Práctica 3

Raúl Soria González

27 de mayo de 2021

Ejercicio 1

Para el primer ejercicio se nos pedía crear una función `trayectoria434` que calculase los coeficientes de los tres polinomios que definen una trayectoria 4-3-4.

Esta función recibe como parámetros los valores de la articulación en los puntos de inicio (pI), despegue (pD), asentamiento (pA) y fin (pF), además hay que pasarle el tiempo que se tarda en realizar cada uno de los segmentos.

La función calculará cada uno de los coeficientes que conforman los tres polinomios. Algunos de estos coeficientes son sencillos de calcular a partir de las condiciones que representan. Para obtener otros, deberemos resolver un sistema de ecuaciones. El sistema de ecuaciones lo he resuelto planteándolo como una multiplicación matricial.

Una vez que tenemos los 14 coeficientes que representan las 14 condiciones que se deben cumplir, los agrupamos según a la función que pertenezcan y los devolvemos. Obteniendo así tres funciones por cada llamada.

Ejercicio 2

En este ejercicio se nos pide realizar una animación del un manipulador RR con brazos l_1 y l_2 con longitud 1 y cuya trayectoria comenzase en el punto $pI = (1, 0)$, $pD = (1, 0.1)$, $pA = (1.5, 0.1)$, $pF = (1.5, 0)$.

Para ello lo primero es calcular el valor de cada articulación en los puntos de ruta haciendo uso del problema citemático inverso.

Con estos valores de la articulación en los puntos clave, obtenemos los coeficientes de las funciones de la trayectoria para cada segmento. Para esto llamaremos a la función `trayectoria434` dos veces, una por cada articulación.

Ahora creamos 3 vectores de tiempos con 1 segundo para cada segmento normalizado gracias al cambio de variable. Para los segmentos 1 y 2 el tiempo va desde 0 hasta 1, para el tercer segmento va desde -1 hasta 0. Todos ellos con intervalos de 0.05 segundos.

Lo siguiente es crear dos vectores, uno por cada articulación, donde almacenaremos los valores de las articulaciones muestreados evaluando los polinomios obtenidas con la función `trayectoria434` para cada instante de tiempo. Para resolver los polinomios he utilizado la función `polyval` de numpy.

Estos valores de las articulaciones que conforman la trayectoria se los pasamos a la función que creamos en la práctica anterior `animacion_trayectoria_pcd` obteniendo así la animación de la trayectoria que pasa por los puntos pI, pD, pA y pF.

Yo esperaba que el resultado fuese una trayectoria recta, que desde pI hasta pD fuese en linea recta y del mismo modo los demás segmentos de dicha trayectoria. Pero al final, como podemos apreciar en la Figura **??** la trayectoria es curva, pasa por todos los puntos pero trazando una parábola. También he hecho otro recorrido con unos puntos clave más separados para que se observe mejor (Figura **??**).

Ejercicio 3

En este ejercicio se nos pedía estudiar la primera y segunda derivada de los polinomios obtenidos con `trayectoria434` para así observar los cambios de velocidad y aceleración articular.

Para obtener las derivadas he utilizado `polyder` de numpy y he representado las gráficas en la Figura **??**.

En la primera gráfica podemos ver la variación de velocidad, aceleración y además la posición en el espacio articular que iba tomando la articulación q_1 en la trayectoria. En la segunda podemos observar las gráficas correspondientes a la articulación q_2.

Como podemos ver, cuando la velocidad toma valores negativos es cuando el giro que está realizando es en sentido de las agujas del reloj. Los picos que observamos en las gráficas de aceleración se corresponden con los cambios de segmento.

También podemos observar como tanto en le instante inicial como en el final, la velocidad y la aceleración son nulas.

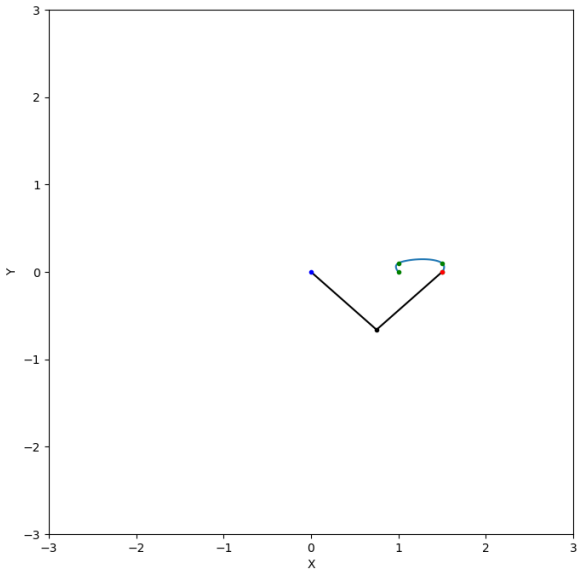

Figura 1: Ejercicio 2

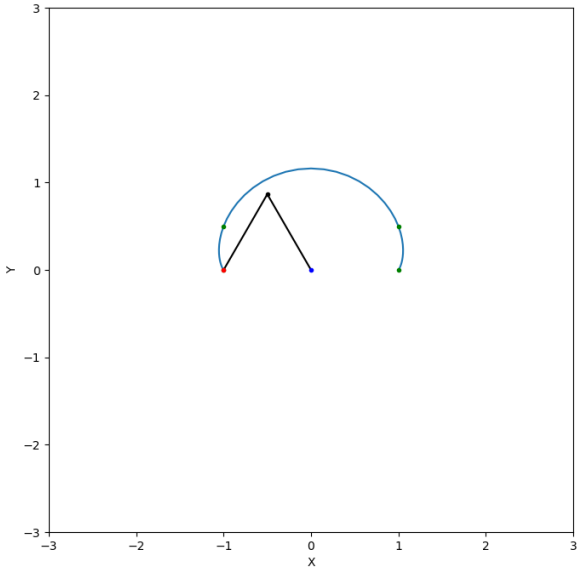

Figura 2: Trayectoria más grande

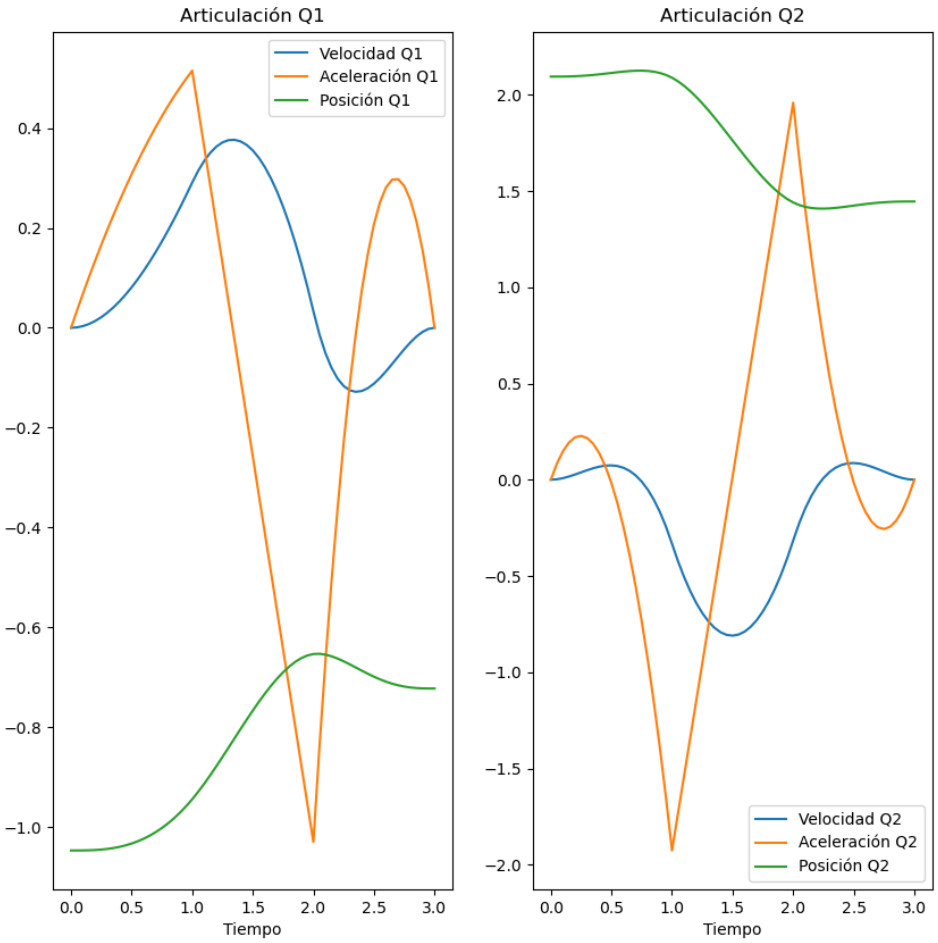

Figura 3: Gráficas de Velocidad y Aceleración

Archivos de código

practica3.py

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


def pcd(q1, q2, l1, l2):
    x1 = l1 * np.cos(q1) + l2 * np.cos(q1 + q2)
    y1 = l1 * np.sin(q1) + l2 * np.sin(q1 + q2)

    # return x1.round(5), y1.round(5)
    return x1, y1


def pci(xf, yf, l1, l2):
    cosq2 = (xf ** 2 + yf ** 2 — l1 ** 2 — l2 ** 2) / (2 * l1 * l2)
    sinq2 = np.sqrt(1 — cosq2 ** 2)
    q2 = np.arctan2(sinq2, cosq2)

    alpha = np.arctan((l2 * sinq2) / (l1 + l2 * cosq2))
    beta = np.arctan2(yf, xf)
    q1 = beta — alpha

    return q1, q2


##################################################################
###################     Ejercicio 1     ##########################
##################################################################

def trayectoria434(qI, qD, qA, qF, t1, t2, t3):
    c10 = qI
    c11 = 0
    c12 = 0
    c20 = qD
    c30 = qF
    c31 = 0
    c32 = 0

    mat1 = np.array([[1, 1, 0, 0, 0, 0, 0],
```

```
     [3 / t1, 4 / t1, —1 / t2, 0, 0, 0, 0],
     [6 / t1 ** 2, 12 / t1 ** 2, 0, —2 / t2 ** 2, 0, 0, 0],
     [0, 0, 1, 1, 1, 0, 0],
     [0, 0, 0, 0, 0, —1, 1],
     [0, 0, 1 / t2, 2 / t2, 3 / t2, —3 / t3, 4 / t3],
     [0, 0, 0, 2 / t2 ** 2, 6 / t2 ** 2, 6 / t3 ** 2, —12 / t3 ** 2]])
    inv = np.linalg.inv(mat1)

    mat2 = np.array([[qD — qI],
                     [0],
                     [0],
                     [qA — qD],
                     [qA — qF],
                     [0],
                     [0]])

    coef = inv @ mat2

    c13 = coef[0][0]
    c14 = coef[1][0]
    c21 = coef[2][0]
    c22 = coef[3][0]
    c23 = coef[4][0]
    c33 = coef[5][0]
    c34 = coef[6][0]

    f1 = np.array([c14, c13, c12, c11, c10])
    f2 = np.array([c23, c22, c21, c20])
    f3 = np.array([c34, c33, c32, c31, c30])

    return f1, f2, f3


def dibujar_trayectoria_pcd(q1s, q2s, l1, l2, pI, pD, pA, pF):
    vx, vy = [], []
    for i in range(len(q1s)):
        x, y = pcd(q1s[i], q2s[i], l1, l2)
        vx.append([x])
        vy.append([y])

    plt.plot(vx, vy)
```

```
    plt.plot(pI[0], pI[1], 'g.')
    plt.plot(pD[0], pD[1], 'g.')
    plt.plot(pA[0], pA[1], 'g.')
    plt.plot(pF[0], pF[1], 'g.')
    ax = plt.gca()
    fig = plt.gcf()
    ax.set_aspect('equal')
    fig.set_size_inches(8, 8)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.ylim([- (l1 + l2 + 1), l1 + l2 + 1])
    plt.xlim([- (l1 + l2 + 1), l1 + l2 + 1])
    dibujar_robot(q1s[len(q1s) - 1], q2s[len(q2s) - 1], l1, l2)
    plt.show()


def dibujar_robot(q1, q2, l1, l2):
    x0, y0 = 0, 0
    x1, y1 = pcd(q1, 0, l1, 0)
    x2, y2 = pcd(q1, q2, l1, l2)
    x, y = [x0, x1, x2], [y0, y1, y2]
    plt.plot(x, y, 'k')
    plt.plot(x1, y1, 'k.')
    plt.plot(x2, y2, 'r.')
    plt.plot(x0, y0, 'b.')


def animacion_trayectoria_pcd(q1s, q2s, l1, l2, pI, pD, pA, pF):
    n = min(len(q1s), len(q2s))
    for i in range(1, n+1):
        plt.clf()
        dibujar_trayectoria_pcd(q1s[0:i], q2s[0:i],
                                l1, l2, pI, pD, pA, pF)
        plt.pause(0.001)


##############################################################
###################     Ejercicio 2     ######################
##############################################################

l1 = l2 = 1
```

```
t1 = t2 = t3 = 1

pI = [1, 0]
pD = [1, 0.1]
pA = [1.5, 0.1]
pF = [1.5, 0]

q1I, q2I = pci(pI[0], pI[1], l1, l2)
q1D, q2D = pci(pD[0], pD[1], l1, l2)
q1A, q2A = pci(pA[0], pA[1], l1, l2)
q1F, q2F = pci(pF[0], pF[1], l1, l2)

f11, f12, f13 = trayectoria434(q1I, q1D, q1A, q1F, t1, t2, t3)
f21, f22, f23 = trayectoria434(q2I, q2D, q2A, q2F, t1, t2, t3)

t1m = np.arange(0, 1, 0.05)
t2m = np.arange(0, 1, 0.05)
t3m = np.arange(—1, 0.05, 0.05)

tiempo_total = np.arange(0, 3.05, 0.05)

res11 = []
res21 = []
for t in t1m:
    res11.append(np.polyval(f11, t))
    res21.append(np.polyval(f21, t))

res12 = []
res22 = []
for t in t2m:
    res12.append(np.polyval(f12, t))
    res22.append(np.polyval(f22, t))

res13 = []
res23 = []
for t in t3m:
    res13.append(np.polyval(f13, t))
    res23.append(np.polyval(f23, t))

res1 = res11 + res12 + res13
res2 = res21 + res22 + res23
```

```
animacion_trayectoria_pcd(res1, res2, l1, l2, pI, pD, pA, pF)


#################################################################
###################     Ejercicio 3     #########################
#################################################################

fv11 = np.polyder(f11)
fv12 = np.polyder(f12)
fv13 = np.polyder(f13)
fv21 = np.polyder(f21)
fv22 = np.polyder(f22)
fv23 = np.polyder(f23)

fa11 = np.polyder(fv11)
fa12 = np.polyder(fv12)
fa13 = np.polyder(fv13)
fa21 = np.polyder(fv21)
fa22 = np.polyder(fv22)
fa23 = np.polyder(fv23)

v11 = []
v21 = []
for t in t1m:
    v11.append(np.polyval(fv11, t))
    v21.append(np.polyval(fv21, t))

v12 = []
v22 = []
for t in t2m:
    v12.append(np.polyval(fv12, t))
    v22.append(np.polyval(fv22, t))

v13 = []
v23 = []
for t in t3m:
    v13.append(np.polyval(fv13, t))
    v23.append(np.polyval(fv23, t))

v1 = v11 + v12 + v13
```

```
v2 = v21 + v22 + v23

a11 = []
a21 = []
for t in t1m:
    a11.append(np.polyval(fa11, t))
    a21.append(np.polyval(fa21, t))

a12 = []
a22 = []
for t in t2m:
    a12.append(np.polyval(fa12, t))
    a22.append(np.polyval(fa22, t))

a13 = []
a23 = []
for t in t3m:
    a13.append(np.polyval(fa13, t))
    a23.append(np.polyval(fa23, t))

a1 = a11 + a12 + a13
a2 = a21 + a22 + a23

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(121)
ax.set_title("Articulación Q1")
ax.plot(tiempo_total, v1, label='Velocidad Q1')
ax.plot(tiempo_total, a1, label='Aceleración Q1')
ax.plot(tiempo_total, res1, label='Posición Q1')
ax.set_xlabel("Tiempo")
ax.set_ylabel("")
plt.legend()

ax = fig.add_subplot(122)
ax.set_title("Articulación Q2")
ax.plot(tiempo_total, v2, label='Velocidad Q2')
ax.plot(tiempo_total, a2, label='Aceleración Q2')
ax.plot(tiempo_total, res2, label='Posición Q2')
ax.set_xlabel("Tiempo")
ax.set_ylabel("")
plt.legend()
```

```
plt.show()
```