Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Simulación Dinámica de Robots usando Simscape (Simulink). Aplicación en el Movimiento de Robots Humanoides y Cuadrúpedos

Autor:
Sánchez-Girón Coca, Celia

Tutor:
Herreros López, Alberto

(Departamento de Ingeniería de Sistemas y Automática)

Valladolid, julio 2022

AGRADECIMIENTOS

En primer lugar, quiero dar las gracias a mi tutor de Trabajo de Fin de Grado Alberto Herreros por dirigir mi proyecto y ayudarme siempre que lo necesitaba.

A mis padres por la paciencia que han tenido durante los cuatro años de carrera y por el apoyo que me han dado siempre para continuar con mis metas.

A mis amigos y compañeros de curso por motivarme y animarme para superar las dificultades.

Por último, gracias Miguel por ser mi pilar en la carrera y por ser el mejor compañero de camino desde el primer día.

RESUMEN

En el presente proyecto se realiza el estudio dinámico de diferentes modelos robóticos, entre los que destacamos varios modelos humanoides y un robot cuadrúpedo.

Para conseguirlo se utilizará el programa Matlab, así como otras herramientas como Simscape para conseguir la simulación de los modelos o appDesigner para que el usuario pueda jugar con el dispositivo robótico utilizando la interfaz. Además, se incorporarán elementos de control para estabilizar el sistema y mejorar el movimiento del robot.

El proyecto se plantea como una ampliación de una librería de control robótico, que se centraba en el estudio cinemático de robots diferentes. En el desarrollo del TFG, se expondrán varios ejemplos para comprender el funcionamiento de los modelos robóticos paso a paso.

Palabras clave: Simscape, dinámica, control, humanoide, cuadrúpedo, robot

ABSTRACT

In this project, the dynamic study of different robotic models is carried out, among which we highlight several humanoid models and a quadruped robot.

To achieve this, the Matlab program will be used, as well as other tools such as Simscape to simulate the models or appDesigner so that the user can play with the robotic device using the interface. In addition, control elements will be incorporated to stabilize the system and improve the movement of the robot.

The project is conceived as an extension of a robotic control library, which focused on the kinematic study of different robots. In the development of the TFG, several examples will be presented in order to understand the operation of the robotic models step by step.

Keywords: Simscape, dynamics, control, humanoid, quadruped, robot

Contenido

CAPÍTULO I: INTRODUCCIÓN

1.1 INTRODUCCIÓN

La fabricación y utilización de robots es algo muy reciente en la historia de la humanidad. La primera vez que se empleó el término robot fue en 1920 en la obra del escrito Checoslovaco Karel Čapek "Rossum's Universal Robots". En apenas 100 años esta fantasía de ficción se ha vuelto real y no solo eso, sino que los robots se han convertido en unos dispositivos con los que convivimos en nuestra vida diaria.

En el mundo de la movilidad, muchas empresas han apostado por introducir robots en su cadena de producción para mejorar su rendimiento y sus resultados finales. Esto ha supuesto una gran ventaja, ya que se han podido emplear los robots para llevar a cabo tareas repetitivas o tareas que suponían un gran esfuerzo físico para los trabajadores.

Igualmente, en otros muchos sectores como el de la construcción o el sector de la electrónica se utilizan cada vez más robots industriales y colaborativos para lograr unos resultados precisos y exactos con las especificaciones del cliente.

Además, en el campo de la medicina los robots son de gran ayuda para recuperar la funcionalidad neuromuscular o musculoesquelética de un paciente con problemas de movilidad. Actualmente, se están implantando modelos robóticos en varias terapias de recuperación muscular de forma que estos modelos puedan adaptarse a las necesidades de cada usuario en función de su lesión y así ofrecer un entrenamiento personalizado a cada paciente.

En los últimos años ha surgido un gran interés por los robots humanoides. Estos modelos robóticos pueden ser diseñados para tener unas propiedades mecánicas muy superiores a las de los humanos; además de esto, suelen tener las extremidades, articulaciones y dimensiones similares a las de una persona de carne y hueso. Gracias a estas características, se ha podido someter a este tipo de robots a situaciones que son peligrosas para los seres humanos para evaluar las posibles consecuencias. Por ejemplo, estos modelos son útiles a la hora de evaluar la eficacia de material de protección como cinturones, airbags… o pueden ser utilizados también para la inspección y mantenimiento de las centrales de energía para liberar a los trabajadores humanos de tareas laboriosas e inseguras.

Por otra parte, los robots humanoides se han utilizado para simular movimientos que sí que pueden realizar los seres humanos. Por ejemplo, en las actividades de rehabilitación uno de los mayores inconvenientes para los pacientes es el aburrimiento y monotonía de los ejercicios, hasta el punto de

que tras varias repeticiones el paciente se cansa y deja de prestar atención a la correcta ejecución del ejercicio. En esta situación de frustración, los robots humanoides pueden ser de gran ayuda ya que los pacientes pueden imitar los movimientos del robot y tendrán un incentivo para continuar todas las repeticiones hasta el final. Esta interacción robot-usuario supone un gran avance en los procesos de rehabilitación, ya que se transforma una experiencia de recuperación muscular en un juego de mímica.

Para conseguir unos movimientos resultantes que sean lo más parecidos posibles a los movimientos de una persona real, es necesario controlar la estabilidad del robot y las posibles oscilaciones que pueden presentar las articulaciones del robot al moverse. Ante este problema surge la necesidad de plantear nuevos diseños de control para acercar lo máximo posible los movimientos del humanoide a los de un humano real.

1.1 JUSTIFICACIÓN DEL PROYECTO

En los últimos años el control de los robots móviles se ha convertido en un aspecto esencial en el campo de la robótica. Algunas de las empresas líderes en robótica como ABB o Boston Dynamics han conseguido que sus modelos realicen desplazamientos y saltos casi imposibles, pero estos avances los han conseguido hace relativamente poco. El rápido crecimiento de la robótica móvil ha obligado a los ingenieros de hoy en día a desarrollar nuevos sistemas de control para dominar los movimientos de los robots.

Este proyecto nace por la necesidad de tener una guía desarrollada del control dinámico en modelos robóticos y su control. Lo que se pretendía en este trabajo es diseñar una serie de objetos y modelos que puedan universalizar el funcionamiento de cualquier robot humanoide.

A diferencia de otras herramientas de simulación, Matlab y Simulink ofrecen una serie de características que permiten simular un modelo robótico de forma dinámica. Esto significa que se tendrán en cuenta aspectos como la masa de las estructuras que forman el robot o el torque que hay que aplicar a los motores para mover una articulación. Se ha decidido aprovechar esta ventaja y estudiar la dinámica de los robots, de forma que las simulaciones obtenidas serán similares al funcionamiento de un modelo robótico real.

Igualmente, gracias a la herramienta Simulink y otros útiles como appDesigner se podrá realizar una simulación de varios modelos robóticos para que el usuario juegue con ellos mediante una consola fácil de manejar. El proyecto, por lo tanto, tiene un gran componente educativo con el que el usuario puede emplear las funciones ya diseñadas o bien crear nuevos

métodos para controlar su robot. A lo largo del proyecto se presta especial atención a los modelos robóticos antropomórficos y se analizan diferentes aspectos para conseguir un movimiento estable y lo más parecido posible al de un robot real. Por otro lado, también se incorpora un modelo cuadrúpedo mucho más estable que dará más libertad al usuario para probar nuevos movimientos sin que el robot pierda el equilibrio.

En cuanto al campo de la robótica, además de realizarse un estudio dinámico de diferentes robots, se han implementado una serie de lazos de control para mejorar la estabilidad y el movimiento de los modelos. El desarrollo de nuevas estrategias de control será muy útil para lograr el control de futuros modelos robóticos, por lo que este TFG podrá servir de guía para proyectos mucho más complejos.

Es por ello que el proyecto presenta un valor teórico, ya que servirá como referencia para investigaciones futuras de robótica y estructura de control, y un valor práctico, porque las simulaciones obtenidas nos darán información sobre el comportamiento de un robot real bajo unas condiciones determinadas.

1.2 MOTIVACIÓN DEL AUTOR

Durante mi etapa universitaria estudié junto al resto de mis compañeros el funcionamiento de diferentes sistemas robotizados de tipo brazo robótico; gracias a esto descubrí mi interés por la robótica y comprendí los fundamentos básicos de la cinemática en robots.

En varias asignaturas realizamos simulaciones de modelos robóticos empleando RAPID de RobotStudio o la librería Robotic Toolbox de Matlab [1], sin embargo, no llegamos a estudiar con profundidad el comportamiento dinámico de dichos modelos.

Para ampliar mis conocimientos en un área que me interesaba, mi tutor Alberto Herreros me recomendó un proyecto en el que se estudiara no solo la cinemática sino la dinámica de un robot. Las asignaturas de mi carrera se enfocaban sobre todo en robots industriales, por lo que apenas había visto el funcionamiento de un robot móvil, lo que suponía un reto, pero también una motivación para mí.

En la actualidad, estoy cursando el grado de Ingeniería Biomédica donde estudié conceptos relacionados con la anatomía del cuerpo humano y la importancia de los robots humanoides en el ámbito social. La idea de

combinar la parte antropomórfica en el campo de la robótica me pareció muy atractiva, por eso decidí que el trabajo encajaba muy bien con lo que estaba buscando.

1.3 OBJETIVOS

El principal objetivo de este proyecto es el estudio del modelo dinámico en diferentes modelos robóticos humanoides y cuadrúpedos. En el presente proyecto se expondrán una serie de ejemplos para comprender el funcionamiento de diferentes tipos de robots y realizar simulaciones con unos resultados verosímiles.

También se tiene como objetivo comprobar el efecto que tienen diferentes controladores sobre el sistema para mejorar el comportamiento del robot. Se plantearán diferentes lazos de control internos y externos para mejorar la estabilidad del sistema y alcanzar los objetivos predeterminados por el usuario.

Para la realización de este proyecto se seguirán una serie de objetivos intermedios:

- Familiarizarse con las herramientas de Simscape y Robotic Toolbox de Simulink y Matlab entendiendo sus fundamentos y aplicarlas a los sistemas robóticos.
- Comprender los diferentes métodos de resolución matemática de los modelos robóticos (conceptos de Denavit-Hartenberg, URDF…)
- Estudiar el problema cinemático y dinámico de modelos robóticos industriales y no industriales.
- Definir entornos de simulación que permitan el movimiento coordinado de las articulaciones de forma cinemática y dinámica.
- Diseñar una interfaz para que el usuario pueda jugar con los diferentes robots y comprender los fundamentos esenciales de la robótica móvil.
- Crear una librería de robots que puedan controlarse de forma universal mediante métodos y objetos estandarizados.
- Adaptar los distintos modelos robóticos en Simulink para lograr su control mediante funciones genéricas.
- Crear un sistema de lazos de control anidados para aumentar la estabilidad del modelo y manejar al robot mediante movimientos automáticos.
- Desarrollar una serie de ejemplos que muestren el movimiento de las articulaciones del robot, así como la similitud que presenta el modelo humanoide con un humano real.

1.5 ANTECEDENTES (PROYECTOS RELACIONADOS CON ESTE)

La Escuela de Ingenierías Industriales de la Universidad de Valladolid cuenta con varios modelos robóticos reales con los que se han realizado estudios y proyectos de investigación con un fin educativo. Antes de comenzar la investigación, se realizó una búsqueda bibliográfica en la base de datos de la Universidad de Valladolid como paso previo al desarrollo del TFG.

En el año 2014 Miguel Ángel Mato San José enfocó su TFG al control cinemático y dinámico de robots comerciales usando Robotic Toolbox de Matlab. En el proyecto se recoge el modelado numérico del robot industrial de ABB IRB-120 y se define una estación de Simulink para modelar y simular el sistema mecánico [2] .

En 2019 Carlos Jiménez llevo a cabo un trabajo de fin de máster que se basaba en el diseño de un sistema didáctico que permitía al usuario jugar al ajedrez con un brazo robótico industrial (modelo IRB-120). Se crearon una serie de objetos para jugar al ajedrez mediante softwares de diseño 3D y se utilizó Matlab para implementar el software del juego para realizar la gestión del tablero y los jugadores [3] .

En 2020 Sandra Arévalo realizó como proyecto de fin de carrera un análisis cinemático y dinámico de robots industriales, así como un estudio del control y comunicaciones de los robots. En él estableció las bases teóricas de la robótica y desarrolló una serie de estaciones para simular diferentes movimientos tanto en un entorno virtual como en el modelo robótico real. En este trabajo se empleó la librería SimRob que va a ser clave para el desarrollo del proyecto; el TFG a desarrollar se puede ver como una ampliación de la investigación llevada a cabo por Sandra [4] .

CAPÍTULO II: ESTADO DEL ARTE

2.1 REPRESENTACIÓN DE UN MODELO ROBÓTICO

Un robot está constituido por un conjunto de piezas o eslabones que se mantienen unidos mediante una serie de conexiones llamadas articulaciones. Gracias a su configuración articular el robot tendrá un número determinado de grados de libertad en función del número de articulaciones. Esta idea de un dispositivo robótico como una sucesión de eslabones se puede representar con definiciones matemáticas diferentes. Entre ellas destacamos el algoritmo de Denavit-Hartenberg y el modelo robótico en formato URDF.

2.1.1 DENAVIT-HARTENBERG (DH)

El método de Denavit-Hartenberg representa los modelos robóticos con un número i de eslabones o "bodies" que se enumeran desde la base del manipulador (eslabón 0) hasta el efector final o extremo del robot (i). Igualmente, el robot cuenta con articulaciones que se definen desde 1 a i. Cada una de las mismas permite que los eslabones contiguos se muevan de forma relativa. Podemos definir la orientación y posición que ocupa en el espacio cada body respecto al body anterior mediante los parámetros de Denavit Hartenberg. Para diseñar la tabla DH empleamos 4 transformaciones que relación los sistemas de referencia entre eslabones:

- Rotación alrededor del eje zi−1 un ángulo θi.
- Traslación a lo largo de zi−1 una distancia di.
- Traslación a lo largo de xi una distancia ai.
- Rotación alrededor del eje xi un ángulo αi.

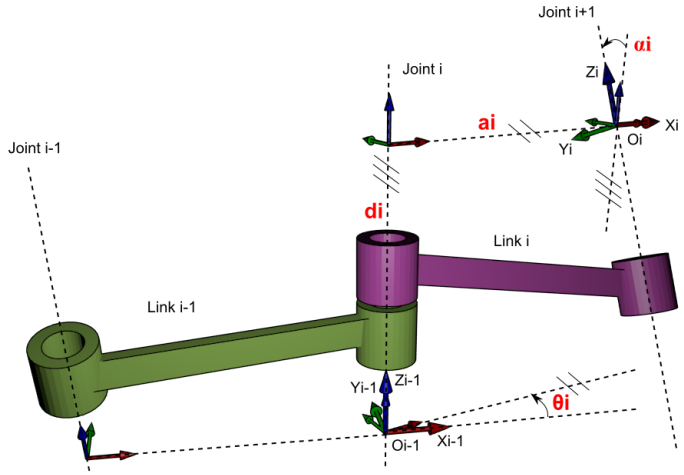


Ilustración 1: Parámetros clásicos DH [5]

Con estos parámetros podremos obtener una matriz para cada eslabón que nos aporta información sobre el estado de un body respecto a su body anterior:

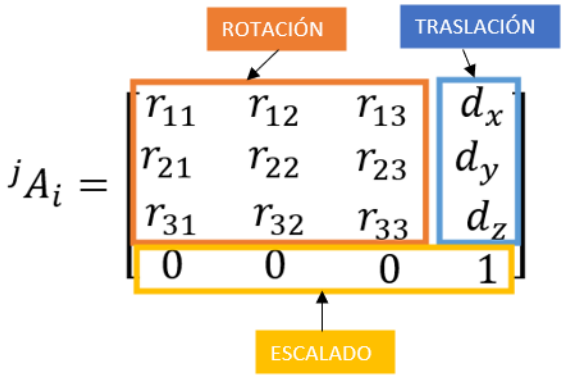
$$^{i-1}A_i = Rotz(\theta_i)T(0,0,d_i)T(a_i,0,0)Rotx(\alpha_i) \qquad (1)$$

Desarrollando la ecuación anterior se consigue la siguiente matriz:

$$^{i-1}A_i = \begin{bmatrix} C\theta i & -C\alpha i S\theta i & S\alpha i S\theta i & aiC\theta i \\ S\theta i & C\alpha i C\theta i & -S\alpha i C\theta i & aiS\theta i \\ 0 & S\alpha i & C\alpha i & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

A partir de estas transformaciones básicas de paso de eslabón podemos obtener la matriz de transformación homogénea (MTH).

Cuando la herramienta de un robot se mueve, esta realiza una serie de traslaciones y rotaciones respecto a un sistema fijo (la base de nuestro robot manipulador). Toda esta información sobre la posición del TCP la podemos recoger en una matriz de 4x4 llamada Matriz de Transformación Homogénea. Según se indica en la Figura, la matriz 3x3 de la izquierda representa la orientación en el espacio del TCP (por la combinación de rotaciones en x, y, z), la última columna (3x1) indica la traslación del TCP en el espacio y la última fila en robótica se define como un vector [0 0 0 1].



$$ {}^{j}A_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

La MTH la obtenemos al multiplicar todas las matrices de transformaciones básicas entre eslabones desde el primer body (0) hasta el sistema del TCP. Igualmente podemos representar de forma parcial la posición y orientación de cualquiera de los bodies del robot con respecto al sistema de coordenadas de la base. Por ejemplo, si queremos representar la posición y orientación del 3º body respecto a la base del robot:

$$ {}^{0}_{3}A = {}^{0}_{1}A * {}^{1}_{2}A * {}^{2}_{3}A \qquad (3) $$

Donde ${}^{1}_{2}A$ representa la posición y orientación del segundo enlace respecto al primero, ${}^{2}_{3}A$ del tercero respecto al segundo…

Además, la MTH permite transformar un vector de posición referido a un punto por otro vector de posición referido a otro punto diferente. Por ejemplo, si tengo el vector A con coordenadas x,y,z y el vector B con otras coordenadas x,y,z diferentes, la MTH nos permite especificar las rotaciones y traslaciones que hay que hacer para pasar del sistema A al sistema B. Esta MTH nos permite hacer un cambio de coordenadas de cualquier punto arbitrario del sistema A al sistema B. Por ejemplo, si conozco la posición de un punto P respecto a un sistema de referencia B, puedo utilizar la MTH para conocer su posición respecto a otro sistema de referencia A.

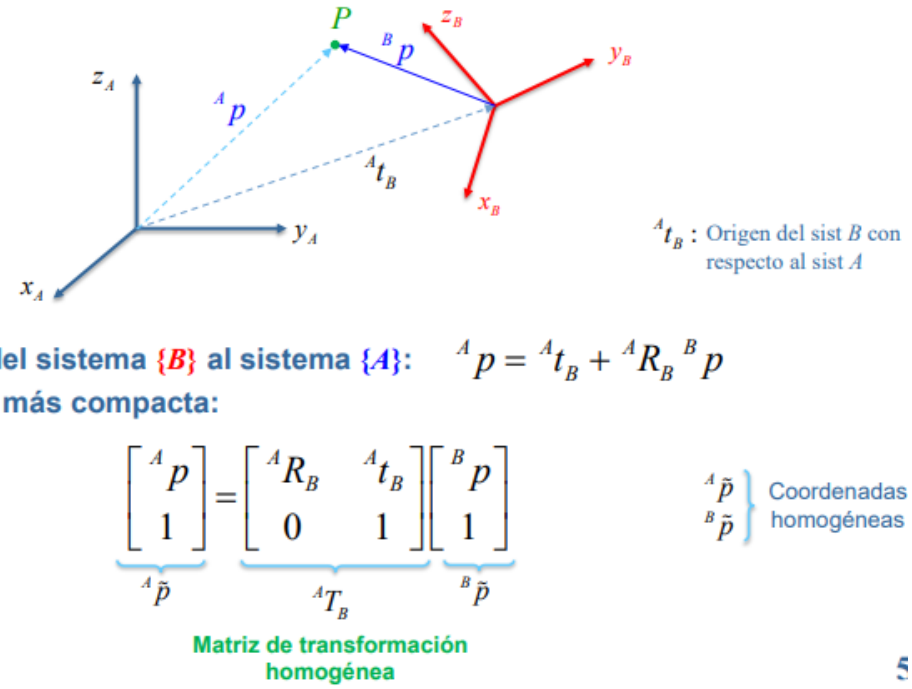


Ilustración 2: Posición punto respecto a otro sistema de referencia [5]

Igualmente podemos calcular la posición y orientación del robot utilizando el método geométrico. Este consiste en despejar los valores que definen el efector final del robot utilizando relaciones trigonométricas entre bodies y ángulos entre eslabones. Este método suele aplicarse a robots sencillos con pocos grados de libertad, por lo que en nuestro caso no se utilizará el método geométrico.

Estos son los métodos habituales que se utilizan para la resolución del problema cinemático directo e inverso.

Los parámetros DH son fáciles de calcular para un robot cualquiera, por lo que se podrá obtener una estructura de un robot simple fácilmente conociendo el ángulo de rotación de las articulaciones y la distancia entre articulaciones. Sin embargo, el algoritmo de Denavit..Hartenberg solo nos proporciona información relacionada con la configuración articular y la posición del TCP del robot. No se tienen en cuenta parámetros como la masa de los eslabones, por ejemplo, por lo que no será posible representarlo de forma dinámica. Además, teniendo en cuenta que cada sistema de referencia se plantea en función del anterior, en un modelo robótico con muchas articulaciones el proceso será tedioso y se deberá realizar el cálculo analítico de los parámetros tantas veces como articulaciones haya.

2.1.2 UNIFIED ROBOTICS DESCRIPTION FORMAT (URDF)

La configuración DH se ha utilizado principalmente para resolver el problema cinemático directo de la mayoría de los manipuladores robóticos hoy en día.

A pesar de esto, se han desarrollado nuevos entornos y herramientas de simulación en los que se imita el comportamiento de un robot de forma mucho más real. Un ejemplo de ello es la herramienta de simulación Gazebo de ROS. ROS es un entorno de programación que incluye muchas librerías para trabajar con modelos robóticos de forma verosímil; para modelar un sistema robótico en dicho entorno se suele utilizar un tipo de fichero llamado URDF.

El formato URDF (Universal Robot Description Formal), se trata de un archivo en formato .xml en el que se definen las especificaciones físicas del modelo robótico (longitud de sus componentes, aspectos visuales de cada body…), por lo que la simulación será más detallada que en el caso de utilizar únicamente los parámetros DH. El formato URDF suele ser utilizado al trabajar con robots desarrollados mediante ROS, como se ha mencionado. Se emplea para diseñar el robot como un conjunto de estructuras rígidas (links), que definen cada eslabón del robot con ciertos parámetros como la inercia, unidas por articulaciones (joints), que establecen propiedades cinemáticas y dinámicas de cada articulación.

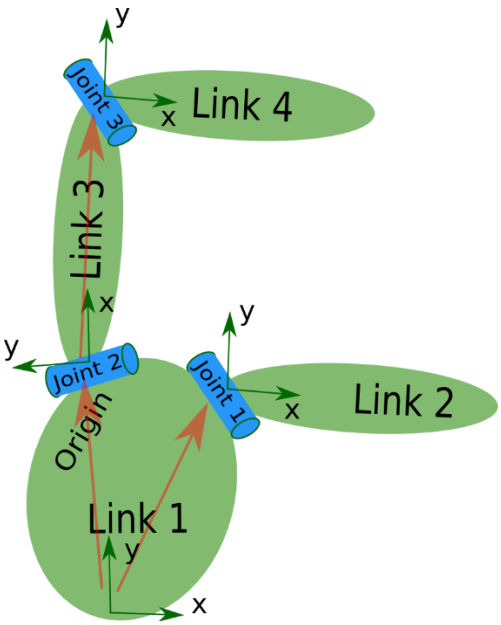


Ilustración 3: Estructura de árbol en URDF [6]

Así mediante las articulaciones (joint) podremos unir varios cuerpos rígidos con diferentes características dinámicas y cinemáticas. El formato .xml que utilizan los ficheros URDF contiene información de los ejes de tal forma que pueda ser interpretado de forma sencilla tanto por un humano como por una máquina:

```
% <robot name>
%       <link name>
%               <inertial>
%                       <origin xyz rpy />
%                       <mass value />
%                               <box size />
%                       <inertia ixx iyy izz ixy ixz iyz />
%               </inertial>
%               <visual name>
%                       <origin xyz rpy />
%                       <geometry>
%                               <cylinder radius length />
%                               <sphere radius />
%                               <mesh filename />
%                       </geometry>
%                       <material>
%                               <color rgba />
%                       </material>
%               </visual>
%       </link>
%       <joint name type>
%               <origin xyz rpy />
%               <parent link />
%               <child link />
%               <axis xyz />
%               <dynamics damping />
%               <limit lower upper  />
%       </joint>
% </robot>
```

En la configuración URDF de un robot podemos encontrar dos elementos bien diferenciados:

<link> o eslabón, que guarda las características visuales u otras propiedades como la inercia o la masa que permitirán simular estados dinámicos, y <joint>

o articulación que une a dos eslabones seguidos. Mediante "joint" se pueden conectar dos "link" donde uno de ellos será la estructura base (parent) y el link unido a este será el eslabón secundario (child).

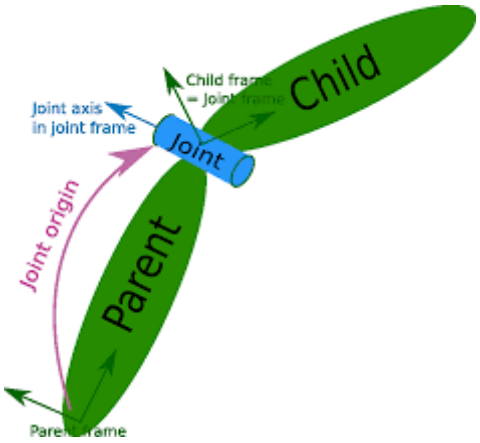


Ilustración 4: Estructura link padre-hijo en URDF [6]

La ventaja de las herramientas Matlab y Simulink es que ambas pueden trabajar con este tipo de archivos, de forma que podemos importar un modelo URDF mediante la librería Simscape MultiBody (Simulink) o con Robotics System Toolbox (Matlab).

- Con la librería Robotics System Toolbox se puede importar la estructura de un robot a partir de un fichero urdf a un objeto RigidBody mediante la función *importrobot().* Además, esta librería contiene varios robots guardados en memoria en formato .mat a los que podemos acceder con el comando *load().* El objeto RigidBody es interesante ya que puede guardar la configuración de un robot mediante los parámetros de la matriz DH o a través de la información contenida en el fichero URDF.
- Con el entorno Simscape MultiBody podemos importar un archivo urdf mediante el comando *smimport()* y obtener un fichero Simulink que define la posición de cada eje del robot y los componentes físicos entre ejes. Estos componentes físicos están definidos por su dinámica y sus ficheros de visualización.

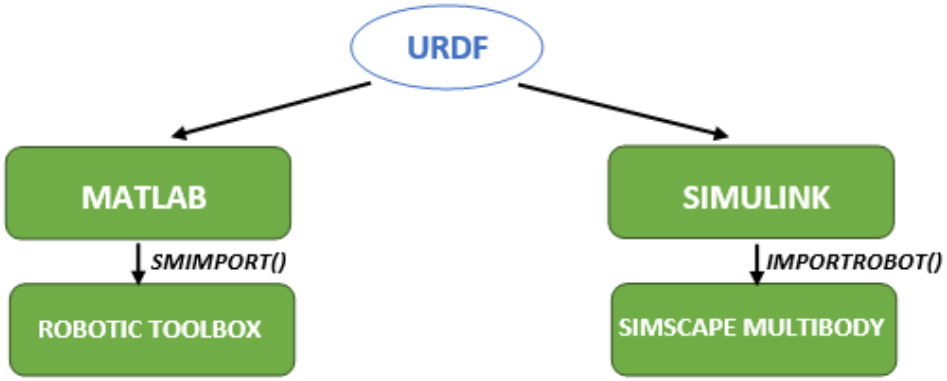


Ilustración 5: Relación URDF, Matlab y Simulink

Si el modelo robótico que se quiere plantear tiene un gran número de grados de libertad, se suele utilizar un formato diferente denominada XACRO, que se trata de un lenguaje XML que consigue diseñar ficheros URDF más reducidos y de fácil lectura en comparación con los archivos URDF.

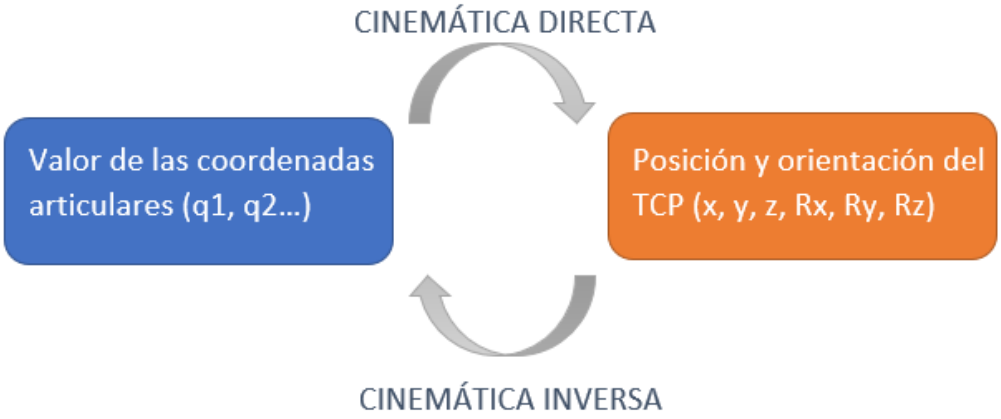
2.2 MODELADO MOVIMIENTO DE UN ROBOT

2.2.1 MODELO CINEMÁTICO

La cinemática de modelos robóticos analiza los movimientos que realiza el robot en el espacio teniendo como referencia un sistema de coordenadas fijo. Aquí se estudian y se relacionan los valores relacionados con las coordenadas articulares del robot y la posición y orientación del TCP (Tool Central Point) que es el punto que se utiliza para conocer el estado del efector final. A la hora de estudiar la cinemática podemos encontrar dos métodos diferentes:

*Cinemática directa: el objetivo de la cinemática directa es encontrar la posición y orientación de la herramienta a partir de los ángulos de las articulaciones, es decir, vamos desde la base del robot a través de ángulos hasta el TCP del robot. Para la cinemática directa normalmente tiene una solución única; dado un robot del que conocemos los ángulos que forman sus articulaciones entonces la posición y orientación del TCP será única.

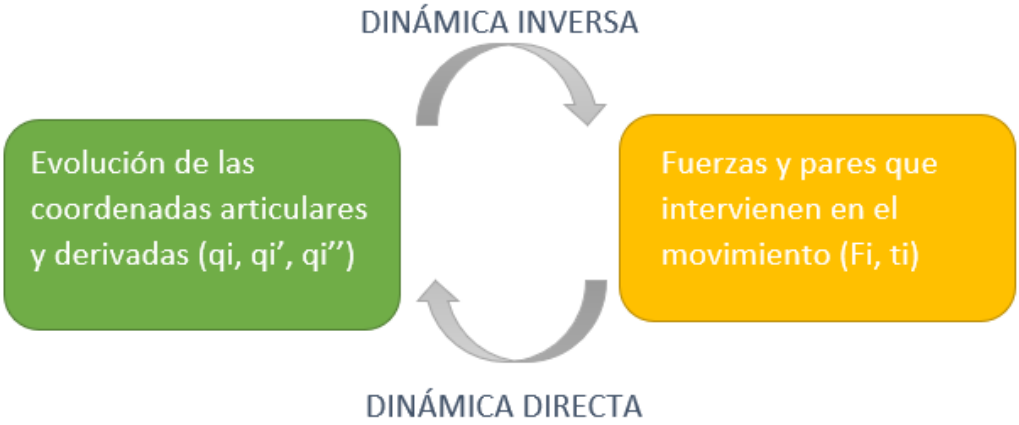
*Cinemática inversa: trata de encontrar los ángulos de las articulaciones del robot dada una orientación y posición de la herramienta determinada. Esto quiere decir que a partir de la situación de la herramienta queremos encontrar los ángulos de las articulaciones. La diferencia con la cinemática directa es que podemos encontrar varias soluciones para la cinemática directa.



2.2.2 MODELO DINÁMICO

En robótica la dinámica estudia la relación existente entre las fuerzas generalizadas que actúan en el mecanismo robótico y el movimiento del robot. Al hablar de fuerzas se incluyen conceptos como las fuerzas y pares de las articulaciones que en intervienen en el movimiento. Por otro lado, el movimiento del robot se define mediante la trayectoria articular y la trayectoria cartesiana del efector final; este movimiento del robot se basa en las variables de posición, velocidad y aceleración de las coordenadas articulares.

Se consideran dos problemas dinámicos que relacionan la evolución de las coordenadas articulares y sus derivadas con las fuerzas y pares del sistema:



El modelo dinámico directo obtiene la posición y sus derivadas de las coordenadas articulares en función de las fuerzas que intervienen en el movimiento del robot. Por otro lado, el modelo dinámico inverso calcula las fuerzas y pares que intervienen en el movimiento del modelo robótico en función de la evolución de q y sus derivadas.

El modelo dinámico es un modelo no lineal que tiene una complejidad significativa a la hora de su resolución; por esta razón con frecuencia se utilizan métodos iterativos para hallar la solución. Entre los principales métodos para estudiar la dinámica de un robot destacan el método de newton-Euler, que es un método iterativo, y el método del Lagrangeano, que es un método cerrado.

En el caso del método de Lagrange o método de Euler-Lagrange, se trata de un método basado en la energía del sistema donde el Lagrangiano se calcula como la diferencia entre la energía cinética (T(q,q)) y la energía potencial del sistema (U(q)):

$$L(q,\dot{q}) = T(q,\dot{q}), -U(q) \qquad (4)$$

Las ecuaciones de movimiento del método Lagrange Euler dan como resultado las fuerzas generalizadas que se aplican sobre las articulaciones del robot (τ_i):

$$\frac{d}{dt}\left(\frac{dL}{d\dot{q}_i}\right) - \frac{dL}{dqi} = \tau_i \qquad (5)$$

A partir de las ecuaciones anteriores se obtiene el modelo dinámico generalizado para cualquier sistema mecánico. En él se contemplan una serie de fuerzas que se pueden resumir en la siguiente ecuación:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \qquad (6)$$

- $M(q)$: matriz de masa o inercia, donde se tiene en cuenta el peso de las articulaciones.
- $C(q,\dot{q})$: matriz que contabiliza las fuerzas centrífugas y de Coriolis.
- $g(q)$: fuerza de la gravedad.

De forma teórica, al modelar dinámicamente un robot se busca encontrar M, C, g que definen el modelo dinámico del robot.

En los ejemplos de este proyecto, la entrada al modelo robótico será el par; para controlar la evolución de las coordenadas articulares se realimentará la posición real de cada articulación y hará la diferencia con la posición deseada. Para regular esto será necesario un sistema de lazos de control que se explicará más adelante.

2.2.3 CONTROLADORES BÁSICOS Y ARQUITECTURAS DE CONTROL

El objetivo de un controlador es encontrar el método para generar la señal de entrada apropiada para que el sistema produzca la salida (referencia) deseada. La acción de control es la responsable de pasar a los motores las instrucciones precisas para conseguir el movimiento deseado de la articulación.

Para controlar el modelo robótico total lo que se hace es emplear sistemas de control en cada una de las articulaciones, es decir, en cada motor que compone el robot.

Un motor en un body del robot no existe de forma aislada; está conectado a un body que presenta dos efectos que afectan al comportamiento del motor: añade una inercia extra y añade un torque debido al peso del brazo. Estos parámetros variarán en función de la configuración articular. Podemos

explicar la influencia de la masa de los bodies en los motores, articulaciones, con este ejemplo.

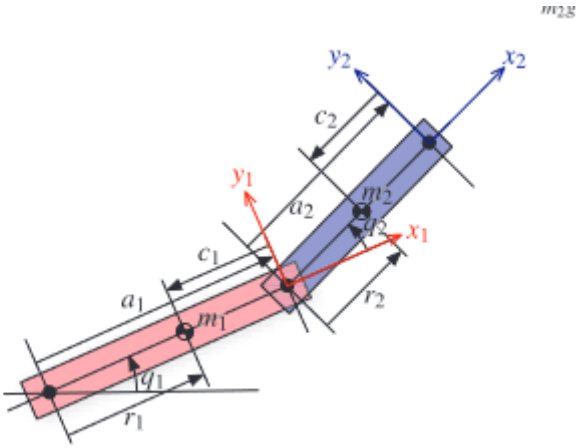


Ilustración 6: Relación propiedades dinámicas entre dos eslabones contiguos [7]

En este ejemplo de una articulación con dos brazos, podemos considerar que la primera articulación está directamente relacionada con el primer body de color rojo. Si asumimos que la masa de este body está concentrada en su centro de masa, entonces el body presentará una inercia adicional de $m_1 * r_1^2$. El motor de la articulación 1 también experimentará la inercia del brazo azul y esto dependerá del valor del ángulo q2, donde la inercia del brazo cuando está recto será mayor que la inercia del mismo cuando está doblado.

Además, la gravedad también actuará sobre el centro de masa del brazo rojo y creará un torque en la articulación 1 que será proporcional a cosq1. La gravedad actúa igualmente en el centro de masa del brazo azul y crea otro torque en la articulación 1 que será más pronunciado cuanto mayor sea la distancia desde el punto de actuación al motor.

En resumen, el efecto del movimiento de las articulaciones en una serie de enlaces mecánicos no es trivial. El movimiento de cualquier articulación se ve afectado por el movimiento de las demás articulaciones del robot, de forma que al trabajar con muchas articulaciones esto se convierte en un problema complejo que es necesario gestionar.

Cuando estamos trabajando con el modelo cinemático, no tiene sentido introducir un controlador porque le estamos introduciendo la posición, velocidad y aceleración directamente y el sistema empleará estos parámetros para calcular el par según una ecuación matemática. El sistema despejará el torque y el robot efectuará un movimiento de rotación perfecto sin necesidad de controlar la acción del motor. La estrategia de control es lo que

diferenciará en gran medida un sistema cinemático de uno dinámico, por ello es necesario explicar algunos conceptos clave que se emplearán a lo largo del trabajo.

Para lograr el control del modelo robótico, debemos tener en cuenta dos ideas fundamentales: el tipo de controlador (P,I,D,PD…) y la estructura de control, es decir, cómo se organizan los controladores en el sistema.

Las acciones básicas que puede tomar un controlador en cualquier proceso son las siguientes:

- **Control proporcional (P):** se trata de un amplificador con una varianza ajustable, donde la relación entre la salida del controlador u(t) y el error de la señal e(t), con Kp como la constante de proporcionalidad, es:

$$u(t) = K_p * e(t) \quad (7)$$

- **Control integral (I):** el valor de la salida u(t) es directamente proporcional a la primera integral del error e(t) con constante de integración Ki. Esto es lo mismo que:

$$\frac{du(t)}{dt} = K_i * e(t) \quad (8)$$

La acción de control integral hace que, si el valor del error se duplica, el valor de la salida varía dos veces más rápido. Para un error de cero, el valor de la salida permanece estacionario.

- **Controlador derivativo (D):** la acción derivativa estima cómo de rápido el error está creciendo o disminuyendo. La salida del controlador es proporcional a la derivada del error, donde Kd es la ganancia derivativa:

$$u(t) = \frac{de(t)}{dt} * K_d \quad (9)$$

- **Control proporcional-integral (PI):** hay un ajuste donde Kp es la constante de proporcionalidad, que ajusta la acción proporcional, y Ti el tiempo integral, ajusta la acción integral.

$$u(t) = K_p * e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt \qquad (10)$$

Al utilizar el controlador proporcional, siempre permanece un error que no se puede anular únicamente con la acción proporcional. Una forma de eliminar este error estacionario es utilizar un controlador integral (PI). Al utilizar el elemento integral en el controlador, se está empleando información de momentos anteriores de la simulación. Si se detecta la presencia de un error estacionario no nulo, entonces el elemento integral provocará que la salida continue creciendo hasta que se elimine el error.

- **Controlador proporcional-derivativo (PD):** aquí Kp es la ganancia proporcional y Td es una constante denominada tiempo derivativo, donde ambas variables son ajustables, donde la salida del controlador es proporcional a la tasa de cambio de la señal del error. La acción derivativa se refiere al control de la velocidad; el tiempo derivativo es el intervalo de tiempo durante el cual la acción de la velocidad adelanta el efecto de la acción de control proporcional. El control derivativo determina cómo nos estamos acercando a la referencia y reducir el valor de la salida antes de que se produzca un sobrepaso del valor deseado. La acción D tiene una naturaleza de previsión.

$$u(t) = K_p * e(t) + \frac{de(t)}{dt} * K_p T_d \qquad (11)$$

- **Controlador proporcional-integral-derivativo (PID):** integra las tres acciones de control básicas para utilizar lo mejor de cada uno de los controladores. Si se detecta que el error estacionario no es nulo, predominará la acción integral para que la salida continue creciendo hasta que se elimine el error; si el error disminuye rápidamente predominará la acción derivativa para reducir el exceso de la salida y evitar que se sobrepase el valor deseado. De esta forma utilizamos el error presente, pasado y la estimación del error en el futuro para obtener la salida adecuada. Para decidir en qué proporción participa cada uno de los elementos en el controlador hay que variar el valor de las constantes.

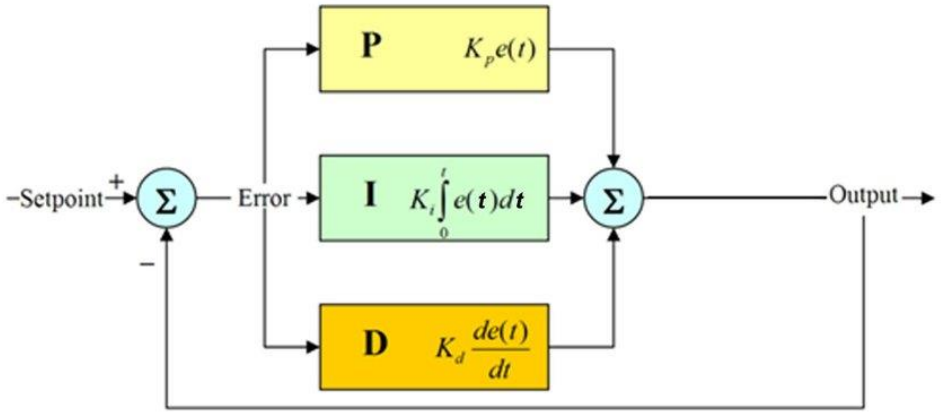


Ilustración 7: Esquema acción de control PID [8]

En cuanto a la arquitectura de control, se suele emplear el siguiente diagrama de bloques para controlar un sistema estándar. El esquema se ha obtenido con la herramienta sisotool de Matlab, con la que se obtienen controladores de forma automática para controlar la planta ante las perturbaciones. El diagrama presenta un prefiltro (F), una planta (G), un filtro adicional (H) y un controlador (C).

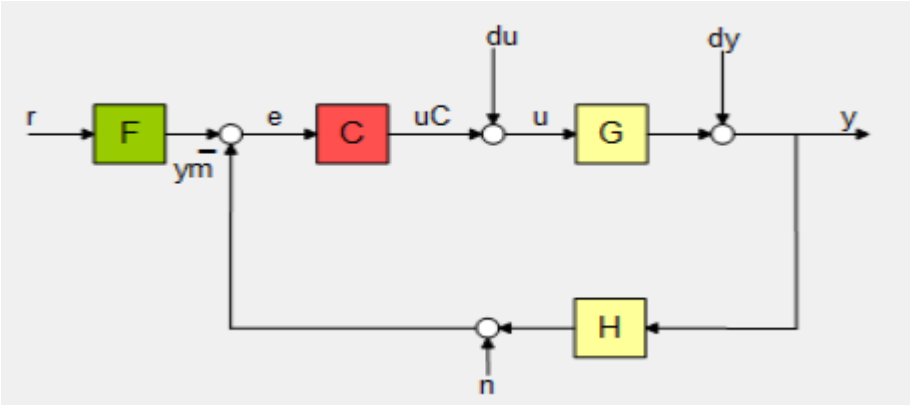


Ilustración 8: Arquitectura de control con un lazo de realimentación [9]

El diagrama más común en robots industriales cuenta con un lazo interno con acción derivativa y uno externo proporcional. El lazo interno se aplica a sistema inestables para estabilizarlos rápidamente durante la simulación; en nuestro caso el controlador derivativo (C2) se encarga de suprimir el ruido. Igualmente cuenta con un lazo de eliminación del error estacionario mediante un control proporcional (C1) que va reduciendo el error poco a poco.

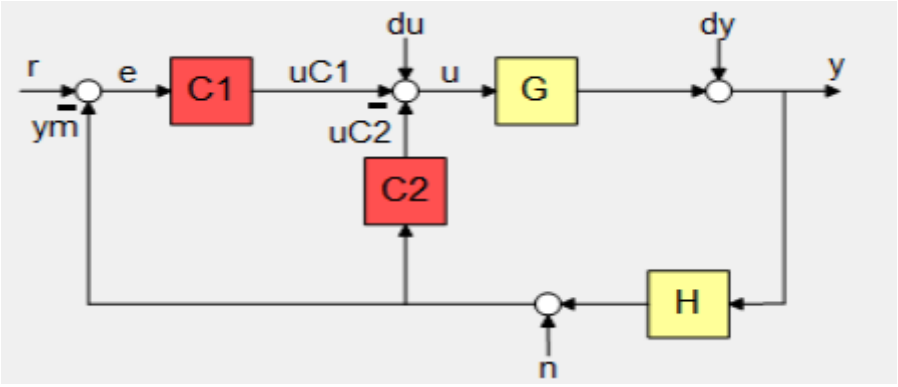


Ilustración 9: Arquitectura de control con múltiples lazos anidados [9]

En los siguientes ejemplos se utilizará un lazo interno para controlar todos los modelos robóticos dinámicos del proyecto. Este lazo interno será el controlador básico para eliminar el ruido de cada articulación y así contribuir a la estabilidad del robot. El lazo interno tendrá una componente proporcional y otra derivativa como se ha explicado en el esquema de "sisotool" anterior.

A diferencia de los diagramas de bloques obtenidos con "sisotool", en los ficheros de Simulink se calculará la velocidad articular directamente con el bloque de la articulación y se empleará un controlador proporcional para conseguir la acción derivativa. El diagrama de bloques que se ha utilizado en el proyecto cuenta con una acción proporcional C1 a partir de la posición y una derivativa C2 a partir de la velocidad.

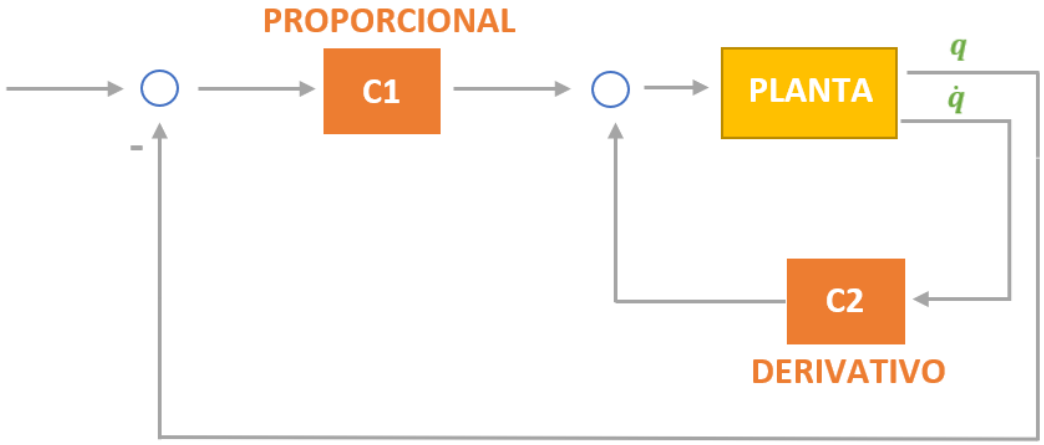


Ilustración 10: Diagrama de bloques "controladores básicos PD" del proyecto

Aprovechamos que el sistema nos proporciona la magnitud de la velocidad para aplicar simplemente un controlador proporcional en vez de uno derivativo en función de la posición; a pesar de utilizar un Kp la acción sigue siendo derivativa. El procedimiento que se está siguiendo para diseñar el controlador es el que se aplica en los robots reales con el elemento encoder.

El encoder es un elemento que transforma movimientos mecánicos en pulsos eléctricos. Para determinar cuántos pulsos se ha movido el encoder encontramos dos pines A y B fijos que envían una señal de 1 bit cuando entran en contacto con la parte conductora del encoder (rectángulos blancos). Cuando el encoder comienza a girar, los pines van recogiendo estos contactos en una señal con forma de onda como la de la imagen. Como resultado final, el encoder nos indica la dirección de giro del motor y cuántos pulsos se han acumulado en el contador.

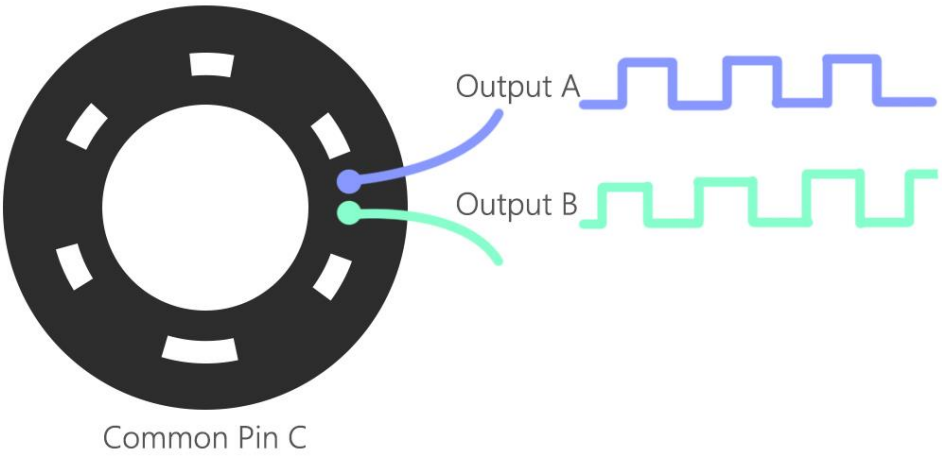


Ilustración 11: Funcionamiento de un encoder incremental [10]

El encoder va sumando incrementos, no aporta la posición actual en valor absoluto, y si se suman estos incrementos se obtendrá la posición del motor. Si dicho sumatorio se divide entre el tiempo de simulación se obtendrá la velocidad. Gracias a esto, el mismo sensor que mide la posición puede medir la velocidad gracias a su carácter incremental.

La acción derivativa analiza cómo de rápido varía la posición articular, de forma que, si la velocidad es baja (la magnitud que se pasa a la entrada del controlador), el tiempo en el que la acción de rapidez se adelanta al efecto de acción P es más grande; como consecuencia la acción derivativa tendrá una precisión baja. En el caso contrario, si la velocidad es muy elevada entonces

el tiempo en el que la acción de rapidez se adelanta al efecto de acción P es reducido; cuando el Td es demasiado pequeño es posible que no haya un efecto significativo de la acción derivativa en el sistema. Encontrar el valor del Td óptimo es importante para eliminar las oscilaciones y conseguir la estabilidad del sistema.

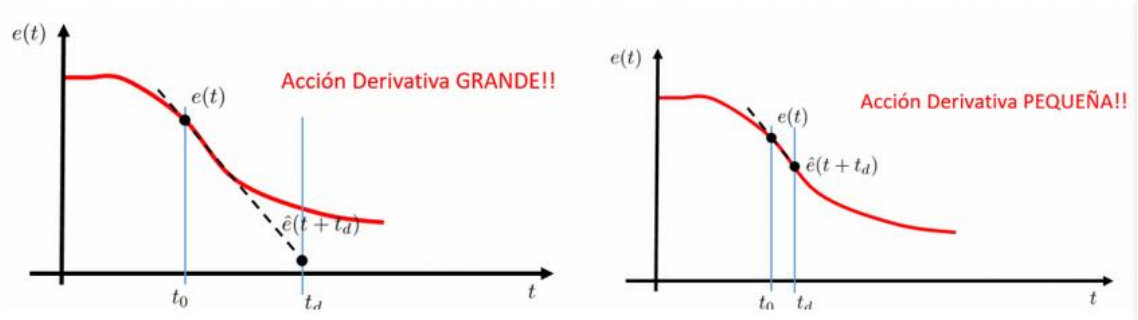


Ilustración 12: Efecto de la Td en la acción derivativa [11]

La arquitectura de control se explicará de forma detallada en cada uno de los modelos robóticos que funcionen de forma dinámica.

2.3 HERRAMIENTAS DE TRABAJO

2.3.1 MATLAB

Matlab [12] es una sofisticada herramienta de ingeniería disponible para resolver problemas matemáticos de forma sencilla. La palabra Matlab viene de las palabras Matrix Laboratory (Laboratorio Matricial), es por ello por lo que la especialidad de este software es resolver vectores y matrices, pero con ella también se pueden resolver cálculos mucho más complejos, simular procesos, crear interfaces gráficas para mostrar al usuario y muchas más aplicaciones.

Matlab utiliza un lenguaje de programación de alto nivel, por lo que es mucho más fácil aplicarlo y usarlo a diferentes proyectos; por esta razón muchos centros de investigación y universidades han elegido esta herramienta. En el campo de la robótica Matlab tiene una gran popularidad, ya que posee un gran número de usos destinados a las aplicaciones relacionadas con la robótica, así como librerías y funciones que facilitan el trabajo del programador.

Para conseguir no solo programar mediante código sino además simular empleando modelos, podemos combinar la herramienta Matlab con el entorno Simulink. Se trata de un entorno gráfico donde el modelo a simular se construye arrastrando los diferentes bloques que lo constituye. Aquí encontramos un gran número de librerías de bloques para aplicaciones de control, de comunicación, de análisis de señales, aeroespaciales y muchas más. Los archivos Simulink se guardan con la extensión.mdl. Debido a sus características, el uso de Matlab y la aplicación de la toolbox Simulink ha sido una herramienta muy útil para modelar y simular sistemas robotizados sin necesidad de construir antes un prototipo real.

2.3.3 SIMULINK

Simulink [13] es un entorno de diagramas de bloque destinado al diseño de sistemas para simular el modelo deseado antes de desarrollar el hardware en la vida real.

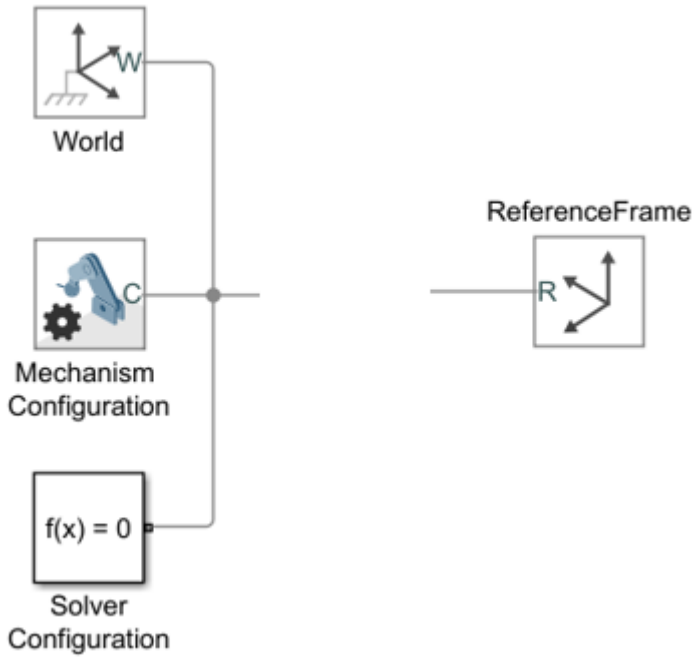
En este proyecto tendrá gran importancia la función smimport () que permitirá convertir un fichero URDF en un modelo Simulink. El fichero URDF inicial contiene información desde los bodies que forman el robot hasta las características articulares del mismo. Gracias a esta transformación, el fichero Simulink contendrá las especificaciones que definen al robot tanto de forma estructural como motriz.

Cuando se importa el fichero a archivo Simulink se generan una serie de bloques de la biblioteca Simscape que se ha explicado anteriormente. Algunos de estos bloques son:

- World: representa el sistema de referencia global del modelo. Este elemento está en reposo absoluto y todos los sistemas de referencia del modelo se definen respecto a este sistema de referencia global.
- Mechanism Configuration: bloque que proporciona parámetros relacionadas con la mecánica y la simulación del sistema robotizado. Aquí se establecen parámetros como la gravedad; si no se incorpora al modelo el vector de aceleración gravitacional se pondrá a cero.
- Solver Configuration: especifica los parámetros del solucionador que necesita nuestro modelo antes de comenzar la simulación.

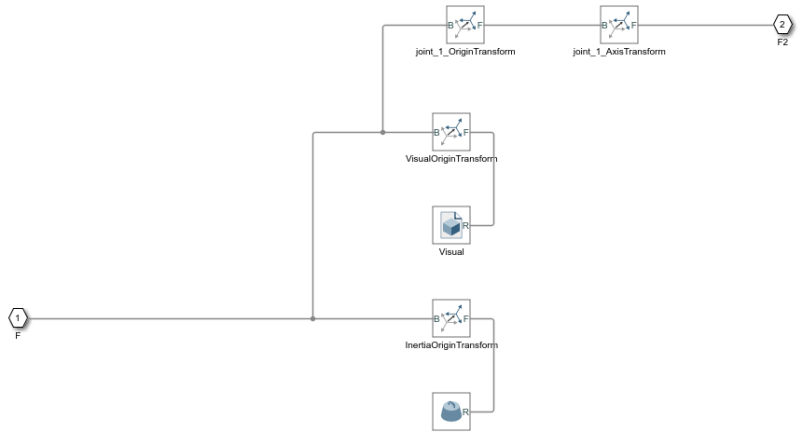
Estos tres bloques constituyen una entrada en el robot del modelo e indicarán la base del mismo. Si el robot está asociado a la base a través de una articulación, entonces esa articulación no podrá moverse.

- Reference Frame: representa un sistema de referencia respecto al que puedo llamar a otros sistemas de referencia.



Bloques Simulink 1: World, Mechanism Configuration, Solver y Reference Frame

Además, cuando se define cada articulación en el bloque del robot se generan varios bloques para especificar los detalles de cada articulación:



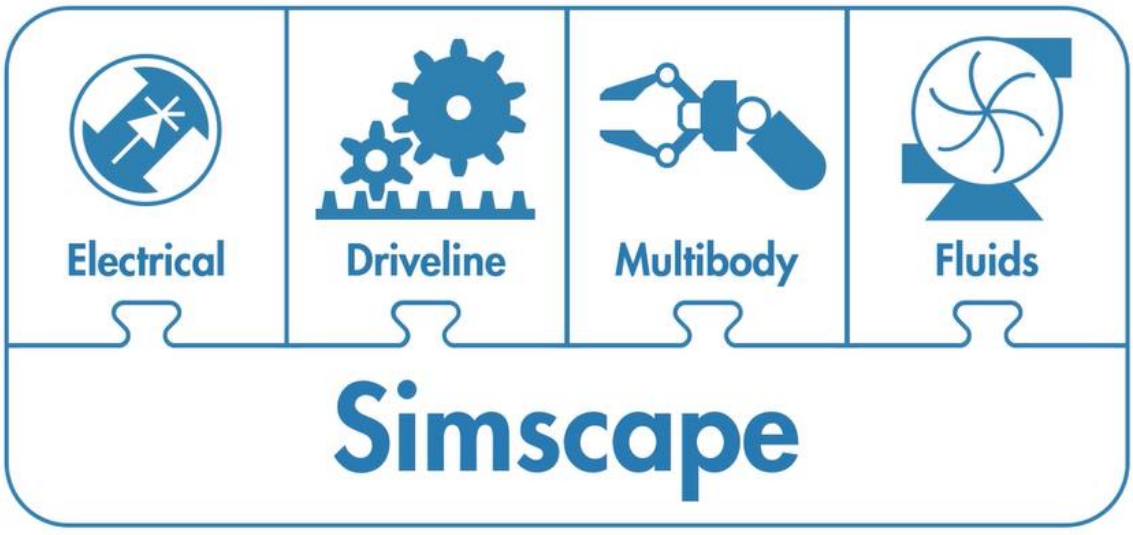
Bloques Simulink 2: Bloque eslabón de un robot

En cada articulación del robot se utilizan diferentes bloques encargados de la transformación de un tipo de elemento a otro (rotación, traslación…). Así con el bloque "joint_Origin_Transform" se da una transformación desde el sistema origen hasta la articulación (joint), "Visual_Origin_Transformation" aplica la transformación desde el origen hasta el bloque que define las características visuales de la articulación y "Inertia_Origin_Transform" hace lo mismo con los aspectos inerciales.

2.3.4 SIMSCAPE MULTI-BODY

Las gráficas de Matlab proporcionan simulaciones muy lentas y poco eficientes, es por ello por lo que se acude a la herramienta Simscape Multibody [14], que nos permite definir el entorno adecuado y simular nuestros robots en tres dimensiones.

Simscape es un entorno de programación visual de Simulink que nos proporciona la tecnología necesaria para simular y analizar sistemas físicos. Simscape ofrece algunos componentes fundamentales para varios dominios físicos; además en la familia de productos Simscape también encontramos Add-on que nos proporcionan modelos adicionales y capacidades de análisis para aplicaciones relacionadas con la energía eléctrica, la transmisión de movimiento, los sistemas mecánicos en 3D y la mecánica de fluidos. Todo esto se puede integrar con sistemas mecánicos en 3D para analizar el comportamiento del modelo y comprobar los resultados de la simulación.



Bloques Simulink 3: Familia Simscape [14]

Simscape Multibody proporciona un entorno de simulación para sistemas mecánicos en tres dimensiones, como robots, material de construcción o vehículos entre otros. Simscape Multibody plantea y resuelve las ecuaciones de movimiento para sistemas mecánicos completos. El sistema Multibody se modela asociando diferentes bloques que representan bodies, articulaciones y elementos de fuerza ente otros. Igualmente se pueden importar archivos CAD incluyendo todas las inercias, masas o geometrías 3d a nuestro modelo. Para comprender mejor la estructura de nuestro modelo, se genera automáticamente una animación 3D para visualizar la dinámica del sistema en la 'Mechanics Explorer'.

2.4 LIBRERÍA SIMROBOT

Este proyecto se plantea utilizando los conceptos fundamentales de la herramienta de Matlab/Simulink SimRob. Esto fue un método desarrollado en el TFG de Sandra Arévalo Fernández, antigua alumna de Ingeniería Industriales en la Universidad de Valladolid. [4] El objetivo de esta librería era permitir el diseño y simulación de diferentes estaciones robóticas de forma estandarizada. El usuario podrá jugar con varios robots y modificar sus estaciones fácilmente para su control. Para comprender el planteamiento de esta librería creada en el TFG mencionado se utilizarán las mismas imágenes que se incluyen en dicho TFG.

Lo interesante de esta herramienta es que combina es capaz de combinar los objetos resultantes tras importar un modelo URDF utilizando Robotic Toolbox y Simscape Multibody.

Es posible importar un objeto RigidBody de un fichero Simulink que a su vez ha sido obtenido de un URDF. Gracias a esto podemos tener en el objeto RigidBody todos los componentes añadidos al fichero Simulink. Así combinamos el objeto RigidBody que podremos utilizar para calcular la cinemática del robot y el fichero Simulink para realizar la simulación. Los resultados obtenidos se mostrarán en una simulación mediante Mechanics Explorer.

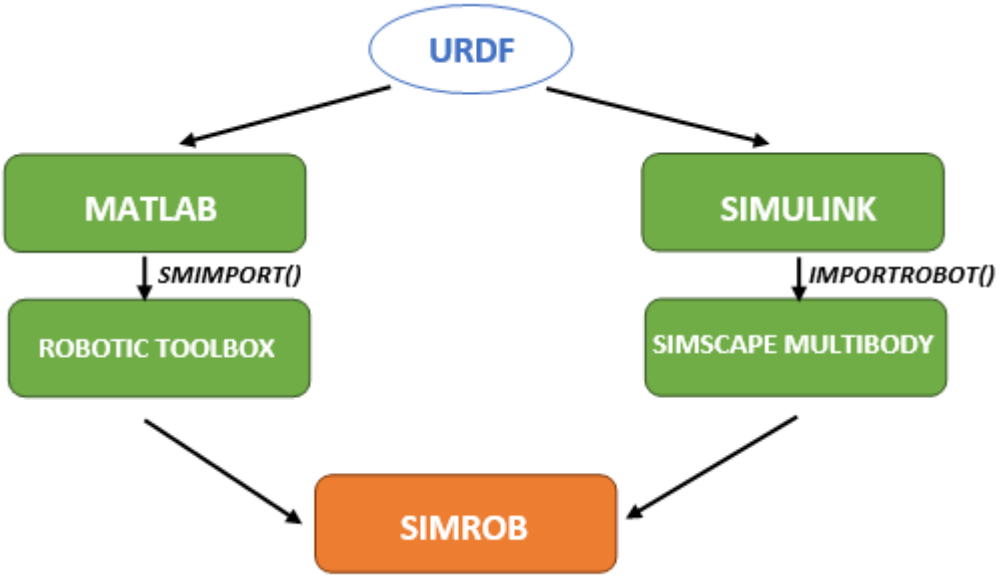


Ilustración 13: Relación SimRob y URDF

2.4.1 CREACIÓN LIBRERÍA DE ICONOS

Para convertir esta librería en un entorno interactivo para el usuario, se incorporaron una serie de robots y piezas para poder diseñar la estación como se desee.

2.4.1.1 ICONOS DE ROBOTS

En dicha librería se incorporaron varios brazos robóticos cuyo número de grados de libertad, configuración articular e incluso número de TCP varía.

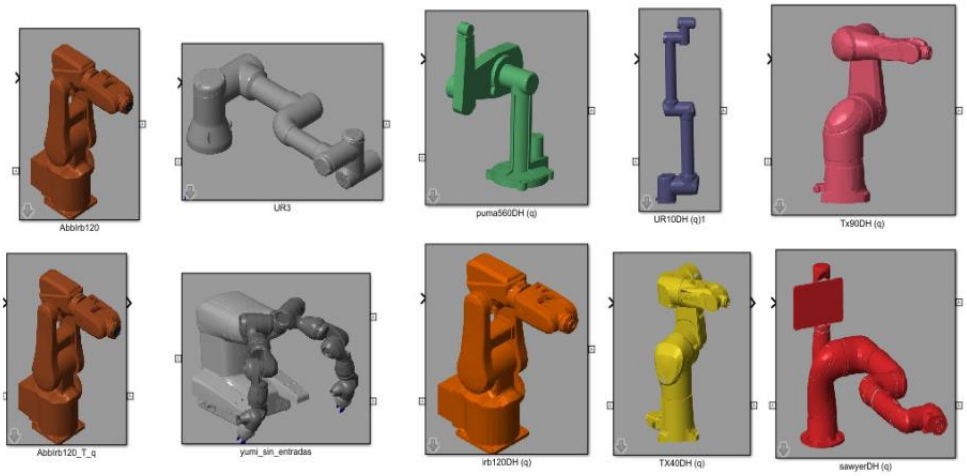
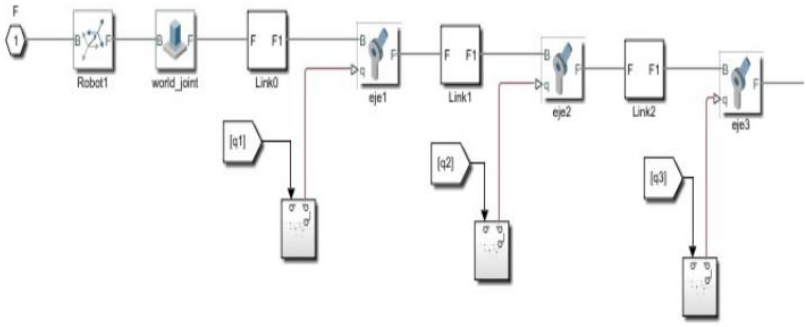


Ilustración 14: Iconos diversos robots librería SimRob [4]

La entrada a cada articulación de los robots en Simulink es la posición de cada articulación, ya que se querían controlar estos robots utilizando un modelo cinemático. Unido a la base del robot hay un bloque Rigid Transform definido como Robt1. En este bloque se define la posición y orientación de la base del robot respecto al sistema global de la estación.



Bloques Simulink 4: Estructura interna de un robot en librería SimRob

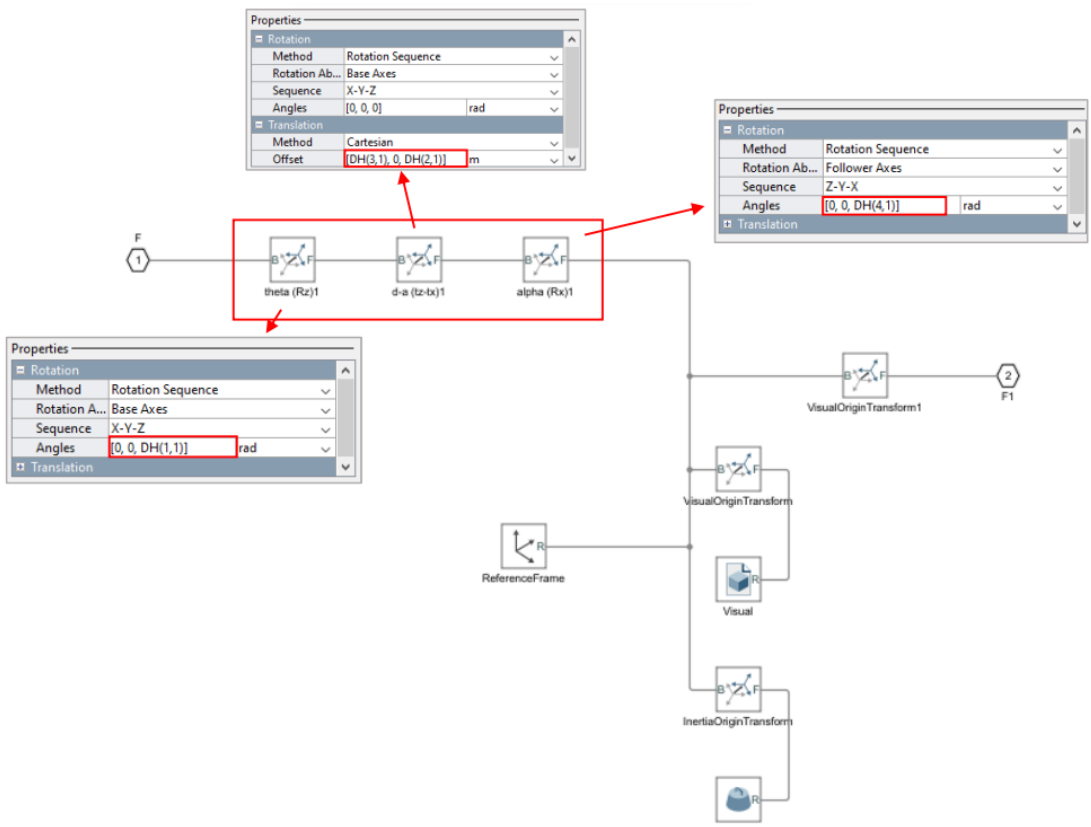
Por otro lado, a cada articulación se le ha pasado el valor de la coordenada articular y la posición del robot inicial que se pasa mediante una máscara.

Los iconos fueron obtenidos combinando los dos métodos mencionados anteriormente:

*Importando el robot a partir de un fichero URDF, del que obtenemos toda la información sobre cinemática y estructura del robot.

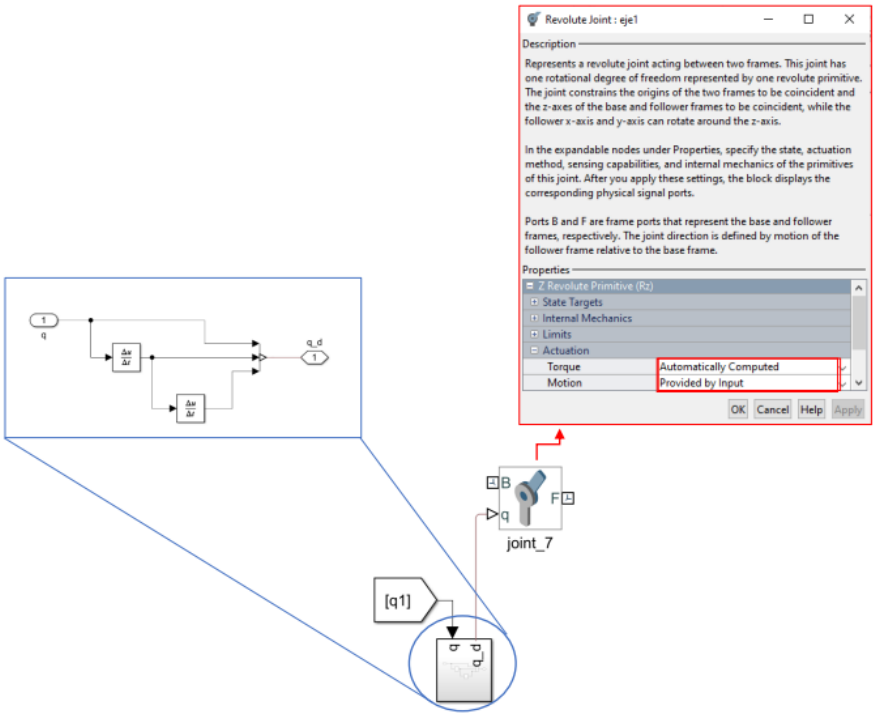
*Calculando los parámetros Denavit-Hartenberg de cada robot. Para que sea más accesible modificar los parámetros se ha creado una máscara en cada icono donde se define la matriz DH.

Se ha modificado el archivo de Simulink que se obtiene al importar un fichero URDF. A este archivo este sistema Simulink se han incorporado los parámetros DH que definen la relación entre un eje del robot y el eje siguiente. Los parámetros DH se han pasado en forma de bloques Rigid Transform, de forma que se le pasa como argumento los valores de la matriz DH mediante una máscara.



Bloques Simulink 5: Estructura interna de un brazo del robot en librería SimRob

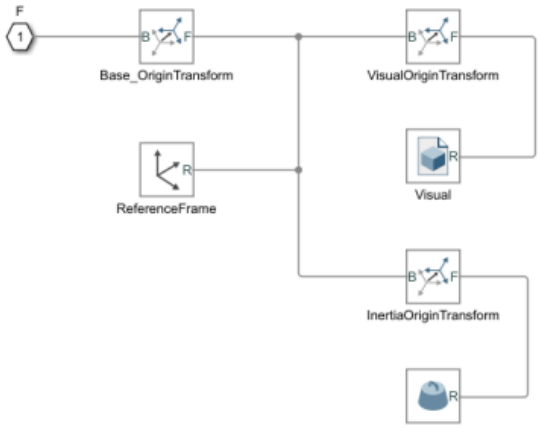
Entonces a la hora de modelar un robot solo habría que definir su configuración (adaptar el modelo en función del número de articulaciones y brazos y añadir los archivos. stl correspondientes) e inicializar ciertas variables como los parámetros DH, el color o algunos parámetros dinámicos que se exigen en el bloque de inercia. Como estamos introduciendo directamente la posición a la que queremos llegar en las articulaciones, hay que activar la opción de calcular el torque automáticamente para que haya una actuación sobre las articulaciones durante la simulación.



Bloques Simulink 6: Bloque joint con ángulo de entrada librería SimRob

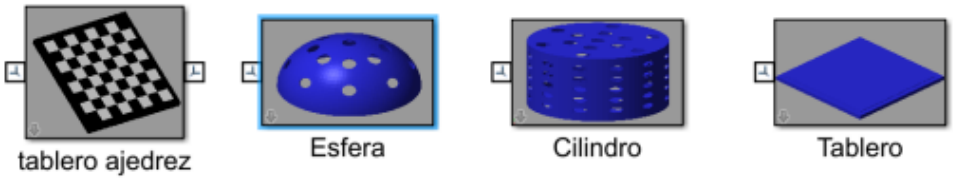
2.4.1.2 ICONOS DE TRABAJO

Igualmente se han diseñado iconos de trabajo que definen objetos de trabajo que pueden estar presentes en el entorno de la simulación. Para conseguirlo se ha creado un subsistema con diferentes bloques que definen la orientación del objeto, sus masas o sus características visuales entre otros aspectos. Para cambiar todos los parámetros distintivos del objeto, se ha establecido una máscara con la que se podrá modificar el icono fácilmente.

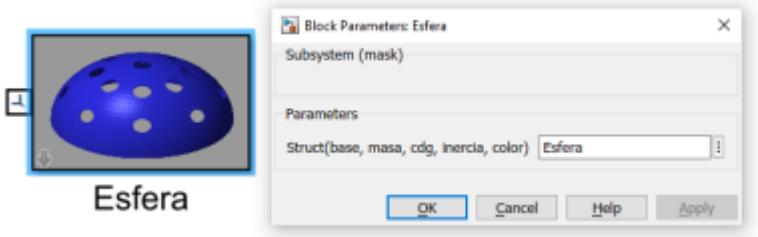


Bloques Simulink 7: Subsistema de un objeto de trabajo librería SimRob

En la librería encontramos los siguientes objetos de trabajo, pero gracias a la configuración de los bloques de Simulink el usuario puede diseñar nuevos objetos de trabajo con herramientas gráficas como CATIA o AutodeskInventor y así generar ficheros .stl propios que podrá pasar a los bloques.



Bloques Simulink 8: Ejemplos iconos de trabajo librería SimRob

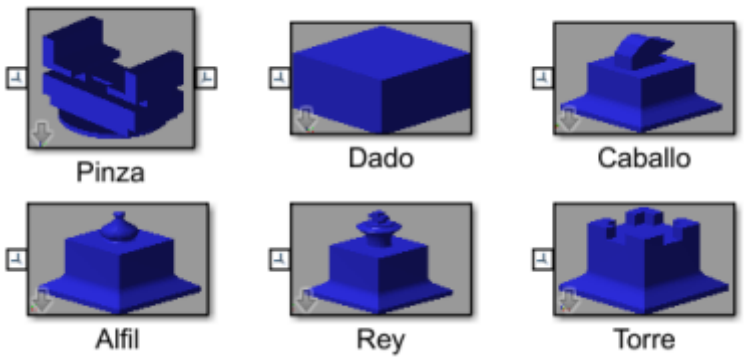


Bloques Simulink 9: Icono de trabajo con parámetros en librería SimRob

Como se puede comprobar, al pasar el objeto Esfera se están pasando a cada uno de los bloques del subsistema que constituye el objeto de trabajo la base del objeto, su masa, su centro de gravedad, inercia y color.

2.4.1.3 ICONOS DE HERRAMIENTAS O CARGAS

Por último, se han creado un icono que sirve como herramienta para que el robot pueda manipular cargas; esta es una pinza que se situará en el extremo del brazo robótico. Igualmente se han incorporado objetos que funcionan como cargas, en este caso son distintas piezas de ajedrez y un dado.



Bloques Simulink 10: Iconos de herramientas o carga en librería SimRob

Como estos objetos se posicionan y se orientan en función del TCP del robot, se ha añadido un bloque RigidTransform que establece la posición de la base del icono referida respecto al sistema TCP del robot, lo que establece el extremo final del robot. Además, se ha introducido un bloque joint_tool que permitirá acoplar una pieza a la siguiente (la herramienta a la carga).

FUNCIÓN PIEZAS()

Los parámetros de dicha máscara (base, color, cdg…) pueden cambiarse de forma manual, lo que ofrece una libertad completa al usuario para definir los objetos de la estación como el quiero. A pesar de esto, para ahorrar tiempo y esfuerzo al usuario se ha creado una función Piezas() que permite introducir todos estos parámetros con una sola variable de entrada.

Los objetos de la clase pieza guardan como información los parámetros requeridos en Simulink y el archivo .stl del objeto. Igualmente, esto es útil porque en Simulink no existe la posibilidad de crear iconos variables, por lo que es necesario implementar algún método para que en la simulación el robot pueda coger y dejar cargas. La función Piezas() se puede aplicar tanto a iconos de trabajo como de carga y herramientas:

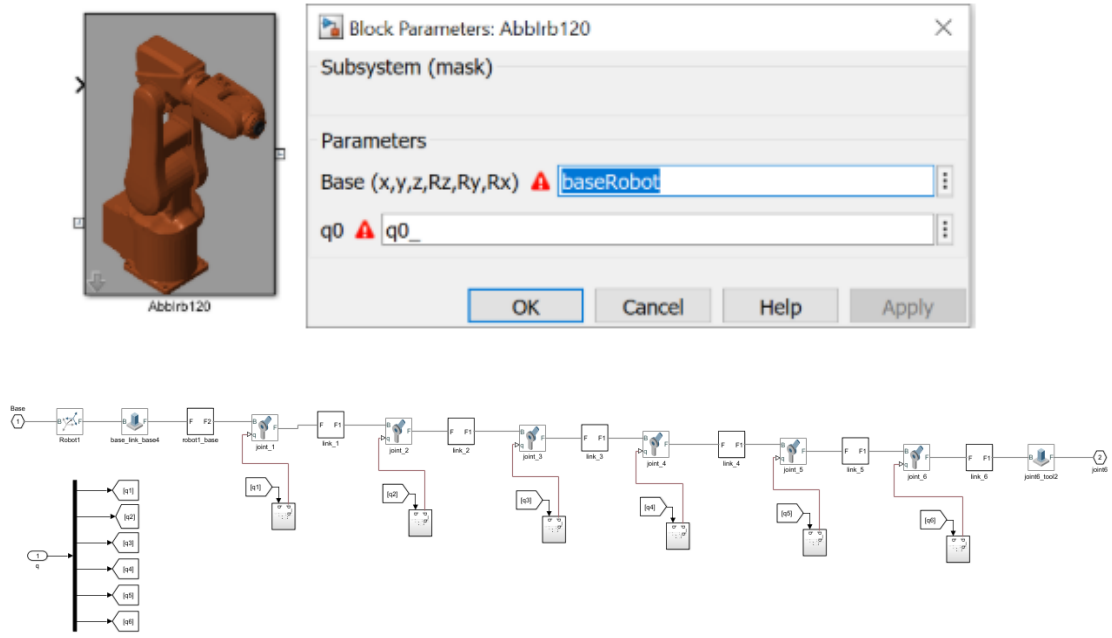
*Con la función Pieza(), si los ficheros .stl son constantes (fijos que son los iconos de trabajo), se modelan directamente desde el fichero Biblioteca

*Con la función Pieza(), si los ficheros .stl son variables (el elemento asociado cambia de propiedades, son la carga y la pinza), se modelan a partir de ficheros temporales del directorio auxStl.

Gracias a esta función se pueden incorporar objetos al entorno fácilmente y tendremos la posibilidad de coger y dejar objetos, como se explicará más adelante.

2.4.2 MODELADO CINEMÁTICO ROBOT

Para comprender mejor el comportamiento de la estación de trabajo y el modelo de robot escogeremos uno de los robots de la librería SimRob y analizaremos su modelado cinemático. Se escoge para trabajar el modelo de ABB irb120. Se trata de un brazo robótico de 6 grados de libertad con una herramienta en forma de pinza. El robot está unido al sistema global del sistema, por lo que permanecerá fijo en el suelo.



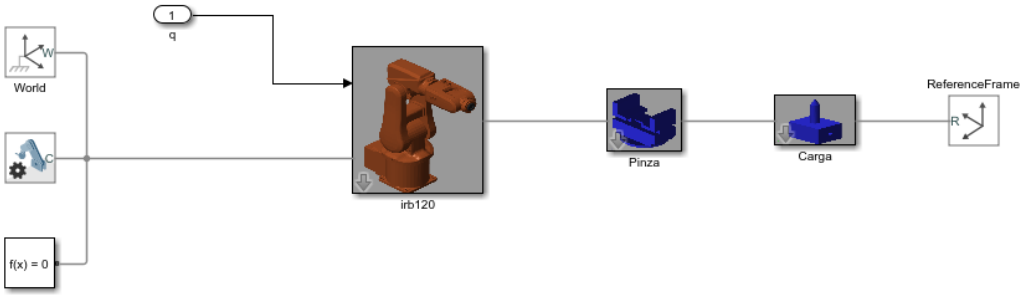
Bloques Simulink 11: Robot modelo ABB irb120

Como vamos a estudiar la cinemática del robot, la entrada a los bloques de las articulaciones es la posición, de forma que el robot llegará a la posición introducida en cada una de las articulaciones. Se pasa como parámetros de la máscara al icono del robot una variable para definir la base posición y orientación de la base del robot y otra variable para definir la configuración articular inicial.

Como se ha explicado anteriormente, cada brazo del robot cuenta con una serie de bloques que definen su posición, orientación, masa, inercia o modelo visual entre otros aspectos.

En primer lugar, estudiamos la cinemática directa del robot, es decir, la obtención del TCP del robot a partir de las coordenadas articulares. Los datos

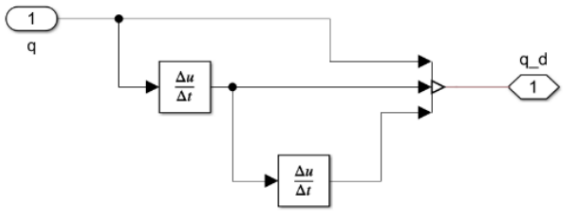
que introduce el usuario serán los ángulos que se desean mover cada una de las articulaciones del robot; a partir de estos datos el robot cambiará su configuración.



Bloques Simulink 12: Estación modelo ABB irb120

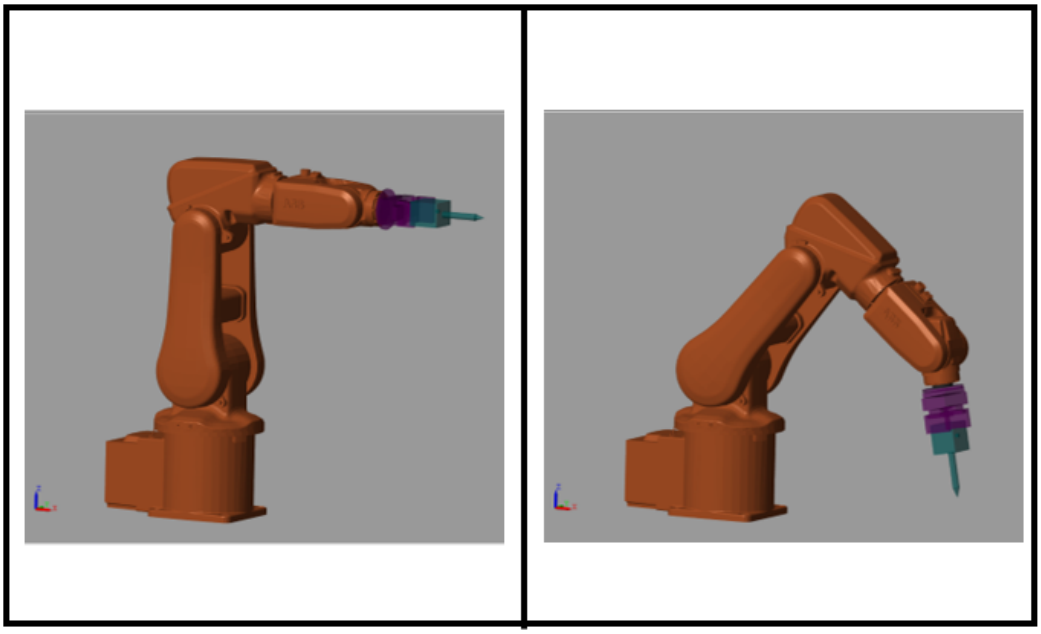
En el interior de cada bloque de posición de las articulaciones se calcula la primera y segunda derivada de la variable q. Estas derivadas se emplean a la hora de calcular el torque que computariza automáticamente (opción elegida en los bloques de articulaciones) Matlab mediante la siguiente ecuación:

$$\tau = J\frac{d^2q}{dt^2} + B\frac{dq}{dt} + Kq \qquad (12)$$



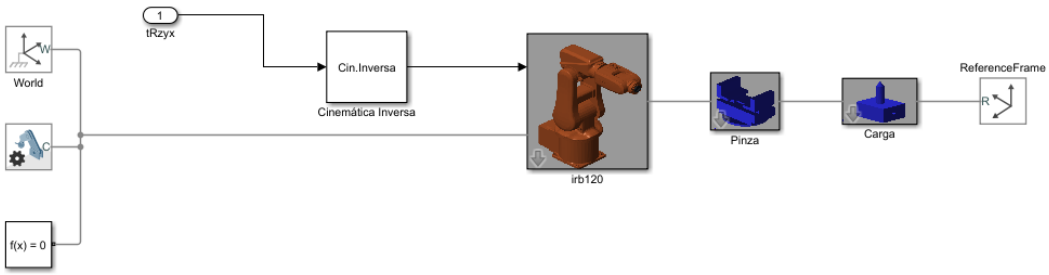
Bloques Simulink 13: Bloque posición

El resultado es la simulación del robot a una configuración articular concreta (rotación 30 grados de las articulaciones 2 y 5)



Simulación 1: ABB irb120 cinemática directa 30 grados

Para plantear el modelo cinemático inverso, se ha introducido un bloque encargado específicamente de realizar la transformación de un vector de posición de tipo [x, y, z, Rx, Ry, Rz], que representa el TCP del robot, a coordenadas articulares. En este caso el usuario introduce dicho vector de posición y tras la transformación se introduce de nuevo la posición articular en el subsistema del robot para que este alcance una determinada configuración articular.



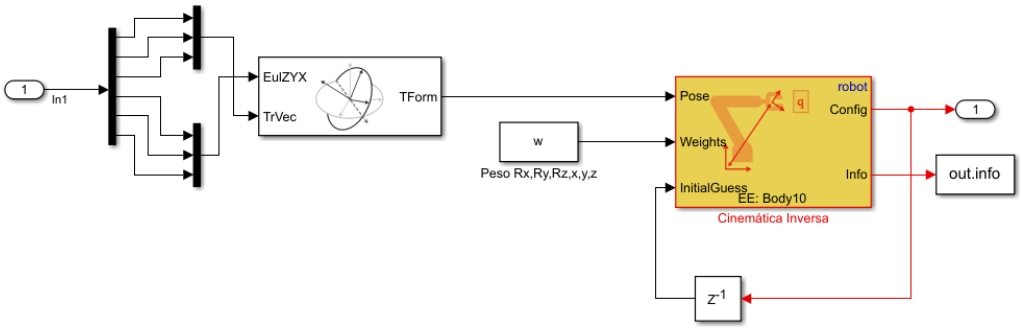
Bloques Simulink 14: Estación modelo ABB irb120 (cinemática inversa)

En el bloque de cinemática inverso se han dispuesto diferentes subbloques para obtener como resultado las coordenadas articulares del robot.

En primer lugar, el vector de entrada con las coordenadas que indican la posición y la rotación del TCP se introduce a un bloque “Coordinate

Transformation Conversion" que convierte las coordenadas que representan la entrada (en este caso vector de traslación para la posición y ángulos de Euler para la rotación) a una representación específica (TForm, matriz de transformación homogénea en este caso). Para que esta conversión se realice correctamente todos los vectores deben ser de tipo columna.

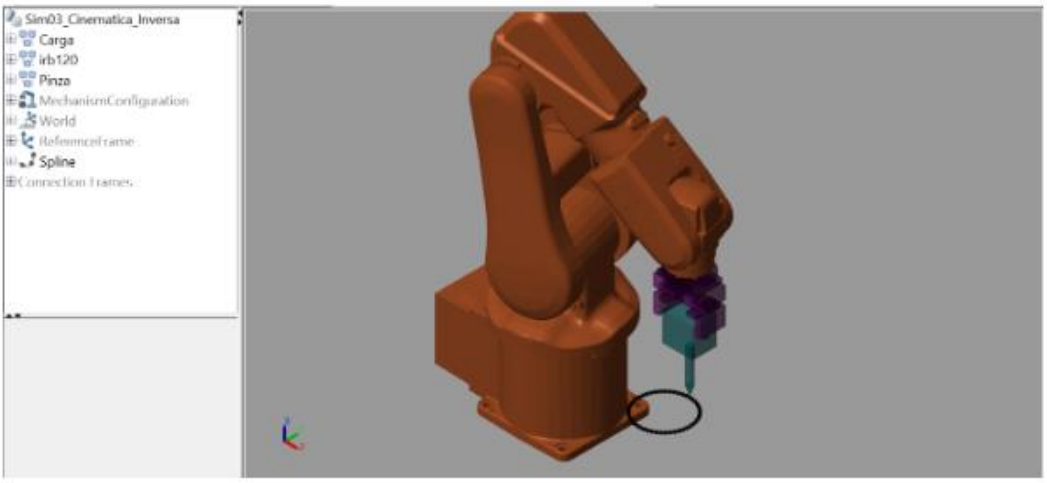
A continuación, el bloque "Inverse Kinematics" utiliza un solver (solucionador) de cinemática inversa para calcular las coordenadas articulares en función del pose del efector final. Además, se debe especificar los pesos para establecer un método de optimización y una aproximación inicial de la configuración del modelo robótico. El vector de pesos indica la importancia que se le dan a los elementos del vector que determina el TCP. Por ejemplo, si al vector [Rx, Ry, Rz, x, y, z] se le da el valor de [0 0 0 1 1 1], entonces se optimiza la posición, es decir, el robot realizará su movimiento dando prioridad a la posición del TCP antes que a su rotación.



Bloques Simulink 15: Bloque cinemática inversa

La librería Robotic System Toolbox ofrece algoritmos de optimización local a la hora de calcular la cinemática inversa. Esto significa que el algoritmo modifica los parámetros continuamente hasta que consiga pasar de un punto inicial al punto deseado. Matlab plantea un método iterativo en el que se parte de una aproximación inicial de la solución y se busca minimizar el error (función de coste) entre la configuración real y la configuración deseada. Cuando el error es nulo, entonces el robot ha alcanzado las coordenadas articulares adecuadas y se para el proceso iterativo del algoritmo.

Mediante cinemática inversa puedes indicar al robot que vaya a un punto o realice una trayectoria concreta.



Simulación 2: ABB irb120 Cinemática inversa

2.4.3 CLASE KIN

Además de plantear el modelo cinemático inverso mediante un bloque de Simulink, también es posible definir la cinemática inversa utilizando una clase destinada al movimiento de modelos robóticos.

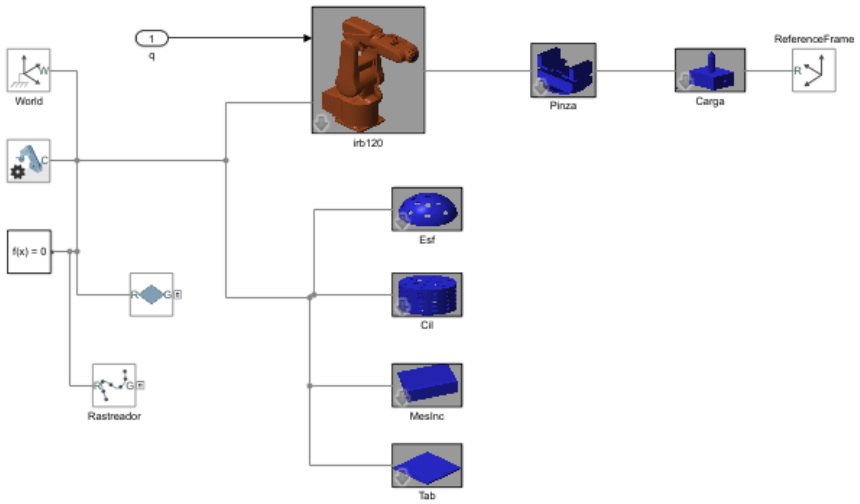
Para conseguir mover al robot aplicando la cinemática se empleó una clase desarrollada por mi tutor Alberto Herreros, clase *Kin*, para poder realizar un simulador del robot empleando las herramientas de Multibody de Simulink y RigidBody de Matlab explicadas anteriormente que se utilizan en la librería SimRob.

La clase Kin nos da la relación que existe entre los puntos inicial y final al que se quiere llegar con respecto a los ejes del robot. Este plantear el modelo cinemático directo utilizando las coordenadas articulares y el cinemático inverso por optimización local.

Los métodos fundamentales de la clase Kin son los movimientos habituales de cualquier lenguaje para la programación de un sistema robotizado: MoveAbsJ (movimiento por coordenadas articulares), MoveJ (movimientos por coordenadas cartesianas),MoveL (movimiento en línea) y MoveC (movimiento en arco). El método MoveAbsJ() recibe como argumento la configuración articular del robot, por lo que aplica la cinemática directa para mover las articulaciones del robot. El resto de los métodos de movimiento del objeto Kin están destinados a la resolución del modelo cinemático inverso (MoveJ(), MoveL(), MoveC()…).

A la clase Kin se le pasa como argumento el RigidBody asociado a la estación, la estación Simulink, la herramienta deseada y el valor de las coordenadas articulares iniciales (q0=[0 0 0 0 0 0]). El objeto RigidBody será el que defina el robot; este no se visualizará, solo se emplea para calcular la cinemática directa e inversa. Por otra parte, el fichero Simulink Multibody se pueden añadir las herramientas que se quieran; se emplea MultiBody ya que la simulación de movimientos es más continua en Simulink como se explicó en apartados anteriores.

Por ejemplo, para el modelo ARB120, podemos distinguir entre el RigidBody, que se trata del propio robot de Matlab, y la estación Simulink en la que se encuentren las herramientas y los objetos con los que trabajará en robot.



Bloques Simulink 16: Estación Simulink Multibody con objetos y herramientas

El objeto Kin rob se define de la siguiente manera:

```
rob= Kin(robot, Estacion, bodyCarga, q0);
```

A partir del objeto Kin que se obtiene, se pueden aplicar diferentes métodos para incluir y quitar cuerpos, grabar los movimientos del robot, reproducirlos… que se detallan en “Capítulo VI: Anexos”.

En conclusión, con la clase Kin se ha conseguido una herramienta que calcula la cinemática directa e inversa igual que lo hace cualquier simulador. Para ello Kin importa el modelo Robotic Toolbox que le permite calcular las cinemáticas inversas. Gracias al desarrollo de la librería SimRob podemos no solo calcular la cinemática del robot sino simular su movimiento gracias a la incorporación de Simscape Multibody.

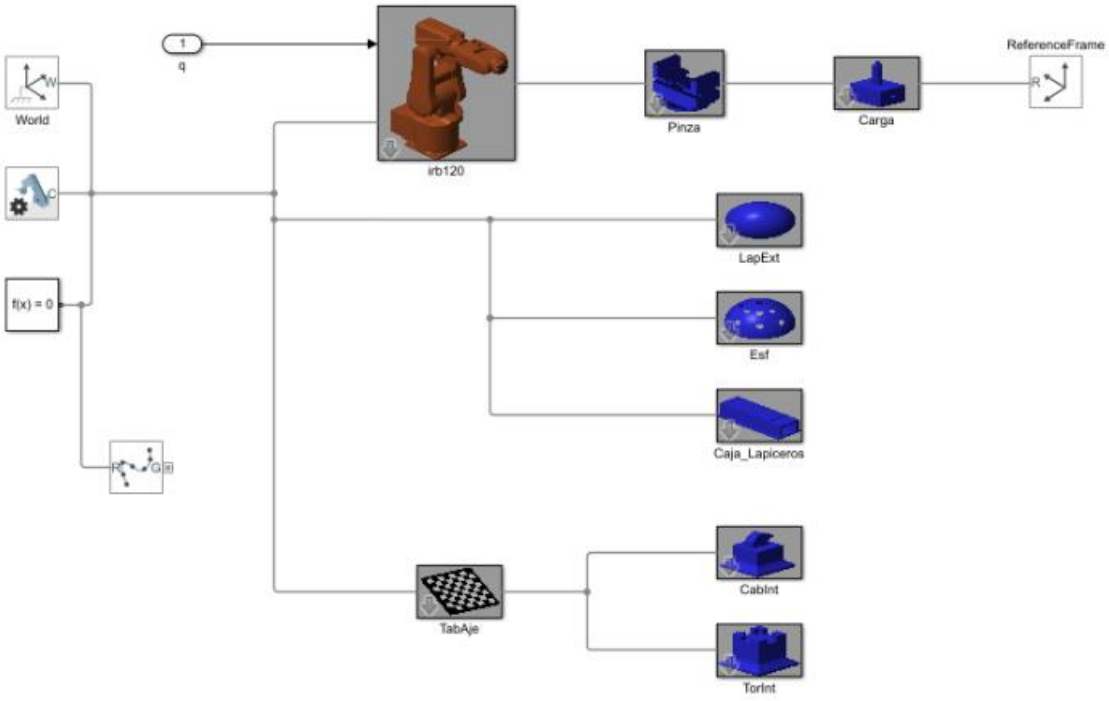
CLASE PIEZA (PEGAR EN)

La clase Kin tiene muchas funciones encargadas del movimiento del robot de forma cinemática. En caso de que se desee manipular cargas y jugar con la herramienta, es necesario crear un nuevo método. Para conseguir esto se ha creado dentro de la clase Pieza el método PegarEn que permite pegar todas las propiedades de un objeto en otro objeto.

PegarEn(this1, this2, wobj1, wobj2)

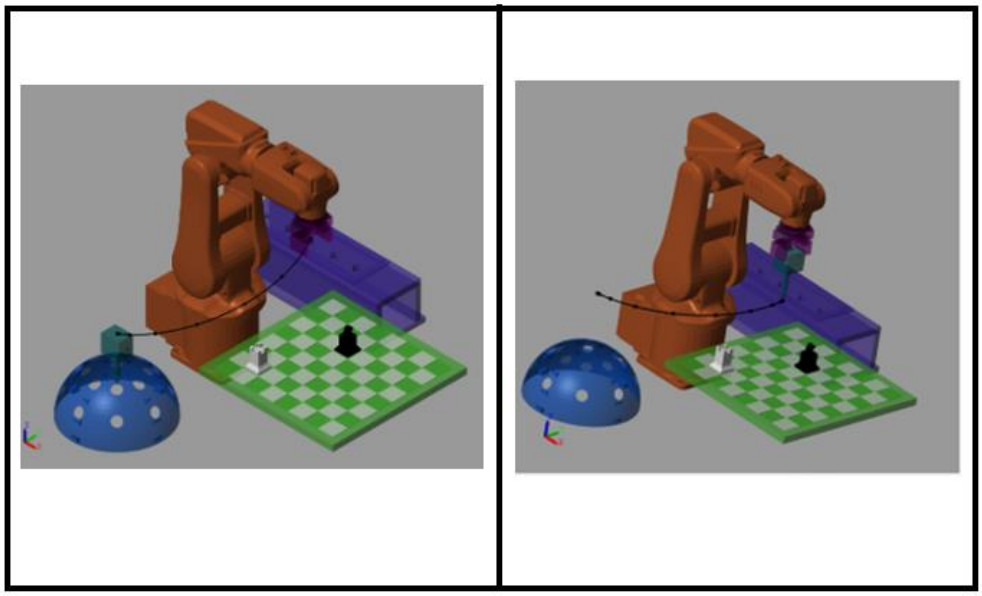
De esta forma se pegan las propiedades del objeto this2 en this1 donde wobj1 es la referencia [x,y,z,Rz,Ry,Rx] del objeto this1 y wobj2 es la referencia [x,y,z,Rz,Ry,Rx] del objeto this2. La base del objeto this1 se modifica de forma que en valor absoluto el objeto no se mueva.

Para poner la pieza que queremos que esté asociada a la base que no se mueve, cargamos vacío (Null) para poner la pieza fuera de la carga; gracias a esto se mueve el robot, pero la pieza permanecerá en su posición. En caso de que queramos que una pieza se mueva con la herramienta, intercambiaremos la pieza asociada a la base con la carga del robot.



En el diagrama superior se ha definido un objeto LapExt como un icono con .stl nulo. Para dejar el lápiz en el entorno de la simulación, por ejemplo, en el objeto esfera, se deja la carga que se está manipulando en una posición relativa a la esfera. Para tomar la carga se intercambia la pieza del icono exterior (LapExt) a la carga del robot.

```
PegarEn(LapExt, Carga, [], aux.tRzyx(rob.Pose()));

PegarEn(Carga, LapExt, aux.tRzyx(rob.Pose), []);
```



Simulación 3: ABB irb120 Tomar y dejar lápiz

La librería SimRob presenta aspectos muy interesantes para modelar robots, como por ejemplo el uso de RigidBody para calcular la cinemática del robot y el entorno Simscape para realizar la simulación. Además, el hecho de realizar una librería con varios brazos robóticos es muy interesante para comprender las similitudes entre los diferentes robots que da libertad al usuario para crear la estación como más le guste. La estandarización de los elementos de Simulink y los métodos para aplicarlo a varios modelos puede resultar muy útil tanto a nivel de simulación como en un modelo robótico real.

Este proyecto pretende ser una ampliación de esta librería, de forma que se emplearan los fundamentos de esta herramienta para diseñar varias estaciones para robots humanoides. Además, esta herramienta tiene un gran potencial para la resolución del problema no solo cinemático sino dinámico de un robot; por todo esto, el trabajo se centrará en el estudio dinámico de varios robots humanoides como parte una extensión de la librería SimRob.

CAPÍTULO III: MODELADO DINÁMICO Y FUERZAS EXTERNAS

Casi todos los simuladores de robótica trabajan utilizando el modelo cinemático; este método es de resolución sencilla y siempre obtendremos una solución exacta a partir de la ecuación. La ventaja que nos ofrece Simscape que no presentan otros simuladores como RobotStudio es que se puede llevar a cabo el estudio dinámico del robot

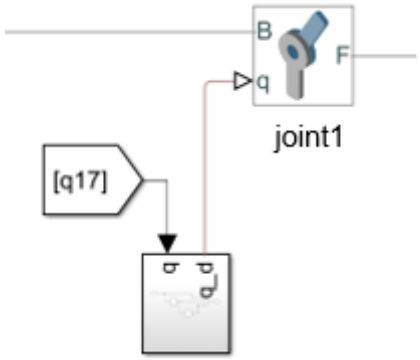
El proyecto se enfocará en el estudio del modelo dinámico de los robots. Inicialmente se utilizará el mismo robot ABB irb120 utilizado en los capítulos anteriores para comprender el funcionamiento de la dinámica en un robot cualquiera.

3.1 MODELADO DINÁMICO ROBOT

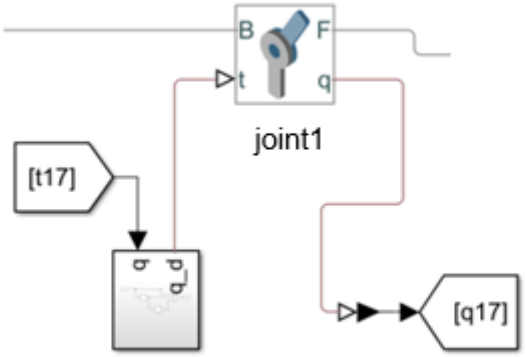
3.1.1 DISEÑO DE LA ESTACIÓN

A la hora de estudiar la dinámica del robot, la entrada que se pasa al robot es directamente el par que se va a aplicar sobre cada articulación. A partir del valor del torque, se calcularán las coordenadas articulares que debe alcanzar el robot. El modelo dinámico tiene en cuenta las masas y fuerzas que actúan sobre el robot, por ello representa el comportamiento de un modelo robótico real en el que, en función de los parámetros de entrada, el robot puede llegar a su objetivo final o no.

Los bloques de las articulaciones tienen la entrada B y la salida F que presentaban también las articulaciones del modelo cinemático; representan los sistemas de ejes sobre el que se posiciona la articulación actual y la siguiente. La diferencia de la estación con el modelo cinemático es que en los bloques de las articulaciones se le pasa el torque como entrada y tienen como salida la posición de la articulación, lo que nos permitirá controlar el movimiento del robot para que llegue a la posición adecuada, como se explicará en el apartado “2.2.3 Controladores básicos y arquitectura de control”. Esto es diferente del caso cinemático, donde los bloques de las articulaciones solo presentaban una entrada que era el ángulo que se quería mover cada articulación. No presentaba ninguna salida ya que no tenía sentido establecer un sistema de control para el caso cinemático porque se consigue un movimiento ideal a partir de la q de entrada.



Bloques Simulink 17: Articulación modelo cinemático

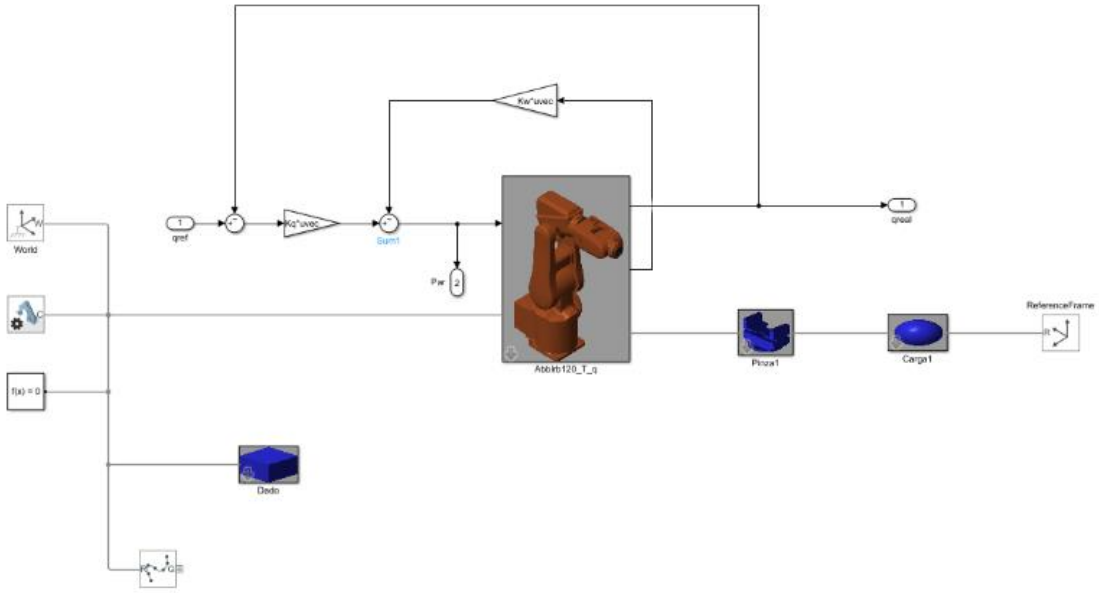


Bloques Simulink 18: Articulación modelo dinámico

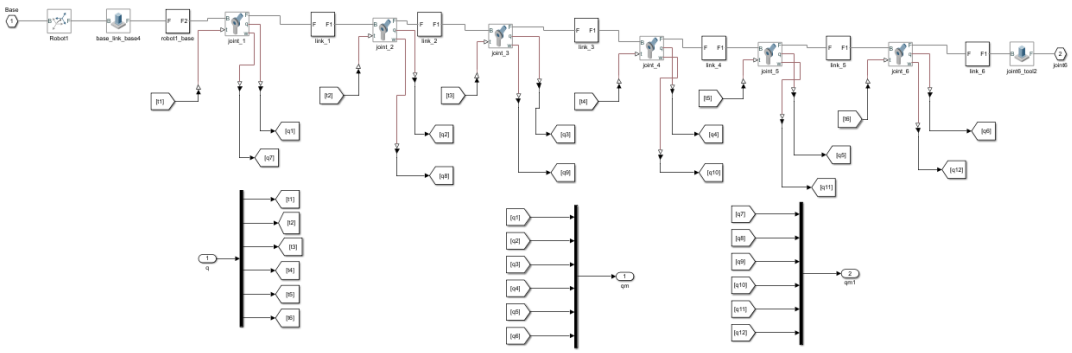
Cuando la diferencia entre la posición articular real y deseada del robot es mayor que cero, significa que el robot aún no ha alcanzado su q final, por lo que se continúa introduciendo par al sistema para que las articulaciones continúen su movimiento. Si el error es cero (diferencia es nula), entonces significa que las articulaciones han llegado al sitio que queríamos por lo que se deja de aplicar torque en las articulaciones.

Además, se ha añadido otro lazo de control que se encarga de seguir una velocidad de referencia para minimizar el error y eliminar las oscilaciones del robot que se explicará más adelante.

El robot está unido al sistema global del sistema, por lo que permanecerá fijo en el suelo.



Bloques Simulink 19: Modelo robótico ABB irb120 dinámica

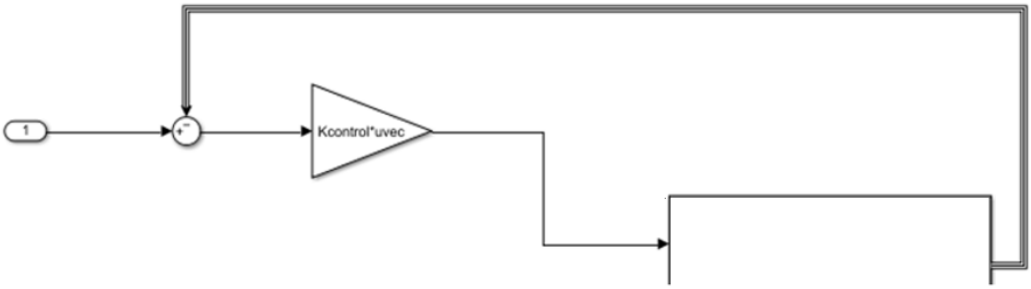


Bloques Simulink 20: Robot ABB irb120 dinámica

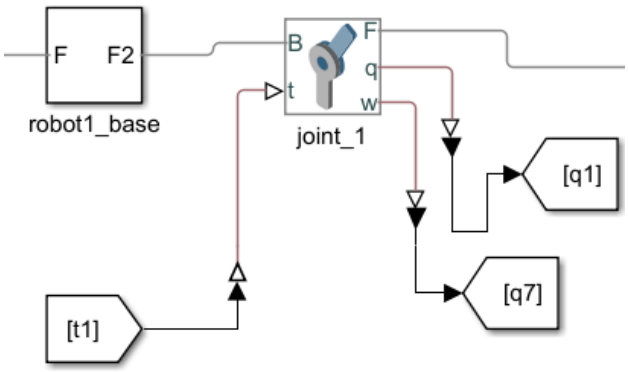
3.1.2 LAZO INTERNO DE CONTROL

El controlador que se aplica para controlar la dinámica del robot en un principio es de acción proporcional. Una vez que se ha realizado la diferencia entre la posición articular real del robot y la posición a la entrada del sistema, el resultado pasa por un controlador proporcional. Si la diferencia de posiciones es muy grande, significa que el robot aún no ha alcanzado su q final, por lo que la salida del controlador aumentará para introducir un torque mayor a los motores y que estos continúen su movimiento. Si el error es muy pequeño, entonces las articulaciones han llegado prácticamente a su posición

final y la salida del controlador se irá reduciendo para dejar de aplicar torque en las articulaciones.



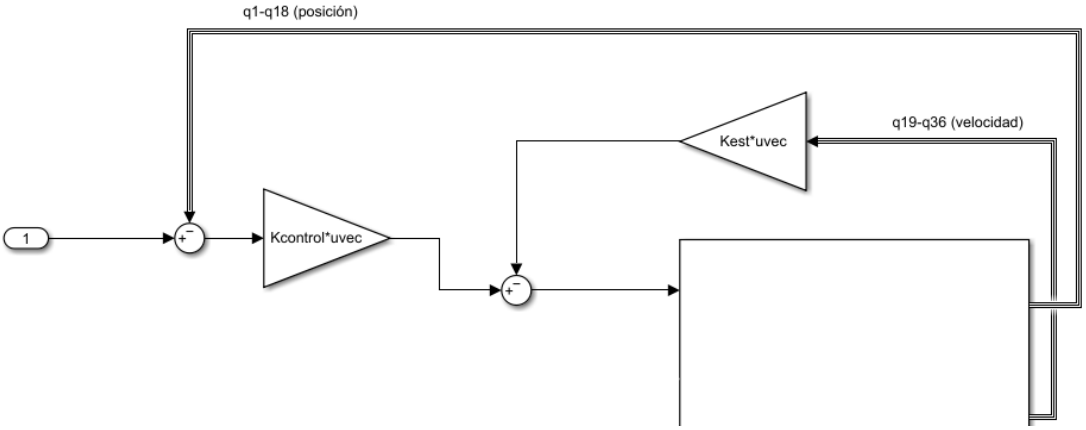
Para integrar el control derivativo al modelo robótico se modificaron los bloques de las articulaciones para obtener de salida la velocidad. Según la ecuación de la acción derivativa necesitamos la derivada del error (la magnitud a la entrada del controlador), que será la derivada de la posición articular, es decir, la velocidad articular. A diferencia de los bloques articulares en el modelo dinámico con control únicamente proporcional, los bloques en el modelo con control derivativo presentan una salida para la velocidad.



Bloques Simulink 21: Articulación modelo dinámico controlador PD

Es decir, el robot ABB irb120 presenta un control tanto de posición como de velocidad (proporcional y derivativo). Para ello se crearán dos salidas en cada bloque articulación que serán controlados por un lazo: del q1-q6 proporcional y del q7-q12 derivativo.

El controlador proporcional (Kcontrol) minimiza el error tomando como referencia la salida posición q del robot. Por otro lado, el controlador derivativo (Kest) no tiene referencia; su función es reducir las oscilaciones para que los movimientos de las articulaciones sean más suaves.

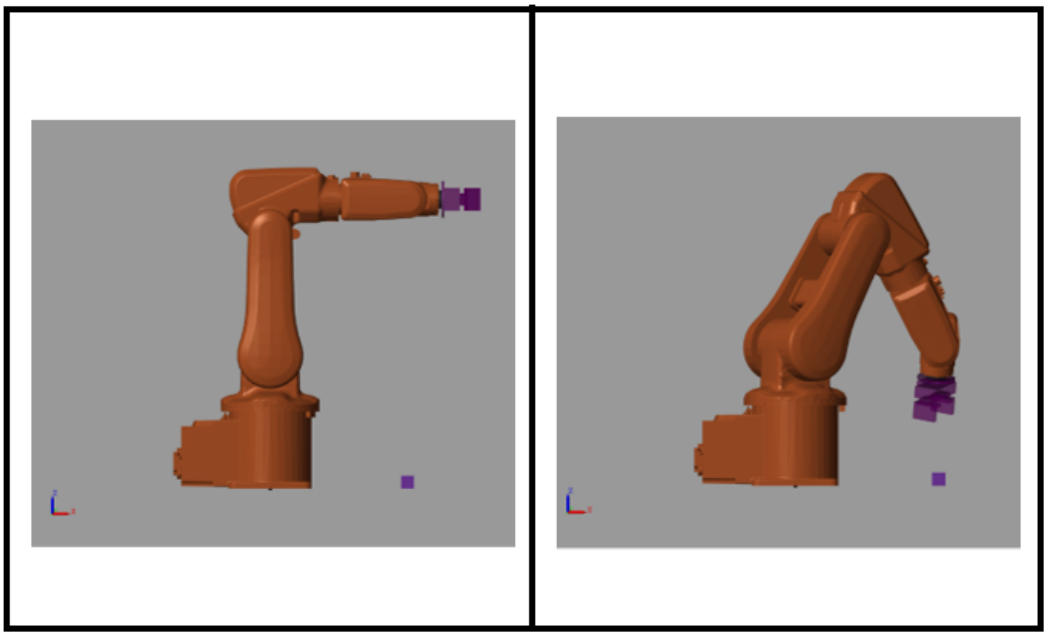


Así estamos planteando un controlador interno PD que aporta un núcleo interno de estabilidad donde la realimentación proporcional lleva al robot al objetivo y la derivativa elimina las vibraciones que dan inestabilidad al humanoide.

3.1.3 APLICACIONES DEL MODELO ROBÓTICO

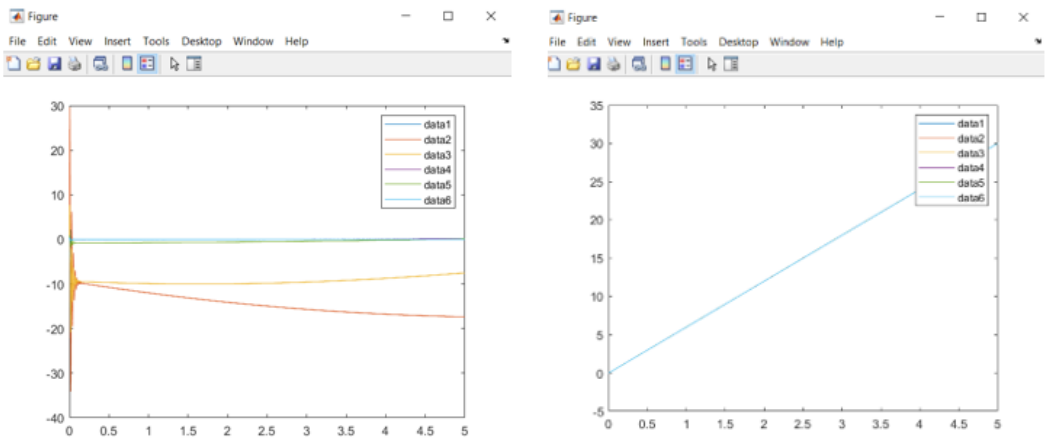
La clase Kin se puede usar tanto en estaciones modeladas con el problema dinámico como el cinemático; aunque la entrada al robot sea diferente (torque o posición articular), la entrada al sistema es la posición articular en los dos casos. Se podrá emplear la función Kin explicada anteriormente para calcular la cinemática directa e inversa del modelo robótico, es decir, para calcular relación que hay entre los dos puntos y los ejes del robot; no tiene relación con el modo de simulación del robot.

Realizamos pruebas para comprender el funcionamiento de este modelo. En primer lugar, estudiaremos la cinemática directa en el robot ABB rib120. Damos la orden de movimiento de rotación de 30 grados a todas las articulaciones del robot. El robot es capaz de realizar el movimiento y esto lo ha conseguido aplicando un torque determinado en sus articulaciones.



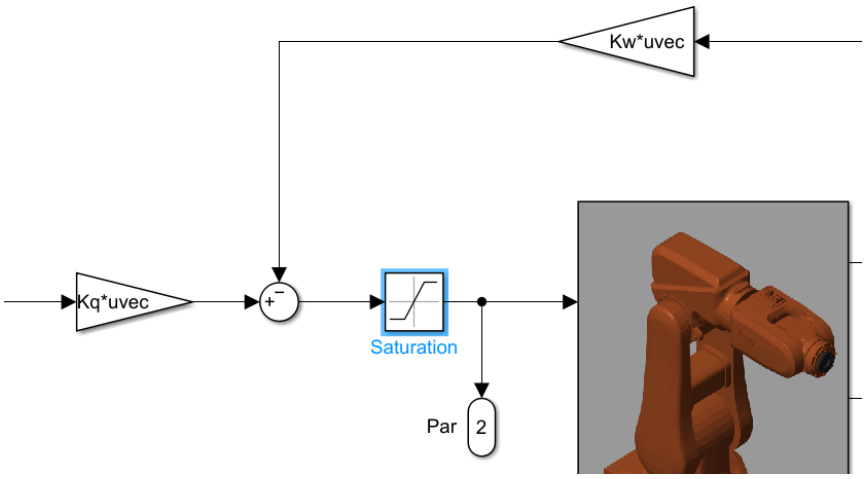
Simulación 4: ABB irb120 rotación 30 grados con cinemática directa por dinámica

Representamos en una gráfica la evaluación del par que se ejerce en cada articulación. Para que el robot alcance la posición inicial, ha sido necesario aplicar un torque mucho mayor de forma absoluta en las articulaciones 2 y 3 que en las 5 y 6. Esto tiene sentido porque en los brazos robóticos industriales es normal que los motores de la base del robot (se encargan de sujetar el brazo de mayor tamaño) tengan que ejercer más fuerza para mover el conjunto de bodies del robot. Los motores de las articulaciones del extremo no requieren un par tan elevado porque solo se encargan de rotar el extremo final del robot que pesa mucho menos. Además, vemos que todas las articulaciones llegan a la posición articular de 30 grados de forma lineal en la simulación.



Gráfica 1: Torque y posición en articulaciones (rotación q 30 grados)

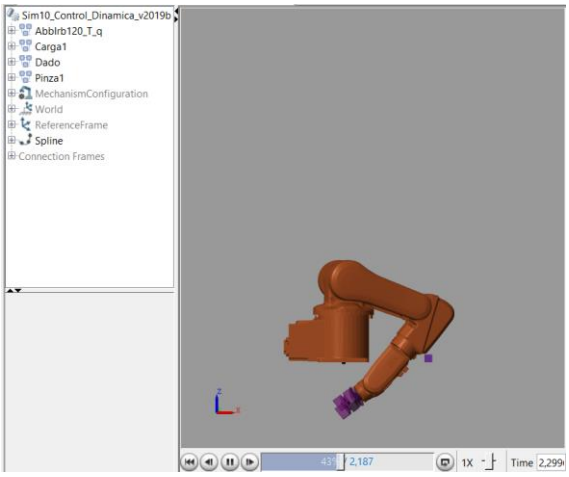
En los sistemas robóticos reales, los motores tienen unos límites de torque para que no roten más allá de una posición determinada. Se puede simular el efecto de este límite definiendo una variable que determine el torque de saturación máxima y mínima. Se añade un bloque de saturación una vez que se haya obtenido el par.



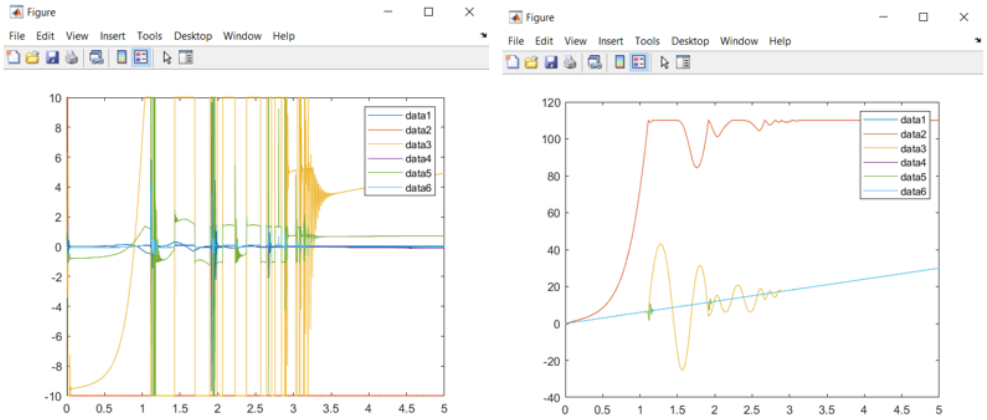
Bloques Simulink 22: Bloque saturación

Por ejemplo, si ponemos un torque de saturación de 10 la articulación 2 del robot no funcionará correctamente durante la simulación, ya que para alcanzar la posición articular deseada se tiene que ejercer un torque en la articulación 2 de -20 unidades. En otras palabras, con esta saturación los motores no soportan el peso del robot y las articulaciones no realizarán el movimiento demandado.

Este fallo se ve reflejado en los torques del resto de las articulaciones ya que se produce un desequilibrio general de las articulaciones del robot. Igualmente afecta a la posición articular final, provocando que varias articulaciones no lleguen a los 30 grados.



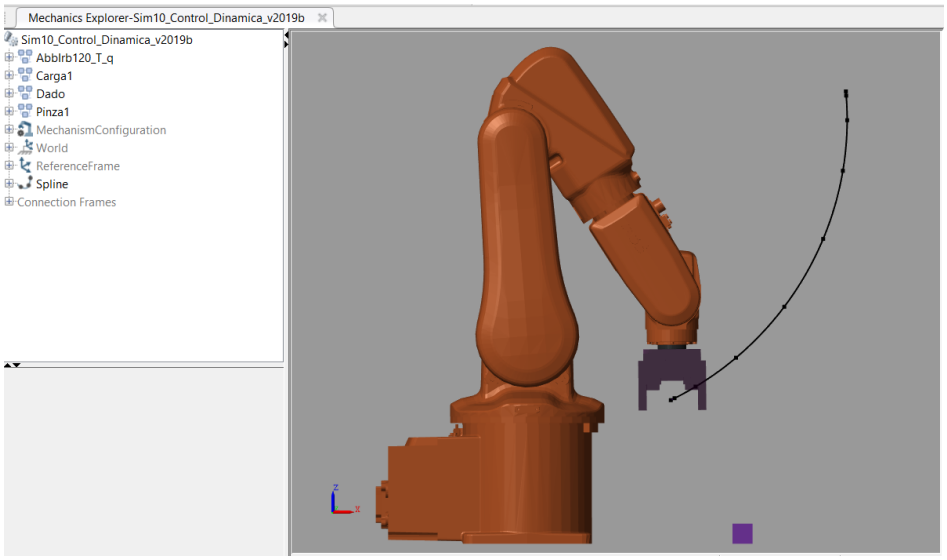
Gráfica 2: ABB irb120 al limitar el torque (Tsat=10)



Gráfica 3: Torque y posición en articulaciones (rotación 30 grados) con Tsat=10

A continuación, se realiza una simulación aplicando la cinemática inversa. Para ello aplicamos el objeto Kin para definir un objeto rob al que mover con las funciones de cinemática inversa.

```
robot= importrobot(Estacion).
robot.showdetails
bodyPinza= 'Body09'.
% Inicio variables posición inicial y rastreador
q0= [0,0,0,0,0,0];
pose_= rand(3,3)*1e-3;


rob= Kin(robot, Estacion, bodyPinza, q0);
```



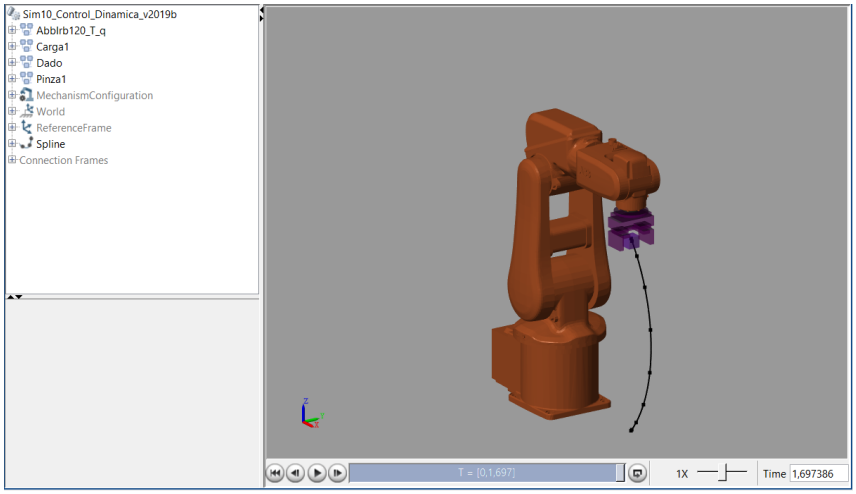
Simulación 5: ABB irb120 movimiento dinámico por cinemática inversa

Así se ha estimado la cinemática inversa con la clase Kin y se han simulado los resultados dando al robot un comportamiento dinámico.

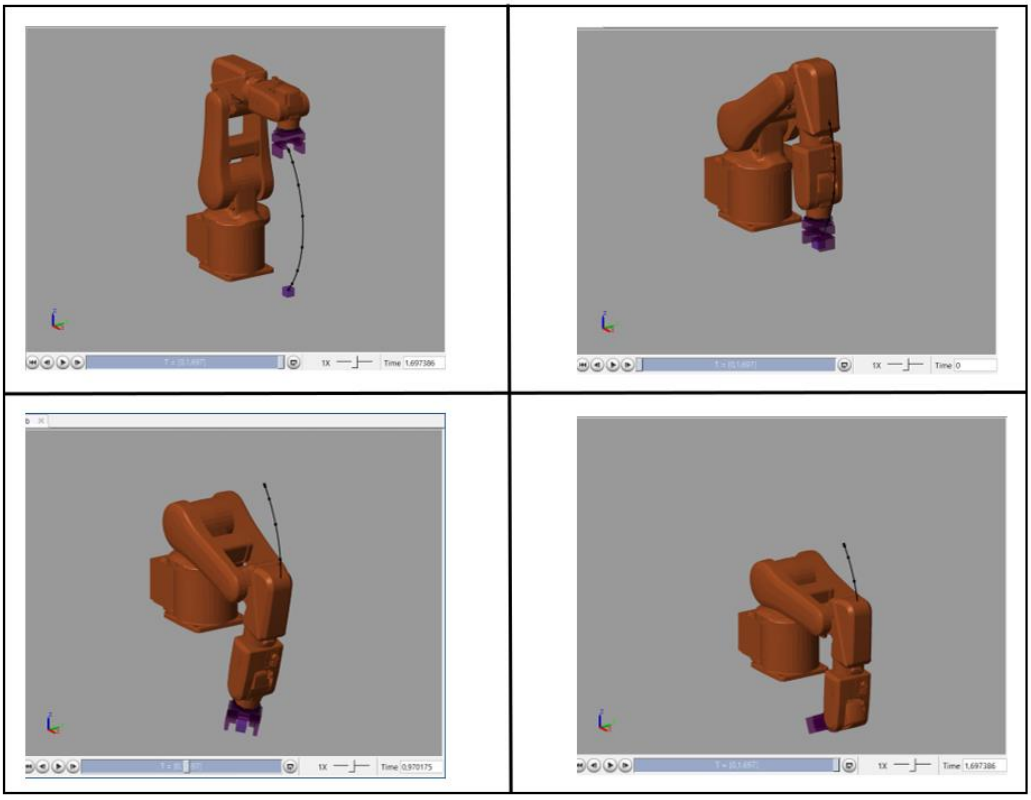
Una de las diferencias que existen entre los modelos dinámicos y los cinemáticos es que se tiene en cuenta la masa y los pesos en la simulación. Al diseñar un modelo robótico real hay que pensar en la fuerza que tienen los motores para sujetar el propio peso del robot y las cargas que deben manipular. En el modelo cinemático esto no se tiene en cuenta, por eso el robot ABB en los ejemplos cinemáticos podía levantar cualquier carga de cualquier peso sin provocar problemas u oscilaciones en la simulación.

Se realiza un ejemplo cambiando el peso del dado que debe coger el robot. Se pueden variar los parámetros masa, cdg e inercia cambiando los valores de la máscara de los iconos de la librería SimRob, que utilizamos en Simulink.

En un principio el dado está unido a la base del sistema global; para tomar y dejarlo se aplicará la función PegarEn que copia las propiedades de un objeto del archivo Simulink en otro objeto. El dado cuenta con una serie de parámetros como la masa, el centro de gravedad o la inercia que pueden provocar cambios en el movimiento del robot si varían. Como el cubo es m uy pequeño en comparación con la pinza, no se realizarán modificaciones del centro de masa ya que las variaciones serán insignificantes. Cuando el valor de masa es de 1kg el robot puede tomar la caja correctamente. Se realizaron varias pruebas hasta que se asignó un valor de masa tan grande que el robot no pudo levantarla y al cogerla con la pinza el brazo del robot se cayó hacia abajo. Cuando se llama a la función PegarEn, la caja deja de estar referenciada al sistema global, por lo que ya no está fija en el suelo. Como la caja pesa tanto que el robot no puede sostenerla, todo el brazo robótico cedió hacia abajo y no puedo levantar la carga.



Simulación 6: ABB irb120 manipula carga (masa=1kg)



Simulación 7: ABB irb120 manipula carga (masa muy elevada)

En resumen, cuando se trabaja con un robot dinámico se simula un modelo robótico real en el que el par que ejercen las articulaciones o la masa de los bodies son características que pueden dar problemas al robot real. Con el método cinemático se calcula el torque de cada articulación de forma matemática sin tener en cuenta las propiedades del robot real como la masa o la inercia.

3.2 FUERZAS EXTERNAS

Según la tercera Ley de Newton "si un objeto A ejerce una fuerza sobre un objeto B, entonces el objeto B debe ejercer una fuerza de igual magnitud en dirección opuesta sobre el objeto A." [15] La tercera Ley de Newton explica por qué una pelota bota cuando choca sobre el suelo, ya que el suelo ejerce una fuerza sobre la pelota que hace que está se eleve, y la pelota hace la misma fuerza sobre el suelo, pero no lo notamos porque el suelo está fijo.

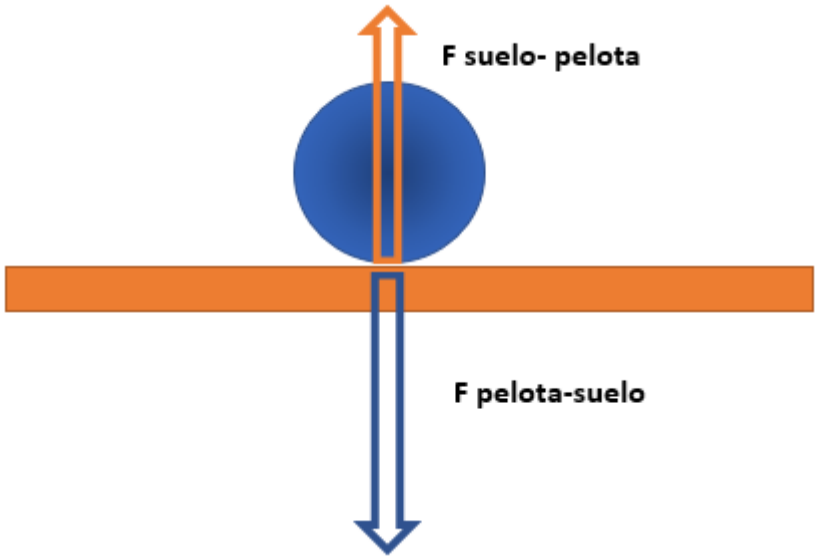
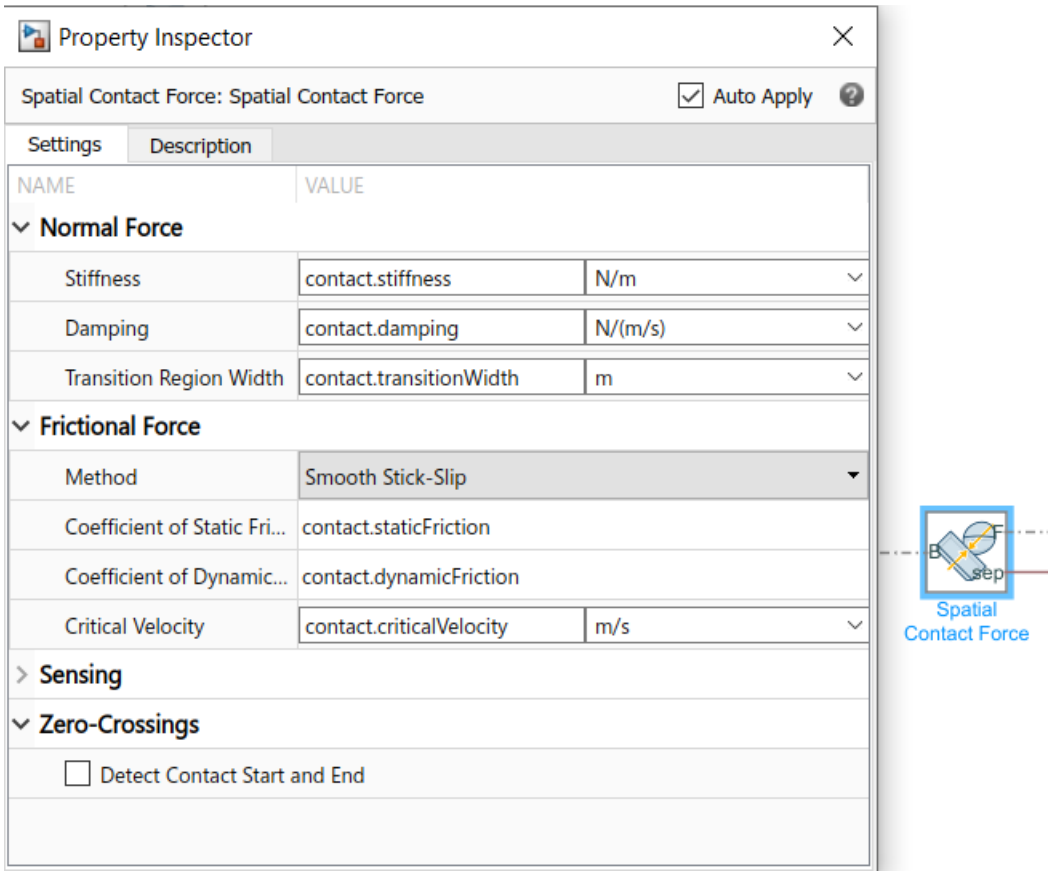


Ilustración 15: Tercera Ley de Newton

Cuando la pelota choca con el suelo, es decir, cuando se trabaja con cuerpos sólidos que entran en contacto, las fuerzas de contacto entre dichos cuerpos juegan un papel importante en el comportamiento de estos bloques sólidos. La respuesta al contacto entre los sólidos dependerá de factores como la rigidez del suelo o las características de la pelota que hacen que la fuerza disminuya a lo largo del tiempo. Para controlar la fuerza de contacto entre sólidos empleamos el bloque de Simscape "Spatial Contact Force".

El bloque Spatial Contact Force aplica una fuerza de contacto entre los dos bloques que conecta el bloque. Las fuerzas tienden a evitar que un sólido penetre dentro del otro sólido porque actúan en una dirección que acelera las geometrías alejándolas entre sí cerca del punto de contacto. Las fuerzas aplicadas son iguales y opuestas y se encuentran a lo largo de una línea de acción común.



Bloques Simulink 23: Spatial Contact Force

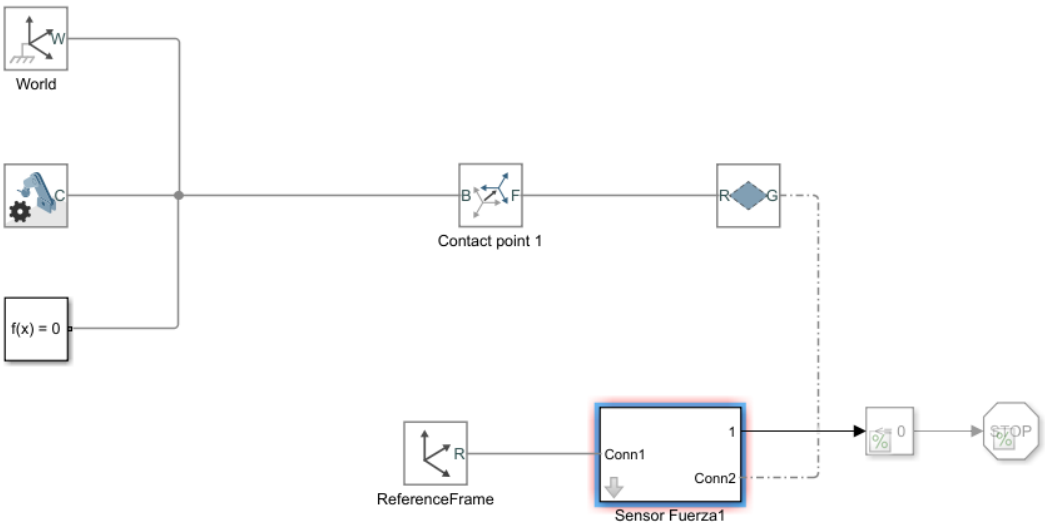
Algunos de los parámetros que podemos modificar dentro de este bloque son:

- Rigidez: resistencia a que la geometría penetre cuando entran en contacto. Cuanto mayor sea la rigidez, el contacto entre sólidos será más "duro".
- Amortiguamiento: representa la pérdida de energía al realizar el contacto entre cuerpos. Cuanto mayor sea el coeficiente de amortiguamiento, más energía se perderá cuando las geometrías colisiones y las vibraciones provocadas por el contacto serán amortiguadas más rápido.
- Coeficiente de fricción estática: Relación entre la magnitud de la fuerza de fricción y la magnitud de la fuerza normal cuando la velocidad tangencial es cercana a cero.
- Coeficiente de fricción dinámica: Relación entre la magnitud de la fuerza de fricción y la magnitud de la fuerza normal cuando la velocidad tangencial es grande.
- Velocidad crítica: Velocidad que determina la mezcla entre los coeficientes estáticos y dinámicos de fricción.

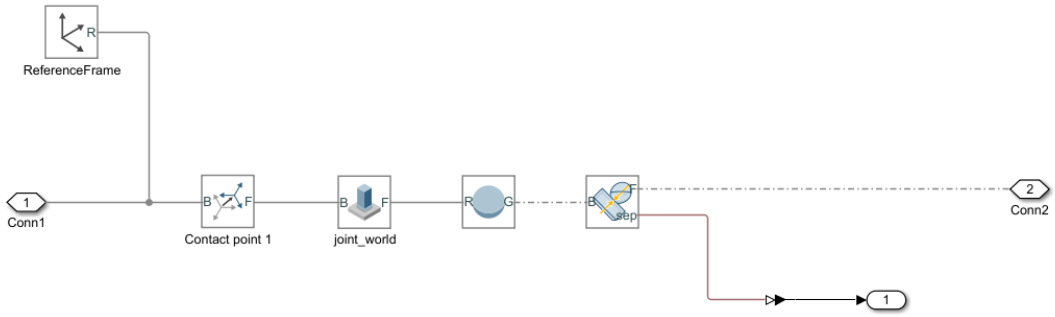
Estos parámetros de contacto entre dos sólidos se podrán modificar al igual que otros parámetros sensoriales como la distancia de separación entre

sólidos o la magnitud de la fuerza normal. Para explicar el funcionamiento de las fuerzas en Simulink, se continuará con el ejemplo de la pelota que bota sobre el suelo que se modelará en Simscape. Como solo tienen lugar fuerzas verticales las variaciones en los coeficientes de fricción y velocidad critica no serán significativas, ya que la pelota no se arrastra por el suelo. Se estudiará los efectos de la rigidez y el amortiguamiento en el contacto entre sólidos.

En el diagrama de Simulink, se ha definido un plano infinito unido de forma fija al sistema global, que actuará como suelo. Este suelo se conecta al bloque “Sensor fuerza” en el que se encuentra el bloque Spatial Contact Force que une las geometrías del plano infinito (suelo) y una esfera (pelota). La pelota no está unida a la base, si no que se dispone en el aire al comenzar la simulación. La posición de la bola al inicio se define mediante una máscara que hay en el bloque “Sensor Fuerza”.

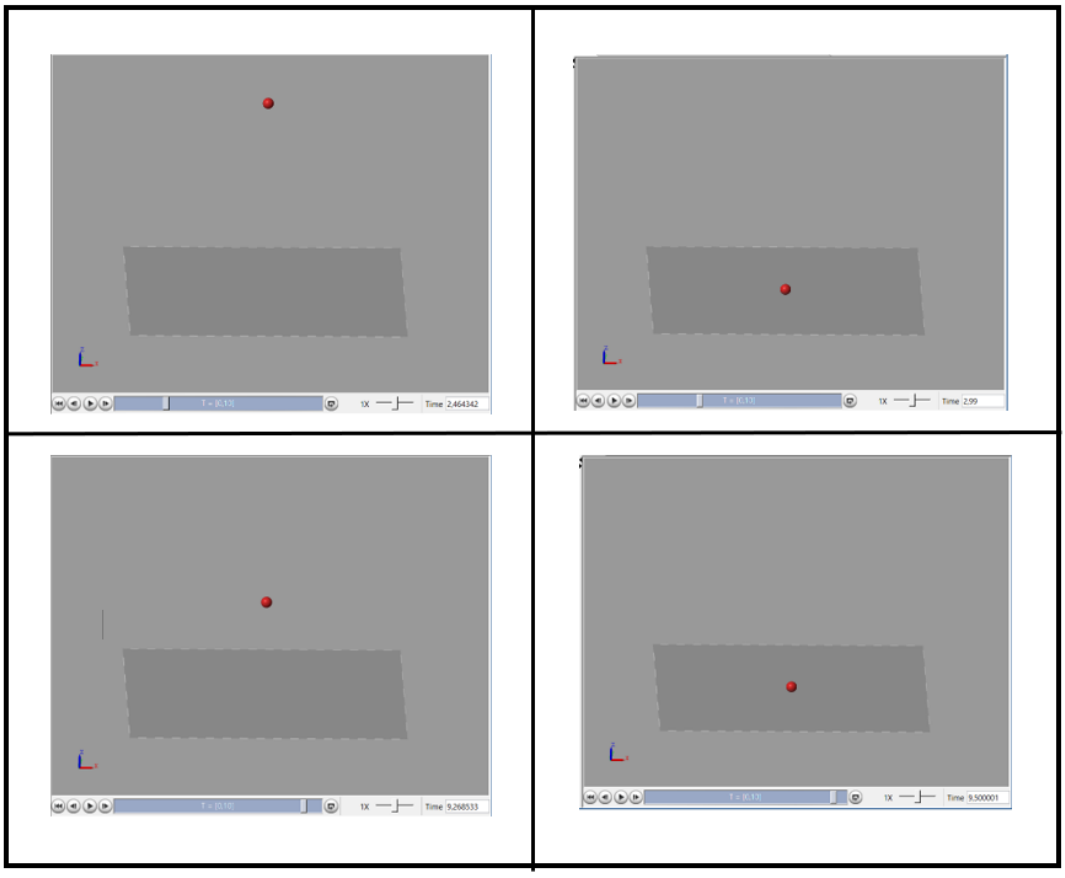


Bloques Simulink 24: Modelo pelota que bota contra el suelo



Bloques Simulink 25: Sensor de fuerza

Durante la simulación la pelota ejerce una fuerza contra el suelo que le hace rebotar. Con los parámetros establecidos en el bloque Spatial Contact Force a medida que avanza la simulación los botes de la pelota van perdiendo fuerza y altura, lo que está relacionado con el coeficiente de amortiguamiento. Por otra parte, se comprueba la rigidez porque la pelota no penetra en el suelo, si no que choca con el mismo.



Simulación 8: Pelota botando

En este ejemplo particular, si el valor de la rigidez fuera muy pequeño, el suelo sería poco rígido, es decir, actuaría como si fuera un suelo de otro material menos sólido y la bola podría penetrar en él o incluso atravesarlo.

Si disminuimos el coeficiente de amortiguamiento, se perderá menos energía en cada bote, por lo que la pelota seguirá rebotando de forma fuerte contra el suelo llegando sin reducir considerablemente la altura del bote a lo largo del tiempo. Con el amortiguamiento también trabajamos con la fuerza de la gravedad, ya que simulamos el efecto que esta aceleración tiene en la pelota.

Este ejemplo se ha escogido porque el suelo y la pelota entran en contacto a través de un único punto de la superficie. En caso de escoger otra figura como un cubo o un prisma, la superficie de contacto entre sólidos sería muy grande y el método del bloque Spatial Contact Force no sería suficiente.

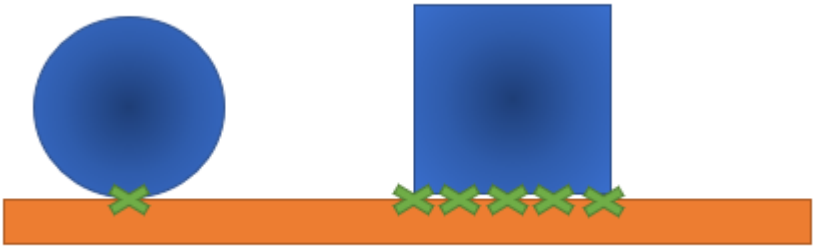


Ilustración 16: Puntos de contacto entre suelo y geometrías simples

Los parámetros de rigidez y amortiguamiento se han escogido haciendo varias simulaciones y comprobando qué resultado era más representativo. También es posible hallar estos parámetros de forma que la relación entre los sólidos que entran en contacto este totalmente controlada. Por ejemplo, hay proyectos desarrollados por Matlab en los que se crear un algoritmo de optimización para que el comportamiento de la bola en Simscape sea lo más parecido posible al comportamiento de una pelota real cuando bota. Aquí se ha modelado un sistema en el que la pelota bota de una forma y en un periodo de tiempo determinado y todo esto se consigue modificando los parámetros de contacto.

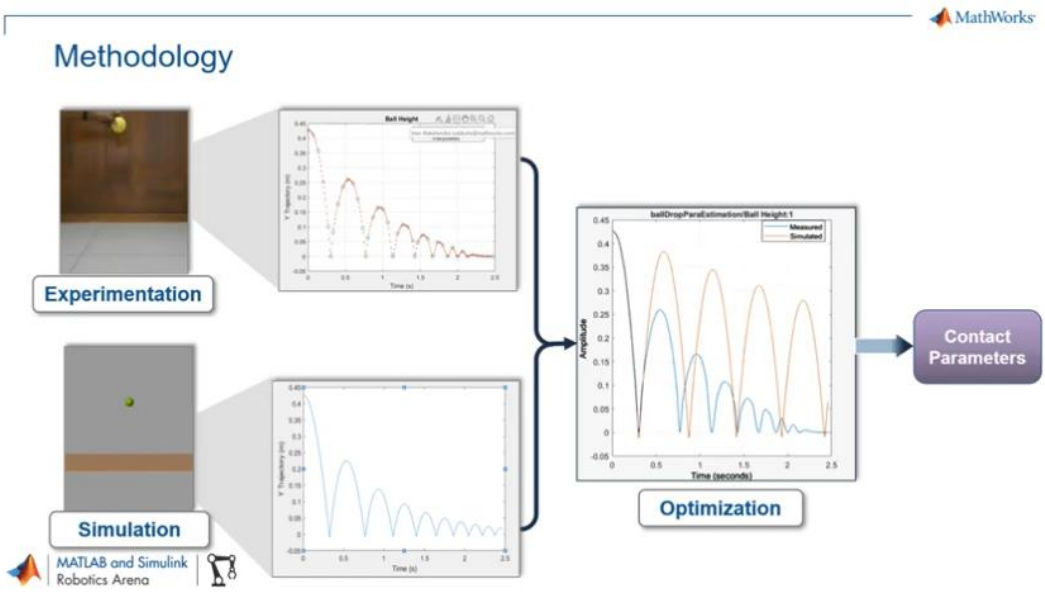
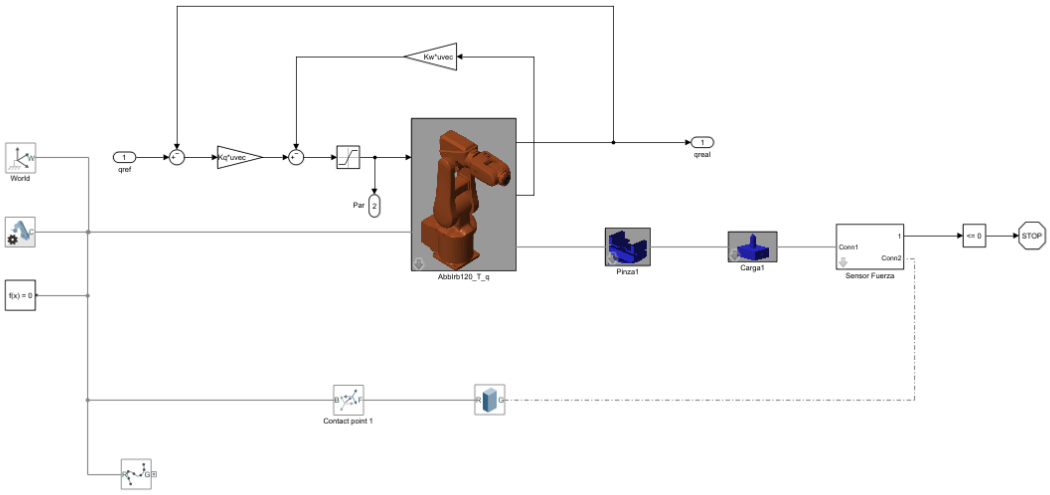


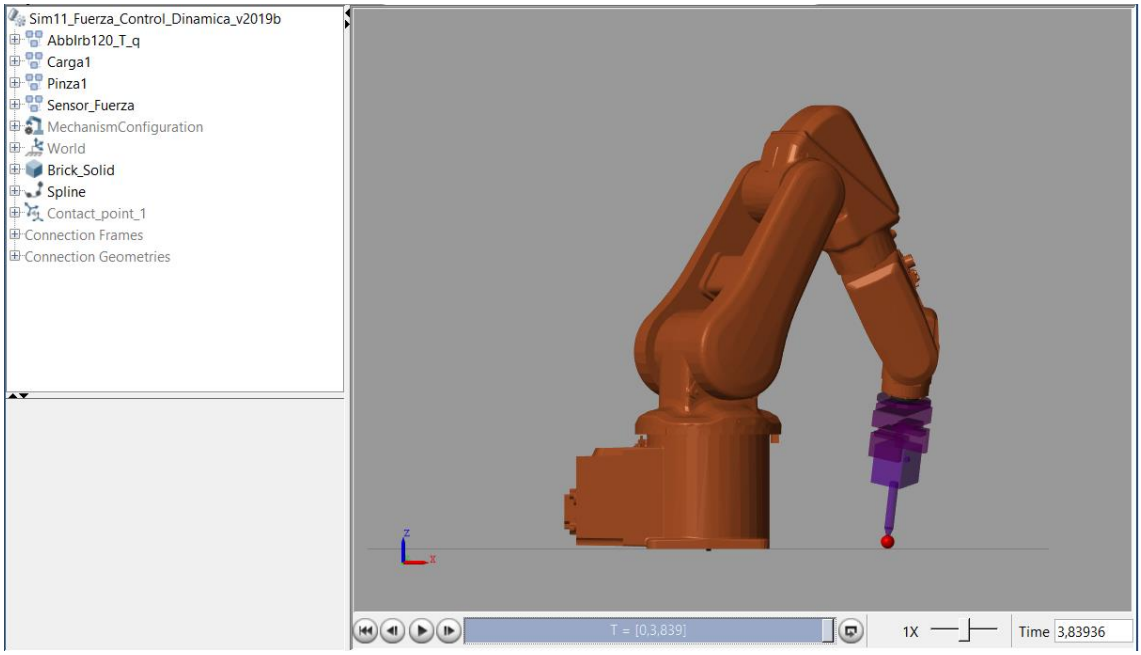
Ilustración 17: Experimentación pelota botando con optmziación [16]

3.3 APLICACIÓN DE FUERZAS

La relación entre fuerzas puede útil para establecer un sensor de fuerza o limitador que indique hasta qué punto debe trabajar el robot. Esto se puede aplicar al brazo robótico, de forma que la esfera que estaba libre en la simulación se una de forma fija al extremo del robot humanoide, en el ejemplo se ha situado al final de la herramienta.

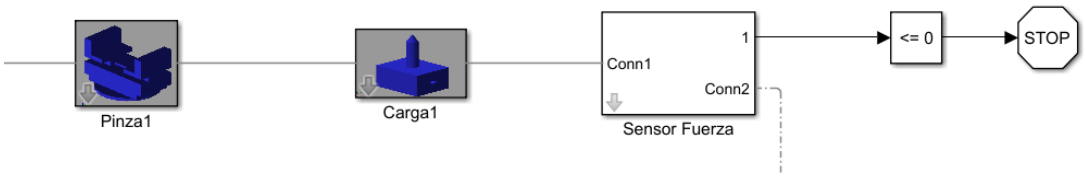


Bloques Simulink 26: ABB irb120 aplicación sensor de fuerza



Simulación 9: ABB irb120 aplicación sensor de fuerza

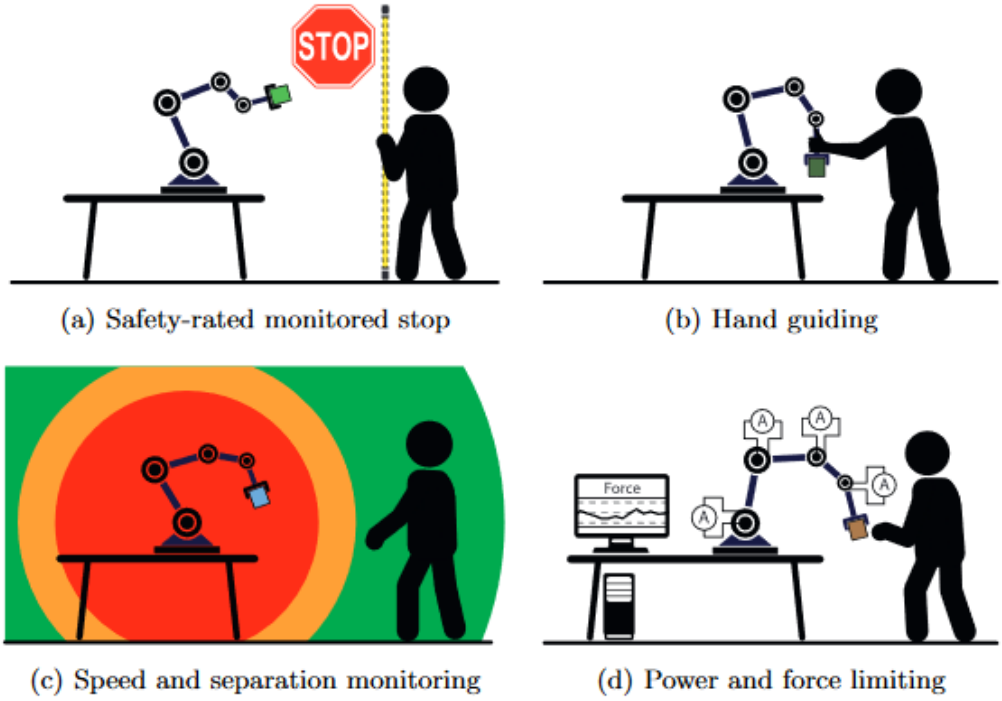
Para que la esfera no atraviese el suelo se podrían variar los parámetros de rigidez y amortiguamiento para cambiar las propiedades del suelo, pero esto daría lugar a oscilaciones y a un tiempo de simulación muy grande una vez que la esfera haya entrado en contacto con el suelo. La solución que se ha encontrado ha sido modificar el bloque "Sensor fuerza" incorporando una condición: si la distancia de separación entre geometrías es menor o igual a cero, entonces la simulación se para automáticamente.



Bloques Simulink 27: Sensor de fuerza con interrupción

El dispositivo sensorial de fuerza desarrollado puede tener un gran número de aplicaciones en campos muy diferentes. Una de las aplicaciones podría ser la incorporación del sensor en robots colaborativos en los que hay interacción directa entre usuario y robot. Cuando el usuario ejerza un esfuerzo sobre el sensor (situado en el extremo del robot) el robot sigue el esfuerzo del paciente y hace el movimiento correspondiente al esfuerzo inverso. El sensor

de fuerza detecta la fuerza del usuario y el robot hace un movimiento para que el esfuerzo resultante sea 0 para frenar al robot. Este sería el ejemplo más sencillo de robots colaborativos, donde el usuario puede guiar al usuario con su fuerza. Un ejemplo de esta aplicación es la técnica de Hand guiding donde se aprovecha la fuerza del robot y los sensores de fuerza para guiar al robot con la mano. Igualmente, otra aplicación es la de un sistema de protección que detecta si la intensidad de un motor pasa un determinado límite. Si esto ocurre hay que dejar de aplicar fuerza en el motor para evitar que se rompa la estructura robótica.



(a) Safety-rated monitored stop (b) Hand guiding

(c) Speed and separation monitoring (d) Power and force limiting

Ilustración 18: Operaciones colaborativas con robots según la norma ISO/TS 15066 [17]

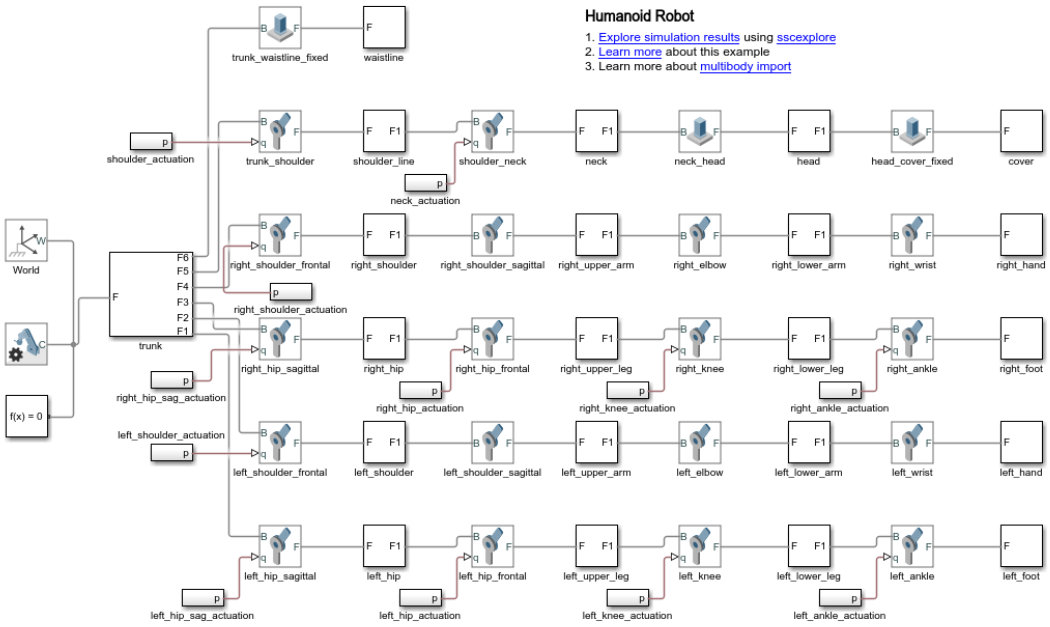
CAPÍTULO IV: HUMANOIDE

Una vez comprendidos los conceptos fundamentales de cinemática y dinámica orientados a modelos robóticos de Matlab y Simulink, se decidió aplicar dichos aspectos a un robot de mayor número de grados de libertad y complejidad. Hasta ahora, se ha trabajado con brazos robóticos que son sistemas robotizados fijos a una base, pero se consideró que los fundamentos de la herramienta SimRob podrían aplicarse a aspectos mucho más complejos como es el control de robots móviles. Para ello se ha ampliado la librería SimRob aplicando los conceptos expuestos a robots humanoides e incorporando nuevos conceptos relacionados con el modelado dinámico de robots.

4.1 MODELO ROBÓTICO DE MATLAB

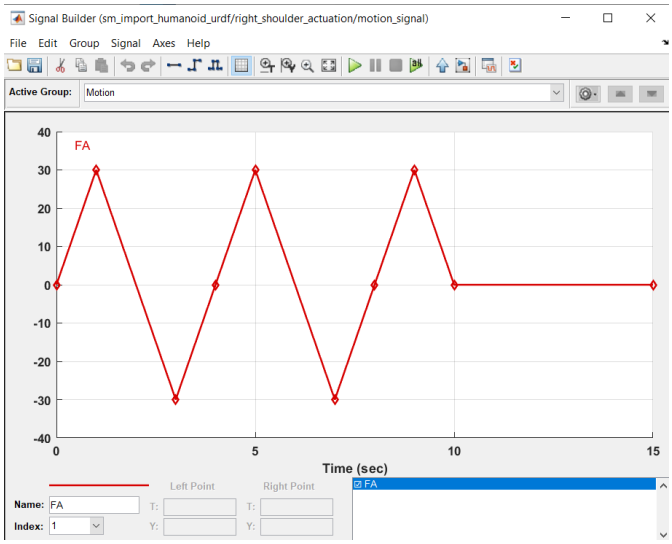
Matlab ofrece un ejemplo “sm_humanoid.urdf “que ha sido importado utilizando el comando smimport. En dicho modelo hay ciertos bloques de actuación articular, p, como entrada a las articulaciones para que le robot pudiera simular ciertos movimientos.

Al importar el modelo URDF del humanoide directamente desde Matlab obtenemos un archivo Simulink con varios bloques para definir las articulaciones y las entradas de las mismas.

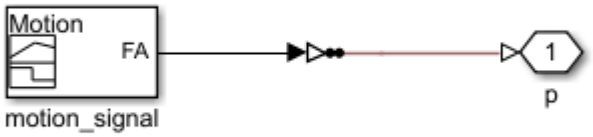


Bloques Simulink 28: Robot Humanoide propuesto por Matlab

Cuando comienza la simulación de Matlab el robot mueve automáticamente las piernas y los brazos, las articulaciones de los hombros, la cadera y las rodillas. En el esquema se observan varios iconos que representan cada articulación del humanoide y algunos bloques con la letra “p” (posición angular) que sirven de entrada a la articulación. Dentro de cada bloque de entrada “p” encontramos un bloque Motion que crea diferentes grupos de señales de forma lineal por partes. Estas señales son las que definen el movimiento de cada articulación porque indican la posición en grados que tiene que rotar cada articulación rotacional. Por ejemplo, para balancear el hombro con un ángulo de 30 grados se dispone esta señal.



Bloques Simulink 29: Señal de entrada propuesta por MatLab



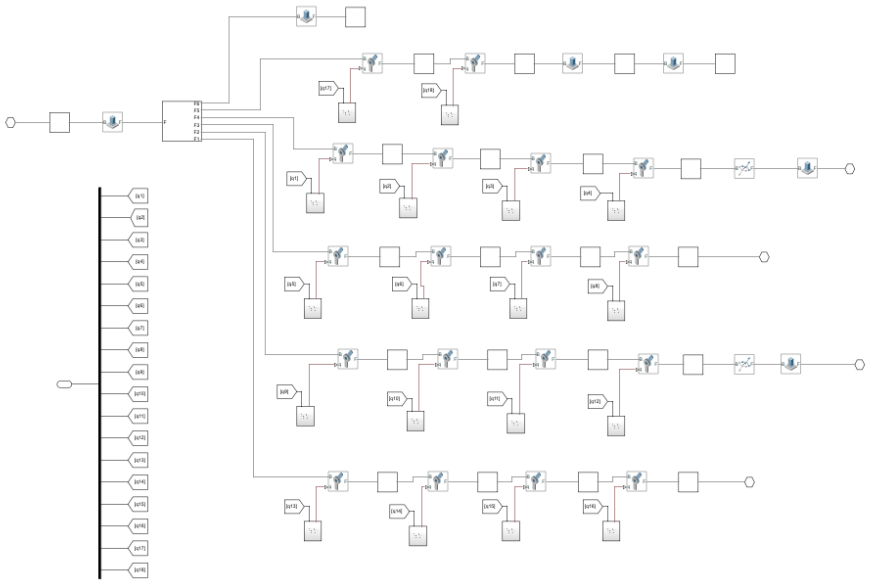
A pesar de que este método de movimiento articular es válido, el usuario debe modificar los datos de manera gráfica (cambiando la figura de la gráfica) y tan solo se podrá trabajar con las articulaciones que tienen como entrada un bloque p. Además, para cambiar el grado de rotación de las articulaciones se debe hacer de forma individual, a diferencia de las estaciones modeladas en los ejemplos con el robot ABB irb120, donde las coordenadas articulares se almacenaban en una única variable q. Igualmente, con el modelo que ofrece Matlab el tronco del robot permanece unido al sistema de referencia base, por lo que el robot no podrá desplazarse durante la simulación y será, al fin y al cabo, un humanoide fijo igual que el brazo robótico.

Este modelo tiene un gran potencial y aplicando los conceptos de dinámica y cinemática explicados anteriormente podremos lograr movimientos mucho más complejos e incluso similares a los de una persona real. Para conseguirlo se emplearán las funciones que proporciona Matlab para el control de robots y se realizarán mejoras para optimizar los movimientos del humanoide.

Se trata de un robot humanoide con 22 bodies que constituyen la cabeza, tronco y extremidades del robot. El robot presenta 4 grados de libertad en cada brazo, 4 grados de libertad en cada pierna y dos en la cabeza. En la

estructura encontramos 18 articulaciones rotacionales (4 para cada brazo, 4 para cada pierna, una para la articulación tronco-hombro y otra para hombro-cuello). Igualmente cuenta con 4 articulaciones fijas que se reparten entre la cabeza y la cintura. Según la configuración de Matlab, para representar de forma visual el robot construido utilizamos un archivo stl por cada body empleado.

Como se ha mencionado anteriormente, el formato de entrada de las posiciones articulares no es cómodo ni práctico para el usuario, por lo que en cada bloque de articulación se pasará la variable q de la misma forma que en las estaciones explicadas en el apartado "2.4 Librería SimRob"



Bloques Simulink 30: Humanoide base

En el modelo de Simulink aprovecharemos la estructura y la configuración del robot (se mantienen los 22 bodies con sus articulaciones), pero se harán modificaciones a la hora de introducir las posiciones y otros parámetros para controlar el robot.

4.2 MODELO CINEMÁTICO DEL HUMANOIDE

En el robot planteado por Matlab el modelo está enfocado a un estudio mayoritariamente cinemático, donde la entrada al robot es la posición angular y a partir de esta el robot calcula de forma matemática cómo resolver el planteamiento. Aquí se plantea un sistema de lazo abierto donde la única entrada es el ángulo que se moverá una articulación. Con el modelo cinemático, el sistema calcula automáticamente el torque (Torque

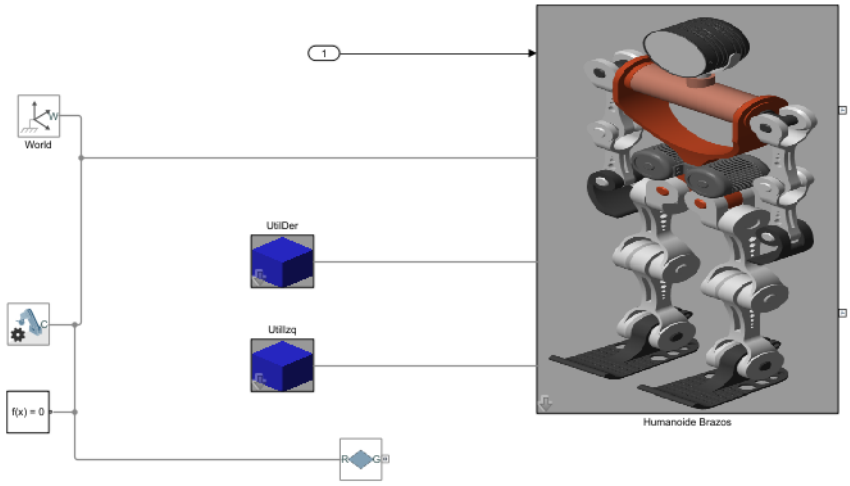
Automatically Computed en cada joint) movimiento resolviendo la siguiente ecuación:

$$T = J\frac{d^2q}{dt^2} + B\frac{dq}{dt} + Kq \qquad (13)$$

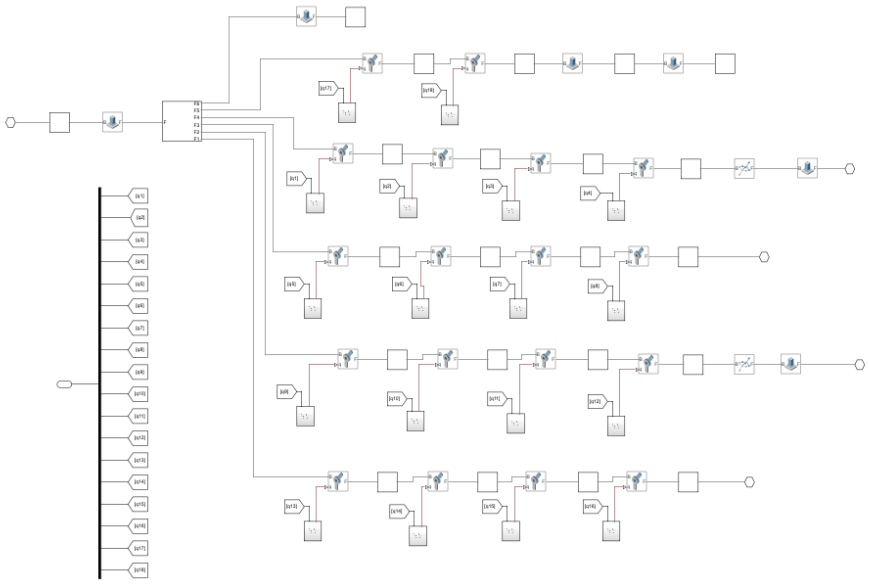
Para comprender mejor el modelo cinemático se llevaron a cabo una serie de ejemplos para lograr su control.

4.2.1 DISEÑO DE LA ESTACIÓN

Como en un estudio dinámico no se analizan las fuerzas que producen el movimiento del objeto, se decidió mantener fijo al robot por la cadera. Con esto conoceremos la posición exacta de la cadera, por lo que a la hora de plantear el modelo cinemático inverso del robot sabremos la posición de cada articulación. Al plantear algunas simulaciones, se ha partido del modelo Simulink base para el humanoide y se han realizado algunos cambios en el modelo de (insertar o quitar objetos, herramientas, introducir nuevos TCPs…).



Bloques Simulink 31: Modelo robótico Humanoide modelo cinemático



Bloques Simulink 32: Humanoide modelo cinemático

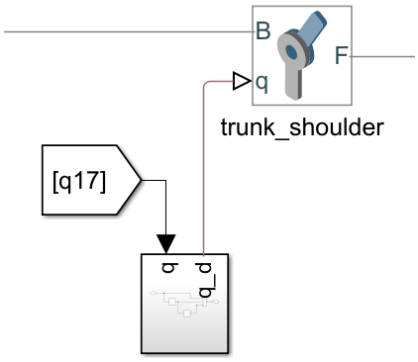
En el diseño de las articulaciones del robot, comprobamos que presenta dos salidas para las manos del robot y dos para los pies.

Ilustración 19: Modelado mano derecha e izquierda

Ilustración 20: Modelado pie derecho e izquierdo

En el caso de las manos presentan una articulación de cero grados de libertad que ejerce de TCP. Esto se ha dispuesto así ya que las manos del humanoide actúan como TCP del robot para coger objetos y manipular herramientas. Al salir del subsistema del humanoide se unen con dos bloques UtilIzq y UtilDer.

Como se va a realizar un análisis cinemático del humanoide, la entrada a este es la posición articular y a partir de esta Matlab calculará el torque automáticamente.



Bloques Simulink 33: Bloque articulación modelo cinemático

El robot humanoide cuenta con un torso fijo a la base y 4 extremidades que parten de él móviles, por lo que se puede interpretar como si fuera un robot compuesto de 4 brazos robóticos unidos a una misma base. Cada extremidad se mueve de forma independiente a las demás como si se tratara de un robot único.

Se aplicaron diferentes aspectos de la librería SimRob para conseguir la resolución del problema cinemático. En primera instancia se probaron algunos ejemplos con la clase Kin para mover el robot empleando la cinemática directa e inversa. A continuación, se plantea un ejemplo en el que se mueve el brazo izquierdo y la pierna derecha con la clase Kin.

```
q0= zeros(1,18);
rob= Kin(robot, Estacion, PieD, q0);
deg= pi/180;
q= zeros(1,18);

%CINEMÁTICA DIRECTA
q(9)= -80*deg;
rob.MoveAbsJ(q);

%CINEMÁTICA INVERSA
pose= rob.Pose;
pose(1:3)= pose(1:3)*1.2;
rob.MoveJ(pose) % Mover a posición modificada
```

Código 1: Movimiento cinemática directa e inversa del robot

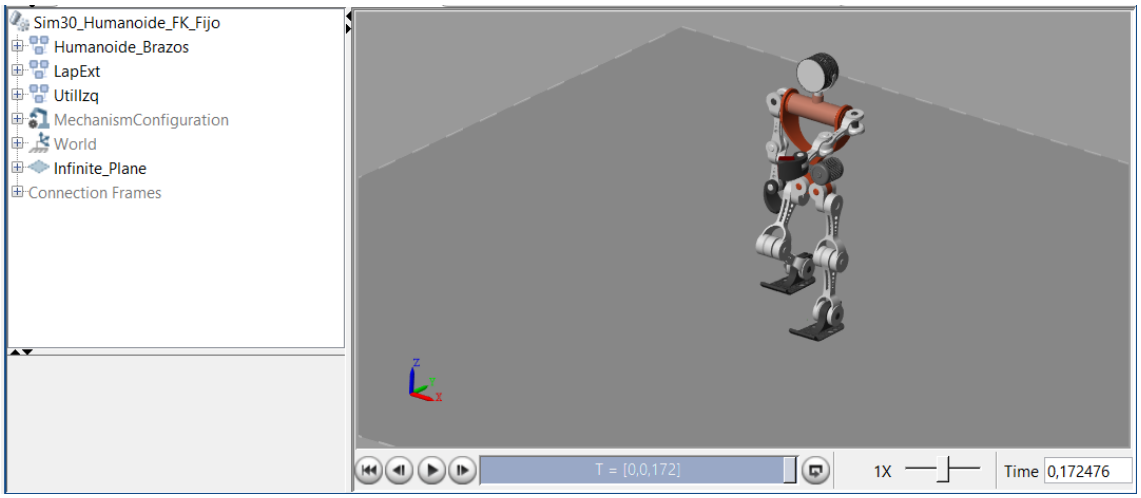
En el código se define el objeto Kin pasándole como herramienta PieD.

Con el movimiento MoveAbsJ() se mueve la articulación 9 (en nuestro caso el hombro) por cinemática directa a una posición de -80 grados respecto a la

configuración inicial. Como es cinemática directa, se pasa como argumento las coordenadas articulares deseadas, por lo que no influye la herramienta que se había definido en el objeto Kin.

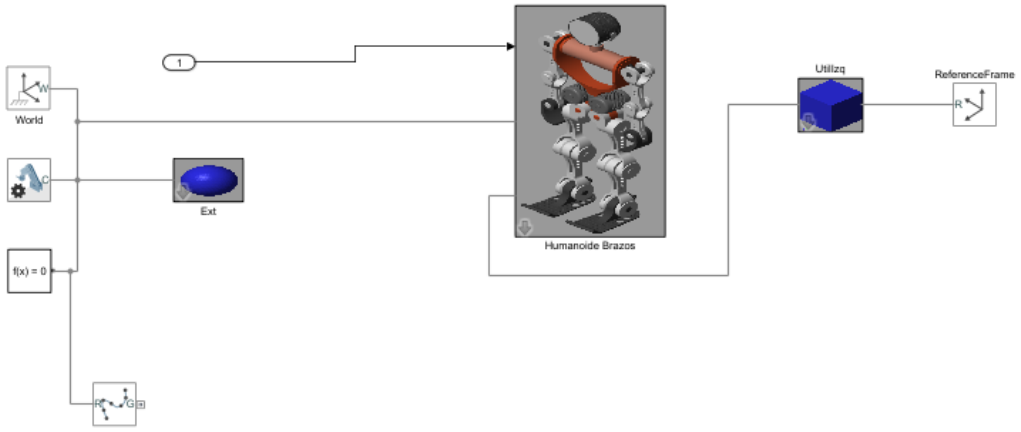
Por otro lado, el método MoveJ() logra el movimiento del pie derecho mediante cinemática inversa. Lo que se ha hecho es modificar los valores de las posiciones del 1 al 3 del vector pose del objeto definido, es decir, las posiciones en las que se encuentran las coordenadas x y z del pose, multiplicándolos por una constante cualquiera. El pose modificado se pasa a MoveJ() como argumento para que el body definido en el objeto qin varíe su posición.

Las extremidades se mueven de forma independiente, de forma que en una única simulación el robot pueda mover articulaciones diferentes empleando el modelo directo y el inverso utilizando las funciones de Kin.



Simulación 10: Humanoide mueve extremidades por cinemática

Además, podemos combinar los métodos de la clase Kin para mover las articulaciones con el método PegarEn de la clase Pieza para cambiar cargas de una posición a otra. Para poder dejar y coger la pieza es necesario modificar la estación inicial y crear una pieza asociada a la base para que el robot pueda abandonar la carga de su mano.



Bloques Simulink 34: Modelo robótico Humanoide para manipular cargas

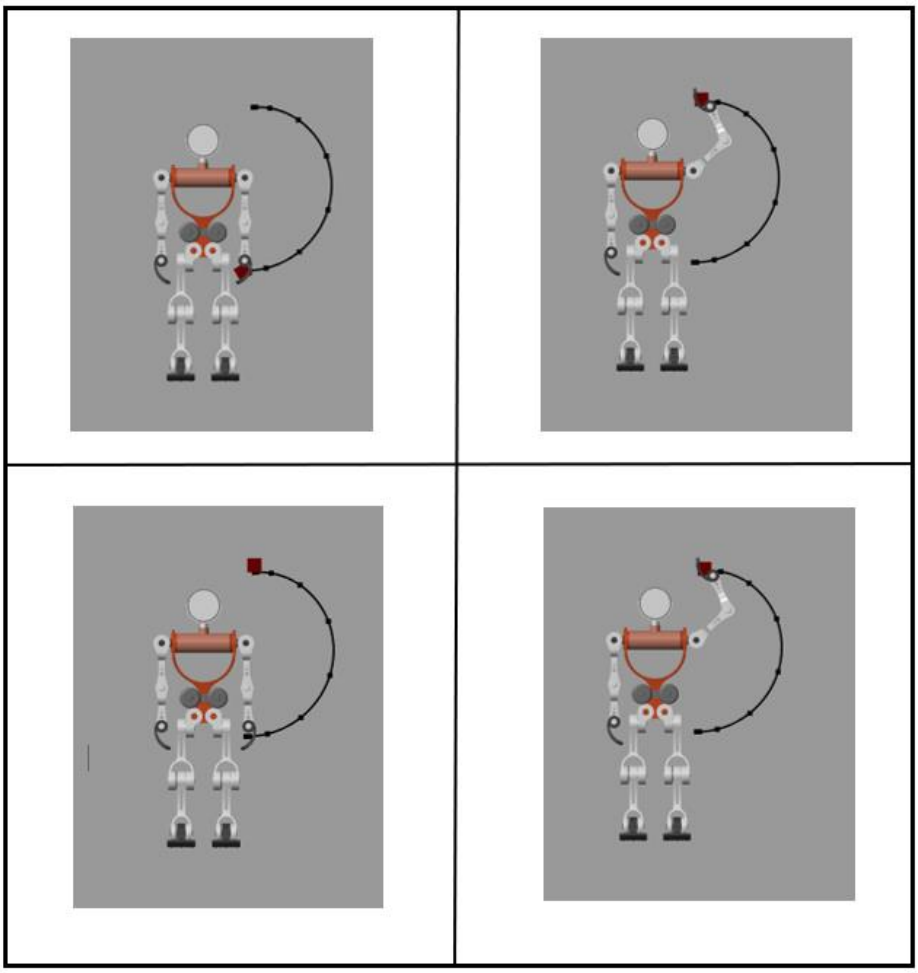
En el siguiente ejemplo se mueve el brazo izquierdo del robot para dejar la caja arriba, el brazo retorna a la posición inicial sin la caja, el brazo vuelve a tomar el objeto y por último regresa a su posición inicial para acabar la simulación.

```
clear all;
deg= pi/180;

Estacion= 'Sim30_Humanoide_FK_Fijo_mod'
q0_= zeros(1,8)*deg;
pose_= rand(3,3)*1e-3;
baseRobot= [[0,0,0]*1e-3,[0,0,0]*deg];

Pieza('UtilIzq', 'Dado');
UtilIzq.Tcp([[0,0,10]*1e-3, [0,0,0]*deg]);
UtilIzq.Color([0.5,0,0,1]);

Pieza('LapExt', 'Null');

robot=importrobot(Estacion);
q1=zeros(1,18);
robmd=Kin(robot, Estacion, 'Body10', q1);

aux= Hmat; aux.Rad;
%MOVIMIENTO BRAZO HACIA ARRIBA Y DEJA PIEZA
puntomd= [[100,0,100]*1e-3,[0,0,0]*deg];
robmd.MoveJ(puntomd);
PegarEn(LapExt, UtilIzq, [], aux.tRzyx(robmd.Pose()));

%VUELVE A POSICIÓN INICIAL
robmd.MoveAbsJ(q1);

%MOVIMIENTO BRAZO HACIA ARRIBA Y COGE PIEZA
robmd.MoveJ(puntomd);
PegarEn(UtilIzq, LapExt, aux.tRzyx(robmd.Pose), []);

robmd.MoveAbsJ(q1);
```

Código 2: Movimiento tomar y dejar caja robot Humanoide

Con la función *PegarEn()* se deja el icono de la carga (Utilzq) en una posición asociada a la base del sistema (Ext). Gracias a ello se mueve el robot, pero la caja permanece en la última posición. Para volver a tomar la carga se intercambia la pieza del icono exterior a la carga del robot. De esta forma el movimiento obtenido a partir del código anterior sería el siguiente:

- La mano izquierda del robot (Body10) se mueve por cinemática inversa por el camino más simple hacia el punto "puntomd", que se pasa a MoveJ().
- Deja la pieza en la última posición mediante la función *PegarEn().* Así intercambio las propiedades de la carga con el icono del exterior. Como Ext tiene como referencia la referencia global, entonces se pasa como matriz homogénea de la referencia a Ext el signo []. La matriz homogénea de la referencia a UtilIzq se obtiene a partir del pose actual del objeto Kin.
- La mano vuelve la posición inicial por cinemática directa con la función MoveAbsJ(). Se introducen las coordenadas articulares iniciales (vector q1=zeros(1,18)).
- La mano vuelve al "puntomd" igual que antes
- Toma la pieza que se había quedado arriba con la función PegarEn(). En este caso copiamos las propiedades del objeto Ext en UtilIzq. Como consecuencia los argumentos de la función PegarEn() cambian de posición (las propiedades entre sí y las matrices entre sí).
- El robot mueve el brazo a la posición inicial de nuevo.



Simulación 11: Humanoide Acción dejar y coger caja con función PegarEn()

Otro método con el que podemos mover las articulaciones de forma cinemática es directamente con el comando sim() de Matlab. Con este se puede simular un modelo específico utilizando la configuración existente en el modelo que le pasemos como argumento.

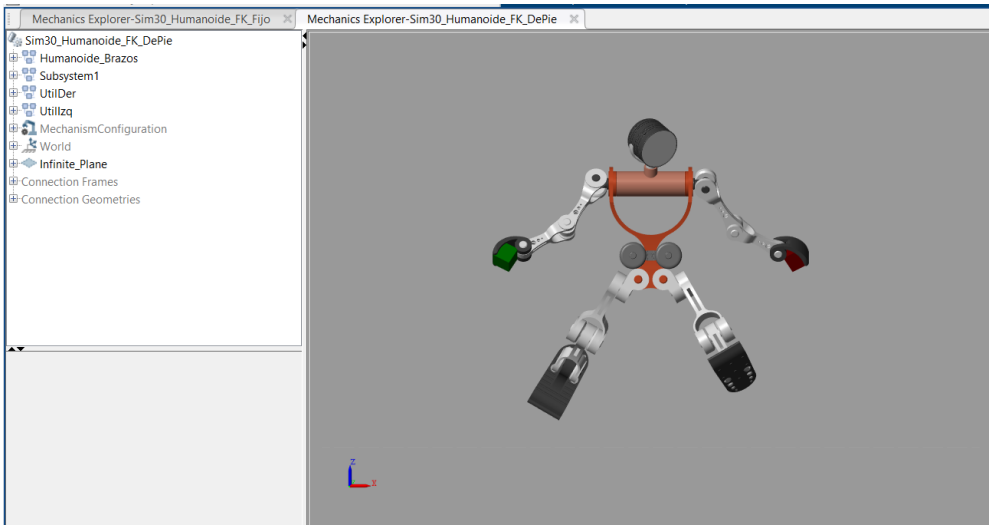
Si se desea simular un robot, será necesario pasara como argumentos el modelo de Simulink que se va a simular, el vector de tiempos de la simulación y un vector indicando los grados que se quiere mover la articulación deseada.

En el siguiente ejemplo movemos todas las articulaciones, de la 1 a la 18, 30 grados para ver el comportamiento del robot humanoide.

```
deg= pi/180;

% Variables internas de Kin
baseRobot= [[0,0,440]*1e-3,[0,0,0]*deg];
q0_= zeros(1,18);


Estacion= 'Sim30_Humanoide_FK_DePie';
%Estacion= 'Sim30_Humanoide_Libre';

Pieza('UtilDer', 'Dado');
UtilDer.Base([[0, 0, 0]*1e-3, [0,0,0]*deg])
UtilDer.Color([0,0.5,0,1]);

Pieza('UtilIzq', 'Dado');
UtilIzq.Base([[0, 0, 0]*1e-3, [0,0,0]*deg])
UtilIzq.Color([0.5,0,0,1]);

nPts= 1e4;
t= linspace(0, 25, nPts)';
q= linspace(0, 30*deg, nPts)';
Q= repmat(q,[1,18]);
open_system(Estacion)
sim(Estacion,t,[],[t,Q]);
```

Código 3: Movimiento articulaciones 30 grados en Humanoide

Al simularlo, se observa como las articulaciones realizan el movimiento definido por el usuario. También se aprecia que como el robot está sujeto de forma fija por la cadera, esta parte del robot no cambia de posición. No solo eso, si no que debido al movimiento de las dos extremidades inferiores el robot queda flotando en el aire al enviar esta instrucción.



Simulación 12: Humanoide mueve todas las articulaciones 30 grados

Con este modelo podemos controlar los movimientos del robot de forma más o menos sencilla sin embargo no se tiene en cuenta masas ni otros parámetros relacionados con las fuerzas. Además, no se tiene en cuenta la estabilidad ni el equilibrio del robot, el humanoide nunca se caerá porque está fijo a la base a través de la cadera. No tiene sentido aplicar las funciones cinemáticas de Kin porque los movimientos de los humanos no son iguales que los de un robot; a esto se añade que con Kin solamente se indica al robot que llegue a una posición, pero no se puede dar ninguna orden para que realice un movimiento. Solo es capaz de definir la forma de llegar a un punto concreto, pero no sirve para hacer movimientos en los que se necesitan varias posiciones articulares de una misma articulación a lo largo de la simulación.

Este modelo da pie a situaciones irreales que no podrían darse en los movimientos de una persona, por ello decidimos ir un paso más allá y diseñar un modelo dinámico en el que se tenga en cuenta la fuerza de la gravedad, las fuerzas normales y la estabilidad del robot entre otros aspectos. El movimiento de un ser humano solo puede definirse mediante un modelo dinámico.

4.3 MODELO DINÁMICO DEL HUMANOIDE

4.3.1 DISEÑO DE LA ESTACIÓN

El estudio dinámico se centra en las causas que dan lugar al movimiento de un cuerpo. A diferencia de la cinemática, donde se pasa como entrada una

coordenada articular y el robot calcula matemáticamente el torque que hay que ejercer, en el modelo dinámico se pasa al robot directamente el torque deseado.

Como se explicó en el ejemplo con el brazo robótico, la diferencia entre el modelo cinemático y dinámico reside en el bloque de las articulaciones. En este ejemplo, la articulación del codo tiene como entrada la posición en el caso cinemático y no presenta ninguna salida. Por otro lado, el bloque articular en el modelo dinámico tiene como entrada el torque y de salida la posición articular que es vital para controlar la posición del humanoide.

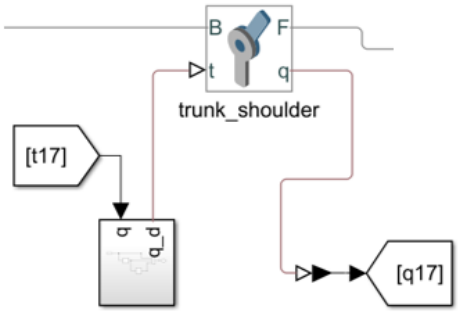


Ilustración 21: Bloque articulación en modelo dinámico

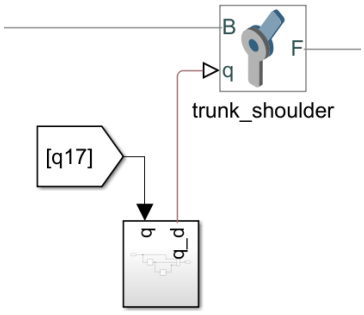
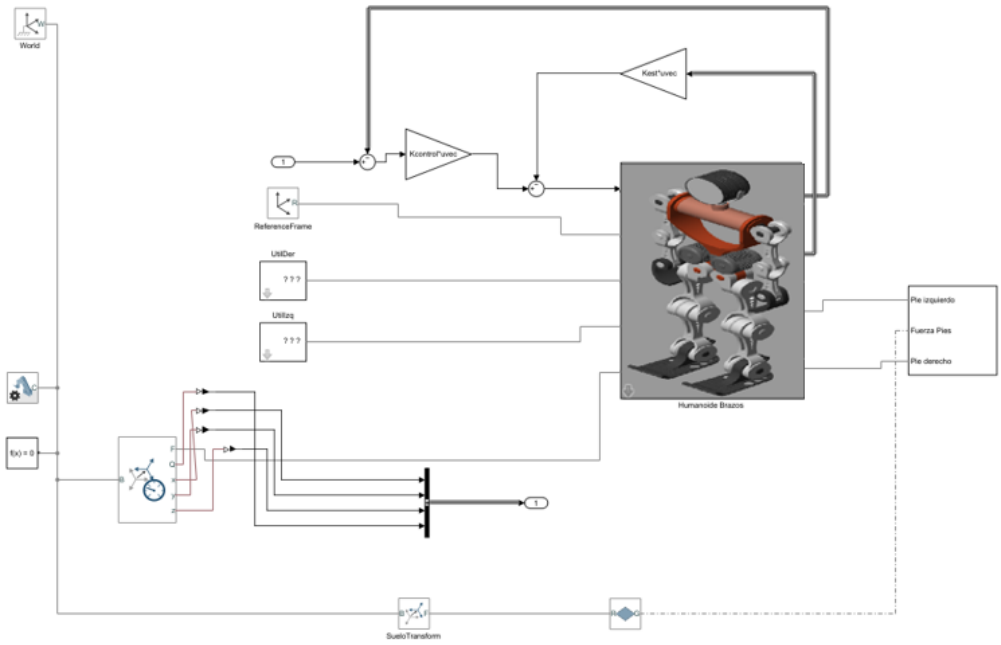


Ilustración 22: Bloque articulación en modelo cinemático

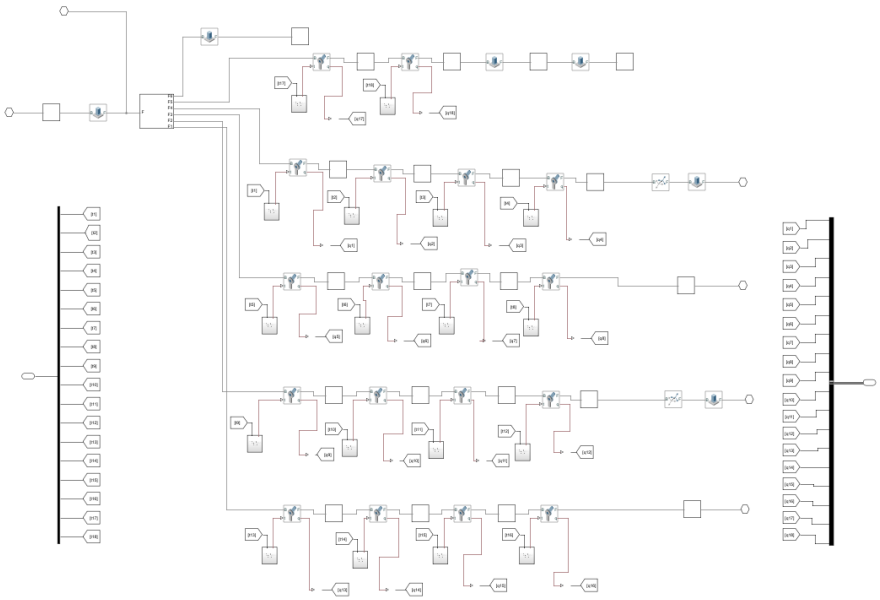
Más adelante se modificará el bloque de nuevo para incorporar una nueva salida; esto se explicará en el apartado de control.

Los modelos dinámicos del humanoide tendrán como entrada al sistema la posición de todas las articulaciones que determinará el usuario. La salida será la posición y rotación en cuaternios de la base del robot (hemos considerado que la base será la cadera). Partiendo de la estructura que presentaban las estaciones de modelado cinemático diseñamos una nueva

estación para trabajar con el humanoide de forma dinámica. La estación cuenta con el mismo número de articulaciones y TCP para manipular cargas. Como estamos modelando el humanoide de forma dinámica, hay una realimentación de la posición para conocer si la posición articular real del robot está cerca o no de la deseada. Si esta posición aún es muy diferente se continuará alimentando las articulaciones con un torque hasta que el error sea cero, como se explicó en el subcapítulo "3.1 Modelado dinámico robot".

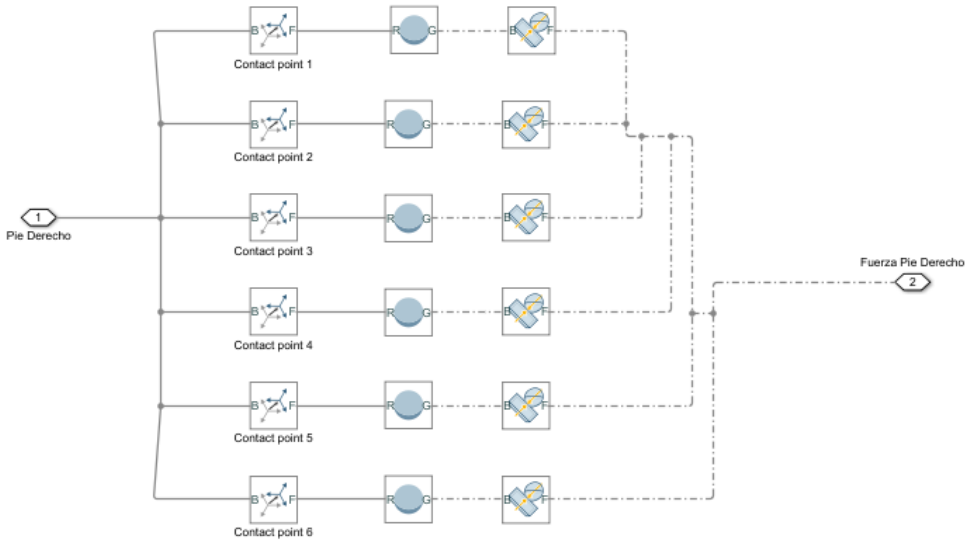


Bloques Simulink 35: Modelo robótico para modelado dinámico



Bloques Simulink 36: Robot para modelo dinámico

Por otra parte, los pies salen del subsistema del humanoide y se unen con otro subsistema que presenta como salida la fuerza que ejercen los pies.



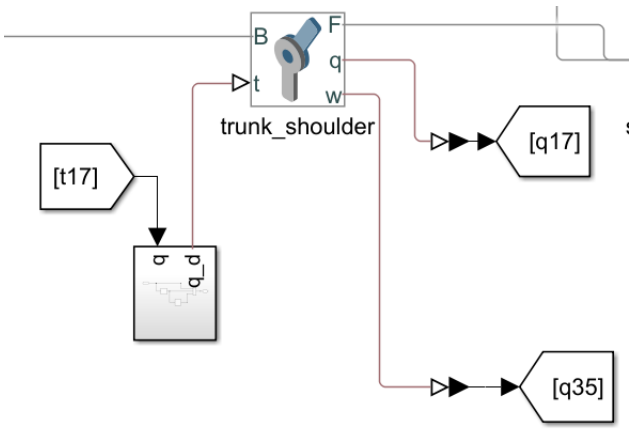
Bloques Simulink 37: Subsistema Fuerza Pies Humanoide

Para modelar el robot de forma dinámica es necesario que este ejerza una fuerza contra el suelo para sujetar el cuerpo del humanoide utilizando

solamente los pies. Para ello se deben definir las esferas Contact Proxis que se muestran en la ilustración. Cada esfera está unida por un lado a un bloque Rigid Transform para cambiar la posición de las esferas y por otro lado a un bloque Spatial Contact Force. A través de este último bloque se establece el contacto entre cada esfera y el suelo; en el siguiente apartado se explicará de forma más detallada la importancia de estas esferas de contacto en el control del robot dinámico. El modelado de la fuerza que ejercen las esferas contra el suelo se detalla a continuación en "4.3.3 Contacto pies-suelo"

4.3.2 LAZO INTERNO DE CONTROL

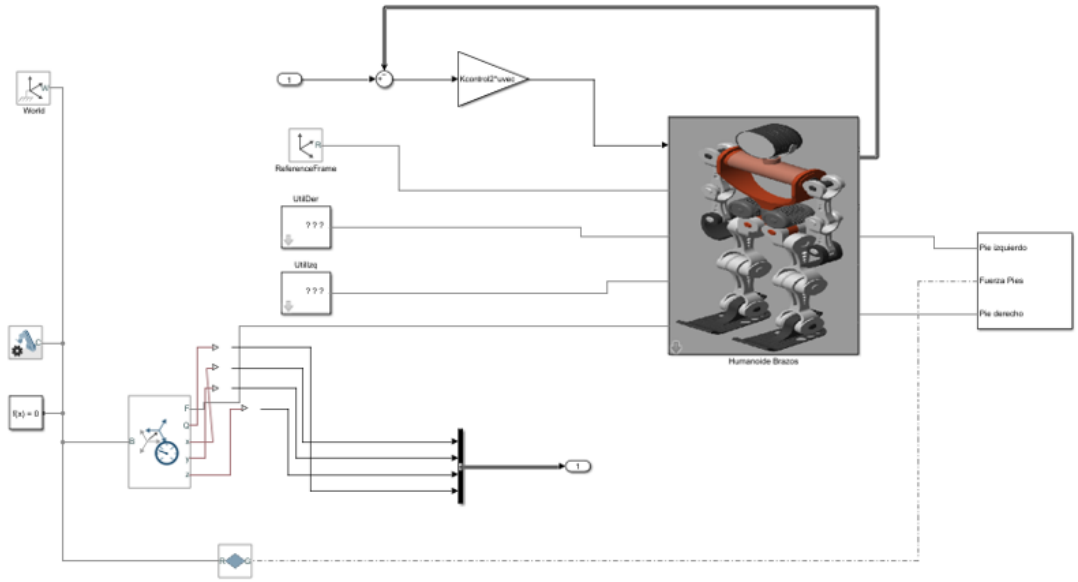
En apartados anteriores se ha mencionado que para lograr el modelado dinámico del robot ha sido necesario incorporar una realimentación con un sistema de control. Este sistema básico de control presenta una acción proporcional y una acción derivativa, que conseguiremos con las salidas de la posición y la velocidad de los bloques de las articulaciones.



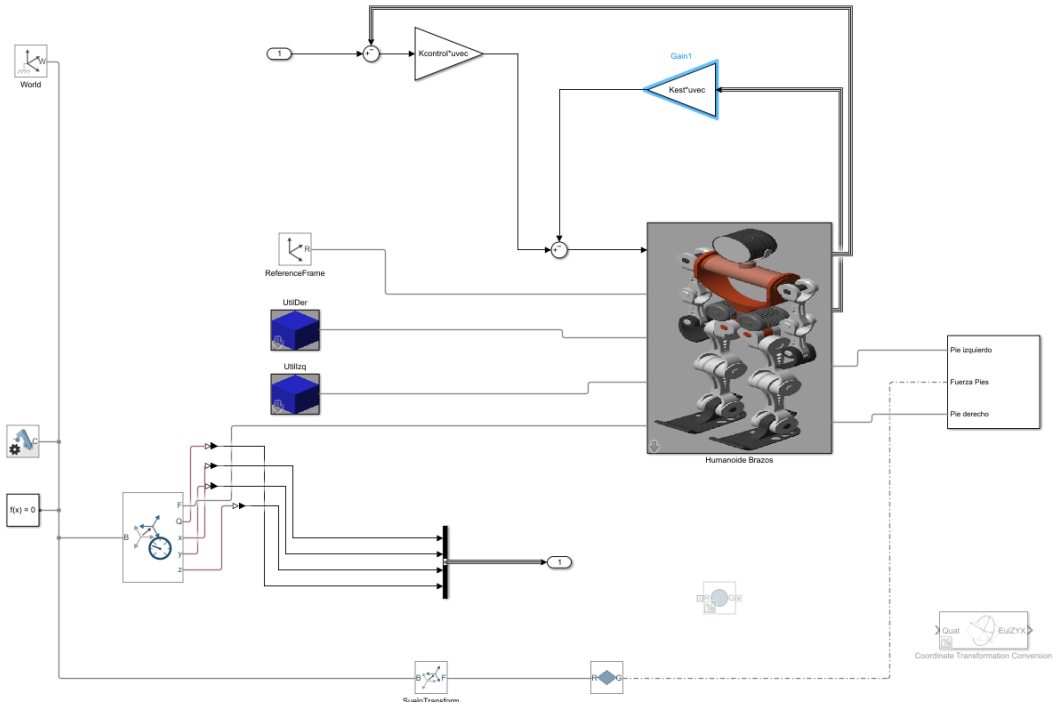
Bloques Simulink 38: Bloque articulación modelo dinámico PD

Con el objetivo de eliminar el error estacionario se empleó un control proporcional. Es cierto que se podía haber aplicado un control PI, pero como en el sistema funcionaba correctamente utilizando un proporcional no hemos incorporado el elemento integral.

Por otra parte, se integró la acción derivativa para eliminar las oscilaciones en los movimientos del robot. Para comprobar el efecto de este controlador en el robot, graficamos la posición y orientación de la cintura del robot utilizando un controlador proporcional y un controlador PD interno.

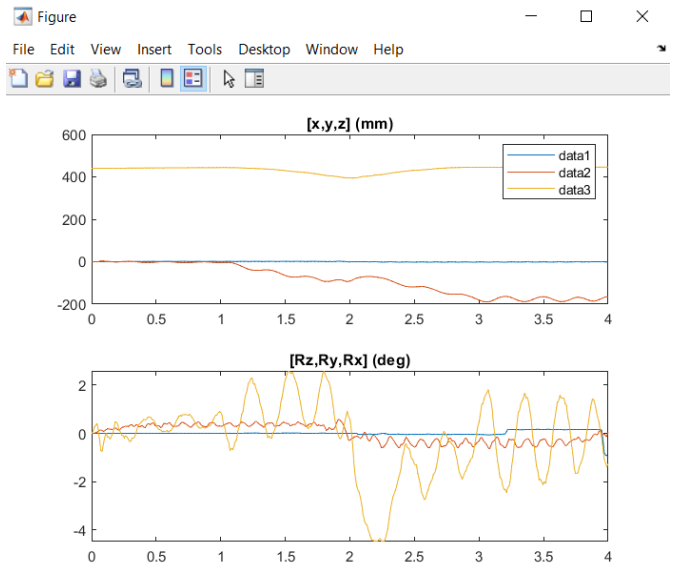


Bloques Simulink 39: Robot con controlador P



Bloques Simulink 40: Robot con controlador PD

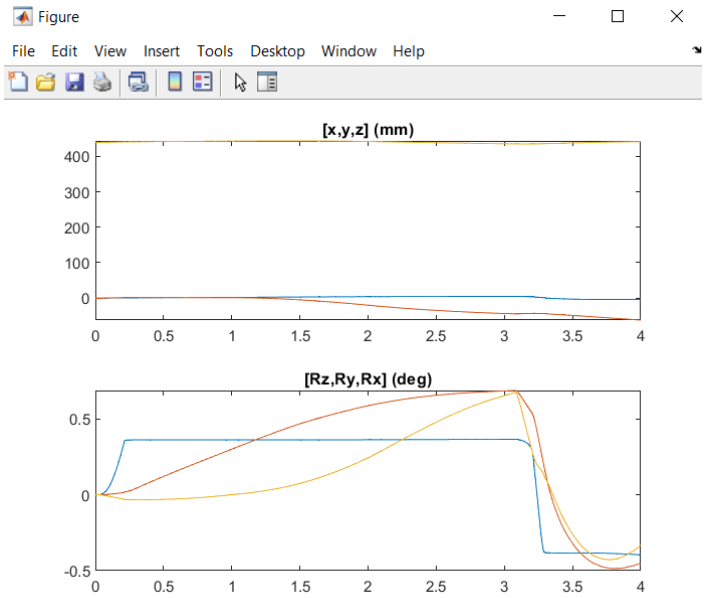
En primer lugar, representamos la posición y orientación de la cadera durante la simulación con el controlador proporcional. Aquí se representa la posición en x (Azul), y (rojo) y z (amarillo).



Gráfica 4: Posición robot con controlador P

En la gráfica observamos como las líneas amarilla y azul (x, z) de la gráfica de la posición permanecen más o menos constantes, mientras que la línea que representa el eje y va descendiendo con el tiempo. Con esto podemos suponer que el eje e indica la dirección de desplazamiento del robot cuando se mueve (avanzará en el eje y en el sentido negativo).

A continuación, se muestra la gráfica que representa la posición y rotación de la cadera del robot al incorporar un controlador proporcional derivativo al ejemplo anterior:



Gráfica 5: Posición robot con controlador PD

Podemos comprobar que, al introducir un controlador derivativo, tanto la posición como la orientación del robot cambia de forma mucho más suave y hay un menor número de oscilaciones. Con este controlador el robot moverá las articulaciones de forma mucho menos abrupta, por lo que se el movimiento se asemejará más al de un humano real.

Tras incorporar estos dos lazos, se consigue un modelo robótico estabilizado, por lo que los lazos de control explicados serán la base en los siguientes modelos de estación. Al estudiar el control de los humanoides, cada uno de los lazos de control formará un sistema independiente, de forma que se pueden anidar unos controladores dentro de otros para tener más control sobre el modelo.

4.3.3 CONTACTO PIES-SUELO

Uno de los aspectos más importantes que hay que comprender al analizar el modelo dinámico del humanoide es el modelado de la fuerza de contacto entre los pies del humanoide y el suelo. Con el objetivo de conseguir un modelo humanoide lo más parecido posible a una persona real, se planteará el modelo dinámico del robot. Para este modelado vamos a considerar que el humanoide está sujeto únicamente por las plantas de los pies, como una persona real. Este se sostiene gracias a la fuerza que ejercen las plantas de los pies sobre la superficie, que hace que el robot no se caiga por gravedad. Si el humanoide estuviera cogido por la cadera como en los ejemplos anteriores, sería un cuerpo totalmente fijo a la base que no podría desplazarse ni desestabilizarse independientemente del movimiento; esto es muy diferente de lo que sucede en un modelo real.

Como se ha expuesto en el apartado “3.2 Fuerzas externas” para controlar la fuerza de contacto entre dos sólidos, en este caso el pie y el suelo, se emplea el bloque Spatial Contact Force. Sin embargo, aquí surge el inconveniente de la superficie de los pies. Simscape Multibody emplea un método de penalización basado en puntos para modelar el contacto entre cuerpos. Este método significa que el bloque Spatial Contact Force aplica las fuerzas de contacto necesarias a sus cuerpos conectados en los puntos con la penetración máxima entre los dos cuerpos. Cada bloque de fuerza de contacto espacial solo aplica una única fuerza de contacto para cada cuerpo en cada paso de tiempo. Por ejemplo, en la siguiente ilustración, si la caja A penetra en la caja B, entonces se ejercerá una fuerza para compensar la esquina de la caja que tenga una penetración mayor. Tras aplicar esta fuerza, la caja A rotará y de nuevo habrá otro punto de penetración máximo sobre el

que actuará otra nueva fuerza. Este comportamiento es muy exigente para el Solver y además disminuirá notablemente la velocidad de la simulación.

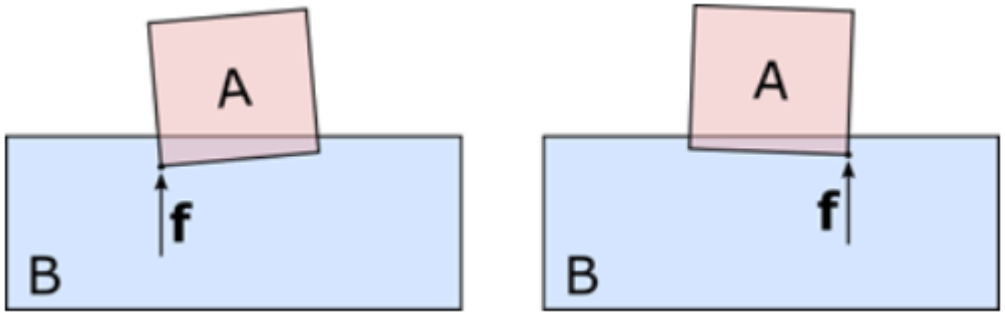


Ilustración 23: Contacto con método basado en puntos [18]

Para solucionar este impedimento, se utiliza el concepto de Contact Proxies, que se tratan de formas simples, como esferas, por ejemplo, para representar las partes de contacto entre los cuerpos. Así en el ejemplo anterior, utilizando Contact Proxis se anclarían 8 esferas, una en cada esquina, en vez de una sola esfera que modele el contacto entre los dos sólidos. A medida que se establece el contacto, la fuerza normal en cada esquina inferior será un cuarto del peso de la caja A.

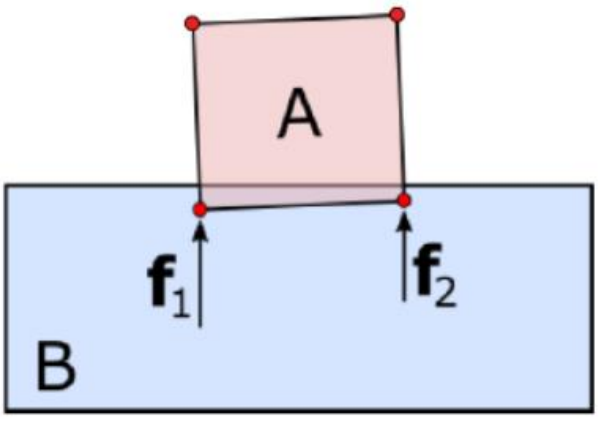
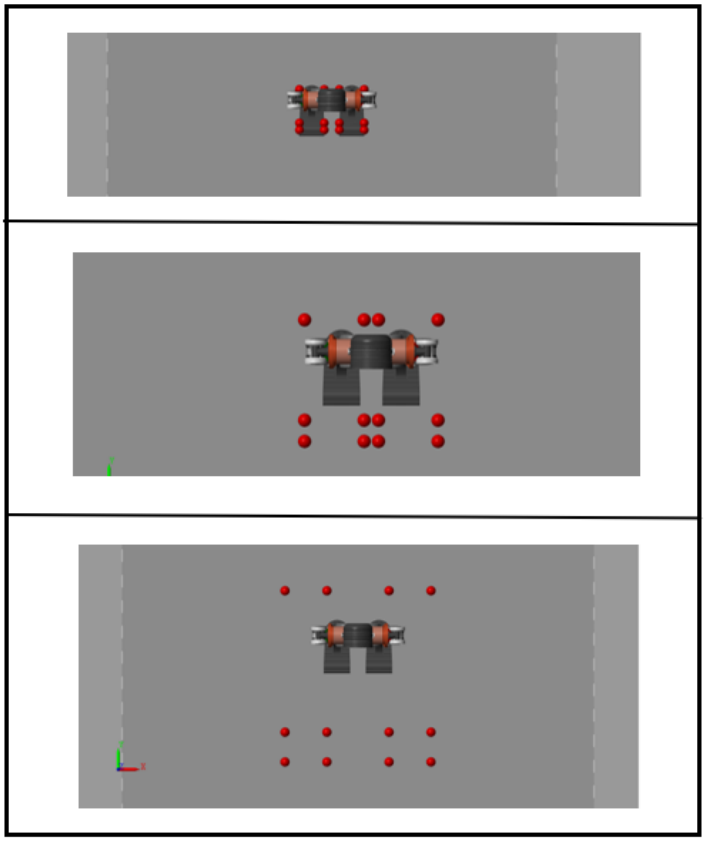


Ilustración 24: Contacto con método Contact Proxis [18]

Al utilizar Contact Proxies la rapidez y robustez del modelo varía; no solo influyen parámetros relacionados con la fricción o las fuerzas de contacto, sino que también afectan otras condiciones como la distancia de separación entre las esferas de contacto. A continuación, se muestran tres ejemplos diferentes en los que se ha variado la distancia de las esferas de contacto.



.

Simulación 13: Áreas diferentes de Contact Proxies humanoide

En el primero de los ejemplos, las esferas están justo en los límites de la planta del pie. Esto imita de forma aproximada a las fuerzas de contacto que hacen nuestros pies con el suelo, ya que el área de contacto de las esferas es del tamaño de la planta del pie. Sin embargo, debido a esta área tan reducida, el robot se vuelve muy inestable y se cae antes de dar el primer paso.

En el último caso, la distancia entre las esferas es mucho mayor que en el caso anterior. Aquí la superficie de contacto entre las esferas y el suelo es mucho más grande por lo que el robot tendrá una gran estabilidad al caminar. A pesar de esto, el ejemplo 3 no representa un modelo humano real, ya que la proporción entre el área de las plantas de los pies y el área que crean las esferas es muy diferente.

Para trabajar en los diferentes ejemplos utilizaremos un caso intermedio, en el que el área que forman las esferas sea más o menos aproximada a la superficie de la planta del pie pero que estén lo suficientemente separadas como para que el robot pueda mantenerse estable cuando camina.

Igualmente, también se tiene que considerar el área del suelo, ya que las esferas deben estar en contacto con la superficie para ejercer la fuerza; si reducimos la superficie del suelo es posible que alguna de las esferas quede fuera del sistema de fuerzas. Teniendo en cuenta estos detalles podremos conseguir que el robot se desplace sin ninguna fuerza externa que lo sostenga.

Para el resto de los ejemplos se cambiará el color de las esferas para hacerlas invisibles y así estudiar el resto de los aspectos del modelo robótico.

4.3.4 CLASE HUM Y PRPOPIEDAD EJE

Hasta ahora para definir el movimiento de una articulación lo que se hacía era indicar el valor en grados que se quería rotar la articulación e indicar el número de la articulación que se quiere mover (del 1 al 18). Esto sirve para el modelo humanoide que se ha obtenido con Matlab, pero existen muchos otros modelos humanoides que tienen un número de articulaciones diferentes que giran en direcciones también diferentes al modelo de Matlab.

Tras analizar en profundidad el humanoide que proporcionaba Matlab se consideró que todo lo que se ha aplicado hasta ahora podría destinarse igualmente a casos con humanoides diferentes. Para controlar diferentes humanoides de forma universal se planteó la idea de crear un objeto universal que defina de forma sencilla las articulaciones. Para nombrar las articulaciones se ha seguido la siguiente nomenclatura, pero el usuario puede definir los nombres que desee para las articulaciones:

*Tres primeras letras de cada articulación

*D si es del lado derecho e I del izquierdo.

*F en caso de que la articulación se mueva frontal (adelante, atrás), L si se mueven de forma lateral o G si giran sobre el eje z de la articulación.

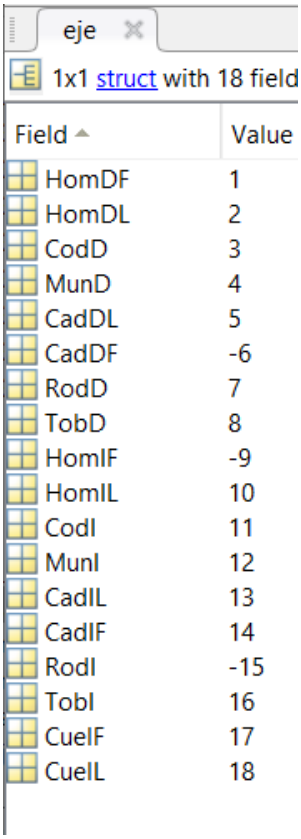


Ilustración 25: Objeto "eje" para humanoide "Matlab"

Las articulaciones de cada uno de los robots se guardarán en una propiedad "eje" donde los nombres de las articulaciones comunes en los robots humanoides que empleamos coinciden. Por ejemplo, para mover la rodilla derecha se podrá trabajar con la articulación RodD en cualquiera de los humanoides. Además, se pueden modificar los valores de las articulaciones para que giren en el mismo sentido y tener un control universal de las articulaciones de cualquier robot humanoide. Con esto se pretendía unificar todos los robots disponibles para controlarlos con los mismos comandos.

Este objeto eje es una de las propiedades de la clase Hum, que nos permite crear un objeto para modelar diferentes robots de forma dinámica. A esta clase se le pasa como argumentos la estación Simulink donde se ha planteado el modelo robótico, el objeto eje donde se guardan las articulaciones del robot, la base del robot (posición y rotación) y las coordenadas articulares iniciales del robot.

Hum.m

```
classdef Hum < handle
% Alberto Herreros López
% Versión 3-6-2022
    properties
       eje
       tQ
       q0
       baseRobot
       mdl
       dim
       nPts
    end % properties

    methods
        function this= Hum(mdl, eje, baseRobot, q0)
        % this= Hum(mdl, eje, baseRobot, q0)
        % mdl: modelo de simulink
        % eje: Estructura con las articulaciones
        %      y sus correspondiente valores en la entrada
        % baseRobot: [x,y,z,Rz,Ry,Rx] de la base del robot
        % q0: Valor de las articulaciones en la posición inicial

           this.mdl= mdl;
           this.eje= eje;
           this.dim= length(q0);
           this.q0= repmat(q0,[2,1]);
           this.baseRobot= repmat(baseRobot,[2,1]);
           this.Reset();
        end

```

Código 4: Clase Hum

Para definir cualquier movimiento del robot se crearán vectores joint donde en la primera columna se indicará el número de la articulación que se quiere mover y en la segunda columna el ángulo que se quiere desplazar la articulación. En un mismo vector joint se podrán definir desplazamientos de varias articulaciones a la vez y así se determina el movimiento del humanoide.

$$joint = [H.eje.articulación1, ang1; H.eje.articulación2, ang2 \dots] \qquad (14)$$

Estos vectores joint se pasarán como argumento la función que el usuario desee ejecutar. El objeto Hum presenta una serie de métodos para que el usuario pueda aprender cómo funciona el robot. Entre ellos destacamos TestMove() para hacer pruebas de los movimientos, AddMove() para añadir el movimiento definito a la simulación… Estos y el resto de los métodos se detallan "Capítulo VI: Anexos".

4.3.3.1 APLICACIÓN UNIVERSAL A HUMANOIDES

En este proyecto se incorporaron 3 robots humanoides diferentes para aplicar los métodos vistos anteriormente. Los modelos robóticos se tomaron de la plataforma de desarrollo colaborativo GitHub. Se trata de un servicio que permite construir y mantener software de las aplicaciones de cualquier desarrollador [19] . De aquí se realizó una búsqueda exhaustiva para encontrar sistemas robóticos que pudieran ser adaptados a las características del humanoide de Matlab. Los bodies que constituyen el modelo humanoide han sido extraídos de archivos visuales .stl. Para facilitar la integración de los modelos en el proyecto, se exploraron modelos cuya estructura ha sido creada a partir de ficheros .stl y se descartaron robots que partían de otros archivos como .dae. Igualmente, como el robot humanoide inicial se importaba de un archivo .urdf se descartaron los modelos obtenidos a partir de ficheros .xacro con formato de lenguaje .xml.

Teniendo en cuenta estos requisitos, se importaron dos nuevos robots humanoides y un modelo cuadrúpedo que se adaptaron para poder trabajar con ellos de la misma forma que el humanoide de Matlab.

HUMANOIDE ROID

El modelo robótico Roid es un humanoide a pequeña escala que se aplica principalmente al ámbito educativo. Se escogió este modelo porque los resultados obtenidos podrían reflejarse en un modelo físico real si el usuario así lo prefiere.

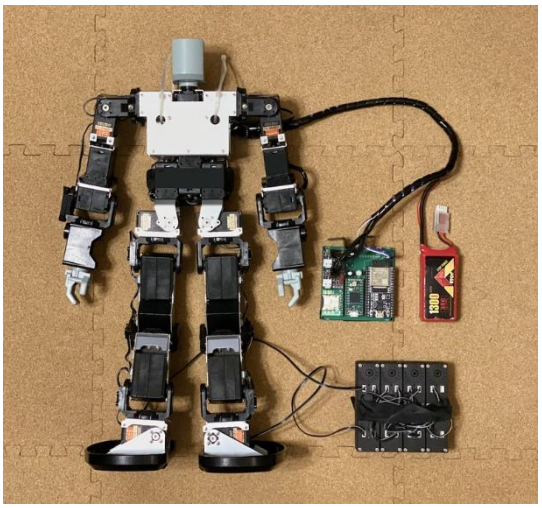
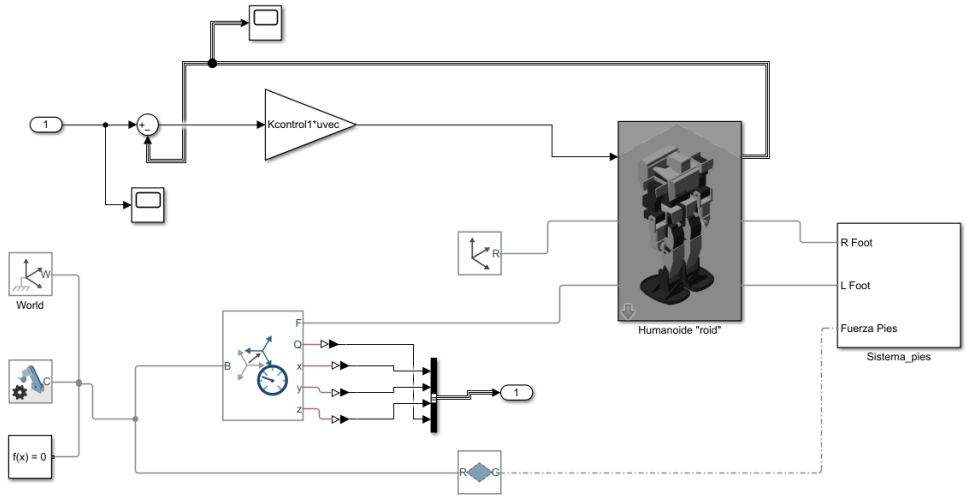


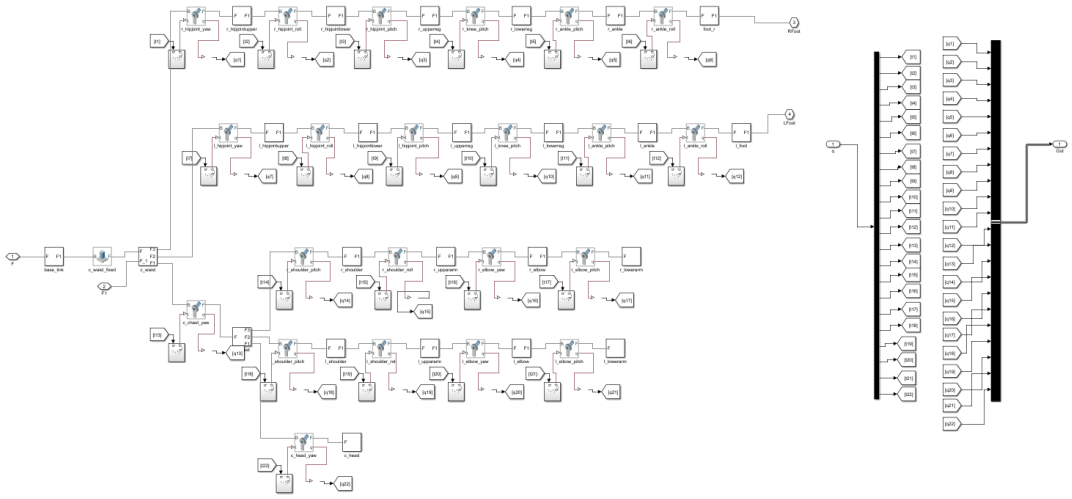
Ilustración 26: Imagen obtenida de [23]

Es un modelo que presenta 22 articulaciones rotativas en el que las articulaciones de las extremidades inferiores están referenciadas a la cintura y las articulaciones de las extremidades superiores y la cabeza están referenciadas al pecho; este a su vez se referencia a la cintura del robot. Los movimientos de roid son similares a los del robot de Matlab: puede rotar las mismas articulaciones a excepción de las muñecas; además tiene la capacidad de rotar las piernas en el eje z, los tobillos en el eje x y los brazos en el eje z, cosa que no puede hacer el humanoide inicial.

Se ha adaptado el modelo de Simulink para poder trabajar con el robot en un entorno dinámico, por ello se ha definido una realimentación de la posición articular y un subsistema para controlar la fuerza de los pies contra el suelo.



Bloques Simulink 41: Modelo humanoide "roid"



Bloques Simulink 42:Humanoide "roid"

HUMANOIDE ZJUDANCER

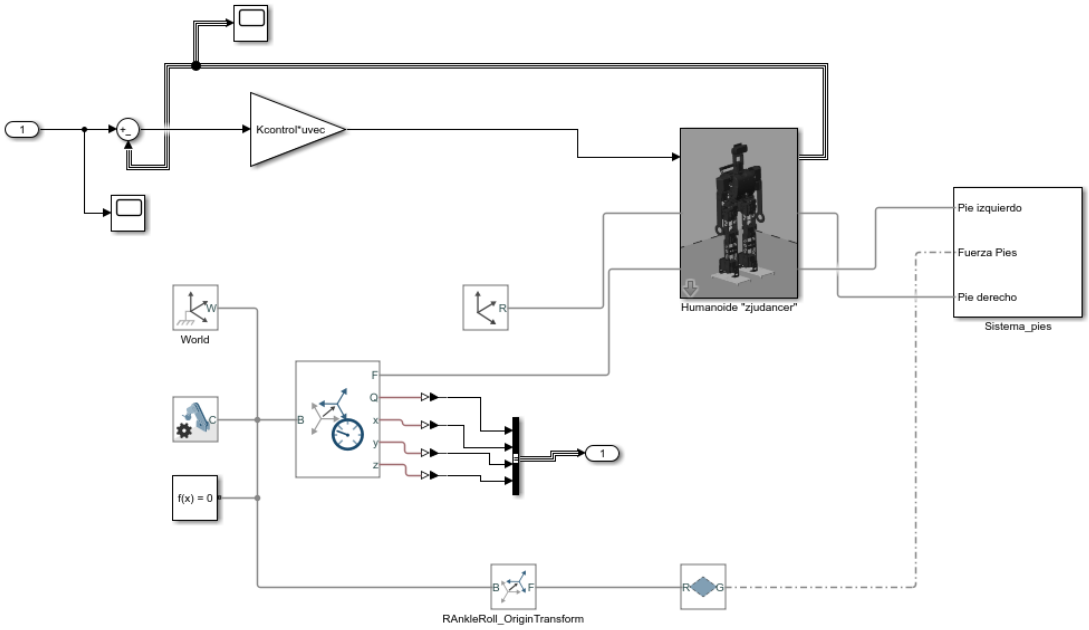
El humanoide Zjudancer fue presentado como candidato al concurso internacional RoboCup Humanoid League, cuyo objetivo es formar un equipo de robots humanoides que sea capaz de ganar contra el campeón mundial de fútbol de carne y hueso para el año 2050. Este robot ganó el segundo puesto en el año 2019 por su

Ilustración 27: Imagen obtenida de [22]

estabilidad y su gran número de grados de libertad, por ello se consideró que podría ser interesante para el proyecto.

Es un modelo que presenta 18 articulaciones rotativas en el que las articulaciones de las extremidades superiores, inferiores y cabeza están referenciadas al pecho que es el sistema de referencia global del robot. Los movimientos de zjudancer son similares a los del robot de Matlab: puede rotar las mismas articulaciones a excepción de las muñecas; además tiene la capacidad de rotar las piernas en el eje z y los tobillos en el eje x cosa que no puede hacer el humanoide inicial.

De la misma forma que antes, se ha adaptado el modelo de Simulink para poder trabajar con el robot en un entorno dinámico, por ello se ha definido una realimentación de la posición articular y un subsistema para controlar la fuerza de los pies contra el suelo.



Bloques Simulink 43: Modelo humanoide "zjudancer"

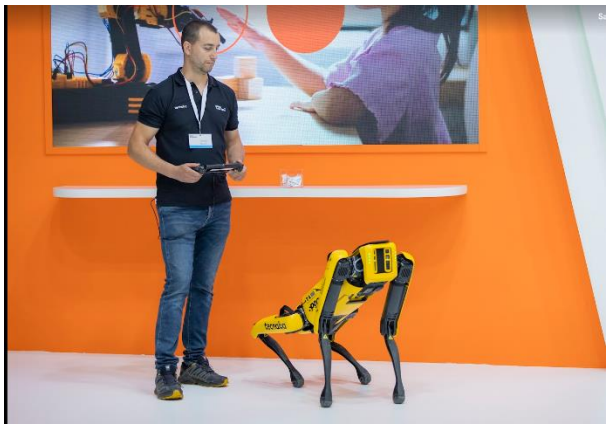
Bloques Simulink 44: Humanoide "zjudancer"

CUADRÚPEDO "YOBOTICS"

Los métodos y funciones empelados se pueden utilizar para todo tipo de sistemas robotizados independientemente de su fisiología antropomórfica. Para verificar la universalidad de las herramientas utilizadas se ha tomado un robot cuadrúpedo que se ha adaptado para que se pueda mover con los mismos comandos que los humanoides anteriores.

Ilustración 28: Imagen obtenida de [20]

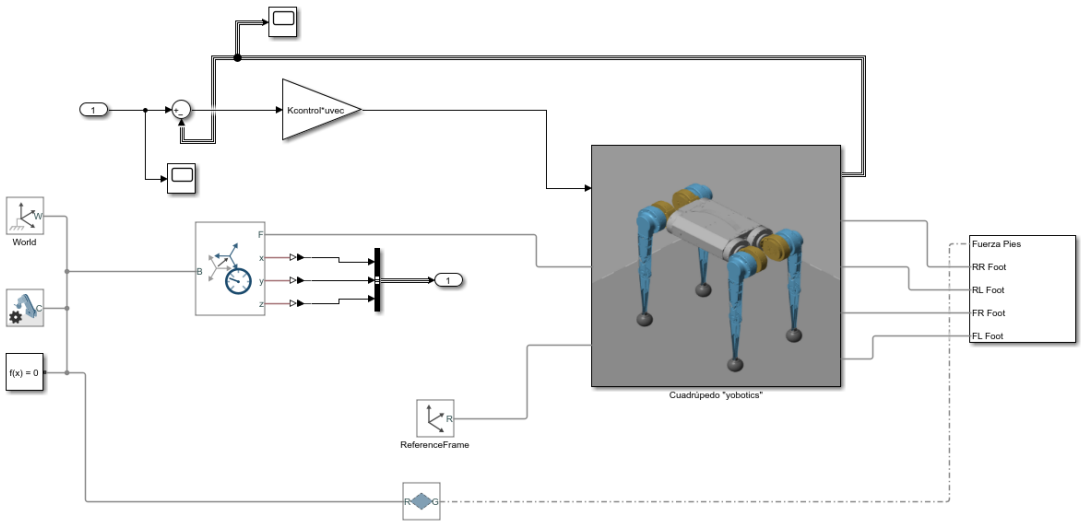
El modelo cuadrúpedo obtenido tiene los mismos grados de libertad y configuración que algunos de los robots cuadrúpedos más avanzados del mercado. Un ejemplo es el robot cuadrúpedo de Boston Dynamics con visión inteligente que ha presentado el centro tecnológico español Tecnalia en la Bienal de la Máquina Herramienta este mismo año 2022. Dicho robot está dotado de capacidades móviles y de reconocimiento inteligente del entorno en el que se aplican las últimas tecnologías de robótica flexible. [20]. Ante esta innovadora apuesta, se decidió incorporar al proyecto el modelo yobotics que cuenta con el mismo número de articulaciones y grados de libertad que el robot presentado por Tecnalia.



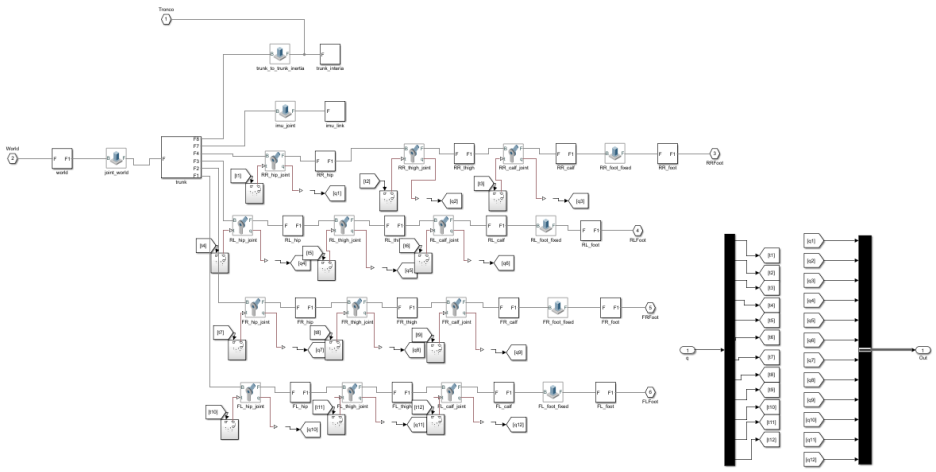
El modelo yobotics cuenta con 12 articulaciones rotativas que le permiten al robot mover cada pata de forma frontal, lateral y doblar la rodilla. Igualmente se ha adaptado el modelo de Simulink para que el robot trabaje de forma dinámica. En este caso se ha utilizado una esfera de contacto en cada pata porque era suficiente para lograr el contacto entre el suelo y el pie del robot.

Las órdenes que se den al robot no serán iguales que las que se pasan a los humanoides.

Por ejemplo, para que un humanoide ande hay que movilizar las dos extremidades inferiores, sin embargo, aquí hay que controlar las cuatro extremidades. A pesar de esto, los objetos y los controladores que se utilicen para optimizar el comportamiento del robot son iguales que en los otros modelos.



Bloques Simulink 45: Modelo cuadrúpedo "yobotics"



Bloques Simulink 46: Cuadrúpedo "yobotics"

Para el caso de los humanoides, podemos hacer una comparación de los números de cada articulación y su sentido (positivo o negativo) que se han asignado en el objeto “eje” en cada modelo:

Nombre articulación	Matlab	Roid	Zjudancer
HomDF	1	-14	1
HomDL	2	15	X
CodD	3	17	2
MunD	4	X	X
CadDL	5	2	4
CadDF	6	-3	-5
RodD	7	4	6
TobD	8	5	7
HomIF	-9	-18	17
HomIL	10	19	X
CodI	11	21	18
MunI	12	X	X
CadIL	13	8	10
CadIF	14	-9	-11
RodI	15	10	12
TobI	16	11	13
CuelF	17	X	16
CuelL	18	22	15
CadDG	X	1	3
CadIG	X	7	9
TobDL	X	6	8
TobIL	X	X	14
CodDG	X	16	X
CodIG	X	20	X
TroL	X	13	X

Así, si por ejemplo se quiere realizar una rotación de 30 grados del codo derecho si no se hubiera implementado el objeto “eje” el código sería:

```
%MATLAB
joint= [3, 30];
tQ= MoveHum(tQ, joint, tinc);
sol= sim(Estacion,tQ(:,1),[],tQ);

%ROID
joint= [17, 30];
tQ= MoveHum(tQ, joint, tinc);
sol= sim(Estacion,tQ(:,1),[],tQ);
```

```
%ZJUDANCER
joint= [2, 30];
tQ= MoveHum(tQ, joint, tinc);
sol= sim(Estacion,tQ(:,1),[],tQ);
```

En cambio, con el objeto eje podríamos definir el movimiento de cualquiera de los humanoides, definidos como H, con estas líneas;

```
%MATLAB, ZJUDANCER,ROID
H= Hum(Estacion, eje, baseRobot_, q0_);

joint= [H.eje.CodD, 30];
H.AddMove(joint, 1)

H.Sim
```

Con esto se ha conseguido controlar de forma universal cualquier modelo robótico humanoide empleando un objeto único. El objeto Hum también podría aplicarse al modelo cuadrúpedo, sin embargo, hay que tener en cuenta que un robot de 4 piernas no camina de la misma forma que un robot bípedo. En el apartado "Capítulo VI: Anexos" se muestran dos códigos de una posible combinación de articulaciones y rotaciones para que pueda caminar un robot humanoide y un robot cuadrúpedo.

4.3.5 APRENDIZAJE MANUAL

4.3.5.1 FUNCIONES PARA EL APRENDIZAJE

Una forma de controlar el robot es enseñar al humanoide a realizar un movimiento paso a paso. Esto quiere decir que para que el humanoide se mueva es necesario pasar a cada articulación los grados que se quiere mover. Este método es parecido a cuando se enseña a andar a un niño, donde hay que explicar al niño como doblar las rodillas y como apoyar los pies para que no se caiga.

Para que el usuario pueda aprender de forma manual cómo funciona el robot, se creó una clase Hum con el que se define el objeto humanoide como se ha explicado anteriormente en "4.3.4 Clase Hum y objeto eje".

Dentro de esta clase se han creado diferentes funciones para diseñar los movimientos básicos del robot de forma dinámica como son TestMove o AddMove. A partir de estos métodos se han diseñado funciones para hacer acciones completas, por ejemplo, dar un paso o andar. Para ello se ha

introducido en la función diferentes joint que combinándolos dan lugar al movimiento adecuado.

Para simular un paso completo se ha definido el método Paso() que emplea cuatro vectores joints para determinar los movimientos articulares que hacen falta para dar un paso. En el caso de la función Marcha() algunos vectores joint se repetirán en bucle para programar el número de pasos que dará el robot. Marcha() se forma mediante 6 pasos que se repiten de 3 en 3 con el pie derecho y con el izquierdo; lo que se ha hecho es repetirlo como ciclos para que se convierta en un aprendizaje "automático". Además, podemos plantear los movimientos del robot en función de la amplitud del paso, teniendo en cuenta la proporción de pesos que guardan las diferentes articulaciones.

```
function [joint1,joint2,joint3,joint4]=Paso (this)
    %  Mueve la cadera, rodillas de la pierna derecha y los dos brazos
    joint1= [this.eje.CadIF, 60; this.eje.RodI, 60; this.eje.HomDF, 30, ; this.eje.HomIF, -30];
    joint2= [this.eje.RodD, 60; this.eje.TobD, -60];
    joint3= [this.eje.CadIF, 0; this.eje.RodI, 0];
    joint4= [this.eje.RodD, 0; this.eje.TobD, 0; this.eje.HomDF, 0; this.eje.HomIF, 0];

end
```

```
function [joint1,joint2,joint3,joint4,joint5,joint6,joint7,joint8]=Marcha(this)
   %Primeros dos movimientos para empezar
   paso=60;
    joint1= [this.eje.CadIF, paso; this.eje.RodI, paso; this.eje.HomDF, paso/2, ; this.eje.HomIF, -paso/2];
    joint2= [this.eje.RodD, paso; this.eje.TobD, -paso];

   %Movimientos que se repetirán en bucle
    joint3= [this.eje.CadIF, 0; this.eje.RodI, 0];
    joint4= [this.eje.CadDF, paso;  this.eje.TobD, 0; this.eje.HomDF, -paso/2, ; this.eje.HomIF, paso/2];
    joint5= [this.eje.RodI, paso; this.eje.TobI, -paso];
    joint6= [this.eje.CadDF, 0; this.eje.RodD, 0];
    joint7= [this.eje.CadIF, paso; this.eje.TobI, 0; this.eje.HomDF, paso/2, ; this.eje.HomIF, -paso/2];
    joint8= [this.eje.RodD, paso; this.eje.TobD, -paso];

end
```

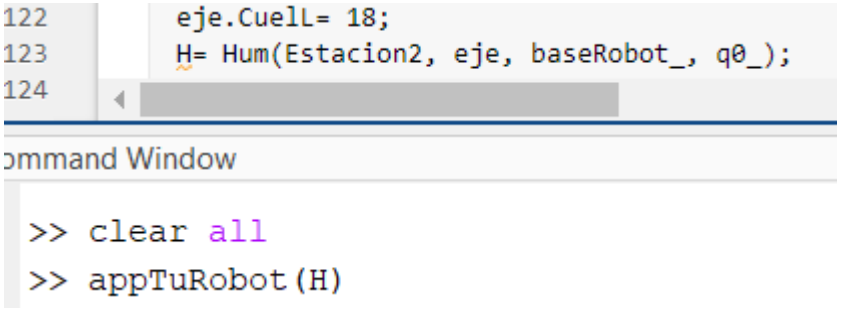
Una vez planteados los diferentes joint, se podrán añadir a la simulación final mediante el comando AddMove. El usuario podrá plantear cualquier movimiento combinado que desee de esta forma sin necesidad de probar el movimiento antes si así lo desea.

Se han tomado estas dos nuevas funciones para comprobar el movimiento del modelo humanoide, sin embargo, el usuario puede jugar con el robot e incorporar tantas funciones como desee para guardar movimientos que le hayan parecido interesantes para controlar el humanoide.

4.3.5.2 INTERFAZ PARA EL APRENDIZAJE

El aprendizaje manual permitirá al usuario interaccionar con el robot que desee para que pueda controlar diferentes movimientos y realizar una simulación completa. Con el objetivo de hacer más ameno el control del robot al usuario, se ha propuesto el diseño de una interfaz con la extensión de Matlab AppDesigner. En ella se disponen una serie de controles con los que el usuario puede jugar para crear el movimiento de su robot. Para facilitar la comprensión de la interfaz al usuario, se han enumerado los botones de forma que vaya realizando las diferentes acciones de control en el orden adecuado.

La interfaz se inicializa directamente desde Matlab; hay que escribir en la ventana de comandos el nombre del archivo AppDesigner y pasar como argumento el nombre del objeto Hum que se ha definido para el robot que queramos simular. Por ejemplo, para el robot humanoide de Matlab:



Una vez que se llama a la app se mostrará la interfaz al usuario automáticamente. Dentro del objeto Hum se ha definido un parámetro numérico que servirá para indicar con qué modelo de los tres se está trabajando. En función de este parámetro se señalará con un recuadro amarillo cuál de los tres robots se ha importado cuando se abra la interfaz.

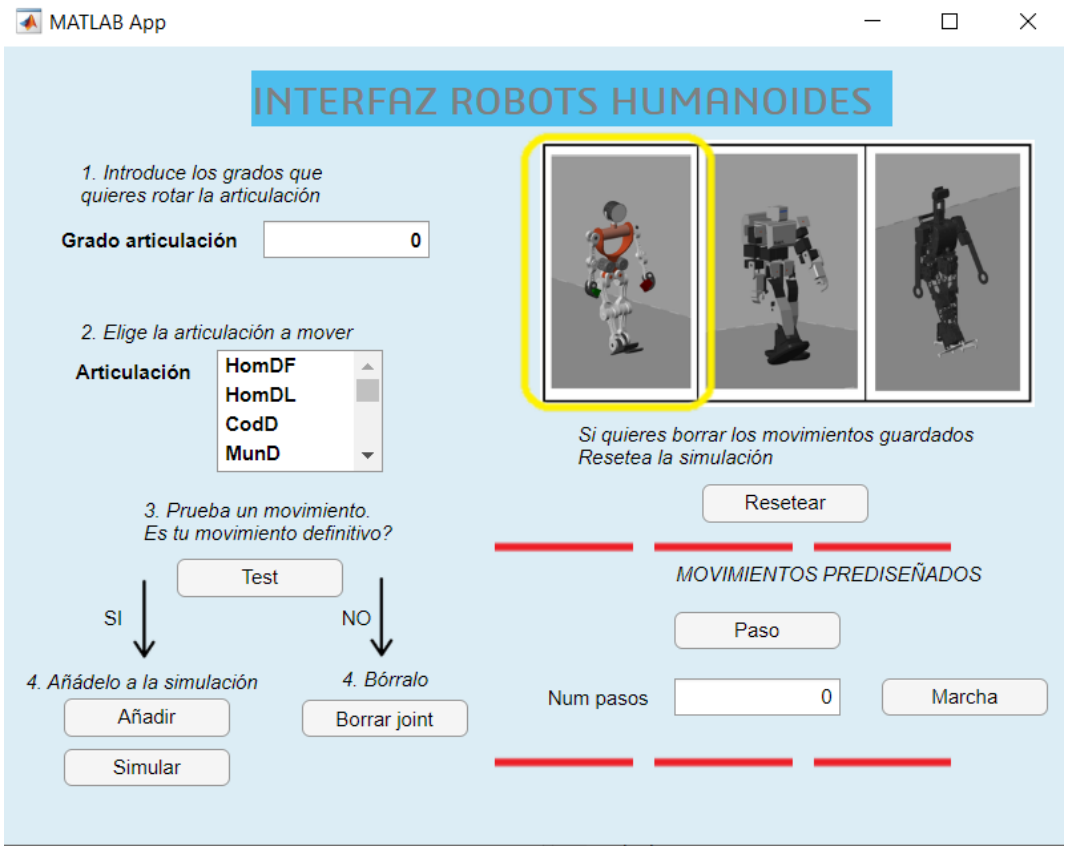


Ilustración 29: Interfaz AppDesigner

En primer lugar, el usuario debe introducir la posición en grados que desea alcanzar con una articulación. Seguidamente debe seleccionar de la lista la articulación que desea rotar. A continuación, puede hacer click sobre el botón Test para probar el movimiento. El usuario puede continuar añadiendo varias rotaciones simultáneas en diferentes articulaciones con el botón Test; todos estos movimientos se realizarán a la vez en el mismo intervalo temporal. Cuando el usuario considere que el conjunto de rotaciones de las articulaciones es definitivo, entonces podrá añadirlo a la Simulación. Al hacer click sobre Simular se podrán ver todos los movimientos que se han almacenado hasta el momento.

Si el usuario considera que el conjunto de rotaciones que ha planteado no es adecuado, entonces podrá pulsar sobre Borrar Joint para comenzar otro conjunto de rotaciones. Es importante resaltar que una vez que el usuario añada un movimiento a la simulación este no podrá eliminarse de forma individual, por ello el botón Borrar Joint nos da la oportunidad de rectificar el movimiento antes de añadirlo de forma definitiva.

En caso de que el usuario quiera borrar todos los movimientos almacenados en la simulación podrá hacer click sobre resetear para empezar de cero.

Igualmente se han establecido una serie de botones para que el humanoide lleve a cabo movimientos prediseñados como dar un paso o caminar el número de pasos que se le indique.

4.3.5.3 APLICACIÓN UNIVERSAL A HUMANOIDES

Durante el diseño de la interfaz para controlar el movimiento del robot humanoide de Matlab, se tuvo en cuenta su posible aplicación a otros modelos humanoides diferentes. Para poder controlar los movimientos del humanoide a través del interfaz, basta con definir el objeto Hum con la estación y las articulaciones correspondientes del humanoide que queremos simular. Una vez creado el objeto humanoide el usuario podrá jugar con los botones para hacer que su humanoide se mueva.

En la clase Hum que se ha explicado anteriormente, las articulaciones de los ejemplos de robots humanoides se han almacenado con el mismo nombre en una estructura "eje". En la interfaz se ha escogido un componente ListBox que muestra Items de una estructura. Como las articulaciones también están almacenadas en una variable de tipo estructura se ha podido pasar este argumento a la ListBox. Gracias a este planteamiento, las articulaciones de los tres robots comparten el mismo nombre y se podrán seleccionar de la misma manera en los tres casos a pesar de que el nombre de una articulación esté asociada a números diferentes en los modelos humanoides.

El problema de la rotación se ha solucionado modificando el signo de cada una de las articulaciones para que la rotación sea la misma en los tres casos al introducir un valor positivo o negativo de grados. Un ejemplo de esta aplicación de la interfaz se encuentra en el apartado "Capítulo V: Aplicaciones y líneas futuras".

Con este planteamiento, el usuario podrá jugar con los robots que se han mostrado en este informe e incluso añadir nuevos robots a la galería adaptándolos a estas funciones. De hecho, este tipo de control del movimiento podría aplicarse no solo a robots humanoides sino a muchos otros modelos robóticos cuadrúpedos o con movimientos totalmente diferentes.

4.3.6 LAZOS EXTERNOS DE CONTROL

Hasta ahora se ha conseguido que los modelos se mueven de forma estable gracias al lazo interno de control, pero tenemos que detallar qué ángulo tiene que tomar cada articulación en cada momento de la simulación. El software

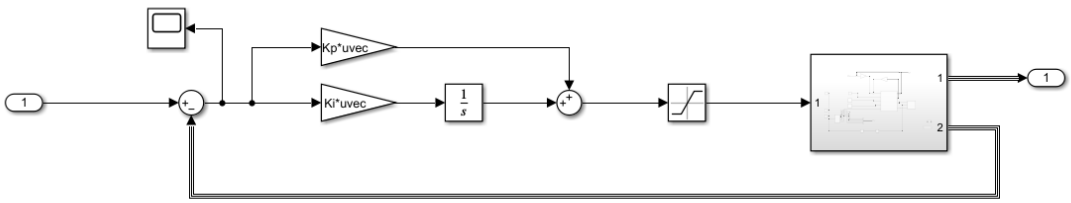
diseñado para el aprendizaje manual es útil para que el usuario juegue con el robot y comprenda su funcionamiento por prueba y error, sin embargo, este modelo tiene un gran potencial y se pueden añadir lazos de control a niveles superiores para lograr un objetivo concreto. Mediante controladores externos "inteligentes" el robot podrá realizar acciones como resultado de un acontecimiento, un comportamiento similar a los reflejos o incluso pensamiento de una persona real.

4.3.5.3 CONTROLADOR AUTOMÁTICO EN POSICIÓN DE EJES

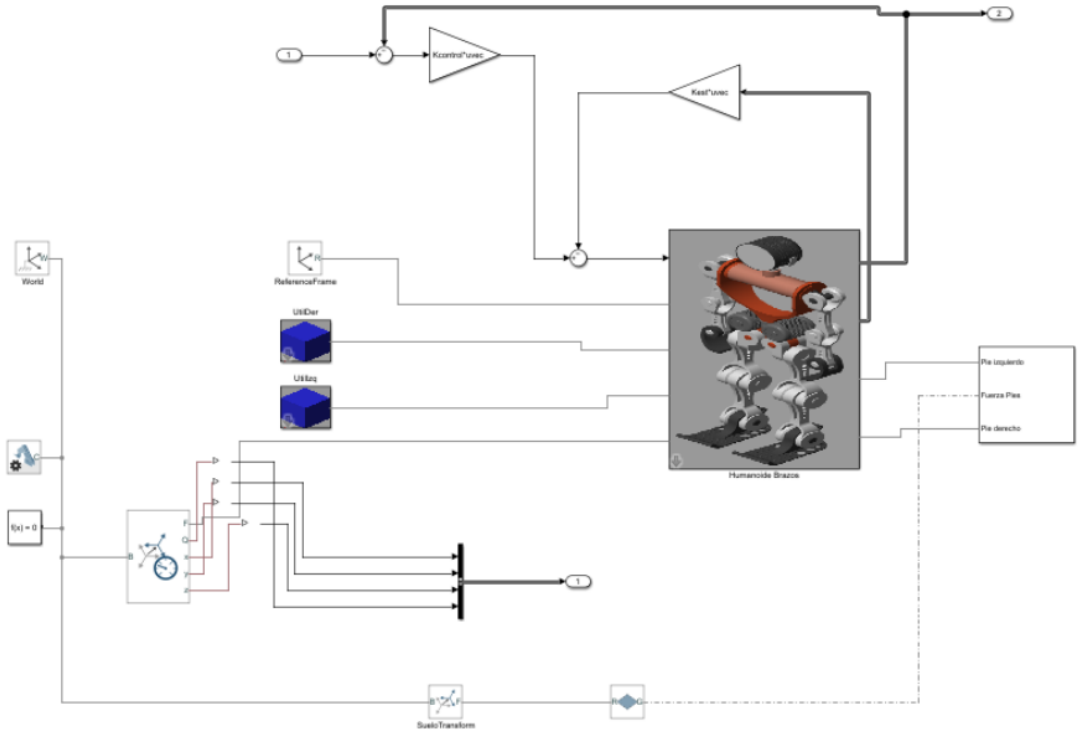
El método de aprendizaje manual es útil para que el usuario se acerque un poco más al mundo de la robótica y comprenda el funcionamiento del modelo a simular, sin embargo, esto no representa el movimiento real de una persona. Cuando el ser humano realiza una acción, lo hace persiguiendo un objetivo y los movimientos de las articulaciones salen de forma automática. Por ejemplo, si queremos desplazarnos hacia adelante, el objetivo es avanzar en dirección recta y para conseguirlo movemos de forma inconsciente las piernas.

En el siguiente ejemplo se tomará como objetivo alcanzar la posición de los ejes deseada. Para ello se incorporará un lazo externo con acción PI al modelo dinámico anterior (que ya presentaba el lazo interno de control básico que se ha explicado anteriormente).

En este lazo de control se realimentará la posición de los ejes para que vayan a las referencias elegidas de forma automática. Esto lo conseguiremos introduciendo un controlador proporcional-integral de todas las articulaciones (18 Pis) para conseguir la referencia objetivo.



Bloques Simulink 47: Lazo de control automático en posición de ejes



Bloques Simulink 48: Modelo robótico para lazo de control automático en posición de ejes

Se emplea una acción de tipo proporcional-integral para reducir el error estacionario en el sistema. Los dos controladores emplean como entrada la posición articular y a partir de ella definen la acción de control. A continuación, se muestra el diagrama de bloques simplificado de este sistema, donde la planta corresponde con el modelo dinámico con el controlador básico PD:

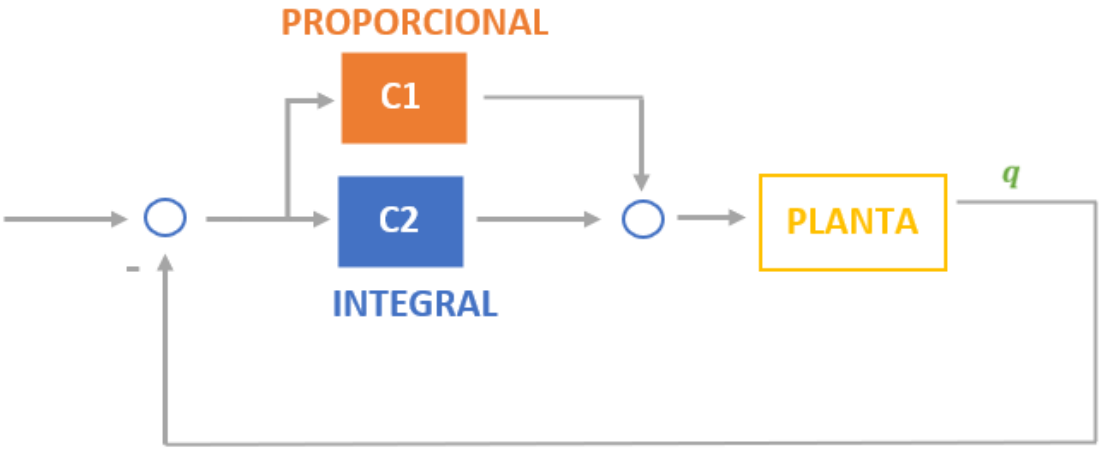


Ilustración 30: Diagrama bloques sistema con controlador de ejes por posición

Para conseguir la referencia deseada con el PI se almacenarán el conjunto de todos los movimientos que debe hacer el robot en la simulación en una estructura "joint". En este ejemplo el robot dará dos pasos aplicando un conjunto de 6 movimientos en sus articulaciones. De esta forma se creará una estructura joint, de 6 celdas y en cada una de ellas se dispondrá cada articulación y la posición del eje en grados, y un vector de trayectorias (tQ), de 6 filas y un número de columnas igual al número de articulaciones y el tiempo.

joint

1x6 cell

	1	2	3	4	5	6
1	[14,30;-15,...	[7,30;8,-30]	[14,0;-15,0]	[-6,30;8,0;1,...	[-15,30;16,-...	[-6,0;7,0]
2						
3						

Ilustración 31: Estructura joint con control automático

joint | joint{1, 1}

joint{1, 1}

	1	2
1	14	30
2	-15	30
3	1	15
4	-9	-15
5		

Ilustración 32: Estructura joint{1,1} con control automático

H.tQ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	0.2618	0	0	0	0	0	0	0	0.2618	0	0	0	0	0.5236	-0.5236	0	0	0
2	2	0.2618	0	0	0	0	0	0.5236	-0.5236	0.2618	0	0	0	0	0.5236	-0.5236	0	0	0
3	3	0.2618	0	0	0	0	0	0.5236	-0.5236	0.2618	0	0	0	0	0	0	0	0	0
4	4	-0.2618	0	0	0	0	-0.5236	0.5236	0	-0.2618	0	0	0	0	0	0	0	0	0
5	5	-0.2618	0	0	0	0	-0.5236	0.5236	0	-0.2618	0	0	0	0	0	-0.5236	-0.5236	0	0
6	6	-0.2618	0	0	0	0	0	0	0	-0.2618	0	0	0	0	0	-0.5236	-0.5236	0	0

Ilustración 33: Vector tQ con control automático

Esto es diferente de los casos anteriores, donde el vector tQ acumulaba los valores en radianes que tenía que alcanzar cada articulación de la simulación a lo largo del tiempo, de forma que cada incremento del tiempo (1 segundo) se almacenaban 10 valores en radianes para una articulación. Además, se creaban 6 vectores diferentes joint para añadir cada movimiento a la simulación.

joint1	[14,40;-15,40;1,2...
joint2	[7,40;8,-40]
joint3	[14,0;-15,0]
joint4	[-6,40;8,0;1,-20;-9...
joint5	[-15,40;16,-40]
joint6	[-6,0;7,0]
joint7	[14,40;16,0;1,20;-...
joint8	[7,40;8,-40]

Ilustración 34: Vectores joint sin control automático

H.tQ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1.0000e-03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.1120	0.0388	0	0	0	0	0	0	0	0.0388	0	0	0	0	0.0776	-0.0776	0	0	0
3	0.2230	0.0776	0	0	0	0	0	0	0	0.0776	0	0	0	0	0.1551	-0.1551	0	0	0
4	0.3340	0.1164	0	0	0	0	0	0	0	0.1164	0	0	0	0	0.2327	-0.2327	0	0	0
5	0.4450	0.1551	0	0	0	0	0	0	0	0.1551	0	0	0	0	0.3103	-0.3103	0	0	0
6	0.5560	0.1939	0	0	0	0	0	0	0	0.1939	0	0	0	0	0.3879	-0.3879	0	0	0
7	0.6670	0.2327	0	0	0	0	0	0	0	0.2327	0	0	0	0	0.4654	-0.4654	0	0	0
8	0.7780	0.2715	0	0	0	0	0	0	0	0.2715	0	0	0	0	0.5430	-0.5430	0	0	0
9	0.8890	0.3103	0	0	0	0	0	0	0	0.3103	0	0	0	0	0.6206	-0.6206	0	0	0
10	1	0.3491	0	0	0	0	0	0	0	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
11	1.0010	0.3491	0	0	0	0	0	0	0	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
12	1.1120	0.3491	0	0	0	0	0	0.0776	-0.0776	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
13	1.2230	0.3491	0	0	0	0	0	0.1551	-0.1551	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
14	1.3340	0.3491	0	0	0	0	0	0.2327	-0.2327	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
15	1.4450	0.3491	0	0	0	0	0	0.3103	-0.3103	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
16	1.5560	0.3491	0	0	0	0	0	0.3879	-0.3879	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
17	1.6670	0.3491	0	0	0	0	0	0.4654	-0.4654	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
18	1.7780	0.3491	0	0	0	0	0	0.5430	-0.5430	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
19	1.8890	0.3491	0	0	0	0	0	0.6206	-0.6206	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
20	2	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
21	2.0010	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.6981	-0.6981	0	0	0
22	2.1120	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.6206	-0.6206	0	0	0
23	2.2230	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.5430	-0.5430	0	0	0
24	2.3340	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.4654	-0.4654	0	0	0
25	2.4450	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.3879	-0.3879	0	0	0
26	2.5560	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.3103	-0.3103	0	0	0
27	2.6670	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.2327	-0.2327	0	0	0
28	2.7780	0.3491	0	0	0	0	0	0.6981	-0.6981	0.3491	0	0	0	0	0.1551	-0.1551	0	0	0

Ilustración 35: Vector tQ sin control automático

Al emplear un sistema en lazo de control automático en ejes el robot podrá mover sus articulaciones indicando únicamente la posición final de la articulación a la que se quiere llegar, que será la referencia. Así el humanoide llega a esta referencia de forma automática, sin pasarle de forma detallada los grados que debe rotar cada articulación.

Con este planteamiento, el robot no sigue un proceso de aprendizaje, sino que ya ha asimilado el movimiento y es capaz de realizarlo sabiendo tan solo el punto de destino. En este caso no será necesario hacer interpolaciones entre dos posiciones articulares, por lo que está reflejando la capacidad de aprendizaje de un humano real. Se puede comparar al proceso de aprender a caminar de un niño; primero hay que enseñarle cómo apoyar el pie y doblar las rodillas para que no se caiga y una vez que ha aprendido cómo dar un paso el movimiento de caminar lo realizará de forma automática, sin pensar cómo tiene que mover cada articulación de su pierna.

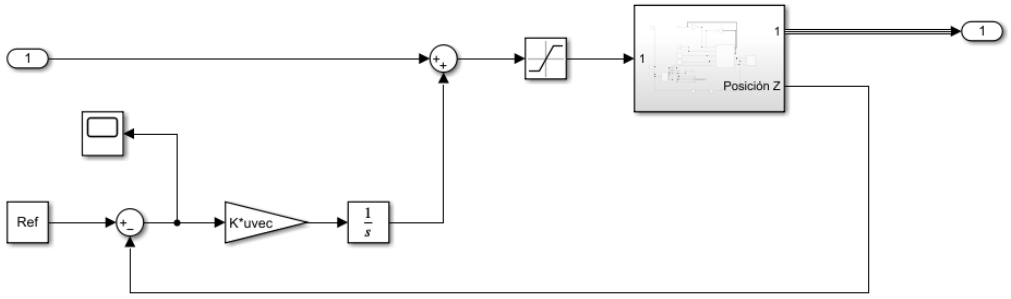
4.3.6.2 CONTROL AUTOMÁTICO CON OBJETIVO

Además, se pueden incorporar otras realimentaciones extras para conseguir que el robot siga una referencia u objetivo distinto de la posición de los ejes. Aquí planteamos un controlador inteligente que puede servir para optimizar el objetivo que queramos: la posición de las articulaciones del robot, la estabilidad, la distancia entre los pies y el suelo… Gracias a esto se abre un

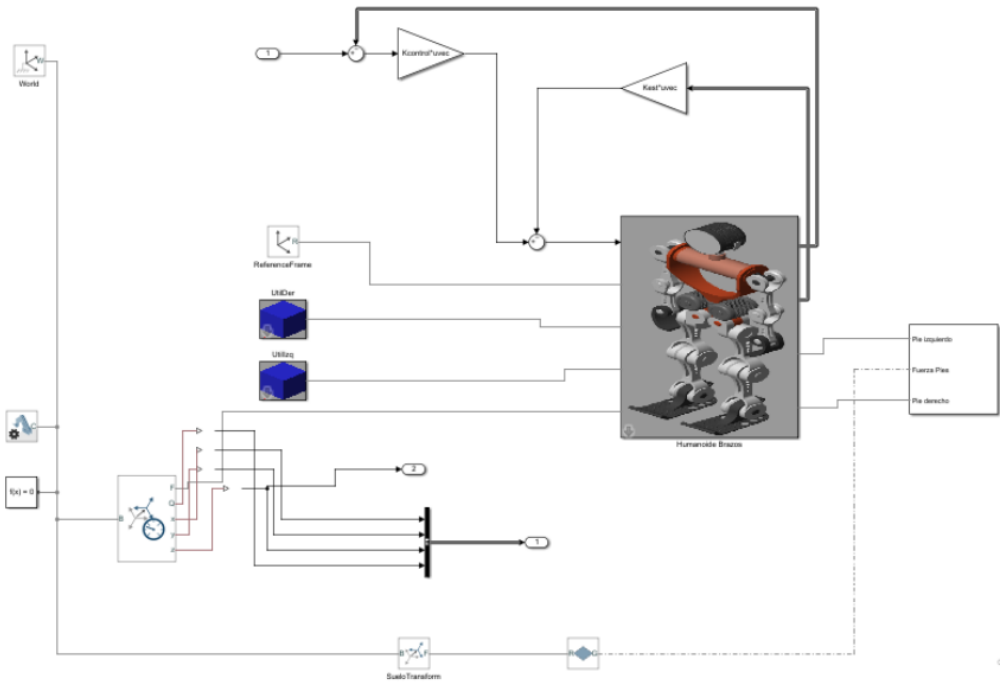
abanico enorme de posibilidades para controlar el robot, pero esto es muy difícil de implantar y diseñar por lo que se deja como posible línea de investigación para proyectos futuros.

Para comprender mejor los lazos externos que se podrían aplicar se plantea un ejemplo de control automático que tiene como objetivo la posición z de la cadera del robot.

Empleamos un controlador integral para minimizar el error entre la posición real en el eje z de la cadera y la referencia deseada.



Bloques Simulink 49: Lazo de control automático con objetivo



Bloques Simulink 50: Modelo robótico para lazo de control automático con objetivo

El diagrama de bloques sería el siguiente:

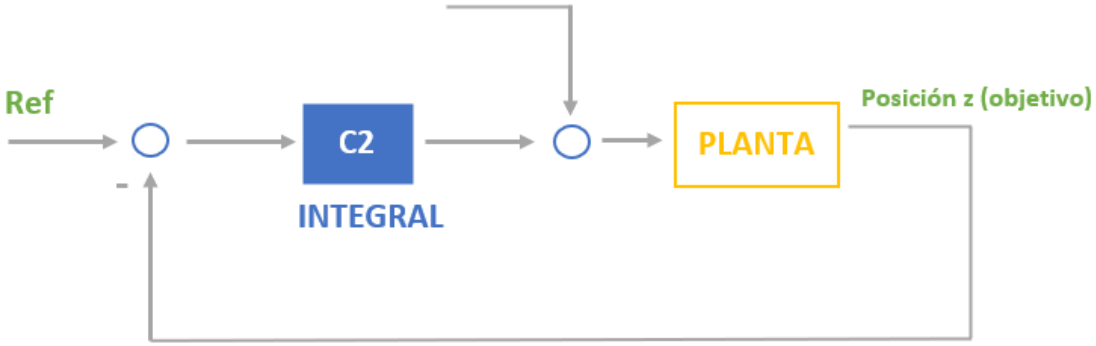
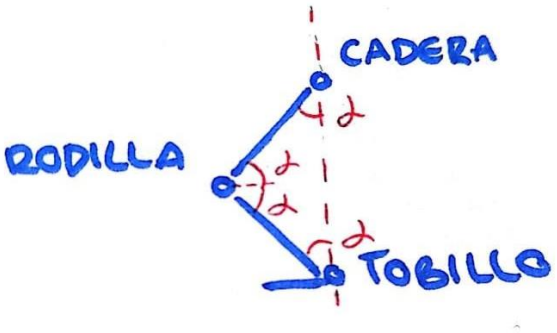


Ilustración 36: Diagrama bloques sistema con controlador por objetivo y referencia

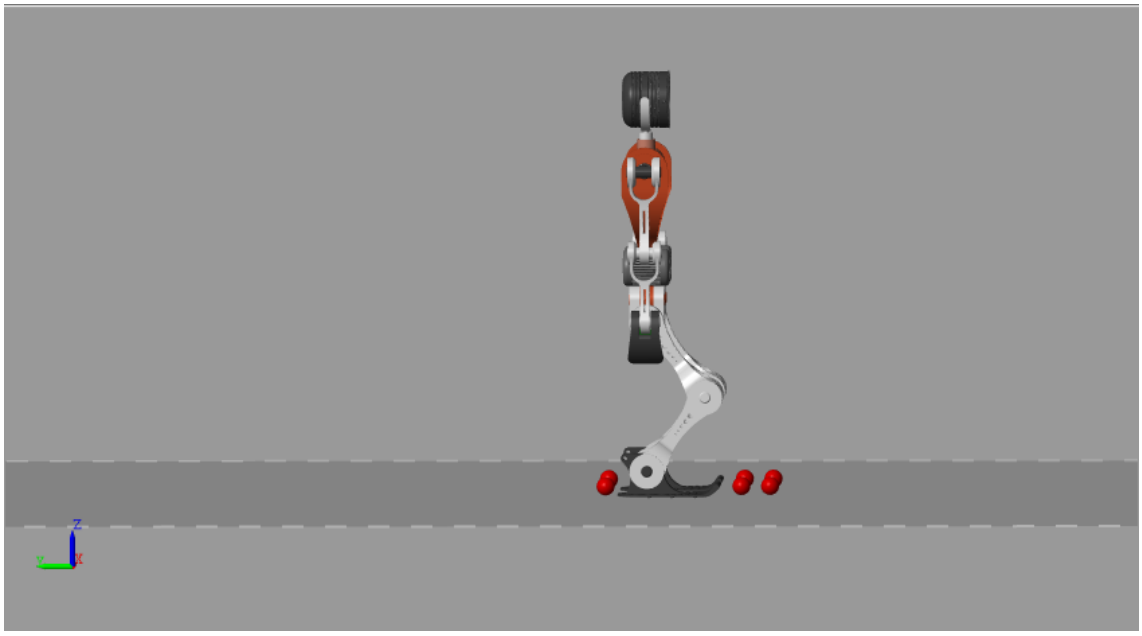
Se ha establecido como condición que cuando el robot cumpla el objetivo, es decir, cuando la posición de la cadera en el eje z llegue a un valor determinado, el usuario realice una acción que en nuestro caso será levantar los brazos.

Para conseguir esto, se ha dado a las articulaciones de las extremidades inferiores diferentes pesos para que se realice el movimiento que desee el usuario. Aquí no pasaremos un valor exacto a la articulación, si no una proporción de lo que tiene que moverse en función del resto de articulaciones, como se observa en la figura.



Tan solo se moverán las articulaciones de las extremidades inferiores ya que son las que nos interesan para que el robot se desplace en el eje z (arriba y abajo porque así se podrá agachar). Se podrían incorporar todas las articulaciones del humanoide, pero hay que tener en cuenta que es muy difícil definir las estrategias de control y que en ocasiones si se establece un lazo de control muy complicado podría perjudicar al sistema.

Ilustración 37: Relación de proporción del ángulo en la extremidad inferior



Simulación 14: Robot con control automático con objetivo

En este caso si comprobamos el contenido del vector tQ no encontramos ninguna indicación de movimiento en las articulaciones. No se ha indicado el ángulo que debe rotar cada articulación por ello el vector de trayectorias es una matriz de ceros. Tampoco observamos ningún valor diferente de cero en la matriz de las coordenadas articulares, ya que no es necesario pasar la posición que debe tomar cada articulación al autómata. El robot realiza de forma "autónoma" el movimiento correspondiente para llegar al objetivo.

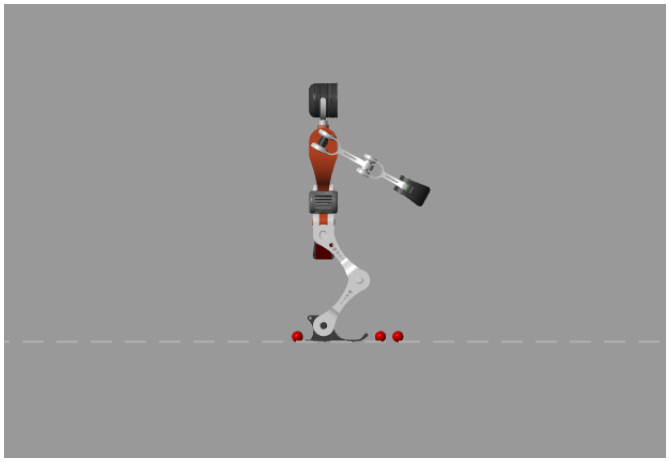
H.tQ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1.0000e-03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1.1120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2.2230	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3.3340	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4.4450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	5.5560	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6.6670	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	7.7780	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	8.8890	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11																			
12																			

Ilustración 38:Vector tQ con controlador automático con objetivo

Sin embargo, que el movimiento sea automático no significa que el robot elija qué articulaciones mover. Por ejemplo, simulamos el mismo ejemplo de antes pero además se ha dado un valor a la articulación del hombro para que se mueva al alcanzar la referencia en el eje z. Este movimiento no tiene sentido, ya que el movimiento del hombro no influye en el objetivo; con esto se puede demostrar que el humanoide no es inteligente para decidir sobre qué articulaciones hay que actuar para llegar al objetivo. Se puede indicar que se muevan otras articulaciones en función de la referencia, pero puede que se

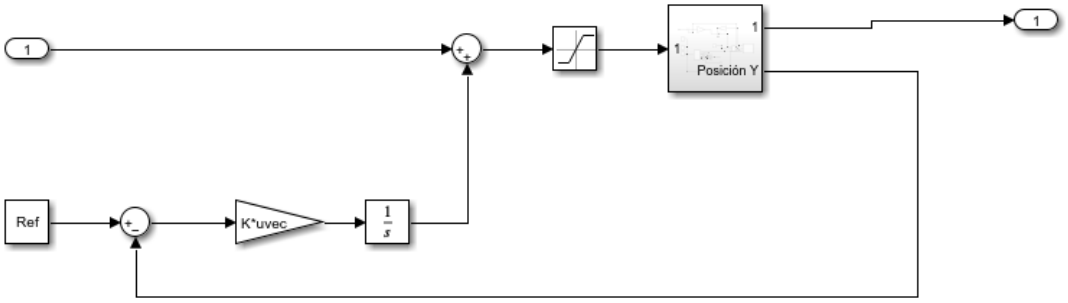
muevan sin sentido, por ello es importante estudiar las articulaciones que realmente pueden minimizar el objetivo.



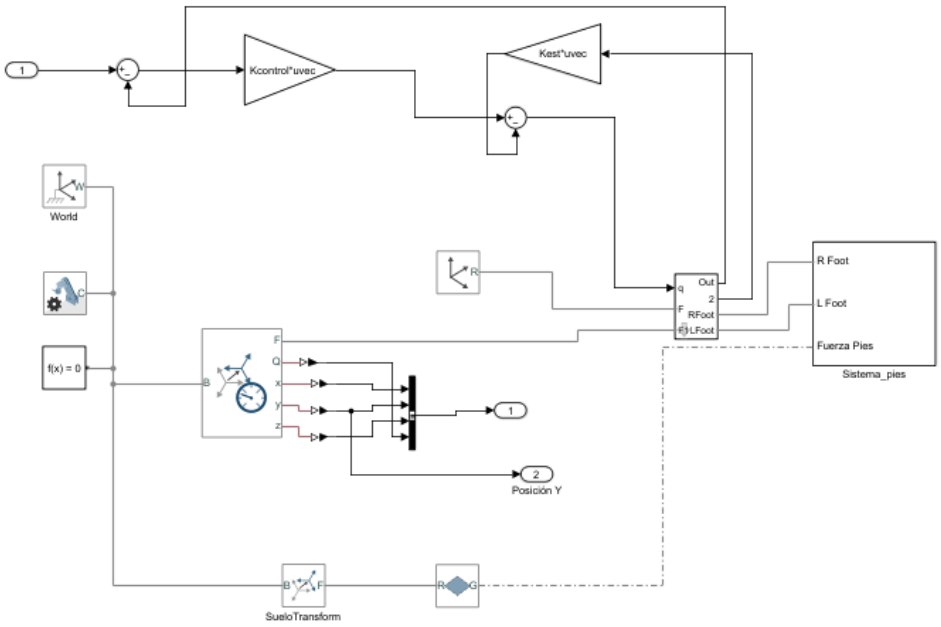
Simulación 15: Robot con controlador automático con objetivo, mueve articulación sin sentido

4.3.6.3 APLICACIÓN UNIVERSAL A HUMANOIDES

Todos los controles expuestos hasta ahora tienen un carácter universal; podrán ser utilizados en cualquiera de los robots estudiados para lograr su control. Basta con modificar la estación de Simulink introduciendo los lazos de control correspondientes. Por ejemplo, controlamos el robot roid empleando un lazo de control por referencia. El objetivo que deberá alcanzar el humanoide será el de la posición en el eje y, que según el sistema de ejes del modelo corresponde, es el que sitúa al robot hacia la derecha o la izquierda.

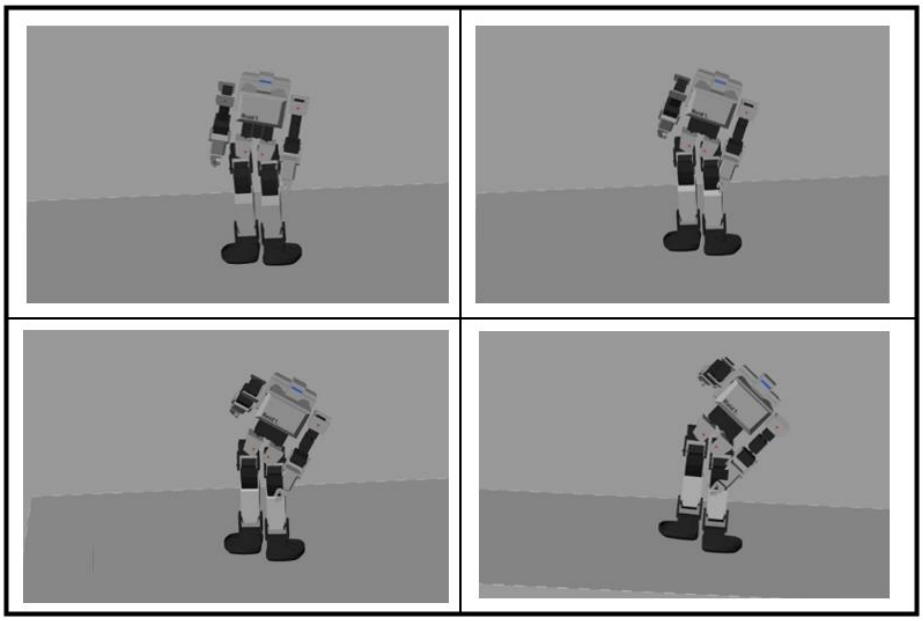


Bloques Simulink 51: Lazo de control automático con objetivo (robot roid1)



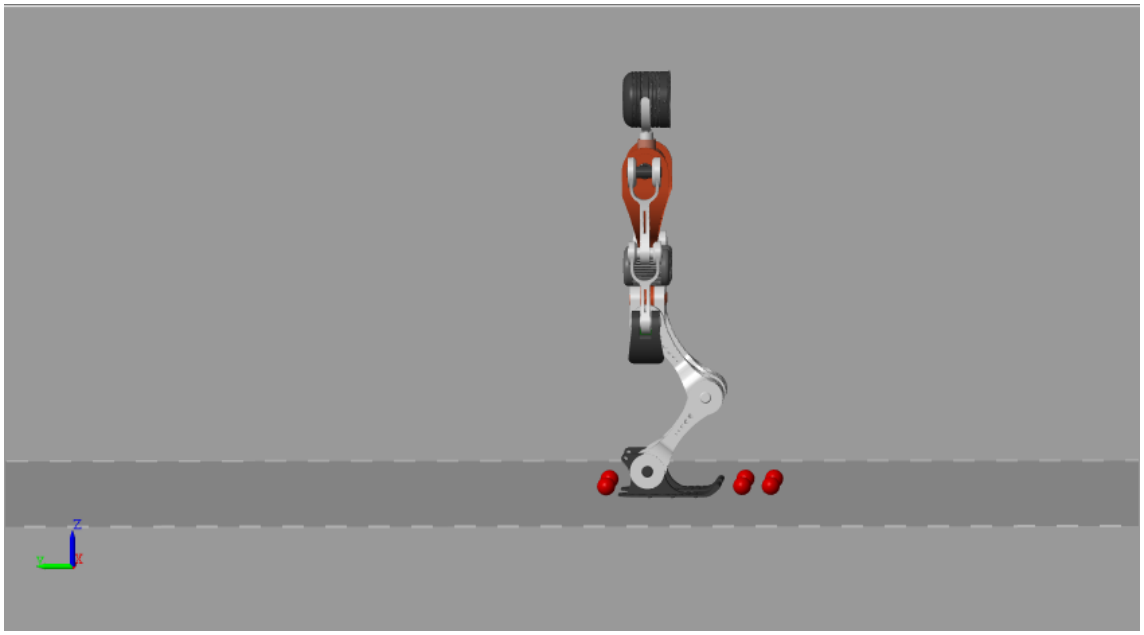
Bloques Simulink 52: Modelo robótico lazo de control automático con objetivo (robot roid1)

Según el sistema de referencia global, al aumentar cada vez más la y, el robot se inclinará hacia la derecha para intentar llegar a la posición indicada por el usuario.

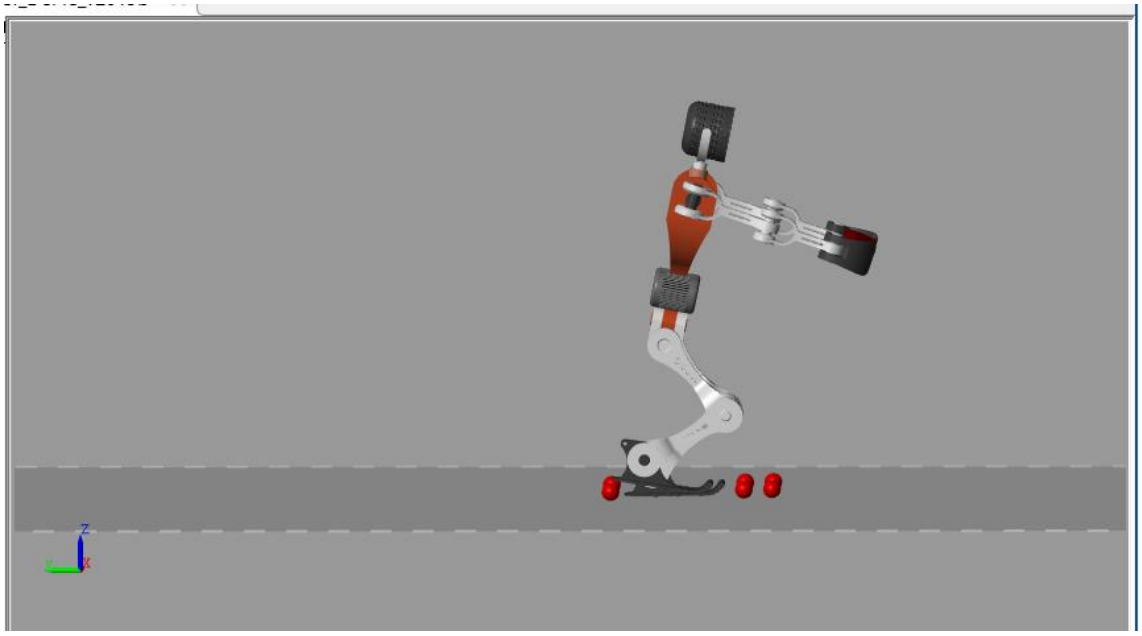


Simulación 16: Robot roid desplazando referencia en eje Y

Este controlador sirve para controlar la estabilidad del modelo robótico completo. Gracias a este controlador, se pueden poner condiciones emulando situaciones que se pueden dar en una persona real. Por ejemplo, podemos establecer una condición en la que cuando el humanoide alcance un valor en el eje z muy bajo (en este ejemplo hemos definido menos de 400) el humanoide extenderá sus brazos para aguantar su cuerpo ante una posible caída. La reacción del humanoide es similar a la de una persona cuando se cae hacia delante; el robot considera que se está cayendo y le mando poner las manos.



*Simulación 17: Robot Humanoide Ref en z (Ref=400*1e^-3)*



*Simulación 18: Robot Humanoide Ref en z (Ref=380*1e^-3)*

Como se puede observar hay diferencia entre los sistemas de control de lazo interno básicos en todos los modelos dinámicos y los lazos externos extras. El control básico permite que las articulaciones del robot realicen movimientos continuos; es responsable de la ejecución de movimientos precisos y el mantenimiento de equilibrio. Se podría comparar la función del control interno con la función del cerebelo, que se encarga de que el sistema muscular esquelético consiga movimientos suaves, uniforme y coordinados. Por otro lado, los lazos externos de control inteligente dan al robot la capacidad de razonar cuál es la mejor opción para que el robot alcance un objetivo. Aquí no se pasa a cada articulación el ángulo que tiene que rotar, solo se indica el punto final y el robot "decide" cómo llegar. Un símil de esto puede ser el cerebro, ya que controla acciones más o menos conscientes que involucran un nivel superior de procesamiento.

Como hemos observado, la incorporación de lazos externos de control inteligente otorga al robot la capacidad de cumplir un objetivo concreto. En el proyecto hemos implementado un ejemplo de control inteligente muy sencillo, creando un sistema de lazos de control anidados con el controlador PD básico. El usuario puede jugar con el modelo e introducir más sistemas de control para que tenga un comportamiento determinado. A pesar de esto, al meter varios lazos a la vez puede suceder que se crucen entre sí y acaben por perjudicar la estabilidad del modelo robótico, por ello es de gran importancia comprender la estructura de control de los robots.

ALGORITMOS DE OPTIMIZACIÓN PARA EL DISEÑO DE CONTROLADORES

Para establecer los sistemas de control en el modelo dinámico se han dado unos valores a las constantes de los controladores de posición y velocidad que se han obtenido por prueba y error. Si establecemos otros valores diferentes a los controladores el movimiento resultante resulta más o menos oscilatorio. Los parámetros de los controladores se podían haber encontrado igualmente siguiendo una estrategia determinada mediante métodos de optimización.

El objetivo de cualquier algoritmo de optimización será encontrar el mejor resultado para un objetivo determinado (reducir las oscilaciones, llegar a una posición, que el sistema no se caiga…). En nuestro ejemplo, lo que se debería hacer es optimizar los parámetros del controlador (K) para alcanzar el objetivo que queramos. De forma resumida podemos encontrar dos tipos de algoritmos para lograr la optimización:

*Algoritmos de optimización local: se parte de un punto inicial o varios puntos (local con varios intentos) y se evalúa alrededor de cada punto la mejor posición. El humanoide de Matlab presenta 18 parámetros en su controlador, por lo que habría que evaluar alrededor del punto en 18 dimensiones diferentes. Una vez que no se encuentre ningún punto más, entonces se considera que se ha llegado al mejor punto que se podría llegar desde el punto de partida.

*Algoritmos de optimización global: se analizan todos los puntos del espacio de trabajo y se evalúa la función en cada uno de los puntos. Aquí se miran los resultados de la optimización de todos los puntos, por lo que para hacerlo con 18 dimensiones tendríamos que trabajar con números de un orden muy elevado para obtener un resultado representativo. Como esto es muy complejo cuando las dimensiones son muy altas se suele acudir a algoritmos evolutivos como por ejemplo los algoritmos genéticos.

Con los algoritmos genéticos se sigue alguna estrategia para evaluar los mejores resultados del espacio de trabajo. Para ello se analiza la función en el espacio de búsqueda, se toman los mejores resultados y la próxima evaluación se realizará con una combinación de los mejores resultados obtenidos. Se está eligiendo a los “hijos” que han dado mejores resultados y no están uniformemente repartidos en el espacio de búsqueda, por lo que no se pierde tiempo en evaluar profundamente las zonas donde los resultados no serán adecuados. La ventaja de estos algoritmos es que optimizan la zona donde hay mejores resultados, pero siguen comprobando en menor medida las zonas con resultados no favorables, por lo que si hay algún punto que nos ofrece los mejores resultados y está aislado también es posible encontrarlo.

Varios autores han aplicado algoritmos genéticos, ya que realiza una criba para quedarse con los mejores resultados y es más rápido que otros métodos de optimización. Un ejemplo de ello es el modelo Train Humanoid Walker [21] de Matlab, que utiliza el modelo humanoide de Matlab y lo entrena empleando algoritmos genéticos y aprendizaje por refuerzo. En el ejemplo se pretende que el humanoide camine sin que se caiga (se minimiza que caiga por debajo de un determinado valor, los movimientos laterales y la rotación del torso). Con el algoritmo se intenta encontrar el patrón de control óptimo para que las articulaciones del humanoide realicen una rotación determinada.

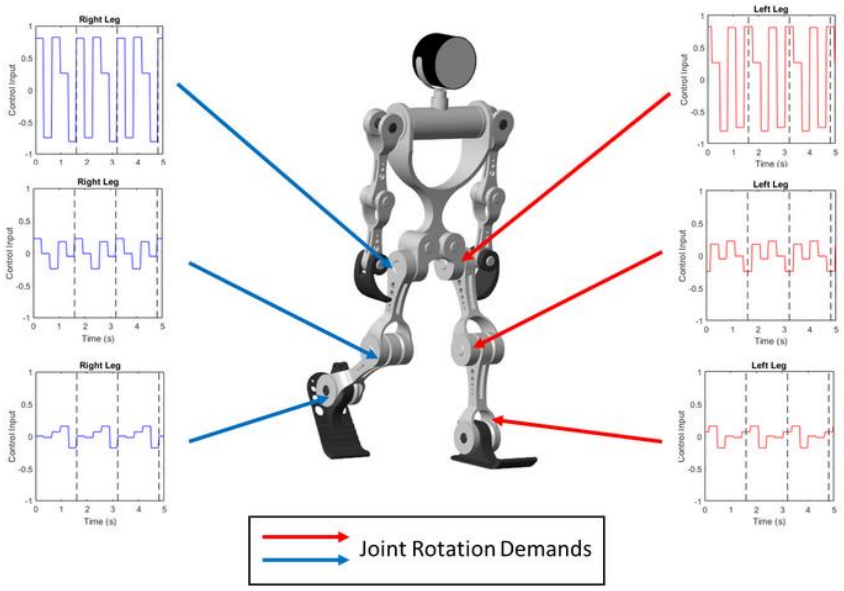


Ilustración 39: Modelo Train Humanoid Walker [21]

El problema es que nuestro modelo emplea 18 lazos en cada controlador, por lo que el proceso de optimización global con algoritmos genéticos también sería muy largo y tedioso.

Como el modelo funciona correctamente con los valores determinados de forma manual (obtenidos por tanteo), no se aplicarán algoritmos de optimización para los controladores.

CAPÍTULO V: APLICACIONES EN LOS MODELOS

EJEMPLO 1: CONTROL HUMANOIDES MEDIANTE INTERFAZ

Una aplicación de las funciones diseñadas en este proyecto es la construcción de una interfaz para que el usuario juegue y comprenda el funcionamiento de diferentes robots humanoides. El objetivo de este instrumento es demostrar la universalidad de los métodos diseñados utilizando una consola para controlar cualquier robot.

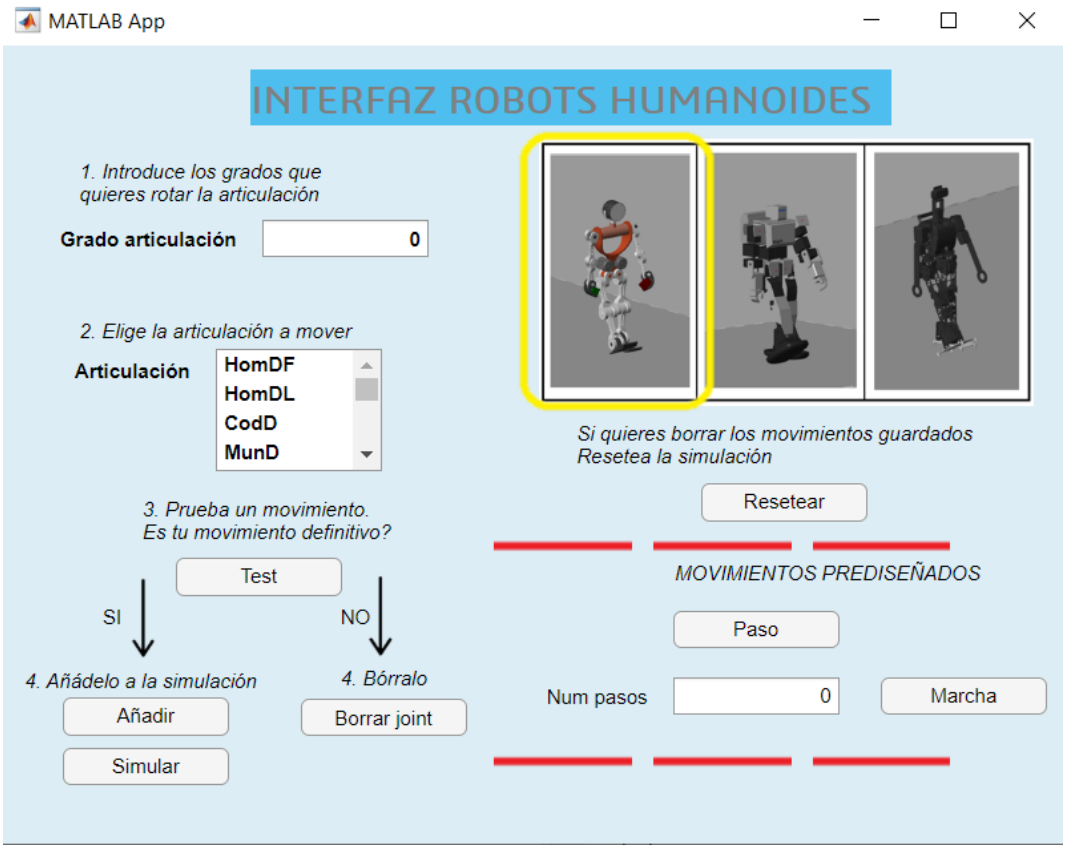
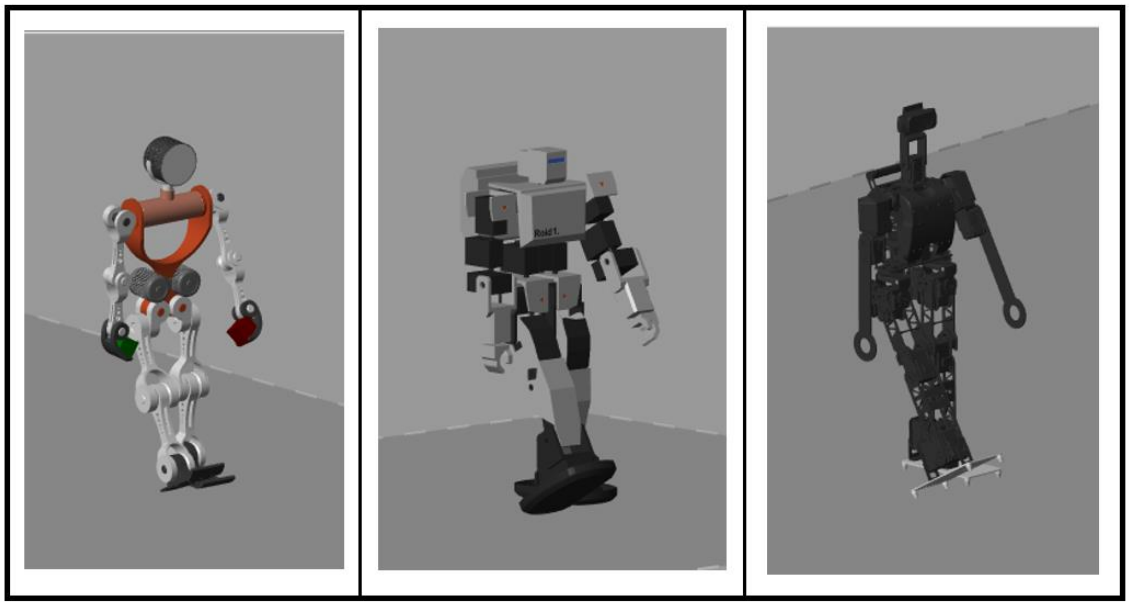


Ilustración 40: Interfaz para Robots Humanoides en AppDesigner

A continuación, se plantea una combinación de movimientos mediante la interfaz y se comprueba el resultado simulado en cada robot. Definimos mediante la consola que el robot lleve su articulación del Hombro Izquierdo Frontal a una posición de 30 grados y la articulación de Cadera Derecha Frontal a una posición de 30 grados igualmente.

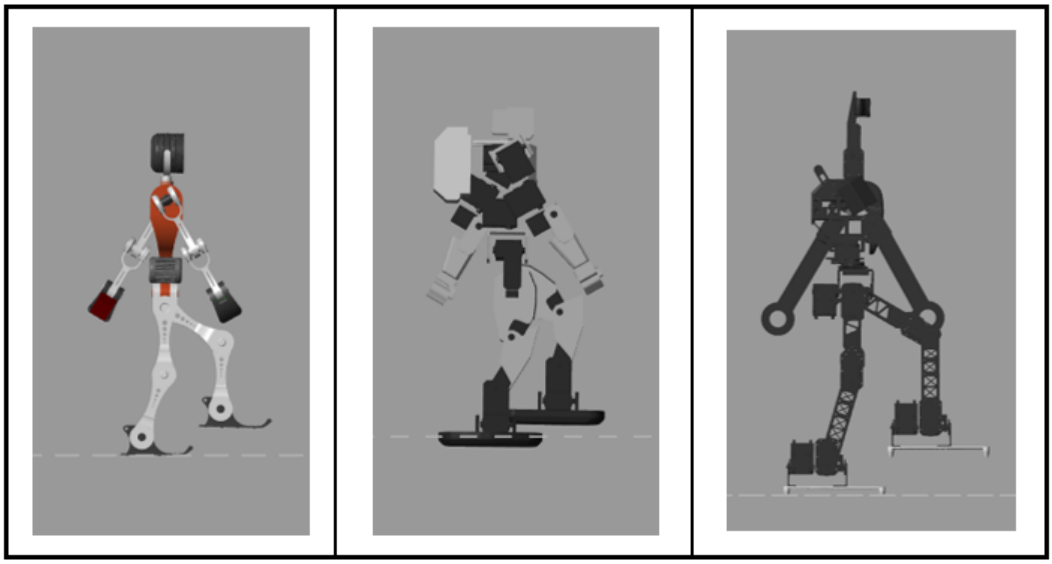


Simulación 19: Movimiento humanoides universal

Como se puede observar, los tres robots ejecutan los movimientos planteados en la articulación adecuada y en el sentido adecuado. Esto es posible gracias al diseño de la configuración articular de los robots y el objeto "eje" que almacena sus articulaciones. En cada uno de los robots se ha determinado el número de articulación y el sentido de rotación de forma que al seleccionar una opción en la ListBox Articulación se mueva la misma articulación y en el mismo sentido independientemente del robot que importemos.

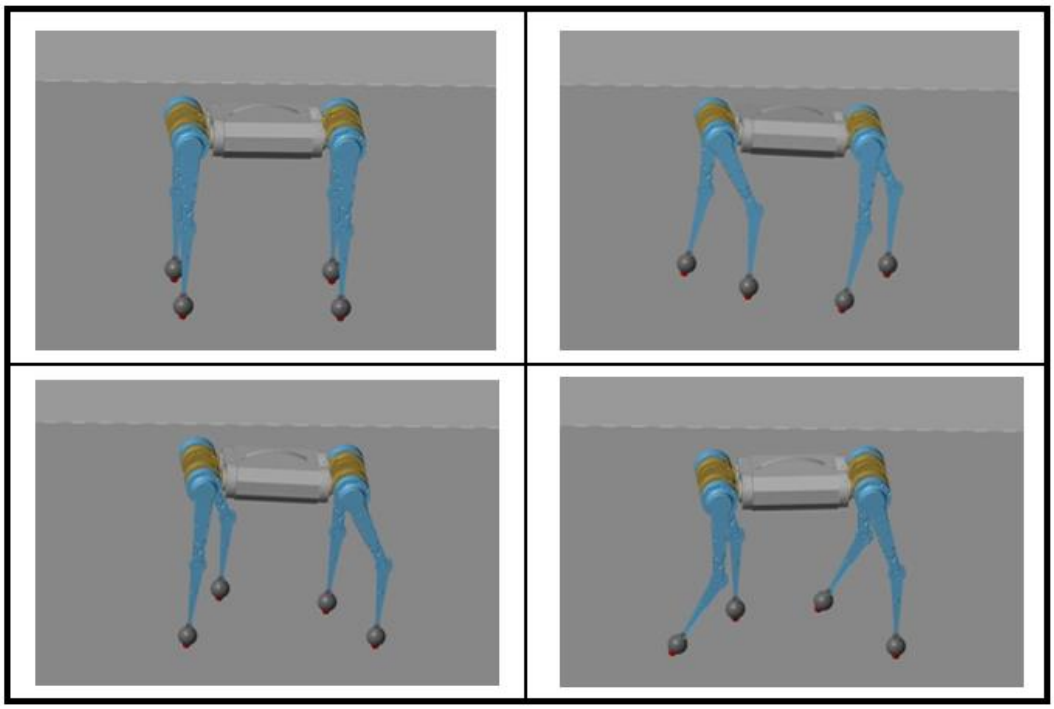
EJEMPLO 2: MOVIMIENTO MARCHA EN LOS MODELOS

Gracias al diseño de la estación y el objeto Hum se han podido aplicar las mismas funciones a los tres modelos humanoides a través de la interfaz. Un ejemplo de esta aplicación universal de los métodos es el movimiento de marcha con la función Marcha() del objeto Hum. La acción de andar se puede conseguir en cualquiera de los robots introduciendo el ángulo correspondiente en las articulaciones de las extremidades inferiores. Como se detalló anteriormente, se ha creado un objeto "eje" que almacena el número de las articulaciones de cada robot, de forma que para que los tres modelos muevan la rodilla es posible que en cada caso se mueva una articulación numerada de forma diferente, pero tendrá el mismo nombre en los tres modelos.



Simulación 20: Movimiento marcha humanoides universal

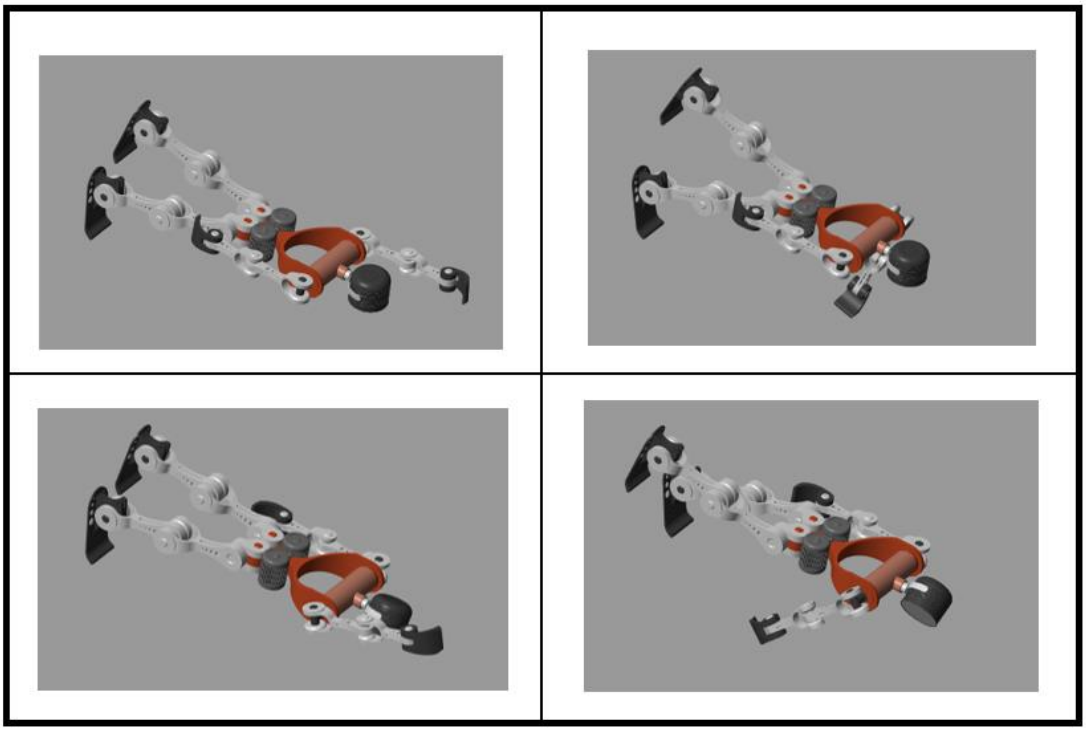
También se puede aplicar el movimiento de marcha en el robot cuadrúpedo, sin embargo, hay que tener en cuenta que la forma de designar cada articulación es diferente en el caso del cuadrúpedo.



Simulación 21: Movimiento marcha cuadrúpedo

EJEMPLLO 3: HUMANOIDE NADANDO

Un objetivo interesante que se ha estudiado al trabajar con humanoides es la estabilidad. Para que el movimiento del modelo humanoide sea lo más parecido posible al de un humano real se ha realizado un estudio de las fuerzas de las esferas que hay en los pies del humanoide, de forma que estuviera sujeto únicamente por los pies. A pesar de esto, es posible mantener al humanoide agarrado por una parte del cuerpo si así se desea. Por ejemplo, si se quiere simular el movimiento de natación del humanoide es posible girar el sistema de referencia global del robot y mantenerlo fijo por la cintura. Con este procedimiento, podremos controlar las extremidades de forma dinámica como en los ejemplos anteriores pero el robot permanecerá toda la simulación fijo en el mismo sitio.



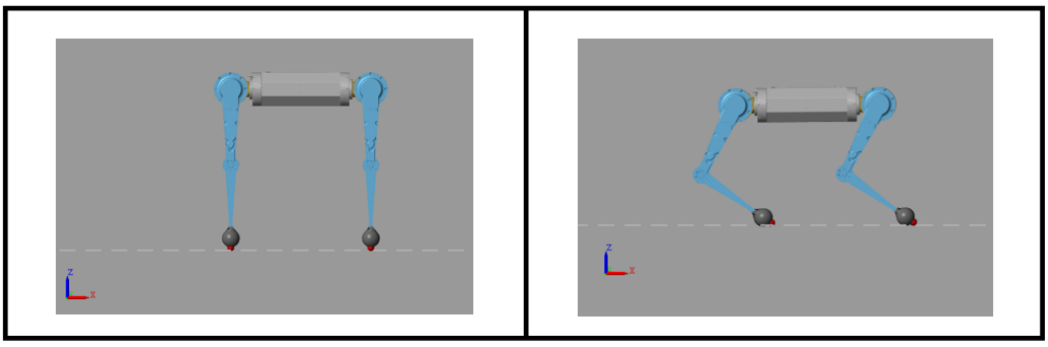
Simulación 22: Humanoide realizando movimiento de natación

EJEMPLO 4: ESTABILIDAD ROBOT CUADRÚPEDO

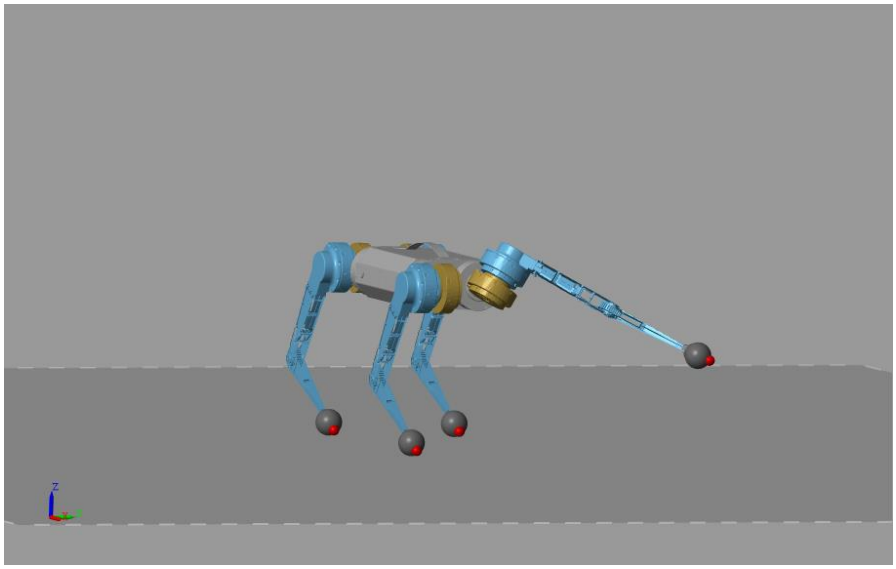
En el proyecto se realiza un estudio dinámico de tres modelos humanoides y uno cuadrúpedo; este se ha enfocado en el análisis de la estabilidad y los movimientos de los modelos antropomórficos. A pesar de ello, el modelo cuadrúpedo tiene un gran potencial, porque es un robot que al presentar 4 puntos de apoyo tiene mucha más estabilidad que los modelos humanoides.

Por ejemplo, podría mantenerse de pie con solo tres extremidades y realizar prácticamente cualquier movimiento con la cuarta extremidad.

Además, es posible cambiar la configuración articular inicial del robot, de forma que se asemeje a los modelos de perros robóticos reales. Para conseguirlo se ha dado un ángulo de 30 grados a las articulaciones superiores (Cadera frontal) y un ángulo de 90 a cada rodilla. Con esto se obtiene una posición inicial más estable.



Simulación 23: Configuraciones articulares iniciales cuadrúpedo



Simulación 24: Robot cuadrúpedo estable con 3 extremidades

EJEMPLO 5: ACCIÓN-REACCIÓN POR CONTROL MEDIANTE POSICIÓN OBJETIVO

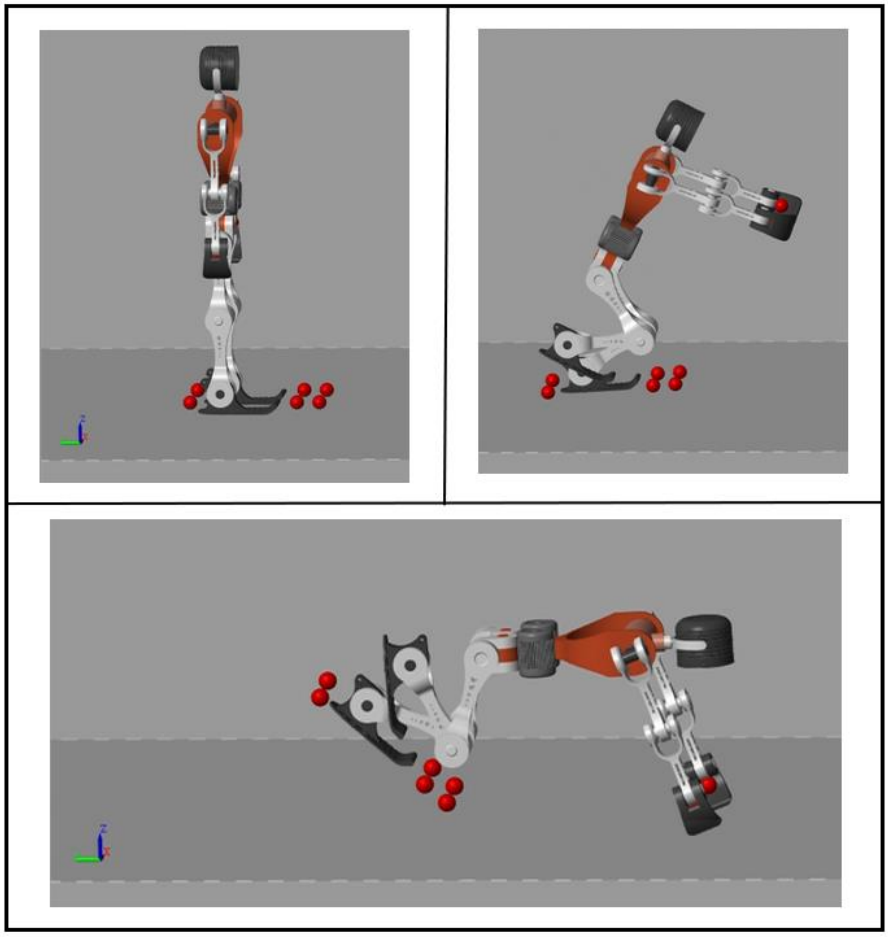
Una aplicación de los lazos de control diseñados en los diferentes modelos robóticos es el de simular los reflejos que "tiene" un modelo humanoide. Se pueden emplear lazos de control para realimentar al robot un objetivo específico y minimizarlo para que simulara el movimiento de los músculos del

cuerpo de forma automática, es decir, para que simule el reflejo de un músculo.

Un ejemplo característico de los humanos es poner los brazos hacia adelante cuando detecta que se va a caer, de forma que se pueda frenar la caída con las manos. Para conseguirlo, se ha puesto como objetivo minimizar la posición de la cadera en el eje z para que se alcance una referencia introducida en el sistema. La referencia que se ha elegido se alcanza cuando el robot realiza un ángulo determinado con las extremidades inferiores con el que se mantiene el equilibrio. Cuando se detecte que la posición de la cadera está lejos de la referencia, por ejemplo, si el robot realiza un ángulo con las piernas que le hace perder el equilibrio, entonces se produce un movimiento automático que activa la rotación de los hombros para extender los brazos hacia adelante.

Además, se han incorporado en las manos del modelo humanoide dos esferas de contacto como las que presenta el robot en los pies. Cuando las esferas de las manos toquen el suelo, la simulación se detendrá habiéndose completado así el movimiento de reflejo por caída.

Otra posible opción sería que cuando las manos del humanoide tocasen el suelo, el sistema dejará de introducir torque a mayores a las articulaciones, de forma que el robot permanece en la última posición alcanzada y a continuación se podrían realizar más movimientos en la simulación. Esto no se ha conseguido porque una vez que los pies del robot dejan de estar en contacto con el suelo, se pierde el control sobre el humanoide, por lo que sería de gran dificultad controlar el par de las articulaciones. Igualmente, para realizar esta opción sería necesario aplicar dinámica inversa para obtener la posición, velocidad y aceleración articular a partir del torque ejercido, por lo que se optó por finalizar la simulación directamente.



Simulación 25: Movimiento automático humanoide al caerse hacia adelante

CAPÍTULO VI: CONCLUSIONES Y LÍNEAS FUTURAS

5.1 CONCLUSIONES

En este informe se han recogido los procedimientos y resultados obtenidos durante el proyecto de fin de grado, para la consecución del título de Grado de Ingeniería Electrónica Industrial y Automática. El objeto último del proyecto es comprender y plantear el modelo dinámico de diferentes robots empleando la herramienta de simulación Simscape.

Durante el desarrollo del proyecto se han empleado diferentes herramientas como Simscape y Robotic Toolbox y se ha comprendido su importancia a la hora de definir modelos robóticos. Gracias a esto, se han desarrollado los modelos cinemático y dinámico de diferentes robots industriales, humanoides y cuadrúpedos y se han analizado y observado sus diferencias, tanto en la estructura del diagrama de bloques de Simulink como en la simulación final.

Los métodos y estaciones desarrolladas se han diseñado con el objetivo de conseguir una universalidad a la hora de controlar diferentes modelos robóticos. Se ha conseguido crear una librería de robots que pueden controlarse empleando funciones estandarizadas. Para conseguirlo se han modificado los sistemas de Simulink pertinentes y se ha creado un objeto Hum para almacenar propiedades como la posición, configuración articular o lista de articulaciones de los diferentes robots en una sola variable.

Igualmente, se han aplicado las diferentes herramientas de la familia MATLAB para diseñar una interfaz con la que el usuario pueda dar sus primeros pasos en el mundo de la robótica.

Por último, se ha establecido un sistema de control mediante lazos anidados para reducir el error estacionario del sistema y minimizar las oscilaciones durante el movimiento. Con esto se ha conseguido una simulación estable y coherente con los resultados deseados.

Todo lo explicado anteriormente se ha podido aplicar a varios ejemplos diferentes que demuestran que es posible controlar robots con estructuras diferentes a través de un mismo método.

Dicho esto, se puede afirmar que se han cumplido los objetivos planteados inicialmente.

5.2 LÍNEAS FUTURAS

Como línea futura se podría realizar una investigación mucho más extensa sobre modelos cuadrúpedos, ya que al ser un modelo muy estable el usuario podría jugar con él sin preocuparse de que la estructura se desestabilizase, como puede pasar fácilmente con los ejemplos humanoides. El hecho de presentar 4 extremidades le da libertad al usuario de probar movimientos con menos probabilidad de fallo, ya que al trabajar con humanoides hay que tener en cuenta el paso que se va a dar o la articulación que se va a rotar para que este no se caiga.

Un posible proyecto de investigación con el robot cuadrúpedo es la simulación de la mula robótica en entornos con superficies diferentes. A modo de sensor de fuerza, se podría situar un indicador (esfera de contacto) en el pie del robot, de forma que cuando toque el suelo mantenga el último torque que se ha aplicado a las articulaciones. Con esto sería posible conocer realmente donde está el suelo en terrenos nos familiares. A pesar de ser una aplicación interesante, las técnicas para conseguirlo son muy complejas y no se ha desarrollado en el proyecto, por lo que se deja como posible línea de investigación en el futuro.

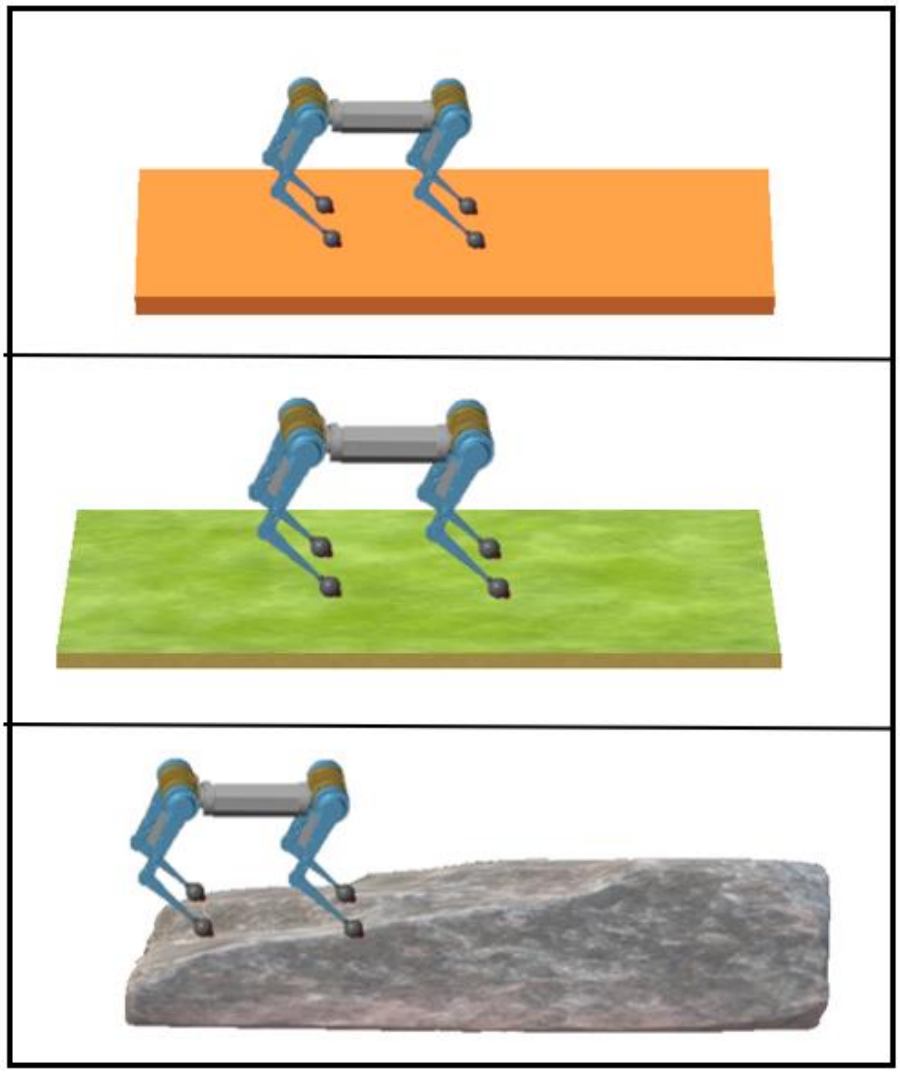


Ilustración 41: Diferencia de contacto entre los pies del cuadrúpedo y el suelo en diferentes superficies

Como se ha mencionado en el trabajo, otra línea futura podría ser la aplicación de la esfera de contacto que se utiliza para sujetar a los humanoides por los pies como parte de un sistema sensor de fuerza. En la actualidad, los robots colaborativos empleados en industrias cuentan con sensores de fuerza en todas las articulaciones, de forma que cuando el actuador (la bola) se activa, los sensores de fuerza se comportan de una manera determinada. Se podría realizar un nuevo proyecto empleando la esfera para recrear un robot colaborativo o cualquier otro dispositivo que precise de un sensor de fuerza en sus articulaciones.

Igualmente, en el ámbito de control se podría llevar a cabo un proyecto mucho más extenso definiendo otros objetivos externos. A esto le podríamos

aplicar algoritmos de optimización para comprobar cómo cambia la estrategia de control en función del algoritmo que se aplique.

Como posible proyecto se podrían establecer nuevos objetivos en el modelo humanoide o cuadrúpedo (minimizar su caída hacia adelante, la distancia entre los pies y el suelo, mejorar estabilidad…) y aplicar un aprendizaje reforzado para alcanzar la meta deseada.

CAPÍTULO VI: ANEXOS

6.1 MÉTODOS CLASE "KIN"

Métodos de Rigid-Body

- Tool (): Permite cambiar de herramienta, o entre cargas y herramientas.

```
rob.Tool(bodyCarga)
```

- Wobj, Body, DelBody: Pueden incluir y quitar cuerpos en el robot Rigid-Body, pero no se aplicarán en este proyecto.

Métodos de visualización

- Show ([Joint], [pose]): Sin argumentos, muestra el robot en la posición actual.

  Con el argumento Joint muestra el robot en la posición articular dada y cambia la posición interna del robot. El argumento pose rastrea la posición del TCP [x y z rx ry rz].

- Rec(value): Recoge los movimientos del robot en una grabación.
- Rep (): Reproduce lo que se ha grabado con Rec().

Métodos de movimientos básicos

- MoveAbsJ (Joint, [nPts]): se mueven las articulaciones del robot en función de las coordenadas articulares que se pasan con el argumento Joint (matriz que contiene las coordenadas articulares). Se puede definir también el número de puntos interpolados entre la posición inicial y la deseada.

```
q= [0,0,0,0,90,0]*deg.
rob.MoveAbsJ(q).
```

- MoveJ (pose, [Hwobj], [nPts]): se mueve el efector final del robot en función del pose que se pasa como argumento (matriz de 6 columnas de forma [x y z rx ry rz]). Para desplazarse el robot elige el camino más sencillo. Podemos definir el argumento Hwobj para que los puntos definidos sean respecto al eje de referencia Hwobj.

```
x1= [0.4000, 0.1330, 0];
x2= [0.1500, -0.3000, 0];
y1= [0.7232, -0.0536, 0.1000];
h= Hmat();
```

```
% Matriz del objeto mesa
Hmesa= h.T3pts([x1; x2; y1]);

punto= [[100,100,0]*1e-3,[0,0,180]*deg];

rob.MoveJ(punto, Hmesa);
```

- MoveL (pose, [Hwobj], [nPts]): se mueve el efector final del robot en función del pose que se pasa como argumento (matriz de 6 columnas de forma [x y z rx ry rz]). Para desplazarse el robot elige el camino más corto (trayectoria lineal). Podemos definir el argumento Hwobj para que los puntos definidos sean respecto al eje de referencia Hwobj.

```
puntos= [[200,300,200]*1e-3, [0,0,180]*deg];
rob.MoveL(puntos);
```

- MoveC (pos1, pos2, [Hwobj], [nPts]): la herramienta dibuja un arco que pasa por los dos puntos que se pasan como argumentos (formato [x y z]). Podemos definir el argumento Hwobj para que los puntos definidos sean respecto al eje de referencia Hwobj.

Métodos Internos

- Pose(joint,[Hwobj]): método que aplica la cinemática directa; tiene como argumento las coordenadas articulares del robot y devuelve la posición de la herramienta con coordenadas cartesianas de tipo [x y z rx ry rz].

- Joint(pose,[w]): método que aplica la cinemática inversa; tiene como argumento las posiciones cartesianas de la herramienta del robot y devuelve las coordenadas articulares en formato q=[q1 q2 q3 q4 … qn].

- Robot (): Devuelve una propiedad del robot que ha podido verse alterada.
- Sim (mdl, type, value): Ordena realizar la simulación. Para ello se pasan como argumentos el fichero Simulink donde está el robot y los objetos, el valor de la velocidad o del tiempo y el tipo de medida (velocidad o tiempo).

6.2 MÉTODOS CLASE “HUM”

- **TestMove():** mueve el robot desde la última posición simulada para comprobar el nuevo movimiento. El usuario podrá utilizar esta función para hacer pruebas con diferentes movimientos del robot antes de añadirlos de forma definitiva a la cadena de movimientos finales. Debemos pasar como argumento el vector joint, compuesto por la articulación que se va a mover y

los grados que se moverá, y el incremento del tiempo (consideramos 1 segundo). Con este comando se simula el movimiento desde la última simulación, por lo que el humanoide comenzará a testear el movimiento desde la última posición que se guardó en la memoria. Esto es de gran utilidad, porque podremos conocer la posición de las articulaciones del robot y también la posición del robot en el entorno en todo momento.
En el código de Matlab se plantea que si la matriz que contiene las rotaciones de las articulaciones, tQ, no está vacía, entonces tomamos los valores de la última fila, que recogen la última posición articular que se ha registrado. Este movimiento lo simularemos mediante la función Sim sin almacenarlo en la memoria.

```
function TestMove(this, joint, tinc)
   deg= pi/180;
   if isempty(this.tQ)
      qi= this.q0(1,:);
   else
      qi= this.tQ(end,2:end);
   end
   t= linspace(0, tinc, this.nPts)';
   Q= repmat(qi, [this.nPts, 1]);
   for i= 1:size(joint,1)
   Q(:, abs(joint(i,1)))= linspace(qi(abs(joint(i,1))), sign(joint(i,1))*joint(i,2)*deg, this.nPts)';
   end
   Sim(this, this.q0(2,:), this.baseRobot(2,:), [t,Q])
end
```

- **AddMove():** cuando el usuario ha comprobado que el movimiento que ha diseñado es definitivo, lo añadirá a la simulación mediante la función AddMove. El movimiento se almacenará en la memoria del objeto Hum. A continuación, se puede pulsar simular para comprobar el resultado de la simulación hasta el momento. Debemos pasar como argumento el vector joint, compuesto por la articulación que se va a mover y los grados que se moverá, y el incremento del tiempo. En el código de Matlab se plantea, al igual que en la función anterior, que si la matriz que contiene las rotaciones de las articulaciones, tQ, no está vacía, entonces tomamos los valores de la última fila, que recogen la última posición articular que se ha registrado. A partir de estos datos guardaremos el nuevo movimiento en tQ.

```
function AddMove(this, joint, tinc)
    deg= pi/180;

   if isempty(this.tQ)
      t0= 0;
      qi= this.q0(1,:);
   else
      t0= this.tQ(end,1);
      qi= this.tQ(end,2:end);
   end
   t= linspace(t0+1e-3, t0+ tinc, this.nPts)';
   Q= repmat(qi, [this.nPts, 1]);
   for i= 1:size(joint,1)
   Q(:, abs(joint(i,1)))= linspace(qi(abs(joint(i,1))),sign(joint(i,1))*joint(i,2)*deg, this.nPts
   end

   this.tQ= [this.tQ; [t,Q]];
end
```

- **AddRef():** es similar a la función AddMove, pero aquí los movimientos de las diferentes articulaciones se guardarán en una estructura de tantas celdas como movimientos realice el robot. Dentro de cada celda se indicarán en una columna el número de la articulación que se mueve y en la segunda columna la posición que se quiere alcanzar con la articulación. Esta función será útil al plantear el sistema en lazo de control automático.

```
function AddRef(this, joint, tinc)
   deg= pi/180;
   if isempty(this.tQ)
      t0= 0;
      qi= this.q0(1,:);
   else
      t0= this.tQ(end,1);
      qi= this.tQ(end,2:end);
   end
   t= t0+ tinc;
   for i= 1:size(joint,1)
       qi(abs(joint(i,1)))= sign(joint(i,1))*joint(i,2)*deg;
   end
   this.tQ= [this.tQ; [t,qi]];
end
```

- **Reset():** inicializa el humanoide al valor inicial sus coordenadas articulares (q0) y la posición del humanoide en el espacio (baseRobot). Permitirá al usuario descartar todos los movimientos guardados en la memoria para comenzar la simulación desde el principio. En el script se ha planteado que cuando se llame a Reset(), la segunda fila de las coordenadas articulares, que contiene las coordenadas articulares actuales, tome el valor de la primera fila,

que será siempre una fila de ceros. Además, se borran los valores del vector tQ.

```
function Reset(this)
   this.q0= repmat(this.q0(1,:),[2,1]);
   this.baseRobot= repmat(this.baseRobot(1,:),[2,1]);
   this.nPts= 1e2;
   this.tQ= [];
end
```

- **Sim():** Simula los elementos guardados. Aquí se hace un test de movimiento desde la primera posición articular guardada en memoria. Al simular el movimiento de una articulación, realizará la rotación que se indique en grados de forma absoluta, no relativa a la última posición de la articulación.

```
function [t, pose]= Sim(this, q0, baseRobot, tQ)

   if nargin== 1 % Caso general
       q0= this.q0(1,:);
       baseRobot= this.baseRobot(1,:);
       tQ= this.tQ;
   end
   assignin('base', 'q0_', q0);
   assignin('base', 'baseRobot_', baseRobot);
   sol= sim(this.mdl, tQ(end,1), [], tQ);
   if nargin==1
      this.q0(2,:)= this.tQ(end,2:end);
      aux= Hmat(); aux.Rad;
      aux.tQ(sol.yout(end,:));
      this.baseRobot(2,:)= aux.tRzyx();
   end
   if nargout > 0
      aux= Hmat(); aux.Rad;
      n= size(sol.yout,1);
      pose= zeros(n, 6);
      for i= 1:n
         aux.tQ(sol.yout(i,:));
         pose(i,:)= aux.tRzyx;
      end
      t= sol.tout;
   end
end
```

De esta forma se podrán extrapolar todas estas funciones empleadas a varios robots humanoides diferentes del robot de Matlab inicial.

6.3 DEFINICIÓN CÓDIGO ROBOTS HUMANOIDES

HUMANOIDE MATLAB

```
%ROBOT HUMANOIDE MATLAB
clear all


deg= pi/180;
baseRobot_= [[0,0,440]*1e-3,[0,0,0]*deg];
q0_= zeros(1,18);
num=1;
Kcontrol= diag(ones(1,18)*1e2);
Estacion= 'Sim30_Humanoide_LA_DePie_v2019b';


% 1: Hombro d frontal
eje.HomDF= 1;
% 2: Hombro d lateral
eje.HomDL= 2;
% 3: Codo d
eje.CodD= 3;
% 4: Muñeca d
eje.MunD= 4;
% 5: Cadera lateral d
eje.CadDL= 5;
% 6: Cadera frontal d (q pie se queda en suelo)
eje.CadDF= -6;
% 7: Rodilla d
eje.RodD= 7;
% 8: Tobillo d
```

```
eje.TobD= 8;
% 9: Hombor i frontal
eje.HomIF= -9;
% 10: Hombro i lateral
eje.HomIL= 10;
% 11:Codo i
eje.CodI= 11;
% 12:Muñeca i
eje.MunI= 12;
% 13:Cadera lateral i
eje.CadIL= 13;
% 14:Cadera frontal i
eje.CadIF= 14;
% 15:Rodilla (negativo)
eje.RodI= -15;
% 16:Tobillo
eje.TobI= 16;
% 17: Cuello frontal
eje.CueIF= 17;
% 18: Cuello lateral
eje.CueIL= 18;
H= Hum(Estacion, eje, baseRobot_, q0_, num);
```

ROID

```
%ROBOT ROID1
clear all
Estacion='roid1_mod.slx';
```

```
num=2;
deg= pi/180;
Kcontrol= diag(ones(1,22)*1e3);
baseRobot_= [[0,0,280]*1e-3,[0,0,0]*deg];
q0_= zeros(1,22);


%1: Cadera d gira
eje.CadDG=1;
%2: Cadera lateral d
eje.CadDL= 2;
% 3: Cadera frontal d (q pie se queda en suelo)
eje.CadDF= -3;
% 4: Rodilla d
eje.RodD= 4;
% 5: Tobillo frontal i
eje.TobD= 5;
% 6: Tobillo lateral i
eje.TobDL= 6;
% 7: Cadera i gira
eje.CadIG=7;
% 8: Cadera lateral i
eje.CadIL= 8;
% 9: Cadera frontal i (q pie se queda en suelo)
eje.CadIF= -9;
% 10: Rodilla i
eje.RodI= 10;
% 11: Tobillo frontal i
eje.TobI= 11;
% 12: Tobillo lateral i
```

```
eje.TobIL= 12;
% 13: Tronco lateral
eje.TronL=13;
% 14: Hombro d frontal
eje.HomDF= -14;
% 15: Hombro d lateral
eje.HomDL= -15;
% 16: Codo gira d
eje.CodDG= 16;
% 17: Codo d
eje.CodD=17;
% 18: Hombro i frontal
eje.HomIF= -18;
% 19: Hombro i lateral
eje.HomIL= 19;
% 20: Codo gira i
eje.CodIG= 20;
% 21: Codo i
eje.CodI=21;
% 22: Cuello
eje.CuelL=22;


H= Hum(Estacion, eje, baseRobot_, q0_, num);
```

ZJUDANCER

```
%ROBOT ZJUDANCER
clear all
```

```
Estacion='zjudancerURDF_mod_2.slx';
num=3;
deg= pi/180;
Kcontrol= diag(ones(1,18)*1e3);
baseRobot_= [[0,0,-520]*1e-3,[0,0,90]*deg];
q0_= zeros(1,18);


%1: Hombro Derecho frontal
eje.HomDF=-1;
%2: Codo derecho
eje.CodD=2;
%3: Cadera derecha gira
eje.CadDG=3;
%4: Cadera derecha lateral
eje.CadDL=4;
%5: Cadera derecha frontal
eje.CadDF=-5;
%6: Rodilla derecha
eje.RodD=6;
%7: Tobillo derecho
eje.TobD=7;
%8: Tobillo derecho lateral
eje.TobDL=8;
%9: Cadera izquierda gira
eje.CadIG=9;
%10: Cadera izquierda lateral
eje.CadIL=10;
%11: Cadera izquierda frontal
eje.CadIF=-11;
```

```
%12: Rodilla izquierda
eje.RodI=12;
%13: Tobillo izquierdo
eje.TobI=13;
%14: Tobillo izquierdo lateral
eje.TobIL=14;
%15: Cuello lateral
eje.CuelL=15;
%16: Cuello frontal
eje.CuelF=16;
%17: Hombro izquierdo frontal
eje.HomIF=-17;
%18: Codo izquierdo frontal
eje.CodI=18;


H= Hum(Estacion, eje, baseRobot_, q0_,num);
```

6.4 CÓDIGO PARA MOVIMIENTO “MARCHA”

HUMANOIDES

```
tinc=1;
%  Mueve la cadera, rodillas de la pierna derecha y los dos brazos
joint1= [H.eje.CadIF, 60; H.eje.RodI, 60; H.eje.HomDF, 30, ; H.eje.HomIF, -30];
H.AddMove(joint1, tinc)
% Mueve la rodilla y tobillo de la pierna derecha
joint2= [H.eje.RodD, 60; H.eje.TobD, -60];
H.AddMove(joint2, tinc)
% Pone la cadera y rodilla derecha en su posición inicial
joint3= [H.eje.CadIF, 0; H.eje.RodI, 0];
H.AddMove(joint3, tinc)
% Pone la rodilla y tobillo derecho en su posición inicial
```

```
joint4= [H.eje.RodD, 0; H.eje.TobD, 0; H.eje.HomDF, 30; H.eje.HomIF, 0];
H.AddMove(joint4, tinc)
[t, pose]= H.Sim;
```

CUADRÚPEDOS

```
tinc=1;
joint1=[H.eje.FLCadF, -30; H.eje.RRCadF,-30; H.eje.FLRod,30;
H.eje.RRRod,30];
H.AddMove(joint1, tinc)
joint2=[H.eje.FRRod,30;H.eje.RLRod,30];
H.AddMove(joint2, tinc)
joint3=[H.eje.FLCadF, 0; H.eje.RRCadF,0;H.eje.FLRod,0;H.eje.RRRod,0];
H.AddMove(joint3, tinc)
joint4=[H.eje.FRRod,0;H.eje.RLRod,0];
H.AddMove(joint4, tinc)
H.Sim
```

BIBLIOGRAFÍA

[1] «Robotic System Toolbox,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/products/robotics.html. [Último acceso: mayo 2022].

[2] M. Á. M. S. José y A. Herreros López, «Simulación, control cinemático y dinámico de robots comerciales usando la herramienta de MatLab Robotic Toolbox,» TFG, Valladolid, 2014.

[3] C. J. Jiménez y A. Herreros López, «Diseño de un sistema robótico educativo para jugar al ajedrez con robots industriales,» TFM, Valladolid, 2019.

[4] S. A. Fernández y A. Herreros López, «Aplicación de Simulink y MatLab para análisis cinemático y dinámico, control y comunicaciones de robots industriales,» TFG, Valladolid, 2020.

[5] Ollydbg, «Classic DH parameters,» Wikimedia, 2014 marzo 21. [En línea]. Available: https://commons.wikimedia.org/wiki/File:Classic-DHparameters.png.

[6] G. Walck, «Introduction to Robot Modeling in ROS,» CITEC, Universitat Bielefeld, 2015.

[7] P. Corke, «Robotics, Vision and Control. Fundamental Algorithms in MATLAB,» So, 2016. [En línea]. Available: doi: 10.1007/978-3-319-54413-7.

[8] í. Gútiez, «Programación Siemens. Control temperatura PID en Step 7,» Bilbao, 2014.

[9] «Control System Designer (SISO),» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/help/control/ref/controlsystemdesigner-app.html. [Último acceso: mayo 2022].

[10] S. Hosseini, «How rotary encoder works?,» ElectroPeak, [En línea]. Available: https://electropeak.com/learn/rotary-encoder-how-it-works-how-to-use-with-arduino/. [Último acceso: mayo 2022].

[11] S. A. C. Giraldo, «Acción de control derivativo PID,» Control Automático Educación, 2020. [En línea]. Available: https://controlautomaticoeducacion.com/control-realimentado/accion-de-control-derivativo-control-pid/. [Último acceso: mayo 2022].

[12] «MATLAB,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/products/matlab.html. [Último acceso: abril 2022].

[13] «SIMULINK,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/products/simulink.html. [Último acceso: abril 2022].

[14] «Simscape Multibody,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/products/simulink.html. [Último acceso: mayo 2022].

[15] «Newtons third law,» Khan Academy, 2019. [En línea]. Available: https://es.khanacademy.org/science/physics/forces-newtons-laws/newtons-laws-of-motion/a/what-is-newtons-third-law. [Último acceso: 2022 abril].

[16] «Estimating Bouncing Ball Contact Parameters,» Software Mathworks, versión 2021b. [En línea]. Available: https://www.matlabcoding.com/2020/07/estimating-bouncing-ball-contact.html. [Último acceso: mayo 2022].

[17] ISO, «Robots and robotic devices – Collaborative robots. Technical specification, International Organization for Standardization,» 2016.

[18] «Contact Proxies,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/help/physmod/sm/ug/use-contact-proxies.html. [Último acceso: julio 2022].

[19] «Humanoid robots,» GitHub, [En línea]. Available: https://github.com/. [Último acceso: 2022 mayo].

[20] A. Legasa, «Tecnalia presenta un robot con visión inteligente para zonas industriales de difícil acceso,» *Crónica Vasca,* 14 junio 2022.

[21] «Train Humanoid Walker,» Software Mathworks, versión 2021b. [En línea]. Available: https://es.mathworks.com/help/deeplearning/ug/humanoid_walker.html. [Último acceso: 2022 julio].

[22] «RoboCup Humanoid League,» 2019. [En línea]. Available: https://humanoid.robocup.org/.

[23] «Feedback Control Architectures,» Software Mathworks, versión 2021b. [En línea]. [Último acceso: mayo 2022].

[24] Ninagawa, «Robot Roid1,» [En línea]. Available: https://note.com/ninagawa123/n/n2d51c3443d31. [Último acceso: julio 2022].

[25] A. M.Shelat, «MedLine Plus,» NIH, [En línea]. Available: https://medlineplus.gov/spanish/ency/esp_imagepages/18008.htm. [Último acceso: julio 2022].