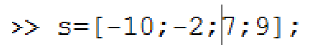
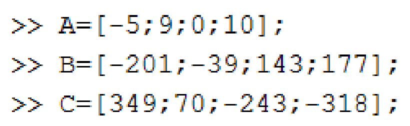
Christian Craig
CSC 412
hw6

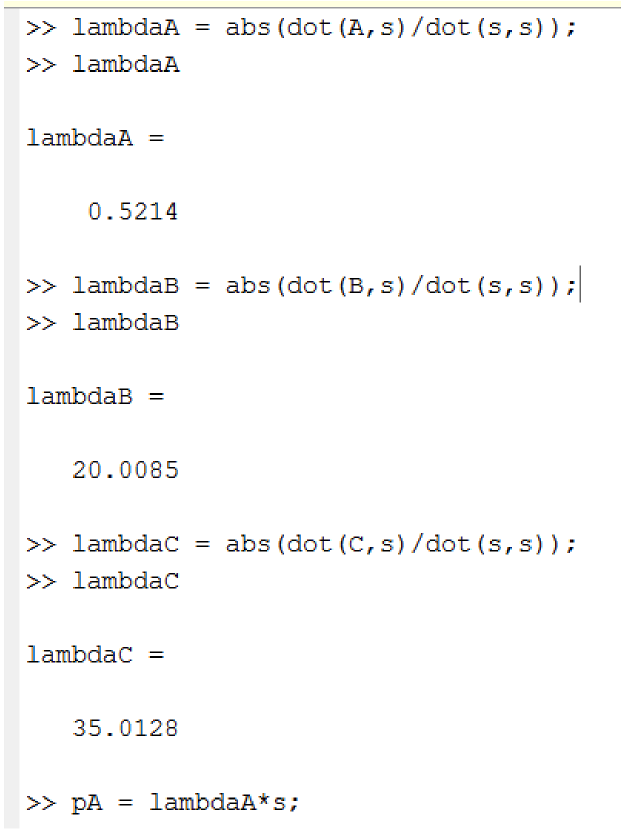
2. (Higher Dimensional Fire Engine Problem, 5pt) You're looking to buy a property in 4-dimensional space. There's only one (straight, infinite) road in your universe, it leads through the origin, and it has direction [-10;-2;7;9]. There are three properties to choose from, property A at [-5;9;0;10] property B at[-201;-39;143;177], and property C at[349;70;-243;-318]. You are very sensitive to noise, and the only road in your universe is very busy, so you want to buy the property which is as far away from the road as possible. Which property do you choose? Hint: for each property determine how far it is from the road. Include those distances in your answer. Hint 2: you'll need projections, and orthogonal projections for each of the three points.



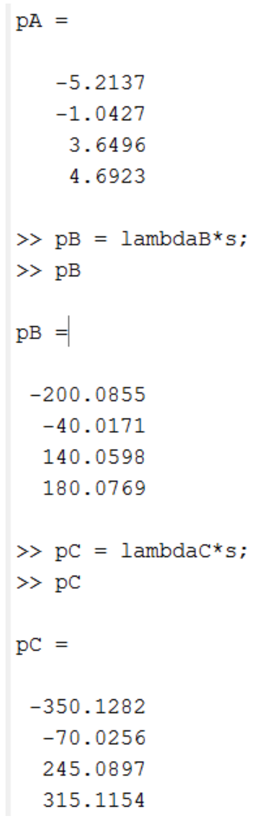
```
>> s=[-10;-2;7;9];
```



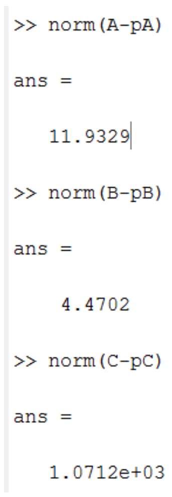
```
>> A=[-5;9;0;10];
>> B=[-201;-39;143;177];
>> C=[349;70;-243;-318];
```



```
>> lambdaA = abs(dot(A,s)/dot(s,s));
>> lambdaA

lambdaA =

    0.5214

>> lambdaB = abs(dot(B,s)/dot(s,s));
>> lambdaB

lambdaB =

   20.0085

>> lambdaC = abs(dot(C,s)/dot(s,s));
>> lambdaC

lambdaC =

   35.0128

>> pA = lambdaA*s;
```



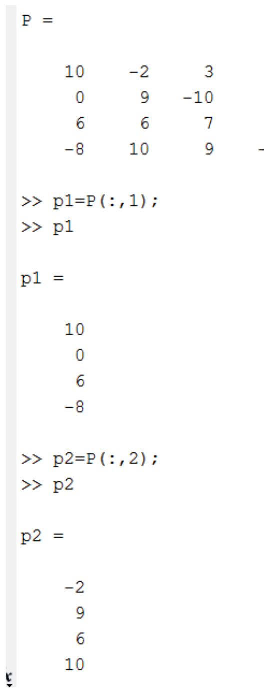
```
pA =

   -5.2137
   -1.0427
    3.6496
    4.6923

>> pB = lambdaB*s;
>> pB

pB =

 -200.0855
  -40.0171
  140.0598
  180.0769

>> pC = lambdaC*s;
>> pC

pC =

 -350.1282
  -70.0256
  245.0897
  315.1154
```



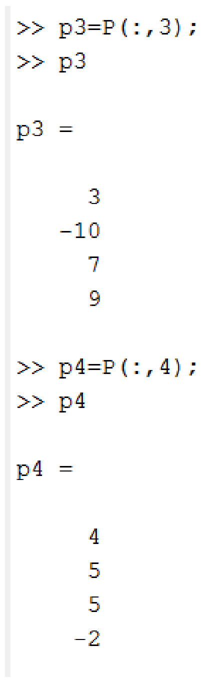
```
>> norm(A-pA)

ans =

   11.9329

>> norm(B-pB)

ans =

    4.4702

>> norm(C-pC)

ans =

   1.0712e+03
```

Property A returns the largest value, which tells us that it's furthest away from the road

3. (Hyperplane, 5pt) In this problem you want to find a hyperplane through the points in P = [10 -2 3 4;0 9 -10 5;6 6 7 5;-8 10 9 -2], where each column represents a point. To do this, follow these steps:

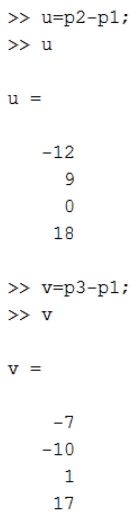


```
P =

    10    -2     3
     0     9   -10
     6     6     7
    -8    10     9    -

>> p1=P(:,1);
>> p1

p1 =

    10
     0
     6
    -8

>> p2=P(:,2);
>> p2

p2 =

    -2
     9
     6
    10
```

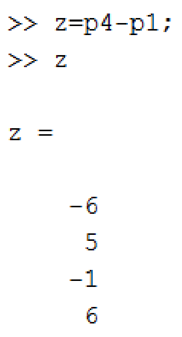


```
>> p3=P(:,3);
>> p3

p3 =

     3
   -10
     7
     9

>> p4=P(:,4);
>> p4

p4 =

     4
     5
     5
    -2
```

a) [2pt] Pick one of the points (columns), let's say the first one, and compute the differences between the other three columns and that column. (In class we did this using matrix muliplication, but you can also name the columns, say p1, p2, p3 and p4, and calculate p2-p1, p3-p1 and p4-p1.).

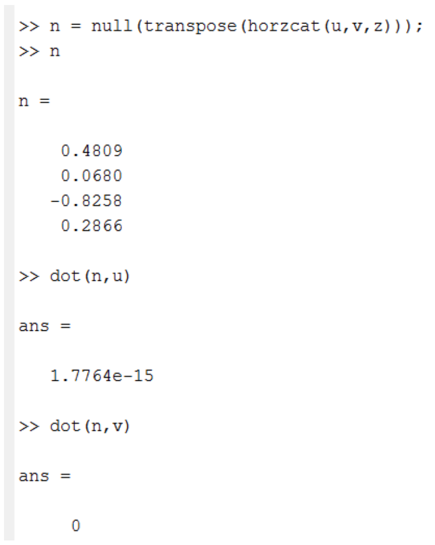


```
>> u=p2-p1;
>> u

u =

   -12
     9
     0
    18

>> v=p3-p1;
>> v

v =

    -7
   -10
     1
    17
```

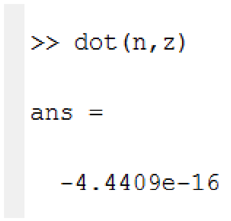


```
>> z=p4-p1;
>> z

z =

    -6
     5
    -1
     6
```

b) [2pt] Determine a normal vector for of the three difference vectors you determined in a), that is, find a unit vector n that is orthogonal to each of the three difference vectors. Hint: use null() on the appropriate matrix. A very similar question was on the last homework.



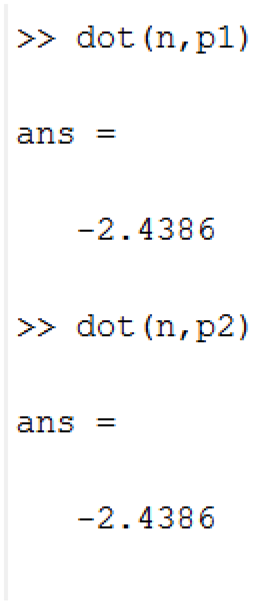
```
>> n = null(transpose(horzcat(u,v,z)));
>> n

n =

    0.4809
    0.0680
   -0.8258
    0.2866

>> dot(n,u)

ans =

   1.7764e-15

>> dot(n,v)

ans =

     0
```



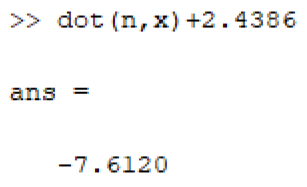
```
>> dot(n,z)

ans =

  -4.4409e-16
```

c) [1pt] The hyerplane is H = {x: dot(n,x) = dot(n,p)}, where p is a point in the hyperplane (e.g. p1). The expression **<u>dot(n,x)-dot(n,p)</u>** is the distance (positive or negative) of a point x from the hyperplane H. Calculate that distance for x = [6;-13;8;-19].

"+" because subtracting negative



```
>> dot(n,p1)

ans =

   -2.4386

>> dot(n,p2)

ans =

   -2.4386
```

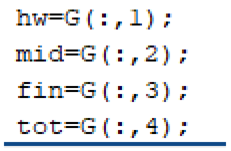


```
>> dot(n,x)+2.4386

ans =

   -7.6120
```

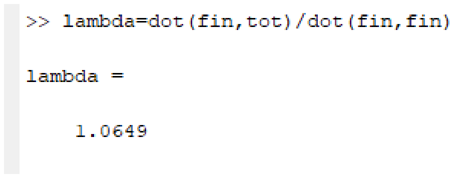
below hyperplane

4. (Regression, 10pt +5EC) You have gotten access to the following (fictitious) data from a class. It's in a matrix G, stored in the file grades.mat in hw6-4.zip (save the zip file, extract the grades.mat file; don't double-click on the .mat file, Access will grab it; start Matlab, and use Open to open the file hw6-4.mat; the new variable G should then be available in Matlab). There is a row for every student. The first column is the student's score on the homework (hw), the second column the sconre on the midterm (mid), the third column the score on the final (fin), and the fourth, and last, column, the total class score second (tot). Using regression analysis, we want to test how well the class score is predicted by the midterm and final scores using two different models.

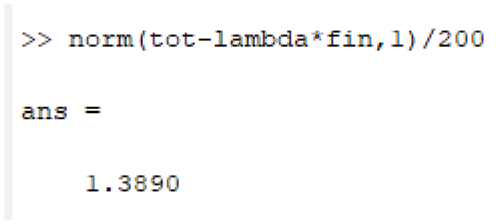
a) [4pt]Try a simple linear model, that is, we test whether tot = lambda*fin for some factor lambda. Use linear regression to find the best value for lambda. With that value, calculate the average error per student, that is, ||tot-lambda*fin||_1 /(number of students). Note: the norm here is the 1-norm.



```
hw=G(:,1);
mid=G(:,2);
fin=G(:,3);
tot=G(:,4);
```

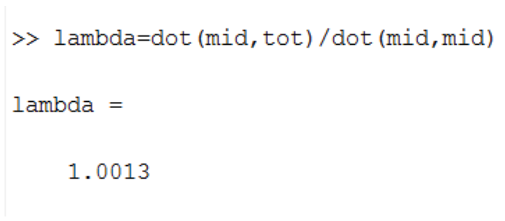


```
>> lambda=dot(fin,tot)/dot(fin,fin)

lambda =

    1.0649
```

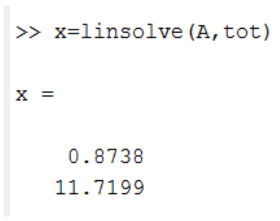


```
>> norm(tot-lambda*fin,1)/200

ans =

    1.3890
```

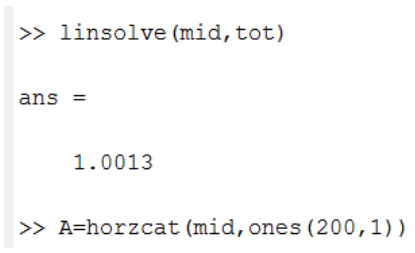
b.) [4pt] Next, try an affine model, in which we add a fixed percentage to the midterm score, so we suspect that tot = lambda*mid + mu. Use linear regression to find the best values for lambda and mu. With those values, calculate the average error per student, that is, ||tot-lambda*mid - mu||_1/(number of students). Note: again, work with the 1-norm.



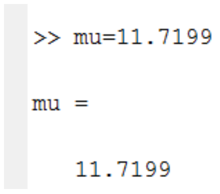
```
>> lambda=dot(mid,tot)/dot(mid,mid)

lambda =

    1.0013
```



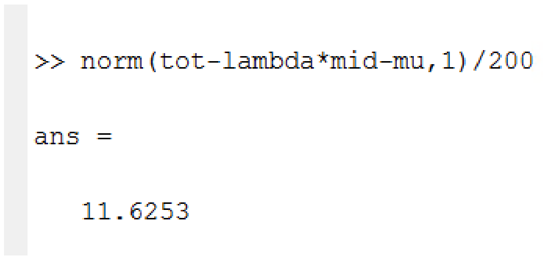
```
>> x=linsolve(A,tot)

x =

    0.8738
   11.7199
```



```
>> linsolve(mid,tot)

ans =

    1.0013

>> A=horzcat(mid,ones(200,1))
```



```
>> mu=11.7199

mu =

   11.7199
```

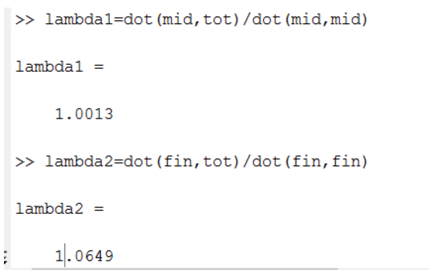


```
>> norm(tot-lambda*mid-mu,1)/200

ans =

   11.6253
```

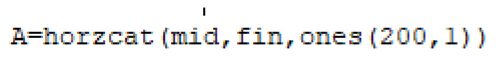
c) [2pt] Based on your results from a) and b), which model is more accurate?

Comparing the error results of the two, I would select the first model.

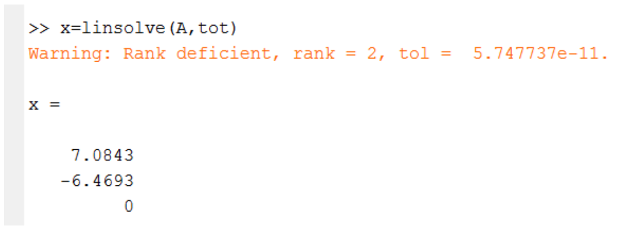
d) [Extra credit, 5EC] Try a multilinear model, that is, try tot = lambda1*mid + lambda2*fin + mu. Use linear regression to find the best values for lambda1, lambda2 and mu, and calculate the average error per student.



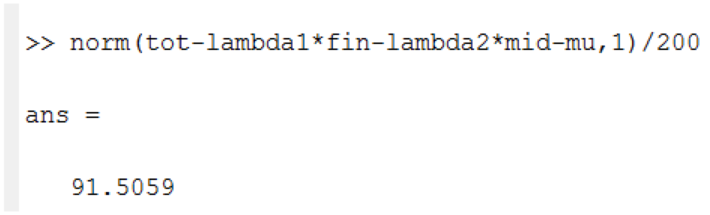
```
>> lambda1=dot(mid,tot)/dot(mid,mid)

lambda1 =

    1.0013

>> lambda2=dot(fin,tot)/dot(fin,fin)

lambda2 =

    1.0649
```



```
A=horzcat(mid,fin,ones(200,1))
```



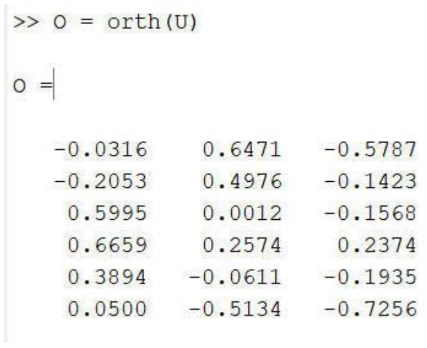
```
>> x=linsolve(A,tot)
Warning: Rank deficient, rank = 2, tol =  5.747737e-11.

x =

    7.0843
   -6.4693
         0
```



```
>> norm(tot-lambda1*fin-lambda2*mid-mu,1)/200

ans =

   91.5059
```

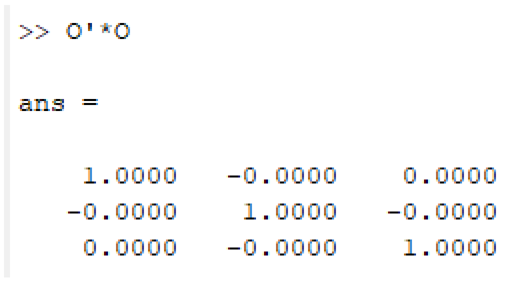
Note: You can use any of the methods we have seen to perform the regression analysis (normal equations, or linsolve directly). Do not use any other built-in linear regression functionality of Matlab.

5.) (Coordinates, 10pt) We are given a set of three vectors U = [-5 -7 3;-7 -3 1;10 -7 -1;9 -10 -4;7 -4 0;5 4 5] . The corresponding matrix has rank 3, so U is a basis of a three-dimensional subspace of R6. Using Matlab's orth function, you calculate an orthonormal basis of U:



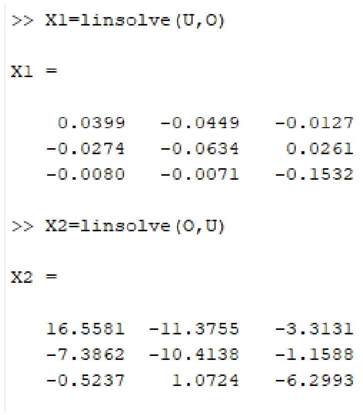
```
>> O = orth(U)

O =

   -0.0316    0.6471   -0.5787
   -0.2053    0.4976   -0.1423
    0.5995    0.0012   -0.1568
    0.6659    0.2574    0.2374
    0.3894   -0.0611   -0.1935
    0.0500   -0.5134   -0.7256
```

a) [1pt] Verify that O is an orthonormal basis (which is the same as O being an orthogonal matrix). Hint: this is easy. One matrix multiplication.

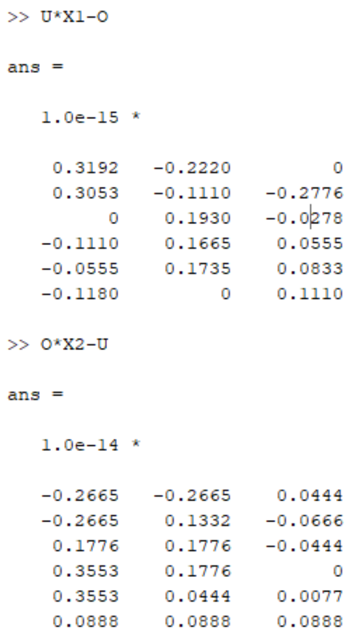


```
>> O'*O

ans =

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

b) [3pt] Verify that O and U span the same subspace (so O is an orthonormal basis of span(U)). Hint: we've seen two ways to do this: use the linsolve method here: X1 = linsolve(U,O) and X2 = linsolve(O,U), and then look at UX1-O and OX2-U.



```
>> X1=linsolve(U,O)

X1 =

    0.0399   -0.0449   -0.0127
   -0.0274   -0.0634    0.0261
   -0.0080   -0.0071   -0.1532

>> X2=linsolve(O,U)

X2 =

   16.5581  -11.3755   -3.3131
   -7.3862  -10.4138   -1.1588
   -0.5237    1.0724   -6.2993
```

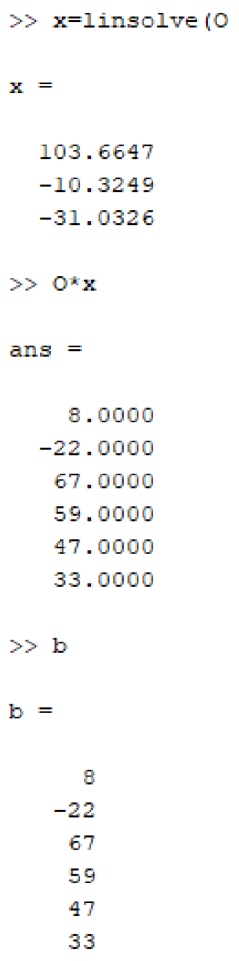


```
>> U*X1-O

ans =

   1.0e-15 *

    0.3192   -0.2220         0
    0.3053   -0.1110   -0.2776
         0    0.1930   -0.0278
   -0.1110    0.1665    0.0555
   -0.0555    0.1735    0.0833
   -0.1180         0    0.1110

>> O*X2-U

ans =

   1.0e-14 *

   -0.2665   -0.2665    0.0444
   -0.2665    0.1332   -0.0666
    0.1776    0.1776   -0.0444
    0.3553    0.1776         0
    0.3553    0.0444    0.0077
    0.0888    0.0888    0.0888
```

Both are close to zero, so we can say they span the same subspace

c) [3pt] The vector b = [8;-22;67;59;47;33] lies in the span of U (it is U*[5;-3;4]). We want to write b in the basis O, that is, we want to find x so that Ox = b. We'll do this in two ways:

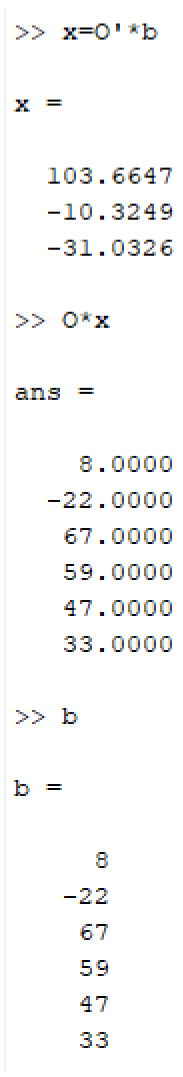
1. Use linsolve to find x. Verify the answer



```
>> x=linsolve(O

x =

  103.6647
  -10.3249
  -31.0326

>> O*x

ans =

    8.0000
  -22.0000
   67.0000
   59.0000
   47.0000
   33.0000

>> b

b =

     8
   -22
    67
    59
    47
    33
```

2. Use the methods of Fourier coefficients. Hint: This requires one simple matrix multiplication, no linsolve, no matrix inversion. Remember that O'*O = I.

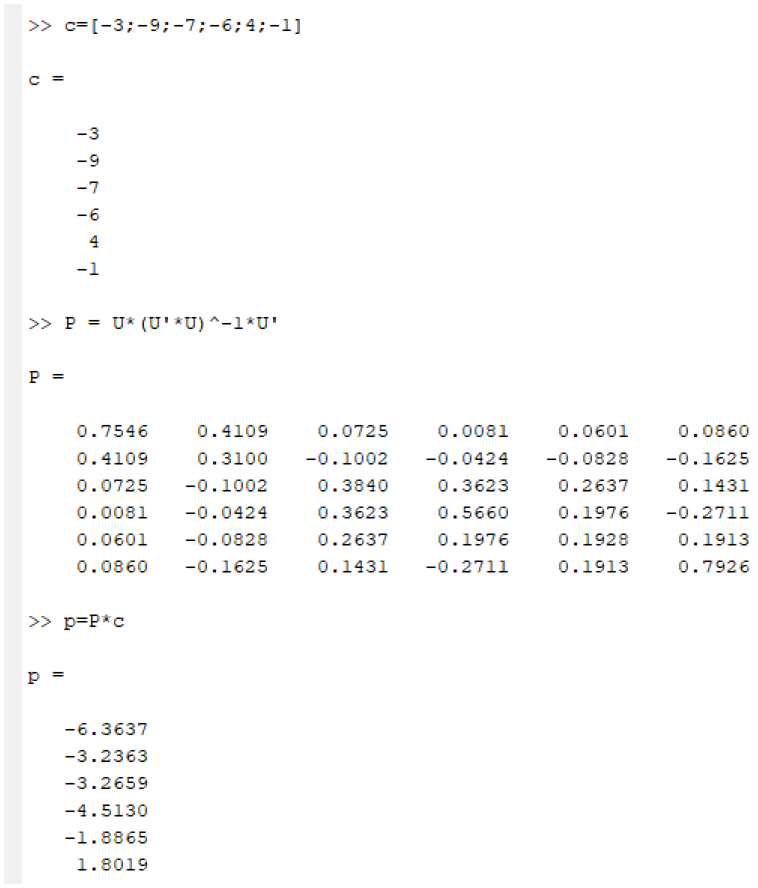
If we have an orthonormal basis, no longer need linsolve



```
>> x=O'*b

x =

  103.6647
  -10.3249
  -31.0326

>> O*x

ans =

    8.0000
  -22.0000
   67.0000
   59.0000
   47.0000
   33.0000

>> b

b =

     8
   -22
    67
    59
    47
    33
```

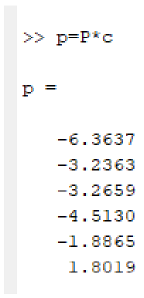
The two results should be the same.

d) [3pt] The vector c=[-3;-9;-7;-6;4;-1] does not line in the span of O (which is the same as the span of U). Find the projection p of c onto O (or U, the same thing).

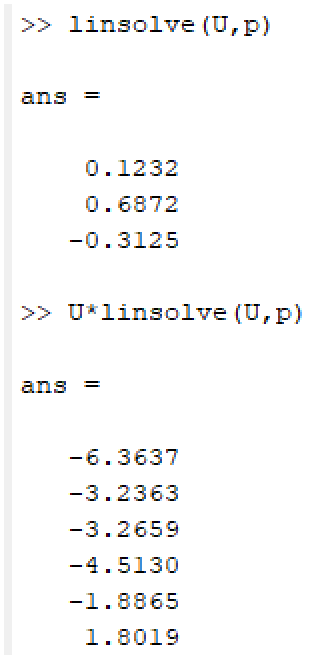


```
>> c=[-3;-9;-7;-6;4;-1]

c =

    -3
    -9
    -7
    -6
     4
    -1

>> P = U*(U'*U)^-1*U'

P =

    0.7546    0.4109    0.0725    0.0081    0.0601    0.0860
    0.4109    0.3100   -0.1002   -0.0424   -0.0828   -0.1625
    0.0725   -0.1002    0.3840    0.3623    0.2637    0.1431
    0.0081   -0.0424    0.3623    0.5660    0.1976   -0.2711
    0.0601   -0.0828    0.2637    0.1976    0.1928    0.1913
    0.0860   -0.1625    0.1431   -0.2711    0.1913    0.7926

>> p=P*c

p =

   -6.3637
   -3.2363
   -3.2659
   -4.5130
   -1.8865
    1.8019
```

1. Use linsolve to find the best solution x to Ox = c. With that x, calculate p, the closest point to c in the span of O. Hint: It's Ox.

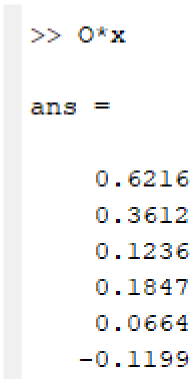


```
>> p=P*c

p =

   -6.3637
   -3.2363
   -3.2659
   -4.5130
   -1.8865
    1.8019
```



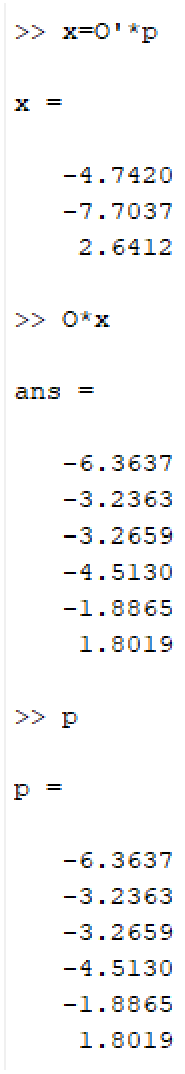
```
>> linsolve(U,p)

ans =

    0.1232
    0.6872
   -0.3125

>> U*linsolve(U,p)

ans =

   -6.3637
   -3.2363
   -3.2659
   -4.5130
   -1.8865
    1.8019
```

←Same as p



```
>> O*x

ans =

    0.6216
    0.3612
    0.1236
    0.1847
    0.0664
   -0.1199
```

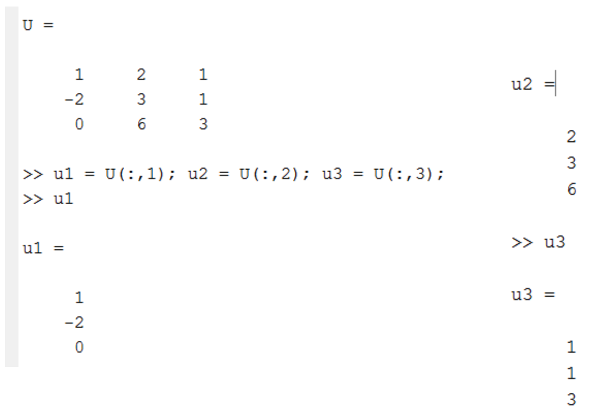
2. Use the methods of Fourier coefficients. Hint: This requires only two matrix multiplications. Hint 2: Page 415 has the formula for the full projector; since O is orthonormal, the formula can be simplified (again, O'*O = I).



```
>> x=O'*p

x =

   -4.7420
   -7.7037
    2.6412

>> O*x

ans =

   -6.3637
   -3.2363
   -3.2659
   -4.5130
   -1.8865
    1.8019

>> p

p =

   -6.3637
   -3.2363
   -3.2659
   -4.5130
   -1.8865
    1.8019
```

The two results for p should be the same.

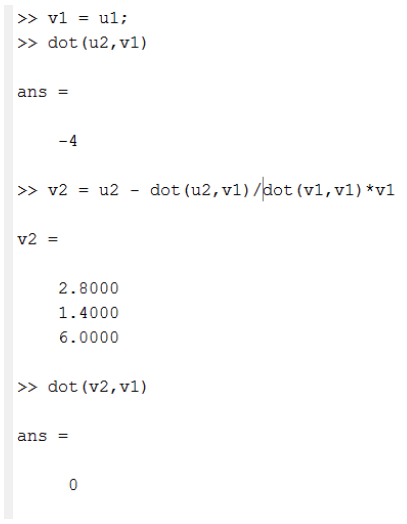
6. (Gram-Schmidt, 10pt) Let U = [1 2 1; -2 3 1; 0 6 3]. Use Matlab for the following, but do not use the qr() function.



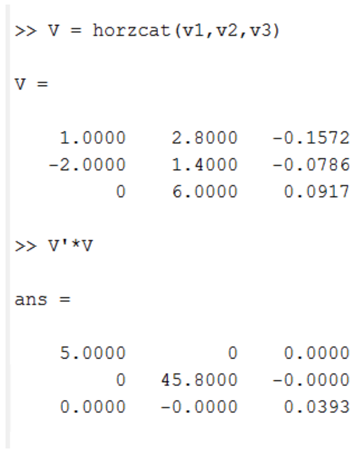
```
U =

     1     2     1
    -2     3     1
     0     6     3

>> u1 = U(:,1); u2 = U(:,2); u3 = U(:,3);
>> u1

u1 =

     1
    -2
     0
```

```
u2 =

     2
     3
     6

>> u3

u3 =

     1
     1
     3
```

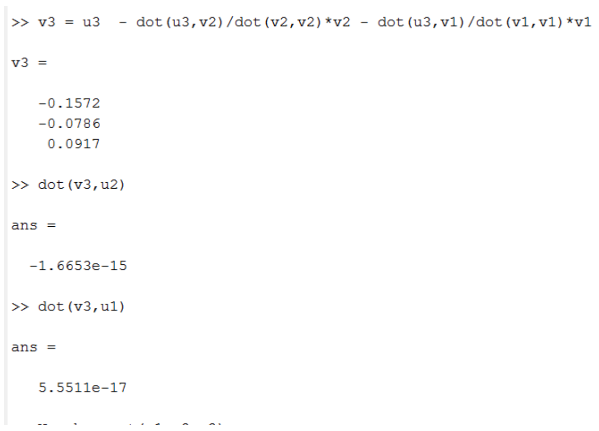
a) [5pt] Perform the Gram-Schmidt orthogonalization to obtain an orthogonal basis of the column-space of U. Start with the three columns u1, u2, u3 of U, and show how to obtain an orthogonal basis V ={v1,v2,v3} (you can follow the slides, or the book, on pages 310ff). Don't normalize the vectors v1, v2, v3 in this step yet.



```
>> v1 = u1;
>> dot(u2,v1)

ans =

    -4

>> v2 = u2 - dot(u2,v1)/dot(v1,v1)*v1

v2 =

    2.8000
    1.4000
    6.0000

>> dot(v2,v1)

ans =

     0
```

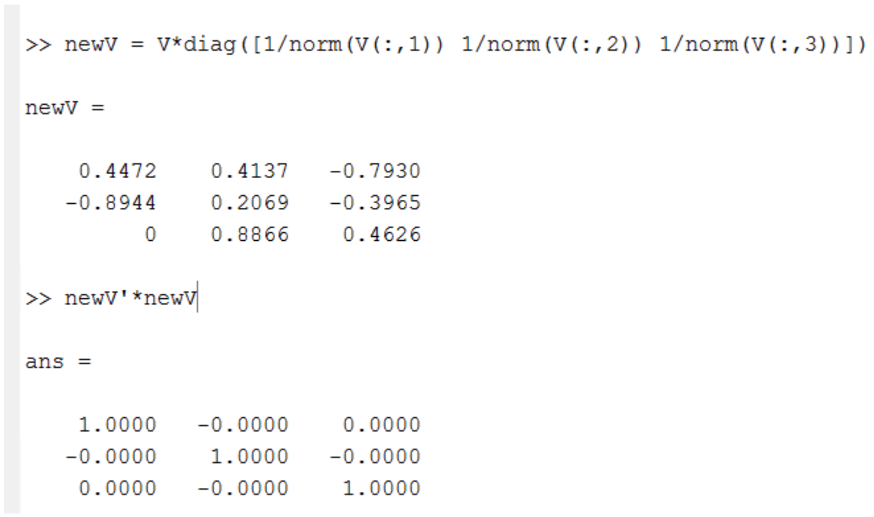


```
>> V = horzcat(v1,v2,v3)

V =

    1.0000    2.8000   -0.1572
   -2.0000    1.4000   -0.0786
         0    6.0000    0.0917

>> V'*V

ans =

    5.0000         0    0.0000
         0   45.8000   -0.0000
    0.0000   -0.0000    0.0393
```



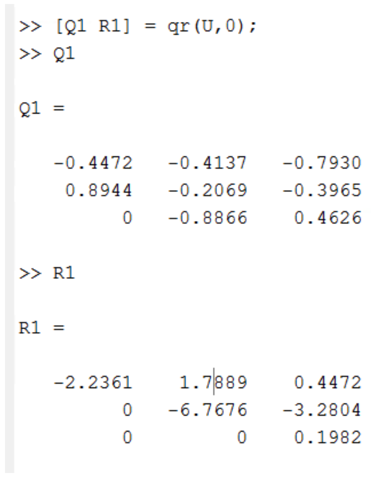
```
>> v3 = u3  - dot(u3,v2)/dot(v2,v2)*v2 - dot(u3,v1)/dot(v1,v1)*v1

v3 =

   -0.1572
   -0.0786
    0.0917

>> dot(v3,u2)

ans =

  -1.6653e-15

>> dot(v3,u1)

ans =

   5.5511e-17
```

b) [2pt] Find an orthonormal basis Q of the span of U, based on your results from a): normalize each of v1, v2, and v3 to obtain orthogonal unit vectors q1, q2, q3 making up Q. Verify that Q is orthonormal.

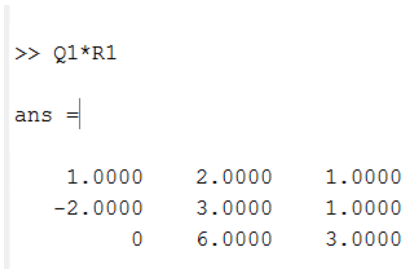


```
>> newV = V*diag([1/norm(V(:,1)) 1/norm(V(:,2)) 1/norm(V(:,3))])

newV =

    0.4472    0.4137   -0.7930
   -0.8944    0.2069   -0.3965
         0    0.8866    0.4626

>> newV'*newV

ans =

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

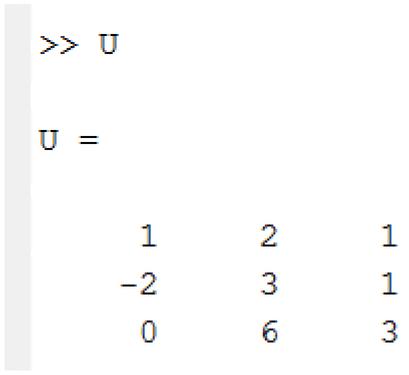
c) [3pt] Find the corresponding (upper triangular) matrix R, so that U = QR. Hint: you can use the formulas on slide 37, I fixed the typo in the formula. Verify that U = QR.



```
>> [Q1 R1] = qr(U,0);
>> Q1

Q1 =

   -0.4472   -0.4137   -0.7930
    0.8944   -0.2069   -0.3965
         0   -0.8866    0.4626

>> R1

R1 =

   -2.2361    1.7889    0.4472
         0   -6.7676   -3.2804
         0         0    0.1982
```



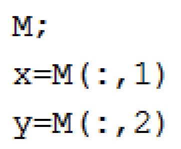
```
>> Q1*R1

ans =

    1.0000    2.0000    1.0000
   -2.0000    3.0000    1.0000
         0    6.0000    3.0000
```



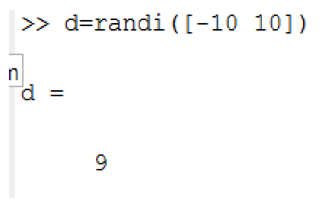
```
>> U

U =

     1     2     1
    -2     3     1
     0     6     3
```

7. (Extra Credit, 7pt) You have collected some data you suspect is of the form y = a sin(x) + bx2 + cx+ d (with some added noise), where y and x are the dependent/independent variable, and a, b, c, and d are the unknown (but fixed) parameters you want to determine. Download, unzip and import hw6-7.zip for the y and x data you observed, it contains a matrix M = (x|y)..

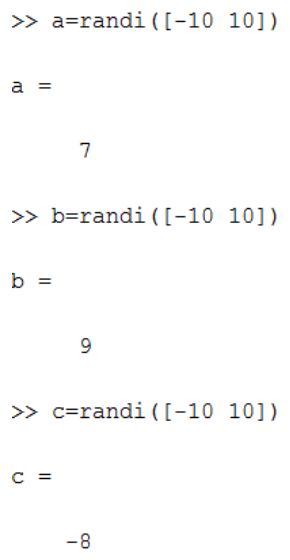
a) [1pt] Scatter plot x, y; does this look like (noisy) quadratic data with a periodic component? Hint: If you're plot looks wrong, you may be using the wrong plotting function.



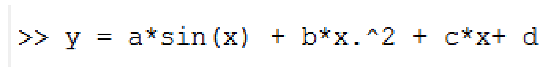
```
M;
x=M(:,1)
y=M(:,2)
```



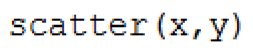
```
>> d=randi([-10 10])
n
d =

     9
```



```
>> a=randi([-10 10])

a =

     7

>> b=randi([-10 10])

b =

     9

>> c=randi([-10 10])

c =

    -8
```

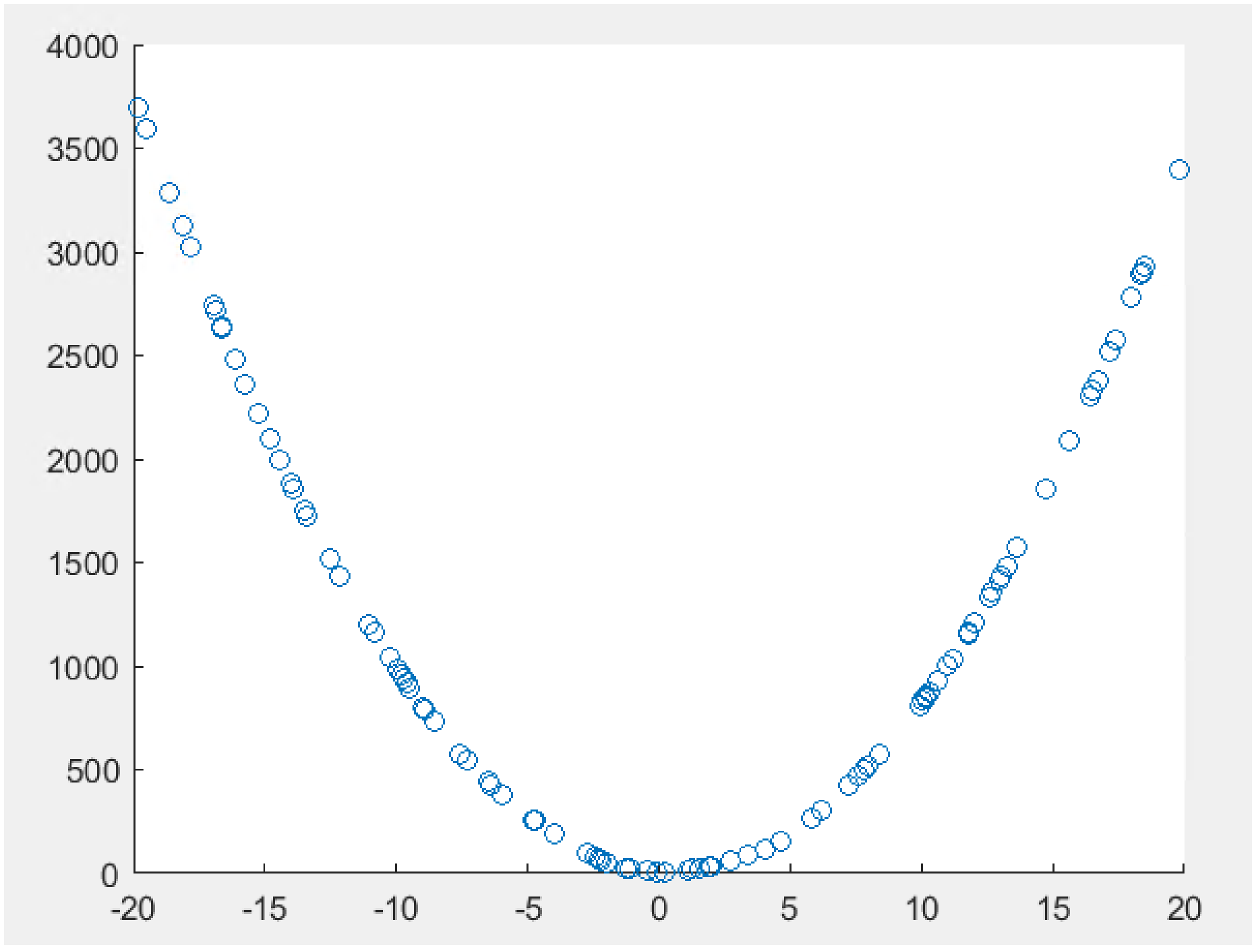


```
>> y = a*sin(x) + b*x.^2 + c*x+ d
```



```
scatter(x,y)
```

Scatter below (next page)



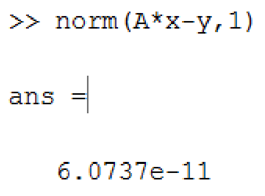
b) [4pt] Use linear regression to find linear-least-squares estimates for a, b, c, and d.

```
>> A=horzcat(x.^2,sin(x),x,ones(100,1))
```

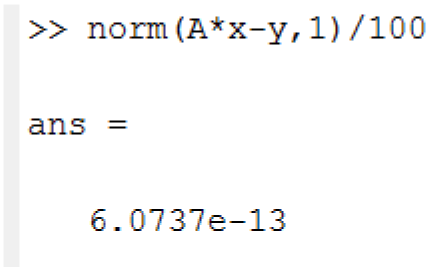
```
>> x=linsolve(A'*A,A'*y)

x =

    9.0000
    7.0000
   -8.0000
    9.0000
```

c.) [2pt] Compare y to the prediction by your model. What's the worst difference? What's the average difference (using the 1-norm)

Both the worst and average difference are close to zero (not too confident in this answer)



```
>> norm(A*x-y,1)

ans =

   6.0737e-11
```



```
>> norm(A*x-y,1)/100

ans =

   6.0737e-13
```