MT3005 - ROBÓTICA I

# Proyecto No. 1: Diseño y Control Cinemático de un Cuadrúpedo (Simulación)


María Fernana Girón (16820)
Domenico Dellachiessa (16036)
Jacqueline Guarcax (16142)
Eduardo Santizo Olivet (16089)
Sección 10
10 de mayo de 2020

# I. Matriz Denavit-Hartenberg

## 1. Modelo del Robot 3D para patas delanteras

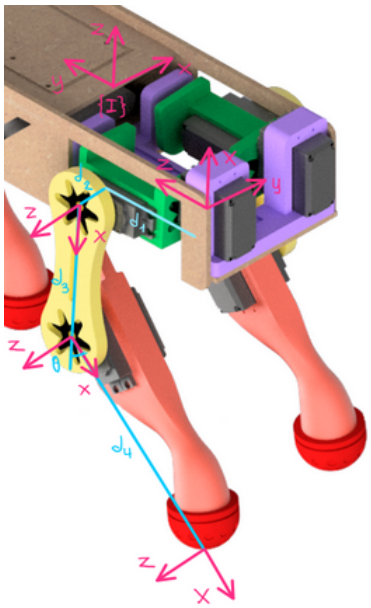



Figura 1: Patas delanteras del cuadrúpedo

| q | $\theta$ | $b$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| q1 | 0 | d1 | 0 | $\pi/2$ |
| q2 | $\pi$ | d2 | d3 | 0 |
| q3 | $\pi/6$ | 0 | d4 | 0 |

Cuadro 1: Parámetros Denavit-Hartenberg

Dimensiones:

$$d1 = 50mm$$

$$d2 = 15mm$$

$$d3 = 80mm$$

$$d4 = 129mm$$

## 2. Modelo del Robot 2D para patas traseras

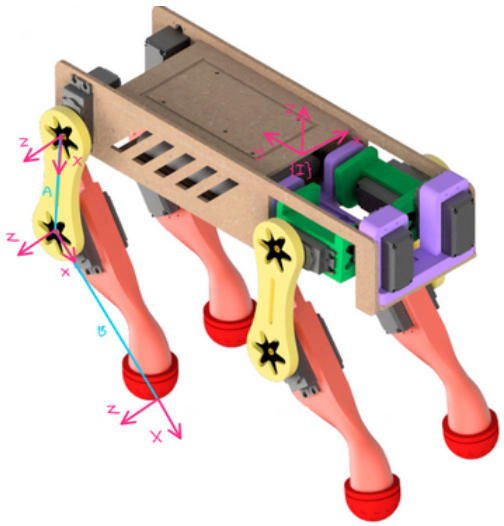


Figura 2: Patas traseras del cuadrúpedo

| q | $\theta$ | $b$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| q1 | $\pi/6$ | 0 | A | 0 |
| q2 | $4\pi/6$ | 0 | B | 0 |

Cuadro 2: Parámetros Denavit-Hartenberg

Dimensiones:

$$A = 80mm$$

$$B = 129mm$$

Con los parámetros de Denavit-Hartenberg definidos se procedió a definir las patas como objetos *SerialLink* en matlab. Con los objetos definidos se utilizó la función *teach* para visualizar y manipular las patas, de esta forma se obtuvo una mejor visión del movimiento que se esperaba que realizaran.

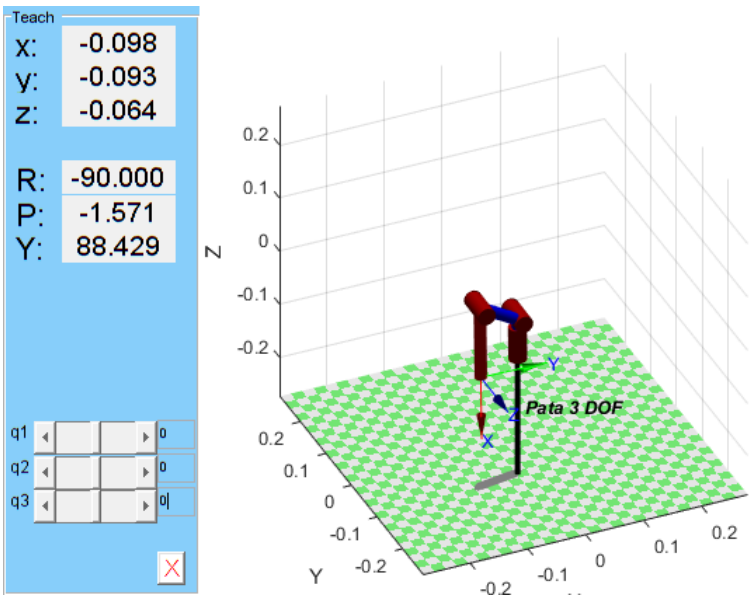



Figura 3: *SerialLink* pata delantera

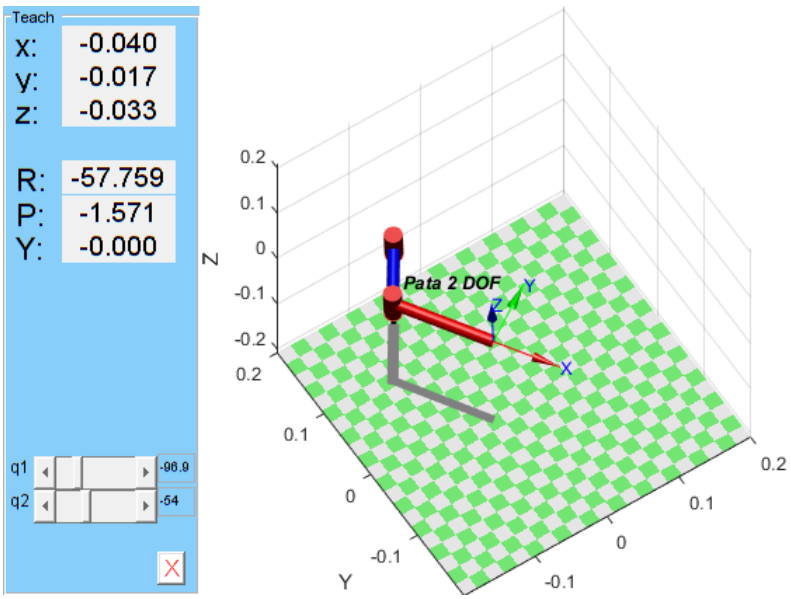



Figura 4: *SerialLink* pata trasera

# II. Cinemática Directa

Para ambos casos la cinemática directa se obtuvo con la matrices de Denavit-Hartenberg. Inicialmente se obtuvo la matriz de transformación homogénea $A_j$ para los ángulos de Denavit-Hartenberg.

$$\mathbf{A}_j \equiv^{j-1} \mathbf{T}_j = \mathrm{Rot}_z\left(\theta_j\right) \mathrm{Transl}_z\left(d_j\right) \mathrm{Transl}_x\left(a_j\right) \mathrm{Rot}_x\left(\alpha_j\right) \tag{1}$$

A partir de esto se obtuvo la cinemática directa en forma de matriz de transformación homogénea. Empleando la convención DH, en donde la cinemática directa se define como:

$$B\mathbf{T}_E(\mathbf{q}) = \prod_{j=1}^{N} \mathbf{A}_j\left(q_j\right) \tag{2}$$

Adicionalmente, se emplearon transformaciones de base y de herramienta, según el caso. De esta forma, se consideró la posición inicial de las patas del robot respecto al marco inercial. Y la posición del efector final de la pata, respecto al inicio de la misma.

## Modelo 3D

Matriz DH:

$$DH = \begin{bmatrix} q(1) & d1 & 0 & \mathrm{pi}/2 \\ q(2)+\mathrm{pi} & d2 & d3 & 0 \\ q(3)+\mathrm{pi}/6 & 0 & d4 & 0 \end{bmatrix}$$

Transformación de Base:

$$I\mathbf{T}_B = \begin{bmatrix} 0.9996 & 0.0274 & 0 & -0.0284 \\ -0.0274 & 0.9992 & 0.0274 & -0.0785 \\ 0.0008 & -0.0274 & 0.9996 & -0.0251 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

## Modelo 2D

Matriz DH:

$$DH = \begin{bmatrix} q(1)+\mathrm{pi}/6 & 0 & A & 0 \\ q(2)+4\mathrm{pi}/6 & 0 & B & 0 \end{bmatrix}$$

Transformación de Base:

$$I\mathbf{T}_B = \begin{bmatrix} 0.9981 & -0.0548 & -0.0274 & -0.0640 \\ 0.0548 & 0.9985 & 0 & 0.1588 \\ 0.0274 & -0.0015 & 0.9996 & -0.0341 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

# III. Cinemática Inversa

Se empleó la cinemática inversa numérica completa mediante un cálculo iterativo descrito en la ecuación 3. Lo que se busca es acumular las configuraciones pasadas y el error para llegar a la pose final deseada.

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{J}^{-1}\left(\mathbf{q}_k\right)\mathbf{e}_k \tag{3}$$

Para el cálculo de la pseudo inversa del jacobiano se seleccionó el de Levenberg-Marquardt 4. Esta decisión se tomó debido a que evita que el algoritmo presente problemas en las singularidades y provee una solución con mayor continuidad. Este comportamiento es útil porque se desea que el robot presente movimientos naturales.

$$\mathbf{J}^{\dagger} \rightarrow \mathbf{J}^{\top}\left(\mathbf{J}\mathbf{J}^{\top} + \lambda^2\mathbf{I}\right)^{-1} \tag{4}$$

En cuanto al cálculo del error se seleccionó una tolerancia de $1x10^{-6}$ para la posición y $1x10^{-5}$ para la orientación. El error está descrito por la siguiente ecuación:

$$\mathbf{e}_k = \begin{bmatrix} \mathbf{e}_p[k] \\ \mathbf{e}_o[k] \end{bmatrix} \tag{5}$$

En donde el error de posición esta definido conforme a 6 y el de orientación respecto a 8 derivado del cálculo efectuado mediante cuaterniones.

$$\mathbf{e}_k = \mathbf{o}_d - \mathbf{o}\left(\mathbf{q}_k\right) \tag{6}$$

$$\mathcal{Q}_e[k] = \mathcal{Q}_d * \mathcal{Q}^{-1}\left(\mathbf{q}_k\right) = \{\eta_e[k], \boldsymbol{\epsilon}_e[k]\} \tag{7}$$

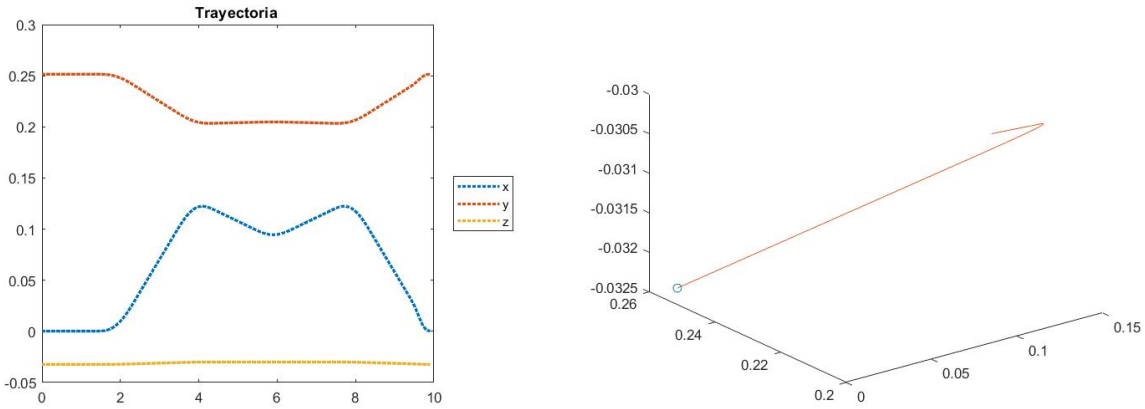
$$\mathbf{e}_o[k] = \boldsymbol{\epsilon}_e[k] \tag{8}$$

Cabe resaltar que si la configuración inicial se encuentra más cercana a la configuración final se obtendrán trayectorias más suaves. Esta idea es la que permite la generación de la trayectoria deseada dándole así mayor importancia a este procedimiento.

# IV. Generación de trayectoria

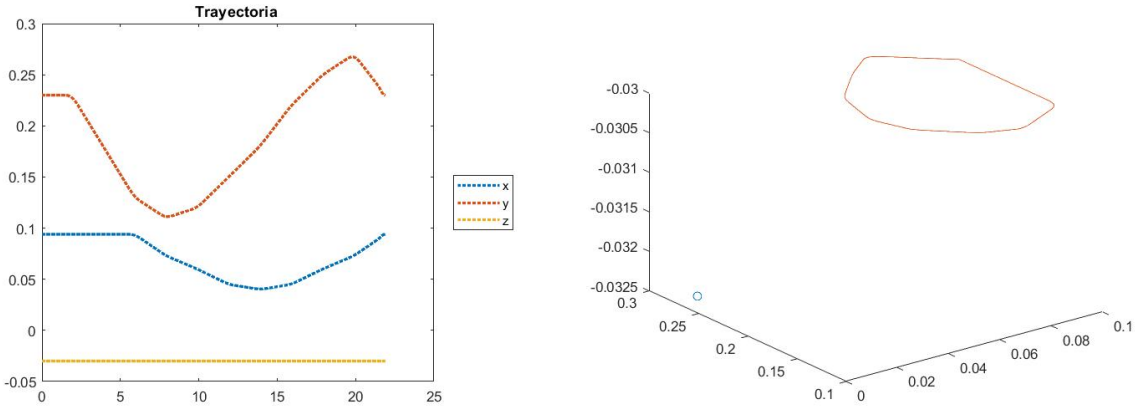
Para generar la trayectoria inicialmente se plantearon los puntos necesarios para generar el movimiento deseado para las patas delanteras y traseras. En el caso de las patas traseras, el movimiento es únicamente en dos dimensiones. Mientras que a las patas delanteras, se les agregó un grado más de libertad para mejorar la locomoción del cuadrúpedo.

Con los puntos establecidos se utilizó la función *mstraj* de la *Robotic Toolbox* para generar la trayectoria. Seguido de esto, se planteó la cinemática inversa para cada conjunto de coordenadas generado en la trayectoria y se procedió a almacenar las configuraciones resultantes.



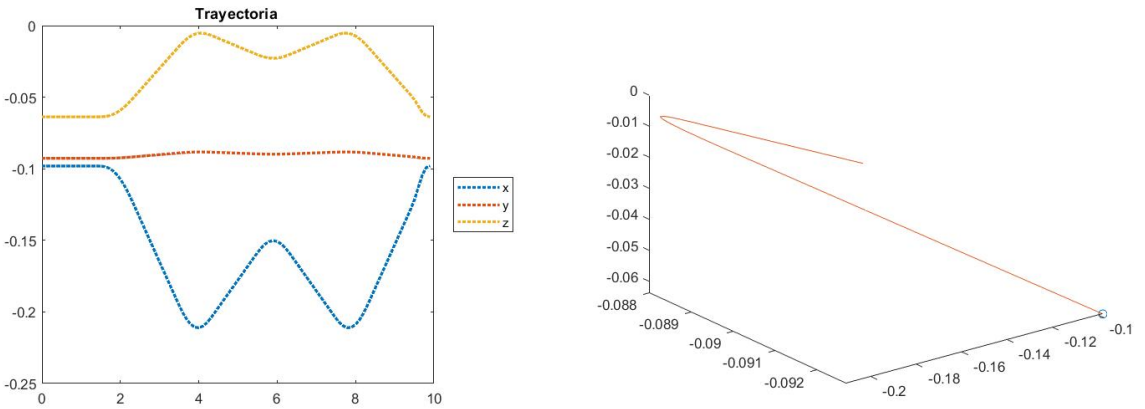

(a) Trayectoria para cada coordenada respecto al tiempo (b) Trayectoria multi-segmento

Figura 5: Simulación de la trayectoria del robot parándose




(a) Trayectoria para cada coordenada respecto al tiempo (b) Trayectoria multi-segmento

Figura 6: Simulación de la trayectoria del robot caminando




(a) Trayectoria para cada coordenada respecto al tiempo (b) Trayectoria multi-segmento

Figura 7: Simulación de la trayectoria para tercer grado de libertad