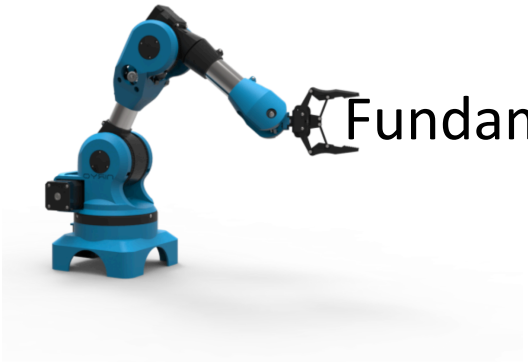
FACULTAD DE INGENIERIA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY



Fundamentos de Robótica Industrial

Trayectorias

27 de Mayo del 2025

Como en cada uno de los laboratorios, éste contará con una primera parte donde se realizará un control de lectura escrito, con el fin de verificar que estudió el material facilitado por los docentes y conoce cómo se llevará a cabo el laboratorio.

Laboratorio 3

Objetivo

Este laboratorio tiene por objetivo resolver una trayectoria circular en el plano xy que será recorrida por el puntero del manipulador.

Fundamentos y Metodología

Antes de la realización del laboratorio es indispensable haber leído y tener presente la guía de “Buenas prácticas para trabajo de laboratorio”, disponible en el EVA del curso.

Materiales a utilizar

Se utilizará:

- Manipulador de 4 grados de libertad (figura 1)
- USB 2D (figura 2)
- Fuente de alimentación 12V
- Computadora
- Soporte de marcador (figura 3)
- Marcador

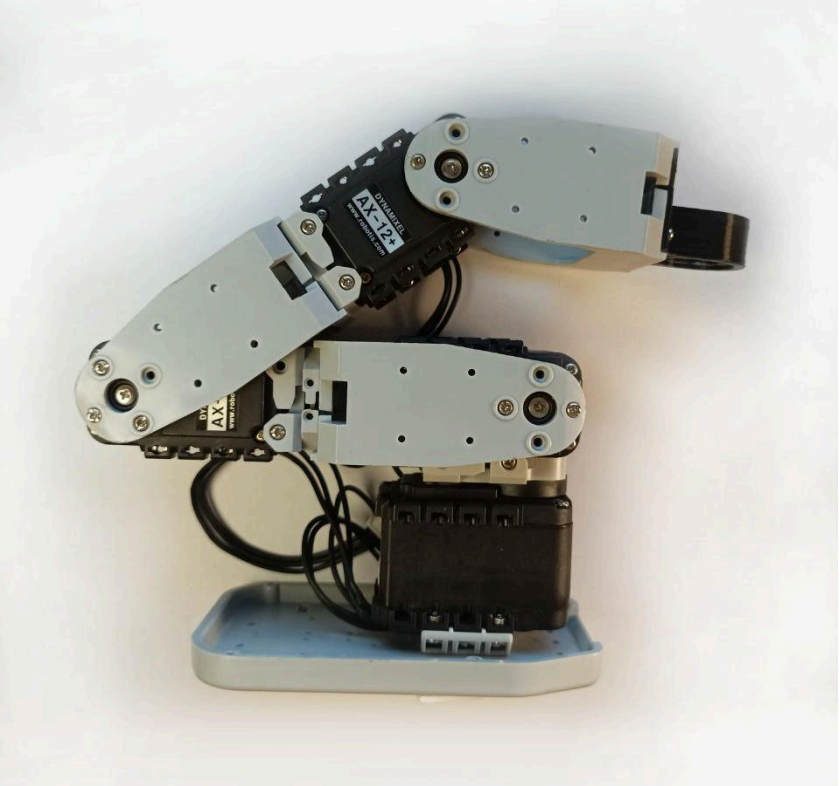


Figura 1: Referencia del manipulador a utilizar

Figura 2: Módulo USB U2D2 de Robotis

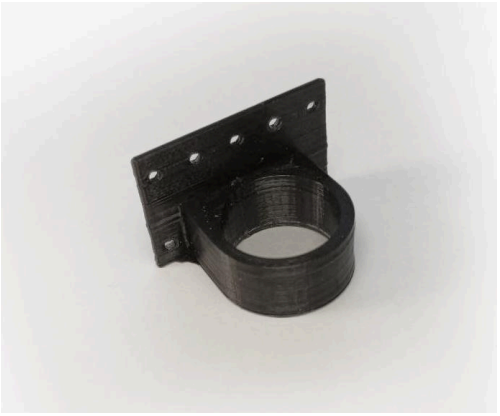


Figura 3: Soporte de marcador

Trayectorias

Teniendo en cuenta que se conoce la cinemática inversa para un punto cualquiera del espacio de trabajo del manipulador, resolver una trayectoria determinada para el robot no es más que calcular la cinemática inversa, uno a uno, de los puntos de la misma. Dado que en una trayectoria pueden encontrarse infinitos puntos, es necesario discretizar la misma en una cantidad razonable para que la trayectoria real recorrida por el puntero del manipulador no difiera mucho de la trayectoria teórica. Un ejemplo podría ser discretizar una circunferencia en 360 puntos y resolver la cinemática inversa para cada uno de ellos. Sin embargo, el movimiento de la terminal entre dos puntos puede describirse de varias formas, la cuales resultarán en que el movimiento sea fluido o espasmódico.

Para desplazar el extremo del manipulador de un punto A a un punto B dentro del espacio de trabajo existen varias estrategias, una alternativa puede ser que cada articulación se mueva a su velocidad máxima, por lo que algunas articulaciones pueden alcanzar su posición final antes que otras. Otra forma sería mover todas las articulaciones de manera proporcional a su recorrido para que todas comiencen y terminen simultáneamente. Cada estrategia tendrá ventajas y desventajas y la elección de cualquiera de ellas dependerá del objetivo del movimiento.

Cada trayectoria requerirá su análisis particular, ya que al discretizar la misma en pequeños tramos de igual longitud (en el espacio de trabajo), puede que la velocidad (de las articulaciones) en algunos de dichos tramos sobrepase los límites del actuador. Para evitar esto, una alternativa puede ser suavizar la zona donde se dan las mayores velocidades de la trayectoria, lo que se puede conseguir discretizando en tramos de menor longitud (pero de igual tiempo) en dichas zonas.

Trayectorias en el espacio articular

Para la generación de trayectorias (en el espacio articular) se utilizarán los siguientes interpoladores:

- **Interpolador de tercer orden**

  Asumiendo que se conoce la **configuración inicial y la configuración final** a la que se desea llevar la terminal del robot, se puede conocer (a partir de cinemática inversa) la posición inicial y final de cada articulación.
  Además, se deberá imponer la **velocidad en dichos puntos**, que puede ser “reposo” o una velocidad de “paso”.

  Considerando una articulación cuya posición y velocidad, para el instante inicial y el final son:

$$\theta(t_i) = \theta_i \qquad \dot{\theta}(t_i) = 0$$
$$\theta(t_f) = \theta_f \qquad \dot{\theta}(t_f) = 0$$

  Puede imponerse una ***trayectoria polinomial de tercer orden*** entre los puntos extremos, encontrando los coeficientes de la siguiente ecuación que satisfacen las condiciones iniciales

y finales.

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

De esta forma se obtienen los valores que debe tomar cada constante del polinomio para cada articulación, de forma de alcanzar el punto deseado mediante el movimiento polinomial de cada articulación.

- **Interpolador de quinto orden:**

  En el caso donde las ***aceleraciones requeridas pueden no ser alcanzables*** por los actuadores que se poseen, o simplemente se desea controlar además, la aceleración durante la trayectoria, limitando sus mínimos, máximos y dándole continuidad, es posible utilizar ***interpoladores de quinto orden***, donde la posición, velocidad y aceleración quedan definidas como sigue:

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4$$

$$\ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3$$

  Donde como incógnitas se tienen **6 constantes** para cada tramo de trayecto, y por lo tanto se necesitan 6 condiciones para determinar los **coeficientes c_0 a c_5.**

  Estas por lo general son: las posiciones, velocidades y aceleraciones al comienzo y al final:

$$\theta(t_i) = \theta_i \quad \dot{\theta}(t_i) = 0 \quad \ddot{\theta}(t_i) = \ddot{\theta}_i$$
$$\theta(t_f) = \theta_f \quad \dot{\theta}(t_f) = 0 \quad \ddot{\theta}(t_f) = \ddot{\theta}_f$$

- **Interpolador trapezoidal (lineal con extremos parabólicos):**

  Una alternativa a los polinomios de orden alto (3er o 5to) es usar una ***interpolación lineal*** (polinomio de primer orden) entre el punto de salida y el de llegada, sin embargo, ***no se recomienda*** este tipo de interpolación puesto que implica cambio bruscos de velocidad en los puntos, y altas aceleraciones, provocando movimientos espasmódicos.

  Como solución a estos cambios bruscos, se propone ***transformar el polinomio lineal en una parábola*** al comienzo de la trayectoria y próximo al final, creando una curva continua de posición y velocidad. De esta forma, se definirá un t_b que es el "tiempo de combinación" (blending time) hasta el cual la curva es parabólica, luego el movimiento será lineal hasta t_f-t_b donde comenzará nuevamente la misma parábola, pero invertida (siempre que se considere simétrica que es lo usual).

La trayectoria queda definida por una función partida:

$$\theta(t) = \begin{cases} c_{0p} + c_{1p}t + \frac{1}{2}c_{2p}t^2 & \text{si: } t \in \text{ parabola} \\ c_{0L} + c_{1L}t & \text{si: } t \in \text{ lineal} \end{cases}$$

De t=0 a t=t_b se tiene una parábola para la cual se deben conocer ciertas condiciones para determinarla, por ejemplo: posición inicial, velocidad inicial y **se puede imponer una aceleración (que será constante) como la velocidad de rotación ω que se desea en el tramo constante dividido entre t_b.**

$$\begin{cases} \theta(t_i) = \theta_i \\ \dot{\theta}(t_i) = 0 \\ \ddot{\theta}(t_i) = \omega/t_b \end{cases}$$

De aquí se obtiene que para la zona parabólica (t<t_b):

$$\theta(t) = \theta_i + \frac{\omega}{2t_b}t^2$$
$$\dot{\theta}(t) = (\omega/tb)t$$
$$\ddot{\theta}(t) = \omega/t_b$$

Luego para determinar la trayectoria en la zona lineal se deberá igualar la expresión de la posición y de la velocidad en t_b, tanto desde la expresión parabólica como en la lineal, para que el resultado sea continuo.

$$\dot{\theta_p}(t_b) = \dot{\theta_L}(t_b)$$

$$\frac{\omega}{t_b}t_b = \omega = c_{1L}$$

Obteniendo:

$$\theta(t) = \theta_i - \frac{\omega}{2}t_b + \omega t$$
$$\dot{\theta}(t) = w$$
$$\ddot{\theta}(t) = 0$$

La parábola final es simétrica a la primera, pero con una aceleración negativa. Entonces la parábola final (de t=t_f-t_b a t=t_f) puede expresarse de la siguiente forma:

$$\theta(t) = \theta_f - \frac{\omega}{2t_b}(t_f - t)^2$$
$$\dot{\theta}(t) = \frac{\omega}{t_b}(t_f - t)$$
$$\ddot{\theta}(t) = -\frac{\omega}{t_b}$$

Trayectorias en el espacio de trabajo

Obtener las trayectorias en el espacio de trabajo se refiere a expresar el movimiento del robot relativo al sistema de referencia cartesiano global (fijo en algún punto del laboratorio), representando la posición y la orientación de la terminal del robot.
Las trayectorias en línea recta, son por excelencia las más utilizadas, sin embargo pueden usarse cualquier otro tipo de generador de trayectorias, de hecho, todas aquellas que se vieron para el espacio articular aplican para representar el movimiento de la terminal.
La principal diferencia entre ambas definiciones es que al utilizar el espacio cartesiano, se deben utilizar repetidamente las ecuaciones de la cinemática inversa para determinar los valores de las variables articulares a imponer en cada actuador del robot para reproducir el movimiento.
Nuevamente, la cantidad de puntos utilizados para representar la trayectoria repercutirá en la precisión con la que se aproximará la terminal a la trayectoria esperada, dado que (despreciando el error de posicionamiento del sistema) los únicos lugares que se asegura que la terminal alcanza son los puntos definidos.

El procedimiento para determinar las trayectorias articulares en función de la trayectoria deseada en la terminal se puede escribir como un loop como se muestra a continuación:

- Asumiendo que en el estado actual (t_K) se conoce la posición de todas las articulaciones.
- Asumiendo que se conoce la función P(t) que indica la posición (y orientación) de la terminal para todo t.
- Entonces para conocer los valores en el instante t_{K+1}

1. Incrementar el tiempo: $t_{K+1}=t_K+\Delta t$
2. Calcular posición y orientación de la terminal [$P(t_{K+1})$]
3. Calcular los valores articulares [$q_i(t_{K+1})$] para el estado $P(t_{K+1})$ a través de cinemática inversa
4. Guardar valores
5. Si $t_{K+1} < t_{FINAL} \rightarrow$ Volver al punto 1.

Una vez que se conocen los valores $q_i(t_K)$ para todas las articulación (i = 1,...,n) y todos los tiempos t_K (k=0,..., k_{FINAL}) se tendrá un vector para cada articulación que representará la variación discreta en función del tiempo de la posición de cada junta. Con esto, se podrá determinar una trayectoria continua también en función del tiempo $q_i(t)$, que podrá ser programada, derivada para calcular otras magnitudes, etc.

Código

Se utilizará el código para la cinemática inversa del manipulador de 4GDL (resuelto en el laboratorio 2) para lograr el movimiento del puntero en la trayectoria deseada. En este caso, al reutilizar el código de Cinemática Inversa, se mantiene también la restricción de que el último eslabón debe permanecer horizontal.

Para ello se hará uso de nuevas herramientas de Python como la definición de una función. En Python (y en programación en general), una función es un bloque de código reutilizable que realiza una tarea específica y de manera independiente al resto del programa. Las funciones permiten organizar el código en secciones lógicas, evitar repeticiones y hacerlo más legible y mantenible. En el anexo puede verse un ejemplo de trayectoria parabólica para el cual, modificando algunos detalles, pueden obtenerse movimientos para diversas trayectorias.

Para la resolución de la función “circunferencia”, se hará uso del esquema mostrado en la figura 4, donde el sistema de coordenadas será el mismo que conforma el de la base del manipulador (S_0) utilizado en el laboratorio. Dicha función utilizará las coordenadas cartesianas (x, y, z) para cada punto de la circunferencia, que se puede describir en función del centro, el Radio y el angulo..
Notar que el centro de la circunferencia se encuentra en un punto cualquiera del espacio de trabajo útil distinto del origen del sistema de coordenadas.

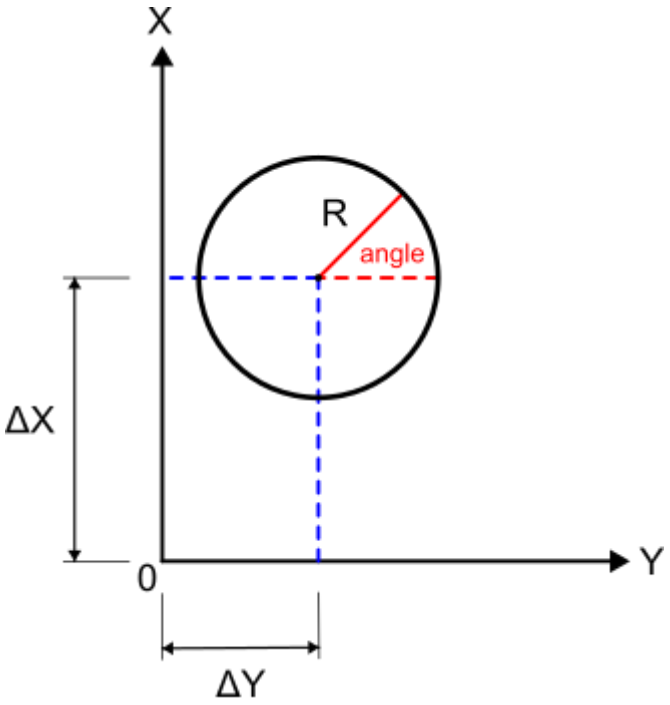


Figura 4: Esquema de circunferencia

Descripción de tareas:

1. Resolución de trayectoria circular
2. Desarrollo de dichas resoluciones en Python
3. Conexión del manipulador a la PC y verificación de su funcionamiento
4. Elección de parámetros correspondientes (ubicación en el espacio del centro de la cfa., radio, velocidad de trabajo, etc.)
5. Verificar el correcto funcionamiento del manipulador dibujando la trayectoria en la estación de trabajo con el marcador colocado en su soporte

Anexo

Ejemplo (parábola)

```
import math as m

speed = 150
step = 1

A=(1.0/50.0) #Parametros de la parabola
B=0
C=100

a1 =
a2 =
a3 =
a4 =

#Definicion de la funcion de nombre puntoPbla que recibe los parametros
recorrdoy y deltaZ
#Devuelve el array de coordenadas del punto (x,y,z)
def puntoPbla(recorridoy, deltaZ):
    y = recorridoy
    x = A*(y**2)+B*y+C
    z = deltaZ
    pto = [x, y, z]

    return pto


def cinematicaInversa(x, y, z):...#Insertar resolucion de cinematica
inversa para 4GDL


    #T en posicion del 0 al 1023
    T1 = (T1*(1023/300)+(1023/2))
```

```
    T2 = (T2*(1023/300)+(1023/2))
    T3 = (T3*(1023/300)+(1023/2))
    T4 = (T4*(1023/300)+(1023/2))

    return {1: T1, 2: T2, 3: T3, 4: T4}


array_ptos_pbla = [] #Inicializacion de un array vacio para cargarle
los datos

for punto_pbla in range(-50, 50, step):
    array_ptos_pbla.append(puntoPbla(punto_pbla, deltaZ)) #Se agrega un
valor por cada iteracion relativa al range


for i in array_ptos_pbla:
    final_pos = cinematicaInversa(i[0], i[1], i[2]) #Una vez cargados
los valores se le aplica la cinematica inversa a cada valor

    for j in range(4):
        chain.goto(j + 1, final_pos[j + 1], speed=speed,
blocking=False) #Ver apendice Lab 2
```

Código Cfa.:

```
import math as m

speed = 150
deltaX =     # Distancia en x desde el origen hasta el centro de la cfa
deltaY =     # Distancia en y desde el origen hasta el centro de la cfa
deltaZ =     # Distancia en z desde el origen hasta el centro de la cfa
step = 5      # Dimension de la discretizaccion
radio =   # Radio de la cfa
max_angulo_cfa = 360

# Pueden agregarse mas grados

a1 =
a2 =
a3 =
a4 =

# Funcion que devuelve las coordenadas de un punto
def puntoCfa(angle, radio, deltaX, deltaY, deltaZ):
    x =
    y =
    z =
    pto = [x, y, z]

    return pto


def cinematicaInversa(x, y, z):...#Insertar resolucion de cinematica
inversa para 4GDL



    # T1 en posicion del 0 al 1023 que trae el motor
    T1 = (T1*(1023/300)+(1023/2))
    T2 = (T2*(1023/300)+(1023/2))
    T3 = (T3*(1023/300)+(1023/2))
    T4 = (T4*(1023/300)+(1023/2))
```

```
    return {1: T1, 2: T2, 3: T3, 4: T4}


array_ptos_cfa = [] #Inicializacion de un array vacio para cargarle los
datos
# punto_cfa va de 0 a 360

for punto_cfa in range(0, max_angulo_cfa, step):
    array_ptos_cfa.append(puntoCfa(punto_cfa, radio, deltaX, deltaY,
deltaZ)) #Se agrega un valor por cada iteracion relativa al range

for i in array_ptos_cfa:
    final_pos = cinematicaInversa(i[0], i[1], i[2]) #Una vez cargados
los valores se le aplica la cinematica inversa a cada valor

    for j in range(4):
        chain.goto(j + 1, final_pos[j + 1], speed=speed,
blocking=False)
```