Práctica 2

Antonio Jesús Heredia Castillo

10 de mayo de 2020

Ejercicio 1

En este ejercicio tenemos que implementar una función que resuelva el problema cinemático directo, es decir, que dados los ángulos y la longitud de cada articulación nos devuelva la coordenada (x, y) donde se encuentra el extremo de nuestro robot. Para ello he implementado las formulas:

$$p_x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$p_y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

Ejercicio 2

En este ejercicio se nos pide que imprimamos los puntos por los que pasará el extremo de nuestro robot, dados unos puntos.

```
def dibujar_trayectoria_pcd (q1s, q2s , l1 , l2 ):
        xs = []
        ys  = []
        for i,q1 in enumerate(q1s):
                x,y = pcd(q1,q2s[i],l1, l2)
                xs.append(x)
                ys.append(y)
        #Ejercicio 3
        dibujar_robot(q1,q2s[i],l1,l2)
        plt.plot(xs,  ys)
```

Como se puede ver en el código lo que realizo es un for sobre los ángulos de $q1s$ y en cada iteración cojo un angulo de $q1s$ y otro de $q2s$, se los paso a la función para calcular la cinemática directa guardando los resultados en un vector para luego mostrarlos todos y que se vea la trayectoria completa. Ademas se puede ver lo que añado para el Ejercicio 3 para que se vea lo que seria el brazo del robot.

Ejercicio 4

En la Figura 1 podemos ver el resultado de usar las funciones de los tres ejercicios anteriores.

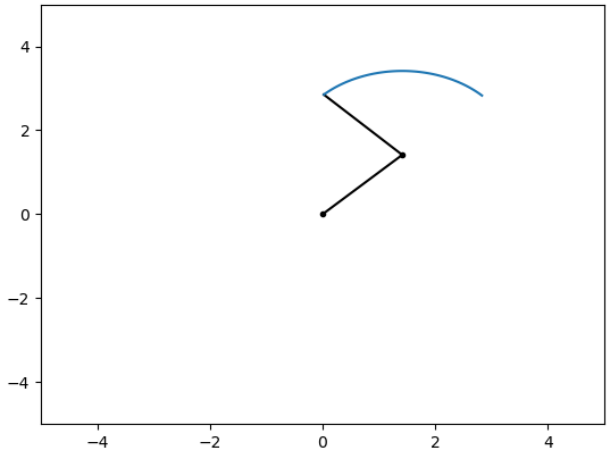

Figura 1: Trayectoria realizada con las especificaciones del ejercicio 4

Ejercicio 6

Se puede ver en el ejecutando el código. En la "animación" podemos ver como hasta que el primer brazo llega a los 90º el segundo brazo aumenta su angulo de forma positiva, una vez que la primera articulación pasa los 90º la segunda articulación empieza a decrecer quedando al final la trayectoria de la Figura 2.

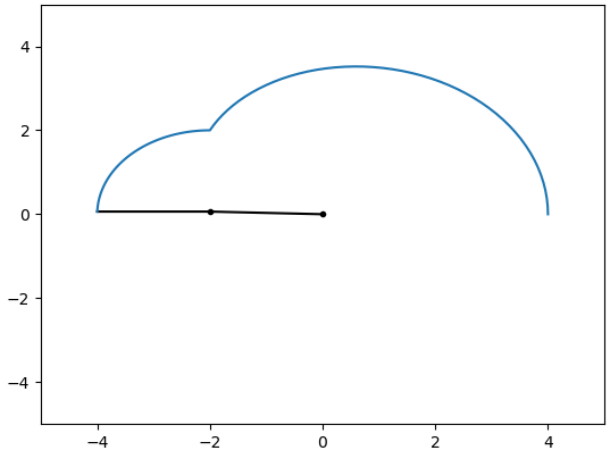

Figura 2: Trayectoria realizada con las especificaciones del ejercicio 6

Ejercicio 7

El problema cinemático inverso se resuelve igual de fácil que el directo. Con la cinemática inversa conseguimos los ángulos que tiene que tener nuestras articulaciones para conseguir que el extremo del robot este en un punto determinado. Para ello hacemos uso de las siguientes ecuaciones:

$$\cos\theta_2 = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}$$

$$\tan\theta_1 = \frac{y(L_1 + L_2\cos\theta_2) - xL_2\sin\theta_2}{x(L_1 + L_2\cos\theta_2) + yL_2\sin\theta_2}$$

Como sabemos, en el problema cinemático inverso, la posición en la que se deba encontrar la articulación i depende de la articulación $i+1$ en caso de que esta no sea la ultima. Por eso, el angulo θ_1 depende de θ_2.

Ejercicio 9

En este apartado podemos ver el resultado de hacer uso de las funciones implementadas en los dos ejercicios anteriores, para ello haremos pruebas con diferentes listas de puntos. Aunque para ver mas detalles aconsejo ver la animación en el código.
Desde el punto $(2, 0)$ hasta el punto $(0, 2)$:

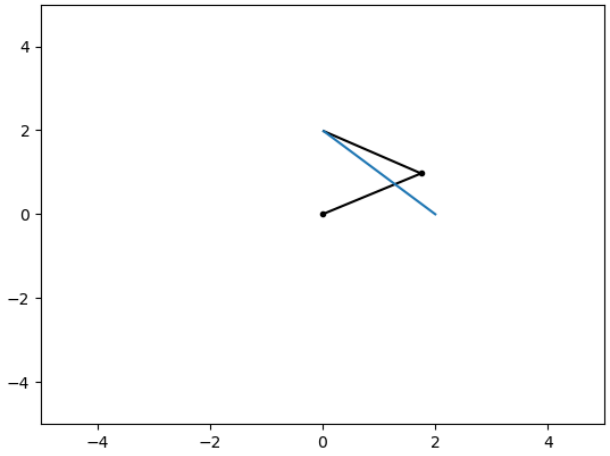

Figura 3: Desde el punto $(2, 0)$ hasta $(0, 2)$

Como podemos ver en la Figura 3, en este caso realiza la trayectoria, cosa que no pasara igual en los dos siguientes apartados.
Desde el punto $(1, 1)$ hasta el punto $(-2, 1)$:

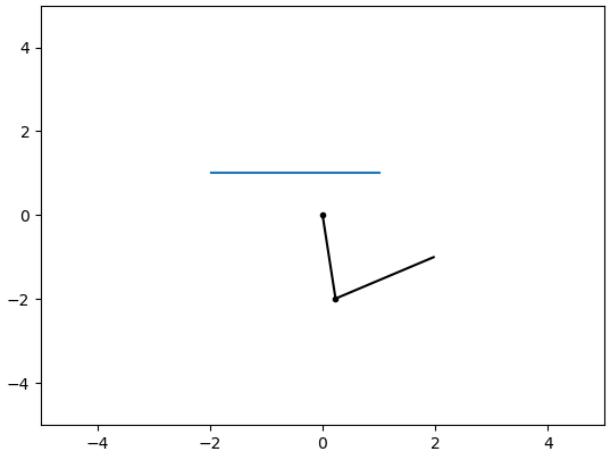

Figura 4: Desde el punto $(1, 1)$ hasta $(-2, 1)$ con arctan

En la Figura 4 podemos ver como no acaba bien la simulación y hace cosas raras las articulaciones. Esto es por que **arctan** no determina correc-

tamente en que cuadrante se encuentra la solución, creando asi problemas de continuidad. Esto se puede solucionar utilizando **arctan2**.

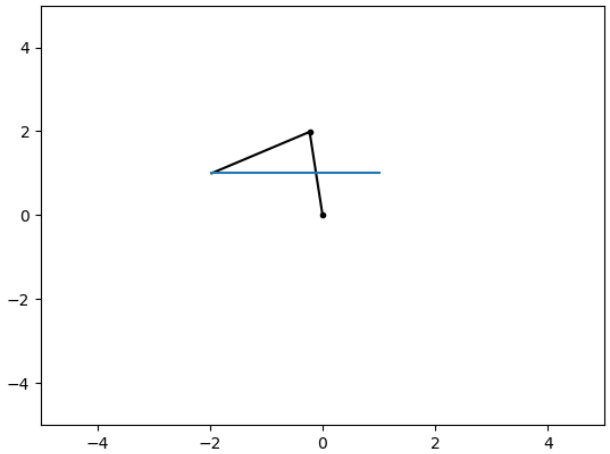

Figura 5: Desde el punto $(1, 1)$ hasta $(-2, 1)$ con arctan2

Teniendo al final la solución correcta como podemos ver en la Figura 5. Desde el punto $(1, 0)$ hasta el punto $(-1, 0)$:

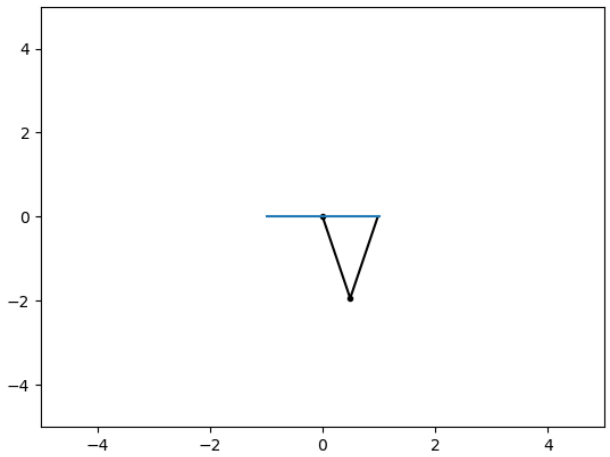

Figura 6: Desde el punto $(1, 0)$ hasta $(-1, 0)$ con arctan

En la Figura 6 podemos ver como el extremo del robot empieza acaba en

el mismo sitio, en cambio la trayectoria no es esa, esto pasa otra vez por hacer uso de **arctan**. Lo solucionamos igual que en caso anterior usando **arctan2**.

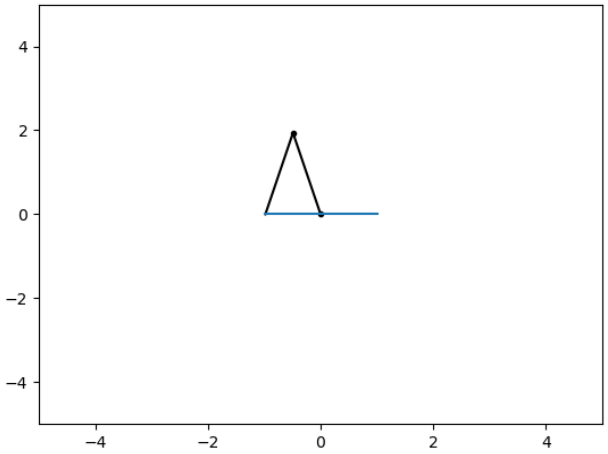

Figura 7: Desde el punto $(1, 0)$ hasta $(-1, 0)$ con arctan2

Pero si vemos la animación, volvemos a ver que la articulación se mueve de forma extraña. Cuando $x > 0$ la articulación 2 esta en una posición con $y < 0$ en cambio cuando $x < 0$ la posición de la segunda articulación pasa a ser $y > 0$. Esta vez es culpa del **arccos**, que es el mismo pero en negativo. Este problema lo solucionaremos en el siguiente ejercicio.

Ejercicio 10

En la modificación de este ejercicio conseguiremos que por culpa del coseno, la articulaciones hagan cambio bruscos. Esto lo conseguiremos haciendo que "el robot" pueda distinguir cual giro provoca un salto brusco y cual es un giro fluido. Consiguiendo así que se solucione el salto que encontramos en la Figura 7. En la Figura 8 podemos ver como ya conseguimos que la articulación 2 siempre tenga una posición de $y < 0$. Como siempre esto se podrá ver mejor en la ejecución del código.

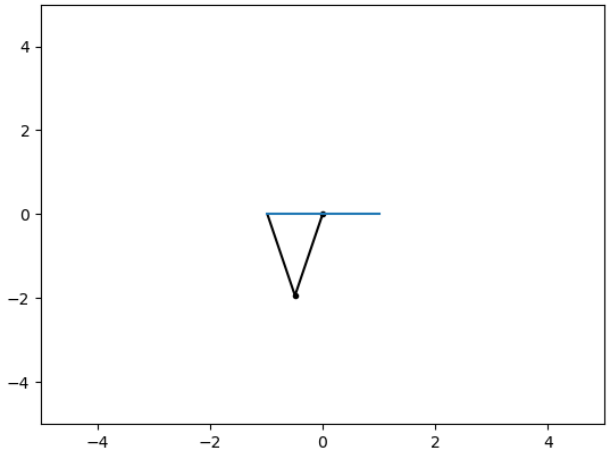

Figura 8: Desde el punto $(1, 0)$ hasta $(-1, 0)$ con arctan2 y viendo cuando hay un cambio brusco.

Archivos de código

ejercicio1-2-3-4.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 10 01:43:47 2020

@author: antonio
"""

import numpy as np
import matplotlib.pyplot as plt

#Ejercicio 1
def pcd ( q1 , q2 , l1 , l2 ):
    y=l1*np.sin(q1)+l2*np.sin(q1+q2)
    x=l1*np.cos(q1)+l2*np.cos(q1+q2)
    return (x,y)
#Ejercicio 3
def dibujar_robot ( q1 , q2 , l1 , l2 ):
    x0 , y0 = 0 , 0; # Pos. de la articulaci ón 1
    x1 , y1 = pcd ( q1 , 0 , l1 , 0); # Pos. de la articulaci ón 2
    x2 , y2 = pcd ( q1 , q2 , l1 , l2 ); # Pos. del extremo del robot
    x = [ x0 , x1 , x2 ]; y = [ y0 , y1 , y2 ]; # Coordenadas de la trayec .
    plt . plot (x , y , 'k'); # Traza la trayectoria
    plt . plot ( x0 , y0 , 'k.'); # Dibuja la articulaci ón 1
    plt . plot ( x1 , y1 , 'k.'); # Dibuja la articulaci ón 2

#Ejercicio 2
def dibujar_trayectoria_pcd (q1s, q2s , l1 , l2 ):
    xs = []
    ys  = []
    for i,q1 in enumerate(q1s):
        x,y = pcd(q1,q2s[i],l1, l2)
        xs.append(x)
        ys.append(y)
    dibujar_robot(q1,q2s[i],l1,l2)
    plt.plot(xs,  ys)

#Ejercicio 4
```

```
q1s=np.full(100, np.pi/4)
q2s=np.linspace(0, np.pi/2, 100)
l1 = l2 = 2
dibujar_trayectoria_pcd(q1s,q2s,l1,l2)
```

ejercicio5-6.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 10 01:44:52 2020

@author: antonio
"""

import numpy as np
import matplotlib.pyplot as plt

#Ejercicio 1
def pcd ( q1 , q2 , l1 , l2 ):
    y=l1*np.sin(q1)+l2*np.sin(q1+q2)
    x=l1*np.cos(q1)+l2*np.cos(q1+q2)
    return (x,y)
#Ejercicio 3
def dibujar_robot ( q1 , q2 , l1 , l2 ):
    x0 , y0 = 0 , 0; # Pos. de la articulaci ón 1
    x1 , y1 = pcd ( q1 , 0 , l1 , 0); # Pos. de la articulaci ón 2
    x2 , y2 = pcd ( q1 , q2 , l1 , l2 ); # Pos. del extremo del robot
    x = [ x0 , x1 , x2 ]; y = [ y0 , y1 , y2 ]; # Coordenadas de la trayec .
    plt . plot (x , y , 'k'); # Traza la trayectoria
    plt . plot ( x0 , y0 , 'k.'); # Dibuja la articulaci ón 1
    plt . plot ( x1 , y1 , 'k.'); # Dibuja la articulaci ón 2

#Ejercicio 2
def dibujar_trayectoria_pcd (q1s, q2s , l1 , l2 ):
    xs = []
    ys  = []
    for i,q1 in enumerate(q1s):
        x,y = pcd(q1,q2s[i],l1, l2)
```

```
        xs.append(x)
        ys.append(y)
    dibujar_robot(q1,q2s[i],l1,l2)
    plt.plot(xs,  ys)

#Ejercicio 5
def animacion_trayectoria_pcd ( q1s , q2s , l1 , l2, titulo ):
    n = min(len( q1s ) , len( q2s ))
    fig1 = plt.figure (titulo)
    for i in range (1 , n ):
        plt . clf ()
        dibujar_trayectoria_pcd ( q1s [0: i ] , q2s [0: i ] , l1 , l2 )
        plt.axis([-5,5,-5,5])
        plt . pause (0.001)

#Ejercicio 6
q1s=np.linspace(0, np.pi, 101)
q2s=np.linspace(0, np.pi/2, 51)
q2sflip = np.delete(np.flip(q2s),0)
q2s=np.hstack((q2s,q2sflip))
l1 = l2 = 2
animacion_trayectoria_pcd(q1s,q2s,l1,l2,"Ejericio 6")
```

ejercicio7-8-9.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 10 01:48:21 2020

@author: antonio
"""
import numpy as np
import matplotlib.pyplot as plt
def pcd ( q1 , q2 , l1 , l2 ):
    y=l1*np.sin(q1)+l2*np.sin(q1+q2)
    x=l1*np.cos(q1)+l2*np.cos(q1+q2)
    return (x,y)
def dibujar_robot ( q1 , q2 , l1 , l2 ):
```

```
    x0 , y0 = 0 , 0; # Pos. de la articulaci ón 1
    x1 , y1 = pcd ( q1 , 0 , l1 , 0); # Pos. de la articulaci ón 2
    x2 , y2 = pcd ( q1 , q2 , l1 , l2 ); # Pos. del extremo del robot
    x = [ x0 , x1 , x2 ]; y = [ y0 , y1 , y2 ]; # Coordenadas de la trayec .
    plt . plot (x , y , 'k'); # Traza la trayectoria
    plt . plot ( x0 , y0 , 'k.'); # Dibuja la articulaci ón 1
    plt . plot ( x1 , y1 , 'k.'); # Dibuja la articulaci ón 2


#Ejercicio 7
def pci(x,y,l1,l2):
    q2 = np.arccos((x*x+y*y-l1*l1-l2*l2)/(2*l1*l2))
    #Descomentar segun se quiera usar arctan o arctan2
    #q1 = np.arctan((y*(l1+l2*np.cos(q2))-x*l2*np.sin(q2))/
                   #(x*(l1+l2*np.cos(q2))+y*l2*np.sin(q2)))
    q1 = np.arctan2((y*(l1+l2*np.cos(q2))-x*l2*np.sin(q2)),(
                    x*(l1+l2*np.cos(q2))+y*l2*np.sin(q2)))
    return q1,q2


#ejercicio 8
def dibujar_trayectoria_pci(xs, ys , l1 , l2 ):
    q1s = []
    q2s  = []
    for i,x in enumerate(xs):
        q1,q2 = pci(x,ys[i],l1, l2)
        q1s.append(q1)
        q2s.append(q2)
    dibujar_robot(q1,q2,l1,l2)
    plt.plot(xs,  ys)

def animacion_trayectoria_pci(xs, ys , l1 , l2, titulo ):
    n = min(len( xs ) , len( ys ))
    fig1 = plt.figure (titulo)
    for i in range (1 , n ):
        plt . clf ()
        dibujar_trayectoria_pci ( xs [0: i ] , ys [0: i ] , l1 , l2 )
        plt.axis([-5,5,-5,5])
        plt . pause (0.001)

l1 = l2 = 2
```

```
#Ejercicio 9
xs = np.linspace(2,0,100)
ys = np.linspace(0,2,100)
animacion_trayectoria_pci(xs, ys , l1 , l2, "Primer apartado—Eje 9" )

xs = np.linspace(1,—2,100)
ys = np.linspace(1,1,100)
animacion_trayectoria_pci(xs, ys , l1 , l2, "Segundo  apartado—Eje 9" )

xs = np.linspace(1,—1,100)
ys = np.linspace(0,0,100)
animacion_trayectoria_pci(xs, ys , l1 , l2, "Tercer apartado—Eje 9" )
```

ejercicio10.py

```
#!/usr/bin/env python3
# —*— coding: utf—8 —*—
"""
Created on Sun May 10 01:44:43 2020

@author: antonio
"""

import numpy as np
import matplotlib.pyplot as plt
def pcd ( q1 , q2 , l1 , l2 ):
    y=l1*np.sin(q1)+l2*np.sin(q1+q2)
    x=l1*np.cos(q1)+l2*np.cos(q1+q2)
    return (x,y)
def dibujar_robot ( q1 , q2 , l1 , l2 ):
    x0 , y0 = 0 , 0; # Pos. de la articulaci ón 1
    x1 , y1 = pcd ( q1 , 0 , l1 , 0); # Pos. de la articulaci ón 2
    x2 , y2 = pcd ( q1 , q2 , l1 , l2 ); # Pos. del extremo del robot
    x = [ x0 , x1 , x2 ]; y = [ y0 , y1 , y2 ]; # Coordenadas de la trayec .
    plt . plot (x , y , 'k'); # Traza la trayectoria
    plt . plot ( x0 , y0 , 'k.'); # Dibuja la articulaci ón 1
    plt . plot ( x1 , y1 , 'k.'); # Dibuja la articulaci ón 2

#Ejercicio 10
```

```
def pci2(x,y,l1,l2):

    q2_1 = np.arccos((x*x+y*y-l1*l1-l2*l2)/(2*l1*l2))
    q2_2 = -q2_1

    q1_1 = np.arctan2((y*(l1+l2*np.cos(q2_1))-x*l2*np.sin(q2_1)),
                      (x*(l1+l2*np.cos(q2_1))+y*l2*np.sin(q2_1)))
    q1_2 = np.arctan2((y*(l1+l2*np.cos(q2_2))-x*l2*np.sin(q2_2)),
                      (x*(l1+l2*np.cos(q2_2))+y*l2*np.sin(q2_2)))

    return (q1_1,q1_2,q2_1,q2_2)

def dibujar_trayectoria_pci2(xs, ys , l1 , l2 ):
    q1s = []
    q2s  = []
    for i,x in enumerate(xs):
        q1_1,q1_2,q2_1,q2_2 = pci2(x,ys[i],l1, l2)

        if(len(q1s) > 0):
            if(abs(q1s[-1]-q1_1)<abs(q1s[-1]-q1_2)):
                q1s.append(q1_1)
                q2s.append(q2_1)
            else:
                q1s.append(q1_2)
                q2s.append(q2_2)
        else:
            q1s.append(q1_1)
            q2s.append(q2_1)



    dibujar_robot(q1s[-1],q2s[-1],l1,l2)
    plt.plot(xs,  ys)

def animacion_trayectoria_pc2i(xs, ys , l1 , l2,titulo ):
    n = min(len( xs ) , len( ys ))
    fig1 = plt.figure (titulo)
    for i in range (1 , n ):
        plt . clf ()
        dibujar_trayectoria_pci2 ( xs [0: i ] , ys [0: i ] , l1 , l2 )
        plt.axis([-5,5,-5,5])
```

```
        plt . pause (0.001)

l1 = l2 = 2

#Probamos el ejercicio 10
xs = np.linspace(1,-2,100)
ys = np.linspace(1,1,100)
animacion_trayectoria_pc2i(xs, ys , l1 , l2, "Segundo  apartado-Eje 10" )
xs = np.linspace(1,-1,100)
ys = np.linspace(0,0,100)
animacion_trayectoria_pc2i(xs, ys , l1 , l2, "Tercer apartado-Eje 10" )
```