00_Project_Intro

January 11, 2022

0.1 Problem 1: Forming Numpy Matrices

We will work quite heavily with numpy matrices. A numpy matrix can be created in a host of ways, but the most straight forward is to use the `np.array` initializer. In this case, each row of the matrix is initialized using an array and the matrix is an array of arrays. For example, the following matrix $ex_{mat} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ can be initialized as

```
ex_mat = np.array([ [1., 2., 3.],
                    [4., 5., 6.]])
```

where the array `[1., 2., 3.]` is the first row and the array `[4., 5., 6.]` is the second.

Create two additional matrices. Let I be a 3×3 identity matrix and R be a 3×3 rotation matrix about the z axis with a rotation of $\theta = \frac{\pi}{4}$.

[3]:
```
import numpy as np # Imports the numpy library and creates the alias np
from IPython.display import display # Used to display variables nicely in␣
 ↪Jupyter

# Print the example matrix
ex_mat = np.array([ [1., 2., 3.],
                    [4., 5., 6.]])
print("ex_mat = ")
display(ex_mat)

# Create an identity matrix
I = np.array([[1, 0, 0],[0,1,0],[0,0,1]])
print("I = ")
display(I)

# Create a 3x3 rotation matrix about the z axis
th = np.pi/4 # Angle of rotation
c = np.cos(th)
s = np.sin(th)
R = np.array([[c,s,0],[-s,c,0],[0,0,1]])
print("R = ")
display(R)
```

```
ex_mat =
```

```
array([[1., 2., 3.],
       [4., 5., 6.]])
```

```
I =
```

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

```
R =
```

```
array([[ 0.70710678,  0.70710678,  0.        ],
       [-0.70710678,  0.70710678,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

0.2 Problem 2: Multiplication

There are two multiplication operators that you can utilize. The first is the asterisk, `*`, and the second is the ampersand, `@`. Be careful as they produce several different results. Perform each multiplication, display the result, and answer the following question.

0.2.1 Question: What is the difference between * and @?

Answer: **Need to answer**

[4]:
```
# Multiply I and R together with the asterisk and display the results
asterisk_multiply = I*R
print('Result of asterisk multiplication:')
display(asterisk_multiply)

# Multiply I and R together with the ampersand and display the results
ampersand_multiply = I@R
print('Result of ampersand multiplication:')
display(ampersand_multiply)
```

```
Result of asterisk multiplication:
```

```
array([[ 0.70710678,  0.        ,  0.        ],
       [-0.        ,  0.70710678,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

```
Result of ampersand multiplication:
```

```
array([[ 0.70710678,  0.70710678,  0.        ],
       [-0.70710678,  0.70710678,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

0.3 Problem 3: Extracting values

Any numpy ndarray with more than a single row is treated as a matrix. Also note that numpy is zero indexed. The matrices are indexed with a double indexing, i.e., *ex_mat[0,0]* will return the top left element.

Extract and display the top row, middle column element from *ex_mat*

[6]:
```
# Get the element from the top row and middle column
val = ex_mat[0][1]
print('Element from ex_mat in top row and middle column:')
display(val)
```

```
Element from ex_mat in top row and middle column:
```

```
2.0
```

[ ]:
```

```