Robótica Industrial
Reto: Programación y Simulación de Baxter

Elkin Javier Guerra Galeano
Universidad EIA
Medellín, Colombia
elkin02011@hotmail.com

***Abstract*—En este escrito se explicara en detalle el proceso que se realizó, para el desarrollo de un programa en Matlab, el cuál simulara el comportamiento del robot Baxter de la empresa Rethink Robotics, primero se analizarán en detalle los cálculos de las cinemáticas directa e inversa del robot, enunciando las simplificaciones y condiciones que se consideraron para la realización de estos modelos, y luego se pasara a una explicación del código y su implementación.**

***Index Terms*—Baxter, Denavit–Hartenberg, Simulación, Programación , Cinemática directa, Cinemática inversa, Grados de libertad.**

I. Introducción

El robot Baxter es considerado como un robot de carácter colaborativo, debido a que esta diseñado para trabajar en conjunto con los seres humanos, es un robot de la compañía *Rethink Robotics*, la cual es una compañía que se especializo en el desarrollo de estos robots de servicio.
El principal objetivo de este escrito es describir el procedimiento que se realizó para el desarrollo, programación y simulación del robot Baxter en Matlab, en un principio se analizarán en detalle los modelos de cinemática directa y cinemática inversa que se plantearon para este robot, con estos conceptos como base principal de conocimiento se pasara a explicar el desarrollo y estructura del código implementado en Matlab para la simulación de Baxter.

II. Cinemática directa

A. *TCD utilizando 7 DOF*

En un principio se realizó al análisis de Denavit-Hartenberg a Baxter considerando todos sus grados de libertad, como bien se sabe este robot es un robot hiper-redundante, debido a que posee mas de 6 grados de libertad.

En la figura 10, se puede apreciar un diagrama simplificado de uno de los brazos del robot Baxter, en este se encuentran definidos el sentido positivo de cada grado de libertad, junto con los sistemas coordenados correspondientes a cada grado de libertad. A partir del esquema anteriormente mencionado se construyo la tabla de parámetros de Denavit-Hartenberg que se puede apreciar en la figura 1.

$i \rightarrow (i+1)$	α_{i-1}	a_{i-1}	d_i	θ_i
$0 \rightarrow 1$	0	0	0	θ_1
$1 \rightarrow 2$	$-\frac{\pi}{2}$	L_1	0	$\theta_2 + \frac{\pi}{2}$
$2 \rightarrow 3$	$\frac{\pi}{2}$	0	L_2	θ_3
$3 \rightarrow 4$	$-\frac{\pi}{2}$	L_3	0	θ_4
$4 \rightarrow 5$	$\frac{\pi}{2}$	0	L_4	θ_5
$5 \rightarrow 6$	$-\frac{\pi}{2}$	L_5	0	θ_6
$6 \rightarrow 7$	$\frac{\pi}{2}$	0	0	θ_7

Fig. 1: Tabla parámetros D-H usando 7 DOF.

B. *TCD utilizando 6 DOF*

Es de suma importancia tener presente que como el robot Baxter posee 7 grados de libertad, realizar una solución analítica de la cinemática inversa de forma analítica es imposible, debido a esto también se realizo un análisis de la cinemática directa de este robot, fijando el tercer grado de libertad siempre en un valor de cero, esto en vistas a poder utilizar este análisis en un futura en la simulación que se realizara del robot, debido a que para la simulación son necesarios tanto el análisis directo como el inverso de la cinemática de Baxter. En la figura 11, se puede apreciar el esquemático de uno de los brazos de Baxter, junto con la definición de todos los sistemas coordenados referentes a los grados de libertad que se utilizaran en el análisis, es importante aclara que una de las medidas a cambiado con respecto al esquema utilizando todos los grados de libertad, en esta ocasión se cambio un poco la posición de *home* del robot, y se utiliza la distancia lh, la cual se define como $\sqrt{l2^2 + l3^2}$. De igual manera que se hizo con el sistema de los 7 grados de libertad, también se dedujo y construyo la tabla de los parámetros de Denavit-Hartenverg para el modelo que solo tiene en cuenta 6 grados de libertad, esto se puede apreciar en la figura 2.

$i \rightarrow (i+1)$	α_{i-1}	a_{i-1}	d_i	θ_i
$0 \rightarrow 1$	0	0	0	θ_1
$1 \rightarrow 2$	$-\frac{\pi}{2}$	L_1	0	θ_2
$2 \rightarrow 3$	0	L_h	0	$\theta_4 + \frac{\pi}{2}$
$3 \rightarrow 4$	$\frac{\pi}{2}$	0	L_4	θ_5
$4 \rightarrow 5$	$-\frac{\pi}{2}$	L_5	0	θ_6
$5 \rightarrow 6$	$\frac{\pi}{2}$	0	0	θ_7

Fig. 2: Tabla parámetros D-H usando 6 DOF.

III. Cinemática inversa

Para el análisis de la cinemática inversa de Baxter, se tubo en cuenta una consideración adicional a la de fijar el tercer grado de libertad del robot en cero, además de esto, para efectos prácticos en la resolución analítica del problema, se considero $l5 = 0$, gracias a estas hipótesis es posible encontrar analíticamente una solución a la cinemática inversa de Baxter. En la figura 4, se muestra la solución analítica de la cinemática inversa de Baxter.

IV. Programación e implementación en Matlab

A. *Estructura del código*

En el proceso de realización de la simulación de Baxter, se crearon muchos subprogramas, los cuales son de gran importancia y desempeñan tareas especificas en el código general de la simulación.

En primera instancia se crearon los programas asociados a resolver tanto la cinemática directa como la inversa de Baxter, ambos programas se realizaron considerando como nulos los valores del tercer grado de libertad y la longitud $l5$, en la figura 3, se muestra una estructura general de las entradas y las salidas de cada uno de estos programas

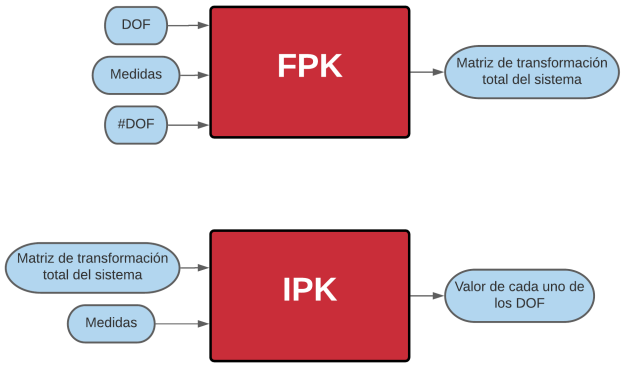

Fig. 3: Estructura general de los códigos FPK.m y IPK.m.

El código encargado de resolver la cinemática directa del robot, tiene como parámetros los valores de los grados de libertad y las longitudes asociadas a la estructura física del robot, esta serie de parámetros se ingresan al programa por medio de una matriz que asemeja a la respectiva tabla de parámetros de Denavit-Hartenverg, de igual manera, el programa tiene otro parámetro de entrada, el cual permite al usuario elegir si quiere analizar la cinemática directa considerando los 7 grados de libertad o solo 6.

El programa calcula la multiplicación de todas las transformaciones que están consignadas en la tabla de parámetros que se ingrese, devolviendo una matriz de transformación total.

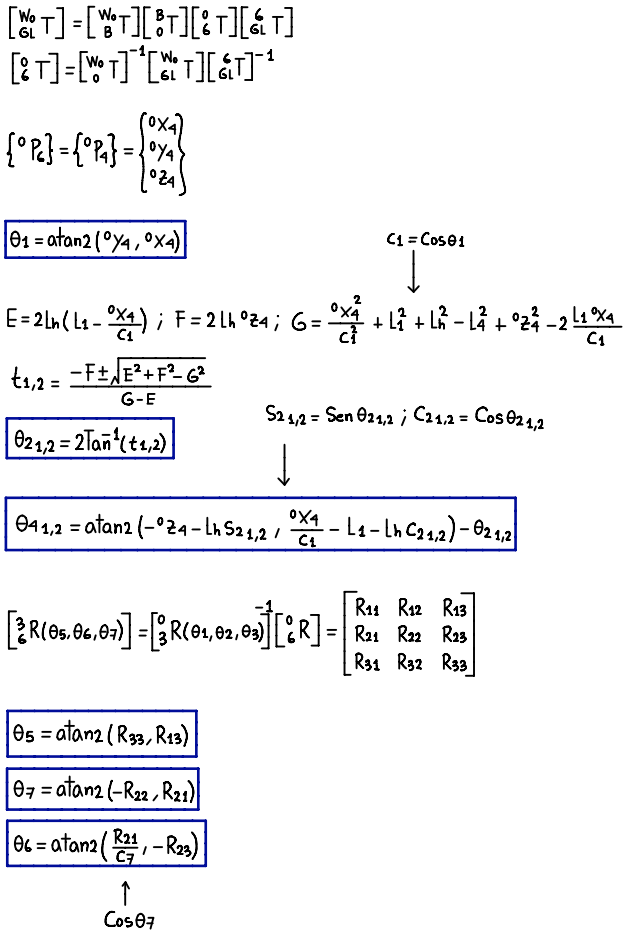

Fig. 4: Solución cinemática inversa Baxter[2]

El programa desarrollado para encontrar la cinemática inversa del robot, se desarrolló en base al análisis y deducción realizados y mostrados anteriormente en la sección donde se hablo de este procedimiento, esta función toma por parámetros una matriz de transformación, la cual contiene la posición y la orientación de la herramienta final del brazo de Baxter, de igual manera también recibe todas las medidas del robot, con estos insumos iniciales y utilizando todas las ecuaciones matemáticas anteriormente deducidas, el programa es capaz de calcular los valores numéricos de cada uno de los seis grados de libertad tenidos en cuenta en el análisis.

Para una correcta organización de todos los códigos real-

izados, se decidió crear un código general para agruparlos de una manera mucho mas compacta y ordenada, debido a esto se creo el archivo *BaxterClass.m*, el cual es una clase, la cual tiene como atributos todas las medidas asociadas a la estructura física de Baxter, y a la vez posee tres métodos diferentes, dos asociados a los códigos explicados anteriormente, donde se resuelven la cinemáticas directas e inversa y otro mas encargado de encontrar los puntos necesarios para graficar las extremidades del robot. En la figura 5, se puede apreciar claramente la estructura general de la clase creada.

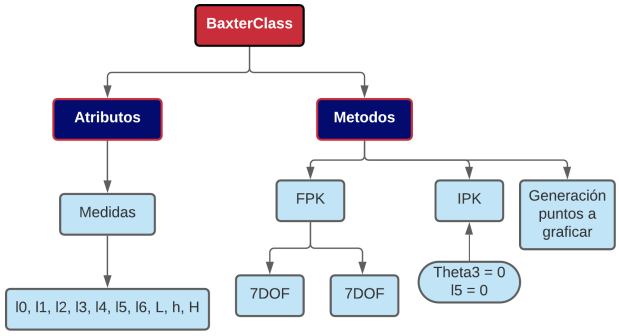

Fig. 5: Estructura general código BaxterClass.m.

El método encargado de encontrar los puntos, es básicamente una función que utiliza el código de la cinemática directa para lograr su objetivo, la cuestión principal y la importancia de este método radica en que, debido a que las tablas de parámetros de Denavit-Hartenverg deducidas al inicio del texto, no tienen en cuenta todos los puntos de la estructura de Baxter, como lo son el punto de la herramienta final y los puntos tanto del comienzo del hombre como la base del robot, es debido a esto que para poder considerar estos puntos y de esa manera poder en un futuro realizar una correcta y adecuada visualización y animación del robot, es necesario de este método.
Todos los códigos y estructuras de programación anteriormente mencionadas y explicadas, se utilizan de manera conjunta en el programa principal encargado de la simulación del robot, el archivo *Baxter_Simulation.m*, posee una estructura un poco mas elaborada en comparación a la de los códigos anteriormente mencionados y explicados, en la figura 6, se puede apreciar la composición general de este programa.
El programa encargado de la simulación de Baxter tiene como parámetros iniciales las condiciones iniciales y finales de las cantidades asociadas a la traslación y a la orientación de ambos brazos del robot, esto debido a que el programa le da la posibilidad al usuario de escoger y programar una trayectoria suave diferente para cada uno de los brazos del robot, con estos valores iniciales el algoritmo calcula los valores asociados a la trayectoria seleccionada por el usuario utilizando un código que se encarga de calcular la trayectoria de quinto orden.

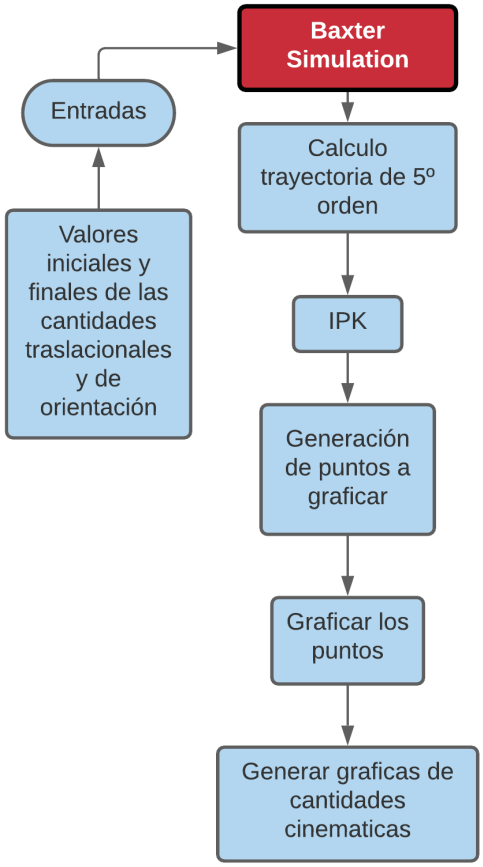

Fig. 6: Estructura general código Baxter_Simulation.m.

Cabe recalcar que el código que se venia mencionando, se realizo con anterioridad, no se explico en detalle debido a que no posee mucha complejidad como los anteriormente mencionados en este escrito, una vez calculados los valores de la trayectoria, tanto los asociados a la traslación como los referentes a la orientación, se pasa a a construir una nueva matriz de transformación, la cual se utilizara como entrada para la funcion encargada del calculo de la cinemática inversa del robot.
Una vez calculados los valores de los grados de libertad de cada uno de los brazos del robot, se utilizan estos valores como entradas para calcular los puntos de cada brazo que permitirán realizar el dibujo y animación del mismo. Finalmente después de haber pasado por toda esa serie de procesos, se pasa a graficar los brazos del robot utilizando los puntos inmediatamente encontrados, una vez se grafican se vuelven a borrar para poder graficar los siguientes puntos en la siguiente iteración y de esa manera poder crear la sensación y el efecto de movimiento. Inmediatamente después de que los brazos llegan a su destino y se termina la simulación, se pasa a graficar las cantidades cinemáticas asociadas a cada brazo con respecto al tiempo, al igual que la variación de los grados de libertad en toda la trayectoria.

B. *Implementación en Matlab*

Para la verificación y validación del código se programo una trayectoria diferente para cada uno de los brazos, se estableció que cada una de estas trayectorias debía durar 6 segundos cada una, para efectos prácticos el código implementado hace que sin importar cual sea la trayectoria que tengan los brazos esta se debe cumplir en el mismo periodo de tiempo, esto con el fin de que los brazos dejen de moverse al mismo tiempo.

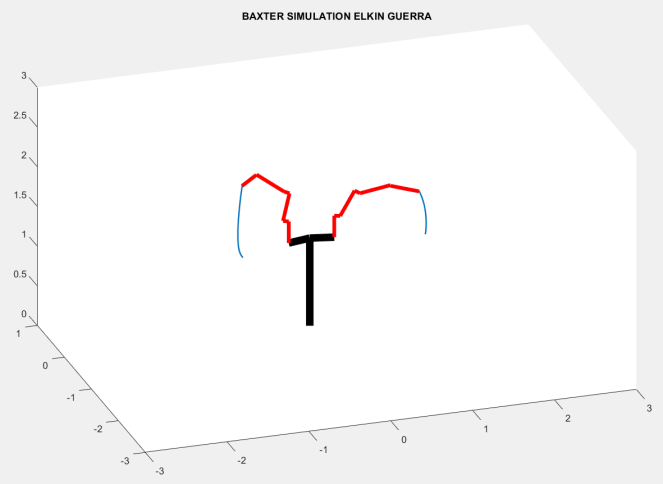

Fig. 7: Simulación Baxter trayectoria suave de frente[1]

Los resultados de la simulación se pueden apreciar en las figuras 7, 8 y 9, en las cuales se puede apreciar el aspecto de Baxter en la simulación, al igual que en las figuras 12 y 13, se pueden apreciar las curvas de las cantidades cinemáticas de cada uno de los brazos de Baxter.

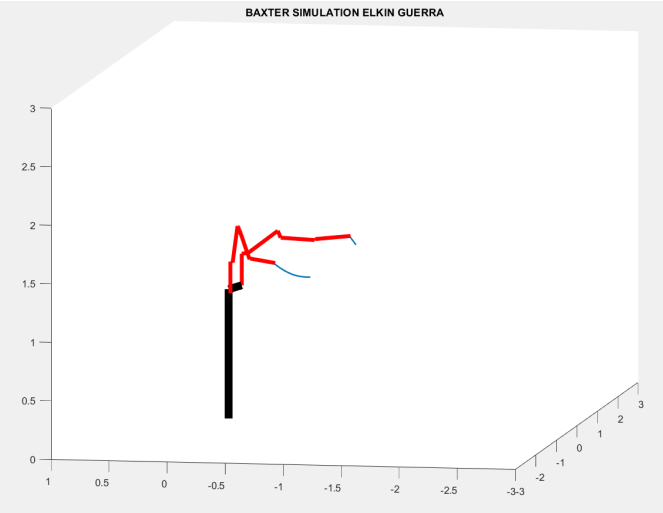

Fig. 8: Simulación Baxter trayectoria suave de perfil[1].

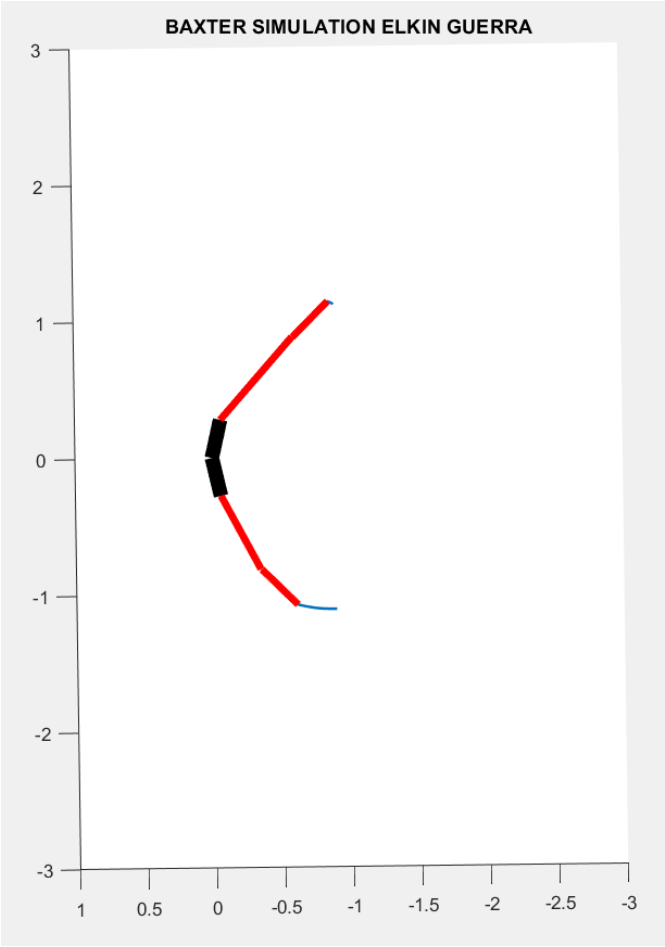

Fig. 9: Simulación Baxter trayectoria suave desde arriba[1].

REFERENCES

[1] MathWorks. *MATLAB and Simulink*. URL: https://www.mathworks.com/?s_tid=gn_logo.

[2] Ph.D Robert L. Williams II. "Baxter Humanoid Robot Kinematics". In: (2017).

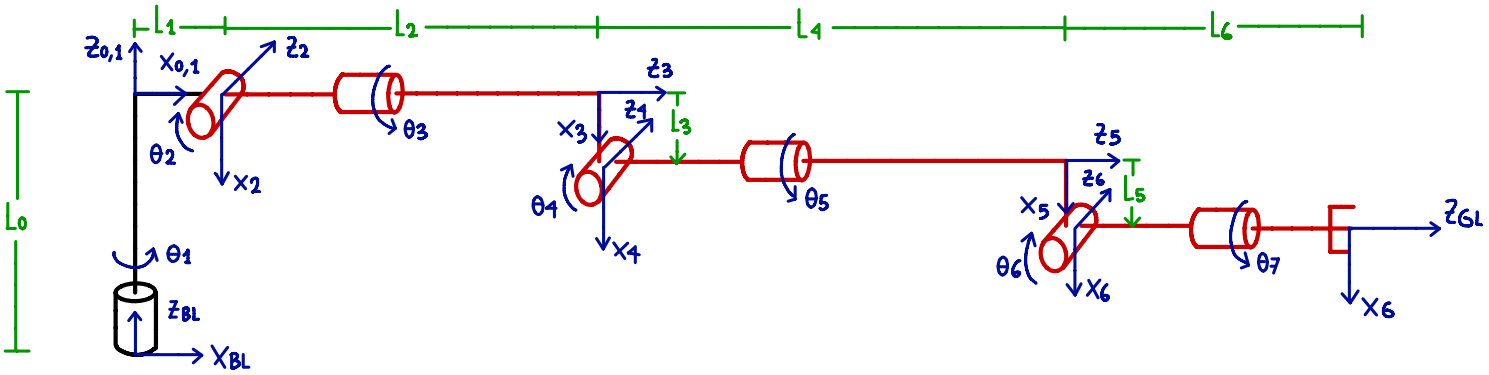

Fig. 10: Diagrama simplificado del brazo de Baxter con 7 DOF.

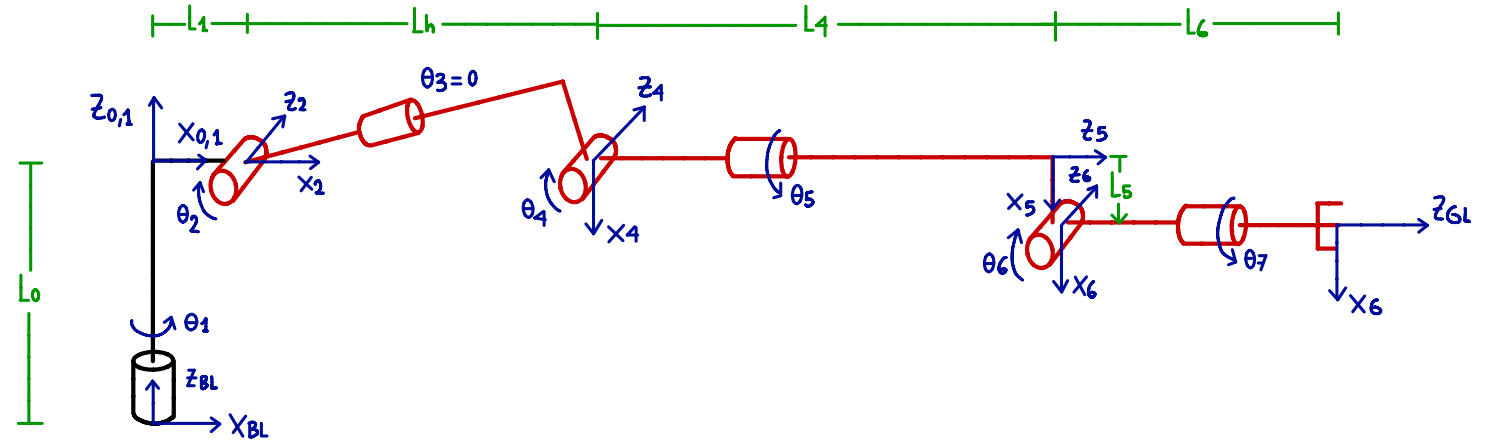

Fig. 11: Diagrama simplificado del brazo de Baxter con 6 DOF.

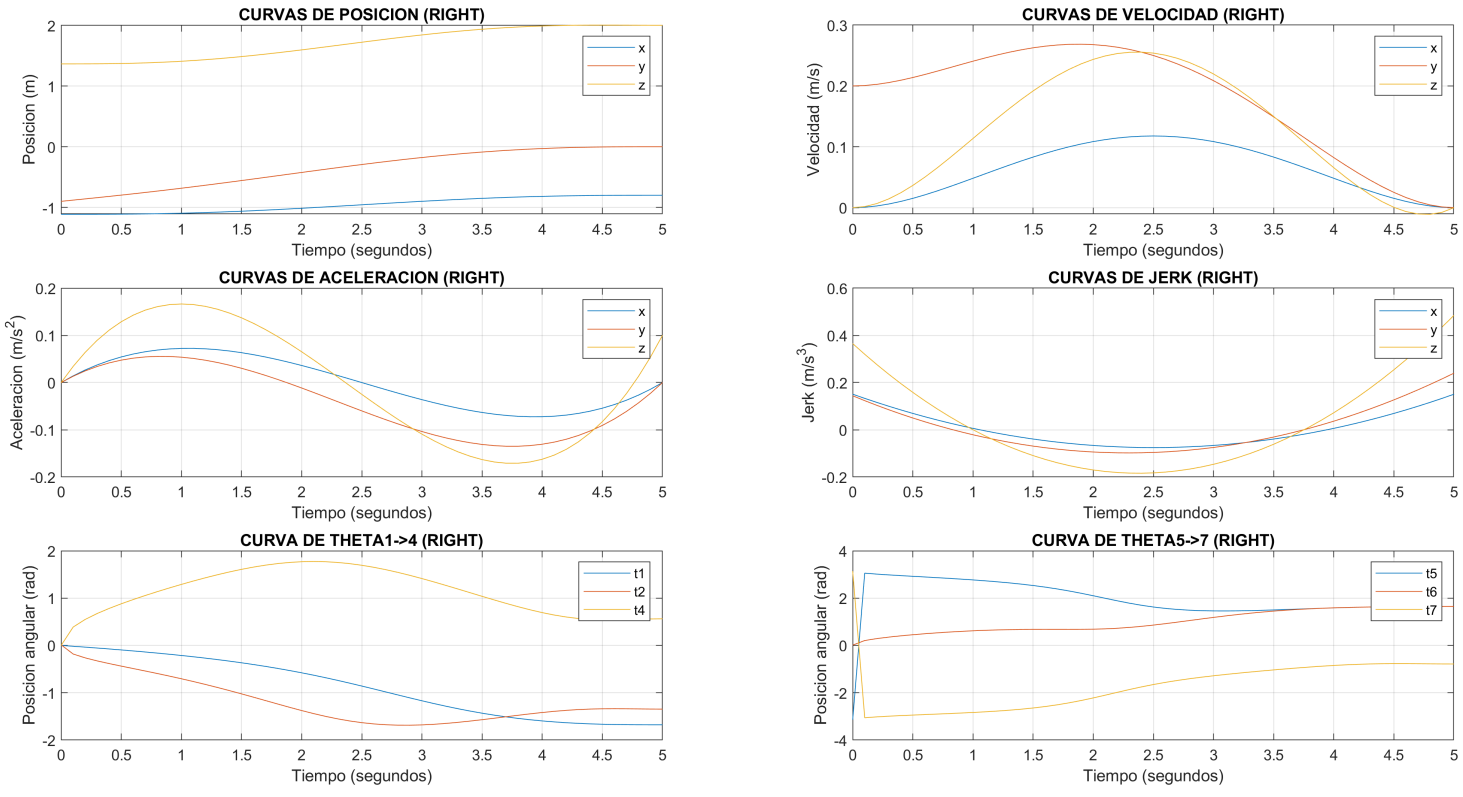

Fig. 12: Gráficas cantidades cinemáticas brazo derecho de Baxter.

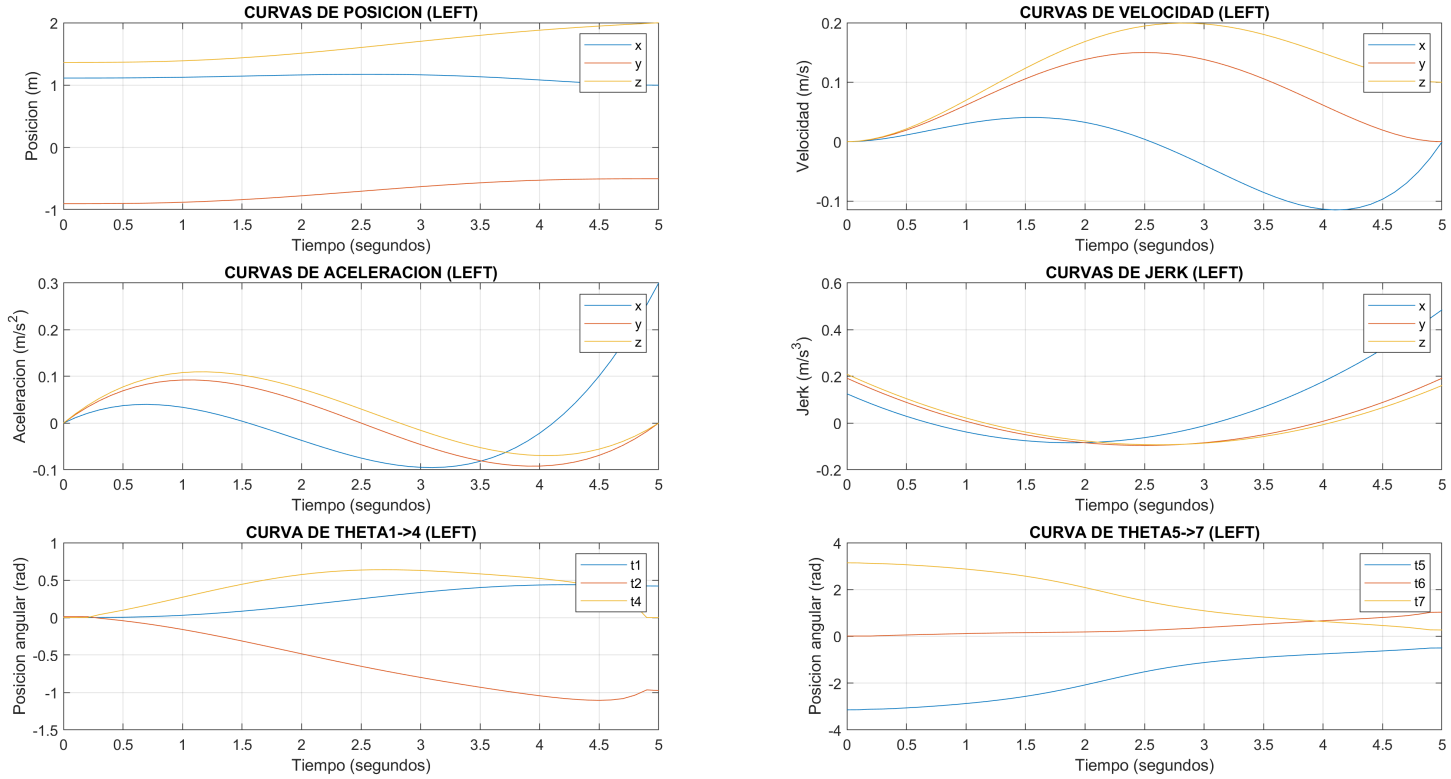

Fig. 13: Gráficas cantidades cinemáticas brazo izquierdo de Baxter.