LAB 6: Orthonormal Bases, Orthogonal Projections, and QR factorization

(Math 250: Sections C3)

Name: Aryan Malhotra

Last 4 digits of RUID: 8256

Date: 12/10/2024

Learning Outcomes

- Geometric aspects of vectors - *norm*, *inner product*, and *orthogonal projection* onto a line.
- The *Gram-Schmidt Process* to change an independent set of vectors into an *orthonormal set*, and the associated A = QR matrix factorization.
- The *orthogonal projection* of a vector onto a subspace.

Table of Contents

Name: Aryan Malhotra.....1
Last 4 digits of RUID: 8256.....1
Date: 12/10/2024.....1
Learning Outcomes.....1
Random Seed.....1
Question 1. Norm, Inner Product, and Orthogonal Projection onto a Line (Total Points - 7).....2
Question 2. Gram-Schmidt Process and QR Factorization (Total Points - 10).....4
Question 3. Orthogonal Projection onto a Subspace (Total Points - 4).....10

Random Seed

Initialize the random number generator by typing the following code, where abcd are the last four digits of your RUID

```
rng('default');
rng(abcd, 'twister');
```

```
% Enter your code here
rng('default');
rng(8256, 'twister');
```

This will ensure that you generate your own particular random vectors and matrices.

The lab report that you hand in must be your own work. The following problems use randomly generated matrices and vectors, so the matrices and vectors in your lab report will not be the same as those of other students doing the lab. Sharing of lab report files is not allowed in this course.

Question 1. Norm, Inner Product, and Orthogonal Projection onto a Line (Total Points - 7)

Generate random vectors u, v $\in \mathbb{R}^2$ by ***u = rvect(2)***, ***v = rvect(2)***. Calculate ***rank([u, v])*** to determine whether they are linearly independent (this is also evident by inspection, since the vectors have integer entries). If the answer is not 2, then generate a new random pair of vectors and calculate the rank. Repeat until the rank is 2, and keep the vectors you generate in your lab writeup.

```
% Enter your code here
u = rvect(2); v = rvect(2); rank([u, v])
```

```
ans =
2
```

Now use these vectors in the following calculations.

The inner product of vectors u and v can be denoted by u · v or <u,v> (different notations).

(i) ***Exercise (Points - 2):*** Read the proof of Theorem 17 in Section 6.7 of the textbook. State (in symbols) the *triangle inequality* relating the norms ||u||, ||v|| and ||u+v|| for a general pair of vectors u, v.

Answer: ||u+v|| ≤ ||u|| + ||v||

Then use Matlab to show that your particular vectors u, v satisfy this inequality. Note that ||u|| is calculated by the Matlab command ***norm(u)***.

```
% Enter your code here
norm(u+v) <= norm(u)+norm(v)
```

```
ans = logical
   1
```

It is true(1)

(ii) ***Exercise (Points - 2):*** Read the proof of Theorem 16 in Section 6.7 of the textbook. State (in symbols) the *Cauchy-Schwarz inequality* relating the inner product u·v and the norms ||u||, ||v|| for a general pair of vectors u, v.

Answer: |u·v| ≤ ||u||*||v||

Then use Matlab to show that your particular vectors u, v satisfy this inequality. Note that inner product is calculated in Matlab by ***u'*v*** when u and v are column vectors of the same size. The absolute value | t | of a number t is calculated in Matlab by ***abs(t)***.

```
% Enter your code here
abs(u'*v) <= norm(u)*norm(v)
```

```
ans = logical
   1
```

```
% It is true (1)
```

(iii) ***Exercise (Points - 1):*** The orthogonal projection of a vector u onto the line L (one-dimensional subspace) spanned by the vector v is $w = \text{proj}_v\, u = \dfrac{u \cdot v}{v \cdot v} v$.

Use Matlab to calculate w for your vectors.

```
% Enter your code here
w = ((u'*v)/(v'*v))*v
```

```
w = 2×1
    5.2809
    8.4494
```

Two vectors are orthogonal if their inner product is zero. Verify by Matlab that the vector z = u − w is orthogonal to v. (If the inner product is not exactly zero but is a very small number of size 10^{-13} for example, then the vectors are considered orthogonal for numerical purposes.)

```
% Enter your code here
z = u-w, z'*v
```

```
z = 2×1
    0.7191
   -0.4494
ans =
5.3291e-15
```

(iv) ***Exercise (Points - 2):*** The formula for w in (iii) can also be written as a matrix-vector product:

$$w = \frac{u \cdot v}{v \cdot v} v = v \frac{v \cdot u}{v \cdot v} = v \frac{v^T u}{v \cdot v} = v \frac{1}{v \cdot v} v^T u = \text{Pu}$$

Use Matlab to obtain the matrix ***P = v*inv(v'*v)*v'*** (note carefully the punctuation and the order of the factors in this formula). Calculate by Matlab that Pu is the vector w for your u and v.

```
% Enter your code here
P = v*inv(v'*v)*v', P*u, w
```

```
P = 2×2
    0.2809    0.4494
    0.4494    0.7191
ans = 2×1
    5.2809
    8.4494
```

```
w = 2×1
    5.2809
    8.4494
```

Explain why P is a 2 × 2 matrix.

Answer: 1/v·v is a scalar, meaning the multiplication can be rewritten as (1/ v·v)*v*v^T. v has size 2x1 and v^T has the size 1x2, meaning when the two are multiplied it makes a 2x2 matrix. The scalar does not affect the matrix's size.

Question 2. Gram-Schmidt Process and QR Factorization (Total Points - 10)

Generate three random vectors w1, w2, w3 $\in \mathbb{R}^3$. Check whether they are linearly independent by calculating ***rank([w1, w2, w3])***. If the answer is not 3, then generate a new random set of vectors and calculate the rank. Repeat until the rank is 3, and again, keep all the vectors you generate in your lab writeup.

```
% Enter your code here
w1=rvect(3), w2=rvect(3), w3=rvect(3), rank([w1, w2, w3])
```

```
w1 = 3×1
     3
     6
     7
w2 = 3×1
     7
     7
     6
w3 = 3×1
     0
     6
     6
ans =
3
```

Now use these vectors in the following calculations.

(a) Since the vectors w1, w2, w3 are chosen at random, it is very unlikely that they are mutually orthogonal. To see this graphically using Matlab, generate a line plotting parameter r and open a graphics window by the commands (be careful with the punctuation)

```
r = 0:0.05:1; hold on
```

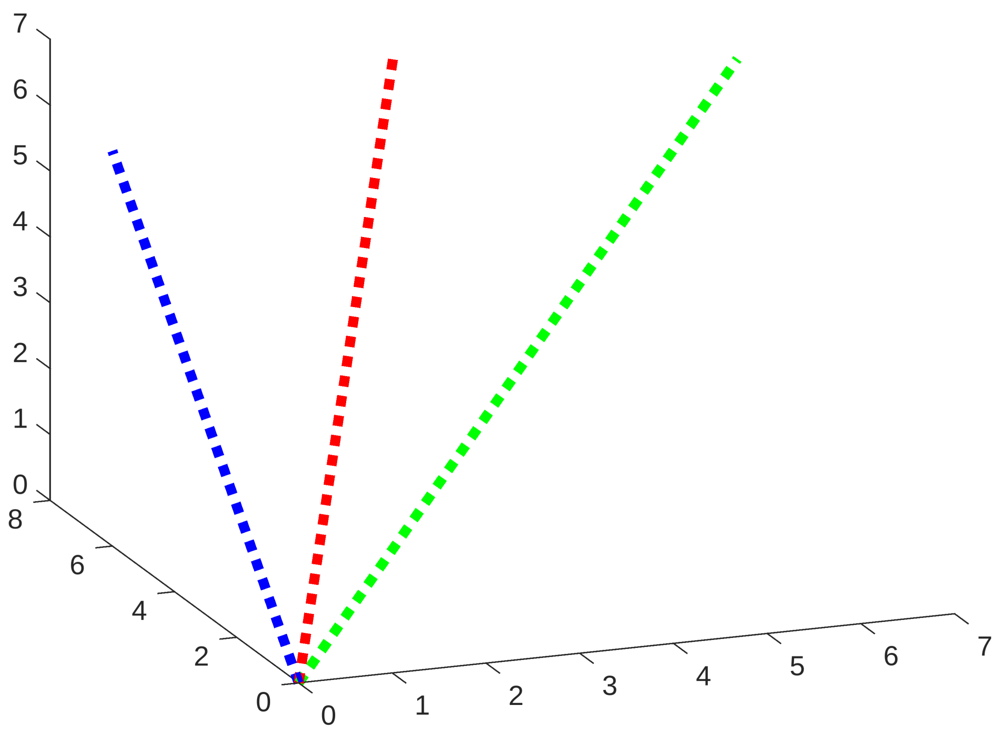
```
% Enter your code here
r = 0:0.05:1; hold on
```

Plot the three vectors in the graphics window as red, green, and blue dotted lines by the commands

```
plot3(r*w1(1),r*w1(2),r*w1(3), 'r:', 'LineWidth', 3.6)
plot3(r*w2(1),r*w2(2),r*w2(3), 'g:', 'LineWidth', 3.6)
plot3(r*w3(1),r*w3(2),r*w3(3), 'b:', 'LineWidth', 3.6)
```

```
% Enter your code here
```

```
plot3(r*w1(1),r*w1(2),r*w1(3), 'r:', 'LineWidth', 3.6)
plot3(r*w2(1),r*w2(2),r*w2(3), 'g:', 'LineWidth', 3.6)
plot3(r*w3(1),r*w3(2),r*w3(3), 'b:', 'LineWidth', 3.6)
```



(i) ***Exercise (Points - 1):*** Using the Rotate 3D command, determine visually whether the vectors are mutually orthogonal or not.

Answer: The vectors are not mutually orthogonal

(b) Now use the vectors w1, w2, w3 to obtain an orthogonal basis for $\mathbb{R}^3$, following the Gram-Schmidt process (see the proof of Theorem 11 in Section 6.4 of the textbook). Set v1 = w1. Define the projection P1 onto the span of v1 as in Question 1(iv), and obtain v2 by removing the component of w2 in the direction v1:

```
P1 = v1*inv(v1'*v1)*v1'; v2 = w2 - P1*w2
```

```
% Enter your code here
v1 = w1, P1 = v1*inv(v1'*v1)*v1', v2 = w2 - P1*w2
```

```
v1 = 3×1
     3
     6
     7
P1 = 3×3
    0.0957    0.1915    0.2234
    0.1915    0.3830    0.4468
```

```
     0.2234     0.4468     0.5213
v2 = 3×1
     3.6489
     0.2979
    -1.8191
```

(i) ***Exercise (Points - 1):*** Calculate the inner product by hand to check that the vectors v1 and v2 are mutually orthogonal (within a negligible numerical error).

Answer:

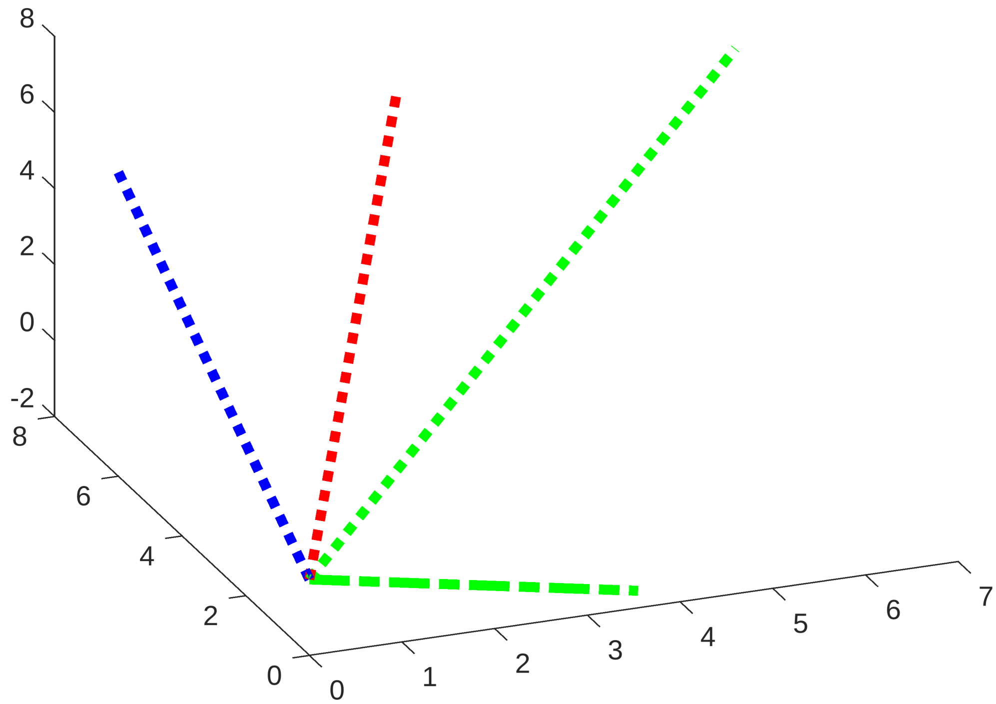
```
v1'*v2
```

```
ans =
8.8818e-15
```

Also add v2 to your graphics window as a dashed-dotted green line by

```
plot3(r*v2(1),r*v2(2),r*v2(3), 'g-.', 'LineWidth', 3.6)
```

```
% Enter your code here
plot3(r*v2(1),r*v2(2),r*v2(3), 'g-.', 'LineWidth', 3.6)
```



Using the Rotate 3D command, rotate the frame to to see that the red line for v1 and the green line for v2 are orthogonal.

Now define P2 as the projection onto the span of v2 and obtain v3 by removing the components of w3 in the directions of v1 and v2:

```
P2 = v2*inv(v2'*v2)*v2', v3 = w3 - P1*w3 - P2*w3
```

```
% Enter your code here
P2 = v2*inv(v2'*v2)*v2', v3 = w3 - P1*w3 - P2*w3
```

```
P2 = 3×3
    0.7967    0.0650   -0.3972
    0.0650    0.0053   -0.0324
   -0.3972   -0.0324    0.1980
v3 = 3×1
   -0.4965
    1.1840
   -0.8020
```

(ii) ***Exercise (Points - 1):*** Calculate inner products by Matlab to check that v3 is orthogonal to the vectors v1 and v2 (within a negligible numerical error).

Answer:

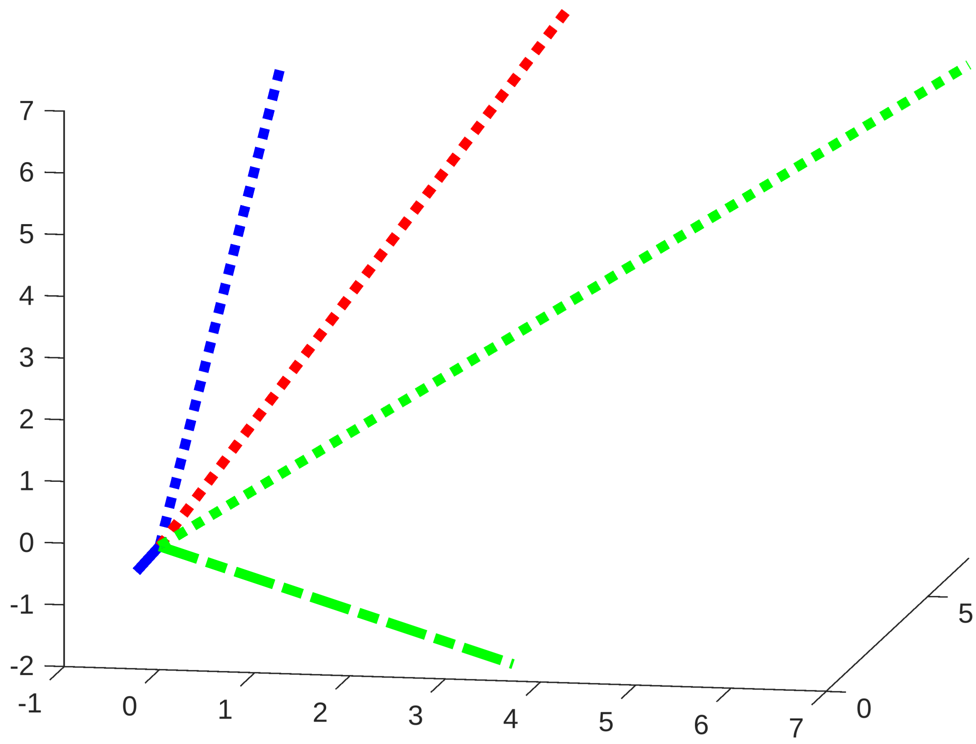
```
% Enter your code here
v1'*v3, v2'*v3
```

```
ans =
7.1054e-15
ans =
-9.3259e-15
```

Add v3 to your plot as a dashed-dotted blue line by

```
plot3(r*v3(1),r*v3(2),r*v3(3), 'b-.', 'LineWidth', 3.6)
```

```
% Enter your code here
plot3(r*v3(1),r*v3(2),r*v3(3), 'b-.', 'LineWidth', 3.6)
```



Using the Rotate 3D command, rotate the frame to see that the red line for v1, the dash-dot green line for v2, and the dash-dot blue line for v3 are mutually orthogonal.

(iii) ***Exercise (Points - 1):*** Obtain a good alignment of the graph that shows orthogonality in perspective and then save your lab report (Include the graph in your lab report).

(c) Now normalize the vectors v1, v2, v3 to obtain an orthonormal basis for $\mathbb{R}^3$ byDefine the matrix Q = [w1, w2, w3] and give written answers to the following questions.

```
u1 = v1/norm(v1), u2 = v2/norm(v2), u3 = v3/norm(v3)
```

```
% Enter your code here
Q = [w1 w2 w3],u1 = v1/norm(v1), u2 = v2/norm(v2), u3 = v3/norm(v3)
```

```
Q = 3×3
     3     7     0
     6     7     6
     7     6     6
u1 = 3×1
    0.3094
    0.6189
    0.7220
u2 = 3×1
    0.8926
    0.0729
```

```
   -0.4450
u3 = 3×1
   -0.3280
    0.7821
   -0.5298
```

Define the matrix ***Q = [u1, u2, u3]*** and give written answers to the following questions.

```
% Enter your code here
Q = [u1, u2, u3]
```

```
Q = 3×3
    0.3094    0.8926   -0.3280
    0.6189    0.0729    0.7821
    0.7220   -0.4450   -0.5298
```

(i) ***Exercise (Points - 1):*** Write a symbolic (not numerical) hand calculation of the entries in the 3 × 3 matrix $Q^T Q$ in terms of the inner products ui · uj . Use this to describe the orthonormal property of {u1, u2, u3} in terms of $Q^T Q$.

Answer:

Q^T * Q = [u1; u2; u3] * [u1 u2 u3] = [u1·u1 u1·u2 u1·u3; u2·u1 u2·u2 u2·u3; u3·u1 u3·u2 u3·u3] = [1 0 0; 0 1 0; 0 0 1] = I If matrix Q has orthonormal columns, Q^T * Q = I

(ii) ***Exercise (Points - 1):*** What is the inverse matrix Q^{-1}?

Answer:

Q^-1 is equal to Q^T

(iii) ***Exercise (Points - 2):*** Now check your answers to questions (i) and (ii) with Matlab calculations..

```
% Enter your code here
Q'*Q, inv(Q), Q'
```

```
ans = 3×3
    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000
ans = 3×3
    0.3094    0.6189    0.7220
    0.8926    0.0729   -0.4450
   -0.3280    0.7821   -0.5298
ans = 3×3
    0.3094    0.6189    0.7220
    0.8926    0.0729   -0.4450
   -0.3280    0.7821   -0.5298
```

(d) The Gram-Schmidt process and normalization give the QR factorization of a matrix. To illustrate this using Matlab, set

```
A = [w1, w2, w3], R = Q'*A
```

```
% Enter your code here
A = [w1, w2, w3], R = Q'*A
```

```
A = 3×3
     3     7     0
     6     7     6
     7     6     6
R = 3×3
    9.6954   10.8299    8.0451
    0.0000    4.0881   -2.2327
    0.0000    0.0000    1.5138
```

(i) ***Exercise (Points - 1):*** Verify by Matlab that A = Q*R.

```
% Enter your code here
Q*R, A
```

```
ans = 3×3
    3.0000    7.0000   -0.0000
    6.0000    7.0000    6.0000
    7.0000    6.0000    6.0000
A = 3×3
     3     7     0
     6     7     6
     7     6     6
```

(ii) ***Exercise (Points - 1):*** Then give a symbolic (not numerical) proof of the fact that R is upper triangular, as follows. Let R have entries rij . Use the property u2 · w1 = 0 to show that r21 = 0. Likewise, use the property u3 · w1 = u3 · w2 = 0 to show that r31 = r32 = 0.

```
% R = Q^T * A = [u1; u2; u3] * [w1 w2 w3] =
% [u1·w1 u1·w2 u1·w3; u2·w1 u2·w2 u2·w3; u3·w1 u3·w2 u3·w3]
% [r11 r12 r13; 0 r22 r23; 0 0 r33]
```

Question 3. Orthogonal Projection onto a Subspace (Total Points - 4)

Generate three random vectors a1, a2, a3 $\in \mathbb{R}^5$ and the matrix A with these vectors as columns:

```
a1 = rvect(5); a2 = rvect(5); a3 = rvect(5); A = [a1, a2, a3]
```

```
% Enter your code here
a1 = rvect(5), a2 = rvect(5), a3 = rvect(5), A = [a1, a2, a3]
```

```
a1 = 5×1
     1
     1
     5
     0
     1
```

```
a2 = 5×1
     3
     9
     2
     3
     1
a3 = 5×1
     4
     4
     1
     0
     3
A = 5×3
     1     3     4
     1     9     4
     5     2     1
     0     3     0
     1     1     3
```

Check whether they are linearly independent by calculating ***rank(A)***. If the answer is not 3, then generate a new random set of vectors and calculate the rank. Repeat until the rank is 3, again keeping all the vectors you generate in your lab writeup. Now use these vectors and matrix in the following.

```
% Enter your code here
rank(A)
```

```
ans =
3
```

(a) Let W = Col(A) be the subspace of $\mathbb{R}^5$ spanned by {a1, a2, a3}. The teaching code ***grams.m*** carries out the steps of the Gram-Schmidt process and normalization, just as you did step-by-step in Question 2. Calculate by Matlab:

```
Q = grams(A); u1 = Q(:,1), u2 = Q(:,2), u3 = Q(:,3)
```

```
% Enter your code here
Q = grams(A), u1 = Q(:,1), u2 = Q(:,2), u3 = Q(:,3)
```

```
Q = 5×3
    0.1890    0.2362    0.6393
    0.1890    0.8865   -0.1066
    0.9449   -0.2284   -0.2310
         0    0.3252   -0.3729
    0.1890    0.0194    0.6225
u1 = 5×1
    0.1890
    0.1890
    0.9449
         0
    0.1890
u2 = 5×1
    0.2362
    0.8865
   -0.2284
    0.3252
    0.0194
u3 = 5×1
    0.6393
```

```
   -0.1066
   -0.2310
   -0.3729
    0.6225
```

(i) ***Exercise (Points - 1):*** Calculate Q^TQ by Matlab and explain why your answer shows that {u1, u2, u3} is an orthonormal set of vectors. (Hint: As in Question 2 (c)(i), relate the inner products to the entries of the matrix.)

```
% Enter your code here
Q'*Q
```

```
ans = 3×3
    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

Answer:

(b) Orthogonal Decomposition v = w + z: The orthogonal projection matrix P from $\mathbb{R}^5$ onto the subspace W is given by the 5 × 5 matrix P=$QQ^T = \mathrm{u1}\ \mathrm{u1}^T + \mathrm{u2}\ \mathrm{u2}^T + \mathrm{u3}\ \mathrm{u3}^T$:

```
P = u1*u1' + u2*u2' + u3*u3'
```

```
% Enter your code here
P = u1*u1' + u2*u2' + u3*u3'
```

```
P = 5×5
    0.5002    0.1769   -0.0231   -0.1616    0.4382
    0.1769    0.8330    0.0007    0.3281   -0.0135
   -0.0231    0.0007    0.9984    0.0119    0.0303
   -0.1616    0.3281    0.0119    0.2448   -0.2259
    0.4382   -0.0135    0.0303   -0.2259    0.4236
```

If v ∈ R^5, then Pv = (u1 · v)u1 + (u2 · v)u2 + (u3 · v)u3.

(i) ***Exercise (Points - 1):*** Generate a random vector ***v = rvect(5)*** and calculate

```
w = P*v, z = v-w
```

```
% Enter your code here
v = rvect(5), w = P*v, z = v-w
```

```
v = 5×1
     2
     9
     8
     8
     0
w = 5×1
    1.1151
   10.4811
    8.0426
    4.6830
   -0.8090
```

```
z = 5×1
    0.8849
   -1.4811
   -0.0426
    3.3170
    0.8090
```

If z = 0 (this is very unlikely) generate another random vector until you get one with z not zero.

(ii) ***Exercise (Points - 1):*** Verify by Matlab that P*w = w and P*z = 0. This shows that w is in the subspace W and that z is the component of v perpendicular to W.

```
% Enter your code here
P*w, w, P*z
```

```
ans = 5×1
    1.1151
   10.4811
    8.0426
    4.6830
   -0.8090
w = 5×1
    1.1151
   10.4811
    8.0426
    4.6830
   -0.8090
ans = 5×1
10^-15 ×
   -0.3331
   -0.8240
    0.1318
   -0.2220
         0
```

(c) The projection matrix P onto the subspace W can be calculated directly from the matrix A, without first orthogonalizing the columns of A, by the formula

PW= $A(A^TA)A^T$. Use Matlab to obtain the matrix

```
PW = A*inv(A'*A)*A'
```

```
% Enter your code here
PW = A*inv(A'*A)*A'
```

```
PW = 5×5
    0.5002    0.1769   -0.0231   -0.1616    0.4382
    0.1769    0.8330    0.0007    0.3281   -0.0135
   -0.0231    0.0007    0.9984    0.0119    0.0303
   -0.1616    0.3281    0.0119    0.2448   -0.2259
    0.4382   -0.0135    0.0303   -0.2259    0.4236
```

(i) ***Exercise (Points - 1):*** Check by Matlab that norm(PW - P) is zero (up to negligible numerical error).

```
norm(PW - P)
```

```
% Enter your code here
norm(PW - P)
```

```
ans =
6.8966e-16
```

Helper Function (Do Not Edit)

```
function v = rvect(m)
v = fix(10*rand(m,1));
end

function A = rmat(m,n)
A = fix(10*rand(m,n));
end
```

```
function dot2dot ( X )
  X(:,end+1) = X(:,1);
  plot ( X(1,:), X(2,:), '.-', 'markersize', 18, 'linewidth', 2 );
  grid ( 'on' );
  axis ( 10 * [ -1, 1, -1, 1 ] );
  axis ( 'square' );
  return
end
```