Inverse of a matrix

Goals

- Determining whether a matrix is invertible and computing its inverse (if it exists) by hand
- Computing inverses as a product of elementary matrices
- Writing a basic python function that determine whether a matrix is invertible and finding the inverse

Please read the following points before starting:

- It is highly recommended to finish the corresponding After-Class Assignments (ACs) before working on these In-Class Assignments (ICs). The ICs are designed to help you start with basic problems solved by hand to develop foundational thinking skills. You will then transition to Python-based commands to tackle slightly more challenging problems.
- As you work through these assignments, keep your lecture notes open for reference as they contain useful information to guide you. If you get stuck, please do not hesitate to raise your hand and ask for help from the instructor or the teaching/learning assistants.
- We strongly encourage you to use class time to work on these problems. Please note that solutions to the In-Class Assignments (ICs) will not be provided, so your best opportunity to resolve any questions or difficulties is by working on them here and discussing them with us.

Okay, now let us start with the first problem.

Calculating an Inverse Matrix

Consider the matrix A given by $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$. Use the Gauss-Jordan Elimination to find A^{-1} by row reducing $[A|I]$ to $[I|A^{-1}]$.

The first step is $\frac{1}{2}$ (row 1) + (row 2):

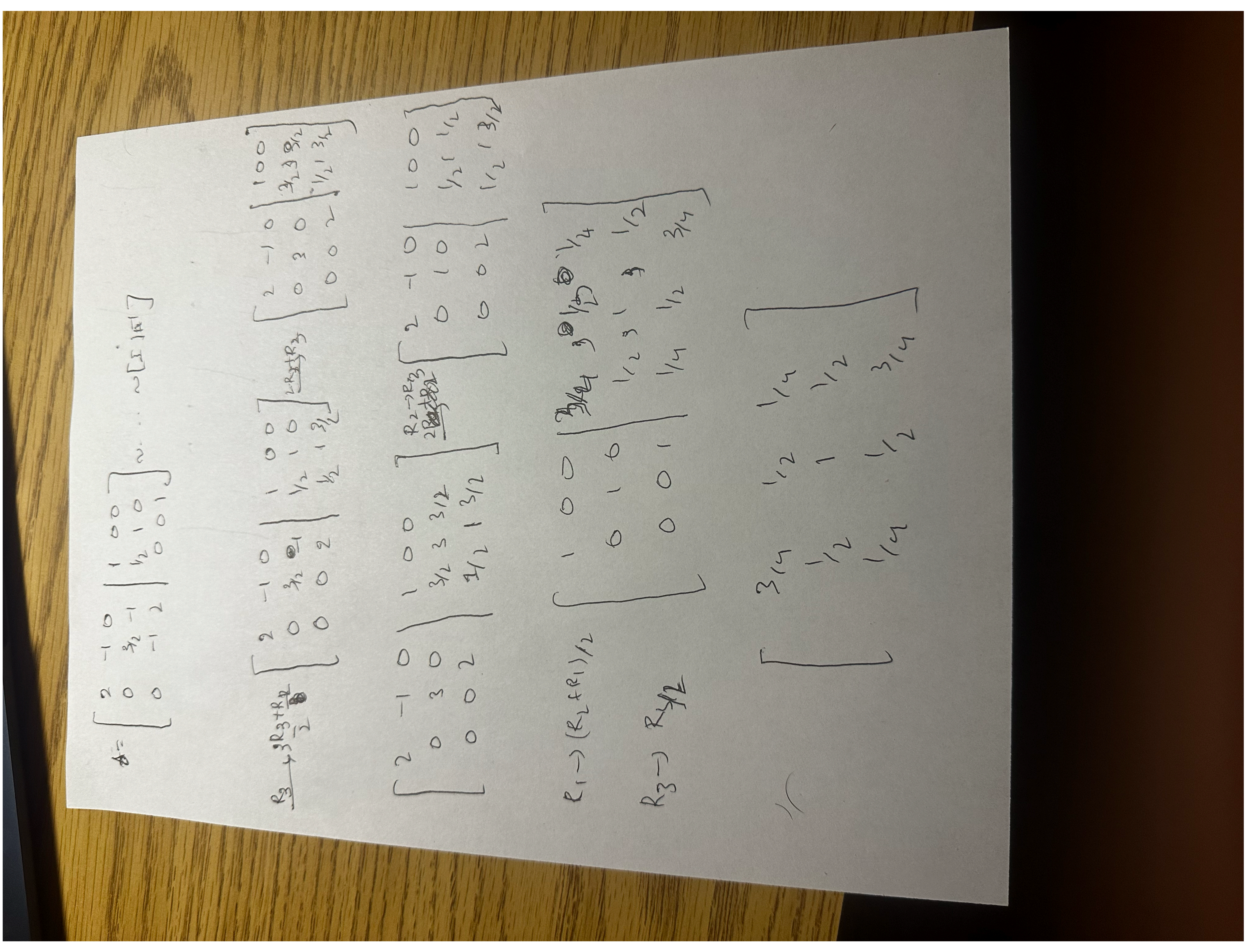
$$\left[\begin{array}{ccc|ccc} 2 & -1 & 0 & 1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 2 & 0 & 0 & 1 \end{array}\right] \sim \left[\begin{array}{ccc|ccc} 2 & -1 & 0 & 1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & \frac{1}{2} & 1 & 0 \\ 0 & -1 & 2 & 0 & 0 & 1 \end{array}\right] \sim \cdots \sim [I|A^{-1}]$$

Problem (BY HAND) (a) Complete the rest of the steps to find A^{-1} by hand (on paper).

(b) What are the corresponding elementary matrices you used?

Import a photo of your calculations or type everything in Latex.

put your answers here



Inverse matrix in Numpy

As usual, Python has a built-in function for numpy arrays that will calculate the inverse of a matrix `np.linalg.inv()`

```
import numpy as np
A = np.matrix([[1, 2, 3], [4, 5, 6], [7,8,7]])
A_inv= np.linalg.inv(A)
```

In [8]:
```
import numpy as np
import sympy as sym
sym.init_printing(use_unicode=True) # Trick to make matrices look nice in jupyter
```

Example. Invert the following matrix A. Store the inverse in a new matrix named A_inv.

In [10]:
```
A = np.matrix([[1, 2, 3], [4, 5, 6], [7,8,7]])
sym.Matrix(A)
```

Out[10]: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 7 \end{bmatrix}$

In [11]:
```
A_inv = np.linalg.inv(A)

sym.init_printing(use_unicode=True) # Trick to make matrixes look nice in jupyter
print(A_inv)
```

```
[[-2.16666667  1.66666667 -0.5       ]
 [ 2.33333333 -2.33333333  1.        ]
 [-0.5         1.         -0.5       ]]
```

Let's check your answer by multiplying `A` by `A_inv`.

In [13]:
```
A@A_inv
```

Out[13]:
```
matrix([[ 1.00000000e+00,  4.44089210e-16,  1.11022302e-16],
        [ 8.88178420e-16,  1.00000000e+00,  2.22044605e-16],
        [ 2.22044605e-15, -4.44089210e-16,  1.00000000e+00]])
```

If you look at the answer, it might look wrong at a first glance. This is the way python performs behind the scenes the operations and gives a result that seems wrong but is actually pretty accurate. A number of this form `-4.4408921e-16` is a number that is pretty close to zero. Actually it is equal to `-0.0000000000000004408921` (having 16 zeroes before the first non-zero decimal). One way to check that this is pretty close to the identity matrix is the following code.

```
np.allclose(A@A_inv, [[1,0,0],[0,1,0],[0,0,1]])
```

In [15]:
```
#Run the code to check that A@A_inv is indeed the identity matrix.
np.allclose(A@A_inv, [[1,0,0],[0,1,0],[0,0,1]])
```

Out[15]: True

DO THIS: Now, consider the following matrix `B`. Try to find its inverse. Explain why you are getting the result you are getting.

In [17]:
```
sym.init_printing(use_unicode=True) # Trick to make matrices look nice in jupyter

B = np.matrix([[1, 2, 3], [4, 5, 6], [2,4,6]])
try:
    B_inv= np.linalg.inv(B)
    sym.init_printing(use_unicode=True)
    sym.Matrix(B_inv)
except:
    print("There is no inverse")
```

```
There is no inverse
```

In [ ]:

How do we create an inverse matrix?

QUESTION: Is an invertible matrix always square? Why or why not?

put your answer here

the a matrix inverse requires that when multiplied by the original matrix, it produces the identity matrix, which is always square , and for this multiplication to be defined, both matrices must have the same number of rows and columns, meaning they must be square matrices

QUESTION: Is a square matrix always invertible? Why or why not? Provide an example.

put your answer here

this is not always true as the detreminent of the matrix cannot be guarentied to be a non zero value

Example. Describe the Elementary Row Operation that is implemented by the following matrix

$$E_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

put your answer here

The elementary row operation implemented by this matrix is swapping row 1 and row 2.

Consider the matrix A given by:

In [27]:
```
A = np.matrix([[3, -3,9], [2, -2, 7], [-1, 2, -4]])
sym.Matrix(A)
```

Out[27]: $\begin{bmatrix} 3 & -3 & 9 \\ 2 & -2 & 7 \\ -1 & 2 & -4 \end{bmatrix}$

Multiply the matrix E_1 from the left with A. What happened to the matrix A ?

In [29]:
```
E1 = np.matrix([[0,1,0],[1,0,0],[0,0,1]])
sym.Matrix(E1)
```

Out[29]: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

In [30]:
```
A1 = E1@A
sym.Matrix(A1)
```

Out[30]: $\begin{bmatrix} 2 & -2 & 7 \\ 3 & -3 & 9 \\ -1 & 2 & -4 \end{bmatrix}$

put your answer here

the row1 and row2 have swapped with each other in matrix A

DO THIS: Define a 3×3 elementary matrix named `E2` that swaps row 3 with row 1. Apply it to the matrix `A`, and store the result as a new variable `A2`.

In [33]:
```
# Put your answer here.
E2 = np.matrix([[0,0,1],[0,1,0],[1,0,0]])
A2 = E2@A
sym.Matrix(A2)
```

Out[33]: $\begin{bmatrix} -1 & 2 & -4 \\ 2 & -2 & 7 \\ 3 & -3 & 9 \end{bmatrix}$

DO THIS: Give a 3×3 elementary matrix named `E3` that adds 3 times the first row to the third row. Apply this elementary matrix to the matrix `A2`, and store the result as a new variable `A3`.

In [35]:
```
# Put your answer here.
E3 = np.array([[1,0,0],[0,1,0],[3,0,1]])
A3 = E3@A2
sym.Matrix(A3)
```

Out[35]: $\begin{bmatrix} -1 & 2 & -4 \\ 2 & -2 & 7 \\ 0 & 3 & -3 \end{bmatrix}$

DO THIS: Give a 3×3 elementary matrix named `E4` that multiplies the second row by $1/2$. Apply this to matrix `A3`, and store the result as a new variable `A4`.

In [37]:
```
# Put your answer here.
E4 = np.array([[1,0,0],[0,1/2,0],[0,0,1]])
A4 = E4@A3
sym.Matrix(A4)
```

Out[37]: $\begin{bmatrix} -1.0 & 2.0 & -4.0 \\ 1.0 & -1.0 & 3.5 \\ 0.0 & 3.0 & -3.0 \end{bmatrix}$

If the above are correct then we can combine the three operators `E2`, `E3`, and `E4` on the original matrix `A` to get `A4` as follows.

In [39]:
```
A = np.matrix([[3, -3,9], [2, -2, 7], [-1, 2, -4]])

sym.Matrix(E4@E3@E2@A)
```

Out[39]: $\begin{bmatrix} -1.0 & 2.0 & -4.0 \\ 1.0 & -1.0 & 3.5 \\ 0.0 & 3.0 & -3.0 \end{bmatrix}$

From previous assignments, we learned that we could string together a bunch of Elementary Row Operations to get matrix A into its Reduced Row Echelon Form. We later discussed that each Elementary Row Operation can be represented by a multiplication by an Elementary Matrix. Thus, the Gauss-Jordan Elimination (using Elementary Row Operations) can be represented as multiplication by a number of Elementary Matrices as follows:

$$E_n \dots E_3 E_2 E_1 A = RREF$$

If A reduces to the identity matrix (i.e. A is row equivalent to I, or $RREF = I$), then A has an inverse, and its inverse is just all of the Elementary Matrices multiplied together:

$$A^{-1} = E_n \dots E_3 E_2 E_1$$

DO THIS: Describe the Reduced Row Echelon Form of a square, invertible matrix.

In [42]:
```
# put your answer here
```

Example: Consider the matrix $A = \begin{bmatrix} 1 & 2 \\ 4 & 6 \end{bmatrix}$. Write A in python as an numpy array.

In [44]:
```
# Write your code here.
A= np.array([[1,2],[4,6]])
sym.Matrix(A)
```

Out[44]: $\begin{bmatrix} 1 & 2 \\ 4 & 6 \end{bmatrix}$

A can be reduced into an identity matrix using the following elementary operators

Operation	Elementary Matrix
Add -4 times row 1 to row 2.	$E_1 = \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}$
Add row 2 to row 1.	$E_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$
Multiply row 2 by $-\frac{1}{2}$.	$E_3 = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix}$

Example. Write the above matrices in python and use them to calculate A^{-1}. Verify your result.

In [47]:
```
# introduce E1, E2, E3 using numpy
E1= np.array([[1,0],[-4,1]])
E2= np.array([[1,1],[0,1]])
E3= np.array([[1,0],[0,-1/2]])
#Calculate A_inv by using E1,E2,E3
A_inv=  E3 @ E2 @ E1
#Check the fact that A_inv is the inverse of A
np.allclose(A_inv@A, [[1,0],[0,1]])
```

Out[47]: True

Example. Let T be the transformation $T(x) = Ax$, where $A = \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}$.

(a) Is A invertible?

(b) Using (a), find the vector x has $T(x) = \begin{bmatrix} -6 \\ -14 \end{bmatrix}$.

In [76]:
```
#put your answer here
#(a) yes A is invertable as determinant is != 0
A=np.array([[1,2],[-1,3]])
T_x=np.array([[-6],[-14]])
A_inv=np.linalg.inv(A)
x=A_inv@T_x
print(x)
```

```
[[ 2.]
 [-4.]]
```

Example. Create a function that will get a matrix A and do the following:
(a) Check if it's square and print a message.
(b) Calculate its inverse if possible.
(c) Return a message saying that the matrix is not invertible if it does not have an inverse.

In [79]:
```
#Complete the following code:
def inverse(A):
    A=np.array(A)
    if A.shape[0]!=A.shape[1]:
        print('the matrix is not square')
    else:
        try:
            A_inv=np.linalg.inv(A)
            return A_inv
        except Exception as e:
            print(f"The Matrix is not invertible!")
```

In [83]:
```
A=np.array([[1,2],[2,1]])
inverse(A)
```

Out[83]:
```
array([[-0.33333333,  0.66666667],
       [ 0.66666667, -0.33333333]])
```

In [85]:
```
B=np.array([[1,2],[1,2]])
inverse(B)
```

```
The Matrix is not invertible!
```

Before you close or submit this In-Class Assignment, please make sure of a few things:

- Did you save the file? `Ctrl + S` like everything else works!
- Is the file in correct format? You need to submit this file in `.pdf` format. To do so, `Ctrl + P` and `Save as pdf` (on Windows) or `command + P` in mac.
- If that does not work, please Google "ipynb to pdf converter" and that should do the job.
- Are the pictures/images rendering correctly in the `.pdf` format?
- If the images are not rendering, you could also use this command in a cell:

```
from IPython.display import display, Image

display(Image(filename="x.jpg", height=400, width=400))
```

THIS ONLY WORKS WITH JPGS.

- Please double check the contents of the file. One of the most common errors students make is submission of an empty file.

In [ ]: