TRABAJO CON MATLAB

DESCRIPCIÓN BREVE

Simulación del robot scorbot con simscape

Laura Ferrer

ÍNDICE

OBJETIVOS

El objetivo principal de este trabajo es realizar una simulación por **Matlab**, en concreto con la herramienta de **Simscape**, del funcionamiento y comportamiento del robot **scorbot.**

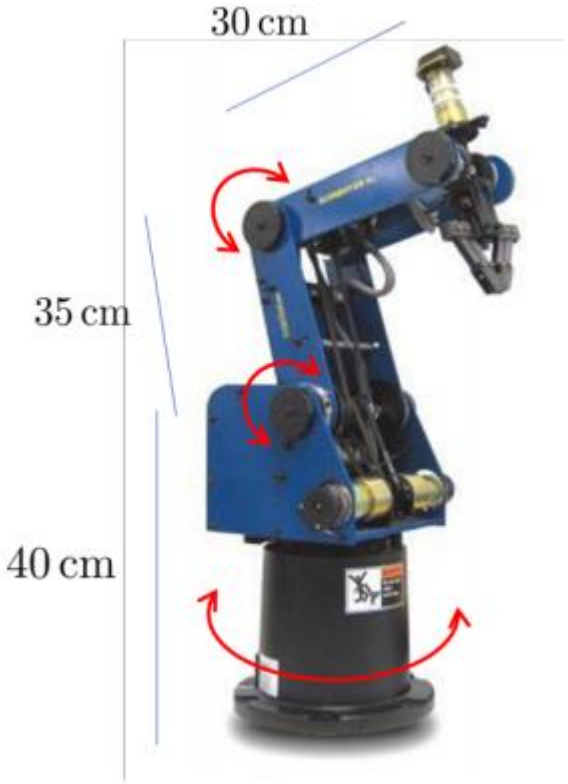


Figura 1: Scorbot

DESCRIPCIÓN

El **Scorbot** es un robot manipulador articulado con cinco articulaciones, en mí caso solo implementare tres, haciendo que solo tenga tres grados de libertad, que son los movimientos independientes que puede realizar cada articulación. Quedando como se muestra en la figura.

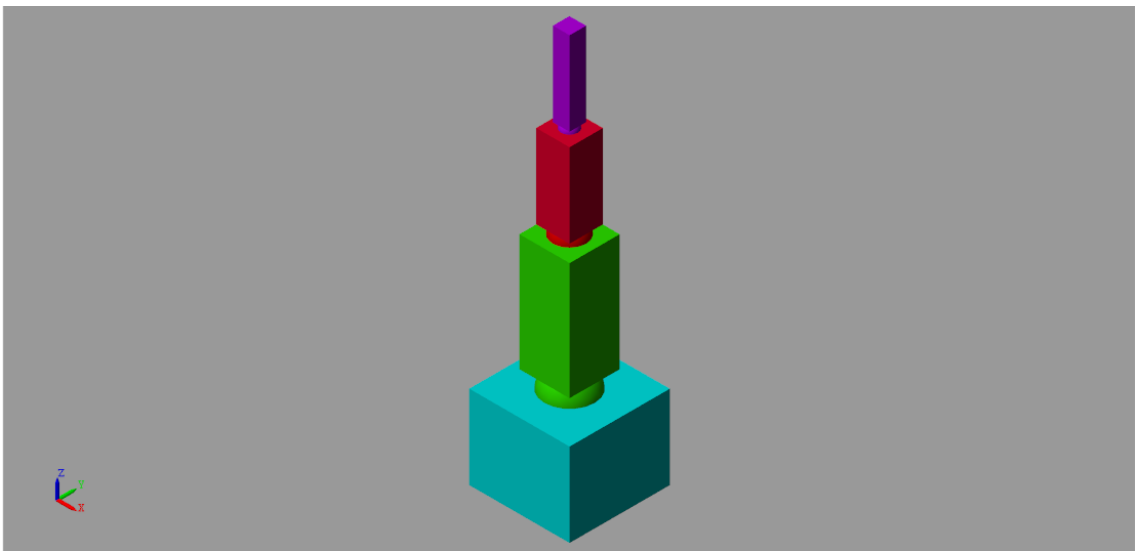


Figura 2: scorbot en Matlab

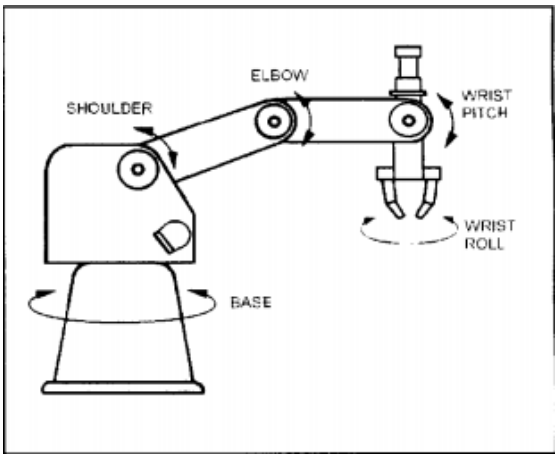


Figura 3: Representación de las articulaciones del scorbot

Número de Articulación	***Nombre***	***Acción***
1	Base	Gira el cuerpo
2	Hombro	Sube o baja el brazo superior
3	Codo	Sube o baja el antebrazo

Tabla 1: Movimiento de las articulaciones del Scorbot

DESARROLLO DE LA SIMULACIÓN

SIMSACPE MULTIBODY

Para realizar la simulación vamos a utilizar un producto de Matlab llamado **Simscape Multibody**. Eta extensión de Matlab permite simular sistemas mecánicos en 3D. Modelando sistemas "multicuerpo" utilizando bloques que sirven de cuerpos, uniones, restricciones, elementos de fuerza y sensores. Ayuda a desarrollar sistemas de control y a probar el rendimiento a nivel de sistema. Puede parametrizar sus modelos utilizando variables y expresiones de MATLAB, así como diseñar sistemas de control para su sistema multicuerpo en **Simulink**. Algunas de sus aplicaciones pueden ser: **Un brazo de robot**, **Excavadora**, **Grúa**, etc.

Para poder implementar cosas en **simscape** hay que tener en cuenta ciertas cosas. La primera de ellas son los tres parámetros que tienen que estar por defecto siempre que queramos realizar un nuevo proyecto. Son siempre el punto de partida. Indican una referencia respecto al entorno en el que vamos a trabajar desde cera y establecen los parámetros mecánicos y de simulación que se aplican a la maquina a la que están conectado los bloques.

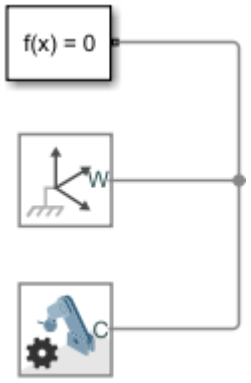


Figura 4: Bloques iniciales

El **solver configuration** especifica los parámetros que el modelo necesita antes de que pueda comenzar la simulación.

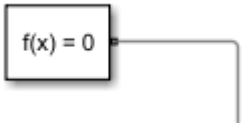


Figura 5: Solver Configuration

El **world frame** representa el marco de referencia global en un modelo. Este marco es inercial y en reposo absoluto. Los ejes del marco son ortogonales y están dispuestos de acuerdo con la regla de la mano derecha.

Figura 6: World Frame

El **mechanism configuration** proporciona parámetros mecánicos y de simulación a un mecanismo, es decir, un grupo autónomo de bloques Simscape Multibody interconectados. Los parámetros incluyen la gravedad. Estos parámetros se aplican solo al mecanismo de destino, es decir, el mecanismo al que se conecta el bloque.

Figura 7: Mechanism Configuration

La orientación de una pieza está definida por lo ejes de coordenadas propios de la pieza. Para conectar piezas entre sí como a nosotros nos interese, podemos modificar estos ejes para poder definir la orientación y la distancia de la siguiente pieza que vayamos a encajar respecto a la pieza anterior. La **rigid transform**:

Figura 8: Rigid Transform

El **revolute joint** representa una articulación con un grado de libertad de rotación. Un primitivo revolucionario proporciona el grado de libertad rotacional. Los orígenes del marco base y seguidor siguen siendo coincidentes durante la simulación.

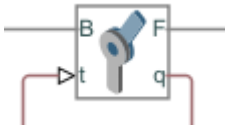


Figura 9: Revolute Joint

	Coeficiente Muelle (N/m)	***Coeficiente de Amortiguamiento (Ns/m)***
Articulación 1	0	146
Articulación 2	16,3	172
Articulación 3	0	130

Tabla 2: Parámetros de las articulaciones

El **brick solid** agrega al marco adjunto un ladrillo sólido con masa y tamaño variables. La masa y las longitudes de los lados (x, y , z) del ladrillo pueden ser constantes o variar con el tiempo. Existen varios tipos:

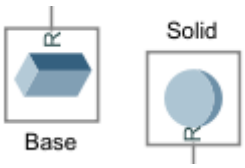


Figura 10: Brick Solid (Cuadrado, Esfera)

Para cada articulación tendremos en cuenta la densidad del aluminio, ya que supondremos que nuestras articulaciones están hechas de este material. Para cambiar la densidad de la articulación nos iremos a uno de los parametros del **brick solid**, y desde allí se le cambiará la densidad de 1000 a 2700.

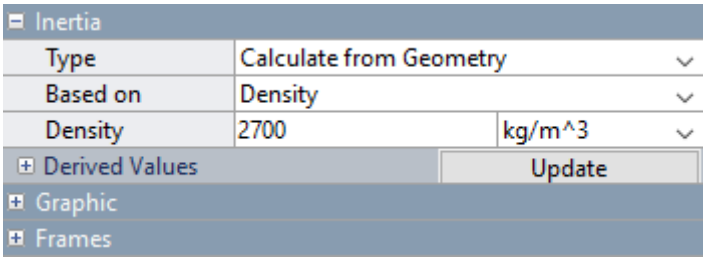


Figura 11: Propiedades del brick solid

SIMULINK

Simulink es una aplicación que permite construir y simular modelos de sistemas físicos y sistemas de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo. Utilizaremos esta herramienta para implementar los motores, sensores y los controladores. Para ello describiremos brevemente que son los motores, sensores y controladores.

Un **motor DC** o de **corriente continua** son convertidores electromecánicas capaces de transformar energía desde un sistema electrónico a un sistema mecánico. Su funcionamiento se basa en la fuerza que se produce sobre un conductor eléctrico recogido por una intensidad de corriente eléctrica en el seno de un campo eléctrico.

MOTOR	
Parámetro	***Valor***
Resistencia del rotor	5,6
Inductancia del rotor	0,048
Inercia	0,028
Coeficiente de fricción	0,0029
Ganancia V s/rad	3,1
	3,9
	6,4
Radio de la rueda	0,071
Voltaje máximo	7,8

Tabla 3: Parámetros del motor

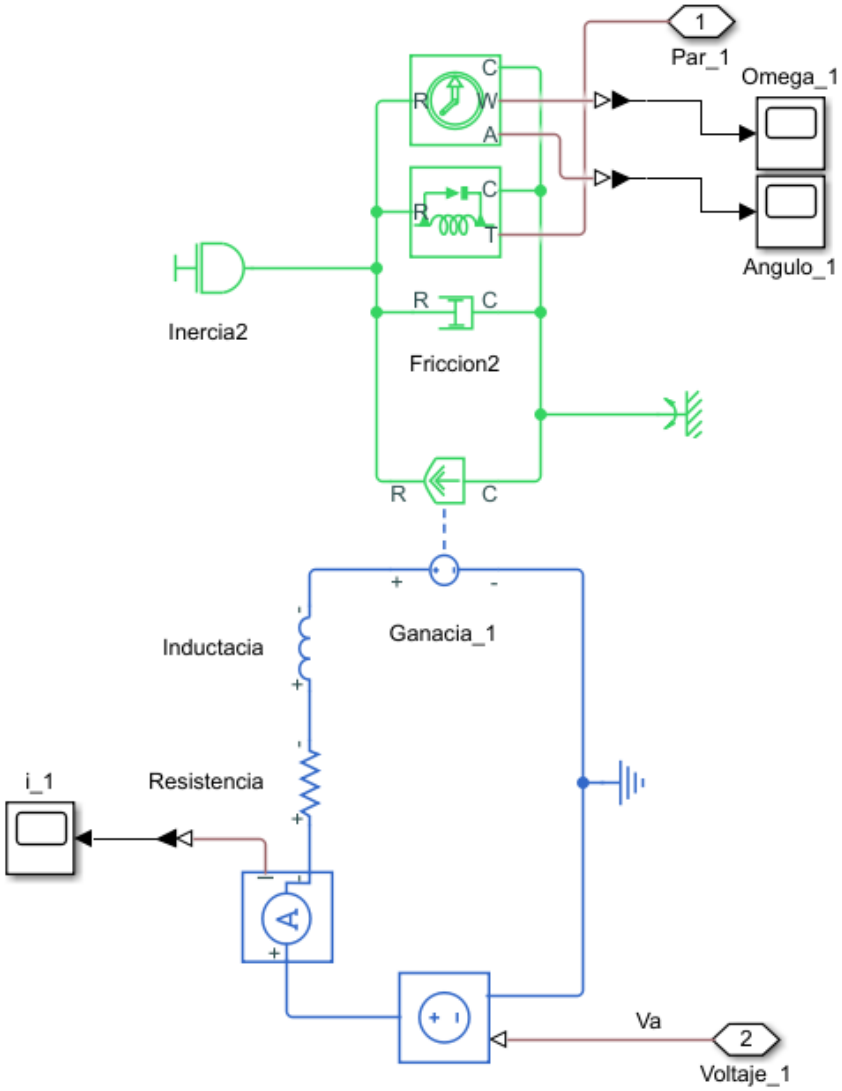


Figura 12: Motor DC en Simulink

El **sensor** es un dispositivo que mide una señal física o estímulo y genera algún tipo de repuesta. Los controladores pueden comportarse de forma ideal, pero en nuestro caso implementaremos varios tipos de errores. Tienen características **estáticas** y **dinámicas.**

- **Características estáticas**: describen la respuesta del sensor una vez todos los efectos transitorios han terminado y la mediad se ha establecido en su valor final. Algunas de estas características son: *Rango de medida*, *Sensibilidad*, *Exactitud*, *Resolución*, *Precisión*, etc. Existen varios errores en los sensores: **errores sistemáticos**, **errores aleatorios**.
- **Características dinámicas**: describen la respuesta del sensor durante el transitorio, antes de que la medida se establezca en el valor final. Se pueden distinguir tres tipos de comportamientos dinámicos: *Orden 0*, *Orden 2*, etc.

Articulación	***Error Sistemático***	***Error Aleatorio***	***Ganancia Estática***	***Constante de Tiempo***
1	0	0,01	1	0
2	0,0045	0,0085	1	0,28
3	0	0,013	1	0,26

Tabla 4: Parámetros de los sensores

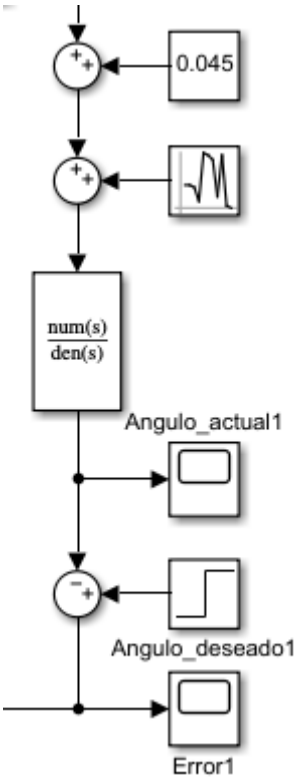


Figura 13: Sensor en Simulink

El **controlador** decide las entradas al sistema en función de la referencia. En nuestro caso usaremos el controlador **PID** que es un mecanismo de control que a través de una lazo de retroalimentación permite regular la velocidad, temperatura, precisión, etc. El **controlador PID** calcula la diferencia entre nuestra variable real contra la variable deseada.

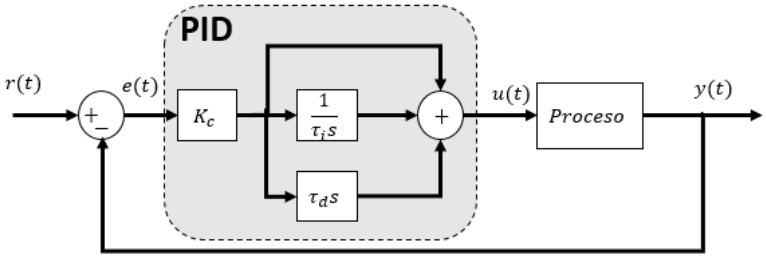


Figura 14: controlador PID

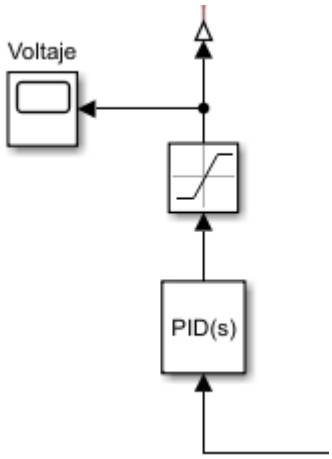


Figura 15: controlador PID en Matlab

CINEMATICA DIRECTA E INVERSA

La **cinemática directa** sustituye en la matriz cinemática o matriz homogénea los valores de los ángulos y longitudes de cada eslabón con variables simbólicas, se multiplican las matrices para obtener la posición del efector final, y se reduce el resultado a su mínima expresión quedando en términos de las longitudes y los ángulos.

La **cinemática inversa** (IK) se utiliza para determinar las configuraciones de juntas de un modelo de robot para lograr una posición de efecto final deseada. Las restricciones cinemáticas de robots se especifican en el modelo de robot en función de la transformación entre juntas.

La convención **Denavit-Hartenberg** permite representar la relación geométrica desde la base de un sistema hasta su último eslabón.

```
clear

L(1) = Link([0, 7, 0, pi/2]);
L(2) = Link([0, 0, 2, 0]);
L(3) = Link([0, 0, 1, 0]);

matriz_DH = SerialLink(L, 'name', 'three link');

qz = [pi/6 pi/6 pi/6];
qn = [2.5 2.5 8];

figure;
subplot 211;
matriz_DH.plot(qz);
subplot 212;
matriz_DH.plot(qn);

disp("+------------------------------------------------------------+")
disp("+-------------------------DIRECTA----------------------------+")
disp("+------------------------------------------------------------+")

disp("qz")
disp(matriz_DH.fkine(qz));

disp("qn")
disp(matriz_DH.fkine(qn));

disp("+------------------------------------------------------------+")
disp("+-------------------------INVERSA----------------------------+")
disp("+------------------------------------------------------------+")

disp("qzi")
T = matriz_DH.fkine(qz);
disp(matriz_DH.ikunc(T));

disp("qni")
T1 = matriz_DH.fkine(qn);
disp(matriz_DH.ikunc(T1));
```

Figura 16: Cinemáticas.

COMPLICACIONES

Al poner las articulaciones demasiado grandes (es decir, en metros) el rozamiento era tal que la articulación no era capaz de sobre pasarlo para moverlo en direccion contraria. Para solucionarlo lo que he hecho es dividir todas las variables, tanto de las articulaciones como de los eslabones y la base entre 10.

Otro problema que me ha surgido ha sido que en un principio el ángulo actual no llegaba al deseado, para solucionar esto he tenido que modificar los parámetros del PID, para que llegara con más potencia y que no sobre oscilará.

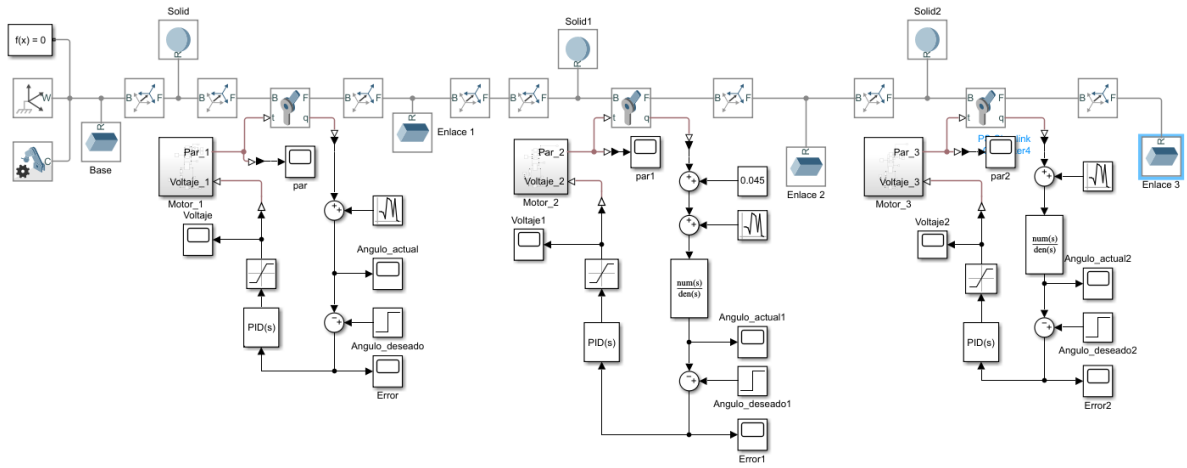
Otro problema que me ha surgido fue al principio al colocar los eslabones (enlaces) puesto que la **rigid transform** la había puesto para que girará en el *eje X 90º* quedando hacia arriba el *eje Y*. Una vez me di cuenta de esto cambie el eje al Y para que el *eje Z* fuera el que indicara el movimiento de la articulación que en este caso es el que consideramos como *eje x*.

Otro problema surgido fue que el PID de las articulaciones dos y tres daban valores por debajo del ángulo deseado, esto se debía a los limites que internamente tenía la articulación. Una vez puesto los límites adecuado, las articulaciones llegan perfectamente al ángulo deseado.

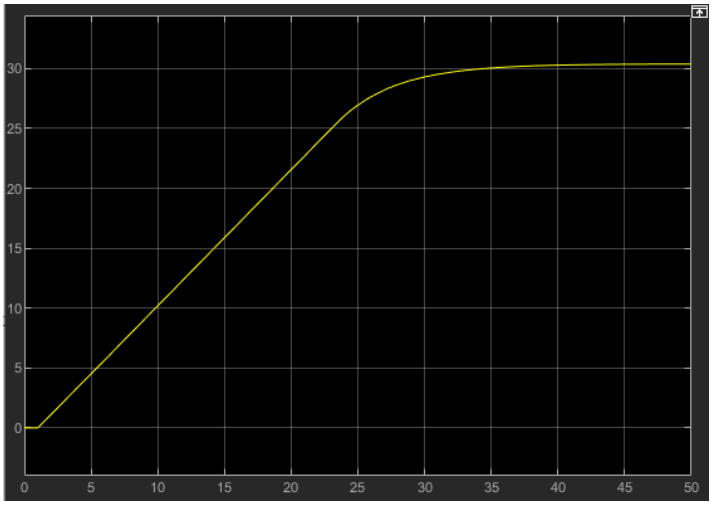
Otro problema que surgió fue que el PID no estaba saturado, es decir, daba valores muy altos porque no había nada que le dijera hasta que valor tenía que moverse provocando que las articulaciones alcanzarán valores gigantes. Para solucionar esto se utilizó la saturación, que consiste en decirle al PID hasta que valor le tiene que dar al motor.

RESULTADO

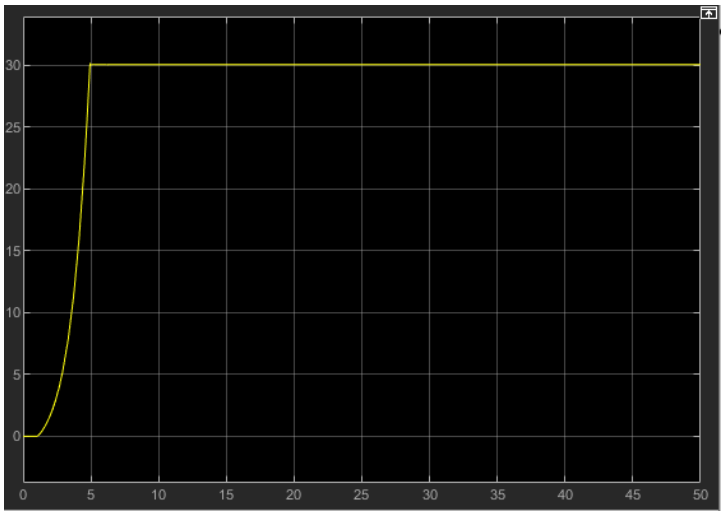
Con sensor



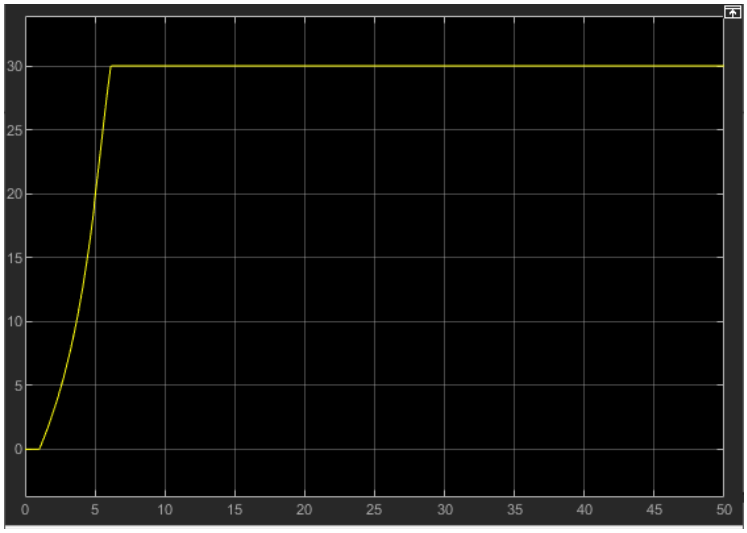
Articulación 1



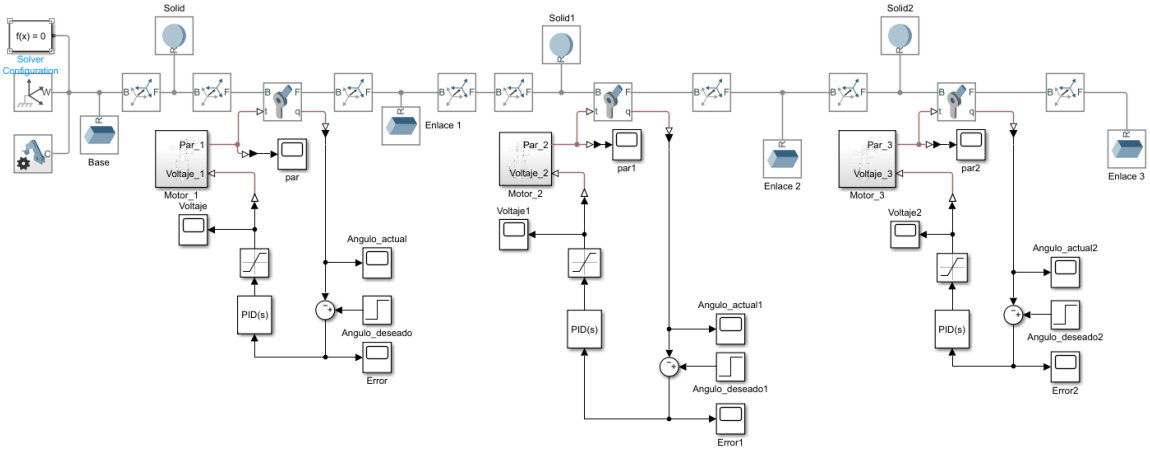
Articulación 2



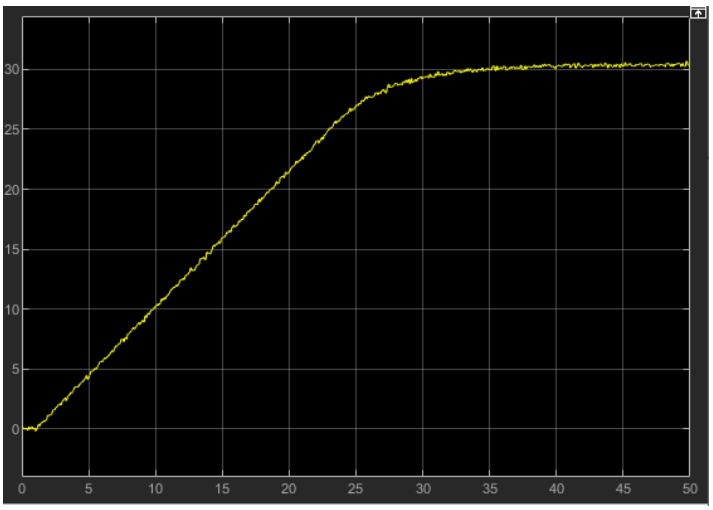
Articulación 3



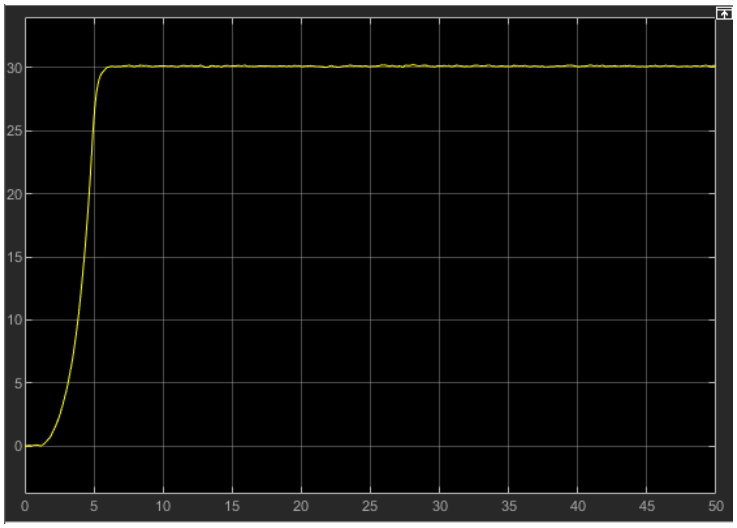
Sin Sensor



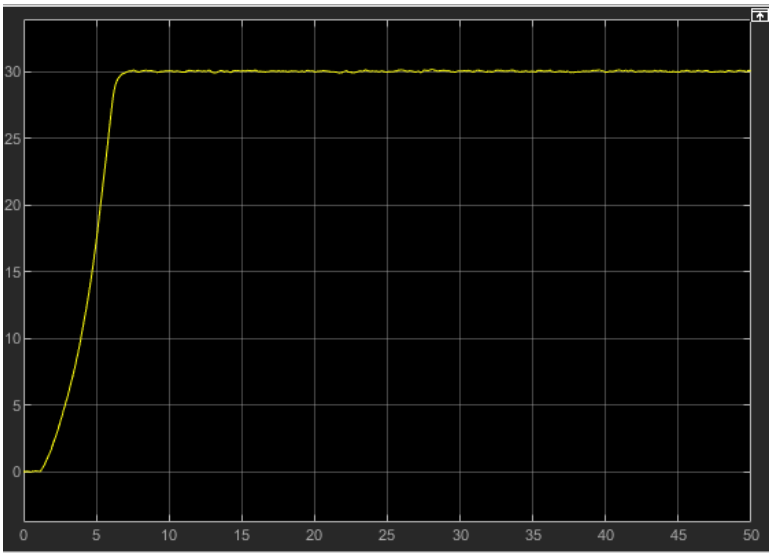
Articulación 1



Articulación 2



Articulación 3

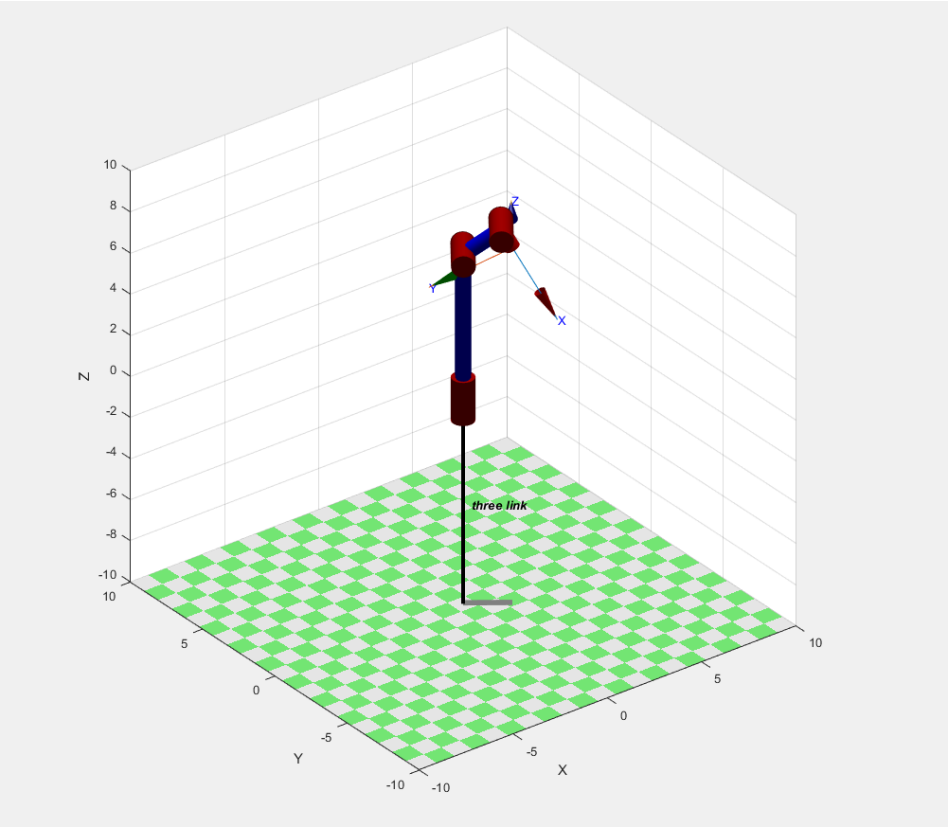


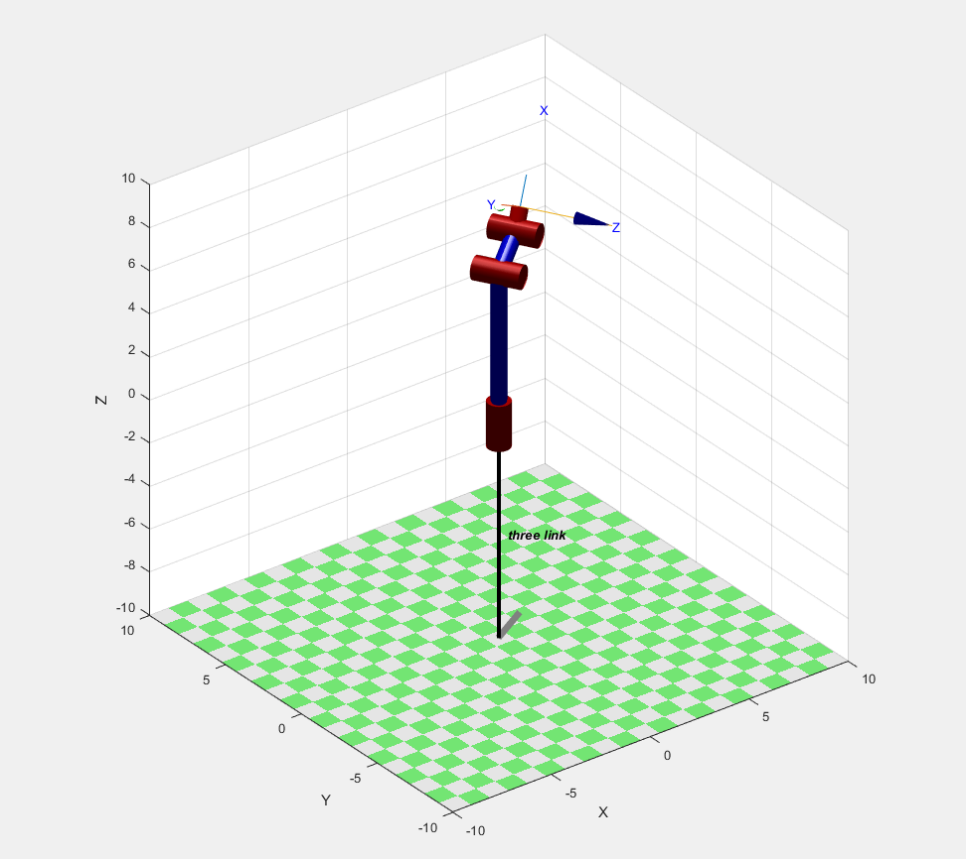
Como podemos comprobar al añadir el sensor la articulación deja de ser ideal.

Resultado Matriz DH

```
>> Matriz_DH
+-------------------------------------------------------------+
+--------------------------DIRECTA----------------------------+
+-------------------------------------------------------------+
qz
    0.433     -0.75       0.5     1.933
     0.25    -0.433    -0.866     1.116
    0.866       0.5 6.123e-17     8.866
        0         0         0         1
qn
      0.381   -0.7048    0.5985     1.665
    -0.2846    0.5265    0.8011    -1.244
    -0.8797   -0.4755 6.123e-17     7.317
          0         0         0         1
+-------------------------------------------------------------+
+--------------------------INVERSA----------------------------+
+-------------------------------------------------------------+
qzi
    0.5236    0.5236    0.5236

qni
   27.6327    8.7832   -4.5664
```


three link
Z
Y
X


three link
X
Y
Z