SEAS 8510 - Analytical MEthods for ML

Homework 2

Due Date: April 6, 2024 (9:00am EST)

Your homework 2 should be submitted as a pdf containing your responses, your code, and your printed results where applicable.

All of the imports

In [ ]:
```
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
%matplotlib inline
```

2.4

Compute the following matrix products, if possible. Calculate by hand and verify in Python.

a.

$$\begin{bmatrix} 1 & 2 \\ 4 & 5 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

This is multiplying a 3x2 matrix with a 3x3 matrix. Since the inner dimensions are different (2 and 3) they cannot be multiplied.

In [ ]:
```
m1 = np.array([[1,2],[4,5],[7,8]])
m2 = np.array([[1,1,0],[0,1,1],[1,0,1]])

print("m1")
print(m1.shape)
print(m1)
```

```
print("m2")
print(m2.shape)
print(m2)

print("m1 X m2")
m3 = np.matmul(m1, m2)
print("Did not work as they are not compatible sizes.")
```

```
m1
(3, 2)
[[1 2]
 [4 5]
 [7 8]]
m2
(3, 3)
[[1 1 0]
 [0 1 1]
 [1 0 1]]
m1 X m2
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[2], line 12
      9 print(m2)
     11 print("m1 X m2")
---> 12 m3 = np.matmul(m1, m2)
     13 print("Did not work as they are not compatible sizes.")

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 2)
```

My results and Numpy agree that these two matrices cannot be multiplied together.

b.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Here we have two 3x3 matrices which can be multiplied together. The result is a 3x3 matrix.

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}1&1&0\\0&1&1\\1&0&1\end{bmatrix}=\begin{bmatrix}1*1+2*0+3*1&1*1+2*1+3*0&1*0+2*1+3*1\\4*1+5*0+6*1&4*1+5*1+6*0&4*0+5*1+6*1\\7*1+8*0+9*1&7*1+8*1+9*0&7*0+8*1+9*1\end{bmatrix}=$$
$$\begin{bmatrix}1+0+3&1+2+0&0+2+3\\4+0+6&4+5+0&0+5+6\\7+0+9&7+8+0&0+8+9\end{bmatrix}=\begin{bmatrix}4&3&5\\10&9&11\\16&15&17\end{bmatrix}$$

In [ ]:
```
m1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
m2 = np.array([[1,1,0],[0,1,1],[1,0,1]])

print("m1")
print(m1.shape)
print(m1)
print("m2")
print(m2.shape)
print(m2)
print("m1 x m2")
m3 = np.matmul(m1, m2)
print(m3.shape)
print(m3)
```

```
m1
(3, 3)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
m2
(3, 3)
[[1 1 0]
 [0 1 1]
 [1 0 1]]
m1 x m2
(3, 3)
[[ 4  3  5]
 [10  9 11]
 [16 15 17]]
```

C.

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Here we have two 3x3 matrices which can be multiplied together. The result is a 3x3 matrix.

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1*1+1*4+0*7 & 1*2+1*5+0*8 & 1*3+1*6+0*9 \\ 0*1+1*4+1*7 & 0*2+1*5+1*8 & 0*3+1*6+1*9 \\ 1*1+0*4+1*7 & 1*2+0*5+1*8 & 1*3+0*6+1*9 \end{bmatrix} =$$
$$\begin{bmatrix} 1+4+0 & 2+5+0 & 3+6+0 \\ 0+4+7 & 0+5+8 & 0+6+9 \\ 1+0+7 & 2+0+8 & 3+0+9 \end{bmatrix} = \begin{bmatrix} 5 & 7 & 9 \\ 11 & 13 & 15 \\ 8 & 10 & 12 \end{bmatrix}$$

In [ ]:

```
m1 = np.array([[1,1,0],[0,1,1],[1,0,1]])
m2 = np.array([[1,2,3],[4,5,6],[7,8,9]])

print("m1")
print(m1.shape)
print(m1)
print("m2")
print(m2.shape)
print(m2)
print("m1 x m2")
m3 = np.matmul(m1, m2)
print(m3.shape)
print(m3)
```

```
m1
(3, 3)
[[1 1 0]
 [0 1 1]
 [1 0 1]]
m2
(3, 3)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
m1 x m2
(3, 3)
[[ 5  7  9]
 [11 13 15]
 [ 8 10 12]]
```

d.

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

Here we have a 2x4 matrix multiplied by a 4x2 matrix which can be multiplied together. The result is a 2x2 matrix.

$$\begin{bmatrix} 1 & 2 & 1 & 2 \\ 4 & 1 & -1 & 4 \end{bmatrix} \begin{bmatrix} 0 & 3 \\ 1 & -1 \\ 2 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 1*0+2*1+1*2+2*5 & 1*3+2*-1+1*1+2*2 \\ 4*0+1*1+-1*2+4*5 & 4*3+1*-1+-1*1+4*2 \end{bmatrix} =$$
$$\begin{bmatrix} 0+2+2+10 & 3+-2+1+4 \\ 0+1+-2+20 & 12+-1+-1+8 \end{bmatrix} = \begin{bmatrix} 14 & 6 \\ 19 & 18 \end{bmatrix}$$

In [ ]:
```
m1 = np.array([[1,2,1,2],[4,1,-1,4]])
m2 = np.array([[0,3],[1,-1],[2,1],[5,2]])

print("m1")
print(m1.shape)
```

```
print(m1)
print("m2")
print(m2.shape)
print(m2)
print("m1 x m2")
m3 = np.matmul(m1, m2)
print(m3.shape)
print(m3)
```

```
m1
(2, 4)
[[ 1  2  1  2]
 [ 4  1 -1  4]]
m2
(4, 2)
[[ 0  3]
 [ 1 -1]
 [ 2  1]
 [ 5  2]]
m1 x m2
(2, 2)
[[14  6]
 [19 18]]
```

e.

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

Here we have a 4x2 matrix multiplied by a 2x4 matrix which can be multiplied together. The result is a 4x4 matrix.

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

$$\begin{bmatrix} 0+12 & 0+3 & 0+-3 & 0+12 \\ 1+-4 & 2+-1 & 1+1 & 2+-4 \\ 2+4 & 4+1 & 2+-1 & 4+4 \\ 5+8 & 10+2 & 5+-2 & 10+8 \end{bmatrix} = \begin{bmatrix} 12 & 3 & -3 & 12 \\ -3 & 1 & 2 & -2 \\ 6 & 5 & 1 & 8 \\ 13 & 12 & 3 & 18 \end{bmatrix}$$

In [ ]:
```
m1 = np.array([[0,3],[1,-1],[2,1],[5,2]])
m2 = np.array([[1,2,1,2],[4,1,-1,4]])

print("m1")
print(m1.shape)
print(m1)
print("m2")
print(m2.shape)
print(m2)
print("m1 x m2")
m3 = np.matmul(m1, m2)
print(m3.shape)
print(m3)
```

```
m1
(4, 2)
[[ 0  3]
 [ 1 -1]
 [ 2  1]
 [ 5  2]]
m2
(2, 4)
[[ 1  2  1  2]
 [ 4  1 -1  4]]
m1 x m2
(4, 4)
[[12  3 -3 12]
 [-3  1  2 -2]
 [ 6  5  1  8]
 [13 12  3 18]]
```

For 2.5 – 2.7, calculate by hand and then show how the results of using numpy.linalg.solve

2.5

Find the set S of all solutions in x of the following inhomogeneous linear systems $Ax = b$, where A and b are defined as follows:

a.

$$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

Using Gauss-Jordan to solve the equations. Create the augmented matrix $\begin{bmatrix} 1 & 1 & -1 & -1 & 1 \\ 2 & 5 & -7 & -5 & -2 \\ 2 & -1 & 1 & 3 & 4 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Multiply row one by 2

$\begin{bmatrix} 2 & 2 & -2 & -2 & 2 \\ 2 & 5 & -7 & -5 & -2 \\ 2 & -1 & 1 & 3 & 4 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Subtract row one from row two $\begin{bmatrix} 2 & 2 & -2 & -2 & 2 \\ 0 & 3 & -5 & -3 & -4 \\ 2 & -1 & 1 & 3 & 4 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Subtract row one from row three

$\begin{bmatrix} 2 & 2 & -2 & -2 & 2 \\ 0 & 3 & -5 & -3 & -4 \\ 0 & -3 & 3 & 5 & 2 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Multiply row one by $\frac{1}{2}$ $\begin{bmatrix} 1 & 1 & -1 & -1 & 1 \\ 0 & 3 & -5 & -3 & -4 \\ 0 & -3 & 3 & 5 & 2 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Multiply row one by 5

$\begin{bmatrix} 5 & 5 & -5 & -5 & 5 \\ 0 & 3 & -5 & -3 & -4 \\ 0 & -3 & 3 & 5 & 2 \\ 5 & 2 & -4 & 2 & 6 \end{bmatrix}$ Subtract row one from row four $\begin{bmatrix} 5 & 5 & -5 & -5 & 5 \\ 0 & 3 & -5 & -3 & -4 \\ 0 & -3 & 3 & 5 & 2 \\ 0 & -3 & 1 & 7 & 1 \end{bmatrix}$ Multiply row one by $\frac{1}{5}$

$$\begin{bmatrix}1&1&-1&-1&1\\0&3&-5&-3&-4\\0&-3&3&5&2\\0&-3&1&7&1\end{bmatrix} \text{Multiply row two by } \frac{1}{3} \begin{bmatrix}1&1&-1&-1&1\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&-3&3&5&2\\0&-3&1&7&1\end{bmatrix} \text{Subtract row two from row one}$$

$$\begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&-3&3&5&2\\0&-3&1&7&1\end{bmatrix} \text{Multiply row two by } -3 \begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&-3&5&3&4\\0&-3&3&5&2\\0&-3&1&7&1\end{bmatrix} \text{Subtract row two from row three}$$

$$\begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&-3&5&3&4\\0&0&-2&2&-2\\0&-3&1&7&1\end{bmatrix} \text{Subtract row two from row four} \begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&-3&5&3&4\\0&0&-2&2&-2\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row two by } -\frac{1}{3}$$

$$\begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&-2&2&-2\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } -\frac{1}{2} \begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&1&-1&1\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } \frac{2}{3}$$

$$\begin{bmatrix}1&0&\frac{2}{3}&0&\frac{7}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&\frac{2}{3}&-\frac{2}{3}&\frac{2}{3}\\0&0&-4&4&-3\end{bmatrix} \text{Subtract row three from row one} \begin{bmatrix}1&0&0&\frac{2}{3}&\frac{5}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&\frac{2}{3}&-\frac{2}{3}&\frac{2}{3}\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } 3$$

$$\begin{bmatrix}1&0&0&\frac{2}{3}&\frac{5}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&1&-1&1\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } -\frac{5}{3} \begin{bmatrix}1&0&0&\frac{2}{3}&\frac{5}{3}\\0&1&-\frac{5}{3}&-1&-\frac{4}{3}\\0&0&-\frac{5}{3}&\frac{5}{3}&-\frac{5}{3}\\0&0&-4&4&-3\end{bmatrix} \text{Subtract row three from row two}$$

$$\begin{bmatrix}1&0&0&\frac{2}{3}&\frac{5}{3}\\0&1&0&-\frac{8}{3}&\frac{1}{3}\\0&0&-\frac{5}{3}&\frac{5}{3}&-\frac{5}{3}\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } -\frac{3}{5} \begin{bmatrix}1&0&0&\frac{2}{3}&\frac{5}{3}\\0&1&0&-\frac{8}{3}&\frac{1}{3}\\0&0&1&-1&1\\0&0&-4&4&-3\end{bmatrix} \text{Multiply row three by } -4$$

$\begin{bmatrix} 1 & 0 & 0 & \frac{2}{3} & \frac{5}{3} \\ 0 & 1 & 0 & -\frac{8}{3} & \frac{1}{3} \\ 0 & 0 & -4 & 4 & -4 \\ 0 & 0 & -4 & 4 & -3 \end{bmatrix}$ Subtract row three from row four $\begin{bmatrix} 1 & 0 & 0 & \frac{2}{3} & \frac{5}{3} \\ 0 & 1 & 0 & -\frac{8}{3} & \frac{1}{3} \\ 0 & 0 & -4 & 4 & -4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ We now have all zeros in row four so the system is inconsistent. There is no solution.

In [ ]:
```
A = np.array([[1,1,-1,-1],[2,5,-7,-5],[2,-1,1,3],[5,2,-4,2]])
b = np.array([[1],[-2],[4],[6]])

print("A")
print(A.shape)
print(A)
print("b")
print(b.shape)
print(b)
print("Solving Ax = b")
sol = np.linalg.solve(A,b)
print(sol.shape)
print(sol)
```

```
A
(4, 4)
[[ 1  1 -1 -1]
 [ 2  5 -7 -5]
 [ 2 -1  1  3]
 [ 5  2 -4  2]]
b
(4, 1)
[[ 1]
 [-2]
 [ 4]
 [ 6]]
Solving Ax = b
(4, 1)
[[ 7.50599938e+14]
 [-3.00239975e+15]
 [-1.12589991e+15]
 [-1.12589991e+15]]
```

Why could Numpy find a solution? It must be using a numerical method that will approximate a solution. The numbers are very small.

b.

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix},\ b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

Using Gauss-Jordan to solve the equations. Create the augmented matrix $\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 3 \\ 1 & 1 & 0 & -3 & 0 & 6 \\ 2 & -1 & 0 & 1 & -1 & 5 \\ -1 & 2 & 0 & -2 & -1 & -1 \end{bmatrix}$ Subtract row one from row two $\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 3 \\ 0 & 2 & 0 & -3 & -1 & 3 \\ 2 & -1 & 0 & 1 & -1 & 5 \\ -1 & 2 & 0 & -2 & -1 & -1 \end{bmatrix}$ Subtract 2 times the first row from the third row $\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 3 \\ 0 & 2 & 0 & -3 & -1 & 3 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ -1 & 2 & 0 & -2 & -1 & -1 \end{bmatrix}$ Subtract -1 times the first row from the fourth row $\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 3 \\ 0 & 2 & 0 & -3 & -1 & 3 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 1 & 0 & -2 & 0 & 2 \end{bmatrix}$ Swap the second and third rows $\begin{bmatrix} 1 & -1 & 0 & 0 & 1 & 3 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 2 & 0 & -3 & -1 & 3 \\ 0 & 1 & 0 & -2 & 0 & 2 \end{bmatrix}$ Subtract -1 times the second row from the first row $\begin{bmatrix} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 2 & 0 & -3 & -1 & 3 \\ 0 & 1 & 0 & -2 & 0 & 2 \end{bmatrix}$ Subtract 2 times the second row from the third row $\begin{bmatrix} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 0 & 0 & -5 & 5 & 5 \\ 0 & 1 & 0 & -2 & 0 & 2 \end{bmatrix}$ Subtract the second row from the fourth row

$\begin{bmatrix} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 0 & 0 & -5 & 5 & 5 \\ 0 & 0 & 0 & -3 & 3 & 3 \end{bmatrix}$ Divide the third row by -5 $\begin{bmatrix} 1 & 0 & 0 & 1 & -2 & 2 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & -3 & 3 & 3 \end{bmatrix}$ Subtract the third row from the first row $\begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 1 & -3 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & -3 & 3 & 3 \end{bmatrix}$ Subtract the third row from the second row $\begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & -3 & 3 & 3 \end{bmatrix}$ Subtract -3 times the third row from the fourth row $\begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ Reading from the matrix, the solution set is: $x_1 = 3 + x_5$ $x_2 = 2x_5$ x_3 is free $x_4 = -1 + x_5$ x_5 is free

In [ ]:

```
A = np.array([[1,-1,0,0,1],[1,1,0,-3,0],[2,-1,0,1,-1],[-1,2,0,-2,-1]])
b = np.array([[3],[6],[5],[-1]])

print("A")
print(A.shape)
print(A)
print("b")
print(b.shape)
print(b)
print("Solving Ax = b")
sol = np.linalg.solve(A,b)
print(sol.shape)
print(sol)
```

```
A
(4, 5)
[[ 1 -1  0  0  1]
 [ 1  1  0 -3  0]
 [ 2 -1  0  1 -1]
 [-1  2  0 -2 -1]]
b
(4, 1)
[[ 3]
 [ 6]
 [ 5]
 [-1]]
Solving Ax = b
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
Cell In[8], line 11
      9 print(b)
     10 print("Solving Ax = b")
---> 11 sol = np.linalg.solve(A,b)
     12 print(sol.shape)
     13 print(sol)

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:396, in solve(a, b)
    394 a, _ = _makearray(a)
    395 _assert_stacked_2d(a)
--> 396 _assert_stacked_square(a)
    397 b, wrap = _makearray(b)
    398 t, result_t = _commonType(a, b)

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:213, in _assert_stacked_square(*arrays)
    211 m, n = a.shape[-2:]
    212 if m != n:
--> 213     raise LinAlgError('Last 2 dimensions of the array must be square')

LinAlgError: Last 2 dimensions of the array must be square
```

I suppose that Numpy is using a numerical method that can only solve for one value and therefore does not work in this case.

2.6

Use Gaussian Elimination, find all solutions of the inhomogeneous equation system $Ax = b$ with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \ b = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

Using Gauss-Jordan to solve the equations. Create the augmented matrix $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ Subtract row one from row three $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix}$ Multiply row three by -1 $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$ Subtract row three from row one $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$ Subtract row three from row two $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{bmatrix}$ Reading from the matrix, the solution set is: x_1 is free $x_2 = 1 - x_6$ x_3 is free $x_4 = -2 - x_6$ $x_5 = 1 + x_6$ x_6 is free

In [ ]:
```
A = np.array([[0,1,0,0,1,0],[0,0,0,1,1,0],[0,1,0,0,0,1]])
b = np.array([[2],[-1],[1]])

print("A")
print(A.shape)
print(A)
print("b")
print(b.shape)
print(b)
print("Solving Ax = b")
sol = np.linalg.solve(A,b)
print(sol.shape)
print(sol)
```

```
A
(3, 6)
[[0 1 0 0 1 0]
 [0 0 0 1 1 0]
 [0 1 0 0 0 1]]
b
(3, 1)
[[ 2]
 [-1]
 [ 1]]
Solving Ax = b
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
Cell In[9], line 11
      9 print(b)
     10 print("Solving Ax = b")
---> 11 sol = np.linalg.solve(A,b)
     12 print(sol.shape)
     13 print(sol)

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:396, in solve(a, b)
    394 a, _ = _makearray(a)
    395 _assert_stacked_2d(a)
--> 396 _assert_stacked_square(a)
    397 b, wrap = _makearray(b)
    398 t, result_t = _commonType(a, b)

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:213, in _assert_stacked_square(*arrays)
    211 m, n = a.shape[-2:]
    212 if m != n:
--> 213     raise LinAlgError('Last 2 dimensions of the array must be square')

LinAlgError: Last 2 dimensions of the array must be square
```

2.7

Find all solutions in $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^3$ of the equation system $Ax = 12x$, where

$A = \begin{bmatrix} 6 & 4 & 3 \\ 6 & 0 & 9 \\ 0 & 8 & 0 \end{bmatrix}$ and $\sum_{i=1}^{3} x_i = 1$

Given $A = \begin{bmatrix} 6 & 4 & 3 \\ 6 & 0 & 9 \\ 0 & 8 & 0 \end{bmatrix}$ $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $Ax = 12x$ and $\sum_{i=1}^{3} x_i = 1$ then $\frac{1}{12}Ax = x$ $\frac{1}{12}\begin{bmatrix} 6 & 4 & 3 \\ 6 & 0 & 9 \\ 0 & 8 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{3}{4} \\ 0 & \frac{2}{3} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ From row three of the matrix $\frac{2}{3}x_2$ = x_3 From row one of the matrix $\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 = x_1$ Substituting for x_3 $\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}\frac{2}{3}x_2 = x_1$ Calculating $\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{2}{12}x_2 = x_1$ Multiply term two by $\frac{4}{4}$ $\frac{1}{2}x_1 + \frac{4}{12}x_2 + \frac{2}{12}x_2 = x_1$ Combine terms two and three $\frac{1}{2}x_1 + \frac{6}{12}x_2 = x_1$ Simplify $\frac{1}{2}x_1 + \frac{1}{2}x_2 = x_1$ Combine terms one and two $\frac{1}{2}x_2 = \frac{1}{2}x_1$ Multiply both sides by 2 $x_2 = x_1$ We now have $x_1 = x_2$ and $x_3 = \frac{2}{3}x_2$. We also know that $\sum_{i=1}^{3} x_i = 1$. Expand the sum $x_1 + x_2 + x_3 = 1$ Substitute for x_1 and x_3 $x_2 + x_2 + \frac{2}{3}x_2 = 1$ Combine terms one, two, and three $2\frac{2}{3}x_2 = 1$ Express as a fraction $\frac{8}{3}x_2 = 1$ Solve for x_2 $x_2 = \frac{3}{8}$ Therefore $x_1 = \frac{3}{8}$ $x_2 = \frac{3}{8}$ $x_3 = \frac{2}{8}$

This seems to be the only solution. I cannot think of a way to present this to Numpy to calculate.

For 2.8, calculate by hand and then show the results of using numpy.linalg.inv

2.8

Determine the inverses of the following matrices if possible:

a.

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

Attempting to invert the matrix by augmenting it with an Identity Matrix and then performing gaussian elimination. Append the identity matrix $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 3 & 4 & 5 & 0 & 1 & 0 \\ 4 & 5 & 6 & 0 & 0 & 1 \end{bmatrix}$ Swap rows 2 and 3 $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 1 \\ 3 & 4 & 5 & 0 & 1 & 0 \end{bmatrix}$ Subtract 2 * Row 1 from Row 2 $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 0 & -1 & -2 & -2 & 0 & 1 \\ 3 & 4 & 5 & 0 & 1 & 0 \end{bmatrix}$ Multiply row 3 by 2 $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 0 & -1 & -2 & -2 & 0 & 1 \\ 6 & 8 & 10 & 0 & 2 & 0 \end{bmatrix}$ Subtract 3 times row 1 from row 3 $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 0 & -1 & -2 & -2 & 0 & 1 \\ 0 & -1 & -2 & -3 & 2 & 0 \end{bmatrix}$ Subtract row 2 from row 3 $\begin{bmatrix} 2 & 3 & 4 & 1 & 0 & 0 \\ 0 & -1 & -2 & -2 & 0 & 1 \\ 0 & 0 & 0 & -1 & 2 & -1 \end{bmatrix}$ The inverse is There is no inverse since the two rows had the same values, the matrix is singular.

In [ ]:
```python
# In code
A = np.array([[2,3,4],[3,4,5],[4,5,6]])

print("A")
print(A.shape)
print(A)
print("Calculating the inverse of A")
sol = np.linalg.inv(A)
print(sol.shape)
print(sol)
```

```
A
(3, 3)
[[2 3 4]
 [3 4 5]
 [4 5 6]]
Calculating the inverse of A
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
Cell In[10], line 8
      6 print(A)
      7 print("Calculating the inverse of A")
----> 8 sol = np.linalg.inv(A)
      9 print(sol.shape)
     10 print(sol)

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:561, in inv(a)
    559 signature = 'D->D' if isComplexType(t) else 'd->d'
    560 extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 561 ainv = _umath_linalg.inv(a, signature=signature, extobj=extobj)
    562 return wrap(ainv.astype(result_t, copy=False))

File c:\Users\Micha\AppData\Local\Programs\Python\Python312\Lib\site-packages\numpy\linalg\linalg.py:112, in _raise_l
inalgerror_singular(err, flag)
    111 def _raise_linalgerror_singular(err, flag):
--> 112     raise LinAlgError("Singular matrix")

LinAlgError: Singular matrix
```

My results and Numpy agree that this matrix has no inverse.

b.

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

Attempting to invert the matrix by augmenting it with an Identity Matrix and then performing gaussian elimination. Append the identity matrix $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ Subtract row 1 from row 3 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ Subtract row 1 from row 4 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$ Subtract row 2 from row 3 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$ Subtract row 2 from row 4 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 & 0 & 1 \end{bmatrix}$ Swap rows 3 and 4 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$ Multiply row 3 by -1 $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$ Subtract row 3 from row 1 $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$ Subtract row 3 from row 2 $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$ Multiply row 3 by -2 $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -2 & 0 & -2 & -2 & 0 & 2 \\ 0 & 0 & -2 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$ Subtract row 3 from row 4 $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -2 & 0 & -2 & -2 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & -2 \end{bmatrix}$ Multiply row 3 by $-\frac{1}{2}$ $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & -2 \end{bmatrix}$ The inverse is $\begin{bmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & -2 \end{bmatrix}$

In [ ]:

```
# In code
A = np.array([[1,0,1,0],[0,1,1,0],[1,1,0,1],[1,1,1,0]])

print("A")
print(A.shape)
print(A)
print("Calculating the inverse of A")
sol = np.linalg.inv(A)
print(sol.shape)
print(sol)
```

```
A
(4, 4)
[[1 0 1 0]
 [0 1 1 0]
 [1 1 0 1]
 [1 1 1 0]]
Calculating the inverse of A
(4, 4)
[[ 0. -1.  0.  1.]
 [-1.  0.  0.  1.]
 [ 1.  1. -0. -1.]
 [ 1.  1.  1. -2.]]
```

The results match. It actually took me multiple tries to get to a solution.