Cinemática de robots

Transformaciones lineales aplicadas a robots articulados

Badenes, Tomás
tomasbadenes@alu.ing.unlp.edu.ar

Fernández, Santiago Adriel
fernandez.adriel@alu.ing.unlp.edu.ar

Majoros, Lorenzo
lorenzomajoros@alu.ing.unlp.edu.ar

Seery, Juan Martín
juan.seery@alu.ing.unlp.edu.ar

Facultad de Ingeniería, Universidad Nacional de La Plata

1. Introducción

La cinemática de robots es la ciencia que estudia el movimiento de cadenas articuladas con múltiples grados de libertad. Se asocia con el estudio de cuerpos rígidos y hace fuerte uso de la geometría. Las técnicas descritas por esta disciplina se aplican en todo tipo de ámbitos, desde animaciones por computadora hasta la biomecánica. Excelentes ejemplos son los brazos utilizados en líneas de ensamblaje o los brazos robot utilizados para operar en el exterior de la estacion espacial internacional.

Aquí se tratará la solo parte más superficial del área: obtener ecuaciones que relacionan un punto en el espacio con, en este caso, rotaciones de motores (los llamados parámetros del robot). Esto es esencial, ya que nos permitiría saber cuánto hay que rotar los motores para que el robot llegue a un punto, a dónde está apuntando en un momento dado, cuál es el alcance del mismo, entre un sinfín de otros ejemplos.

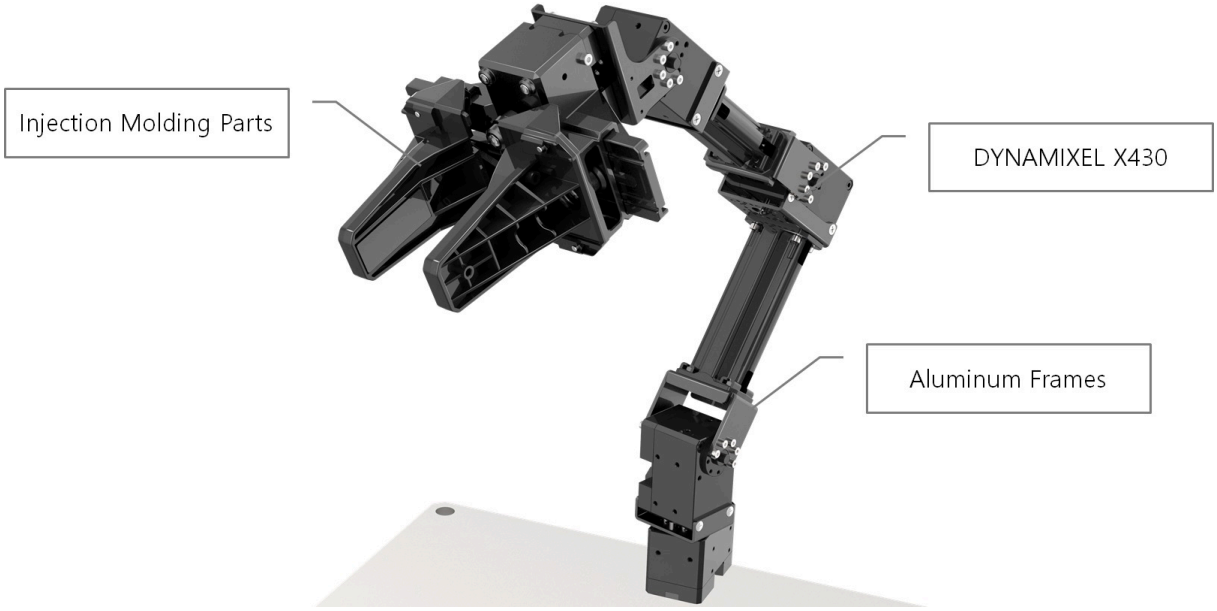

Figura 1: Brazo OpenMANIPULATOR-X

Se verá cómo modelar y controlar a partir de estas operaciones el robot **OpenMANIPULATOR-X**[1], un brazo robótico de 4 grados de libertad creado por la empresa ROBOTIS[2] para el proyecto TurtleBot 3[3] en conjunto a Open Robotics[4]. Se eligió este robot porque es de plataforma, hardware y software abierto, y provee un entorno de simulación. Además, está pensado como proyecto educativo y viene acompañado de extensa documentación, lo cual resulta ideal para el proyecto.

2. Forward kinematics – ¿a dónde estoy apuntando?

Un brazo robótico no es más que un conjunto de articulaciones unidas con enlaces. Para empezar, nos gustaría poder mover el robot. Físicamente, se necesita pasarle electrónicamente a cada uno de los rotores cuántos grados se quiere que roten y los mismos rotarán. ¿Cómo se podría calcular el punto en el que terminará el extremo del brazo dados los ángulos de los motores?

Para ejemplificar, se proponen tres puntos articulados $\{0\}, \{1\}, \{2\}$ unidos por enlaces L_1, L_2. L_i, numéricamente, es la distancia entre $\{i-1\}$ e $\{i\}$. $\{0\}$ se encuentra fija al suelo, y $\{0\}$ y $\{1\}$ pueden rotar sobre un plano. En este ejemplo (y en el robot utilizado) solamente se tienen articulaciones que rotan, lo cual simplifica bastante la teoría necesaria. Estas articulaciones también podrían expandirse, moverse de manera esférica o incluso helicoidal, que por suerte no es el caso.

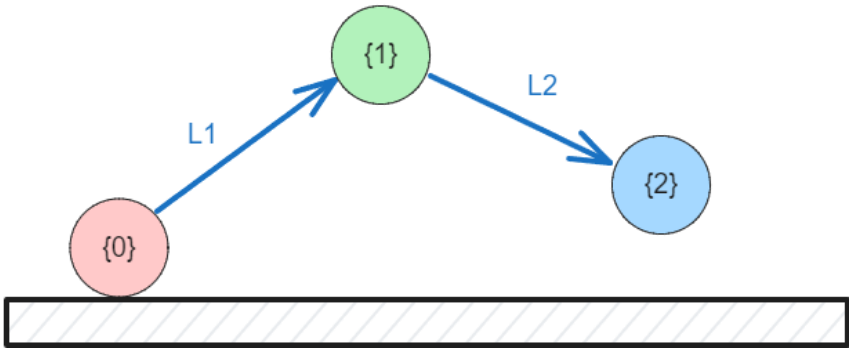

Figura 2: Puntos articulados en el plano XY.

Ahora, a cada articulación $\{i\}$ se le asignará su propio sistema de coordenadas dado por la base arbitraria $B_i = \left\{\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_i\right\}$. Cuando $\{i\}$ rote, también lo harán sus ejes y, por consiguiente, se moverán y trasladarán todos los ejes de las articulaciones $\{j\} : j > i$.

[1] https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview
[2] https://en.robotis.com
[3] https://www.turtlebot.com/turtlebot3/
[4] https://www.openrobotics.org

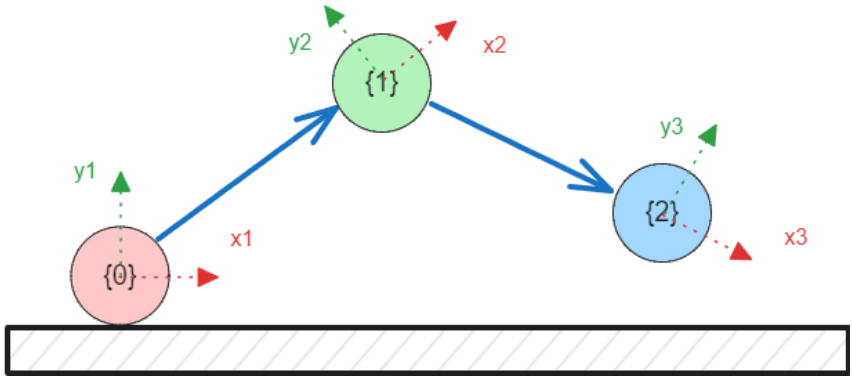

Figura 3: Los puntos articulados en el plano con sus respectivos ejes asociados.

Así, la estrategia será plantear transformaciones que nos permitan pasar de uno de estos sistemas (o espacios) a otro. Así, cuando se rote uno de los espacios, podremos ver dónde terminan los espacios siguientes.

Se plantean las transformaciones T_{i-1}^{i}, que nos permiten pasar de la «base» B_i a la «base» B_{i-1}. Pero esto es imposible. Estas transformaciones necesitan que se mueva el $\mathbf{0}$ de un espacio al siguiente, lo cual no es una transformación lineal.

2.1. Coordenadas homogéneas

Para solucionar esto, se utilizan las llamadas *coordenadas homogéneas* (Cox et al., 2015, cap. 8), que hacen uso de vectores de una dimensión más y los proyectan sobre el subespacio deseado. Al igual que las transformaciones lineales tradicionales nos permiten escalar e inclinar los ejes, y además nos permite trasladarlos (entre otras opciones).

Este tópico es muy extenso y admite muchos tipos de proyecciones. En nustro caso, se utilizarán matrices de este tipo:

$$H = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{d} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}$$

$\boldsymbol{R} \in \mathbb{R}^{n\times n}$ es análoga a una matriz transformación sobre el subespacio proyectado: representa escalación, rotación, inclinación, etc. $\boldsymbol{d} \in \mathbb{R}^n$ es un vector que representa un desplazamiento del origen del subespacio. Ejemplos prácticos en $\mathbb{R}^2$ son:

$$\text{Rotación}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{Traslación}(x) = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

que nos permiten rotar los ejes y trasladarlos en dirección el eje x respectivamente. Finalmente, para obtener los vectores de los subespacios transformados, se aplica

$$\mathbf{v}' = H\mathbf{v}$$
$$\begin{pmatrix} x_{1'} \\ x_{2'} \\ \vdots \\ x_{n'} \\ 1 \end{pmatrix} = H \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

quedando el vector deseado con una componente extra que será siempre 1. Cabe mencionar nuevamente que esta componente simpre queda 1 en este caso particular, y no para toda transformación en coordenadas homogéneas.

2.2. Parámetros de Denavit–Hartenberg

Ahora que tenemos un método para movernos del sistema de referencia de una articulación a la anterior con T_{i-1}^{i}, siendo esta traslaciones y rotaciones en el coordenadas homogéneas, la matriz transformación que nos «lleva» o nos «cambia de base» de la última unión a la base será:

$$T_0^n = T_0^1 \; T_1^2 \; ... \; T_{n-1}^n$$

¿Por qué se querría esta matriz? Porque $T_0^n \cdot (0, ..., 0, 1)^T$ será igual al origen de la punta de nuestro brazo (llamado *end-effector*) pero en las coordenadas de la base del robot (es decir, de nuestro espacio canónico). Nos permite saber dónde está posicionado el robot.

Se llama *forward kinematics* a este campo de estudio (Lunia et al., 2023, cap. 2). Hay varios métodos para obtener las matrices T_{i-1}^{i}, siendo el más popular en $\mathbb{R}^3$ el de Denavit–Hartenberg, conocido como *parámetros de Denavit–Hartenberg*. Consta de representar cada una de las matrices T_{i-1}^{i} en base a cuatro parámetros: una rotación en torno al eje z (θ), una traslación sobre el eje z (d), una traslación sobre el eje x (r) y una rotación en torno al eje x (α):

$$\begin{aligned} T_{i-1}^{i} &= T_{rz}(\theta_i) T_z(d_i) T_x(r_i) T_{rx}(\alpha_i) \\ &= \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & r_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & r_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

La ventaja de este método es que los parámetros son fáciles de obtener. Al momento de modelar, se piensa gráficamente qué rotaciones y traslaciones nos permiten transformar el origen de un sistema de referencia en otro. Particularmente, se suele pensar en la secuencia

1. rotación en torno al eje z (θ_i),
2. traslación sobre el eje z (d_i),
3. traslación sobre el eje x (r_i)
4. y rotación en torno al eje x (α_i)

que sufre el origen de $\{i-1\}$ para transformarse en el origen de $\{i\}$[5]. Nótese que la matriz que se obtiene es exactamente la inversa de esta transformación.

[5]Una visualización del método de obtención de estos parámetros se puede encontrar en youtu.be/rA9tm0gTln8.

Ahora, estas matrices T_{i-1}^{i} son estáticas. Se necesita que varíen según ángulos arbritarios. Las articulaciones del robot *OpenManipulator X*, como se planteó anteriormente, solamente rotan. Se puede plantear que todos nuestros motores rotan en torno a sus respectivos ejes z y que sus matrices sean

$$T_{i-1}^{i}(q_i) = T_{rz}(q_i + \theta_i)T_z(d_i)T_x(r_i)T_{rx}(\alpha_i)$$

siendo q_i los parámetros del robot y θ_i las condiciones iniciales.

2.3. Modelando el *OpenMANIPULATOR-X*

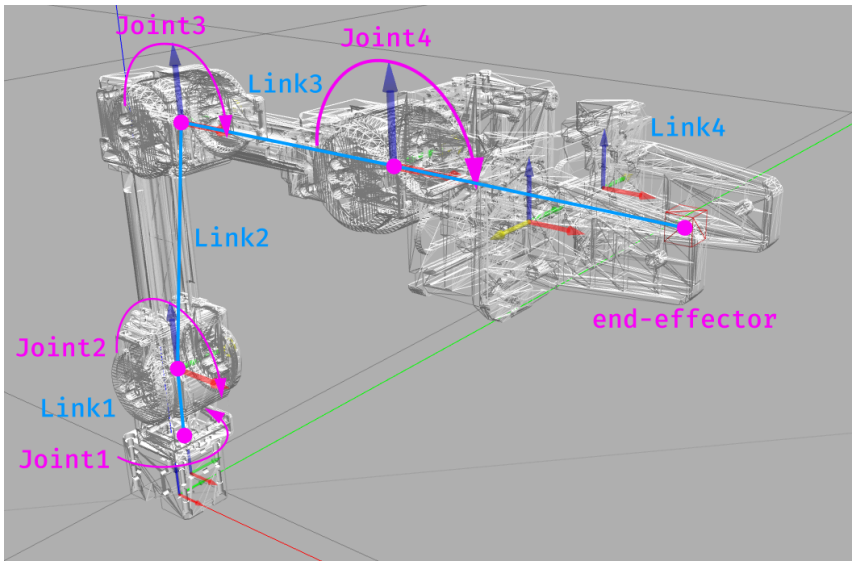

Figura 4: Representación digital del robot con anotaciones.

El robot consta de cuatro rotores. Así, los parámetros de Denavit–Hartenberg se pueden pensar gráficamente de la siguiente manera:

1. El primer rotor rota en torno al eje z de la base, por lo que $\theta_1 = q_1$ variable. Luego, se traslada L_1 sobre el eje z para ubicarse en el origen del siguiente rotor, por lo que $d_1 = L_1$ constante. Como no hay traslaciones sobre el eje x, $r_1 = 0$. Finalmente, como el eje de rotación del siguiente rotor es paralelo la piso, se realiza una rotación de $-\frac{\pi}{2}$ sobre el eje x para que el eje z (eje sobre el que se rotará el siguiente rotor) quede paralelo al piso, por lo que $\alpha_1 = -\frac{\pi}{2}$ constante. Nótese que también se podría rotar sobre $\frac{\pi}{2}$, pero se eligió esta configuración porque el modelo del brazo, como se muestra en la figura, rota con ángulos positivos de manera horaria.
2. El segundo rotor también rotará en torno a su eje z, por lo que $\theta_2 = q_2$ variable. Sin embargo, tal y como están planteados los ejes, la posición inicial del segmento 2 es paralelo al plano, por lo que hay que «levantarlo» para que quede como en la figura. Así, se agrega un desface $\varphi > 0$ tal que $\theta_2 = q_2 - \varphi$. A simple vista, se aprecia que $\varphi \approx \frac{\pi}{2}$, aunque en realidad el valor es ligeramente menor. Como el origen del próximo rotor está sobre el mismo plano que éste, el eje z se mantiene igual, por lo que $d_2 = \alpha_2 = 0$. Finalmente, se traslada L_2 sobre el eje x para ubicarse en el origen del siguiente rotor, por lo que $r_2 = L_2$ constante.
3. Similarmente, el tercer rotor también rotará en torno a su eje z, por lo que $\theta_3 = q_3$ variable. También se tiene en cuenta el desface anteriormente mencionado, solo que esta vez se aplica su opuesto, quedando $\theta_3 = q_3 + \varphi$. Como el origen del próximo rotor está sobre el mismo plano que este, el eje z se mantiene tal cual, por lo que $d_3 = \alpha_3 = 0$. Finalmente, se traslada L_3 sobre el eje x para ubicarse en el origen del siguiente rotor, por lo que $r_3 = L_3$ constante.

4. Por último, el cuarto rotor también rotará en torno a su eje z, por lo que $\theta_4 = q_4$ variable. Ahora, para llegar al *end-effector* (que no es más que un punto arbitario, no un rotor), solo hace falta trasladarse L_4 sobre el eje x, por lo que $r_4 = L_4$ constante y $d_4 = \alpha_4 = 0$.

Enlace	θ_i	d_i	r_i	α_i
1	q_1	L_1	0	$-\frac{\pi}{2}$
2	$q_2 - \varphi$	0	L_2	0
3	$q_3 + \varphi$	0	L_3	0
4	q_4	0	L_4	0

Tabla 1: Parámetros de Denavit–Hartenberg obtenidos.

Así, cada una de las matrices intermedias quedan

$$T_0^1(q_1) = T_{rz}(q_1)T_z(L_1)T_{rx}\left(-\frac{\pi}{2}\right) = \begin{pmatrix} \cos(q_1) & 0 & -\sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_1^2(q_2) = T_{rz}(q_2 - \varphi)T_x(L_2) = \begin{pmatrix} \cos(q_2-\varphi) & -\sin(q_2-\varphi) & 0 & L_2\cos(q_2-\varphi) \\ \sin(q_2-\varphi) & \cos(q_2-\varphi) & 0 & L_2\sin(q_2-\varphi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_2^3(q_3) = T_{rz}(q_3 + \varphi)T_x(L_3) = \begin{pmatrix} \cos(q_3+\varphi) & -\sin(q_3+\varphi) & 0 & L_3\cos(q_3+\varphi) \\ \sin(q_3+\varphi) & \cos(q_3+\varphi) & 0 & L_3\sin(q_3+\varphi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_3^4(q_4) = T_{rz}(q_4)T_x(L_4) = \begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 & L_4\cos(q_4) \\ \sin(q_4) & \cos(q_4) & 0 & L_4\sin(q_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Luego de todo esto, finalmente se puede obtener la posición del *end-effector* con la función:

$$\begin{aligned} P(q_1, q_2, q_3, q_4) &= \left(T_0^1(q_1)\ T_1^2(q_2)\ T_2^3(q_3)\ T_3^4(q_4)\right)\ (0,0,0,1)^T \\ &= T_0^4(q_1, q_2, q_3, q_4)\ (0,0,0,1)^T \end{aligned}$$

3. Inverse kinematics – Obteniendo los parámetros

Es muy interesante aprender sobre *forward kinematics*, pero realmente no tiene demasiadas aplicaciones. En el espacio de la robótica resulta esencial poder realizar el procedimiento inverso: averiguar los ángulos necesarios para que el brazo se ubique en una posición deseada. Obtener esta función inversa se llama *inverse kinematics* (Lunia et al., 2023, cap. 3).

Al buscar la función inversa, no siempre se podrá obtener en forma analítica, ya que hay configuraciones de robots que permiten varios (o hasta infinitos) conjuntos de ángulos para alcanzar un punto. En ese caso, se suele el método de aproximación de Newton–Raphson para obtener una solución numérica.

En el caso del *OpenManipulator X*, se logró obtener una solución analítica.

3.1. Obteniendo la solución

Para empezar, se obtiene la forma analítica de T_0^4, que se puede dividir en dos partes más sencillas: $T_0^4 = T_0^1\ T_1^4$. T_1^4, haciendo un poco de trigonometría, se obtiene:

$$\omega = q_2 + q_3 + q_4$$

$$T_1^4(q_2, q_3, q_4) = \begin{pmatrix} \cos(\omega) & -\sin(\omega) & 0 & L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega) \\ \sin(\omega) & \cos(\omega) & 0 & L_2\sin(q_2-\varphi) + L_3\sin(q_2+q_3) + L_4\sin(\omega) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplicando por T_0^1 se obtiene

$$T_0^4(q_1, q_2, q_3, q_4) =$$

$$\begin{pmatrix} \cos(q_1)\cos(\omega) & -\cos(q_1)\sin(\omega) & -\sin(q_1) & L_2\cos(q_1)\cos(q_2-\varphi) + L_3\cos(q_1)\cos(q_2+q_3) + L_4\cos(q_1)\cos(\omega) \\ \sin(q_1)\cos(\omega) & -\sin(q_1)\sin(\omega) & \cos(q_1) & L_2\sin(q_1)\cos(q_2-\varphi) + L_3\sin(q_1)\cos(q_2+q_3) + L_4\sin(q_1)\cos(\omega) \\ -\sin(\omega) & -\cos(\omega) & 0 & L_1 - L_2\sin(q_2-\varphi) - L_3\sin(q_2+q_3) - L_4\sin(\omega) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Esta monstruosidad que se sale de los márgenes del papel es la matriz de transformación que nos permite permite obtener la posición del *end-effector* a partir de los parámetros del robot (la función $P(q_1, q_2, q_3, q_4)$ definida más arriba). Particularmente, nos queda la siguiente solución:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T_0^4(q_1, q_2, q_3, q_4)\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(q_1)(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)) \\ \sin(q_1)(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)) \\ L_1 - L_2\sin(q_2-\varphi) - L_3\sin(q_2+q_3) - L_4\sin(\omega) \\ 1 \end{pmatrix}$$

Además de las coordenadas (x, y, z) que se quieren alcanzar, para resolver el sistema de ecuaciones también tendremos el ángulo de ataque del brazo. Esto es, cuando el robot se mueva a un punto, puede llegar a este de varios ángulos distintos. No casualmente, este ángulo será ω. Gráficamente, se puede deducir que $\omega = q_2 + q_3 + q_4$, es decir, la suma de todas las rotaciones. Además, como q_1 solo rota sobre el eje z no afecta a este ángulo. (Nótese que ω positivo es cuando el *end-effector* rota en sentido horario, y negativo cuando rota en sentido antihorario.)

Así, nos queda el sistema de ecuaciones:

$$\begin{pmatrix} x \\ y \\ z \\ \omega \end{pmatrix} = \begin{pmatrix} \cos(q_1)(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)) \\ \sin(q_1)(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)) \\ L_1 - L_2\sin(q_2-\varphi) - L_3\sin(q_2+q_3) - L_4\sin(\omega) \\ q_2 + q_3 + q_4 \end{pmatrix}$$

Dividiendo las primeras dos ecuaciones, se obtiene

$$\frac{y}{x} = \frac{\sin(q_1)}{\cos(q_1)} = \tan(q_1)$$
$$q_1 = \arctan\left(\frac{y}{x}\right)$$

Obtenido q_1[6], veremos que lo podemos eliminar del sistema de la siguiente manera:

$$\begin{aligned}\sqrt{x^2+y^2} &= \sqrt{\left(\cos(q_1)^2 + \sin(q_1)^2\right)\left(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)\right)^2} \\ &= \sqrt{\left(L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)\right)^2} \\ \pm\sqrt{x^2+y^2} &= L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega)\end{aligned}$$

Así, se reduce el sistema a

$$\begin{cases}\pm\sqrt{x^2+y^2} = L_2\cos(q_2-\varphi) + L_3\cos(q_2+q_3) + L_4\cos(\omega) \\ z = L_1 - L_2\sin(q_2-\varphi) - L_3\sin(q_2+q_3) - L_4\sin(\omega) \\ \omega = q_2 + q_3 + q_4\end{cases}$$

¡que no es más que un problema de $\mathbb{R}^2$! Lo cual tiene sentido, ya que los rotores 2, 3 y 4 están contenidos en un mismo plano. Resolviendo el sistema con métodos trigonométricos[7], los ángulos finales quedan:

$$\begin{aligned}r &= \pm\sqrt{x^2+y^2} - L_4\cos(\omega) \\ h &= L_1 - z - L_4\sin(\omega)\end{aligned}$$

$$q_2 = \varphi + \arctan\left(\frac{h}{r}\right) \pm \arccos\left(\frac{r^2+h^2+L_2^2-L_3^2}{2L_2\sqrt{r^2+h^2}}\right)$$
$$q_3 = \arctan\left(\frac{h - L_2\sin(q_2-\varphi)}{r - L_2\cos(q_2-\varphi)}\right) - q_2$$
$$q_4 = \omega - q_2 - q_3$$

Esto no queda de forma matricial porque las soluciones no siempre son únicas y cuando no lo son suelen ser finitas, principalmente porque hay funciones trigonométricas de por medio.

Así, para aplicar los contenidos tratados, planteamos controlar una simulación del robot OpenMANIPULATOR-X a partir del desarrollo obtenido anteriormente.

4. Simulando el robot

El proyecto práctico realizado aplicando la teoría anterior se basa en ROS[8] (*Robot Operating System*), un set de herramientas ampliamente utilizado en el campo de la robótica para el desarrollo de software. Se creó un entorno basado en Docker para facilitar la portabilidad del proyecto en el que se incluye la simulación en Gazebo[9] provista por Robotis y el programa que se desarrolló.

[6] Al momento de obtener el q_1 computacionalmente, se utiliza la función `atan2(y,x)` que tiene en consideración el caso $x = 0$ y retorna el ángulo correspondiente a su lugar en el plano en vez de un ángulo entre $-\frac{\pi}{2}$ y $\frac{\pi}{2}$.

[7] Una explicación paso a paso puede verse en youtu.be/1-FJhmey7vk.

[8] https://www.ros.org

[9] https://gazebosim.org/home

Este programa es un CLI (*command-line interface*) permite el ingreso de coordenadas cartesianas que son transformadas a parámetros y enviadas al robot. Esta CLI puede ser utilizada tanto para controlar el robot simulado como un robot real. La función de ingreso de posición recibe cuatro datos, la posición en los ejes x, y y z y el ángulo del *end-effector* al momento de llegar a la posición destino.

El proyecto desarrollado incluyendo el entorno de simulación se encuentra disponible en github.com/b-Tomas/robot-kinematics de manera abierta.

Bibliografía

Cox, D. A., Little, J., & O'Shea, D. (2015). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra* (4.ª ed.). Pearson Educación. https://doi.org/10.1007/978-3-319-16721-3

Lunia, A., Stevens, A., Holt, C., Morgan, R., Norris, J., Poyyamozhi, S., & Zhong, Y. (2023 21). *Modeling, Motion Planning, and Control of Manipulators and Mobile Robots*. Springer International Publishing. https://opentextbooks.clemson.edu/wangrobotics

---

Este documento fue creado con Typst 0.9.0.
Última modificación realizada el 18/11/2023.