly m components,determine whether it is a case when the
%vector b ? Col A.
if rank(A)== rank([A,b])
%If vector b ? Col A, outputs (without any computations): the
%vector p, vector z, and a message. After that, the program
%terminates.
p=b;
z=b-p;
%Remove Negative zeros.
p(p<0 & p>-tol) = 0;
z(z<0 & z>-tol) = 0;
disp("b is in the Col A")
return
%If vector b ? Col A,(1)Determine whether it is a case when b is
%orthogonal to Col A.
else
true=1;
for i =1:n
if(abs(dot(A(:,i),b))>10^-7)
%b is not orthogonal to Col A
true=0;
end
```

```
end

%If b is orthogonal to Col A,outputs (without any computations):
%the vector p, vector z, and a message. After that, the program
%terminates.
if(true == 1)
p=b;
z=b-p;
%Remove Negative zeros.
p(p<0 & p>-tol) = 0;
z(z<0 & z>-tol) = 0;
disp("b is orthogonal to Col A")
return;
%If b is not orthogonal to Col A,
else
if (true ==0)
%Find the solution of the normal equations x^.
%Then calculate vector p = Ax^ . The vector z = b – p.
x=(A'*A)\(A'*b);
p=A*x;
z=b-p;
%Remove Negative zeros.
p(p<0 & p>-tol) = 0;
z(z<0 & z>-tol) = 0;
```

```
%Check whether the vector p is orthogonal to the vector z.
if(abs(dot(p,z))<10^-7)
disp("Yes, p and z are orthogonal! Great Job!")
else
disp("Oops! Is there a bug in my code?")
end
end
end
end
end
end
%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)
```

Yes, p and z are orthogonal! Great Job!

p =

0.9492

2.1599

2.9492

3.9180

5.1287

5.9180

z =

0.0508

-0.1599

0.0508

0.0820

-0.1287

0.0820

%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)

Yes, p and z are orthogonal! Great Job!

p =

-0.2500

0.0000

0.2500

0.2500

0.0000

0.7500

z =

0.2500

0.0000

-0.2500

-0.2500

0.0000

0.2500

%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 =

0.8147 0.9058 0.1270 0.9134 0.6324

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

b is in the Col A

p =

0.8147

0.9058

0.1270

0.9134

0.6324

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)

Yes, p and z are orthogonal! Great Job!

p =

0.7143

0.6286

0.5429

0.4571

0.3714

0.2857

z =

0.2857

-0.6286

0.4571

-0.4571

0.6286

-0.2857

%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

diary off

```
%Exercise 4
>> type shrink
function B = shrink(A)
%Function given.
[~,pivot] = rref(A);
B = A(:,pivot);
end
>> type solvemore
```

```
%Exercise 4

>> type shrink

function X = solvemore(A,b)

%The function find the “exact” solution of a matrix equation Ax=b if the

%equation is consistent, b?Col A; or find the “least-squares” solution

%if the equation is inconsistent, b?Col A .

%Required. Code Given

format long,

A=shrink(A);

[m, n] = size(A);

%To check whether b?Col A or b?Col A

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

 %Part A: If b?Col A

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


 %Required. Code Given

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


 %Part 1A: does not have orthonormal columns

 if(B~=zeros(n))

 disp("A does not have orthonormal columns")

 %the solution of Ax=b

 x1=A\b;

 %Required. Code Given
```

```
X=x1;

end


if(B==zeros(n))

%Part 2A: has orthonormal columns but is not orthogonal

if(m~=n)

disp("A has orthonormal columns but is not orthogonal")

%the solution of Ax=b

x1=A\b;

%Required. Code Given

X=x1;

%Part 3A: is orthogonal

else

disp("A is orthogonal")

x1=A\b;

x2=A'*b;

%Required. Code Given

X=[x1,x2];

N=norm(x1-x2);


fprintf("The norm of the difference between two solutions is N
=\n",N);

end

end
```

```
%Part B: If b?Col A
else
 disp("The system is inconsistent – look for the least-squares solution")
 %Part 1B: unique least-squares solution by solving the normal equations
 %and estimate the least-squares error of the approximation
 x3=((A'*b)\(A'*A))';
 disp("The solution of the normal equations is x3=");
 disp(x3)
 %Required. Code Given
 n1=norm(b-A*x3);


 disp("The least-squares error of the approximation is n1=");
 disp(n1)


 %Part 2B: Find an orthonormal basis U for Col A
 %Check if the matrix A has orthonormal columns (part (a))
 B=closetozeroroundoff(A'*A-eye(n));
 if(B==zeros(n))
 disp("A has orthonormal columns: an orthonormal basis for Col A is
U=A")
 %Required. Code Given
 U=A;
 end
 if(B~=zeros(n))
```

```
%Required. Code Given
U = orth(A);

disp("An orthonormal basis for Col A is U=");
disp(U)
end

%Part 3B: calculate the vector b1, least-squares solution of x4, and
%estimate the least-squares error of the approximation of b by the
%vectors in Col A
%Required. Code Given
b1=U*U'*b;
disp("The projection of b onto Col A is");
disp(b1)

x4=A\b1;
disp("The least-squares solution by using the projection onto Col A is
x4=");
disp(x4)

n2 = norm(b-A*x4);
disp("The least-squares error of this approximation is n2 =");
disp(n2)
```

```
%Part 4B: Calculate the norm n3 of the difference between the
%solutions x3 and x4
n3 = norm(x3 - x4);
disp("The norm of the difference between the solutions is x3= ");
disp(n3)

x=rand(n,1);
n4=norm(b-A*x);
fprintf("An error of approximation of b by Ax for a random vector x in
R^%i is %e",n,n4);
X=[x3 x4];
end
end
>> %a)
>> A=magic(4); b=A( : ,1), A=orth(A)
b =
16
5
9
4
A =
-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 equation is consistent – look for the exact solution

A has orthonormal columns but is not orthogonal

X =

-16.999999999999993

8.944271909999161

3.000000000000000

>> %b)

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

A =

Columns 1 through 3

-0.447213595499958 -0.545634873129948 0.511667273601714

-0.447213595499958 -0.449758363151205 -0.195439507584838

-0.447213595499958 -0.000000000000024 -0.632455532033676

-0.447213595499958 0.449758363151189 -0.195439507584872

-0.447213595499958 0.545634873129987 0.511667273601672

Columns 4 through 5

0.195439507584854 -0.449758363151198

-0.511667273601691 0.545634873129969

0.632455532033676 -0.000000000000002

-0.511667273601694 -0.545634873129966

0.195439507584856 0.449758363151196

b =

0.814723686393179

0.905791937075619

0.126986816293506

0.913375856139019

0.632359246225410

>> 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 =

X =

-1.517501961599936 -1.517501961599936

-0.096093467150121 -0.096093467150121

0.304574206628231 0.304574206628231

-0.567677934732546 -0.567677934732546

-0.086157982822825 -0.086157982822825

>> %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 =

0.097540404999410

0.278498218867048

0.546881519204984

0.957506835434298

>> X = solvemore(A,b)

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

The solution of the normal equations is x3=

13.219850693725473

17.863381290095088

17.935052780168906

The least-squares error of the approximation is n1=

8.573128698484909e+02

An orthonormal basis for Col A is U=

-0.363225569906992 -0.839773278980323 0.335928601456289

-0.511952614823082 0.201051551215513 -0.497476425501395

-0.413098103234259 -0.228706948321817 -0.571876812690966

-0.659789104673461 0.449502219631666 0.559129763025005

The projection of b onto Col A is

0.180796221571844

0.528265668584352

0.297114069487679

0.874251018861863

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

-0.001240611967882

-0.031004149639099

0.087551437445385

The least-squares error of this approximation is n2 =

0.372331330756435

The norm of the difference between the solutions is x3=

28.522615408790429

An error of approximation of b by Ax for a random vector x in R^3 is

3.680584e+01

```
X =

 13.219850693725473 -0.001240611967882

 17.863381290095088 -0.031004149639099

 17.935052780168906 0.087551437445385

>> %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 =

 0.392227019534168

 0.655477890177557

 0.171186687811562

 0.706046088019609

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

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

The solution of the normal equations is x3=

```
 -0.919766830206403

 -0.304662471966935

 0.129779014755094
```

The least-squares error of the approximation is n1=

0.258712381419412

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

The projection of b onto Col A is

0.335274292603541

0.484619709385675

0.342044868603444

0.762998814950236

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

-0.962468842771448

-0.318807035870279

0.135804264782329

The least-squares error of this approximation is n2 =

0.254700337841731

The norm of the difference between the solutions is x3= \n

0.045385396362836

An error of approximation of b by Ax for a random vector x in R^3 is

1.190141e+00

X =

-0.919766830206403 -0.962468842771448

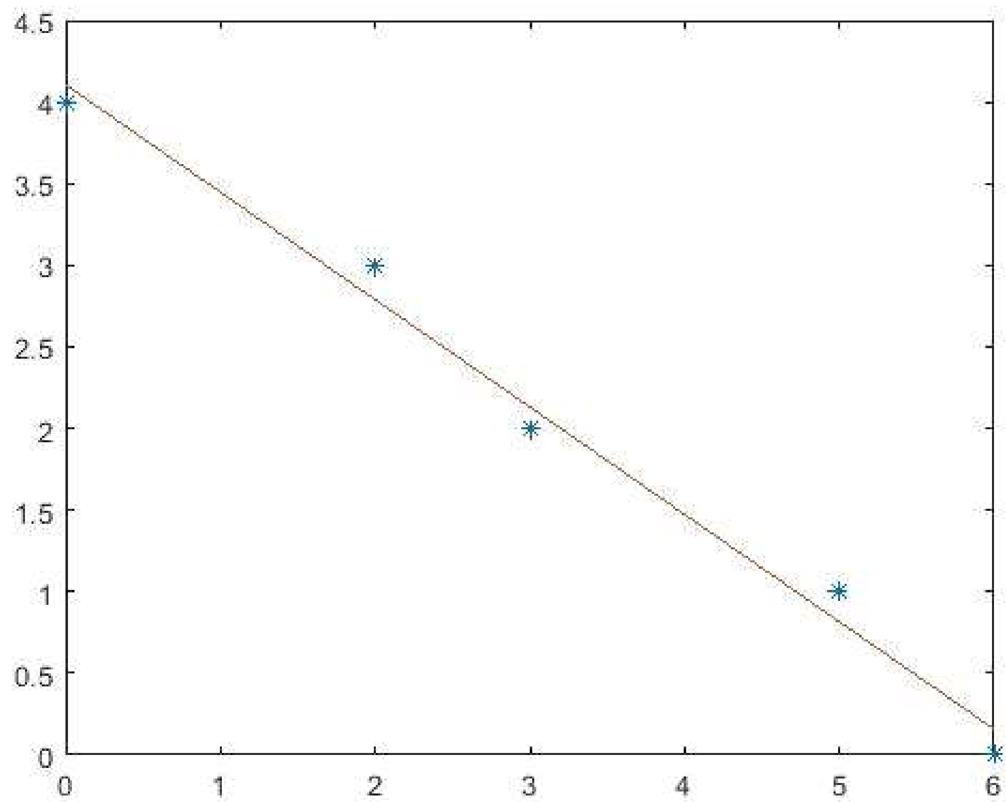
-0.304662471966935 -0.318807035870279

0.129779014755094 0.135804264782329

```
diary Project4
diary on
format compact
```

```
 %Exercise 5
%Matthew Leonard
 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);

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 =
```

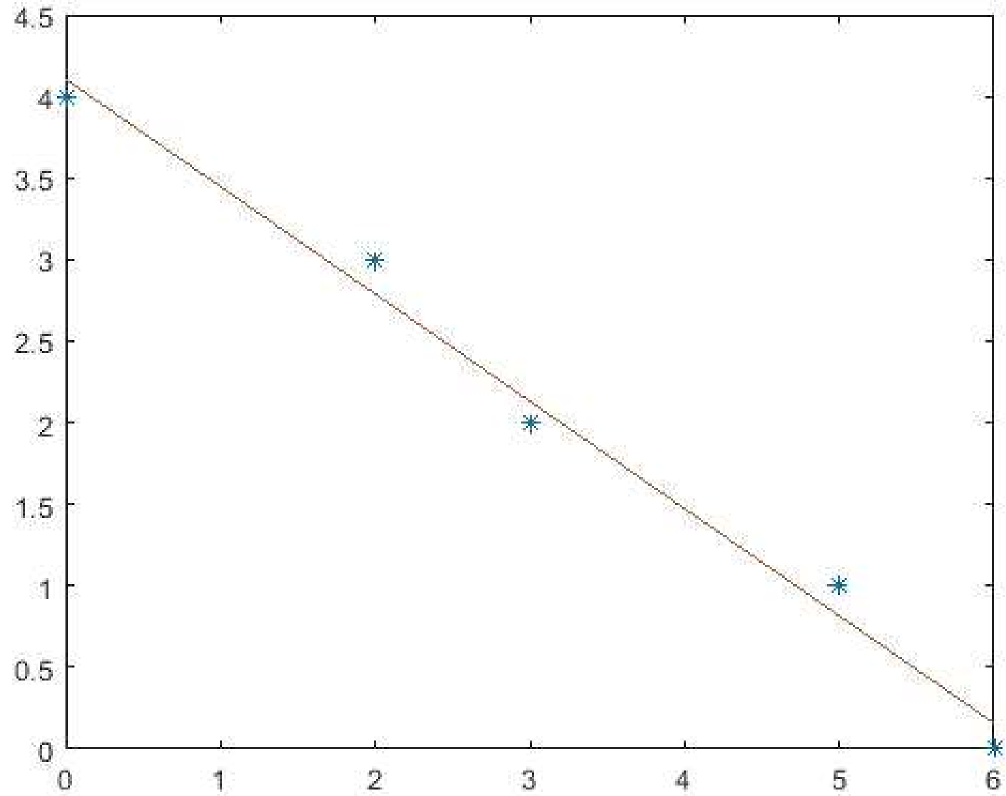
```
   -25/38
    78/19
c2 =
   -25/38
    78/19
N =
   514/1417
Warning: MATLAB has disabled some advanced graphics rendering features by
switching to
software OpenGL. For more information, click here.
Current plot held
P =
78/19 - (25*x)/38
c =
   -25/38
    78/19
```




```
%
%
%Exercise 6
```

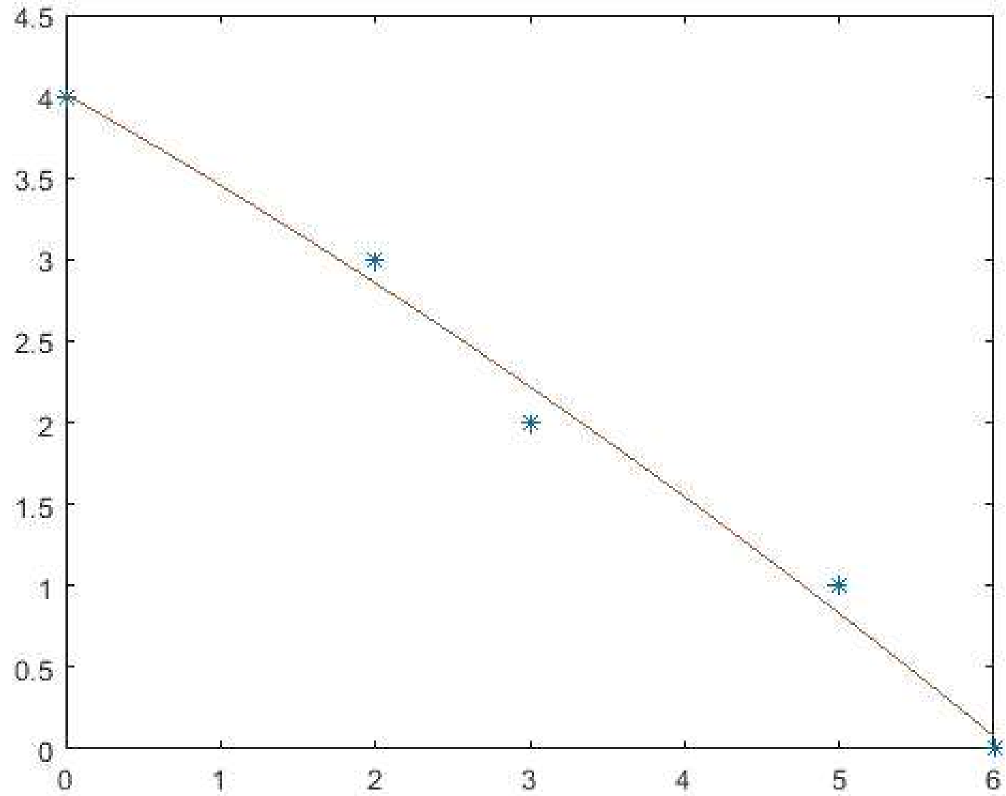
```
%Matthew Leonard
type lstsqpoly

function c = lstsqpoly(x, y, n)
format rat,
x = x';
y = y';
a = x(1);
m = length(x);
b = x(m);
for i = 1:n
   X(:,n-i+1) = x.^i;
end
X(:,n+1) = ones(m,1);
c = lscov(X, y);
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

c = lstsqpoly(x,y,1)
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
%c1, c2, and c all match.
%n=1 is consistent for both lstsqpoly and lstsqline because the matrix is designed the
```

same



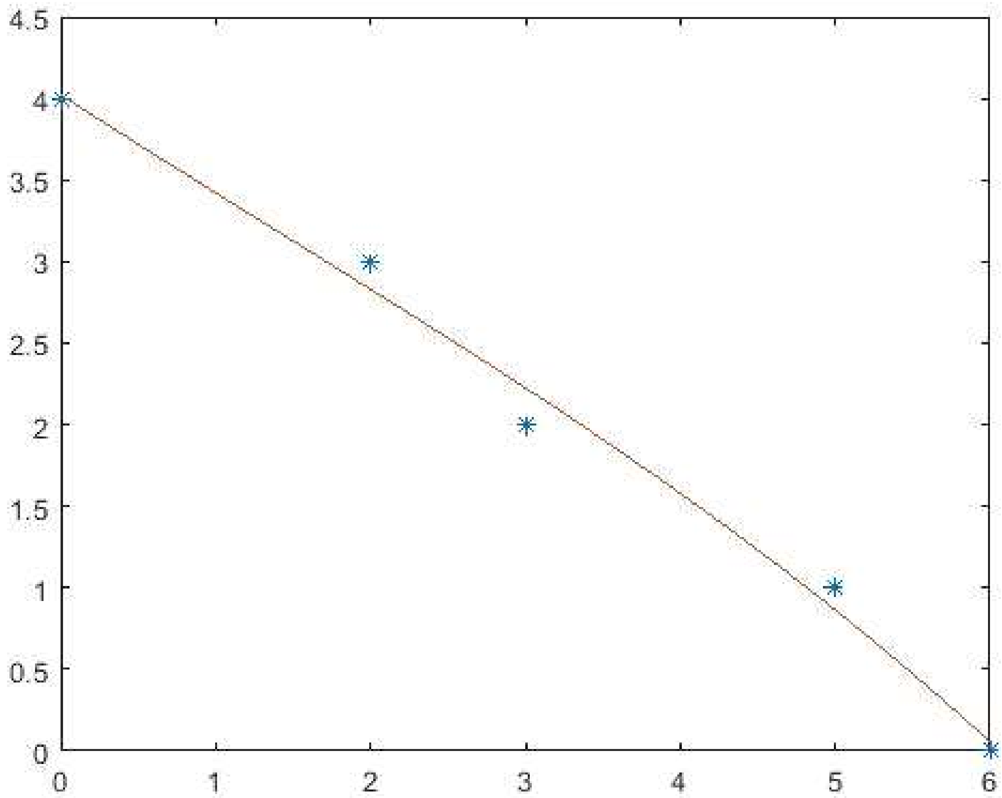

c = lstsqpoly(x,y,2)
c1 =
-3/154
-83/154
309/77
c2 =
-3/154
-83/154
309/77
N =
703/2181
Current plot held
P =
309/77 - (3*x^2)/154 - (83*x)/154
c =
-3/154
-83/154
309/77

%The coeffs of c1, c2, and c all match.



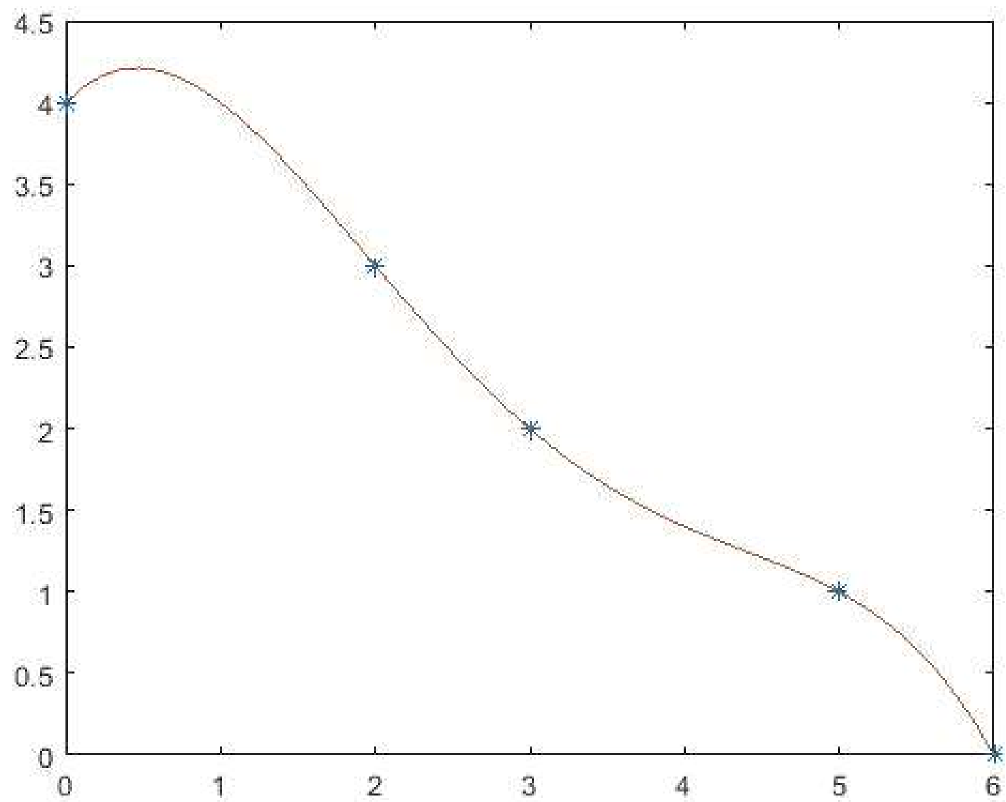

c = lstsqpoly(x,y,3)
c1 =
-1/228
5/266
-983/1596
535/133
c2 =
-1/228
5/266
-983/1596
535/133
N =
454/1425
Current plot held
P =
- x^3/228 + (5417864213378195*x^2)/288230376151711744 - (983*x)/1596 + 535/133
c =
-1/228

```
   5/266
 -983/1596
  535/133
%The coeffs of c1 c2 and c all match.
```




```
c = lstsqpoly(x,y,4)
c1 =
    -1/40
    19/60
   -51/40
    59/60
     4
c2 =
    -1/40
    19/60
   -51/40
    59/60
     4
N =
     1/77464268540332
```

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
%The coeffs of c1, c2, and c all match.




%The plot of n=4 shows strong correlation with the points and the norm proves this as well. Therefore, n=4 is the best fitted line.
diary off