

TEMA 5: Vectores y Matrices

Dpto. de Matemática Aplicada a la Ingeniería Aeroespacial

Informática, 1^{er} semestre
2022 – 2023

Profesor: Juan Antonio Hernández Ramos (juanantonio.hernandez@upm.es)

Coordinador: Javier de Vicente Buendía (fj.devicente@upm.es)

Colaborador: Víctor Javier Llorente Lázaro (victorjavier.llorente@upm.es)

• Listas en Python (RECORDATORIO)

Las listas son un tipo de dato que permite almacenar un conjunto arbitrario de datos:

```
x = [1, 2, 3, 4]
x = [i + 1 for i in range(4)]
x = list('1234') # ["1", "2", "3", "4"]
x = [1, 'Hola', 3.67, [1, 2, 3]]
```

- Son ordenadas.
- Pueden estar formadas por tipos arbitrarios.
- Son indexadas con `[i]`. El índice del 1^{er} elemento es `[0]`.
- Se pueden anidar: meter una dentro de la otra.
- Son mutables: sus elementos pueden ser modificados.
- Son dinámicas: se pueden añadir o eliminar elementos.

```
# Listas
x = [1, 2, 3, ['p', 'q', [5, 6, 7]]]
print(x[3][0])
print(x[3][2][0])
print(x[3][2][2])
```

p
5
7

VARIABLES TIPO `<list>` HACEN UN POSIBLE USO PARA VECTORES Y MATRICES

Otros lenguajes de programación, e.g. Fortran, vectores y matrices son almacenados en variables tipo `<array>`.

¿QUÉ SUCEDERÁ AL OPERAR CON ELLOS?

- Vectores**

En matemáticas un vector es un elemento de un espacio vectorial, por ejemplo $v = (1, 2, 3) \in \mathbb{R}^3$. Este vector en Python podría ser representado de la siguiente manera:

$v = [1, 2, 3]$

- Matrices**

En Python, las matrices se forman anidando listas donde cada lista contenga la fila de la matriz, es decir,

$m = [[\text{Fila 1}], [\text{Fila 2}], [\text{Fila 3}], \dots, [\text{Fila n}]]$

$$m = \begin{pmatrix} 8 & 14 & -6 \\ 12 & 7 & 4 \\ -11 & 3 & 21 \end{pmatrix} \Rightarrow m = [[8, 14, -6], [12, 7, 4], [-11, 3, 21]]$$

Acceso a los elementos de los anteriores vectores y matrices:

$(v[0], v[1], v[2])$

$$\begin{pmatrix} m[0][0] & m[0][1] & m[0][2] \\ m[1][0] & m[1][1] & m[1][2] \\ m[2][0] & m[2][1] & m[2][2] \end{pmatrix}$$

$$m[i][j] = m[i, j]$$

numpy: biblioteca de funciones matemáticas de alto nivel para operar con vectores y matrices. (<https://numpy.org/doc/stable/index.html>)

- Operaciones numéricas

Supongamos dos vectores: $v_1 = (1, 2, 3) \in \mathbb{R}^3$ y $v_2 = (4, 5, 6) \in \mathbb{R}^3$ y se genera otro vector $v_3 = v_1 + v_2 = (5, 7, 9) \in \mathbb{R}^3$. Esta operación matemática trasladada a Python daría como resultado:

```
# Operaciones con listas
v1 = [1, 2, 3]
v2 = [4, 5, 6]
v3 = v1 + v2
print(v3)
```

v_1, v_2, v_3 son de tipo `<list>`

[1, 2, 3, 4, 5, 6]

¿Cómo operamos en Python?

```
# Operaciones con vectores
import numpy as np
v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
v3 = v1 + v2
print(v3)
```

v_1, v_2, v_3 son de tipo `<Array of int32>`

[5, 7, 9]

IMPORTANTE: operadores `+`, `-`, `*`, y `/` sobre arrays realizan las operaciones elemento a elemento.

- Operaciones numéricas

```
# Producto escalar
import numpy as np
v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
p = np.dot(v1, v2)
# p = v1.dot(v2)
print(p)
```

32

```
# Producto vectorial
import numpy as np
v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
v3 = np.cross(v1, v2)
print(v3)
```

[-3, 6, -3]

```
# Producto matricial
import numpy as np
m1 = np.array([[8, 14, -6], [12, 7, 4], [-11, 3, 21]])
m2 = np.array([[5, 1, 3], [1, 1, 1], [1, 2, 1]])
m3 = np.matmul(m1, m2)
print(m3)
```

$\begin{bmatrix} 48 & 10 & 32 \\ 71 & 27 & 47 \\ -31 & 34 & -9 \end{bmatrix}$

- Operaciones sobre arrays

`numpy.reshape(<array>, (filas, columnas))`

```
# Reformular array
import numpy as np
v = np.array([2.0, 4.3, 5.2, 9.3, 6.8, 5.7])
m = np.reshape(v, (2, 3))
# m = v.reshape(2, 3)
print(m)
```

Tipo <Array of float64>



```
[[2.  4.3 5.2]
 [9.3 6.8 5.7]]
```

`numpy.shape(<array>)`

```
# Tamaño array
import numpy as np
m = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
tam = np.shape(m)
# tam = m.shape
print(tam)
```

Tipo <tuple>



```
(2, 4)
```

- Similar a <list>.
- No es mutable.

- Operaciones sobre arrays

`numpy.arange(inicial, final, paso{Opcional})`

```
# Vector espaciado
import numpy as np
a = np.arange(5)
b = np.arange(1, 9, 2)
print(a)
print(b)
```

```
[0, 1, 2, 3, 4]
[1, 3, 5, 7]
```


`numpy.linspace(inicial, final, nº puntos)`

```
# Espacio lineal
import numpy as np
a = np.linspace(0, 1, 6)
print(a)
```

```
[0., 0.2, 0.4, 0.6, 0.8, 1.]
```

- Operaciones sobre arrays


```
# Matrices importantes
import numpy as np
# Matriz de unos
U = np.ones((3, 3))
print(U)
# Matriz de ceros
C = np.zeros((3, 3))
print(C)
# Matriz identidad
I = np.eye(3)
print(I)
# Matriz diagonal
v = np.array([1.0, 2.0, 3.0])
D = np.diag(v)
print(D)
```




`[[1. 1. 1.]`
`[1. 1. 1.]`
`[1. 1. 1.]]`



`[[0. 0. 0.]`
`[0. 0. 0.]`
`[0. 0. 0.]]`



`[[1. 0. 0.]`
`[0. 1. 0.]`
`[0. 0. 1.]]`



`[[1. 0. 0.]`
`[0. 2. 0.]`
`[0. 0. 3.]]`

- **Secciones de arrays**

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

`variable[inicial:final:paso{Opcional}]`

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```



[3.2 1.3 6.5 4.2 5.4]

- **Secciones de arrays**

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

`variable[inicial:final:paso{Opcional}]`

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```



[3.2 1.3 6.5 4.2 5.4]

- **Secciones de arrays**

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

`variable[inicial:final:paso{Opcional}]`

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```



[3.2 9.1 4.2]

- Secciones de arrays

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

`variable[inicial:final:paso{Opcional}]`

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```



[6.5 1.6]

- Secciones de arrays

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

variable[inicial:final:paso{Opcional}]

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```



[4.2 1.6 5.4 9.]

- Secciones de arrays

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

1. Vectores

`variable[inicial:final:paso{Opcional}]`

```
# Secciones de un vector
import numpy as np
v = np.array([3.2, 6.3, 1.3, 9.1, 6.5, 4.1, 4.2, 1.6, 5.4, 9.0])
print(v[0:9:2])
print(v[ : :2])
print(v[ :7:3])
print(v[4: :3])
print(v[6:  ])
print(v[ :5  ])
```

[3.2 6.3 1.3 9.1 6.5]

- **Secciones de arrays**


Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

2. Matrices

variable[inicial:final:paso{Opcional}, inicial:final:paso{Opcional}]

```
# Secciones de una matriz
import numpy as np
v = (np.array([3, 6, 1, 9, 6, 4, 4, 6, 5, 9,
              2, 4, 6, 5, 9, 6, 1, 3, 2, 8]))
m = np.reshape(v, (5, 4))
print(m[0:3,0:2])
print(m[3: ,1: ])
print(m[3  , : ])
```



[3	6]
[6	4]
[5	9]

- **Secciones de arrays**


Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

2. Matrices

variable[inicial:final:paso{Opcional}, inicial:final:paso{Opcional}]

```
# Secciones de una matriz
import numpy as np
v = (np.array([3, 6, 1, 9, 6, 4, 4, 6, 5, 9,
              2, 4, 6, 5, 9, 6, 1, 3, 2, 8]))
m = np.reshape(v, (5, 4))
print(m[0:3,0:2])
print(m[3: ,1: ])
print(m[3  , : ])
```



`[[5 9 6]
 [3 2 8]]`

- **Secciones de arrays**

Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

2. Matrices

variable[**inicial:final:paso**{Opcional}, **inicial:final:paso**{Opcional}]

```
# Secciones de una matriz
import numpy as np
v = (np.array([3, 6, 1, 9, 6, 4, 4, 6, 5, 9,
               2, 4, 6, 5, 9, 6, 1, 3, 2, 8]))
m = np.reshape(v, (5, 4))
print(m[0:3,0:2])
print(m[3: ,1: ])
print(m[3  , : ])
```

[6 5 9 6]

- **Secciones de arrays**

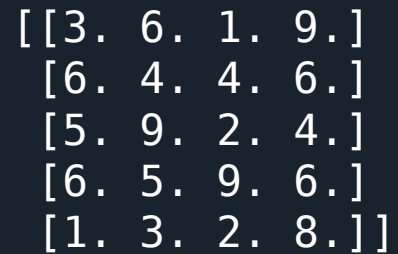
Es posible extraer un patrón ordenado de elementos de un array. Esta operación permite, por ejemplo:

- Extraer rangos de elementos de un vector
- Extraer filas o columnas de una matriz
- Extraer submatrices de una matriz

2. Matrices

```
# Asignación matriz
import numpy as np
m = np.zeros((5, 4))
m[0,:] = np.array([3, 6, 1, 9])
m[1,:] = np.array([6, 4, 4, 6])
m[2,:] = np.array([5, 9, 2, 4])
m[3,:] = np.array([6, 5, 9, 6])
m[4,:] = np.array([1, 3, 2, 8])
print(m)
```

Necesario inicializar la matriz con ceros, por ejemplo, para poder asignar secciones a la matriz.



```
[[3.  6.  1.  9.]
 [6.  4.  4.  6.]
 [5.  9.  2.  4.]
 [6.  5.  9.  6.]
 [1.  3.  2.  8.]
```

EJERCICIOS PROPUESTOS

1. Sea $\alpha_i \in \mathbb{R}$ para todo $i = 1, 2, \dots, n$, escribir la función `Vandermonde(alpha)` que devuelva la matriz $V_{ij} = \alpha_i^{j-1}$ para todo i y j .
2. Sea una matriz $\mathbf{M} \in \mathbb{R}^{n \times n}$, escribir la función `Traza(M)` que devuelva el valor de la traza de la matriz.
3. Sea un vector $\mathbf{v} \in \mathbb{R}^n$ y una matriz $\mathbf{M} \in \mathbb{R}^{n \times n}$, escribir la función `Cuadratica(v,M)` que devuelva la forma cuadrática de la matriz \mathbf{M} , es decir $q = \mathbf{v}^T \mathbf{M} \mathbf{v}$.
4. Sea $V = \{\mathbf{v}_i | i = 1, 2, \dots, n\} \in \mathbb{R}^n$, escribir una función `EsBase(*vectores)` que determine si dichos vectores forman una base en \mathbb{R}^n . **Pista:** desde numpy, llame a la librería de algebra lineal `linalg` e importe la operación `det` para calcula el determinante de la matriz forma por columnas los vectores \mathbf{v}_i .
5. Se dice que una matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$ es estrictamente diagonal dominante (EDD) por filas cuando se satisface:

$$|A_{ii}| > \sum_{j=1, j \neq i}^n |A_{ij}|, \quad 1 \leq i \leq n.$$

Determina mediante una función si dada una matriz es EDD por filas.