Andrés Merino • Semestre 2024-1

1. MATRICES

Una manera para trabajar con matrices en Python, es utilizando la librería `sympy`, con la siguiente línea de comandos:

```
[In]: from sympy import *
```

Con esto, podemos almacenar y visualizar una matriz de la siguiente manera:

```
[In]: A = Matrix([[1, 2, 3], [4, 5, 6], [-1, 0, 2]])
      display(A)
```

[Out]: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ -1 & 0 & 2 \end{bmatrix}$

Cuando se quiera visualizar un único objeto, basta con escribir sunombre al final de la celda:

```
[In]: A
```

[Out]: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ -1 & 0 & 2 \end{bmatrix}$

Para obtener un elemento de la matriz realizamos lo siguiente:

```
[In]: # Entrada de la primera fila y primera columna
      A[0, 0]
```

[Out]: 1

```
[In]: # Entrada de la segunda fila y primera columna
      A[1, 0]
```

[Out]: 4

Para obtener una fila o columna de una matriz realizamos lo siguiente:

```
[In]: # Primera fila
      A[0, :]
```

[Out]: $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

```
[In]: # Segunda fila
      A[1, :]
```

[Out]: $\begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$

[In]:
```python
# Primera columna
A[:, 0]
```

[Out]: $\begin{bmatrix} 1 \\ 4 \\ -1 \end{bmatrix}$

[In]:
```python
# Segunda columna
A[:, 1]
```

[Out]: $\begin{bmatrix} 2 \\ 5 \\ 0 \end{bmatrix}$

Otra forma válida es:

[In]:
```python
# Primera fila
A.row(0)
```

[Out]: $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

[In]:
```python
# Segunda columna
A.col(1)
```

[Out]: $\begin{bmatrix} 2 \\ 5 \\ 0 \end{bmatrix}$

Para comparar la igualdad entre dos matrices, podemos usar el operador ==:

[In]:
```python
# Definimos las matrices
A = Matrix([[1, 0], [0, 1]])
print("A:")
display(A)
B = Matrix([[1, 2], [3, 4]])
print("B:")
display(B)
C = Matrix([[1, 2]])
print("C:")
display(C)
# Comparamos las matrices
print("Es A igual a B: ", A == B)
print("Es B igual a C: ", B == C)
print("Es A igual a A: ", A == A)
```

[Out]:
```
A:
```
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
```
B:
```
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$
```
C:
```
$$\begin{bmatrix} 1 & 2 \end{bmatrix}$$
```
Es A igual a B:  False
Es B igual a C:  False
Es A igual a A:  True
```

2. Operaciones de matrices

Podemos realizar las distintas operaciones de matrices de la siguiente manera:

[In]:
```python
# Suma de matrices
A+B
```

[Out]:
$$\begin{bmatrix} 2 & 2 \\ 3 & 5 \end{bmatrix}$$

[In]:
```python
# Multiplicación por un escalar
2*B
```

[Out]:
$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

[In]:
```python
# Transpuesta
B.T
```

[Out]:
$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

2.1 Multiplicación de matrices

Definamos algunas matrices:

[In]:
```python
A = Matrix([[1, 2, 3], [3, 2, 1], [1, -1, 0]])
print("A:")
display(A)
B = Matrix([[1, 0, 2], [2, 1, 0], [0, 1, 0]])
print("B:")
display(B)
```

[Out]: A:

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

B:

$$\begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

La multiplicación es:

[In]:
```
A*B
```

[Out]:

$$\begin{bmatrix} 5 & 5 & 2 \\ 7 & 3 & 6 \\ -1 & -1 & 2 \end{bmatrix}$$

Podemos elevar una matriz a un exponente con el operador **:

[In]:
```
print('A^0:')
display(A**0)
print('A^1:')
display(A**1)
print('A^2:')
display(A**2)
```

[Out]: A^0:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A^1:

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

A^2:

$$\begin{bmatrix} 10 & 3 & 5 \\ 10 & 9 & 11 \\ -2 & 0 & 2 \end{bmatrix}$$

3. Tipos de matrices

Podemos obtener las dimensiones de una matriz con el siguiente código:

[In]:
```python
# Definimos las matrices
A = Matrix([[1, 0, 1], [0, 1, 1]])
print("A:")
display(A)
print("Las dimensiones de A son:", A.shape)
print("El número de filas de A es:", A.shape[0])
print("El número de columnas de A es:", A.shape[1])
```

[Out]:
A:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

```
Las dimensiones de A son: (2, 3)
El número de filas de A es: 2
El número de columnas de A es: 3
```

Ahora, veamos cómo crear algunas matrices especiales:

- Matriz diagonal:

[In]:
```python
diag(1, 2, 3)
```

[Out]:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

- Matriz identidad:

[In]:
```python
eye(4)
```

[Out]:
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz nula:

[In]:
```python
zeros(2, 4)
```

[Out]:
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- Matriz formada de unos:

[In]:
```python
ones(4, 2)
```

[Out]:
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

4. Sistemas de ecuaciones

Podemos generar una matriz ampliada de la siguiente manera:

[In]:
```
# Defino las matrices
A = Matrix([[1, 2, 3], [3, 2, 1], [1, -1, 0]])
print("A:")
display(A)
B = Matrix([[1, 0, 2], [2, 1, 0], [0, 1, 0]])
print("B:")
display(B)
# Genero la matriz ampliada
print("(A|B):")
Matrix(BlockMatrix([A, B]))
```

[Out]:
```
A:
```
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$
```
B:
```
$$\begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$
```
(A|B):
```
$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 2 \\ 3 & 2 & 1 & 2 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

4.1 Operaciones por filas

- Intercambio de filas:

[In]:
```
# Imprimo la matriz original
print("A:")
display(A)
# Operación de filas
print("Intercambio la fila 0 con la 2:")
A.elementary_row_op('n<->m', row1=0, row2=2)
```

[Out]:
```
A:
```
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$
```
Intercambio la fila 0 con la 2:
```
$$\begin{bmatrix} 1 & -1 & 0 \\ 3 & 2 & 1 \\ 1 & 2 & 3 \end{bmatrix}$$

- Multiplicar una fila por un escalar:

[In]:
```python
# Imprimo la matriz original
print("A:")
display(A)
# Operación de filas
print("Multiplico por 3 la fila 2:")
A.elementary_row_op('n->kn', k=3, row=2)
```

[Out]:
```
A:
```
$$\begin{bmatrix}1 & 2 & 3\\3 & 2 & 1\\1 & -1 & 0\end{bmatrix}$$
```
Multiplico por 3 la fila 2:
```
$$\begin{bmatrix}1 & 2 & 3\\3 & 2 & 1\\3 & -3 & 0\end{bmatrix}$$

- Sumar un múltiplo de una fila con otra:

[In]:
```python
# Imprimo la matriz original
print("A:")
display(A)
# Operación de filas
print("Multiplico por 3 la fila 0 y la sumo a la fila 2:")
A.elementary_row_op('n->n+km', k=3, row1=2, row2=0)
```

[Out]:
```
A:
```
$$\begin{bmatrix}1 & -2 & 1\\2 & 0 & -4\\3 & 1 & -2\end{bmatrix}$$
```
Multiplico por 3 la fila 0 y la sumo a la fila 2:
```
$$\begin{bmatrix}1 & -2 & 1\\2 & 0 & -4\\6 & -5 & 1\end{bmatrix}$$

4.1.1 Ejemplo de Gauss-Jordan

En caso de cometer algún error en uno de los pasos, deberá compilarse todas las celdas desde este punto.

[In]:
```python
# Defino la matriz
A = Matrix([[1, -2, 1], [2, 0, -4], [3, 1, -2]])
print("A:")
display(A)
# Guardo la matriz en una temporal
temp = A
```

[Out]: A:

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

- Primer paso: Multiplico por -2 la primera fila y lo sumo a la segunda.

```
[In]: temp = temp.elementary_row_op('n->n+km', k=-2, row1=1, row2=0)
      display(temp)
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 4 & -6 \\ 3 & 1 & -2 \end{bmatrix}$$

- Segundo paso: Multiplico por -3 la primera fila y lo sumo a la tercera.

```
[In]: temp = temp.elementary_row_op('n->n+km', k=-3, row1=2, row2=0)
      display(temp)
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 4 & -6 \\ 0 & 7 & -5 \end{bmatrix}$$

- Tercer paso: multiplico por $\frac{1}{4}$ la segunda fila.

```
[In]: temp = temp.elementary_row_op('n->kn', k=1/4, row=1)
      display(temp)
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 1{,}0 & -1{,}5 \\ 0 & 7 & -5 \end{bmatrix}$$

- Cuarto paso: Multiplico por -7 la segunda fila y lo sumo a la tercera.

```
[In]: temp = temp.elementary_row_op('n->n+km', k=-7, row1=2, row2=1)
      display(temp)
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 1{,}0 & -1{,}5 \\ 0 & 0 & 5{,}5 \end{bmatrix}$$

- Quinto paso: Multiplico por $\frac{1}{5{,}5}$ la tercera fila.

```
[In]: temp = temp.elementary_row_op('n->kn', k=1/5.5, row=2)
      display(temp)
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 1{,}0 & -1{,}5 \\ 0 & 0 & 1{,}0 \end{bmatrix}$$

4.1.2 Funciones para reducir una matriz

[In]:
```
# Defino la matriz
A = Matrix([[1, -2, 1], [2, 0, -4], [3, 1, -2]])
print("A:")
display(A)
```

[Out]: A:

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

- Forma escalonada sin unos principales.

[In]:
```
A.echelon_form()
```

[Out]:

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 4 & -6 \\ 0 & 0 & 22 \end{bmatrix}$$

- Forma escalonada reducida por fila.

[In]:
```
A.rref(pivots=False)
```

[Out]:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Rango de una matriz.

[In]:
```
A.rank()
```

[Out]: 3

4.2 Resolución de sistemas de ecuaciones

Consideremos el sistema de ecuaciones $Ax = b$, donde

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 2 \\ 2 & 4 & 0 \end{pmatrix} \quad \text{y} \quad b = \begin{pmatrix} 7 \\ 12 \\ 4 \end{pmatrix}.$$

Podemos resolverlo con el comando `solve` de la siguiente manera:

[In]:
```
# Defino las matrices
A = Matrix([[1, 2, 1], [0, 1, 2], [2, 4, 0]])
b = Matrix([7, 12, 4])
# Solución
A.solve(b)
```

[Out]:

$$\begin{bmatrix}-2\\2\\5\end{bmatrix}$$

Sin embargo, si el sistema no tiene solución única, nos dará un error. En estos casos, podemos utilizar `gauss_jordan_solve`:

[In]:

```
# Solución
sol, par = A.gauss_jordan_solve(b)
sol
```

[Out]:

$$\begin{bmatrix}-2\\2\\5\end{bmatrix}$$

Este comando, a más de determinar la solución, presenta los parámetros que esta tendrá en el caso de no tener solución única. Para esto, consideremos el sistema de ecuaciones $Ax = b$, con

$$A = \begin{pmatrix}1 & 2 & 1 & 1\\1 & 2 & 2 & -1\\2 & 4 & 0 & 6\end{pmatrix} \quad \text{y} \quad b = \begin{pmatrix}7\\12\\4\end{pmatrix}.$$

[In]:

```
# Definimos la matrices
A = Matrix([[1, 2, 1, 1], [1, 2, 2, -1], [2, 4, 0, 6]])
b = Matrix([7, 12, 4])
# Solución
sol, par = A.gauss_jordan_solve(b)
print("Solución:")
display(sol)
print("Parámetros:")
display(par)
```

[Out]:

```
Solución:
```

$$\begin{bmatrix}-2\tau_0 - 3\tau_1 + 2\\\tau_0\\2\tau_1 + 5\\\tau_1\end{bmatrix}$$

```
Parámetros:
```

$$\begin{bmatrix}\tau_0\\\tau_1\end{bmatrix}$$