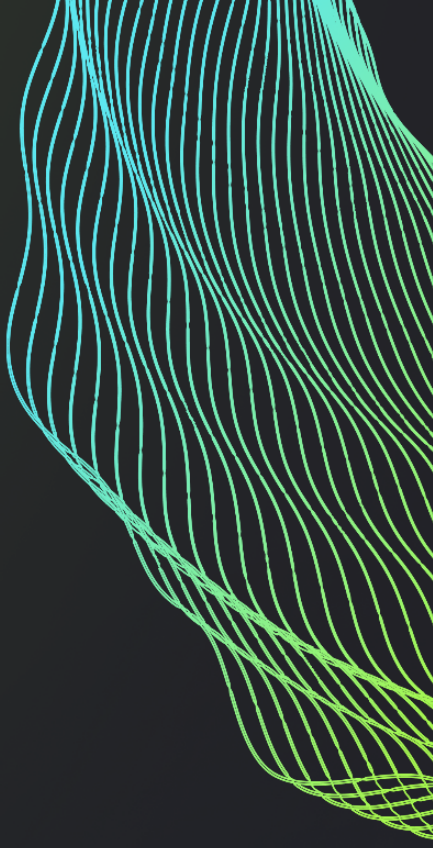
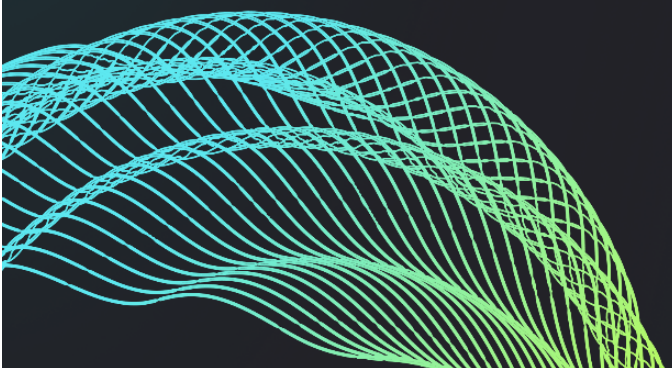


LET'S CODE



from ada

Lista de exercícios - Álgebra Linear

Questão 1.
Utilizando o módulo numpy, crie 5 matrizes e faça as operações de multiplicação entre todas elas. (podem ser de tamanhos variados, mas ao menos duas devem ser matrizes quadradas).

Questão 2.
Calcule a inversa e a transposta da matriz abaixo:

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

Questão 3.
Ache as as soluções do sistema linear a seguir, achando a inversa da matriz principal, e com uma matriz de resultados arbitrária(pode escolhido quaisquer valores para y0, y1, y2):

$$\begin{aligned} x_0 + 5x_1 + 3x_2 &= y_0 \\ 3x_0 + 2x_1 + x_2 &= y_1 \\ x_0 + x_1 + x_2 &= y_2 \end{aligned}$$

Questão 4.
Com base no sistema anterior, o resolva utilizando o método "solve" da biblioteca numpy.

Questão 5.
Dada a matriz principal dos exercícios 3 e 4, calcule o determinante da mesma utilizando o numpy.

Questão 6.
Utilizando numpy e as bibliotecas gráficas do python, mostre que se fizermos uma transformação linear em um vetor, suas bases canônicas também sofrem transformação linear.

Questão 7.
Cria uma matriz quadrada de ordem 4 e calcula os seus respectivos autovalores e autovetores.

Questão 8.
Com base no exercício 7, mostra a matriz diagonal de autovalores.

Questão 9.
Com base na aula sobre as cadeias de Markov, pegue o exemplo resolvido em aula e adicione mais uma loja de carros e crie as probabilidades faltantes da matriz de transição A (ou seja, crie porcentagens quaisquer). Após feito isso perceberá que a matriz de transição será de 16 elementos. Calcule qual é a tendência de devolução, sabendo que o vetor de estado inicial é do tipo:

```
w = np.array([[0], [1], [0], [0]])
```

Questão 10.
Utilizando as mesmas técnicas das aulas referentes a manipulação de imagens, escolha três fotos quaisquer, e as rotacione, crie suas cores negativas e também as decomponha em camadas RGB.

Resoluções ✅

Questão 1.

```
#Exercício 01
# utilizar np.array() para criar as matrizes e
realizar as multiplicações --> um exemplo criando
matrizes 2x2:

import numpy as np

a = np.array([[0,1],[1,2]])
b = np.array([[1,1],[1,5]])
c = np.array([[3,-1],[4,2]])
d = np.array([[8,1],[1,2]])
e = np.array([[2,2],[-3,1]])

print(a*b)
print()
print(c*e)
```

Questão 2.

```
#Exercício 02
# dada a matriz do exercício, calcular a inversa e transposta

import numpy as np

a = np.array([[1,0,0],[-1/2, 0, 1/2],[1/2, 1, -3/2]])

inv = np.linalg.inv(a)
tr = a.T

print(inv)
print()
print(tr)
```

Questão 3.

```
#Exercício 03

import numpy as np

a = np.array([[1,3,1],[5, 2, 1],[3, 1, 1]])
y = np.array([[1],[0],[1]])

inv = np.linalg.inv(a)
result = np.dot(inv,y)

print(result)
```

Questão 4.

```
res = np.linalg.solve(a,y)
print(res)
```

Questão 5.

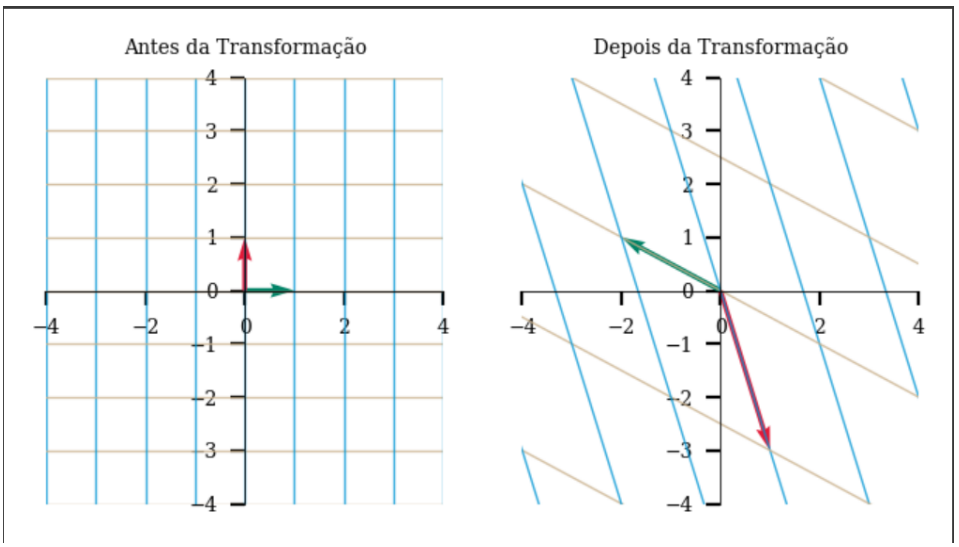
```
det = np.linalg.det(a)
print(det)
```

Questão 6.

```
from plot_helper import *
import numpy
```

```
baseA = numpy.array([(-2,1),(1,-3)])

plot_linear_transformation(baseA)
```



Obs: *Foi utlizado uma biblioteca gráfica pronta (Fonte: https://github.com/Alyssonmach/Algebra-Linear-com-Python/blob/master/Conte%C3%BAdo%20J%20-%20Transforma%C3%A7%C3%B5es%20Lineares/plot_helper.py)*

Questão 7.

```
import numpy as np

matriz = np.array([
                [1,0,2,1],
                [-3,4,-1,0],
                [-1,2,2,-2],
                [0,1,-1,1]
])


autovalores, autovetores = np.linalg.eig(matriz)

print(autovalores)
print()
print(autovetores)
```

Questão 8.

```
matrizDiagonal = np.diag(autovalores)

print(matrizDiagonal)
```

Questão 9.

Fonte de inspiração e utilização da fução pronta:
https://github.com/Alyssonmach/Algebra-Linear-com-Python/blob/master/Conte%C3%BAdo%20S%20-%20Cadeias%20de%20Markov/prog19_CadeiasDeMarkov.ipynb

```
import numpy as np

A = np.array([[50/100, 10/100, 20/100, 30/100],
              [10/100, 20/100, 30/100, 30/100],
              [10/100, 50/100, 20/100, 20/100],
              [30/100, 20/100, 30/100, 20/100]])

w = np.array([[0],
              [1],
              [0],
              [0]])
```

```
# função que realizará a mudança de estado até o ano a ser definido
def estado(matrizTransicao, vetorEstado, transicao):

    # estrutura de repetição for de 0 até (anos - 1)
    for indice in  range(0, transicao):

        # realiza o calculo definido no teorema acima
        vetorEstado = np.dot(matrizTransicao, vetorEstado)

    # retorna o vetor estado após sofrer todas as mudanças de estado
até o ano informado
    return vetorEstado
```

```
# vetor-estado após a 1° observação
print("O vetor-estado na 1° observação é: \n\n{}".format(estado(A, w,
transicao = 1)))
```

```
O vetor-estado na 1° observação é:

[[0.1]
 [0.2]
 [0.5]
 [0.2]]
```

```
# vetor-estado após a 2° observação
print("O vetor-estado na 2° observação é:
\n\n{}".format(estado(A, w, transicao = 2)))
```

```
O vetor-estado na 2° observação é:

[[0.23]
 [0.26]
 [0.25]
 [0.26]]
```

```
# vetor-estado após a 11° observação
print("O vetor-estado na 11° observação é: \n\n{}".format(estado(A, w,
transicao = 11)))
```

```
O vetor-estado na 11° observação é:

[[0.29039567]
 [0.21992943]
 [0.23694151]
 [0.25273339]]
```

```
# vetor-estado após a 11° observação
print("O vetor-estado na 11° observação é: \n\n{}".format(estado(A, w,
transicao = 20)))
O vetor-estado na 11° observação é:

[[0.29040097]
 [0.2199271 ]
 [0.23693803]
 [0.2527339 ]]
```

Questão 10.

Com base no script de exemplo do último vídeo, a resposta é individual e só deve ser replicado com fotos a escolha da pessoa.