

Cheat Sheet Numpy/Matplotlib Matlab

Guía apoyo

Pablo Pizarro R.

pablo.pizarro@ing.uchile.cl

Numpy y Matplotlib son librerías en Python que sirven para realizar operaciones matemáticas tanto de manejo de matrices y vectores como de representación de gráficos, respectivamente.

Para instalar dichas librerías en Linux sólo basta ingresar en el terminal los siguientes comandos:

```
1 sudo pip install numpy
2 sudo pip install matplotlib
```

En caso de estar utilizando Windows, a partir de las versiones 2.7.9 para python 2 y 3.4 para python 3, pip viene incluido y se ejecuta con las instrucciones (estando python en el respectivo PATH):

```
1 python -m pip install numpy
2 python -m pip install matplotlib
```

Finalmente para acceder a las librerías dentro de Python se deben importar haciendo uso de las siguientes instrucciones:

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

A continuación, comandos básicos de numpy y matlab:

■ Nan e Infinito:

```
1 Python: Nan = np.nan; Inf = np.inf
2 Matlab: _nan = NaN; _inf = Inf
```

■ Crear matriz de unos.

```
1 Python: np.ones((2,3))
2     >> [[1, 1, 1],
3         [1, 1, 1]]
4 Matlab: ones(2,3)
5     >> 1 1 1
6         1 1 1
```

■ Crear matriz de ceros.

```
1 Python: np.zeros((2,3))
2     >> [[0, 0, 0],
3          [0, 0, 0]]
4 Matlab: zeros(2,3)
5     >> 0 0 0
6          0 0 0
```

■ Ponderar matriz.

```
1 Python: np.ones((2,3)) * 5
2     >> [[5, 5, 5],
3          [5, 5, 5]]
4 Matlab: ones(2,3) .* 5
5     >> 5 5 5
6          5 5 5
```

■ Tamaño matriz.

```
1 Python: np.zeros((2,3)).shape
2     >> (2, 3)
3 Matlab: size(zeros(2,3))
4     >> 2 3
```

■ Accesos.

```
1 Python: matriz[a,b] o bien matriz[:,b]
2 Matlab: matriz(a,b) o bien matriz(:,b)
```

■ Chequeo contra NaN.

```
1 Python: np.isnan(np.ones((2,3)) * np.nan)
2     >> [[True, True, True],
3          [True, True, True]]
4 Matlab: isnan([1 1 NaN 1 1])
5     >> 0 0 1 0 0
```

■ Chequeo contra Inf.

```
1 Python: np.isinf(np.ones((2,3)) * np.inf)
2     >> [[True, True, True],
3          [True, True, True]]
4 Matlab: isinf([1 1 Inf 1 1])
5     >> 0 0 1 0 0
```

■ Elementos distintos de cero.

```
1 Python: np.nonzero(np.ones((2,3)))
2     >> (array([0, 0, 0, 1, 1, 1]), array([0, 1, 2, 0, 1, 2])) ...
3 Matlab: [Inf Inf 0; 0 Inf Inf]
4     >> 1 1 0
5          0 1 1
```

- Flip up-down y left-right.

```

1 Python: A = np.ones((2,2)) ; A[0,1] = 2; A[1,0] = 3; A[1,1] = 4;
2         np.flipud(A)
3         >> [[3, 4],
4             [1, 2]]
5         np.fliplr(A)
6         >> [[2, 1],
7             [4, 3]]
8 Matlab: flipud([1 2; 3 4])
9         >> 3 4
10          1 2
11          fliplr([1 2; 3 4])
12         >> 2 1
13          4 3

```

- Promedio matriz.

```

1 Python: A = np.ones((2,2)) ; A[0,1] = 2; A[1,0] = 3; A[1,1] = 4;
2         np.mean(A) = 2.5
3 Matlab: mean2([1 2; 3 4]) = 2.5

```

- Gradiente matriz.

```

1 Python: A = np.matrix([[5, 2, 9],[8, 5, 1],[2, 1, 8]])
2 [U,V] = np.gradient(A)
3         >> U = [
4             [ 3.  3.  8. ]
5             [1.5  0.5  0.5]
6             [ 6.  4.  7. ]
7             ]
8         >> V = [
9             [ 3.  2.  7. ]
10            [ 3.  3.5  4. ]
11            [ 1.  3.  7. ]
12            ]
13 Matlab: [U, V] = gradient([5 2 9 ; 8 5 1; 2 1 8])
14         U =
15             3.0000  2.0000  7.0000
16             3.0000  3.5000  4.0000
17             1.0000  3.0000  7.0000
18         V =
19             3.0000  3.0000  8.0000
20             1.5000  0.5000  0.5000
21             6.0000  4.0000  7.0000

```

- Mostrar matriz.

```

1 Python: A = np.matrix([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
2         pl.matshow(A)
3         pl.show()
4 Matlab: A = [1 2 3; 4 5 6; 7 8 9]
5         surf(A) # o bien mesh(A)

```

■ Trasponer matriz.

```
1 Python: (a,b) = ([0, 0, 1, 1, 2, 2],[1, 2, 1, 2, 1, 2]) ; ...
2     pl.transpose((a,b))
3     >> [
4         [0 1]
5         [0 2]
6         [1 1]
7         [1 2]
8         [2 1]
9         [2 2]
10    ]
11 Matlab: transpose([0 0 1 1 2 2; 1 2 1 2 1 2])
12     >> 0 1
13         0 2
14         1 1
15         1 2
16         2 1
17         2 2
```

■ Quiver.

```
1 # Asumiendo vectores U y V del gradiente.
2 Python: pl.figure() ; pl.quiver(U,V) ; pl.show()
3 Matlab: quiver(U,V)
```