Rotaciones

March 4, 2025

1 Rotaciones

1.1 Explicacion Matematica

Las rotaciones son transformaciones geométricas que mueven uno o varios puntos en un espacio bidimensional o tridimensional alrededor de un punto central fijo, llamado centro de rotación, sin alterar la forma ni el tamaño del objeto que se está rotando. La cantidad de rotación se define por el ángulo de rotación, que generalmente se mide en grados y en ocasiones en radianes. Y se pueden describir utilizando matrices de rotación.

1.1.1 Rotaciones en 2D

En un plano cartesiano bidimensional, una rotación se define por un ángulo θ y un centro de rotación.

Matriz de Rotación en 2D:

La transformación de un punto (x, y) (x, y) a un punto rotado (x', y') (x', y') se realiza mediante la multiplicación por la siguiente matriz de rotación:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Donde:

θ es el ángulo de rotación en radianes (positivo para rotaciones en sentido antihorario). $\cos(\theta)$ es el coseno del ángulo de rotación. $\sin(\theta)$ es el seno del ángulo de rotación. Aplicando la Rotación en 2D:

Para obtener las coordenadas del punto rotado (x', y') (x', y'), se multiplica la matriz de rotación por las coordenadas del punto original (x, y) (x, y):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Esto se traduce en las siguientes ecuaciones:

$$x' = x\cos(\theta) - y\sin(\theta) y' = x\sin(\theta) + y\cos(\theta)$$

$$x' = x\cos(\theta) - y\sin(\theta) y' = x\sin(\theta) + y\cos(\theta)$$

Estas ecuaciones describen cómo las coordenadas del punto original (x, y) (x, y) se transforman en las coordenadas del punto rotado (x', y') (x', y') después de una rotación de ángulo θ alrededor del origen.

1.1.2 Rotaciones en 3D

Las rotaciones en el espacio tridimensional involucran rotaciones alrededor de tres ejes: x, y, z. Cada rotación se describe mediante una matriz de rotación de 3x3.

Matrices de Rotación en 3D:

Rotación alrededor del eje x:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Rotación alrededor del eje y:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

Rotación alrededor del eje z:

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combinando Rotaciones en 3D:

Para realizar rotaciones compuestas en 3D, se multiplican las matrices de rotación correspondientes. Es crucial tener en cuenta que el orden de multiplicación importa, ya que las rotaciones en 3D no son conmutativas. La rotación resultante dependerá del orden en que se apliquen las rotaciones individuales.

Ejemplo:

Para rotar un punto primero alrededor del eje x por un ángulo α y luego alrededor del eje y por un ángulo β, se utiliza la siguiente matriz compuesta:

$$R = R_y(\beta)R_x(\alpha)$$
$$R = R_y(\beta)R_x(\alpha)$$

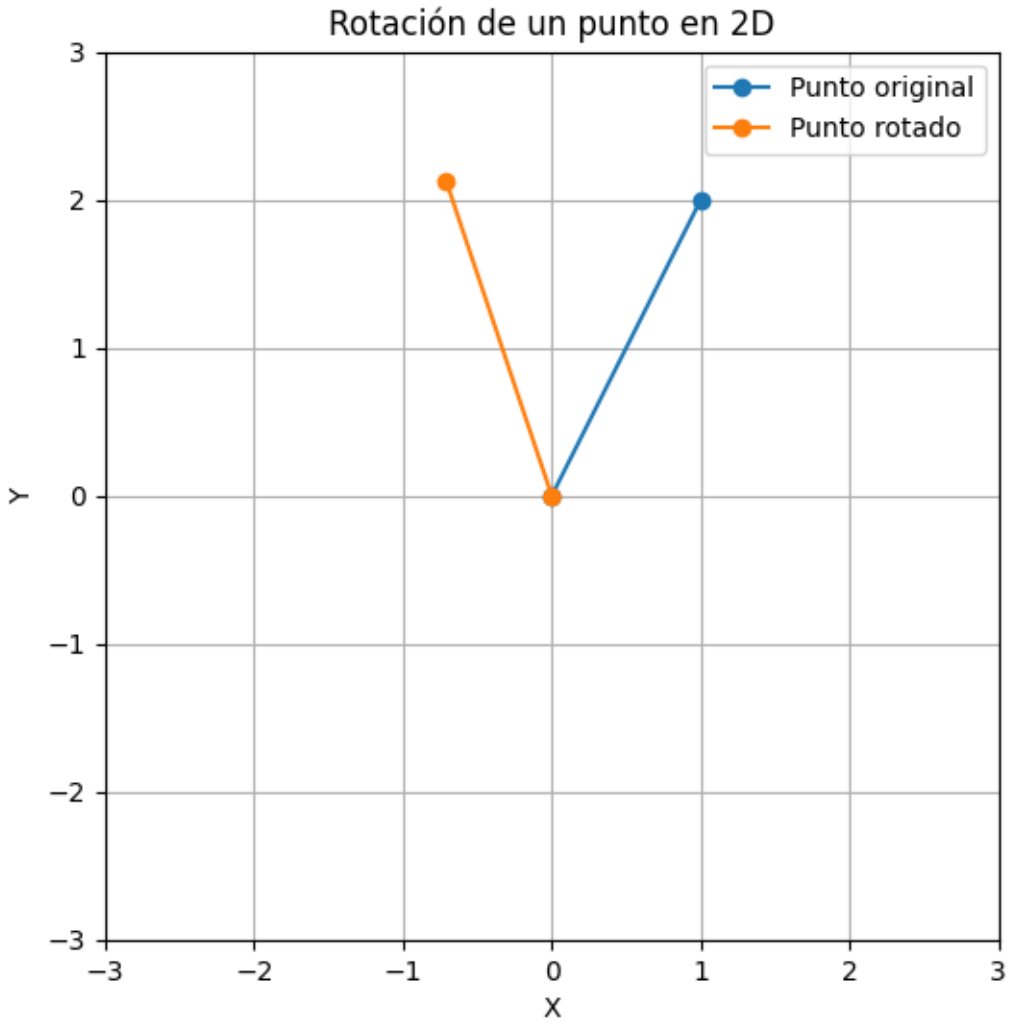
La multiplicación de matrices se realiza de derecha a izquierda, lo que significa que la rotación alrededor del eje x se aplica primero.

1.1.3 Codigo en Python para ejemplo de rotación

[1]:
```
### Ejemplo de Código en Python para rotar un punto

#importar las librerias
import numpy as np
import matplotlib.pyplot as plt

# Definir un punto
punto = np.array([1, 2])

# Definir el ángulo de rotación en radianes
angulo = np.pi / 4  # 45 grados

# Matriz de rotación 2D
matriz_rotacion = np.array([
    [np.cos(angulo), -np.sin(angulo)],
    [np.sin(angulo), np.cos(angulo)]
])

# Aplicar la rotación
punto_rotado = np.dot(matriz_rotacion, punto)

# Imprimir los resultados
print("Punto original:", punto)
print("Punto rotado:", punto_rotado)


# Visualizar la rotación
plt.figure(figsize=(6,6))
plt.plot([0, punto[0]], [0, punto[1]], label='Punto original', marker='o',␣
 ↪linestyle='-')
plt.plot([0, punto_rotado[0]], [0, punto_rotado[1]], label='Punto rotado',␣
 ↪marker='o', linestyle='-')

plt.xlim([-3, 3])
plt.ylim([-3, 3])
plt.title('Rotación de un punto en 2D')
```

```
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```

```
Punto original: [1 2]
Punto rotado: [-0.70710678  2.12132034]
```



1.2 Aplicaciones en Ciencias de datos

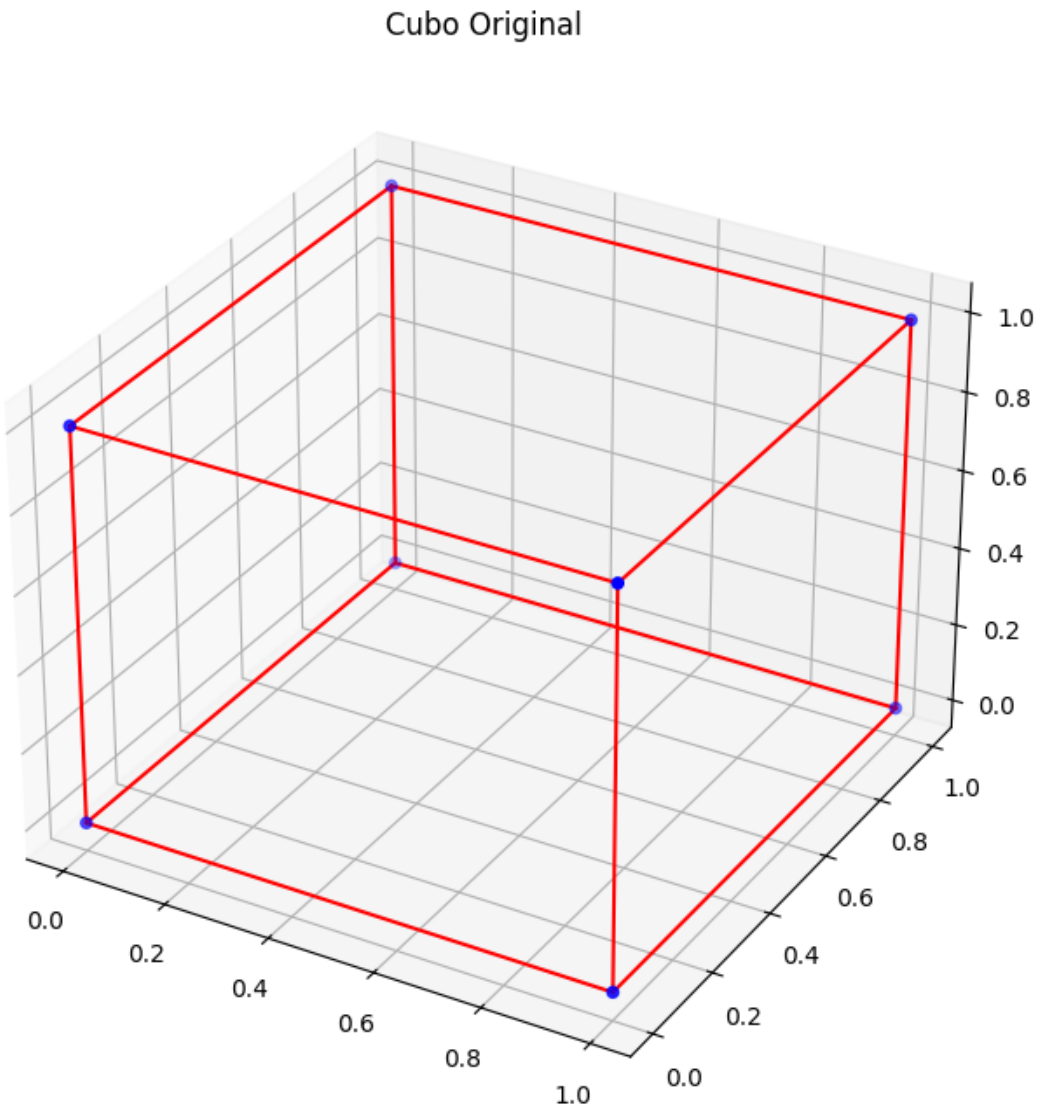
Las rotaciones tienen gran cantidad de aplicaciones en el campo de la ciencia de datos. Algunas de las que hemos usado es en la creación de objetos tridimensionales y su rotación. Por ejemplo los objetos 3D necesitan ser rotados para ser visualizados desde diferentes ángulos. Mediante el uso de las rotaciones se genera la base para la manipulación y visualización de objetos 3D. Esto funciona

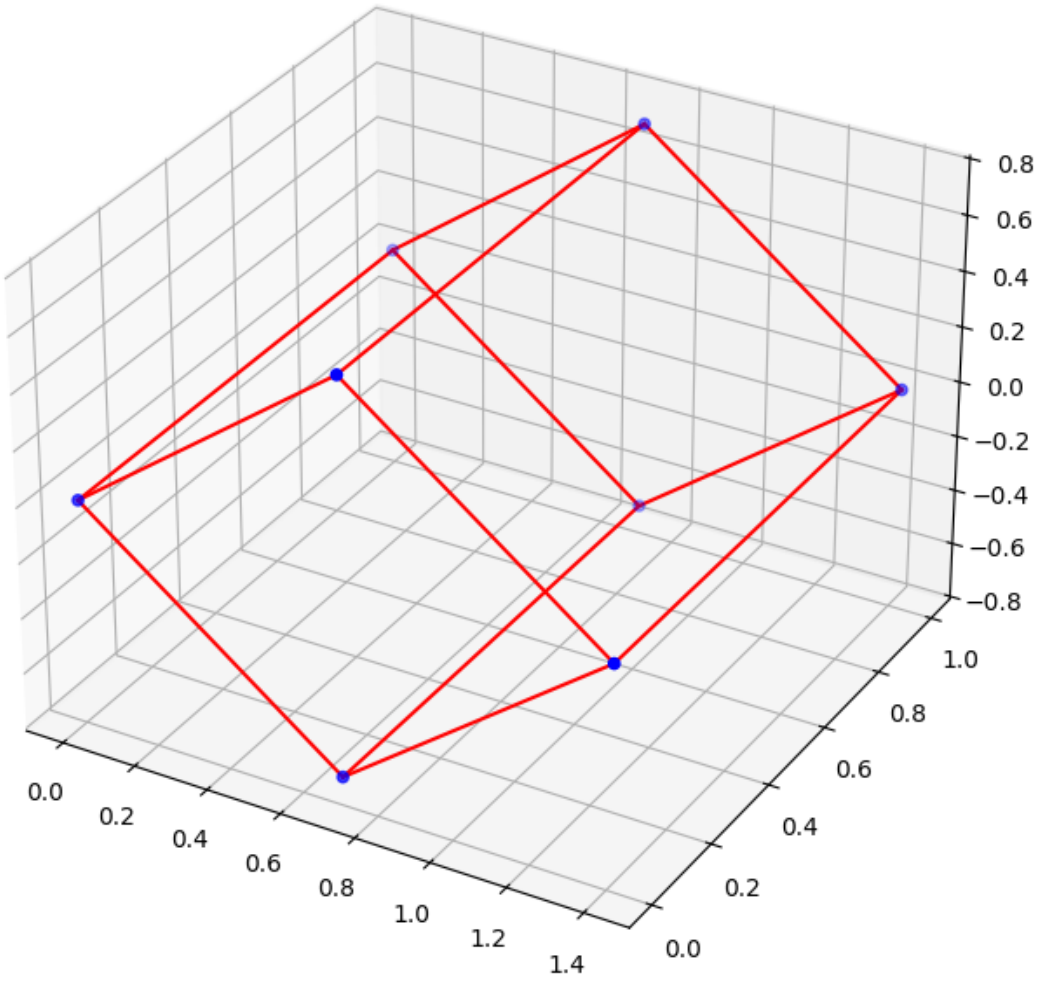
a partir de las matrices de rotación que se aplican para transformar las coordenadas de los vértices de los objetos 3D, permitiendo la rotación alrededor de diferentes ejes.

1.2.1 Ejemplo en Python para rotar un Cubo en 3D con un ángulo de rotación 45°

[2]:
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Definir un cubo simple
puntos_cubo = np.array([
    [0, 0, 0],
    [1, 0, 0],
    [1, 1, 0],
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 1],
    [1, 1, 1],
    [0, 1, 1]
])

# Función para dibujar el cubo
def dibujar_cubo(puntos, ax):
    ax.scatter(puntos[:, 0], puntos[:, 1], puntos[:, 2], c='b', marker='o')
    # Conectar los puntos para formar las aristas del cubo
    edges = [
        [0, 1], [1, 2], [2, 3], [3, 0],
        [4, 5], [5, 6], [6, 7], [7, 4],
        [0, 4], [1, 5], [2, 6], [3, 7]
    ]
    for edge in edges:
        ax.plot(puntos[edge, 0], puntos[edge, 1], puntos[edge, 2], c='r')

# Crear la figura 3D
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Dibujar el cubo original
dibujar_cubo(puntos_cubo, ax)
ax.set_title("Cubo Original")

# Aplicar rotación alrededor del eje y (45 grados)
angulo = np.pi / 4  # 45 grados en radianes
matriz_rotacion_y = np.array([
    [np.cos(angulo), 0, np.sin(angulo)],
    [0, 1, 0],
    [-np.sin(angulo), 0, np.cos(angulo)]
```

```
])
puntos_rotados = np.dot(puntos_cubo, matriz_rotacion_y.T)

# Crear una nueva figura para el cubo rotado
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Dibujar el cubo rotado
dibujar_cubo(puntos_rotados, ax)
ax.set_title("Cubo Rotado (alrededor del eje y)")

plt.show()
```




Cubo Rotado (alrededor del eje y)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4