Table of Contents


```
%%%% MATLAB PROJECT 1, MATH461, LINEAR ALGEBRA FOR SCIENTIST AND ENGINEERS
%%%% Romeo Perlstein, 3/6/2023
%%%% UID: 118030685, section 0123

format short
```

Question 1

a)

```
thetaA = pi()/9
A = [cos(thetaA) -sin(thetaA) ; sin(thetaA) cos(thetaA)]

v = [1;3]

v_rotated = A*v
```

```
thetaB = pi()/10
```

b)

```
B = [cos(thetaB) -sin(thetaB) ; sin(thetaB) cos(thetaB)]

ex_1 = A*B
ex_2 = B*A
if (A*B == B*A)
    fprintf("AB does in fact equal BA \n")
else
    fprintf("AB does NOT equal BA \n")
end
```

c)

no, it doesn't matter which rotation is applied first, because you are simply rotating the vector a by a certain angle first, and then a second certain angle second, so it does not matter which orders the rotation is applied in because the total rotation remains the same. This is proven in b) because as we learned, the output of a function that applies to matricies to a vector is equal to a fucntion with a standard matrix that is the two seperate matricies multiplied together, and in b) we show that the order of multiplication of the matricies does not their result

d)

```
format rat
C = A*B
theta3 = acos(C(1,1))
actual_theta3 = theta3/pi()
```

e)

```
format short
theta_inv = -pi()/9
R_inv_manualls = [cos(theta_inv) -sin(theta_inv) ; sin(theta_inv)
 cos(theta_inv)]

R_inv_check = inv(A)

R_checker_1 = det(R_inv_check)
R_checker_2 = det(R_inv_manualls)

if (rat(R_checker_2) == rat(R_checker_1))
    fprintf("They are equal \n")
else
    fprintf("they are NOT equal \n")
end
```

f)

redefine R so that we can use it better

```
R = A % A already uses theta = pi/9
R_inv = inv(R)
L_naught = [1 0 ; 0 -1]
L_theta = R*L_naught*R_inv
```

g)

```
L_theta_L_naught = L_theta*L_naught
L_naught_L_theta = L_naught*L_theta

if (rat(L_theta_L_naught) == rat(L_naught_L_theta))
    fprintf("They are equal \n")
else
    fprintf("They are NOT equal \n")
end
```

h)

```
format rat
theta4 = acos(L_theta_L_naught(1,1))
actual_theta4 = theta4/pi
```

```
thetaA =

    0.3491


A =

    0.9397   -0.3420
    0.3420    0.9397


v =

     1
     3


v_rotated =

   -0.0864
    3.1611


thetaB =

    0.3142


B =
```

```
    0.9511   -0.3090
    0.3090    0.9511


ex_1 =

    0.7880   -0.6157
    0.6157    0.7880


ex_2 =

    0.7880   -0.6157
    0.6157    0.7880

AB does in fact equal BA

C =

    1446/1835      -684/1111
     684/1111      1446/1835


theta3 =

    1349/2034


actual_theta3 =

      19/90


theta_inv =

   -0.3491


R_inv_manualls =

    0.9397    0.3420
   -0.3420    0.9397


R_inv_check =

    0.9397    0.3420
   -0.3420    0.9397


R_checker_1 =

    1.0000
```

```
R_checker_2 =

     1

They are equal

R =

    0.9397   -0.3420
    0.3420    0.9397


R_inv =

    0.9397    0.3420
   -0.3420    0.9397


L_naught =

     1     0
     0    -1


L_theta =

    0.7660    0.6428
    0.6428   -0.7660


L_theta_L_naught =

    0.7660   -0.6428
    0.6428    0.7660


L_naught_L_theta =

    0.7660    0.6428
   -0.6428    0.7660

They are NOT equal

theta4 =

     710/1017


actual_theta4 =

       2/9
```

Question 2

a)

```
format rat
A_2 = [8 9 3 ; 9 6 5 ; 2 1 9]

A_2_aug = [A_2 eye(3)]

A_2_aug_rref = rref(A_2_aug)

A_2_inv = A_2_aug_rref(:, 4:6)
```

b)

```
A_2_inv_check = inv(A_2)

if(rat(A_2_inv_check) == rat(A_2_inv))
    fprintf("Matching inverses! \n")
else
    fprintf("non-Matching inverses :( \n")
end
```

```
A_2 =

       8             9             3
       9             6             5
       2             1             9


A_2_aug =

  Columns 1 through 5

       8             9             3             1             0
       9             6             5             0             1
       2             1             9             0             0

  Column 6

       0
       0
       1


A_2_aug_rref =

  Columns 1 through 5

       1             0             0          -49/256        39/128
       0             1             0           71/256       -33/128
```

```
       0              0              1             3/256         -5/128

  Column 6

     -27/256
      13/256
      33/256


A_2_inv =

     -49/256         39/128        -27/256
      71/256        -33/128         13/256
       3/256         -5/128         33/256


A_2_inv_check =

     -49/256         39/128        -27/256
      71/256        -33/128         13/256
       3/256         -5/128         33/256

Matching inverses!
```

Question 3

a)

```
format rat
A_3 = [2 0 0 0 ; -7 1 0 0 ; 1 11 -2 0 ; -4 9 3 5]
B_3 = [0 1 2 -1 ; 2 1 0 -1 ; 2 2 2 1 ; -1 2 2 3]

det_A_3 = det(A_3)
det_B_3 = det(B_3)
```

b)

The general fact that would've allowed us to easily compute the determinant of A would be that, since it is a triangular matrix (specifically a lower triangular matrix), the determinant of the matrix is the product of its elements along the diagonal!

c)

```
C_3 = A_3*B_3
det_C_3 = det(C_3)
```

d)

The general fact that could have been used to computer det(C) without matlab is the fact that the det(C) = det(A)det(B) or, the det(AB).

```
A_3 =

       2             0             0             0
      -7             1             0             0
       1            11            -2             0
      -4             9             3             5


B_3 =

       0             1             2            -1
       2             1             0            -1
       2             2             2             1
      -1             2             2             3


det_A_3 =

     -20


det_B_3 =

     -18


C_3 =

       0             2             4            -2
       2            -6           -14             6
      18             8            -2           -14
      19            21             8            13


det_C_3 =

     360
```

Question 4)

```
A_4 = [0 -1 3 4 ; 2 8 3 7 ; 5 6 2 6 ; 6 3 4 5]
```

a)

```
det_A_4 = det(A_4)
```

b)

det(B) = -det(A) = -320 det(C) = -1/2 * det(A) = -640 det(D) = det(A) = 320

c)

```
B_4 = [A_4(3,:) ; A_4(2,:) ; A_4(1,:) ; A_4(4,:)]
C_4 = [A_4(1,:) ; A_4(2,:) ; -2*A_4(3,:) ; A_4(4,:)]
D_4 = [(A_4(1,:) + 8*A_4(3,:)) ; A_4(2,:) ; A_4(3,:) ; A_4(4,:)]
```

d)

```
format short
det_B_4 = int32(det(B_4))
det_C_4 = int32(det(C_4))
det_D_4 = int32(det(D_4))

if(det_B_4 == -320.0000 && det_C_4 == -640.0000 && det_D_4 == 320.0000)
    fprintf("The original calculated values equal Matlabs \ncalculated values!
 \n")
else
    fprintf("The answers from c and d are not equivelent, though they should
 be! \n")
end
```

```
A_4 =

       0            -1             3             4
       2             8             3             7
       5             6             2             6
       6             3             4             5


det_A_4 =

     320


B_4 =

       5             6             2             6
       2             8             3             7
       0            -1             3             4
       6             3             4             5


C_4 =

       0            -1             3             4
       2             8             3             7
     -10           -12            -4           -12
       6             3             4             5


D_4 =
```

```
    40    47    19    52
     2     8     3     7
     5     6     2     6
     6     3     4     5
```

```
det_B_4 =

  int32

   -320
```

```
det_C_4 =

  int32

   -640
```

```
det_D_4 =

  int32

   320
```

The original calculated values equal Matlabs calculated values!

Question 5

a)

```
syms a b c d
A_5 = [a b ; c d]
```

b)

```
A_5_inv = inv(A_5)
```

c)

```
syms e f g h i

B_5 = [a b c ; d e f ; g h i]
B_5_inv = inv(B_5)
```

d)

```
B_5_inv_final = B_5_inv * det(B_5)
```

```
%%%% END OF MATLAB 2 (poggers) %%%%
```

```
A_5 =

[a, b]
[c, d]
```

```
A_5_inv =

[ d/(a*d - b*c), -b/(a*d - b*c)]
[-c/(a*d - b*c),  a/(a*d - b*c)]
```

```
B_5 =

[a, b, c]
[d, e, f]
[g, h, i]
```

```
B_5_inv =

[ (e*i - f*h)/(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g), -(b*i - c*h)/
(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g),  (b*f - c*e)/(a*e*i - a*f*h -
 b*d*i + b*f*g + c*d*h - c*e*g)]
[-(d*i - f*g)/(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g),  (a*i - c*g)/
(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g), -(a*f - c*d)/(a*e*i - a*f*h -
 b*d*i + b*f*g + c*d*h - c*e*g)]
[ (d*h - e*g)/(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g), -(a*h - b*g)/
(a*e*i - a*f*h - b*d*i + b*f*g + c*d*h - c*e*g),  (a*e - b*d)/(a*e*i - a*f*h -
 b*d*i + b*f*g + c*d*h - c*e*g)]
```

```
B_5_inv_final =

[e*i - f*h, c*h - b*i, b*f - c*e]
[f*g - d*i, a*i - c*g, c*d - a*f]
[d*h - e*g, b*g - a*h, a*e - b*d]
```

Published with MATLAB® R2022b