Python para Ciências Exatas

Prof: Dennis

#01: Vetores e matrizes com numpy

https://wiki.python.org/moin/BeginnersGuide

https://numpy.org/

https://numpy.org/doc/stable/user/absolute_beginners.html

O que faremos nessa aula:

- Nesta aula veremos:
  - Como abrir um ambiente de programação python online para criar nossos programas em python sem instalar nada ou mesmo de um celular ou tablet;
  - Como fazer um código muito simples em python;
  - Como importar a biblioteca “numpy” que transforma o python em um poderoso software de matemática;
  - Como criar um vetor e mostra-lo;
  - Como criar uma matriz e mostra-la;
  - Como fazer operações com vetores e matrizes;

O que é Python?

- O Python é uma linguagem de programação, assim como o C, C++, C#, VisualBasic, Fortran,etc...
- No entanto ele possui 3 versões, sendo o "Python 3" o mais recente mas o "Python 2" ainda permanece em uso.
- De fato atualmente estamos na versão 3.9.6. Mas as versões do python 3 são compatíveis entre sí;
- Já um código em python 3 pode não funcionar em python 2 e vice versa.
- Python é uma linguagem essencialmente interpretada, de alto nível e de propósito geral;
- Para criar e executar um programa em python, é preciso um programa que irá decodificar e interpretar as instruções em linguagem de máquina para que seja executada por um processador;
- Existem inúmeras plataformas para se trabalhar com python, sendo as muito populares: JupiterNotebook, Spider, Pycharm;
- Neste curso, sempre que possível usaremos uma plataforma online para fazer nossos programas, o "google colab".

Por que Python?

- São as infindáveis e diversas bibliotecas gratuitas que fazem do python uma linguagem tão versátil.

- Para aplicação em exatas podemos citar rapidamente:
  - Numpy – Vetores, matrizes, calculo numérico em geral;
  - Matplotlib – Gráficos
  - Scipy – Mais aplicações científicas e de calculo numérico
  - Simpy – Matemática simbólica
  - Control – Controle Clássico

- Sem contar bibliotecas para desenvolvimento de Software, Web, Games, IA, etc...

Primeiríssimo código

- Depois dessa breve explicação, vamos abrir um interpretador e fazer nosso primeiro programa em python.
- Vamos usar o "google colab", assim não precisaremos baixar nem instalar nada;
- Acesse: https://colab.research.google.com/
- Se você estiver logado em uma conta "google", vai abrir uma janela. Nela clique em "novo notebook";
- Digite o código, sim, só uma linha:

```
print("Hello, I'm Python!")
```

Agora, vamos aos vetores

- O python permite trabalhar com listas de números sem usar nenhum biblioteca;
- No entanto é usando a biblioteca numpy que realmente transformamos o python em um software de matemática, como o Matlab®, o Scilab ®, o Geogebra ®, entre outras mas com a vantagem de ser uma linguagem de programação de propósito geral.
- Vamos fazer nosso primeiro programa com o numpy. Abra um novo notebook ou acrescente uma célula no último que usamos;
- Digite o código a seguir:

```
import numpy as np

vet1 = np.array([1, 2, 3])
print("vetor 1 =", vet1)

vet2 = np.array([1.5, 2, 4])
print("vetor 2 =", vet2)

vet_s=vet1+vet2
print("vetor 3 =", vet_s)
```

https://numpy.org/doc/stable/reference/generated/numpy.array.html

Agora, vamos aos vetores

- Vamos entender a função de cada linha:

```
import numpy as np   → importa a biblioteca numpy com o apelido de np

vet1 = np.array([1, 2, 3])   → cria o vetor [1 2 3] e guarda na variável "vet1"
print("vetor 1 =", vet1)    → mostra a variável "vet1"

vet2 = np.array([1.5, 2, 4]) → cria o vetor [1.5 2 4] e guarda na variável "vet2"
print("vetor 2 =", vet2)     → mostra a variável "vet2"

vet_s=vet1+vet2    → soma os vetores e guarda em "vet_s"
print("vetor 3 =", vet_s)  → mostra a variável "vet_s"
```

https://numpy.org/doc/stable/reference/generated/numpy.array.html

dos vetores para as matrizes:

- A numpy trata matrizes de forma semelhante aos vetores, vamos ver como defini-las;
- Abra um novo notebook ou acrescente uma célula no último que usamos;
- Digite o código a seguir:

```
import numpy as np
```
$\longrightarrow$ *importa a biblioteca numpy com o apelido de np*

```
mat1 = np.array([[1, 2], [3, 4]])
```
$\longrightarrow$ *cria a matriz* $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ *e guarda na variável "mat1"*

```
print("matriz 1 = \n", mat1)
```
$\longrightarrow$ *mostra a variável "mat1". O "\n" pula a linha e deixa a matriz alinhada.*

```
mat2=2*mat1-2
```
$\longrightarrow$ *Calcula a matriz* $2 \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - 2$ *e guarda na variável "mat2"*

```
print("matriz 2 = \n", mat2)
```
$\longrightarrow$ *mostra a variável "mat2".*

Produto de vetores

- Por padrão, quando se usa o operador de multiplicação "*", o python irá realizar o produto termo a termo entre os elementos do vetor ou matriz;
- Se quisermos fazer o produto escalar entre dois vetores, devemos usar a função "np.dot".
- Se quisermos o produto vetorial entre dois vetores devemos usar a função "np.cross".
- Vamos ver um código de exemplo com essas operações;

Produto de vetores: exemplo

- Abra um novo notebook ou acrescente uma célula no último que usamos;
- Digite o código a seguir:

```
import numpy as np
```
$\longrightarrow$ *importa a biblioteca numpy com o apelido de np*

```
v1 = np.array([1, 2, 3])
v2 = np.array([0, 2., 4.])
```
$\rightarrow$ *cria o vetor* $[1\ 2\ 3]$ *e guarda na variável* "$v1$"

$\rightarrow$ *cria o vetor* $[0\ 2\ 4]$ *e guarda na variável* "$v2$"

```
vprod=v1*v2
```
$\longrightarrow$ *vprod será o produto termo a termo:* "[0 4 12]"

```
vdot=np.dot(v1,v2)
```
$\longrightarrow$ *vdot será o produto escalar:* 1*0+2*2+3*4=16

```
vcross=np.cross(v1,v2)
```
$\longrightarrow$ *vcros será o produto vetorial:*

$$\begin{vmatrix} a_x & a_y & a_z \\ 1 & 2 & 3 \\ 0 & 2 & 4 \end{vmatrix} = [2\ \text{-}4\ \ 2]$$

```
print("v1*v2=", vprod)
print("v1 . v2=", vdot)
print("v1 x v2=", vcross)
```
$\longrightarrow$ *mostra os tres vetores resultantes das 3 operações*

Produto de matrizes e vetores

- Assim como para os vetores, quando se usa o operador de multiplicação "*", o python irá realizar o produto termo a termo entre os elementos das matrizes;
- Se quisermos fazer o produto de matrizes, devemos usar a função "np.dot".
- Lembrando o produto de matrizes: "primeira linha vezes primeira coluna, ..."

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 2 \cdot -1 & 1 \cdot 0 + 2 \cdot 2 \\ 3 \cdot 1 + 4 \cdot -1 & 3 \cdot 0 + 4 \cdot 2 \end{bmatrix} = \begin{bmatrix} -1 & 4 \\ -1 & 8 \end{bmatrix}$$

$$[2x2] \cdot [2x2] \quad = \quad [2x2] \cdot [2x2] = [2x2]$$

- Uma alternativa é criar a matriz com a função np.matrix() ao invés de np.array().
- As variáveis do tipo np.matrix, realizam o produto de matrizes na operação com "*".
- Vamos um código de exemplo com essas operações;

- PS: em python o "elevado" é "**": $z = x^y \leftrightarrow z = x ** y$

Produto de matrizes: exemplo

- Abra um novo notebook ou acrescente uma célula no último que usamos;
- Digite o código a seguir:

```
import numpy as np
```
$\longrightarrow$ *importa a biblioteca numpy com o apelido de np*

```
M1 = np.array([[1, 2], [3, 4]])
```
$\longrightarrow$ *cria a matriz* $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$ *e guarda na variável "M1"*

```
M2 = np.array([[1, 0], [-1, 2]])
```
$\longrightarrow$ *cria a matriz* $\begin{bmatrix}1 & 0\\-1 & 2\end{bmatrix}$ *e guarda na variável "M2"*

```
Mprod=M1*M2
```
$\longrightarrow$ *vprod será o produto termo a termo:* $\begin{bmatrix}1 & 0\\-3 & 8\end{bmatrix}$

```
Mdot=np.dot(M1,M2)
```
$\longrightarrow$ *vdot será o produto escalar:* $\begin{bmatrix}-1 & 4\\-1 & 8\end{bmatrix}$

```
print("M1*M2 = \n", Mprod)
print("M1 . M2 = \n", Mdot)
```
$\longrightarrow$ *mostra as matrizes resultantes das duas operações*

```
Mquad1=M1**2
```
$\longrightarrow$ *eleva um np.array ao quadrado, termo a termo*

```
M3 = np.matrix([[1, 2], [3, 4]])
```
$\longrightarrow$ *cria a matriz* $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$ *na forma de np.matrix e guarda na variável "M3"*

```
Mquad3=M3**2
```
$\longrightarrow$ *eleva uma np.matrix ao quadrado, com o produto de matrizes*

```
print("Mquad1= = \n", Mquad1)
print("Mquad2= = \n", Mquad3)
```
$\longrightarrow$ *mostra as matrizes resultantes das duas operações de elevado*

Outras operações com matrizes

- Podemos também calcular a transporta de uma matriz ou vetor usando np.transpose()
  - A transposta troca as linhas pela colunas

- Ou calcular a matriz inversa usando np.linalg.inv(M1)
  - A inversa é a matriz que quando multiplicada pela matriz original resulta na matriz identidade.

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

- Vamos um código de exemplo com essas operações;

Outras operações com matrizes

- Abra um novo notebook ou acrescente uma célula no último que usamos;
- Digite o código a seguir:

```
import numpy as np
```
$\longrightarrow$ *importa a biblioteca numpy com o apelido de np*

```
M1 = np.array([[1, 2], [3, 4]])
```
$\longrightarrow$ *cria a matriz* $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ *e guarda na variável "M1"*

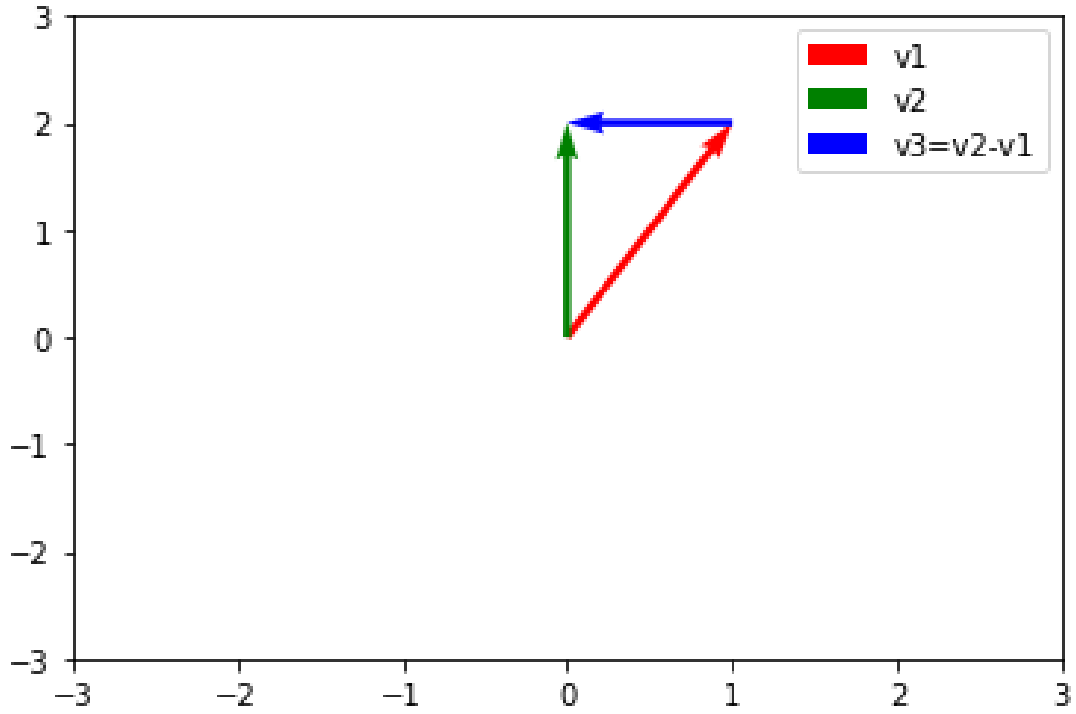
```
M1_T=np.transpose(M1)
```
$\longrightarrow$ *calcula a transposta de M1*

```
M1_inv=np.linalg.inv(M1)
```
$\longrightarrow$ *calcula a inversa de M1*

```
print("M1= = \n", M1)
print("M1 transposta = \n", M1_T)
print("M1 inversa = \n", M1_inv)
```
$\longrightarrow$ *mostra a matriz original e as matrizes resultantes das duas operações*

Extra: mostrando vetores

- O código abaixo mostra como plotar 3 vetores. $v1$ e $v2$ em relação a origem e $v3$ em relação à $v1$:




```
import numpy as np
import matplotlib.pyplot as plt

v1 = np.array([1, 2])
v2 = np.array([0, 2])
v3 = v2-v1;
origem = np.array([0, 0])

plt.figure()
plt.quiver(origem[0],origem[1],v1[0],v1[1], angles='xy', scale_units='xy', scale=1, color=['r'])
plt.quiver(origem[0],origem[1],v2[0],v2[1], angles='xy', scale_units='xy', scale=1, color=['g'])
plt.quiver(v1[0],v1[1],v3[0],v3[1], angles='xy', scale_units='xy', scale=1, color=['b'])
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.legend(['v1','v2','v2-v1'])
```

Exercício 1:

- Dados os vetores abaixo, calcule e mostre seu produto escalar e vetorial.

Dados:

$$\vec{a} = 3\,\overrightarrow{a_x} - 2\,\overrightarrow{a_y} + 5\,\overrightarrow{a_z}$$

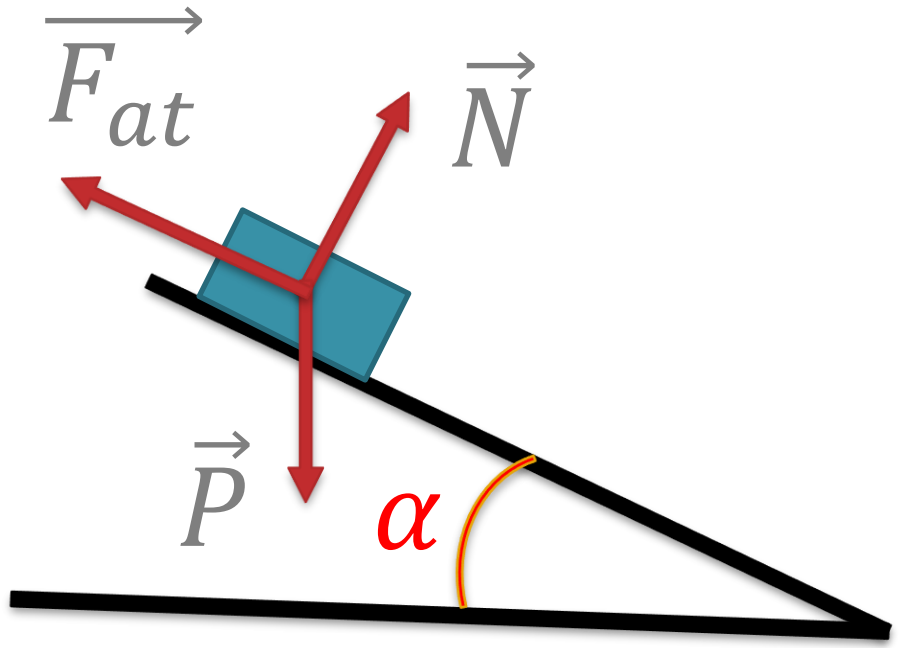
$$\vec{b} = 2\,\overrightarrow{a_x} + 9\,\overrightarrow{a_z}$$

Exercício 2:

- Considere um corpo sobre um plano inclinado, conforme figura abaixo. Escreva um código para calcular a força resultado sobre esse corpo.

Dados:

$$\vec{P} = -10N\ \overrightarrow{a_y}$$
$$\vec{N} = 3N\ \overrightarrow{a_x} + 9N\ \overrightarrow{a_y}$$
$$\overrightarrow{F_{at}} = -2N\ \overrightarrow{a_x} + 1N\ \overrightarrow{a_y}$$



Exercício 3:

- Dadas as matrizes e vetores abaixo, calcule a equação matricial e encontre a matriz A:

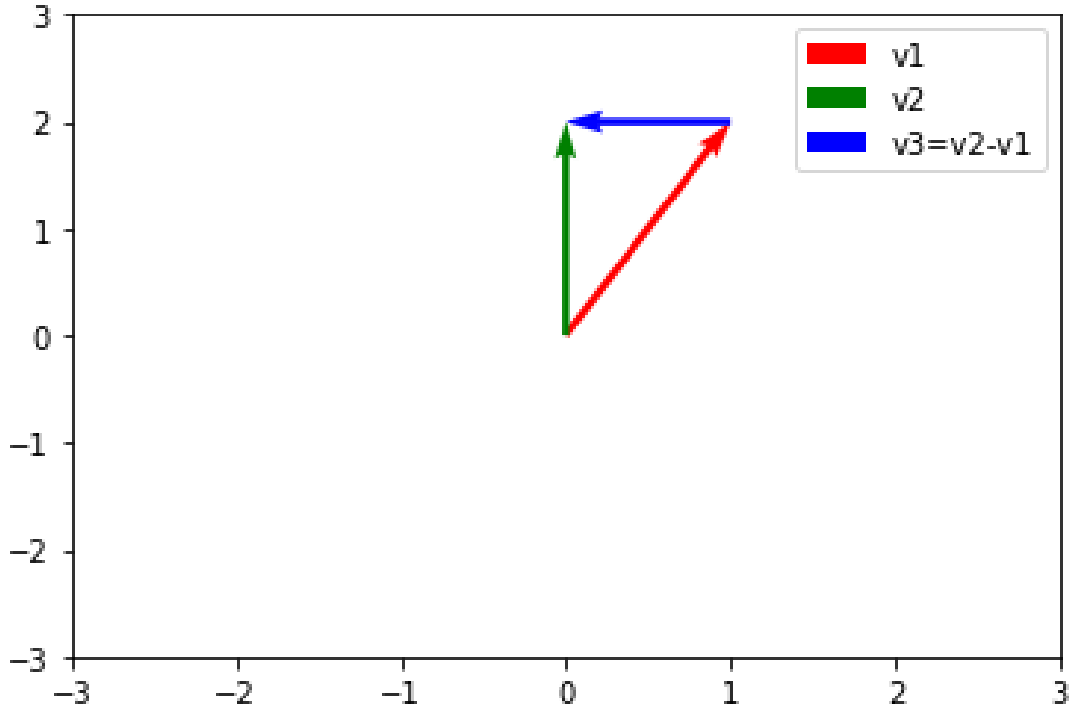
$$M = \begin{bmatrix} 1 & 2 \\ -1 & -2 \end{bmatrix}$$

$$\overrightarrow{v1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\overrightarrow{v2} = \begin{bmatrix} 3 & 4 \end{bmatrix}$$

$$A = 2 \cdot M + v1 \cdot v2$$

```python
import numpy as np

M1 = np.array([[1, 2], [3, 4]])
M2 = np.array([[1, 0], [-1, 2]])

Mprod=M1*M2
Mdot=np.dot(M1,M2)

print("M1*M2 = \n", Mprod)
print("M1 . M2 = \n", Mdot)

Mquad1=M1**2
M3 = np.matrix([[1, 2], [3, 4]])

Mquad3=M3**2

print("Mquad1= = \n", Mquad1)
print("Mquad2= = \n", Mquad3)
```



Python para exatas #01: Vetores e Matrizes com numpy