LAFDS Session 3,4&5 Homework

Name: Rahma Farag Yehia

Group: 2

Question1:

Question 1

$$\begin{bmatrix} m & 1 & 0 \\ 4 & -1 & -1 \\ 0 & 8 & m \end{bmatrix}$$

For vectors to be linearly dependent
Det $= 0$

$$\det = m(-1 \times m + 8) - 4(m)$$

$$= -m^2 + 8m - 4m = -m^2 + 4m = 0$$

$$m^2 - 4m = 0$$

$$m(m-4) = 0$$

$$\boxed{m = 0 \quad \text{or} \quad m = 4}$$

Question2:

Question 2

$$\begin{bmatrix} 1 & 5 & 0 \\ 1 & 5 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix} \xrightarrow[r_3 + r_1]{r_2 - r_1} \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 5 & -1 \end{bmatrix}$$

$$\xrightarrow{r_4 * 1/5} \begin{bmatrix} 1 & 5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & -1/5 \end{bmatrix} \xrightarrow[r_3 * -1]{r_1 - 5r_4} \begin{bmatrix} 1 & 0 & 5 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & -1/5 \end{bmatrix}$$

$$\xrightarrow[r_1 - 5r_3]{r_4 + r_3/5} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

linearly independent

Question3:

Question 3

$$B = \overset{\text{Transformation matrix}}{A}x$$

$$\underset{2\times2}{\begin{bmatrix} 1 & 3 \\ 1 & 5 \end{bmatrix}} = \underset{2\times2}{\begin{bmatrix} a & c \\ b & d \end{bmatrix}} \underset{2\times2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$$

$$\text{Transformation Matric} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 5 \end{bmatrix}$$

$$\text{Inverse of Transformation Matrix} = \frac{1}{5-3} \begin{bmatrix} 5 & -3 \\ -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5/2 & -3/2 \\ -1/2 & 1/2 \end{bmatrix}$$

Question4:

Question 4 $\quad Ax = B$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} -1 & -7 \\ 4 & 0 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} -1 & -7 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 3 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} -1 & -7 \\ 4 & 0 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix} \times \frac{1}{4}$$

$$= \frac{1}{4} \begin{bmatrix} 4 & -8 \\ 12 & 4 \end{bmatrix} = \begin{bmatrix} 1 & -2 \\ 3 & 1 \end{bmatrix}$$

Trans Formation Matrix

Question5:

Question 5

1- $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 3 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ Not in row echelon form

2- $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ reduced row echelon form

3- $\begin{bmatrix} 2 & 6 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ Not in row echelon form

4- $\begin{bmatrix} 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$ Not in row echelon form

5- $\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ row echelon form

6- $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ ← not in row echelon form

Question6:

a-

Question 6

a)

$$\left[\begin{array}{ccc|c|c} 1 & 4 & -1 & 1 & 3 \\ 1 & 5 & 0 & 2 & 2 \\ 0 & 3 & 3 & 3 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|c|c} 1 & 4 & -1 & 1 & 3 \\ 0 & 1 & 1 & 1 & -1 \\ 0 & 3 & 3 & 3 & 1 \end{array}\right] \quad r_2 - r_1$$

$$\left[\begin{array}{ccc|c|c} 1 & 0 & -5 & -3 & 7 \\ 0 & 1 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 4 \end{array}\right] \quad \begin{array}{l} r_1 - 4r_2 \\ r_3 - 3r_2 \end{array}$$

→ For the Vector $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ it is in the span of Columns of A

→ The Columns of A are linearly dependent

→ For The Vector $\begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ it is not in the span of Columns of A

b-

Question 6

b) From part a

$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ Can be represented as

$$-3\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 4 \\ 5 \\ 3 \end{bmatrix}$$

The two basis From part a

$\rightarrow \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$ it is not in the span
$\therefore$ it Can't be represented as linear
Combination of Columns of A

Question7:

a-

Question 7

a) $$\left[\begin{array}{ccc|c} 2 & 0 & -6 & -8 \\ 0 & 1 & 2 & 3 \\ 3 & 6 & -2 & -4 \end{array}\right] \xrightarrow{r1/2} \left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 2 & 3 \\ 3 & 6 & -2 & -4 \end{array}\right]$$

$$r3-3r1 \quad \left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 2 & 3 \\ 0 & 6 & 7 & 8 \end{array}\right] \xrightarrow{r3-6r2} \left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & -5 & -10 \end{array}\right]$$

$$r3/-5 \quad \left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \end{array}\right] \xrightarrow[r1+3r3]{r2-2r3} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

$x_1 = 2$, $x_2 = -1$, $x_3 = 2$

b-

b)

$$\left[\begin{array}{ccc|c} 0 & 1 & 5 & -4 \\ 1 & 4 & 3 & -2 \\ 2 & 7 & 1 & -2 \end{array}\right] \longrightarrow \left[\begin{array}{ccc|c} 1 & 4 & 3 & -2 \\ 0 & 1 & 5 & -4 \\ 2 & 7 & 1 & -2 \end{array}\right]$$

$$r_3-2r_1 \quad \left[\begin{array}{ccc|c} 1 & 4 & 3 & -2 \\ 0 & 1 & 5 & -4 \\ 0 & -1 & -5 & 2 \end{array}\right] \xrightarrow[r_3+r_2]{r_1-4r_2} \left[\begin{array}{ccc|c} 1 & 0 & -17 & 14 \\ 0 & 1 & 5 & -4 \\ 0 & 0 & 0 & 2 \end{array}\right]$$

$\boxed{\text{No Solution}}$

Question8:

Question 8

$$\begin{bmatrix} 2 & -1 & 3 \\ 1 & 0 & 1 \\ 0 & 2 & -1 \\ 1 & 1 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 2 & -1 & 3 \\ 0 & 2 & -1 \\ 1 & 1 & 4 \end{bmatrix}$$

$r_2 - 2r_1$
$r_3 - r_1$

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & -3 & -1 \\ 0 & 2 & -1 \\ 0 & 1 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 2 & -1 \\ 0 & -3 & -1 \end{bmatrix}$$

$r_3 - 2r_2$
$r_4 + 3r_2$

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & -7 \\ 0 & 0 & 8 \end{bmatrix} \xrightarrow{r_3/-7} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 8 \end{bmatrix}$$

$r_1 - r_3$
$r_2 - 3r_3$
$r_4 - 8r_3$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

The Dimension is R^3 (3×3)

Basis are $[2 \ -1 \ 3]$

$[1 \ 0 \ 1]$

$[0 \ 2 \ -1]$

Question9:

Question 9

$$\left[\begin{array}{ccc|c} 2 & 3 & 3 & 1 \\ 0 & -4 & -5 & 2 \\ 6 & 3 & 0 & 3 \\ 1 & 1 & 3 & b \end{array}\right] \longrightarrow \left[\begin{array}{ccc|c} 1 & 1 & 3 & b \\ 0 & -4 & -5 & 2 \\ 6 & 3 & 0 & 3 \\ 2 & 3 & 3 & 1 \end{array}\right]$$

$$\xrightarrow[r2-2r1]{r3-6r1} \left[\begin{array}{ccc|c} 1 & 1 & 3 & b \\ 0 & -4 & -5 & 2 \\ 0 & -3 & -18 & 3-6b \\ 0 & 1 & -3 & 1-2b \end{array}\right] \longrightarrow \left[\begin{array}{ccc|c} 1 & 1 & 3 & b \\ 0 & 1 & -3 & 1-2b \\ 0 & -3 & -18 & 3-6b \\ 0 & -4 & -5 & 2 \end{array}\right]$$

$$\xrightarrow{r1-r2} \begin{array}{l} r3+3r2 \\ r4+4r2 \end{array} \left[\begin{array}{ccc|c} 1 & 0 & 6 & 3b-1 \\ 0 & 1 & -3 & 1-2b \\ 0 & 0 & -27 & 6-12b \\ 0 & 0 & -17 & 6-8b \end{array}\right] \qquad 3-6b+3(1-2b)$$

$$r3/-27 \quad \left[\begin{array}{ccc|c} 1 & 0 & 6 & 3b-1 \\ 0 & 1 & -3 & 1-2b \\ 0 & 0 & 1 & (6-12b)/-27 \\ 0 & 0 & -17 & 6-8b \end{array}\right]$$

$$\begin{array}{l} r4+17r3 \\ r2+3r3 \\ r1-6r3 \end{array} \left[\begin{array}{ccc|c} 1 & 0 & 0 & (3b-1)+6/27\,(6-12b) \\ 0 & 1 & 0 & (1-2b)-\frac{3}{27}(6-12b) \\ 0 & 0 & 1 & (6-12b)/-27 \\ 0 & 0 & 0 & (6-8b)-\frac{17}{27}(6-12b) \end{array}\right]$$

$$(6-8b)-\frac{17}{27}(6-12b)=0$$

$$\frac{20}{9}-\frac{4}{9}b=0 \rightarrow 20-4b=0$$

$$\boxed{b=5}$$

Question10:

Question 10

$$\det(A - I\lambda) = 0 \quad \begin{vmatrix} -2-\lambda & 0 & 1 \\ -5 & 3-\lambda & a \\ 4 & -2 & -1-\lambda \end{vmatrix} = 0$$

at $\lambda = 0$

$$\begin{bmatrix} -2 & 0 & 1 \\ -5 & 3 & a \\ 4 & -2 & -1 \end{bmatrix} = 0$$

$$-2(-3 + 2a) + 1(10 - 12) = 0$$

$$6 - 4a - 2 = 0$$

$$4a = 4 \quad \boxed{a = 1}$$

at $\lambda = 3$

$$\begin{bmatrix} -5 & 0 & 1 \\ -5 & 0 & a \\ 4 & -2 & -4 \end{bmatrix} = 0$$

$$-5(2a) + 10 = 0$$

$$-10a + 10 = 0 \quad \boxed{a = 1}$$

at $\lambda = -3$

$$\begin{bmatrix} 1 & 0 & 1 \\ -5 & 6 & a \\ 4 & -2 & 2 \end{bmatrix} = 0$$

$$(12 + 2a) + (10 - 24) = 0$$

$$12 + 2a - 14 = 0$$

$$2a = 2 \quad \boxed{a = 1}$$

$$\boxed{a = 1}$$

Question11:

Question 11

$$\begin{bmatrix} 1 & -6 \\ 2 & -6 \end{bmatrix}^K$$

① Find eigen values of $\begin{bmatrix} 1 & -6 \\ 2 & -6 \end{bmatrix}$

$|A-I\lambda| = 0 \qquad \begin{vmatrix} 1-\lambda & -6 \\ 2 & -6-\lambda \end{vmatrix} = 0$

$(1-\lambda)(-6-\lambda) + 12 = 0$

$-6 - \lambda + 6\lambda + \lambda^2 + 12 = 0$

$\lambda^2 + 5\lambda + 6 = 0 \qquad \boxed{\lambda_1 = -2}, \boxed{\lambda_2 = -3}$

② For $A^K \longrightarrow \lambda_1 = -2^K \quad \lambda_2 = -3^K$

③ eigen Vectors For A are eigen vectors For A^K

$$\begin{bmatrix} 1-\lambda & -6 \\ 2 & -6-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

at $\lambda_1 = -2$

$$\begin{bmatrix} 3 & -6 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$3x_1 - 6x_2 = 0 \longrightarrow x_1 - 2x_2 = 0$

1st eigen vector $= x_2 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

For $\lambda = -3$

$$\begin{bmatrix} 1-\lambda & -6 \\ 2 & -6-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} 4 & -6 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$2x_1 - 3x_2 = 0$

$2x_1 = 3x_2$

2nd eigen Vector $= x_2 \begin{bmatrix} 3/2 \\ 1 \end{bmatrix}$

$\rightarrow$ Show that the eigen vectors Form basis For R^2

$$\left[\begin{array}{cc|c} 2 & 3/2 & 0 \\ 1 & 1 & 0 \end{array}\right] \xrightarrow{r_1/2} \left[\begin{array}{cc|c} 1 & 3/4 & 0 \\ 1 & 1 & 0 \end{array}\right]$$

$$\xrightarrow{r_2 - r_1} \left[\begin{array}{cc|c} 1 & 3/4 & 0 \\ 0 & 1/4 & 0 \end{array}\right] \xrightarrow{r_2 \times 4} \left[\begin{array}{cc|c} 1 & 3/4 & 0 \\ 0 & 1 & 0 \end{array}\right]$$

$$\xrightarrow{r_1 - 3/4 r_2} \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right]$$

They are linearly indep.

$$A^k = CD^kC^{-1}$$

$$C = \begin{bmatrix} 2 & 3/2 \\ 1 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} -2^k & 0 \\ 0 & -3^k \end{bmatrix}$$

$$C^{-1} = \begin{bmatrix} 1 & -3/2 \\ -1 & 2 \end{bmatrix} * \frac{1}{2 - 3/2}$$

$$C^{-1} = \begin{bmatrix} 2 & -3 \\ -2 & 4 \end{bmatrix}$$

$$A^k = \begin{bmatrix} 2 & 3/2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -2^k & 0 \\ 0 & -3^k \end{bmatrix} \begin{bmatrix} 2 & -3 \\ -2 & 4 \end{bmatrix}$$

Practice with code

Question1:

```
import numpy as np
import pandas as pd
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
B = np.array([[1,2],[3,4],[5,6]])
print("A: ")
print(A)
print("B:")
print(B)

def isSqu(mat):
    if mat.shape[0]==mat.shape[1]:
        return "Square Matrix"
    else:
        return "NOT Square Matrix"

print("A: " ,isSqu(A))
print("B: " ,isSqu(B))
```

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
A = np.array([[1,2,3],[4,5,6],[7,8,9]])
B = np.array([[1,2],[3,4],[5,6]])
print("A: ")
print(A)
print("B:")
print(B)
```

```
A:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
B:
[[1 2]
 [3 4]
 [5 6]]
```

In [3]:
```
def isSqu(mat):
    if mat.shape[0]==mat.shape[1]:
        return "Square Matrix"
    else:
        return "NOT Square Matrix"
```

In [4]:
```
print("A: " ,isSqu(A))
print("B: " ,isSqu(B))
```

```
A:  Square Matrix
B:  NOT Square Matrix
```

Question2:

```
def props(mat):
    #use previous isSqu function
    sq = isSqu(mat)
    print(sq) #print is square or not

    rank = np.linalg.matrix_rank(mat) #calculate the rank
    print("The rank of the matrix is ",rank)

    #calculate the determinate if it is a square matrix
    if(isSqu(mat) == "Square Matrix"):
        print("The determinant of the matrix is ",np.linalg.det(mat))
    else:
        print("Not square matrix can't calculate determinant")

    #check if rank = number of rows
    if rank == mat.shape[0]:
        return 1
    else:
        return 0

print(A)
print(props(A))
print(B)
print(props(B))
```

In [10]:
```
def props(mat):
    #use previous isSqu function
    sq = isSqu(mat)
    print(sq) #print is square or not

    rank = np.linalg.matrix_rank(mat) #calculate the rank
    print("The rank of the matrix is ",rank)

    #calculate the determinate if it is a square matrix
    if(isSqu(mat) == "Square Matrix"):
        print("The determinant of the matrix is ",np.linalg.det(mat))
    else:
        print("Not square matrix can't calculate determinant")

    #check if rank = number of rows
    if rank == mat.shape[0]:
        return 1
    else:
        return 0
```

In [13]:
```
print(A)
props(A)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Square Matrix
The rank of the matrix is  2
The determinant of the matrix is  -9.51619735392994e-16
```

Out[13]: 0

In [14]:
```
print(B)
props(B)
```

```
[[1 2]
 [3 4]
 [5 6]]
NOT Square Matrix
The rank of the matrix is  2
Not square matrix can't calculate determinant
```

Out[14]: 0

Question3:

System1:

"""let sunflowers -> S , Roses -> R , Daises -> D

1.50 S + 5.75 R + 2.6 D = 589.5
S + R + D = 200
-R + D = 20

****

```
A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')
X = np.array([['S'], ['R'], ['D']])
B = np.array([[589.5], [200], [20]], dtype='float64')

#solve Ax=B to get x
x = np.linalg.solve(A, B)
print(x)
x.dtype

print("A:")
print(A)
print(props(A))
print("******")
print("B:")
print(B)
print(props(B))
print("******")
print("x:")
print(x)
print(props(x))
print("******")
```

```
In [10]: A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')
         X = np.array([['S'], ['R'], ['D']])
         B = np.array([[589.5], [200], [20]], dtype='float64')

In [11]: #solve Ax=B to get x
         x = np.linalg.solve(A, B)
         x.dtype

Out[11]: dtype('float64')
```

In [12]:
```
print("A:")
print(A)
print(props(A))
print("******")
print("B:")
print(B)
print(props(B))
print("******")
print("x:")
print(x)
print(props(x))
print("******")
```

```
A:
[[ 1.5   5.75  2.6 ]
 [ 1.    1.    1.  ]
 [ 0.   -1.    1.  ]]
Square Matrix
The rank of the matrix is  3
The determinant of the matrix is  -5.35
1
******
B:
[[589.5]
 [200. ]
 [ 20. ]]
NOT Square Matrix
The rank of the matrix is  1
Not square matrix can't calculate determinant
0
******
x:
[[80.]
 [50.]
 [70.]]
NOT Square Matrix
The rank of the matrix is  1
Not square matrix can't calculate determinant
0
******
```

System2:

"""let potatoes -> p , chicken -> c, oil-> o
#sold is positive , buy is negative
2p + 1c -3o = 0 #zero profit
4p + 2c -6o = 0
1p - 1c +1o = 0"""

A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
X = np.array([['p'],['c'],['o']])
B = np.array([[0],[0],[0]], dtype="float64")

#solve Ax=B to get x
x = np.linalg.solve(A, B)
x.dtype

print("A:")
print(A)
print(props(A))
print("****")**
print("B:")
print(B)
print(props(B))

```
In [38]: A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
         X = np.array([['p'],['c'],['o']])
         B = np.array([[0],[0],[0]], dtype="float64")
```

```
In [39]: #solve Ax=B to get x
         x = np.linalg.solve(A, B)
         x.dtype

---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-39-cc46d9e35350> in <module>
      1 #solve Ax=B to get x
----> 2 x = np.linalg.solve(A, B)
      3 x.dtype

<__array_function__ internals> in solve(*args, **kwargs)

~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in solve(a, b)
    391     signature = 'DD->D' if isComplexType(t) else 'dd->d'
    392     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 393     r = gufunc(a, b, signature=signature, extobj=extobj)
    394 
    395     return wrap(r.astype(result_t, copy=False))

~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_singular(err, flag)
     86 
     87 def _raise_linalgerror_singular(err, flag):
---> 88     raise LinAlgError("Singular matrix")
     89 
     90 def _raise_linalgerror_nonposdef(err, flag):

LinAlgError: Singular matrix
```

Since row 1 and 2 in matrix A are dependent it gives Singular matrix error (det(A) = 0)

In [40]:
```python
print("A:")
print(A)
print(props(A))
print("******")
print("B:")
print(B)
print(props(B))
```

```
A:
[[ 2.  1. -3.]
 [ 4.  2. -6.]
 [ 1. -1.  1.]]
Square Matrix
The rank of the matrix is  2
The determinant of the matrix is  0.0
0
******
B:
[[0.]
 [0.]
 [0.]]
NOT Square Matrix
The rank of the matrix is  0
Not square matrix can't calculate determinant
0
```

Question 4:

```
def reducerow(C, pivotrow, targetrow, pivot):
    #check if pivot is not already 1 or zero since we cant divide by zero
    if (C[pivotrow][pivot] != 1) and (C[pivotrow][pivot] != 0):
        C[pivotrow] = C[pivotrow]/C[pivotrow][pivot]

    target_elem = C[targetrow][pivot]
    #check that the target element is not already 0
    if target_elem !=0:
        #subtract targetrow from element*pivotrow to make the element zero
        C[targetrow] = C[targetrow] - (target_elem*C[pivotrow])

C = np.array([[2,0,-6],[0,1,2],[3,6,-2]])
print("Before")
print(C)
reducerow(C, 0, 2, 0)
print("after")
print(C)
```

In [50]:
```
def reducerow(C, pivotrow, targetrow, pivot):
    #check if pivot is not already 1 or zero since we cant divide by zero
    if (C[pivotrow][pivot] != 1) and (C[pivotrow][pivot] != 0):
        C[pivotrow] = C[pivotrow]/C[pivotrow][pivot]

    target_elem = C[targetrow][pivot]
    #check that the target element is not already 0
    if target_elem !=0:
        #subtract targetrow from element*pivotrow to make the element zero
        C[targetrow] = C[targetrow] - (target_elem*C[pivotrow])
```

In [51]:
```
C = np.array([[2,0,-6],[0,1,2],[3,6,-2]])
print("Before")
print(C)
reducerow(C, 0, 2, 0)
print("after")
print(C)
```

```
Before
[[ 2  0 -6]
 [ 0  1  2]
 [ 3  6 -2]]
after
[[ 1  0 -3]
 [ 0  1  2]
 [ 0  6  7]]
```

Question5:

```
def SolveLinearSystem(A,B):
   C = np.concatenate((A,B), axis = 1)
   C.dtype = "float64"
   #start with pivot 0
   pivot = 0
   for pivotrow in range(A.shape[0]):

      #if pivotelement = 0 swap rows with the first row that doesn't have a 0 in pivotelement index
      if C[pivotrow][pivot] == 0:
         temprow = C[pivotrow]
         nextrow = pivotrow+1
         while(nextrow < A.shape[0]):
            if(C[nextrow][pivot] !=0):
               C[pivotrow] = C[nextrow]
               C[nextrow] = temprow
               #swap done -> exit the loop
               break
            else:
               #check next row
               nextrow = nextrow + 1

      #the target rows are all rows except pivot row
      targetrows = list(range(A.shape[0]))
      del targetrows[pivotrow] # remove index of pivot row from list

      for targetrow in targetrows: #iterate over all target rows
         reducerow(C, pivotrow, targetrow, pivot) #reduce target row
      pivot = pivot+1 #increase the pivot
   print("After row reduction")
   print(C)

   #summation of column elements of A
   return np.sum(C[:,0:A.shape[1]] )

#Test code
A= np.array([[2,0,-6],[0,1,2],[3,6,-2]], dtype = "float64")
print("A:")
print(A)
B = np.array([[-8],[3],[-4]], dtype= "float64")
print("B:")
print(B)
SolveLinearSystem(A,B)
```

In [130]:
```
def SolveLinearSystem(A,B):
    C = np.concatenate((A,B), axis = 1)
    C.dtype = "float64"
    #start with pivot 0
    pivot = 0
    for pivotrow in range(A.shape[0]):

        #if pivotelement = 0 swap rows with the first row that doesn't have a 0 in pivotelement index
        if C[pivotrow][pivot] == 0:
            temprow = C[pivotrow]
            nextrow = pivotrow+1
            while(nextrow < A.shape[0]):
                if(C[nextrow][pivot] !=0):
                    C[pivotrow] = C[nextrow]
                    C[nextrow] = temprow
                    #swap done -> exit the loop
                    break
                else:
                    #check next row
                    nextrow = nextrow + 1

        #the target rows are all rows except pivot row
        targetrows = list(range(A.shape[0]))
        del targetrows[pivotrow] # remove index of pivot row from list

        for targetrow in targetrows: #iterate over all target rows
            reducerow(C, pivotrow, targetrow, pivot) #reduce target row
        pivot = pivot+1 #increase the pivot
    print("After row reduction")
    print(C)

    #summation of column elements of A
    return np.sum(C[:,0:A.shape[1]])

#Test code
A= np.array([[2,0,-6],[0,1,2],[3,6,-2]], dtype = "float64")
print("A:")
print(A)
B = np.array([[-8],[3],[-4]], dtype= "float64")
print("B:")
print(B)
SolveLinearSystem(A,B)
```

```
A:
[[ 2.  0. -6.]
 [ 0.  1.  2.]
 [ 3.  6. -2.]]
B:
[[-8.]
 [ 3.]
 [-4.]]
After row reduction
[[ 1.  0.  0.  2.]
 [ 0.  1.  0. -1.]
 [-0. -0.  1.  2.]]
```

Out[130]: 3.0

Question6:

#Solve System 1
A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')
B = np.array([[589.5], [200], [20]], dtype='float64')
x = np.linalg.solve(A, B) #check with built in function
print("Built in function solution")
print(x)
SolveLinearSystem(A,B)

```
In [21]: #Solve System 1
         A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')
         B = np.array([[589.5], [200], [20]], dtype='float64')
         x = np.linalg.solve(A, B) #check with built in function
         print("Built in function solution")
         print(x)
         SolveLinearSystem(A,B)

         Built in function solution
         [[80.]
          [50.]
          [70.]]
         After row reduction
         [[ 1.  0.  0. 80.]
          [-0.  1.  0. 50.]
          [ 0.  0.  1. 70.]]

Out[21]: 3.0
```

#Solve System 2
A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
B = np.array([[0],[0],[0]], dtype="float64")
#x = np.linalg.solve(A, B) ->Singular matrix error
SolveLinearSystem(A,B)

```
In [132]: #Solve System 2
          A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
          B = np.array([[0],[0],[0]], dtype="float64")
          #x = np.linalg.solve(A, B)  ->Singular matrix error
          SolveLinearSystem(A,B)

          After row reduction
          [[ 1.          0.         -0.66666667  0.        ]
           [ 0.          1.         -1.66666667  0.        ]
           [ 0.          0.          0.          0.        ]]

Out[132]: -0.33333333333333337
```

Question7:

For System1: Sum = 3 we were able to reduce A to the identity matrix form

The solution is : [80

50

70]

which is the last column of the joined matrix AB after row reduction

```
After row reduction
[[ 1.   0.   0.  80.]
 [-0.   1.   0.  50.]
 [ 0.   0.   1.  70.]]
```

For System2: Sum < 3 , This is because we couldn’t reduce the matrix to the identity form as two rows are dependent on each other.

Currently there are infinite number of solutions, and we have a free variable

either we assume the last variable as an unknown variable and get the other two with respect to it or we get another equation that is not dependent on the equations we already have

if we assume x3 = k , we can get x1 = 0.6667x3 and x2 = 1.66667x3

where x3 → oil price, x1 → potatoes price, x2 → chicken price

```
After row reduction
[[ 1.          0.         -0.66666667  0.        ]
 [ 0.          1.         -1.66666667  0.        ]
 [ 0.          0.          0.          0.        ]]

Out[132]: -0.33333333333333337
```

Question 8:

System 1:

```
In [22]: #System1
         A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')
         B = np.array([[589.5], [200], [20]], dtype='float64')

         x = np.linalg.solve(A, B) # Solves a full-rank system of linear equations ax = b.
         print(x)

         np.allclose(np.dot(A, x), B) # Returns True if two arrays are element-wise equal within atolerance.

         [[80.]
          [50.]
          [70.]]
Out[22]: True
```

Same as my Solution

System2:

Singular matrix error

```
In [137]: #system2
          A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
          B = np.array([[0],[0],[0]], dtype="float64")
          x = np.linalg.solve(A, B) # Solves a full-rank system of linear equations ax = b.
          print(x)

          np.allclose(np.dot(A, x), B) # Returns True if two arrays are element-wise equal within atolerance.

          ---------------------------------------------------------------------------
          LinAlgError                               Traceback (most recent call last)
          <ipython-input-137-e0e0c79ae565> in <module>
                2 A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
                3 B = np.array([[0],[0],[0]], dtype="float64")
          ----> 4 x = np.linalg.solve(A, B) # Solves a full-rank system of linear equations ax = b.
                5 print(x)
                6

          <__array_function__ internals> in solve(*args, **kwargs)

          ~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in solve(a, b)
              391     signature = 'DD->D' if isComplexType(t) else 'dd->d'
              392     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
          --> 393     r = gufunc(a, b, signature=signature, extobj=extobj)
              394
              395     return wrap(r.astype(result_t, copy=False))

          ~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_singular(err, flag)
               86
               87 def _raise_linalgerror_singular(err, flag):
          ---> 88     raise LinAlgError("Singular matrix")
               89
               90 def _raise_linalgerror_nonposdef(err, flag):

          LinAlgError: Singular matrix
```

Question 9:

#System 1
A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1,1]], dtype='float64')

#to get inverse of A we put B as identity matrix
B = np.identity(3, dtype="float64")

SolveLinearSystem(A,B)

In [24]:
```
#System 1
A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')

#to get inverse of A we put B as identity matrix
B = np.identity(3, dtype="float64")

SolveLinearSystem(A,B)
```

```
After row reduction
[[ 1.          0.          0.         -0.37383178  1.56074766 -0.58878505]
 [-0.          1.          0.          0.18691589 -0.28037383 -0.20560748]
 [ 0.          0.          1.          0.18691589 -0.28037383  0.79439252]]
```

Out[24]: 3.0

#System 2
A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")

#to get inverse of A we put B as identity matrix
B = np.identity(3, dtype="float64")

SolveLinearSystem(A,B)

In [139]:
```
#System 2
A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")

#to get inverse of A we put B as identity matrix
B = np.identity(3, dtype="float64")

SolveLinearSystem(A,B)
```

```
After row reduction
[[ 1.          0.         -0.66666667  0.33333333  0.          0.33333333]
 [ 0.          1.         -1.66666667  0.33333333  0.         -0.66666667]
 [ 0.          0.          0.          0.          0.          0.        ]]
```

Out[139]: -0.33333333333333337

Question 10:

#System1
A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')

np.linalg.inv(A)

```
In [26]: #System1
         A = np.array([[1.5, 5.75, 2.6],[1, 1, 1], [0, -1, 1]], dtype='float64')

         np.linalg.inv(A)

Out[26]: array([[-0.37383178,  1.56074766, -0.58878505],
                [ 0.18691589, -0.28037383, -0.20560748],
                [ 0.18691589, -0.28037383,  0.79439252]])
```

#System2
A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")

np.linalg.inv(A)

```
In [142]: #System2
          A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")

          np.linalg.inv(A)

          ---------------------------------------------------------------------------
          LinAlgError                               Traceback (most recent call last)
          <ipython-input-142-e0ff62dc8d35> in <module>
                2 A = np.array([[2, 1, -3], [4, 2, -6], [1, -1, 1]], dtype= "float64")
                3
          ----> 4 np.linalg.inv(A)

          <__array_function__ internals> in inv(*args, **kwargs)

          ~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in inv(a)
              543     signature = 'D->D' if isComplexType(t) else 'd->d'
              544     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
          --> 545     ainv = _umath_linalg.inv(a, signature=signature, extobj=extobj)
              546     return wrap(ainv.astype(result_t, copy=False))
              547

          ~/anaconda3/lib/python3.8/site-packages/numpy/linalg/linalg.py in _raise_linalgerror_singular(err, flag)
               86
               87 def _raise_linalgerror_singular(err, flag):
          ---> 88     raise LinAlgError("Singular matrix")
               89
               90 def _raise_linalgerror_nonposdef(err, flag):

          LinAlgError: Singular matrix
```