Práctica 1

Raúl Soria González

23 de abril de 2021

Ejercicio 1

En el primer ejercicio se nos pedía realizar diferentes rotaciones a unos puntos que nos daban respecto a los ejes de coordenadas. Para ello he creado una función que devuelve la matriz de rotacióncon respecto al eje que le indiquemos y con un ángulo específico.

Luego simplemente si quiero realizar una rotación en un eje específico, llamo a la función y multiplico la matriz que devuelve por los puntos del enunciado.

Podemos ver en la imágen (Figura 1) que al ejecutar, se realizan correctamente los giros en ángulos fijos con respecto al sistema A.

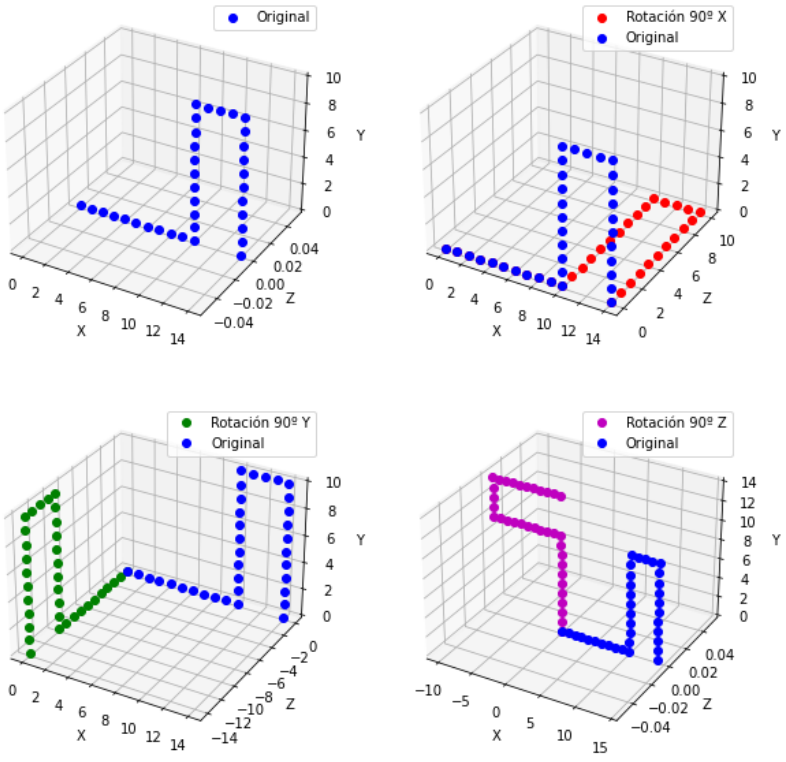

Figura 1: Rotaciones 90º en todos los ejes

Ejercicio 2

Para este ejercicio he utilizado la misma función que en el ejercicio 1 para generar las matrices de rotación y así poder realizar las rotaciones multiplicando por los puntos.

He creado la matriz de rotación final multiplicando las matrices correspondientes a cada eje $R_X * R_Y * R_Z$. Se realizan en este orden ya que son ángulos de Euler y las transformaciones se realizan de izquierda a derecha.

Como podemos ver en la imágen (Figura 2) no obtenemos el mismo resultado si realizamos los giros en distinto orden. En el segundo gráfico vemos qué pasaría si realizamos primero un giro en Z seguido de uno en Y y finalizando con el giro en el eje X.

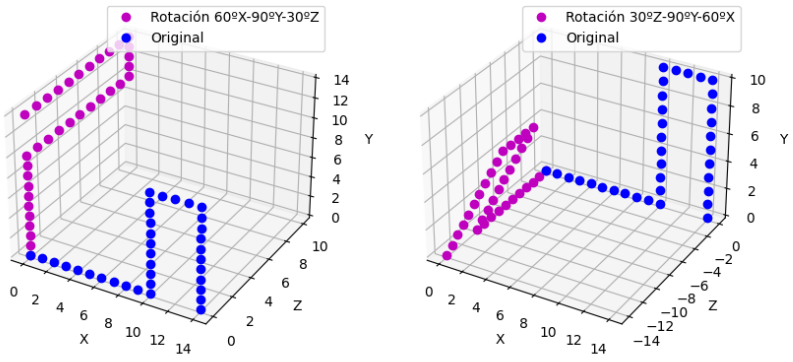

Figura 2: Comparación rotaciones X-Y-Z vs Z-Y-X

Ejercicio 3

En este ejercicio tenemos que leer una matriz con los parametros de Denavit-Hartenberg y devolver una matriz de transformación homogénea.

Para ello he hecho una función que crea las matrices de transformación tanto para la rotación en Z, el desplazamiento en Z, el desplazamiento en X y la rotación en X correspondientes a las columnas de la matriz de entrada. Esto se realiza para cada fila y se van multiplicando en el orden necesario para finalmente obtener la matriz de transformación homogénea.

Para poder utilizar parámetros de articulaciones y longitudes del robot, se ha utilizado la librería sympy.

Archivos de código

ejer1.py

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

pxB = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 10,
                10, 10, 10, 10, 10, 10, 11, 12, 13, 14, 14, 14, 14,
                14, 14, 14, 14, 14, 14, 14])
pyB = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7,
                8, 9, 10, 10, 10, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
pzB = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
pB = np.array([pxB, pyB, pzB])


def rotation(eje, alpha):

    if eje.upper() == 'X':
        rotation = np.array([1, 0, 0,
                             0, np.cos(alpha), -np.sin(alpha),
                             0, np.sin(alpha), np.cos(alpha)])
    elif eje.upper() == 'Y':
        rotation = np.array([np.cos(alpha), 0, np.sin(alpha),
                             0, 1, 0,
                             -np.sin(alpha), 0, np.cos(alpha)])
    else:
        rotation = np.array([np.cos(alpha), -np.sin(alpha), 0,
                             np.sin(alpha), np.cos(alpha), 0,
                             0, 0, 1])

    rotation = rotation.reshape(3, 3)
    return rotation


pBX = rotation('X', np.deg2rad(90)).dot(pB)
xX = pBX[0]
yX = pBX[1]
zX = pBX[2]
pBY = rotation('Y', np.deg2rad(90)).dot(pB)
xY = pBY[0]
```

```
yY = pBY[1]
zY = pBY[2]
pBZ = rotation('Z', np.deg2rad(90)).dot(pB)
xZ = pBZ[0]
yZ = pBZ[1]
zZ = pBZ[2]

fig = plt.figure(figsize=(10,10))

ax = fig.add_subplot(221, projection='3d')
ax.plot(pxB, pzB, pyB, 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend()

ax = fig.add_subplot(222, projection='3d')
ax.plot(xX, zX, yX, 'ro', label='Rotación 90 X')
ax.plot(pxB, pzB, pyB, 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend()

ax = fig.add_subplot(223, projection='3d')
ax.plot(xY, zY, yY, 'go', label='Rotación 90 Y')
ax.plot(pxB, pzB, pyB, 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend()

ax = fig.add_subplot(224, projection='3d')
ax.plot(xZ, zZ, yZ, 'mo', label='Rotación 90 Z')
ax.plot(pxB, pzB, pyB, 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend()

plt.show()
```

ejer2.py

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

pxB = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 10,
                10, 10, 10, 10, 10, 10, 11, 12, 13, 14, 14, 14, 14,
                14, 14, 14, 14, 14, 14, 14])
pyB = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7,
                8, 9, 10, 10, 10, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
pzB = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
pB = np.array([pxB, pyB, pzB])


def rotation(eje, alpha):
    if eje.upper() == 'X':
        rotation = np.array([[1, 0, 0],
                             [0, np.cos(alpha), -np.sin(alpha)],
                             [0, np.sin(alpha), np.cos(alpha)]])
    elif eje.upper() == 'Y':
        rotation = np.array([[np.cos(alpha), 0, np.sin(alpha)],
                             [0, 1, 0],
                             [-np.sin(alpha), 0, np.cos(alpha)]])
    else:
        rotation = np.array([[np.cos(alpha), -np.sin(alpha), 0],
                             [np.sin(alpha), np.cos(alpha), 0],
                             [0, 0, 1]])

    return rotation


t = rotation('X', np.deg2rad(60)).dot(rotation('Y', np.deg2rad(90)))\
    .dot(rotation('Z', np.deg2rad(30)))
pt = t.dot(pB)

t2 = rotation('Z', np.deg2rad(30)).dot(rotation('Y', np.deg2rad(90)))\
```

```
    .dot(rotation('X', np.deg2rad(60)))
pt2 = t2.dot(pB)

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(121, projection='3d')
ax.plot(pt[0], pt[2], pt[1], 'mo', label='Rotación 60X—90Y—30Z')
ax.plot(pB[0], pB[2], pB[1], 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend(loc='upper right')

ax = fig.add_subplot(122, projection='3d')
ax.plot(pt2[0], pt2[2], pt2[1], 'mo', label='Rotación 30Z—90Y—60X')
ax.plot(pB[0], pB[2], pB[1], 'bo', label='Original')
ax.set_xlabel("X")
ax.set_ylabel("Z")
ax.set_zlabel("Y")
plt.legend(loc='upper right')

plt.show()
```

ejer3.py

```
import numpy as np
from sympy import *


def dh(matriz):
    final = np.identity(4)
    for fila in matriz:
        rot_z = np.array([[cos(fila[0]), -sin(fila[0]), 0, 0],
                          [sin(fila[0]), cos(fila[0]), 0, 0],
                          [0, 0, 1, 0],
                          [0, 0, 0, 1]])
        des_z = np.array([[1, 0, 0, 0],
                          [0, 1, 0, 0],
                          [0, 0, 1, fila[1]],
                          [0, 0, 0, 1]])
```

```
        des_x = np.array([[1, 0, 0, fila[2]],
                          [0, 1, 0, 0],
                          [0, 0, 1, 0],
                          [0, 0, 0, 1]])
        rot_x = np.array([[1, 0, 0, 0],
                          [0, cos(fila[3]), -sin(fila[3]), 0],
                          [0, sin(fila[3]), cos(fila[3]), 0],
                          [0, 0, 0, 1]])

        final = final.dot(rot_z)
        final = final.dot(des_z)
        final = final.dot(des_x)
        final = final.dot(rot_x)

    return final


q1, q2, q3 = symbols('q1 q2 q3')
l1, l2, l3 = symbols('l1 l2 l3')
matriz_in = np.array([[q1, l1, 0, 0],
                      [np.deg2rad(90), q2, 0, np.deg2rad(90)],
                      [0, l3 + q3, 0, 0]])

mat = dh(matriz_in)

mat = nsimplify(mat, tolerance=1e-6)

pprint(mat)
```