ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, CONTROL E IMPLEMENTACIÓN DE SISTEMAS ROBÓTICOS AVANZADOS

DISEÑO Y SIMULACIÓN DE UN CONTROL DE SEGUIMIENTO DE TRAYECTORIAS PARA UN ROBOT PARALELO VIRTUAL

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

EDWIN PATRICIO YANCHAPANTA DELGADO

edwin.yanchapanta@epn.edu.ec

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

patricio.cruz@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, EDWIN PATRICIO YANCHAPANTA DELGADO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

EDWIN PATRICIO YANCHAPANTA DELGADO

Certifico que el presente trabajo de integración curricular fue desarrollado por EDWIN PATRICIO YANCHAPANTA DELGADO, bajo mi supervisión.

Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

EDWIN PATRICIO YANCHAPANTA DELGADO

Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

Ing. KLEBER REYNALDO VICENTE ROMERO.

DEDICATORIA

A mis padres Ángel Yanchapanta y María Delgado por representar es apoyo incondicional durante mi vida estudiantil, así como también a mis hermanos ya que siempre me brindaron tanto apoyo emocional como económico.

A Dios, por representar una figura simbólica en mi vida para no rendirme en los momentos más difíciles, por darme esperanza en momentos de enfermedad de mi padre y brindarle salud nuevamente y por representar una luz al final todo el trabajo realizado.

A mis amigos que conocí en la Escuela Politécnica Nacional, por estar presente durante mi vida estudiantil y compartir sus horas de estudio y entretenimiento.

AGRADECIMIENTO

Agradecimiento a mi director del trabajo de integración Ing. Patricio Cruz PhD. por brindarnos consejos para la finalización de este trabajo y también por suplir las dudas forjadas durante el desarrollo de este.

A los integrantes y colaboradores del Proyecto PIGR-20-01, en especial a Jeampool y Kleber por brindarnos guía técnica en el manejo de los documentos ya realizados en el marco de este proyecto.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo se enfoca en el control de trayectoria del robot paralelo 3UPS+1RPU empleado dentro del Proyecto de Investigación PIGR-20-01. Este proyecto se basa en el movimiento de la plataforma móvil del robot paralelo de manera que pueda ser utilizada para la generación de ejercicios de rehabilitación. Para este fin, se implementa este robot paralelo, pero de manera virtual en el entorno de simulación del software CoppeliaSim, para lo cual se emplea el modelo cinemático inverso desarrollado en trabajos anteriores relacionados con el Proyecto de Investigación. Se utiliza un esquema de control en cascada conformado por dos bloques fundamentales, el interno constituido por los controladores de posición de cada de los actuadores lineales pertenecientes a los respectivos eslabones, mientras que, el externo es el controlador de trayectoria. Adicionalmente, el uso de los softwares CoppeliaSim y Matlab es parte fundamental del presente trabajo de titulación. Por un lado, en Matlab-Simulink se implementa el esquema de control de trayectoria, mientras que por otro lado está CoppeliaSim, donde se cuenta con el modelo virtual del robot paralelo 3RPU+1UPS y que gracias a la co-simulación entre estos dos softwares, se pueda ejecutar y comprobar el desempeño de la arquitectura de control.

PALABRAS CLAVE: Co-simulación, Robot Paralelo Virtual, Control de Trayectoria, CoopeliaSim, MATLAB.

ABSTRACT

This work focuses on the trajectory control for the parallel robot 3UPS + 1RPU that is employed in the internal Research Project PIGR-20-01. This project is based on the movement of the mobile platform of the parallel robot, so it is used for the generation of rehabilitation exercises. For this aim, the virtual model of the parallel robot is implemented in the simulation environment of CoppeliaSim. To accomplish this goal, the inverse kinematic model of the robot developed in previous works related with the research project is used. A cascade control scheme is analyzed and implemented which is made up of two fundamental blocks, the internal involves the position controllers of each linear actuator that belongs to their respective links, while the external block is the trajectory controller. In addition, the use of CoppeliaSim and Matlab is a fundamental part of this work. The trajectory control scheme is implemented in Matlab-Simulink, while by connecting to the virtual model of the 3RPU+1UPS robot in CoppeliaSim it is possible to execute and test the performance of the control architecture.

KEYWORDS: Co-simulation, Virtual Parallel Robot, Trajectory Tracking, CoopeliaSim, Matlab

1 INTRODUCCIÓN

El área de la robótica aplicada a la medicina ha crecido en los últimos años, en gran parte, debido a los avances tecnológicos, como nuevos sensores y procesadores más rápidos, por tal razón ya existen robots especializados en cirugía, prótesis y rehabilitación [1]. En este último campo se ha logrado la creación de robots para rehabilitación de columna, extremidades superiores e inferiores[2]. La rehabilitación física consiste en la recreación de ciertos movimientos para el aceleramiento de la recuperación de un paciente; centrándose en un tema en específico; como en la rehabilitación de rodilla, los ejercicios se concentran en el movimiento de la extremidad afectada [3]. Por esto, la rehabilitación de rodilla puede ser replicada por un robot con un adecuado controlador que tome en cuenta los ejes en donde se puede mover, así como también el rango en ángulo del movimiento dependiendo del tipo de ejercicio de rehabilitación [4], para lo cual, previo a una implementación en un robot físico, tanto su estructura como sus controladores deben ser simulados y probados, ya que se trabaja con articulaciones principales dentro del funcionamiento motriz de una persona [2].

El presente trabajo de titulación se encuentra dentro del marco del Proyecto de Investigación Grupal PIGR-20-01 “Generación y seguimiento de trayectorias óptimas basado en el desarrollo de controladores avanzados para un robot paralelo enfocado a la diagnosis y/o rehabilitación de pacientes con lesiones de rodilla ”, en el cual ya se han desarrollado modelos dinámicos y cinemáticos del mismo [5] y donde se plantea el diseño de un robot de cuatro patas adheridas en una plataforma fija y una plataforma móvil conocida como robot paralelo 3UPS+RPU, que tiene aplicaciones en la recreación de movimientos necesarios dentro de una sesión de rehabilitación de rodilla.

Para la recreación de una trayectoria en el robot paralelo 3UPS+RPU es necesario un control de trayectoria, mismo que puede ser desarrollado a través de distintos medios tanto en su fase de simulación como de implementación de los controladores, como es el caso del uso de los softwares CoppeliaSim y Matlab. Por un lado, Matlab permite simular el controlador creado y sintonizarlo, mientras que, el software CoppeliaSim otorga una interfaz para simular los movimientos del robot, gracias a su amplia librería de articulaciones y configuraciones de los objetos creados. El uso de Matlab se vuelve esencial debido a sus características y facilidad para el manejo de matrices, por otro lado el software de CoppeliaSim se lo puede encontrar en una versión gratuita, esto aventaja su uso ya que se puede encontrar gran cantidad de tutoriales en su página oficial [6], y de

igual manera cuenta con las librerías necesarias para utilizarlo con otros programas, en este caso Matlab.

El robot paralelo 3UPS+RPU ya ha sido estudiado en trabajos anteriores realizados, especialmente, en la facultad de Ingeniería Mecánica de la Escuela Politécnica Nacional. Por ejemplo, en [7] se analiza el modelo cinemático inverso y directo del robot, sin embargo, no se profundiza en el tema de los controladores ya que solo se limitan al estudio y desarrollo del modelo cinemático con una simulación del modelo en Matlab. Adicionalmente, en [8] se analiza el diseño de este mismo robot ya enfocado en el aspecto de simulación, pero este trabajo únicamente se enfoca en la parte mecánica y morfología del robot 3UPS+RPU y para las simulaciones toma en cuenta modelos obtenidos en anteriores trabajos, detallando además los tipos de articulaciones que se utilizarían en el modelo físico llegando a implementar la plataforma real sin llegar a usar controladores específicos en sus actuadores. Estos dos trabajos mencionados abarcan mayormente aspectos mecánicos del robot. Por otro lado, en la universidad Politécnica de Valencia, con quienes se está colaborando dentro del Proyecto PIGR-20-01, también han venido trabajando con un robot 3UPS+RPU [9], donde abordan el modelo matemático para el desarrollo de controladores de posición y al igual que este trabajo, realizan simulaciones en Matlab y CoppeliaSim; sin embargo, se utiliza una configuración de los actuadores del robot distinta a la planeada para este trabajo de titulación, sin contar con que para este trabajo se busca desarrollar un controlador de trayectoria.

1.1 OBJETIVO GENERAL

Diseñar un controlador de trayectoria para el robot paralelo 3UPS+RPU del proyecto grupal PIGR-20-01 y simularlo en el entorno de simulación de CoppeliaSim.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar la revisión bibliográfica con respecto al robot paralelo 3 UPS+RPS, modelos dinámicos, cinemáticos y controladores propuestos dentro del proyecto PIGR-20-01, de igual manera del software de simulación de robots CoppeliaSim, a ser utilizado.
- Implementar el robot virtual 3UPS+RPU en el simulador CoppeliaSim y su interconexión con el software Matlab por medio de una API remota.

- Diseñar un controlador de posición para cada eslabón del robot 3UPS+RPU que formaran parte del bloque interno de un control en cascada, cuyo bloque externo corresponderá a un controlador de trayectoria para la plataforma móvil del robot.
- Simular el esquema de control en el entorno CoppeliaSim - Matlab y analizar los resultados obtenidos.

1.3 ALCANCE

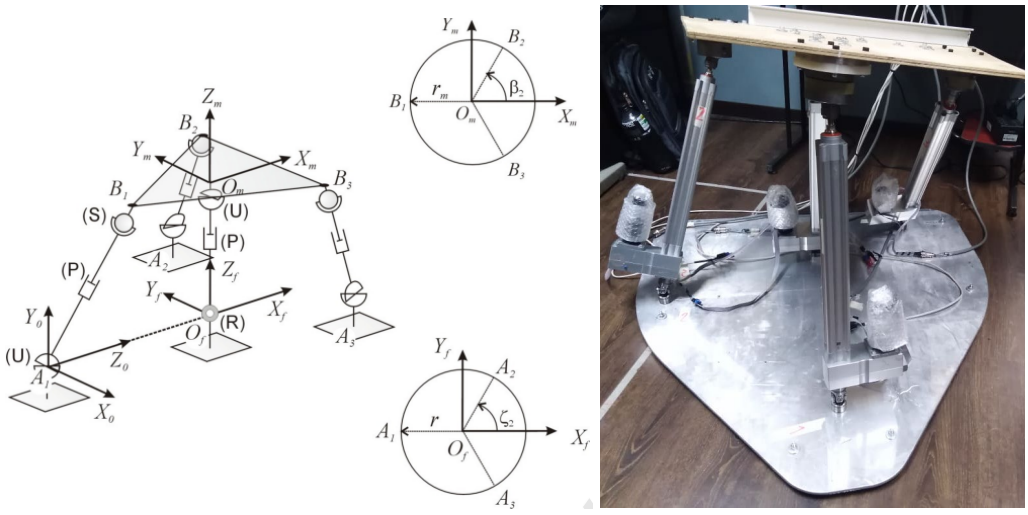
- Se realiza una revisión bibliográfica del robot 3UPS+RPU desarrollado en la Escuela Politécnica Nacional y empleado en el proyecto PIGR-20-01, con el motivo de estudiar y comprender su modelo cinemático, dinámico y su correspondiente estructura física.
- Se revisa información acerca del desarrolló de controladores en el software Matlab-Simulink por medio de bloques, de tal forma que se puedan separar bloques como los de referencia, control y modelo del robot 3UPS+RPU
- Se realiza una revisión bibliográfica sobre el uso del simulador CoppeliaSim y su comunicación con Matlab.
- Se implementa un robot paralelo virtual 3UPS+RPU en el entorno de simulación de CoppeliaSim para una ubicación establecida de cada uno de sus eslabones.
- Se diseña un controlador de posición tipo PID para cada una de las patas, como parte de la estructura del esquema en cascada del control de trayectoria, basados en el modelo del robot revisado.
- Se desarrolla el controlador de trayectoria, completando de esta manera el esquema de control para la plataforma móvil del robot.
- Se implementa del esquema de control de trayectoria en Simulink, primero el bloque de los controladores de posición para su respectivo eslabón y posteriormente, el bloque de control de trayectoria.
- Se simula el esquema de control de la trayectoria en Simulink con comunicación a CoppeliaSim donde se observará el funcionamiento en el modelo CAD implementado.
- Se realizan pruebas del controlador diseñado para posteriormente analizar los resultados obtenidos.

1.4 MARCO TEÓRICO

Un robot paralelo es aquel que consta de una plataforma móvil y una fija unidos por medio de cadenas cinemáticas abiertas, también denominados eslabones o patas, donde la carga de la plataforma móvil se distribuye sobre cada una de las patas del robot [10]. En el caso de este trabajo, se enfocará en el robot Paralelo 3UPS + RPU, robot que se viene desarrollando dentro del proyecto de investigación PIGR-20-01.

1.4.1 ROBOT PARALELO 3 UPS+ RPU

El modelo del robot diseñado en trabajos anteriores cuenta con 4 grados de libertad los cuales son necesarios para cumplir con tareas básicas de rehabilitación. El nombre del robot viene dado por el número de patas y el tipo de articulaciones que une a cada una de estas, es así que este robot cuenta con 3 patas con articulaciones UPS, que significa que cada una de estas posee una articulación universal, una prismática y una esférica. También cuenta con una pata central que posee las articulaciones RPU, lo cual indica que la pata se encuentra conformada por una articulación de revolución, una prismática y una universal; esta configuración del robot mencionado se la puede visualizar en la Figura 1.1. La característica medular del robot es la presencia de dos placas, una fija y otra móvil, en cada una de estas placas se tiene una distribución de patas las cuales ya fueron abordadas en trabajos anteriores, en la Escuela Politécnica Nacional y en la Universidad Politécnica de Valencia [9], [11], donde ya se han visto y analizado las singularidades que presenta cada una de estas configuraciones.



a) **b)**

Figura 1.1. a) Robot 3UPS +RPU para rehabilitación de rodilla [12], **b)** Robot 3UPS+1RPU del proyecto PIGR-20-01.

Como se observa en la Figura 1.1 las articulaciones que se encuentran motorizadas, es decir tienen un actuador, son las articulaciones prismáticas, debido a que presentan un movimiento lineal, haciendo más sencillo ejecutar un control sobre ellos. Estas articulaciones motorizadas son denominadas activas, mientras que las otras articulaciones como las de revolución, esféricas y universales son denominadas articulaciones pasivas dado que dependen de las articulaciones activas para moverse.

La configuración establecida para el robot en este trabajo se muestra en la Figura 1.2, en la cual 3 de los 4 eslabones están ubicados en la plataforma fija en un radio de 0.4 metros y un eslabón está ubicado a 0.15 metros del centro de la plataforma. En la plataforma móvil 3 de los 4 eslabones se encuentran ubicados en un radio de 0.3 metros del centro, mientras que el cuarto eslabón se encuentra ubicada en el centro de la plataforma, a este último se le denomina eslabón o pata central.

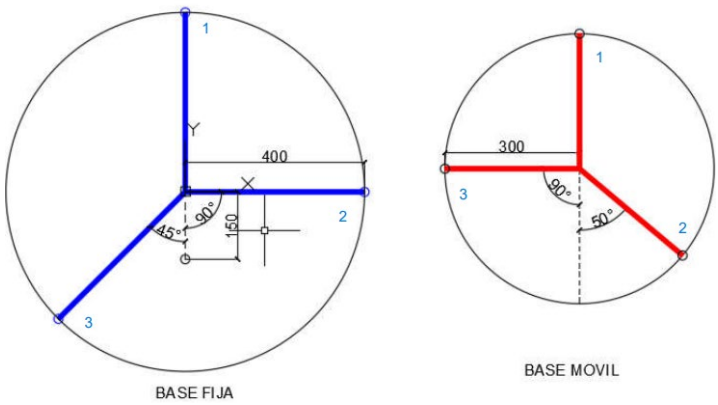


Figura 1.2. Configuración y distribución de los eslabones para la plataforma fija y móvil respectivamente.

En este trabajo se nombran los eslabones del robot de acuerdo a la numeración mostrada en Figura 1.2, es decir se hará referencia a cada uno por su numeración, por ejemplo, la pata central será la pata 4, de igual manera las patas UPR 1,2,3.

1.4.1.1 Articulaciones

Las articulaciones en un robot son de suma importancia puesto que estos son los encargados de limitar o permitir su movimiento. Los tipos de articulaciones usados en el robot paralelo 3 UPS + RPU se presenta de forma gráfica en la Figura 1.3 y se resume a continuación:

- Articulación de revolución o cilíndrica: este tipo de articulación es la más básica que podemos encontrar, este permite un solo grado de libertad en su movimiento, esta

articulación se encuentra en el eslabón 4 denominada RPU la cual cumple se encuentra uniendo este eslabón con la plataforma fija.

- Articulación universal: esta articulación es la unión de dos articulaciones de revolución, esto permite dos grados de libertad al movimiento de los eslabones UPS.

- Articulación esférica: es la articulación con más grados de libertad ya que se puede mover en varias direcciones con un ángulo limitado de movimiento, como su nombre lo indica, se encuentra conformada por una esfera que es la responsable de unir los eslabones con la plataforma móvil.

- Articulación prismática: esta articulación solo permite el movimiento de sus eslabones en un solo sentido, es decir únicamente cuenta con un grado de libertad en sus movimientos, es por esto que es la articulación que posee el movimiento más lineal de los nombrados y que se utilizó para la conformación del robot paralelo 3 UPS + RPU, este es representado en el modelo físico por el cilindro mostrado en la Figura 1.4. (a).

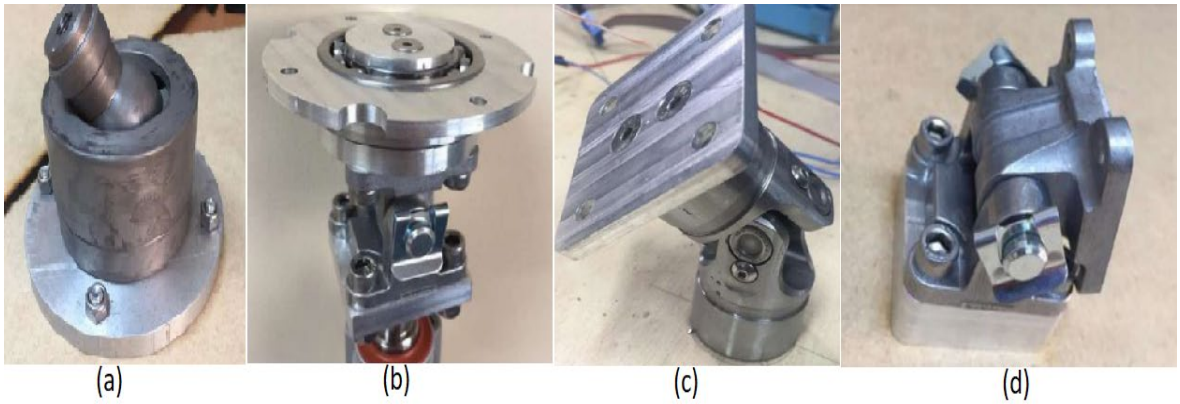


Figura 1.3. a) articulación esférica, b) articulación universal para la pata central, c) articulación universal d) articulación de revolución.

Dentro de este trabajo se clasifica las articulaciones usadas en el Robot 3UPS+RPU en 2 grupos, dependientes e independientes, donde las dependientes son aquellas que no poseen control, es decir, tienen un movimiento libre, mientras que las independientes tienen un motor lo cual permite que se les pueda aplicar control. En la Tabla 1.1 se muestra a donde pertenece cada articulación describiendo además la nomenclatura con la que se lo nombrará en este trabajo.

Tabla 1.1. Clasificación y nomenclatura de las articulaciones dentro del robot 3UPS+RPU

Clasificación	Tipo	Simbología	Pata
Dependientes	Revolución	q_{11}	1
	Revolución	q_{12}	1
	Esférica	q_{24}	1
	Revolución	q_{21}	2
	Revolución	q_{22}	2
	Esférica	q_{24}	2
	Revolución	q_{31}	3
	Revolución	q_{32}	3
	Esférica	q_{34}	3
	Revolución	q_{41}	4
	Revolución	q_{43}	4
	Revolución	q_{44}	4
Independientes	Prismática	q_{13}	1
	Prismática	q_{23}	2
	Prismática	q_{33}	3
	Prismática	q_{34}	4

1.4.1.2 Actuadores

Para mover los vástagos se utilizan motores lineales en cada uno de los cilindros que conforma las patas del robot paralelo. Cada uno de estos motores cuenta con encoders que permiten la realimentación de la posición de cada uno de sus respectivos vástagos, en específico el actuador utilizado para el modelo es el cilindro electromecánico presentado en la Figura 1.4.

Figura 1.4. Cilindro electromecánico Festo Festo ESBF BS-32-100-10P y Motor Maxon 148877.

La gran ventaja presentada en el uso de estos actuadores es su linealidad y facilidad de control, puesto que la posición del vástago se establece por medio del control de un motor que se presenta en la Figura 1.4.

1.4.2 GENERALIDADES DEL SOFTWARE DE IMPLEMENTACIÓN

El entorno de desarrollo de simulación es muy importante dentro del campo de diseño, sea este hardware o software, es por esto que es necesario escoger una plataforma que cumpla con las especificaciones adecuadas con el fin de cumplir el objetivo propuesto. Existen varios softwares disponibles que permiten simular el comportamiento de un robot, cada uno con sus respectivas características. Para este trabajo se utilizarán especialmente dos softwares con los que se realizarán las simulaciones y se comprobará el correcto funcionamiento del controlador de trayectoria, estos son Matlab y CoppeliaSim. El primero, permitirá desarrollar el controlador gracias a la facilidad de sus librerías y de poderse comunicar con otras aplicaciones, como CoppeliaSim. Justamente este último, brindará la facilidad de simular el comportamiento de todo el robot por medio de su interfaz.

1.4.2.1 Matlab

Este software fue creado para facilitar el desarrollo de operaciones complejas como lo es el manejo de matrices, implementación de integrales derivadas, desarrollo de algoritmos, entre otras funciones disponibles. También se puede encontrar una gran facilidad en el manejo del lenguaje ya que se comunica con otros lenguajes de programación como lo es Python, C++, C, etc [13].

En la ventana de trabajo de Matlab es posible crear funciones, matrices o vectores para el manejo de variables y datos para ser usadas en otras aplicaciones o algoritmos que pueden tomar forma por medio del lenguaje de programación utilizado operaciones matemáticas que soporta el software. Para facilitar la creación la programación del software, Matlab presenta la aplicación denominada Simulink que permite realizar esto por medio del diseño basado en diagrama de flujos. Además, es posible implementar en Simulink modelos multidominio, conectar varios modelos y de igual manera organizarlos por medio de bloques. Simulink es una gran opción a la hora de crear modelos matemáticos, puesto que sus librerías abarcan varios campos de la ingeniería y por el lado de robótica posee librerías muy útiles, como es el caso de librerías dedicadas a adquisición de datos y demás bloques para sintonizar controladores [14]. Finalmente, Matlab también permite la creación de interfaces gráficas, los GUIDES, y gracias a esto establece una comunicación interactiva con el usuario.

Generalmente Matlab es usado para implementación de modelos matemáticos donde presenta facilidades notables con respecto a otros lenguajes. Es por esto que para simulación de controladores en un modelo matemático se lo realiza en Matlab, mientras que para la implementación de los controladores en el robot físico ya se busca otro tipo de lenguajes que garanticen una comunicación más ágil y rápida entre el controlador y el actuador [13].

1.4.2.1.1 Comunicación de Matlab-Simulink con otros lenguajes

Los bloques usados en Matlab-Simulink pueden ser de diversos tipos de acuerdo con la necesidad del usuario dentro de los cuales tenemos bloques de operaciones como son la suma, multiplicación, integral, derivada, etc. Así como estos bloques pueden ser tomados de la librería de Simulink, también se los puede personalizar como es el caso de los bloques denominados MATLAB Function dentro de los cuales se pueden programar en el lenguaje de Matlab y de esta manera replicar modelos matemáticos desde los más simples a los más complejos dependiendo únicamente de la capacidad de procesamiento de la computadora de trabajo.

Una forma de comunicación efectiva que posee MATLAB es el uso del bloque S-Function, este bloque permite ejecutar otros tipos de lenguaje dentro de sí, Las funciones S utilizan una sintaxis de llamada especial denominada API de función S que le permite interactuar con el motor de Simulink. Esta interacción es muy similar a la interacción que tiene lugar entre el motor y los bloques integrados de Simulink.

Los bloques S-Function definen cómo funciona un bloque durante las diferentes partes de la simulación, como la inicialización, la actualización, las derivadas, las salidas y la terminación. En cada paso de una simulación, el motor de simulación invoca un método para realizar una tarea específica. Los conceptos básicos de la función S requieren un conocimiento fundamental de las relaciones matemáticas entre las entradas, los estados y las salidas del bloque. En la página de MATLAB se puede encontrar mayor información acerca de todos los métodos de comunicación que presenta este bloque[14].

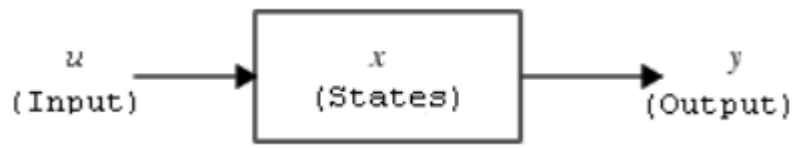


Figura 1.5. Esquema de un bloque S-Function en MATLAB

Se puede elegir el nivel de programación de estos bloques ya que existen bloques S-Function de nivel 1 y nivel 2, dentro del nivel uno se puede programar en lenguajes como C++, C, mientras que en un bloque de nivel 2 solo acepta código de MATLAB.

1.4.2.2 CoppeliaSim

CoppeliaSim es un software que presenta varias características llamativas en la simulación de robots tanto móviles como estáticos, entre las más importantes se encuentran:

- Es multiplataforma, puede ser instalado en los tres sistemas operativos más conocidos, Linux, Windows y Mac.
- La programación de los nodos que unen los eslabones de un robot pueden ser manipulados de tres medios diferentes, uno es por medio de un script dentro del mismo programa, otro a través de un nodo de Ros que se comunica con el programa y el último medio por el que se puede manipular el robot es por una API, con el cual se puede comunicar con aplicaciones externas para la comunicación, control e intercambio de datos.
- El programa soporta varios lenguajes de programación, el principal que utiliza es Lua, el cual es utilizado como lenguaje para los scripts propios del programa, también se encuentran los lenguajes de Python, Matlab y C++.
- El programa ofrece varios diseños de robots con los que podemos integrar controladores y observar su funcionamiento.
- Presenta facilidades para importar y exportar los diseños de robots desde otros programas CAD, de igual manera el software permite configurar la escala y el sistema de referencia.
- Los componentes de un robot se encuentran ordenados en un marco de jerarquía, es decir que existen componentes padres y componentes hijos a los cuales les afecta cualquier cambio físico que se realice en niveles superiores a los que fueron asignados.
- En la interfaz gráfica se puede crear un entorno de simulación para nuestro robot, este entorno puede contar con una trayectoria, obstáculos, luz, etc.

1.4.2.2.1 Objetos en CoppeliaSim

Un resumen de los objetos disponibles para una escena en CoppeliaSim se presentan en la Figura 1.5. Estos poseen varias funciones y aplicaciones, de las cuales se detallan a continuación los empleados en el presente trabajo:

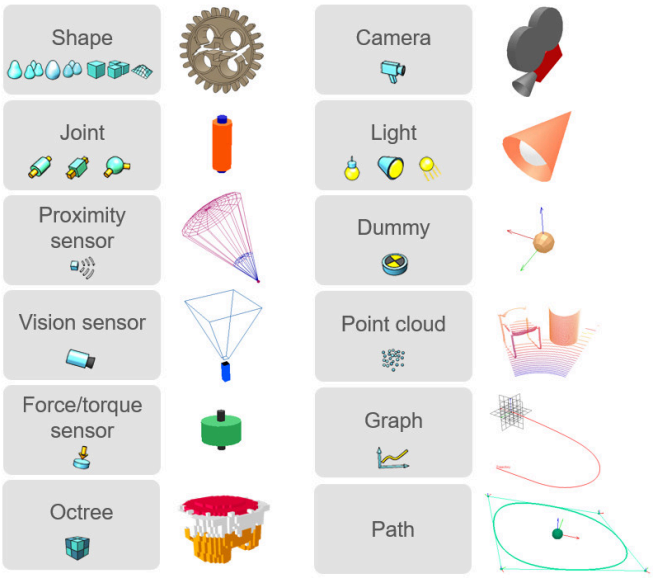


Figura 1.6. Objetos disponibles en las escenas de CoppeliaSim

Formas (shape): es una malla que forma un objeto, este puede ser un cubo, esfera o un cilindro, así como también mallas complejas que forman la parte visual de un modelo o componente de un robot. CoppeliaSim recomienda trabajar siempre con formas o figuras simple, la cuales son cilindros, cubos y esferas[15] . Cuando se importa un robot de una aplicación CAD se aproximan sus piezas a una figura simple como se muestra en la Figura 1.7, para que la simulación no tenga problemas con la falta de recursos computacionales al momento de ejecutarla.

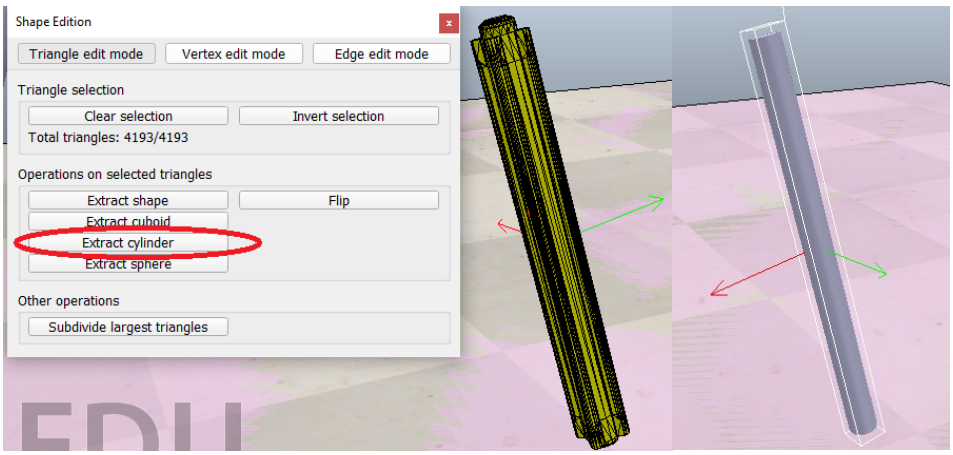


Figura 1.7. Aproximación de un cilindro formado por una malla en una figura simple.

Por medio del menú mostrado en la Figura 1.7 y la selección de la opción *Extract cylinder,* para el caso de este ejemplo, se obtiene el cilindro que nos permitirá simplificar un modelo creado en CoppeliaSim.

Articulaciones(joint): son responsables de unir las piezas móviles de un robot y pueden ser dependientes e independientes. Como se menciona anteriormente, en la explicación referente a la Tabla 1.1, las articulaciones dependientes son aquellas que no poseen control ni actuador, pero se las utiliza como uniones móviles; mientras que, las articulaciones independientes si cuentan con un actuador que se lo emplea para realizar control, sea este de posición o velocidad.

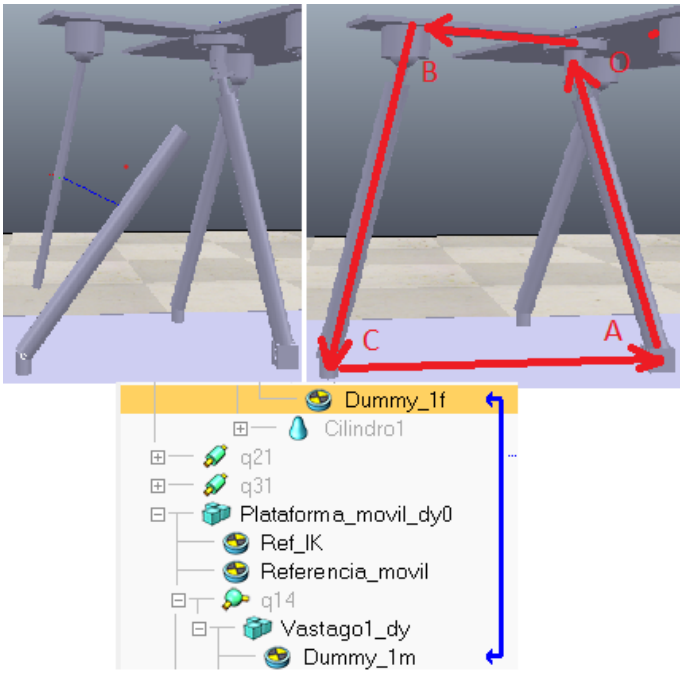


Figura 1.8. Uso de los dummies como enlaces para unir objetos de forma dinámica.

Dummy: es un punto con orientación, dentro de CoppeliaSim presenta muchas formas de usarse, como para establecer enlaces dinámicos y cinemáticos, referencias de objetos, etc. En este trabajo los dummies son usados principalmente para cerrar las cadenas cinemáticas del robot paralelo, como se muestra en la Figura 1.8, donde si no hay este enlace dinámico el objeto cae, pero con los dummies el vástago enlazado no cae. También, por ejemplo, los dummies pueden usarse para obtener la posición del centro geométrico de la plataforma móvil del robot 3UPS+RPU.

De esta manera se pude cerrar una cadena para sistemas cinemáticos cerrados, para el ejemplo de la Figura 1.8 se cierra la cadena conformada por los puntos A,B,C y O.

1.4.2.2.2 Configuración de objetos en CoppeliaSim

La configuración de los objetos o mallas colocados en cada escena es una terea importante y mucho más cuando se trabaja con un robot personalizado, el cual es el caso de este proyecto.

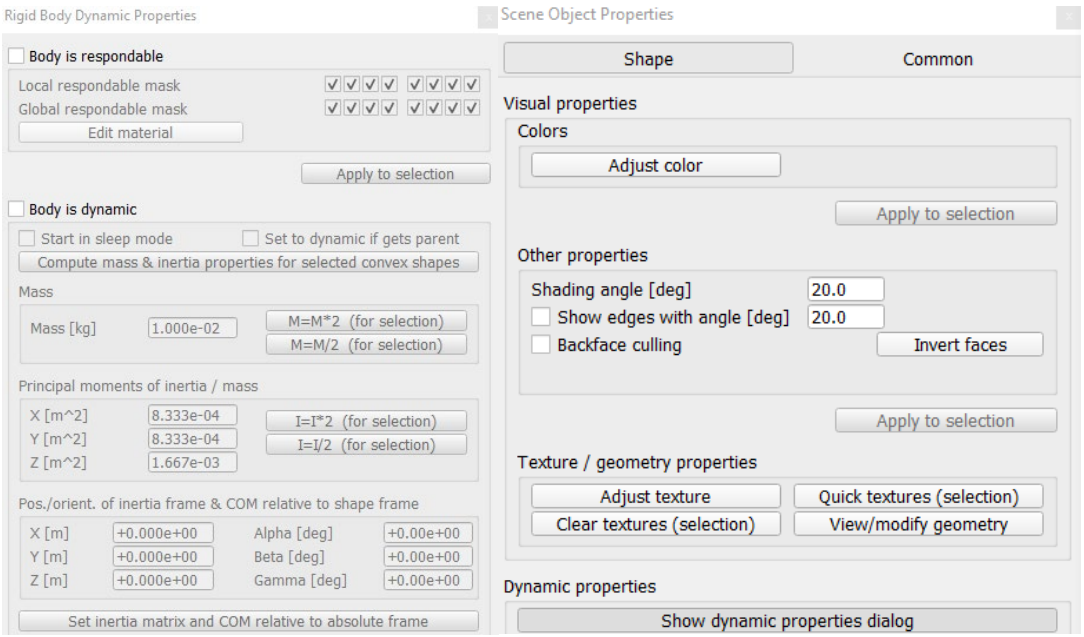


Figura 1.9. Configuración de un objeto en CoppeliaSim.

Los objetos más simples usados son los denominados formas primitivas, los cuales se los puede añadir en la barra de herramientas superior del programa, estas pueden configurarse en modo estático y dinámico. Al configurarse en modo estático el objeto no podrá moverse. Además, se pude configurar si este objeto puede chocar con otros objetos al habilitar la opción *Body is respondable* seleccionando las capas con las que pude interactuar o colisionar, es ahí donde recae la importancia de colocar los objetos en determinadas capas. Por otro lado, al configurar el objeto como dinámico el objeto puede moverse, sea este por colisionar con otro objeto o por la acción de la gravedad. Todas estas opciones de configuración se pueden observar en la Figura 1.9. Es importante mencionar que existen varias operaciones adicionales que se puede lograr con los objetos en una escena, todo esto se pude encontrar más detallado en la página oficial de CoppeliaSim [15].

1.4.2.2.3 Conectividad e interfaces en CoppeliaSim

Como se mencionó, CoppeliaSim presenta varias formas para la comunicación dentro de su interfaz y también con otros programas y estas se pueden dividirse en: intercambio de

datos en el entorno de CoppeliaSim y con aplicaciones exteriores a CoppeliaSim u otras computadoras o dispositivos [6].

En el primer caso, CoppeliaSim puede intercambiar datos por medio de señales que a su vez estas se interpretan como variables globales por medio de funciones o scripts internos. En el segundo, CoppeliaSim puede intercambiar datos con otros programas externos a su interfaz, para este fin existen una gran variedad de métodos de comunicación los cuales se enlistan a continuación:

- Llamado de funciones Script: los scripts también pueden ser llamados por otros medios ya mencionados, como lo es por medio de plugins, a través de un nodo ROS o por medio de una API remota.
- API remota: es la una de las formas más sencillas con la cual podemos comunicar la simulación de CoppeliSim con otro programa o aplicación externa que pueden ser comunicadas por diferentes medios que se resumen en la Tabla 1.2

Tabla 1.2. Versiones para una comunicación tipo API remota [6]**.**

	Facilidad de uso	Habilitación de funciones	Lenguajes
ZeroMQ-basado en una API remota	++	Todos	Python
API remota Legacy	+	Subconjunto	C/C++, Python, Java, Matlab, Octave, Lua.
API remota basada en B0	+	Subconjunto	C/C++, Python, Java, Matlab, Lua.

- ROS: Para la comunicación con ROS existen en sus dos versiones, para ROS y para ROS2, y también se puede comunicar por medio de interfaces desarrollado por otros usuarios a través de un puente entre CoppeliaSim y ROS.
- ZeroMQ: Dentro de los complementos de ZeroMQ podemos encontrar uno que funciona como API y que se comunica por medio de mensajes.
- BlueZero: es un mildware con característica similares a ROS, en el cual podemos conectar procesos y complementos entre varios procesos que se encuentren ejecutándose inclusive si estos no pertenecen a una misma máquina, su comunicación de basa en ZeroQM, es decir por medio del intercambio de mensajes. Dentro de CoppeliaSim ya existen plugins que facilitan esta tarea.
- Puerto Serial: el puerto serial puede ser utilizado para comunicación con otros dispositivos y la simulación.

- Sockets: Esto lo podemos realizar añadiendo una extensión denominada LuaSocket

1.4.2.2.4 Funciones API remotas de CoppeliaSim para comunicación con Matlab

Existe una amplia lista de funciones que utiliza una API remota, las mismas que se puede encontrar en la página oficial de CoppeliaSim [16]. A continuación, se presentan las funciones más importantes mediante las cuales se puede establecer comunicación entre el software de CoppeliaSim y Matlab.

simxStart: esta función es la primera en ser iniciado en el lado de Matlab puesto que inicia un hilo de comunicación con CoppeliaSim. En esta función se necesita especificar los parámetros de connectionAddress que especifica la dirección IP donde se encuentra el simulador CoppeliaSim, connetionPort que especifica el número de puerto mediante el cual se conectarán, waitUntilConnected que activa el tiempo de espera para conexión de los bloques, doNotReconnectOnceDisconnected que activa el tiempo de espera ante una desconexión de la comunicación, finalmente los campos de timeOutInMs y CommThreadCycleInMs especifican los tiempos en milisegundos de los dos últimos campos activos respectivamente.

simxGetObjectHandle: esta función guarda el identificador de una variable o, en este caso, una articulación y lo guarda en una variable para que pueda ser usado por Matlab. Los campos que se deben especificar son: clientID, el nombre del objeto y el modo de operación, para el cual se recomienda el modo simx_opmode_blocking.

simxSetJointTargetPosition: la función establece la posición de referencia de una articulación, para esto es necesario haber colocado previamente la articulación en el modo par/fuerza y de igual manera haber habilitado el motor de la articulación, así como también el control de posición. Los parámetros para esta función son: Client ID, el nombre de la articulación, el valor de la posición en radianes y el modo de operación que al igual que la anterior función se recomienda el modo simx_opmode_blocking.

Los softwares de simulación vistos, son importantes para el desarrollo de la co-simulación entre MATLAB y CoppeliaSim. La Co-simulación es la que facilita el uso de estos programas en sincronización y de esta manera se pude aprovechar la versatilidad del manejo de matrices de MATLAB en un entorno virtual como lo es CoppeliaSim [17].

1.4.3 CONTROLADORES PARA SISTEMAS ROBÓTICOS

A un robot se lo puede analizar como un sistema que necesita un control adecuado para obtener una salida deseada, como puede ser el caso de una posición, trayectoria, fuerza o

velocidad. En este sentido, el robot, al igual que cualquier sistema, necesita de realimentación y una acción de control para llegar a una referencia predefinida. Es aquí donde entran los diversos controladores existentes, unos clásicos como los PID y otros más contemporáneos como los basados en lógica Fuzzy. El diseño de un controlador complejo no siempre es la solución en todos los casos, ya que un PID clásico puede alcanzar errores aceptables, por lo que es necesario analizar el sistema antes de elegir un controlador.

Los principales métodos de control utilizados en robots se encuentran dentro de los denominados sistemas de control lineal y no lineal. Los sistemas de control lineal son los más utilizados en el área de la robótica industrial dada su relativa facilidad al ser implementados en hardware en comparación a los sistemas no lineales. En la gran mayoría de los robots estos sensores son encoders ubicados en cada motor de su respectiva articulación. Para trabajar con sistemas lineales es preciso linealizar previamente un sistema y para facilitar más el cálculo, en algunos casos se puede trabajar con *n* ecuaciones independientes para *n* controladores de sus respectivas articulaciones [18].

1.4.3.1 Esquema de control en cascada

El esquema de control en cascada es muy usado en varias áreas, incluido el campo de la robótica [19]. Este esquema se encuentra formado por dos bloques, uno denominado primario, el cual envía la referencia al control secundario.

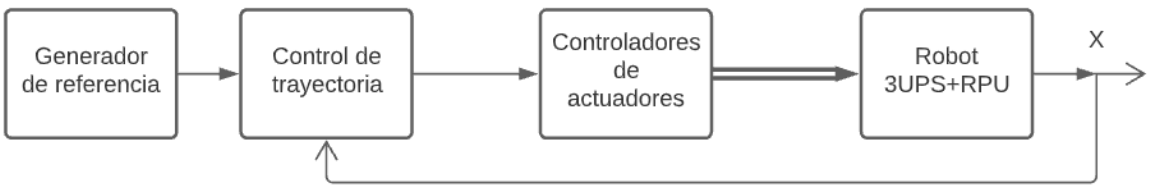


Figura 1.10. Esquema de control de trayectoria para el robot 3 UPS+RPU

Para el robot paralelo 3 UPS+RPU, en la Figura 1.10 se muestra el esquema general de control en cascada, donde se puede notar que se necesitará un control de trayectoria el cual será el encargado de enviar una señal de referencia a cada controlador de posición de su respectivo eslabón. El bloque de los controladores de posición de los actuadores se encargará de generar la acción de control que moverá las articulaciones prismáticas de los eslabones y como realimentación del sistema se obtendrá la posición del centro geométrico de la plataforma móvil, así como también su orientación.

2 METODOLOGÍA

Este capítulo contempla el procedimiento llevado a cabo para desarrollar el controlador de trayectoria para el robot paralelo 3UPS+RPU virtual, así como su implementación en CoppeliaSim. Con este fin se investigó en trabajos anteriores relacionados a este proyecto el modelo cinemático inverso para la creación de un controlador.

2.1 CINEMÁTICA DEL ROBOT 3UPS+RPU

La cinemática inversa del robot 3 UPS + RPU se encuentra detallado en el trabajo [20], en este apartado se sitúa un análisis rápido de las principales ecuaciones para la cinemática inversa y su Jacobiano, que posteriormente serán usadas para el desarrollo del controlador.

2.1.1 CINEMÁTICA INVERSA

La cinemática inversa permite conocer la posición de cada una de las articulaciones del robot, es decir de las articulaciones esféricas, prismáticas y de revolución, a partir de la posición y orientación del centro de la plataforma móvil, estas se encuentran dadas por las variables $x_m, y_m, z_m, \phi,\ \theta, \psi$, las tres primeras para la posición y las tres siguientes para la orientación.

El modelo cuenta con dos sistemas de referencia, uno en la plataforma fija, que se le denominará referencia fija, y el otro que se encuentra en la plataforma móvil. Los puntos de interés son los extremos de cada uno de los eslabones, es decir los puntos A, B, C tanto en la plataforma móvil como en la fija, mismos que se muestran en la Figura 2.1.

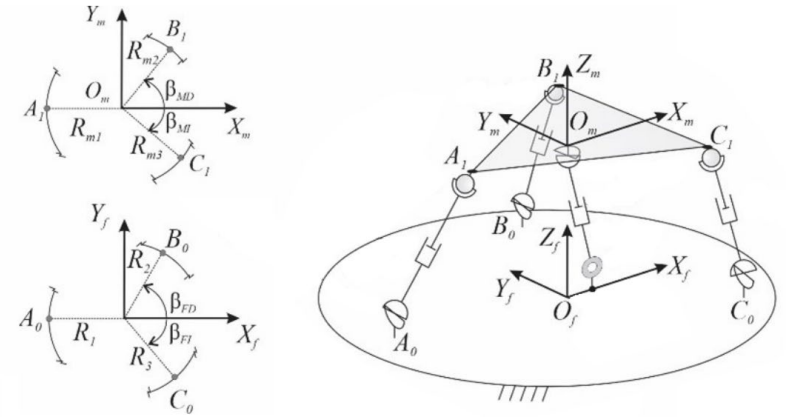


Figura 2.1. Posición de vértice de las patas y sistema de referencia [9].

Para encontrar la matriz de transformación, se emplean los parámetros de Denavit-Hartemberg [7] obtenidos a partir de modelar las articulaciones universales como dos

articulaciones de revolución ubicados, mientras que las articulaciones esféricas son modeladas como tres articulaciones de revolución. Gracias a esto se obtienen los parámetros de las Tablas 2.1 y 2.2.

Tabla 2.1. Parámetros DH para las patas UPS.

Barra i-ésima	α_i	a_i	d_i	θ_i
1	-π/2	0	0	q_1
2	π/2	0	0	q_2
3	0	0	q_3	0
4	π/2	0	0	q_4
5	π/2	0	0	q_5
6	π/2	0	0	q_6

Tabla 2.2. Parámetros DH para la pata RPU.

Barra i-ésima	α_i	a_i	d_i	θ_i
1	$-\pi/2$	0	0	q_1
2	$-\pi/2$	0	q_2	π
3	π/2	0	0	q_3
4	0	0	0	q_4

En estas Tablas q_i representa las variables de la articulación i. En las patas UPS q_1 y q_2 son los ángulos de las articulaciones que conforman la unión universal, q_3 es el valor de desplazamiento de la articulación prismática, mientras que, q_4 y q_5 son las articulaciones con las que se modela la articulación esférica que la une con la plataforma móvil. Por otro lado, en la pata RPU la articulación q_1 es la articulación de revolución que une la pata RPU con el cilindro, q_2 es el desplazamiento de la articulación prismática y por último las articulaciones q_3 y q_4 son articulaciones para modelar la articulación universal.

Por medio de los parámetros de las Tablas 2.1 y 2.2 se establece la matriz de transformación homogénea [7], correspondiente a la siguiente expresión:

$$H_{i_1,i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\cos\alpha_i & r_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\cos\alpha_i & r_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

El siguiente parámetro a encontrar es la matriz de rotación de la plataforma móvil f_{R_m} con respecto al ángulo de Tait-Bryan, de acuerdo a esta notación se asigna 0° al objeto localizado en la horizontal, de forma general se obtienen las Ecuaciones (2.2) y (2.3) considerando que no se tiene ángulo de giro en el eje x.

$$fR_m = R_x(\phi)R_y(\theta)R_z(\psi) \tag{2.2}$$

$$fR_m = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\theta \\ \sin\psi & \cos\psi & 0 \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix} \tag{2.3}$$

También es necesario conocer la ubicación de los eslabones respecto a la referencia de la plataforma fija y móvil, lo cual se lo puede realizar mediante la Ecuación (2.3) en la plataforma fija y la Ecuación 2.4 para ubicar el eslabón en la plataforma móvil.

$$r_{Of} = \begin{bmatrix} R_f.\cos\theta \\ R_f.\sin\theta \\ d \end{bmatrix} \tag{2.4}$$

$$r_{Om} = \begin{bmatrix} R_m.\cos\theta \\ R_m.\sin\theta \\ d_m \end{bmatrix} \tag{2.5}$$

Donde d y d_m representa el desplazamiento del centro de la plataforma fija y móvil del eslabón central a lo largo del eje x, respectivamente. Por medio de los vectores para ubicar los puntos A, B y C de la plataforma móvil y fija hallados, se definen dos caminos para hallar el mismo punto. Por ejemplo, este puede ser el caso de A_1 en la Figura 2.2, donde se puede llegar a ese punto por dos caminos diferentes, una por los puntos $O_f, A_0\ y\ A_1$ y otra por los puntos $O_f, O_m\ y\ A_1$.

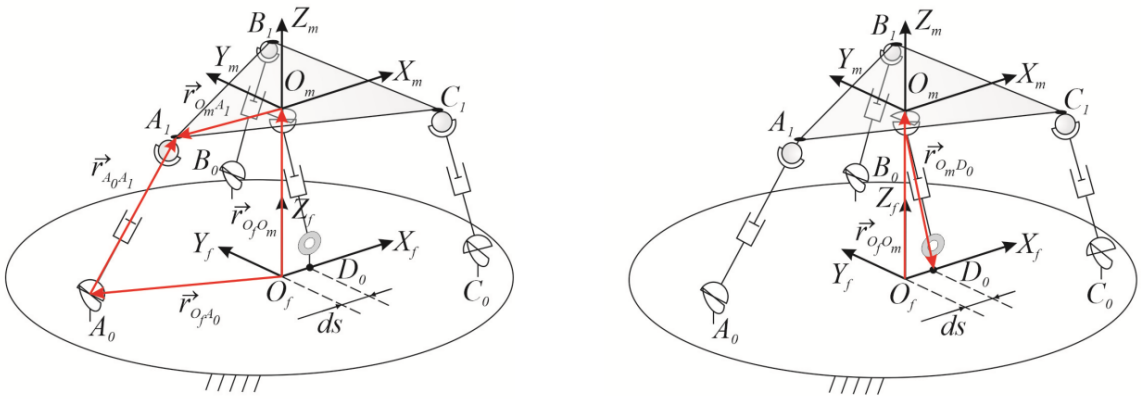


Figura 2.2. Cierre de las cadenas cinemáticas para el punto $A1$ y O_m en la plataforma móvil [9].

Al igual que A, los puntos B y C tienen dos caminos con las que se puede llegar a los puntos formándose un sistema de ecuaciones que tienen la estructura presentada a continuación

$$\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} + f_{Rm}.r_{Om,N} = [H_{FP1}H_{01}(q_{i1})H_{12}(q_{i2})H_{23}(q_{i3})] \quad (2.6)$$

Donde el lado de la izquierda de representa una forma para llegar al punto N que pude ser A, B o C en la plataforma móvil, mientras que el lado izquierdo muestra cómo se pude llegar al mismo punto por medio de la matriz de transformación donde q_i es valor de la articulación para la pata *i*.

Al encontrar las ecuaciones para llegar a los puntos A, B y C de la plataforma móvil se obtienen las ecuaciones correspondientes a la cinemática inversa del robot 3 UPS+RPU correspondientes a las Ecuaciones en (2.7), denominadas también como ecuaciones de restricciones de posición. En estas ecuaciones los valores correspondientes a ϕ y y_m son igualadas a cero debido que el robot no tiene movimiento sobre estos ejes, el proceso de obtención de estas ecuaciones se encuentra detallado en el trabajo [7].

$$\Phi_1 = x_m - R_m \cos\theta \cos\psi - q_{13} \cos q_{11} \sin q_{12} + R$$

$$\Phi_2 = R_m \cos\theta \sin\psi - q_{13} \cos q_{12}$$

$$\Phi_3 = z_m + R_m \sin\theta - q_{13} \sin q_{11} \sin q_{12}$$

$$\Phi_4 = x_m + R_m \cos\beta \cos\theta \cos\psi - R_m \sin\beta \sin\psi - R \cos\beta - q_{23} \cos q_{21} \sin q_{22}$$

$$\Phi_5 = R_m \cos\beta \cos\theta \sin\psi + R_m \sin\beta \cos\psi - R \sin\ \beta + q_{23} \cos q_{22}$$

$$\Phi_6 = z_m - R_m \cos\beta \sin\theta - q_{23} \sin q_{21} \sin q_{22} \quad (2.7)$$

$$\Phi_7 = x_m + R_m \cos\beta \cos\theta \cos\psi + R_m \sin\beta \sin\psi - R \cos\beta - q_{23} \cos q_{31} \sin q_{22}$$

$$\Phi_8 = R_m \cos\beta \cos\theta \sin\psi - R_m \sin\beta \cos\psi + R \sin\beta + q_{33} \cos q_{32}$$

$$\Phi_9 = z_m - R_m \cos\beta \sin\theta - q_{33} \sin q_{31} \sin q_{32}$$

$$\Phi_{10} = x_m + q_{42} \sin q_{41}$$

$$\Phi_{11} = z_m - q_{42} \cos q_{41}$$

El sistema de Ecuaciones (2.7) permite conocer las incógnitas $q_{11}, q_{12}, q_{21}, q_{22}, q_{31}, q_{32}, q_{41}, q_{11}, q_{23}, q_{33}, q_{42}$, ya que el valor de la posición de la plataforma

móvil $x_m, z_m, \theta\ y\ \psi$ son variables de entrada, mientras que, las posiciones de las articulaciones independientes e independientes son la salida a la solución de la cinemática inversa.

2.1.2 JACOBIANO

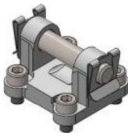
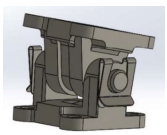
El Jacobiano permite obtener la velocidad de la plataforma móvil a partir de los valores de las velocidades de las articulaciones independientes. Partiendo de las Ecuaciones en (2.7) se puede relacionar las velocidades en las articulaciones con la velocidad generada en la plataforma móvil por medio de la ecuación $v = J(q).\dot{q}$. Para relacionar las ecuaciones en (2.7) se deriva cada fila con respecto a las variables que tenemos en la matriz, mostradas con más detalle en el Anexo III, y resumido en la Ecuación 2.8.

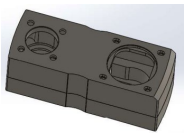
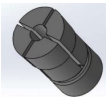
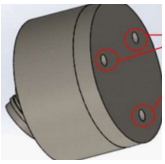
$$\left[\frac{\delta \Phi_i}{\delta q^s}\right] = J(q) \qquad (2.8)$$

2.2 ESPECIFICACIONES PARA EL ROBOT 3UPS+RPU

Las especificaciones de diseño de cada una de las piezas en CAD del robot 3UPS+RPU se encuentran detalladas en [8], mientras que la distribución de los eslabones se muestra en la Figura 1.2. De forma resumida se muestran los componentes usados en el modelo virtual en la Tabla 2.3., Cabe destacar que parámetros como la inercia lo puede determinar CoppeliaSim, ya que se va a trabajar con la aproximación a figuras geométricas simples, como son cilindros, esferas y cubos, debido a los recursos computacionales con los que se cuenta. Un ejemplo de esta aproximación se presentó en la Figura 1.7.

Tabla 2.3. Resumen de los elementos usados para el modelo virtual del robot 3UPS+RPU.

Nombre	Especificaciones	Diseño en SolidWorks
Brida SNCL	Marca: Festo Peso: 71g	
Brida SNCB	Marca: Festo Peso: 103 g	
Junta de revolución SNCL/SNCB	Marca: Festo Peso: 174g	

Conjunto paralelo	Marca: Festo Peso:608g	
Mangito de fijación	Marca: Festo Peso: 19g	
Motor	Marca: Maxon Peso: 0.48 kg	
Junta esférica	Marca: Cicrosa Peso: 1.5 kg	
Junta universal	Marca: Belden Peso: 300 g	

Todos los elementos nombrados fueron diseñados en el software SOLIDWORKS a partir del trabajo desarrollado en [8], donde también se especifica el diseño de la plataforma móvil y fija. También se debe considerar que para unir las articulaciones con los conjuntos paralelos se necesita de acoples, mismo que se detallan en [8]. En el presente trabajo se adaptó en SOLIDWORKS a la configuración de patas de la Figura 1.2, obteniendo como resultado el CAD mostrado en la Figura 2.3. Esta adaptación fue necesaria, debido a los cambios recientes en la configuración de la plataforma real.

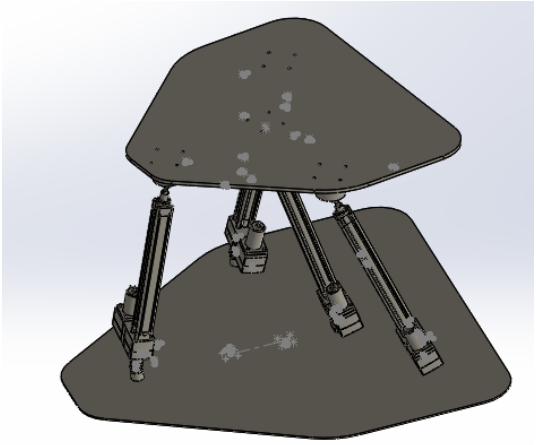


Figura 2.3. Robot adaptado en el software SOLIDWORKS a la configuración de patas del modelo real.

2.3 IMPLEMENTACIÓN DEL MODELO VIRTUAL

Para implementar el robot 3 UPS+RPU se considera primero el diseño del mismo en un software CAD, para propósitos del trabajo se utilizó inicialmente el software de SOLIDWORKS. Una vez completado el modelo en este programa se exportó por medio de un formato .STL al Software de CoppeliaSim. En la importación se puede especificar la escala del robot y también el eje de referencia para la importación, posteriormente se obtiene el resultado mostrado en la Figura 2.4.

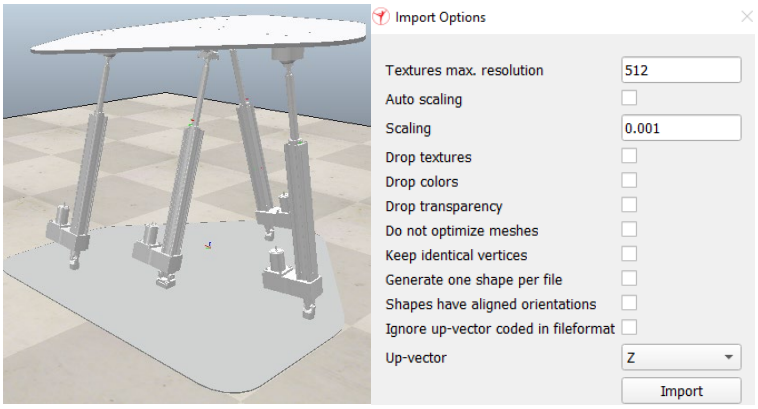


Figura 2.4. Robot importado a CoppeliaSim con sus respectivas opciones de importación.

El modelo en CoppeliaSim se lo puede desarrollar en diferentes capas, para el caso de este proyecto se coloca todo el robot importado en la capa 1 de visualización. Una vez importado el modelo del robot 3 UPS+ 1RPU es necesario agrupar los elementos del robot y colocar sus respectivos nombres en cada parte, los cuales se muestran en la Figura 2.5. De esta forma es posible una rápida identificación de cada uno de los elementos que lo conforman. Por facilidad para la simulación del movimiento, se agrupan los objetos que se mueven en conjunto, como en el caso del actuador M1 que se mueve junto con el cilindro 1, conformando un grupo.

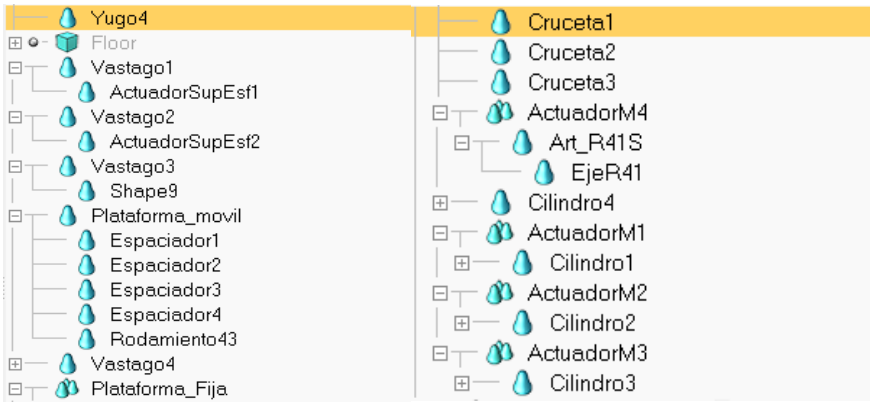


Figura 2.5. Agrupamiento de las figuras importadas desde el programa CAD

Los distintos elementos que se mencionan en la Figura 2.5 se los ubica en el robot en el Anexo I para mayor comprensión de los mismos.

Con cada uno de los elementos del robot ya identificadas con un nombre y agrupados, es importante extraer o aproximar a una forma geométrica simple cada una de las partes del robot y armar un modelo simplificado, este proceso fue explicado en la Sección 1.4.2.2.1. Es importante que todos los elementos geométricos simples extraídos del modelo importado se deben colocar en una capa diferente a la capa que se observará en la simulación como se muestra en la Figura 2.6, esto para separar de forma adecuada las piezas dinámicas del robot de las piezas exportadas como mallas del programa CAD, puesto que este será la que se visualizará durante la simulación del robot.

La extracción de las formas geométricas simples se utiliza para formar la estructura de un robot simplificado, misma que es mostrada en la Figura 2.6. Esta es la que realmente tiene las características dinámicas y cinemáticas con las que se ejecutará la simulación, de esta manera se simplifica la configuración y selección de objetos dinámicos. Por ejemplo, dentro de las características dinámicas se deben configurar el peso correspondiente al robot real de cada una de las figuras geométricas que forman el modelo simplificado.

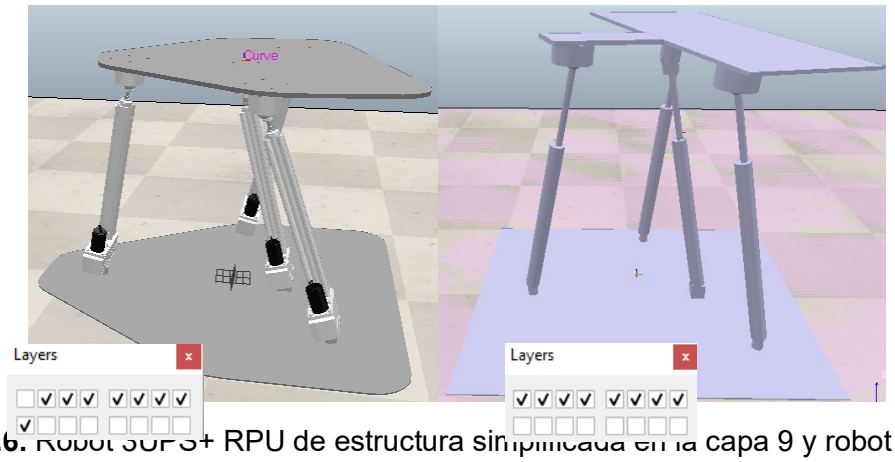


Figura 2.6. Robot 3UPS+ RPU de estructura simplificada en la capa 9 y robot en la capa 1 de presentación.

La obtención del modelo simplificado por figuras simples como cilindros y cubos se lo realiza para reducir el número de procesos computacionales al momento de la simulación. Si bien el modelo con los elementos compuestos por mallas importado del programa CAD se podría simular, este emplearía demasiados recursos de la computadora lo que se vería reflejado en los resultados y utilización de la simulación. La malla importada desde el Programa CAD ayuda principalmente a ubicar cada pieza del robot simplificado y orientarlo, de ahí radica su importancia. Adicionalmente, la malla es la capa del robot que se mostrará durante la simulación.

2.3.1 MODO CINEMÁTICO

Antes de iniciar con la jerarquización de las piezas del robot simplificado y las piezas del CAD importado es necesario que estas se encuentren orientadas con los ejes referencia deseados, es decir se las debe orientar con respecto al marco de referencia global del programa de simulación. Esto se logra haciendo coincidir los ejes de referencia de cada figura con los deseados, un ejemplo se muestra en la Figura 2. 7.

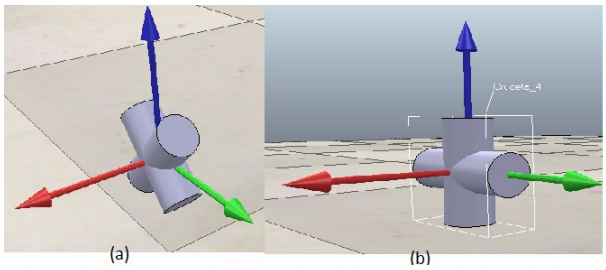


Figura 2.7. (a)Figura no orientada, (b)Figura orientada respecto al marco de referencia global

A continuación, a partir del modelo simplificado mostrado en la Figura 2.6, se ordenan de forma jerárquica los elementos del robot. Considerando que en un orden jerárquico existen objetos padres y objetos hijos, como se muestra en la Figura 2.8, los objetos hijos se mueven junto a los objetos padre.

Figura 2.8. Orden jerárquico de los elementos del robot para el modelo cinemático del robot.

Un ejemplo sencillo para entender la jerarquía entre un objeto padre y un objeto hijo se muestra en la Figura 2.9 donde si se mueve la forma Shape_1 este se moverá con la junta de revolución Joint y la forma Shape 1 manteniendo la ubicación y orientación constante con respecto al objeto padre Shape_1.

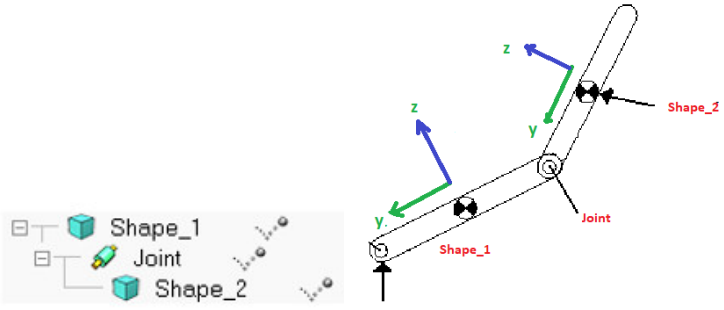


Figura 2.9. Jerarquía de un objeto padre y un objeto hijo

Para cerrar la cadena cinemática del robot se considera la primera cadena la cual está conformada por la plataforma fija, Cilindro 4, yugo 4 y plataforma móvil, por lo que, esta es el padre o base de los otros 3 eslabones restantes. De esta manera, se cierra la cadena por medio de enlaces cinemáticos de dummies (explicados en la Sección 1.4.2.2.1) ubicados en los extremos inferiores de las patas UPS, como se esquematiza en la Figura 2.10.

Para la creación del modelo se utiliza el plugin de cinemática de CoppeliaSim, donde primero se añade un grupo IK, el cual es un plugin de CoppeliaSim que permite resolver problemas de cinemática inversa y directa de un robot, dentro del cual se crean elementos. Cada elemento creado corresponde a una cadena cinemática explicado de forma gráfica en la Figura 2.10.

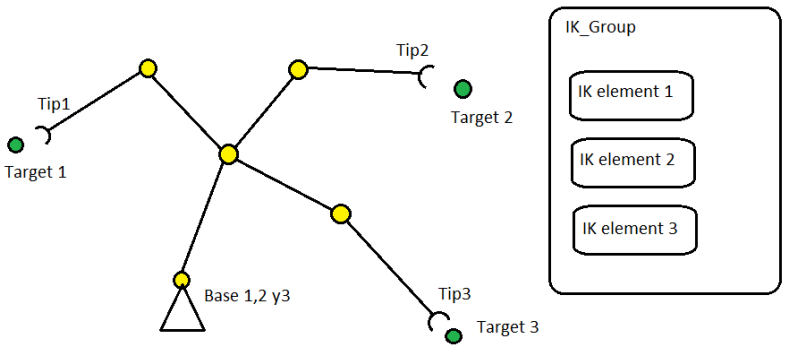


Figura 2.10. Cadena cinemática cerrada de los tres eslabones por medio de la creación de un grupo IK y sus correspondientes elementos cinemáticos.

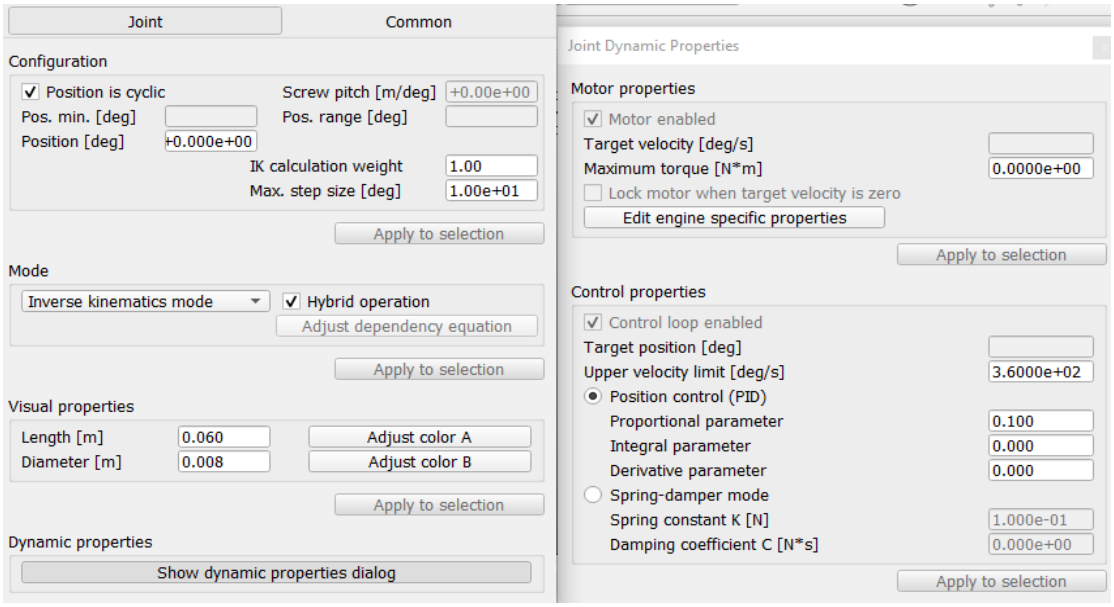


Figura 2.11. Configuración de una articulación dependiente.

Como paso final es necesario configurar todas las articulaciones dependientes con el modo de cinemática, mientras que las articulaciones independientes, en modo pasivo, esta clasificación de articulaciones se encuentra en la Tabla 1.1. De esta manera se pude variar la posición de las articulaciones prismáticas manteniendo cerrada la cadena cinemática. En la Figura 2.11 se muestra la configuración de la articulación dependiente al igual que las independientes. En el Anexo I se detalla cómo se encuentran configuradas las distintas articulaciones.

2.3.2 MODO DINÁMICO

Una vez obtenida la forma simplificada del robot se deben ordenar y ubicar cada una de las partes del robot por niveles para posteriormente ir colocando sus respectivas articulaciones y configurarlas en modo par/fuerza.

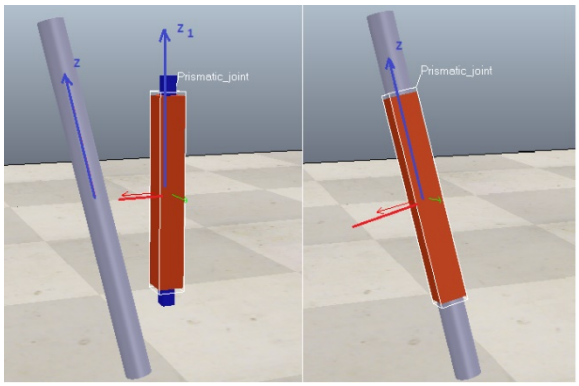


Figura 2.12. Eje z del vástago en la misma dirección del eje z de la articulación prismática.

Lass articulaciones deben ubicarse de tal forma que su eje z coincida con el avance de la articulación prismática, como se muestra en la Figura 2.12. También se debe habilitar el motor y el controlador en las articulaciones que se desea que tenga control, así como habilitar el controlador PID para controlar su posición.

Para unir las articulaciones prismáticas, es decir el cilindro con el vástago, es necesario utilizar un dummy, el cual es una referencia colocada a ambos lados y enlazados y de esta manera cerrar la cadena del robot de tal forma que se pueda simular su movimiento, revisar la Sección 1.4.2.2.1. Por esto, la unión de articulaciones se lo realiza por medio de un enlace dinámico de un dummy, como se muestra en la Figura 2.13, donde se observa el enlace entre el dummy_1m y el dummy_1f y de esta manera unir el vástago 1 con el cilindro 1. En la Tabla 2.4 se muestra todos los enlaces utilizados en el robot en CoppeliaSim

Tabla 2.4. Enlaces de Dummies usados en el robot 3UPS+RPU en CoppeliaSim

Enlace	Dummy 1	Dummy 2	Unión
Dinámico	Dummy_1m	Dummy_1f	Cilindro1-vástago1
Dinámico	Dummy_2m	Dummy_2f	Cilindro2-vástago2
Dinámico	Dummy_3m	Dummy_3f	Cilindro3-vástago3
Dinámico	Dummy_4m	Dummy_4f	Cilindro4-vástago4

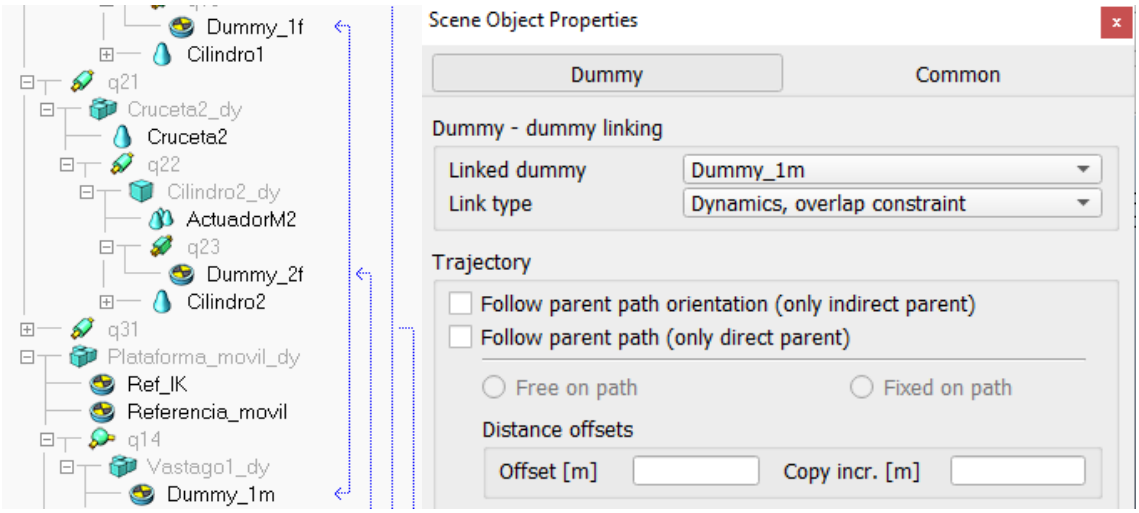


Figura 2.13. Enlace de un dummy con otro para cerrar una cadena cinemática.

Para simular el modelo en modo dinámico se necesitan configurar las articulaciones dependientes e independientes. Las articulaciones independientes se configuran en modo par-fuerza, como se muestra en la segunda imagen de la Figura 2.14 y posteriormente se habilita el motor y de esta manera poder enviar una señal para su respectivo control.

Mientras que las articulaciones dependientes se las configura en un modo híbrido, modo cinemático y dinámico, a excepción de las articulaciones esféricas. En el Anexo I se encuentra detallado la nomenclatura de las articulaciones usadas en este trabajo y de igual manera el modo en el que se encuentra configurado. De este modo se puede simular el comportamiento del robot y evitar de esta manera la caída de la plataforma móvil.

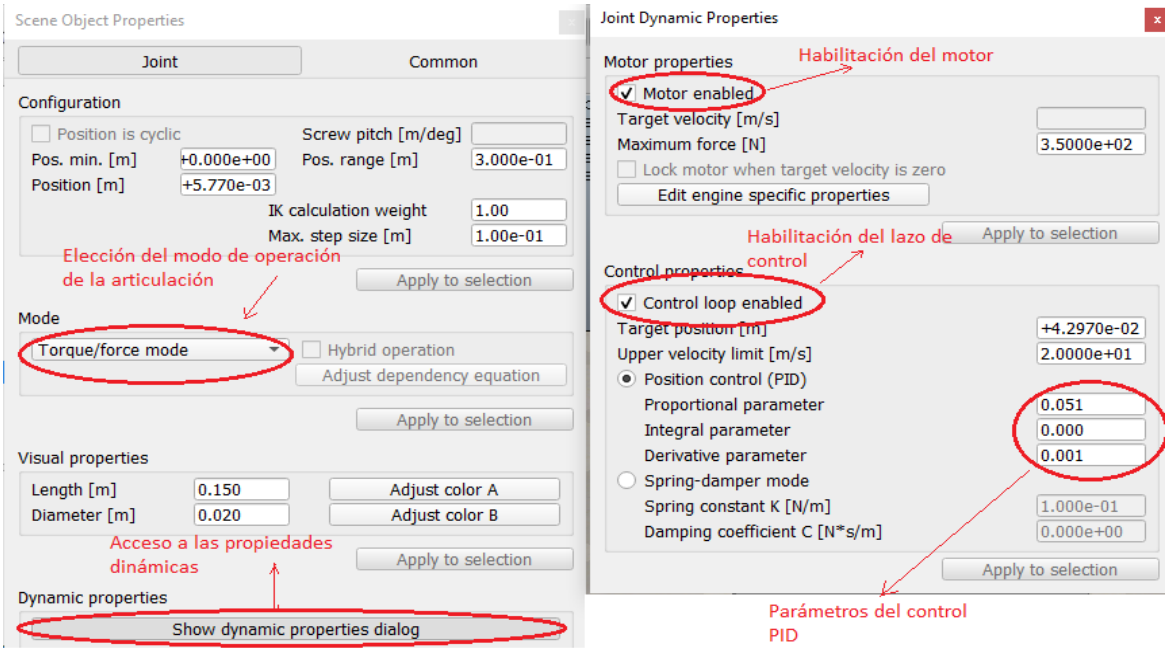


Figura 2.14. Agrupación en niveles de los elementos del robot y opciones de configuración para la articulación.

2.4 ESQUEMA DEL CONTROLADOR DE TRAYECTORIA

El robot 3UPS+RPU consta de 4 eslabones, cada uno cuenta con un actuador lineal en el cual se empleará un controlador de posición tipo PID, que formará parte del esquema de control de trayectoria del robot 3UPS+RPU. Dado que las características de los actuadores son las mismas para cada eslabón el controlador de posición es el mismo. Su desarrollo se lo realiza dentro del entorno de CoppeliaSim debido a su fácil implementación, ya que se lo pude encontrar como parámetro de configuración para una articulación.

El esquema presentado en la Figura 2.15 resume la arquitectura de control. El controlador de trayectoria provee la posición cartesiana de la plataforma móvil, obteniendo como acción de control un valor de velocidad de cada una de las articulaciones, misma que a través del Jacobiano inverso se obtiene la velocidad de los eslabones.

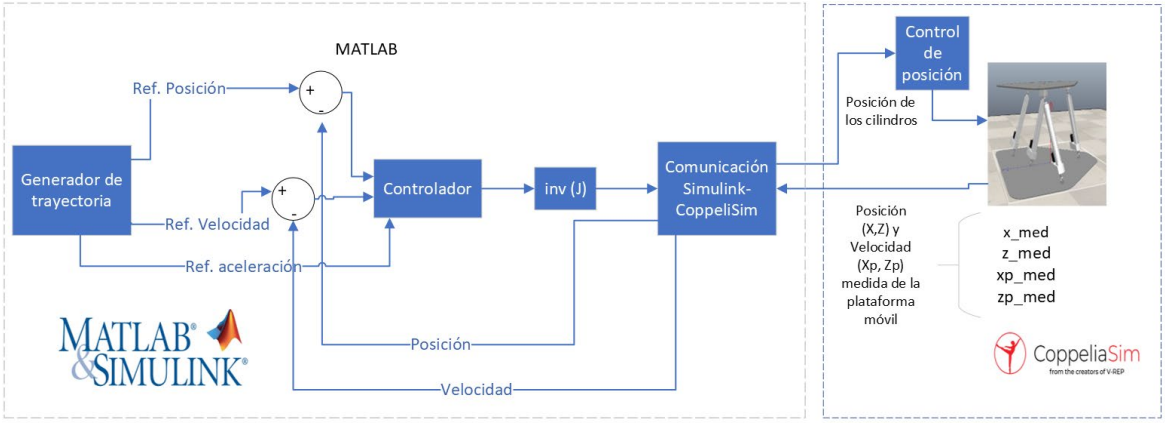


Figura 2.15. Esquema resumido del controlador implementado.

2.4.1 BLOQUE GENERACIÓN DE TRAYECTORIA

Este bloque establece la trayectoria que debe seguir la plataforma móvil del robot paralelo, mismo que es mostrado en la Figura 2.16. Esta es establecida por medio de funciones que varían en el tiempo.

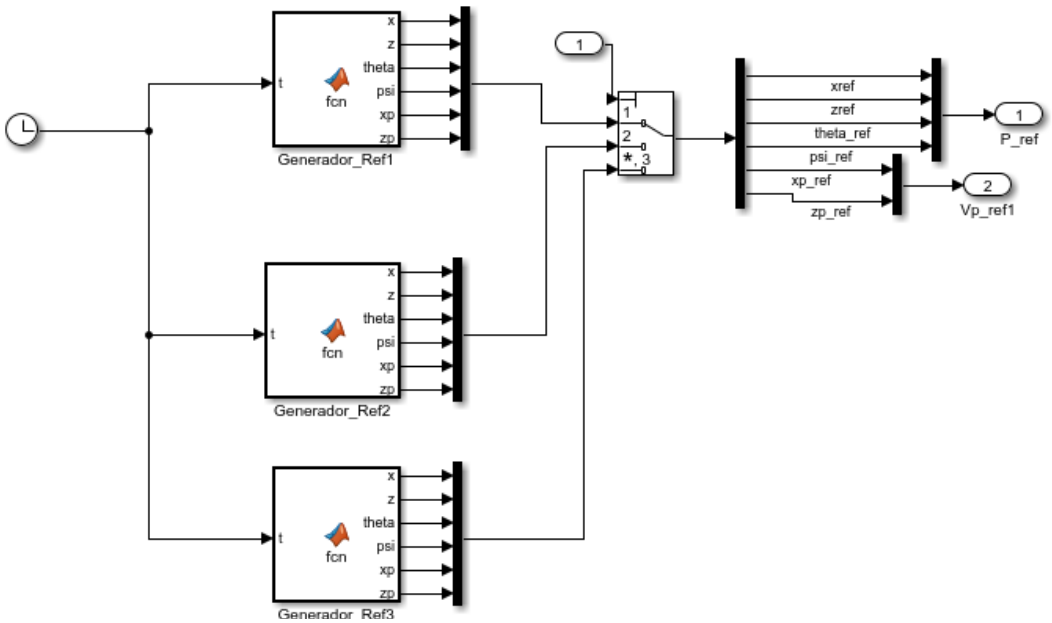


Figura 2.16. Bloque generador de referencia

Para este proyecto, la primera trayectoria disponible en el bloque es un círculo, la segunda es un triángulo rectángulo y la tercera es una trayectoria cuadrada. Para realizar el control, este bloque también entrega los valores de la velocidad y aceleración en los ejes xz.

Cada bloque de referencia, de la Figura 2.16, genera una trayectoria, el bloque Generador_Ref1 genera un triángulo, el bloque Generador_Ref2 genera un círculo, y finalmente el bloque Generador:Ref3 genera un rombo. La referencia 2 y 3 se generan por

medio de un polinomio cúbico, mientras que la segunda referencia genera un círculo. Se detallan a continuación el método de generación de cada uno de los bloques.

2.4.1.1 Trayectoria mediante un polinomio cúbico

En la generación de trayectoria se considera un desplazamiento con velocidades moderadas, tanto en el eje x como en el eje z, para lo cual se utiliza una función polinomial de tercer grado para desplazar la plataforma móvil en x y z, dicha ecuación viene dada por las siguientes expresiones:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (2.9)$$

$$z(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \quad (2.10)$$

A partir de las ecuaciones mostradas se puede determinar la velocidad en los dos ejes de interés mediante las siguientes ecuaciones:

$$\dot{x}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (2.11)$$

$$\dot{z}(t) = b_1 + 2b_2 t + 3b_3 t^2 \quad (2.12)$$

De igual manera se pueden hallar sus respectivas aceleraciones:

$$\ddot{x}(t) = 2a_2 + 6a_3 t \quad (2.13)$$

$$\ddot{z}(t) = 2b_2 + 6b_3 \quad (2.14)$$

Para hallar las contantes a_i y b_i , donde $i \in [1,2,3]$, se toman en cuenta las condiciones iniciales en t_0 y finales en t_f de la trayectoria.

$$x(t_0) = a_0 = x_{m0}; \quad t_0 = 0 \quad (2.15)$$

$$z(t_0) = b_0 = z_{m0}; \quad t_0 = 0$$

$$x(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \quad (2.16)$$

$$z(t_f) = b_0 + b_1 t_f + b_2 t_f^2 + b_3 t_f^3$$

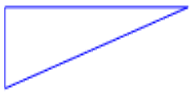
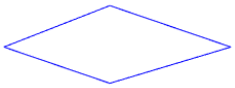
Una de las propiedades del polinomio cúbico es que la pendiente va aumentando progresivamente con el tiempo por lo que no se obtendrán velocidades ni aceleraciones agresivas al inicio ni al final de la trayectoria. Para el presente trabajo, se asume que la velocidad y aceleración tanto al inicio como al final son cero. Con estos datos ya se puede resolver el sistema de 4 ecuaciones y 4 incógnitas obteniendo las siguientes constantes.

$$a_0 = x_0,\ a_1 = 0,\ a_2 = \frac{3}{t_f^2}(x_f - x_0), a_3 = -\frac{2}{t_f^3}(x_f - x_0) \quad (2.17)$$

Las constantes b_i presentan la misma estructura en las ecuaciones que las constantes a_i con la diferencia en sus ejes, es decir, la variable x se remplaza por la variable z.

Mediante el uso del polinomio cúbico se unen puntos para formar trayectorias. Para este trabajo se generaron 2 trayectorias por medio del polinomio cúbico, los puntos de dichas trayectorias se especifican en la Tabla 2.5.

Tabla 2.5. Referencias 1 y 3 generadas por la unión de puntos con el polinomio cúbico

Trayectoria	Puntos de la trayectoria	Tiempo entre cada punto [s]	Figura
2	X Z $\begin{bmatrix} 0 & 0.6 \\ 0 & 0.7 \\ 0.1 & 0.7 \\ 0 & 0.6 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 10 \\ 20 \\ 30 \end{bmatrix}$	
3	X Z $\begin{bmatrix} -0.03 & 0.6 \\ -0.09 & 0.67 \\ -0.03 & 0.75 \\ 0.06 & 0.67 \\ -0.03 & 0.6 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 15 \\ 30 \\ 45 \\ 60 \end{bmatrix}$	

Para generar estas trayectorias por MATLAB se utilizan las Ecuaciones (2.9) a (2.12) y de esta manera generar una matriz que permita hallar los valores de las constantes para unir cualquier punto tanto en x como en el eje z , la misma que es mostrada en las Ecuaciones (2.18).

$$A = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix}$$

$$B = \begin{bmatrix} P_0 \\ P_f \\ 0 \\ 0 \end{bmatrix} \quad (2.18)$$

$$A.C = B$$

Donde P_0 es la posición inicial, P_f es la posición final y C es la matriz donde se hallan las soluciones a las constantes.

2.4.1.2 Trayectoria circular

Una forma simple de generar una trayectoria es establecer una ecuación que describa en el tiempo la posición que debe tener el centro geométrico de la plataforma móvil del robot 3UPS + RPU.

$$x = Acos(\omega t)$$
$$z = A \sin(\omega t) \quad (2.19)$$

De igual manera es posible definir la velocidad de la referencia derivando las Ecuaciones (2.20) quedando como resultado la Ecuación (2.20).

$$\dot{x} = -\omega A sin(\omega t)$$
$$\dot{z} = \omega A cos(\omega t) \quad (2.20)$$

De esta forma se tiene establecido la posición en el espacio de la plataforma móvil. El eje y no se encuentra definido ya que el robot paralelo 3 UPS+RPU no se pude desplazar en ese eje. Para el bloque de Generador_Ref2 de la Figura 2.15 se tomó las ecuaciones 2.19 y 2.20 para la generación de la trayectoria de tal manera que a la salida del bloque tenga posición y velocidad de la trayectoria.

2.4.2 CONTROL DE POSICIÓN

La implementación del controlador de posición se realiza dentro del mismo programa de CoppeliaSim, dentro del cuadro de propiedades de las articulaciones se encuentran las características dinámicas, mismo que es mostrado en la Figura 2.17 Para una articulación, estas incluyen la habilitación de un motor cuando esta opera en modo par/fuerza, además es posible habilitar un control PID, de esta manera el control de posición para los eslabones queda resuelto.

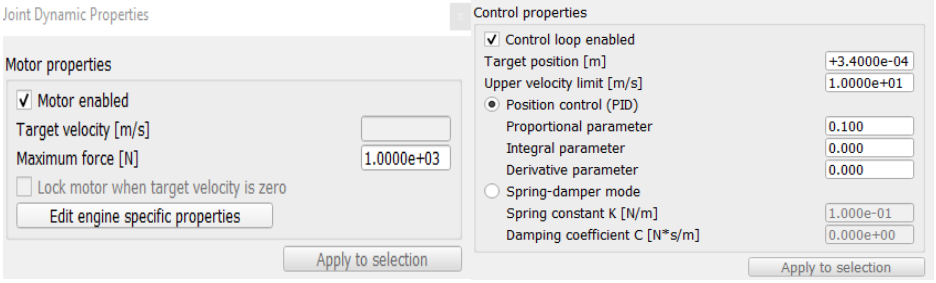


Figura 2.17. Cuadro de configuración dinámica de una articulación prismática.

Para KP de 0.01, en la primera prueba de la Tabla 2.6, se obtiene la respuesta de la Figura 2.18, en la cual se evidencia que ninguno de los cilindros se establece en la referencia, por lo que el primer paso fue variar el valor Kp en pasos de 0.1 hasta conseguir estabilizarlo.

Tabla 2.6. Sintonización del controlador interno de posición para las articulaciones prismáticas.

N prueba	Articulación	Kp	Kd	Ki	ITSE
1	q_{13}	0.01	0	0	0.1011
1	q_{23}	0.01	0	0	0.1018
1	q_{33}	0.01	0	0	0.09454
1	q_{42}	0.01	0	0	0.09361
2	q_{13}	0.05	0	0	0.09235
2	q_{23}	0.05	0	0	0.09238
2	q_{33}	0.05	0	0	0.09242
2	q_{42}	0.05	0	0	0.0923
3	q_{13}	0.05	0	0	0.09236
3	q_{23}	0.05	0	0.001	0.0924
3	q_{33}	0.05	0	0.001	0.0926
3	q_{42}	0.05	0	0.01	0.0923
4	q_{13}	0.05	0	0.04	0.09257
4	q_{23}	0.05	0	0.004	0.0958
4	q_{33}	0.05	0	0.004	0.09244
4	q_{42}	0.05	0	0.04	0.0923
5	q_{13}	0.051	0.001	0	0.09229
5	q_{23}	0.051	0.001	0.001	0.0923
5	q_{33}	0.048	0.001	0.001	0.09227
5	q_{42}	0.05	0.001	0.01	0.09226

Una vez estabilizado se obtuvo la respuesta que se ve en la Figura 2.19, donde se puede observar como la señal ya se estabiliza en la referencia en todos los cilindros, este paso se encuentra detallado en el Anexo VI, donde se muestra las principales razones y criterios para llegar hasta la prueba 5 donde se establece los términos para el controlador interno.

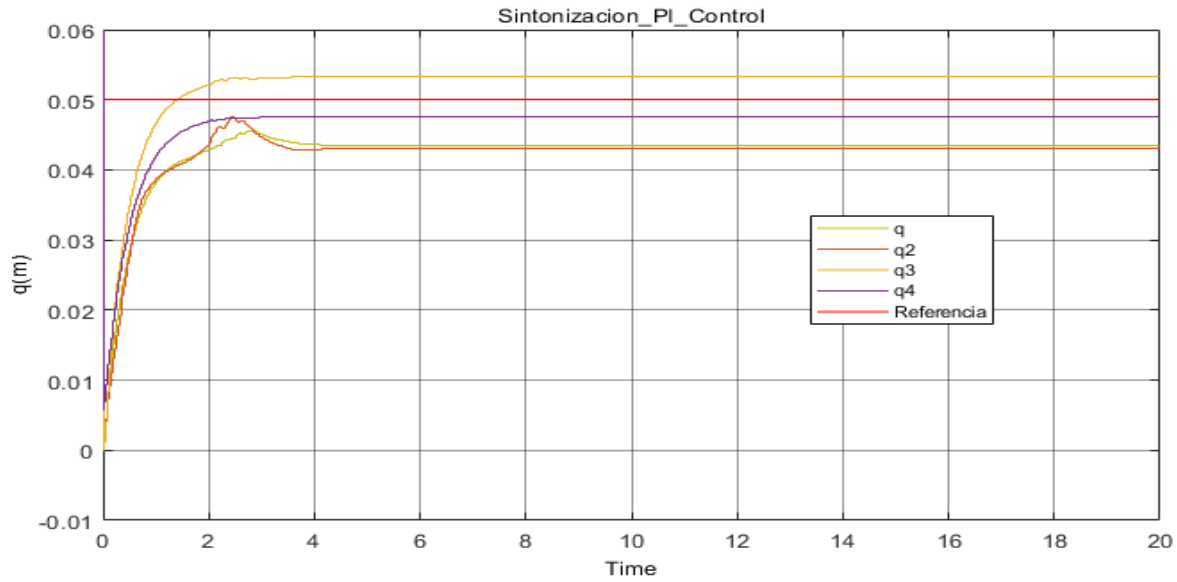


Figura 2.18. Posición de las articulaciones prismáticas para Kp=0.01.

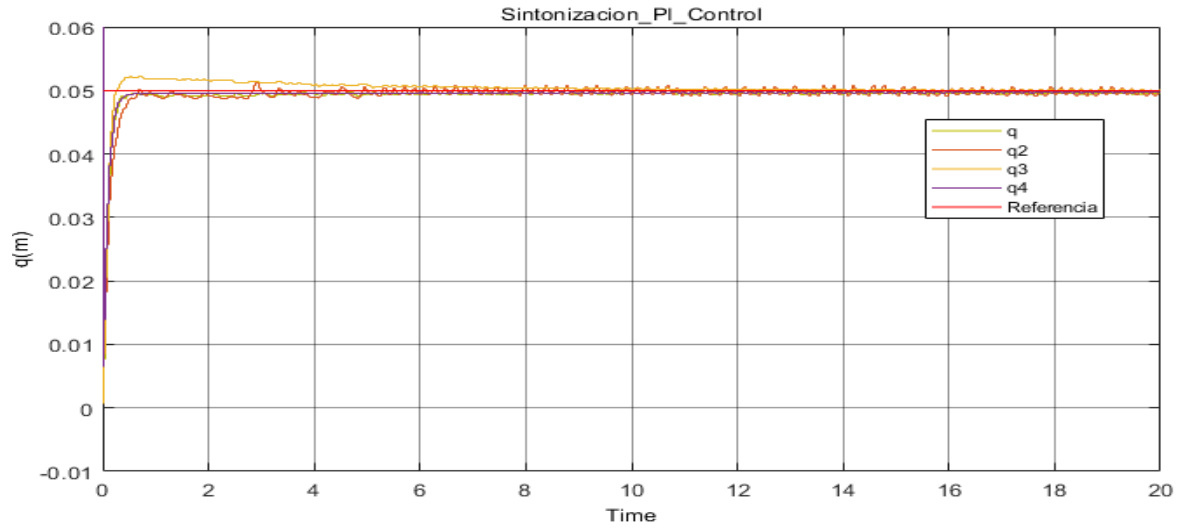


Figura 2.19. Posición de las articulaciones primaticas con la sintonización finalizada.

Finalmente, los parámetros para el controlador interno de posición se establecen en los mostrados para la prueba número 5 de la Tabla 2.6 y mostrados en la Tabla 2.7.

Tabla 2.7. Parámetros para el controlador interno de posición

Articulación	Kp	Ki	Kd
q_{13}	0.051	0.001	0
q_{23}	0.051	0.001	0.001
q_{33}	0.048	0.001	0.001
q_{42}	0.05	0.001	0.01

2.4.3 BLOQUE PARA EL CONTROL DE TRAYECTORIA

El bloque del controlador se encontrará formado por el controlador de trayectoria, y el controlador interno de posición. Cada uno de estos tienen como entrada la posición y orientación de la plataforma móvil, el viene dado por los valores de x,z y .

Para el control de trayectoria se implementa la siguiente ley de control:

$$\dot{x} = inv(J)(\ddot{x}_{ref} + K_v tanh(e_v) + K_p \tanh(e)) \quad (2.21)$$

Donde

J: es la matriz Jacobiana para el modelo cinemático del robot 3UPS+RPU.

K_p: matriz de constantes de proporcionalidad.

K_v: matriz de constantes de velocidad.

e: error de posición

e_v: error de velocidad

$\dot{x}$: velocidad de la articulación prismática

$\ddot{x}_{ref}$= aceleración de referencia.

Para establecer la relación entre K_v y K_p [18] utiliza la relación de la Ecuación (2.22) mediante la cual se logra aproxima un equilibrio entre control de posición y velocidad.

$$Kv = 2\sqrt{K_p} \quad (2.22)$$

2.4.3.1 Sintonización de Kv y Kp

Para la sintonización de los parámetros del controlador se utilizaron las herramientas que brinda MATLAB. Como primer paso se obtuvo la función de transferencia para la planta, y dado que la plataforma móvil de se mueve únicamente en el plano xz, se obtuvo la función de transferencia para sus correspondientes ejes.

La obtención de las funciones de transferencia es importante, ya que estas son utilizadas para la sintonización de los parámetros Kv y Kp que utilizará el controlador de trayectoria. En el proceso mostrado en la Figura 2.20, se utiliza como entrada la posición cartesiana de la plataforma móvil y por medio de la cinemática inversa se envía la posición de cada una de las cuatro patas que conforman el robot 3UPS + RPU.

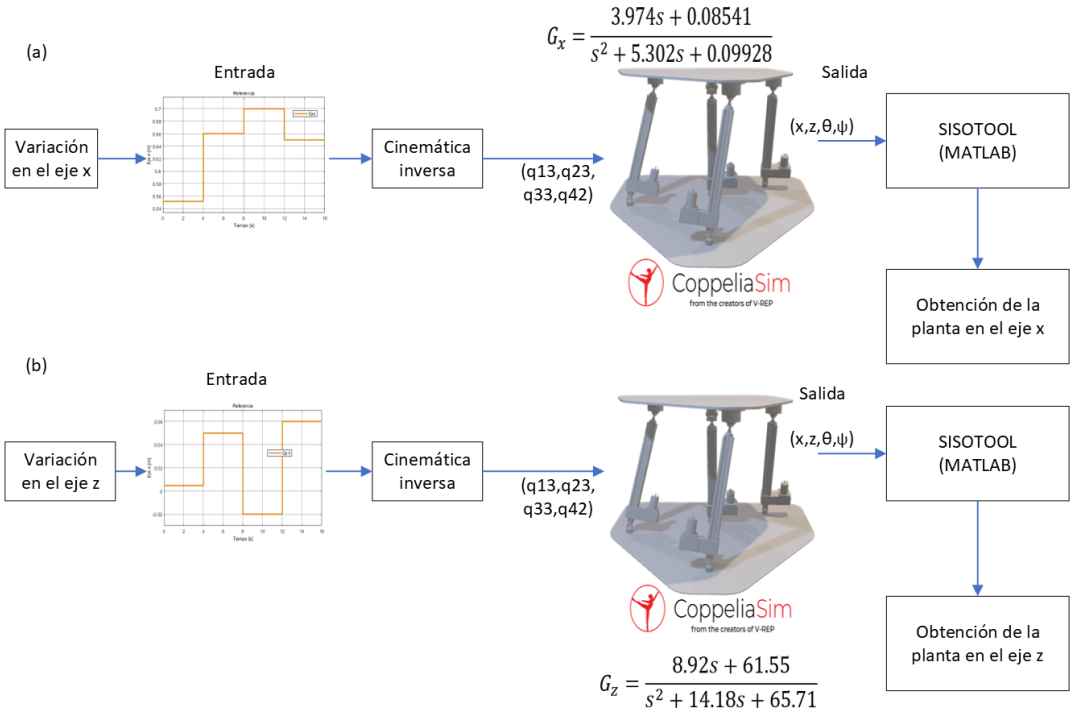


Figura 2.20. (a) Proceso de obtención de la función de transferencia $\boldsymbol{G_x}$, (b) Proceso de obtención de la función de transferencia $\boldsymbol{G_z}$.

En la obtención de la función de transferencia se utilizó la función ident de MATLAB, inicialmente se prueba la planta para una señal de entrada y se observa la respuesta de la planta, como se representa en las Figuras 2.20 y 2.21. Por medio del procedimiento presentado en el diagrama de flujo de la Figura 2.20 se halla las funciones de transferencias G_x y G_z, las mismas que son usadas para encontrar los parámetros tanto de Kv y Kp en los ejes x y z.

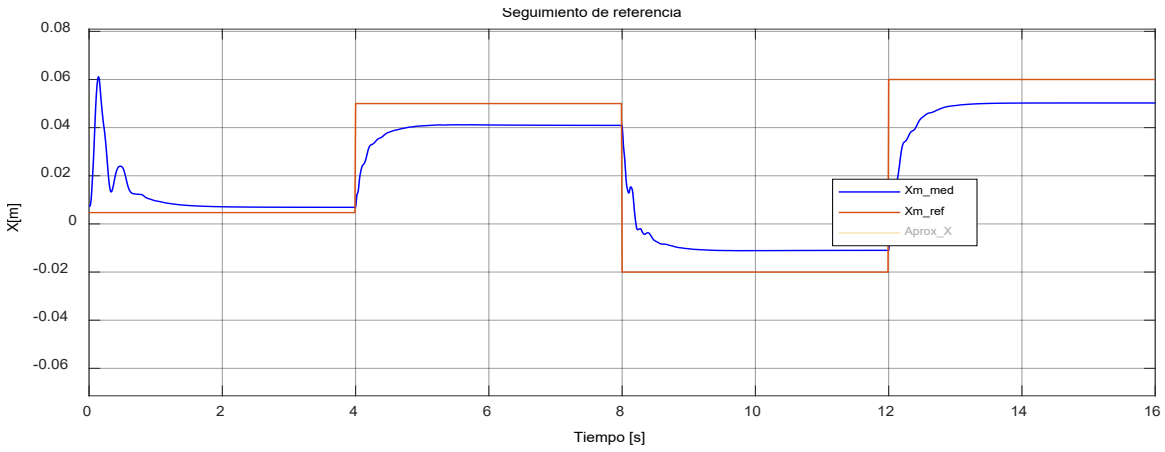


Figura 2.21. Respuesta, en el eje x, del robot 3UPS+RPU ante una señal de entrada.

Para obtener un mejor resultado en la obtención de la planta, al enviar una señal en la entrada del robot se trabaja únicamente con un solo eje, es decir, si variamos la entrada en el eje x se mantiene constante la entrada en z obteniendo de esta manera su función de transferencia y repitiendo el mismo procedimiento para obtener la función de transferencia en el eje z.

Los datos obtenidos de las Figuras 2.21 y 2.22 son enviados a la herramienta ident de MATLAB como se muestra en la Figura 2.23. Esta herramienta permite obtener la función de transferencia de una planta a partir de los datos de la respuesta de la misma ante una entrada.

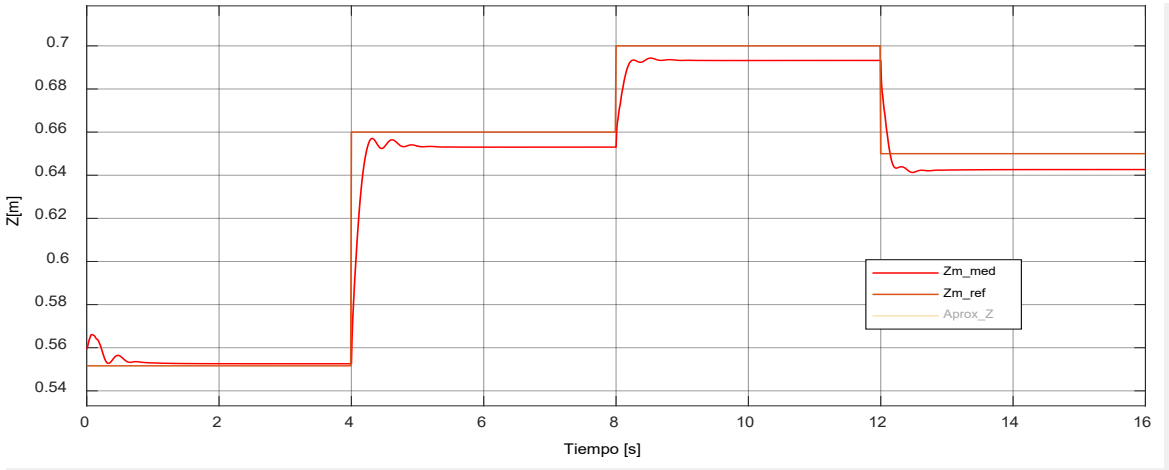


Figura 2.22. Respuesta, en el eje z, del robot 3UPS+RPU ante una señal de entrada.

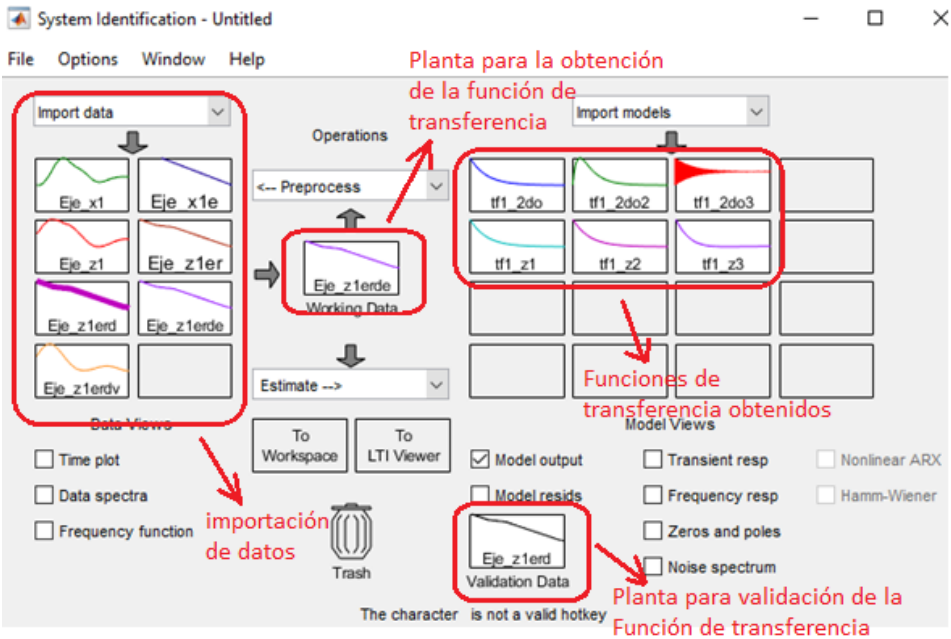


Figura 2.23. Herramienta ident de MATLAB para la obtención de las funciones de transferencia en el eje z y x.

Las funciones de transferencias obtenidas son G_x y G_z correspondientes a las siguientes ecuaciones (2.23) y (2.24) respectivamente. Estas ecuaciones representan la aproximación del comportamiento del movimiento de la plataforma móvil en el eje x y en el eje z obtenidas a través del uso de MATLAB, mismo que se muestra en las Figuras 2.22, 2.23 y 2.20.

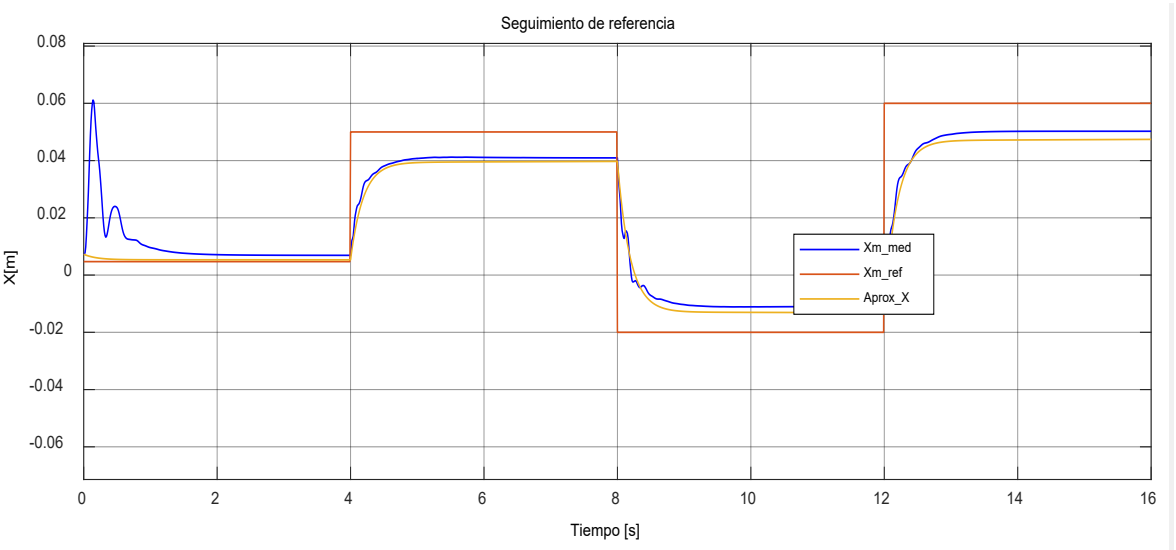


Figura 2.24. Validación de la planta para la función de transferencia $\boldsymbol{G_x}$

$$G_x = \frac{3.974s + 0.08541}{s^2 + 5.302s + 0.09928} \qquad (2.23)$$

$$G_z = \frac{8.92s + 61.55}{s^2 + 14.18s + 65.71} \qquad (2.24)$$

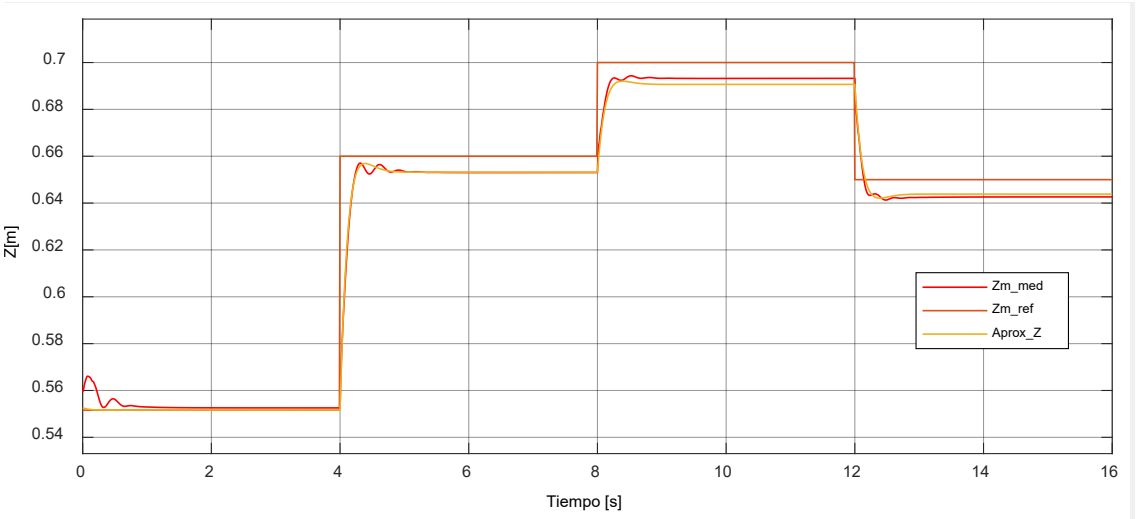


Figura 2.25. Validación de la planta para la función de transferencia $\boldsymbol{G_z}$

Finalmente, para obtener las constantes del controlador se utiliza la herramienta sisotool, aquí se cargan las funciones de transferencia de la Ecuación (2.23) y (2.24). Únicamente nos interesa la parte proporcional, como se observa en la Figura 2.24 y Figura 2.24, para obtener la parte derivativa por medio de la Ecuación (2.22).

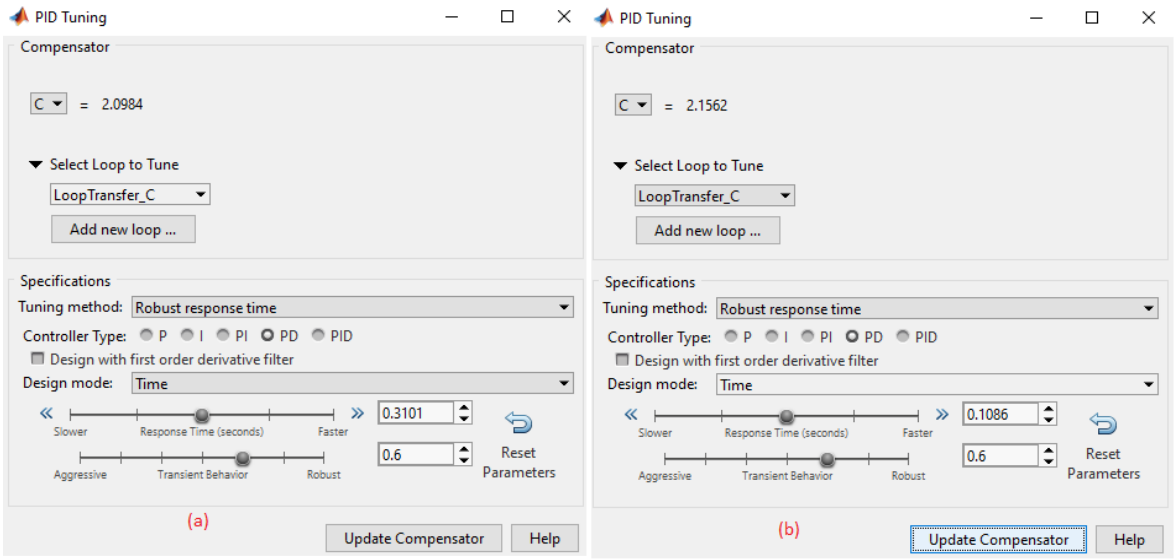


Figura 2.26. Sintonización de controlador P (a) eje x; (b) eje z.

En base a la Figura 2.24 y de la Ecuación (2.22) se presentan los resultados del controlador en la Tabla 2.8 correspondientes a los valores de la diagonal de la matriz de las Ecuaciones (2.25) y (2.26).

$$K_p = \begin{bmatrix} 2.098 & 0 \\ 0 & 2.1562 \end{bmatrix} \quad (2.25)$$

$$K_v = \begin{bmatrix} 2.897 & 0 \\ 0 & 2.92 \end{bmatrix} \quad (2.26)$$

Tabla 2.8. Parámetros del controlador de trayectoria PD.

	Parámetros del controlador	
Eje	Kp	Kv
x	2.0984	2.897
z	2.1562	2.92

El controlador con los parámetros k_p y K_v hallados hasta ahora necesitan de una sintonización manual y rigurosa, por lo que para este fin se necesita se prueba el controlador y se observa su error de posición para una trayectoria dada en la Figura 2.27 obteniendo resultados como los de la Figura 2.28, donde se manifiesta un ligero error al inicio de la trayectoria, esto se arregló únicamente aumentando con el método de prueba

error el valor de K_p correspondiente al eje x hasta reducirlo y obtener un seguimiento de trayectoria satisfactorio visualmente. Finalmente, se estableció los parámetros del controlador en los mostrados en las Ecuaciones (2.27) y (2.28).

$$K_p = \begin{bmatrix} 3.3 & 0 \\ 0 & 2.1562 \end{bmatrix} \quad (2.27)$$

$$K_v = \begin{bmatrix} 2.897 & 0 \\ 0 & 2.92 \end{bmatrix} \quad (2.28)$$

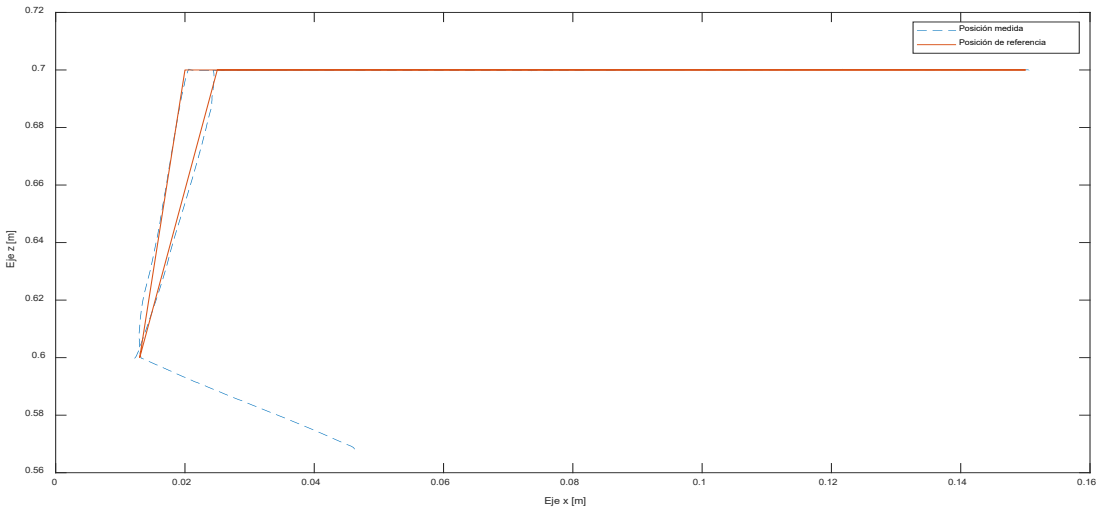


Figura 2.27. Trayectoria para la Prueba del controlador.

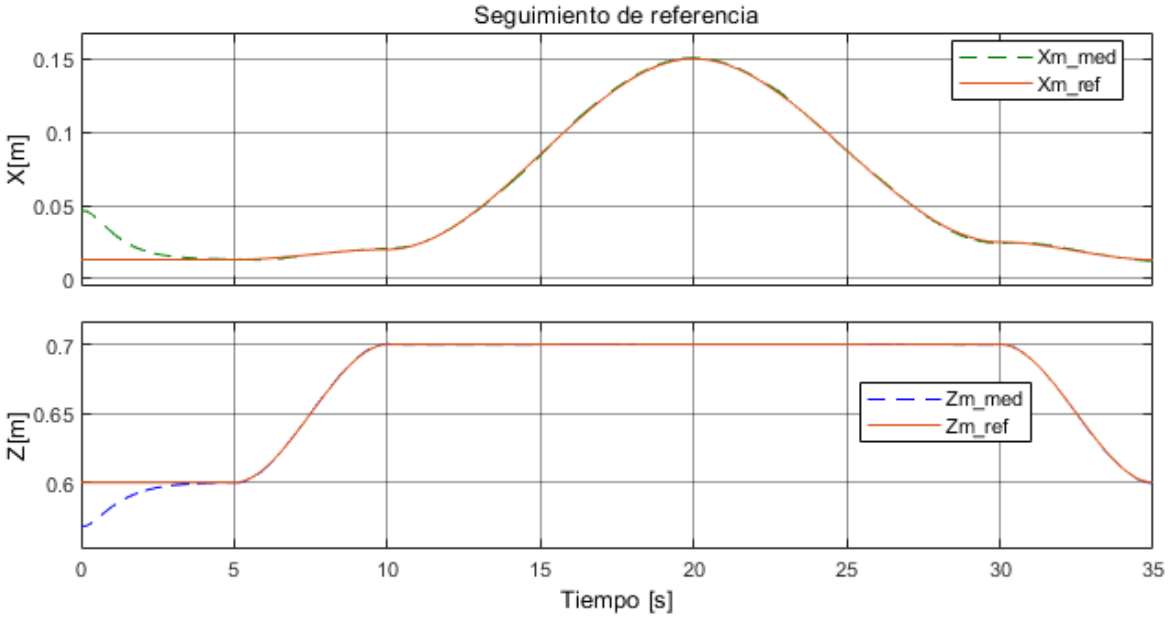


Figura 2.28. Controlador de trayectoria luego de la sintonización.

2.4.4 BLOQUE DE COMUNICACIÓN

El bloque de comunicación con la CoppeliaSim es un bloque S-Function de nivel 1 de MATLAB, mismo que fue explicado en la Sección 1.4.2.1.1. Este bloque permite dividir sus actividades de acuerdo con los estados de simulación, es decir tiene funciones que se ejecutan al inicio de la simulación, al terminar la simulación y una que se repite en cada paso de simulación.

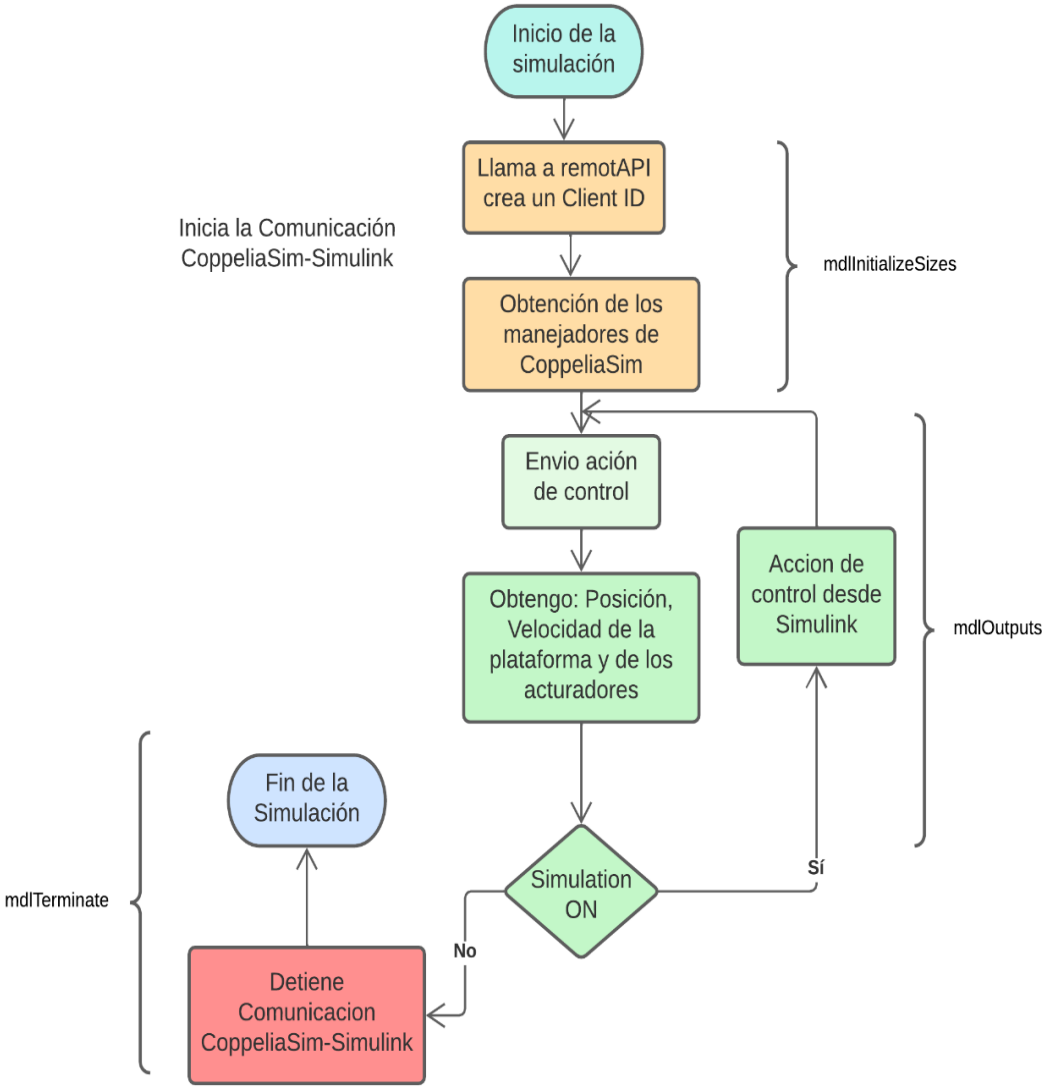


Figura 2.29. Diagrama de Flujo del funcionamiento del bloque de comunicación CoppeliaSim-Simulink.

En la función de inicialización `mdlInitializeSizes` se establece el número de entradas y salidas de la función, la conexión con CoppeliaSim, el modo de comunicación y se

obtienen los manejadores de las articulaciones prismáticas o independientes del robot y también de las referencias de la plataforma móvil.

En la función `mdlOutputs` se aplica la acción de control a cada uno de los eslabones, posteriormente se obtiene la realimentación para el sistema los cuales consisten en la posición de la plataforma móvil, posición de las 4 articulaciones prismáticas y su respectiva velocidad. Esta función es la que se repite en cada paso de simulación.

Finalmente, en la función de finalización `mdlTerminate` se detiene la simulación y se cierra la comunicación con CoppeliaSim para lo cual se destruye el nodo establecido y se lo elimina. El funcionamiento de este bloque se explica forma gráfica en el diagrama de flujo de la Figura 2.2.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Las pruebas realizadas tienen el objetivo de probar el modelo dinámico virtual creado en CoppeliaSim, así como también sintonizar de forma adecuada el controlador interno dentro del entorno de CoppeliaSim. También se pretende el desempeño y sintonización del controlador de trayectoria para la plataforma móvil del robot 3 UPS+RPU.

3.1 PRUEBAS Y RESULTADOS DEL MODELO DINÁMICO VIRTUAL A LAZO ABIERTO

Una parte fundamental del proyecto es la validación del modelo virtual desarrollado en CoppeliaSim con el modelo físico. Para este punto se mide la posición y orientación de la plataforma cuando todos sus vástagos se encuentran en la posición 0, obteniendo la posición mostrada en el vector P y la orientación del vector O correspondiente a las Ecuaciones 3.1 y 3.2 respectivamente.

$$P = [0.0561 \quad 0 \quad 0.7035 \; 0.603][m] \tag{3.1}$$

$$O = [0 \quad -7.82 \quad 30.32][°] \tag{3.2}$$

Mientras que las medidas tomadas del modelo real son las correspondientes a las Ecuaciones 3.3 y 3.4.

$$P = [0.02272 \quad 0 \quad 0.5812][m] \tag{3.3}$$

$$O = [0 \quad -7.45 \quad 25.59][°] \tag{3.4}$$

Se puede evidenciar un error de posición en x de 0.033 m y en el eje z de 0.0218 m, mientras que en orientación un error de 0.37° para el ángulo en θ y para ψ un error de

4.73°, esto se debe principalmente a errores mecánicos en la construcción de las articulaciones esféricas.

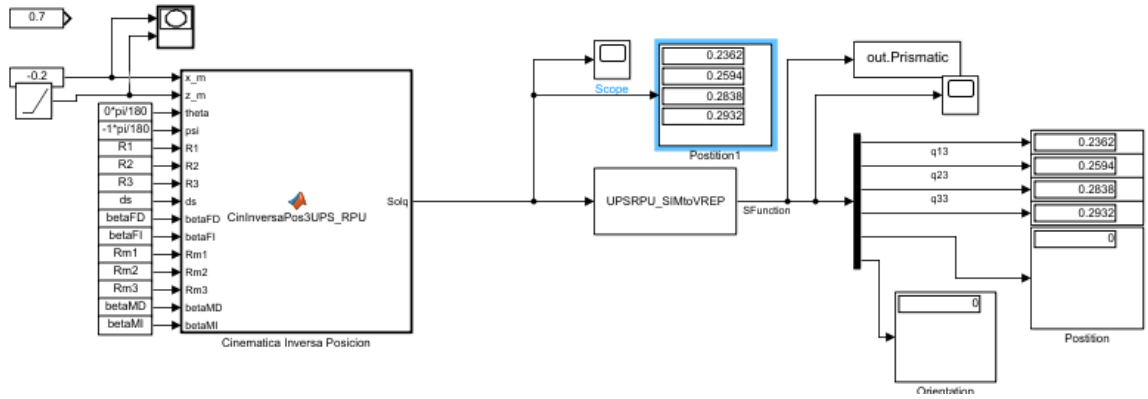


Figura 3.1. Pruebas de lazo abierto del modelo virtual del robot.

Para las pruebas en lazo abierto se utilizó el modelo cinemático inverso con el bloque de comunicación con CoppeliaSim, como se muestra en la Figura 3.1, para enviar una posición variable en z y en el eje x.

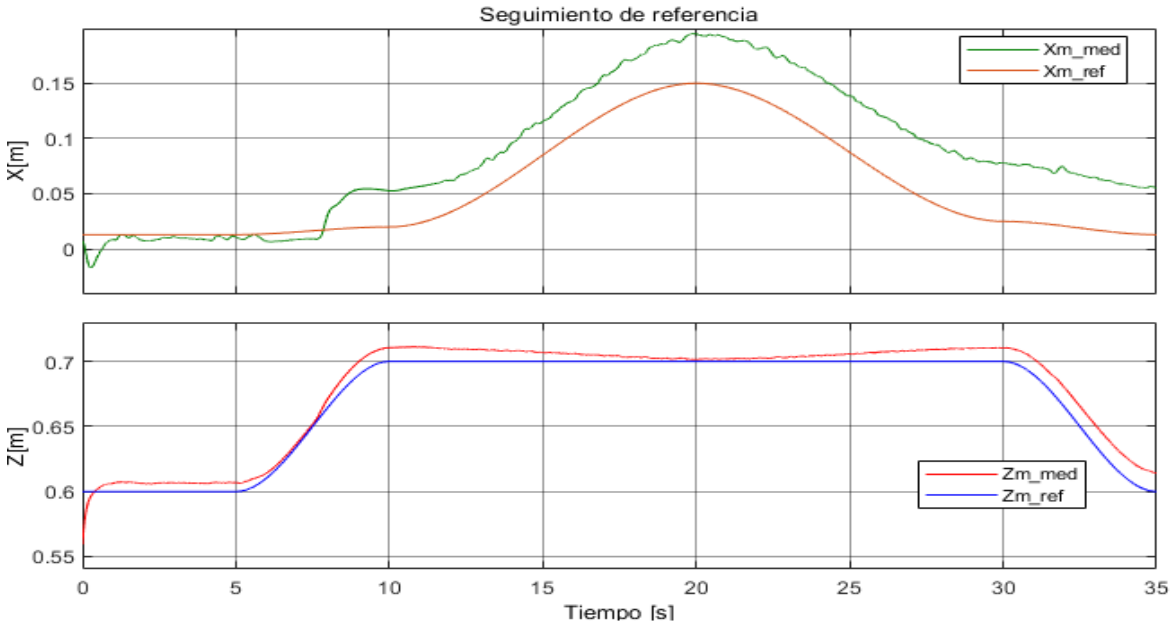


Figura 3.2. Posicionamiento de la plataforma móvil en la prueba de lazo abierto.

Como se pude observar en la Figura 3.2, los resultados obtenidos del lazo abierto del robot paralelo virtual presentan errores significativos en su posición tanto en el eje x como en el eje z. Estos errores evidentes de posición en lazo abierto son esperados, dado que todo lo que se trata de un robot paralelo donde el movimiento de un eslabón influye directamente en la posición de la plataforma móvil, y la razón más importante del error es que la posición se envía por medio de cinemática inversa observado en la Figura 3.3 este modelo no

contempla aspectos como los errores de posición, es por esto que se presencia un error de posición en el eje x y un desplazamiento en el eje z.

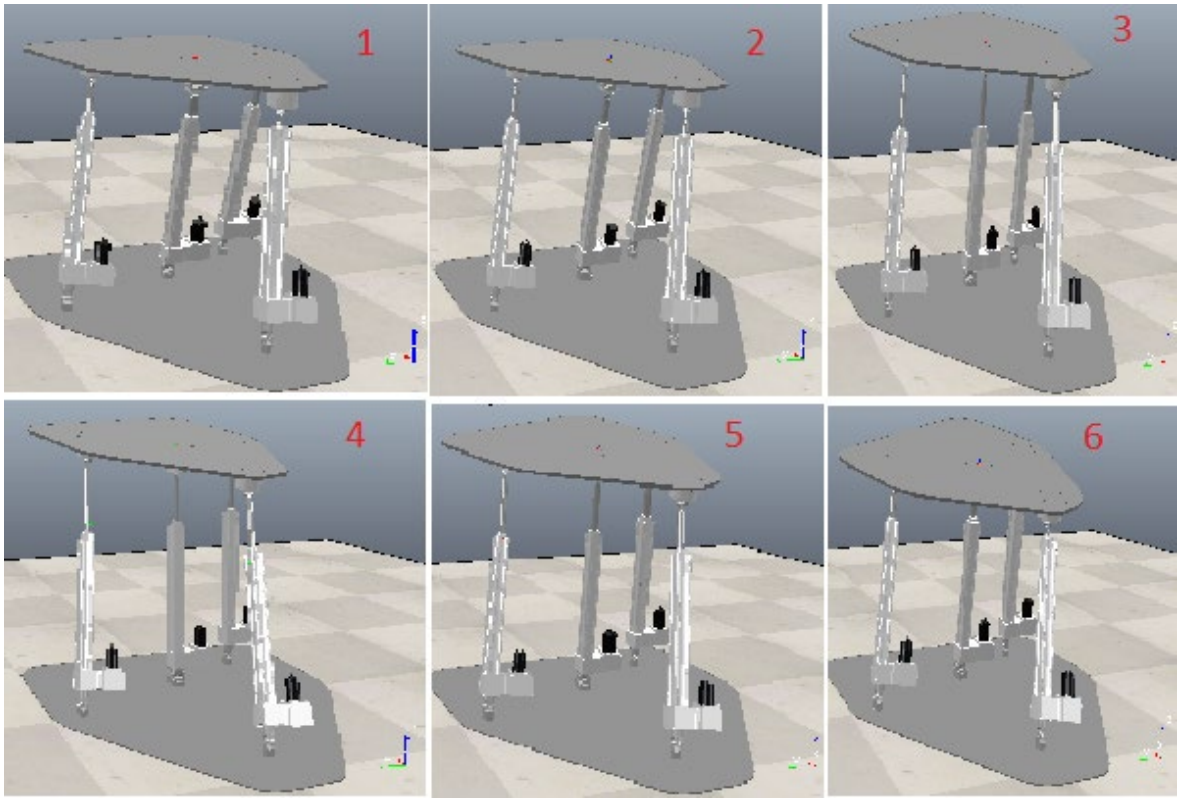


Figura 3.3. Comportamiento del modelo dinámico virtual en CoppeliaSim.

En la Figura 3.3 se puede evidenciar el comportamiento del robot virtual para la referencia de posición en el tiempo para la referencia de la Figura 3.2, donde se pude observar como influye la velocidad de la trayectoria, dado que al aumentar el valor de posición en el eje z se presenta un error considerable en el eje x.

3.2 PRUEBAS Y RESULTADOS DE LOS CONTROLADORES

3.2.1 CONTROLADOR PID INTERNO:

Para las pruebas de controlador del lazo interno se usa la terminal de CoppeliaSim y MATLAB, este último envía la referencia, como se muestra en la Figura 3.4.

En la Figura 3.4 se observa el comportamiento del robot virtual frente a una señal de entrada escalonada, esta misma permite ver el comportamiento de los controladores mismos que presentan una respuesta aceptable ya que no presenta oscilaciones al estabilizarse, de igual manera se verificar en su bajo coeficiente ISE en cada una de las patas en la Tabla 3.1.

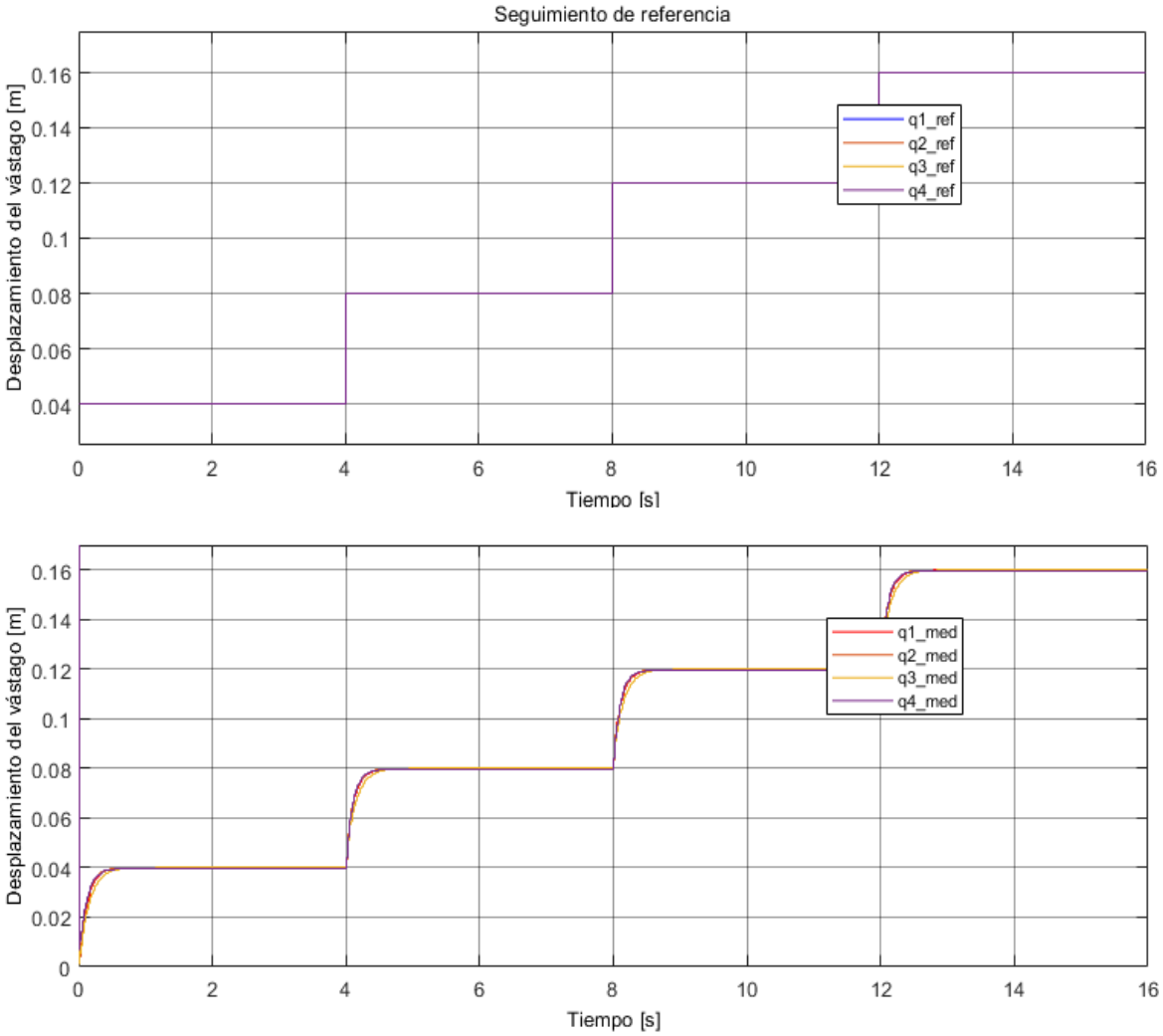


Figura 3.4. PID interno de posición para cada uno de las patas del robot**.**

Tabla 3.1. Rendimiento del controlador interno de posición

	Articulación del robot			
	q_{13}	q_{23}	q_{33}	q_{42}
ISE	0.0943	0.09425	0.09481	0.09428

3.2.2 PRUEBAS DEL CONTROLADOR DE TRAYECTORIA.

Una vez probado el funcionamiento de los modelos virtuales y de los controladores internos de los eslabones, ya es posible comprobar el funcionamiento del controlador de trayectoria. Inicialmente para probar el controlador de trayectoria se establecieron los parámetros del controlador, mismos que están en las Ecuaciones 2.27 y 2.28.

Para estas pruebas es importante mencionar las características de la máquina o computadora donde se realizarán dichas pruebas, las cuales son mencionadas en la Tabla, ya que esto influirá en los resultados de estas.

Tabla 3.2. Especificaciones de la computadora donde se realizarán las pruebas.

Especificaciones del dispositivo	Detalles
Procesador	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
RAM instalada	16 GB.
Tipo de sistema	Sistema operativo de 64 bits, procesador basado en x64.
Tarjeta Gráfica	Ninguna
Tipo de Disco duro	Mecánico
Sistema operativo	Windows10 Education. Experiencia: Windows Feature Experience Pack 120.2212.4180.0

CoppeliaSim admite varios motores de simulación de la dinámica en su respectivo módulo, para este proyecto se usa el motor de física Newton. Este presenta un rango de paso de tiempo de simulación, lo cual no es más que el tiempo más pequeño en un ciclo de simulación, el cual garantiza una buena ejecución. Al salir de dicho rango los datos de la dinámica como la velocidad, fuerza o aceleración no son las adecuadas y el valor de paso de tiempo de simulación es dado por default por la aplicación de CoppeliaSim. Este debe ser lo suficientemente pequeña para asegurar la estabilidad física del robot simulado [15], de tal manera que también puede variar de acuerdo a las características de la computadora donde se simula. Para este trabajo esas características se mostraron en la Tabla 3.2.

Trayectoria 1

La primera trayectoria empleada con el controlador es un círculo, el mismo que es mostrado en la Figura 3.5, como característica principal de esta trayectoria es la suavidad con la que realiza los desplazamientos, tanto en el eje x como en el eje z.

El control de la posición en el eje de las x, es el que más trabajo costó sintonizarlo debido que es imposible aislar completamente la posición en los dos ejes de movimiento que presenta la plataforma móvil. Es por esta razón que se observan como la velocidad y posición en este eje presentan una especie de ruido en algunas partes de la trayectoria.

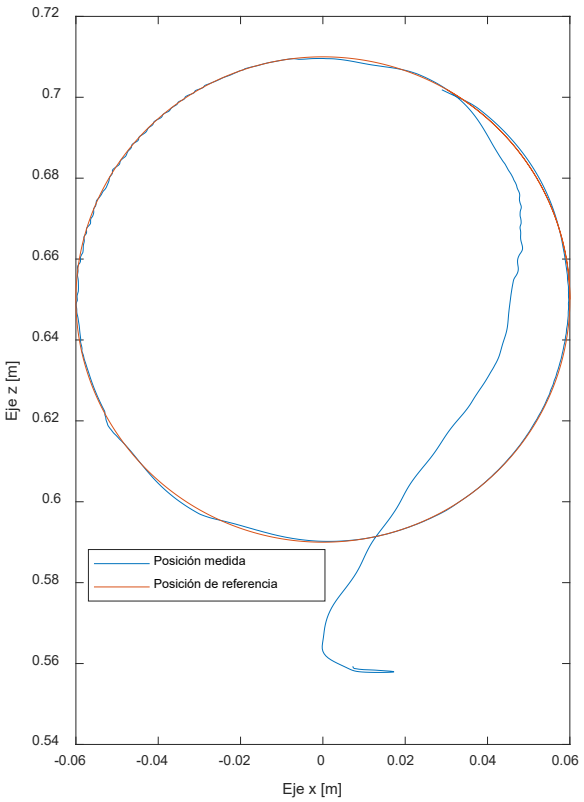


Figura 3.5. Trayectoria 1, círculo.

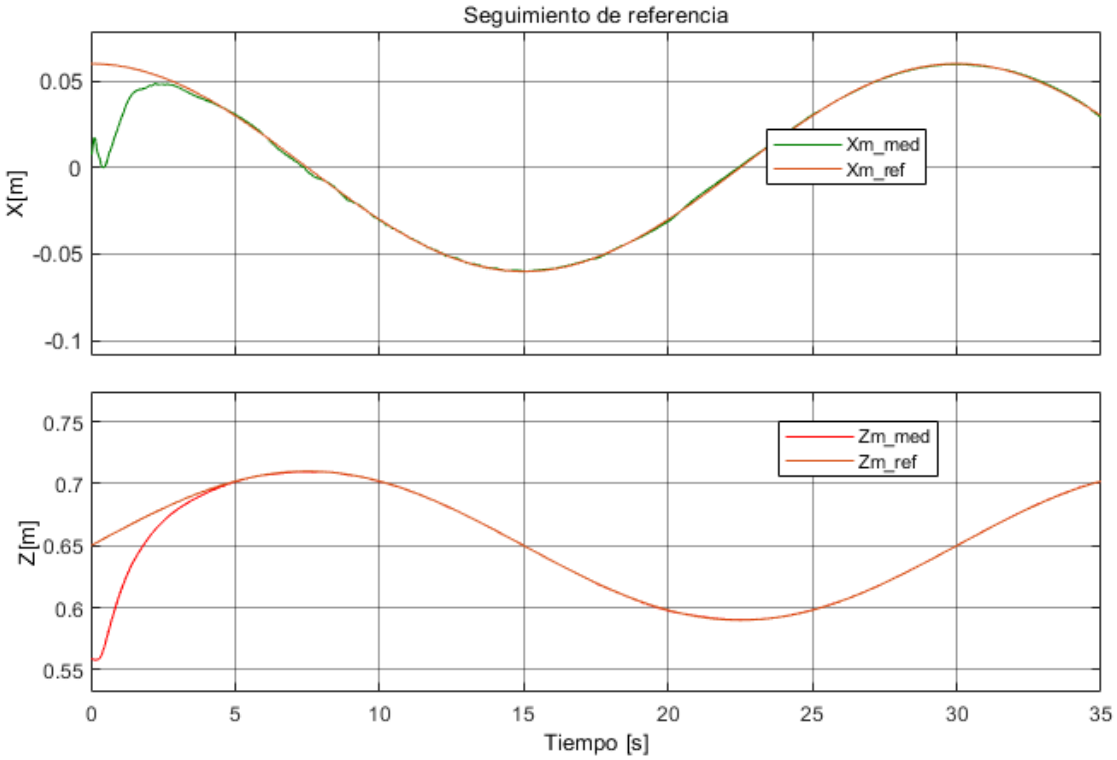


Figura 3.6. Seguimiento de trayectoria en los ejes x y z para la trayectoria1.

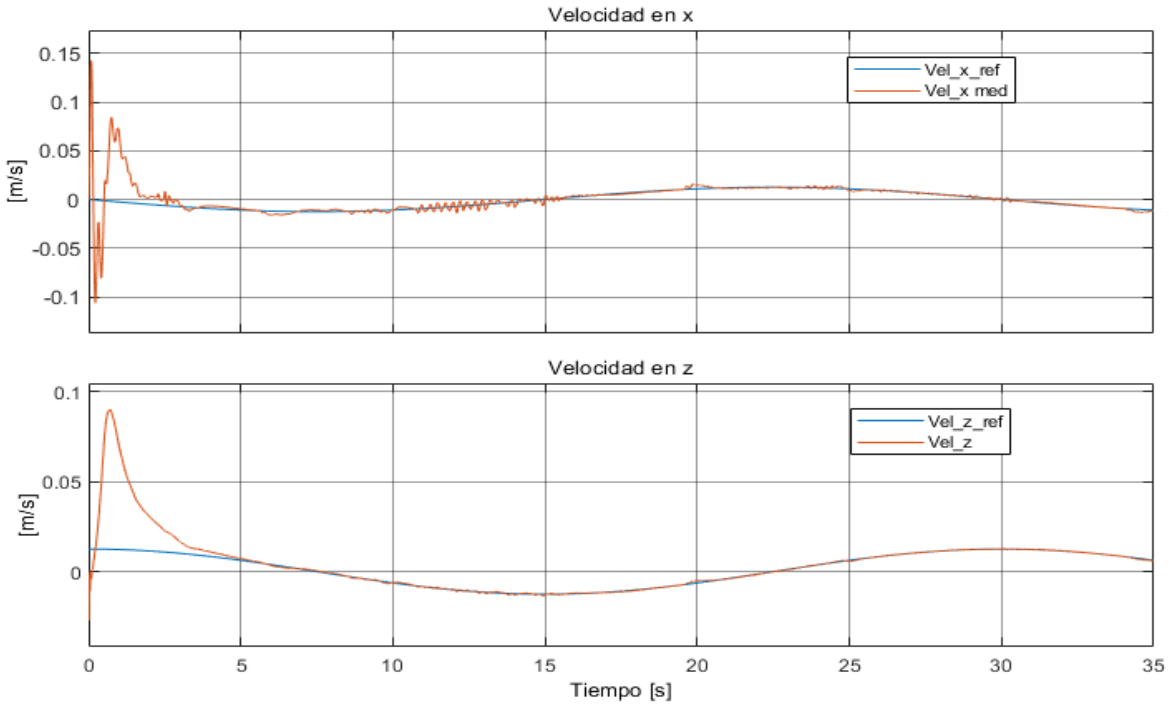


Figura 3.7. Seguimiento de velocidad para la trayectoria1

De igual manera que el movimiento en los ejes, así también se observa en la Figura 3.8 una trayectoria fluida, sin muchos movimientos bruscos al contrario de lo que sucede en la velocidad en la Figura 3.7, sin embargo, esto no afecta al seguimiento de trayectoria.

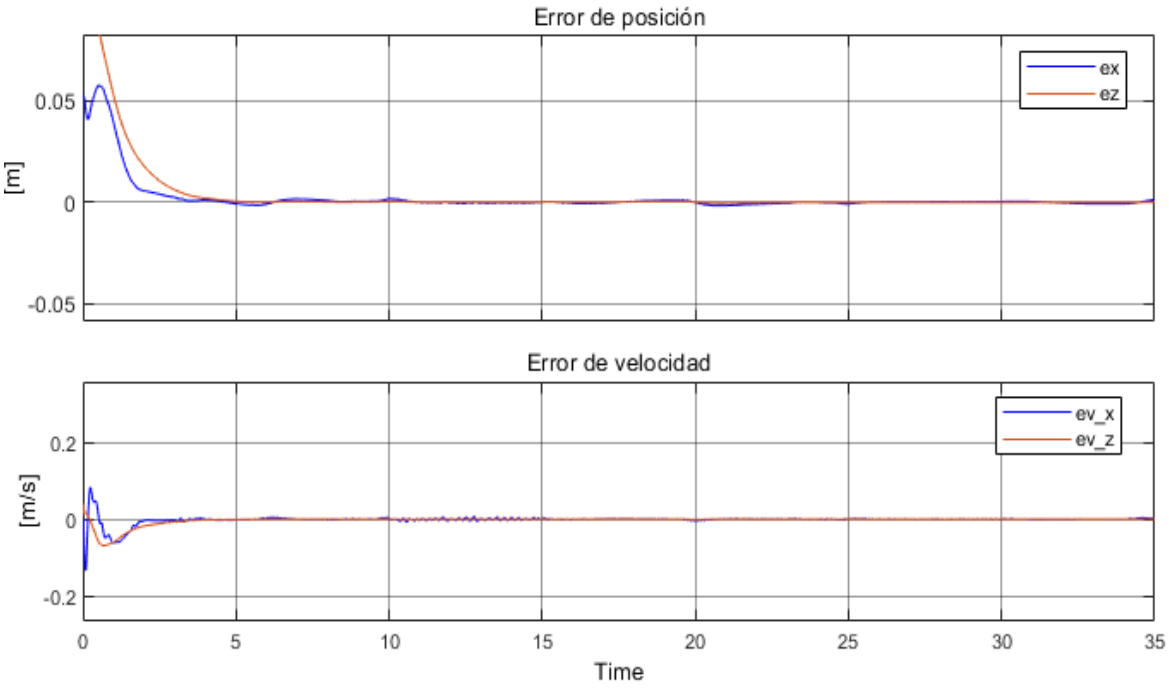


Figura 3.8. Error de posición y velocidad en la trayectoria 1.

Como se observa en la Figura 3.8, el error de posición para la trayectoria 1 está cerca de ser cero, manteniéndose con esta tendencia durante toda la trayectoria. La velocidad en el eje x también es aceptable ya que cumple con el objetivo principal que es el de control de trayectoria.

3.2.2.1.1 Trayectoria 2

Para la trayectoria 2, mostrada en la Figura 3.9, ya se cuenta con movimientos con cambios de dirección más bruscos en comparación a la trayectoria 1, es así que se pude notar un error pequeño en estos cambios de dirección, como lo mostrado en la Figura 3.12 sin en embargo el controlador actúa reduciéndolo teniendo en el eje x un pequeño error al pasar por un punto.

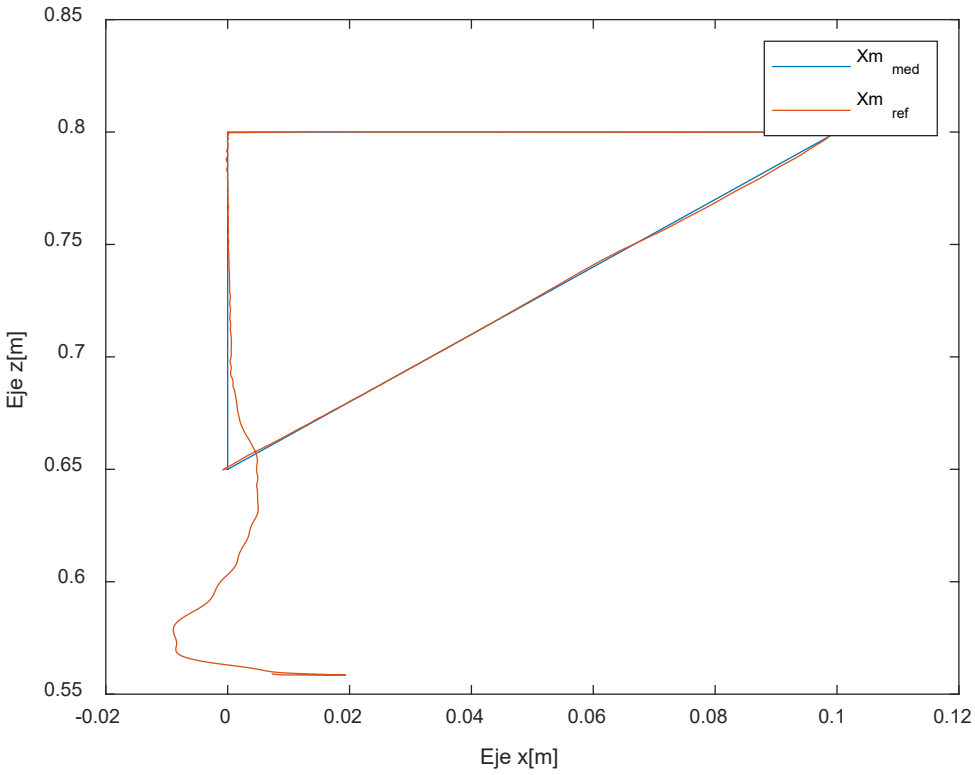


Figura 3.9. Trayectoria 2.

En la velocidad de la plataforma móvil medida para la trayectoria 2, mostrada en la Figura 3.11, se observa como el controlador realiza el seguimiento de la referencia con cierto ruido al mantenerse constante la velocidad en z, mientras que si los dos tienen referencias positivas que varían se presenta un seguimiento sin un error significativo. Por medio de las Figuras 3.12 y 3.11 se puede observar un desempeño aceptable para la trayectoria 2.

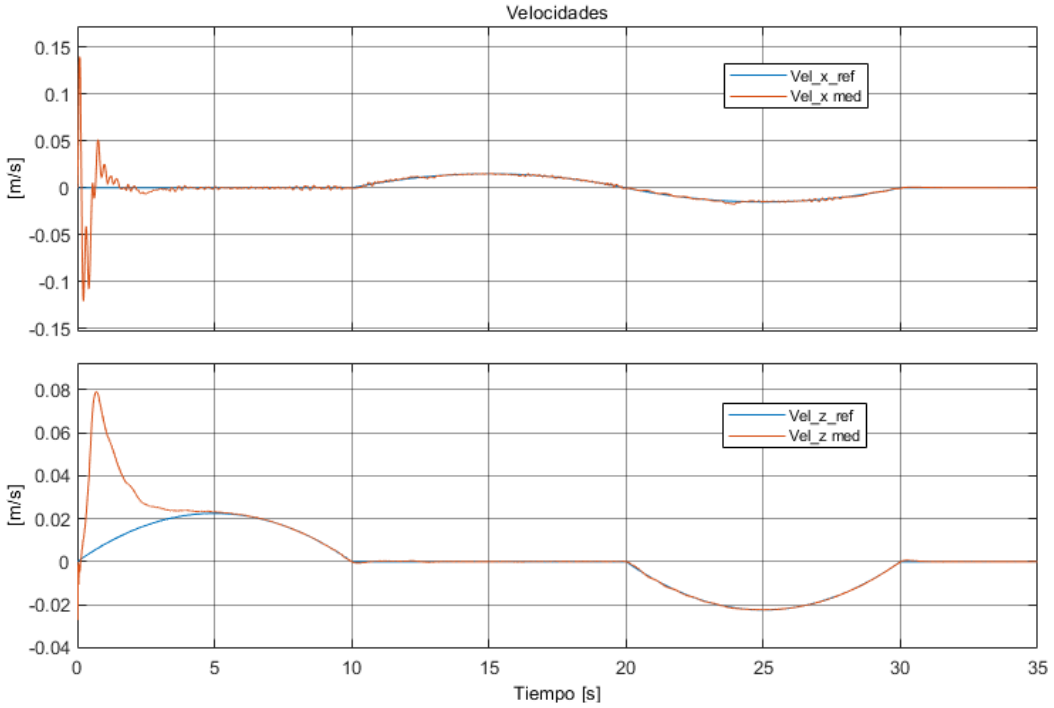


Figura 3.10. Seguimiento de velocidad para la trayectoria 2.

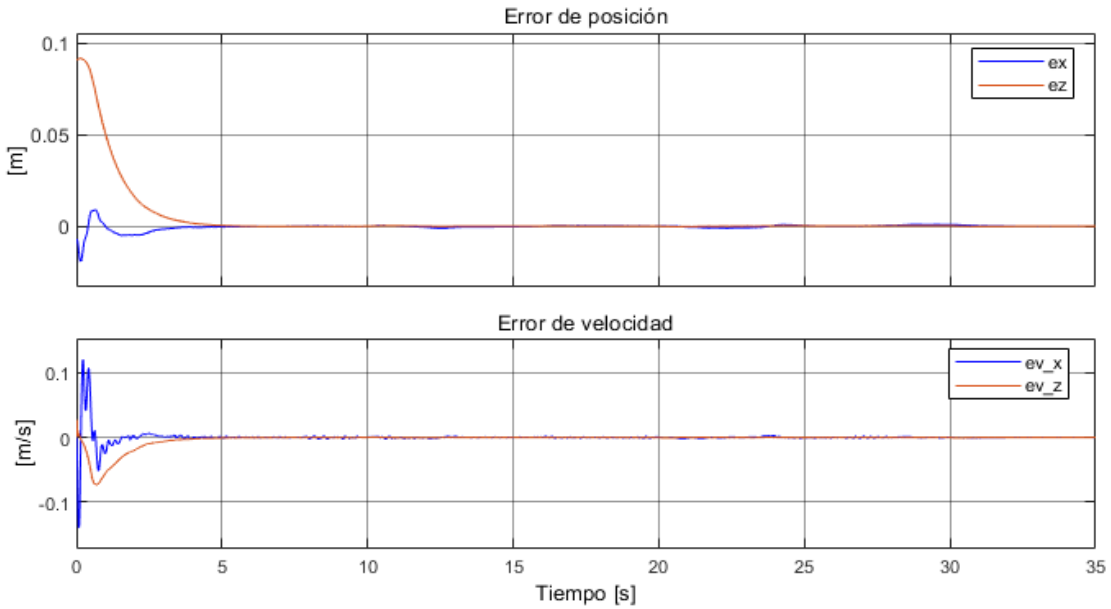


Figura 3.11. Error de posición y velocidad en la trayectoria 2.

Al verse afectada la velocidad, se puede observar en la Figura 3.12 que el error también se ve afectada, teniendo un pequeño rizo en el error de velocidad en el eje x, esto se debe a las razones ya expuestas, al inicio de esta sección, sobre el uso del motor de física Newton.

3.2.2.1.2 Trayectoria 3:

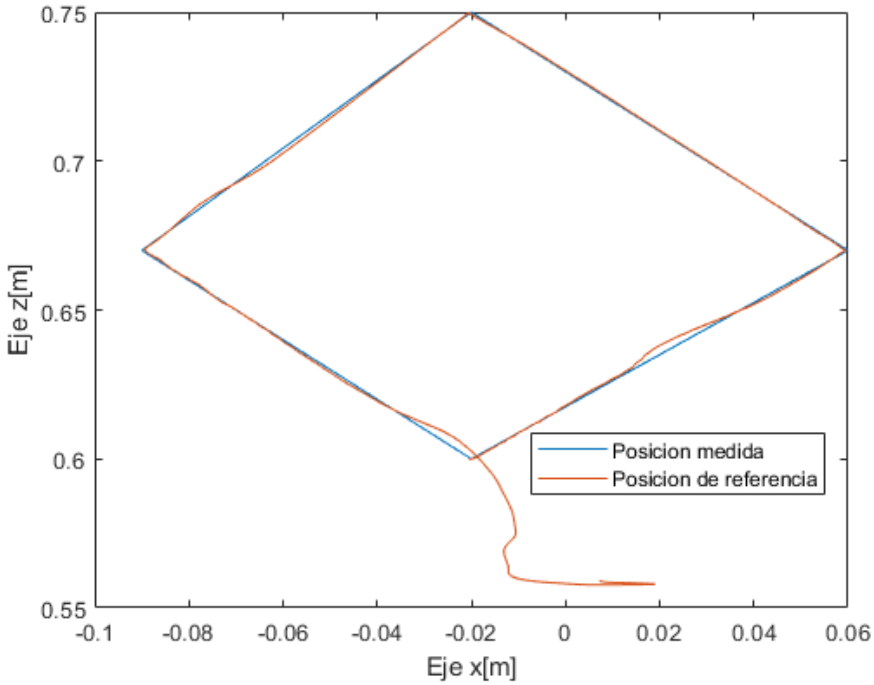


Figura 3.12. Trayectoria 3.

La trayectoria 3 de la Figura 3.13 se encuentra formada por 4 puntos, que a simple vista parecen tener un cambio un poco severo en la dirección al llegar a los extremos de la figura, pero gracias a la generación de referencia por medio de un polinomio cúbico, mostrado en la Sección 2.4.1.1, se reduce la velocidad al pasar por cada uno de estos puntos como se observa en la Figura 3.16, donde se ve que el cambio de posición en ambos ejes es suave.

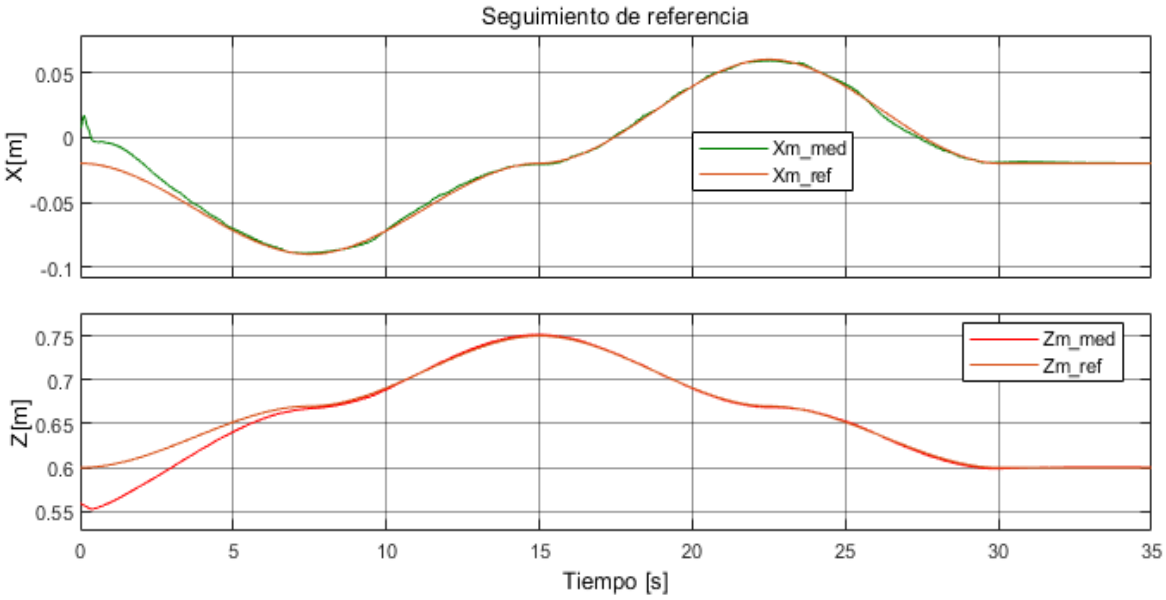


Figura 3.13. Seguimiento de trayectoria para la trayectoria 3.

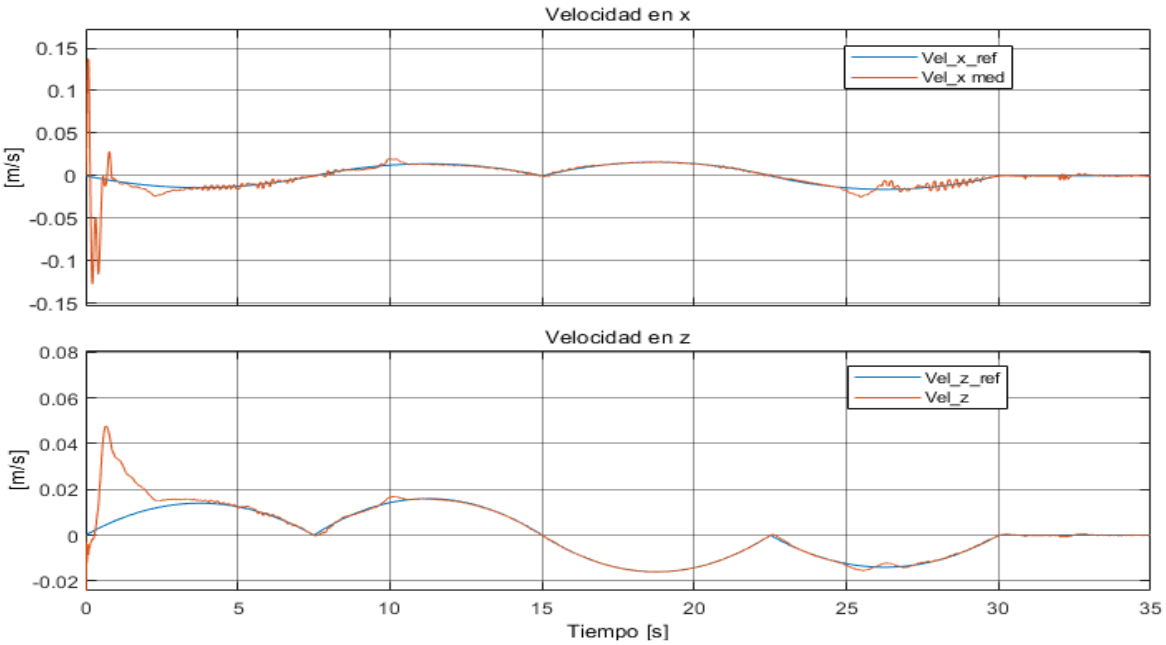


Figura 3.14. Error de velocidad para la trayectoria 3.

El error de velocidad y posición para la trayectoria 3, en la Figura 3.15, muestra la ejecución de la acción de forma suave, sin embargo, estas formas de ondas muestran un pequeño ruido causado también por el ruido en la velocidad, principalmente en el eje x, provocado por el uso del motor de simulación Newton al igual que en las anteriores trayectorias mostradas.

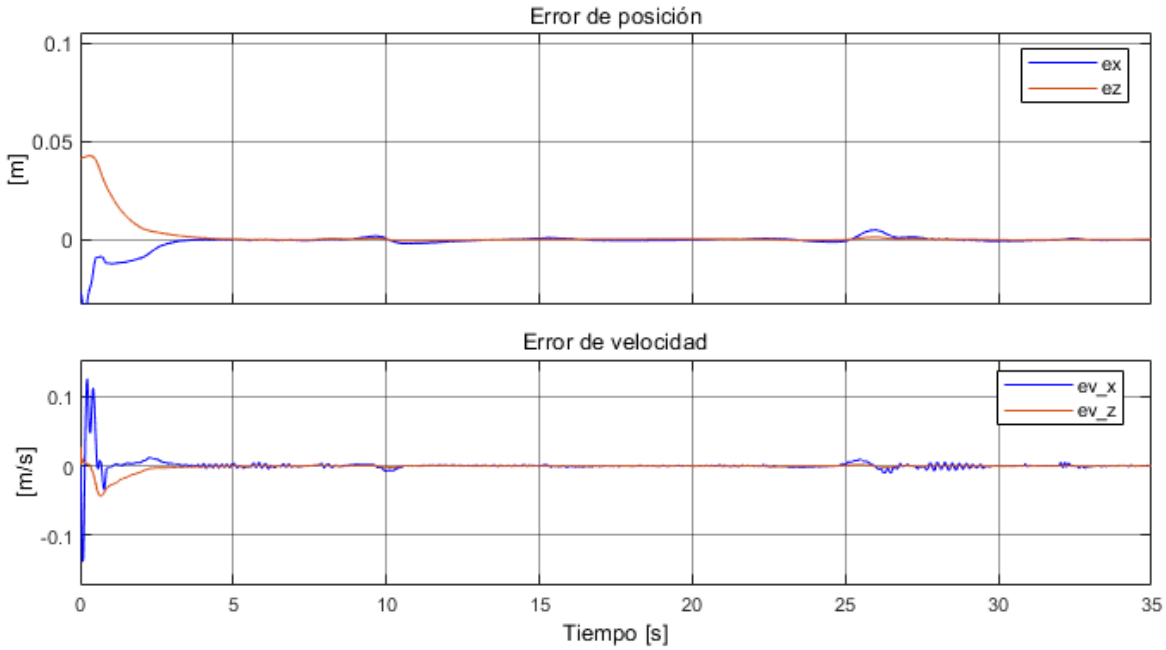


Figura 3.15. Error de posición y velocidad en la trayectoria 3.

En sí, los errores mostrados en las Figuras 3.9, 3.12 y 3.16, todas tienden a ubicarse en 0, sin embargo, existen pequeñas perturbaciones que se pude mostrar en la Figura 3.16, estas perturbaciones muestran la estrecha relación que tienen el movimiento en los ejes x y z, es decir, si se varía la posición en cualquiera de los ejes también se será afectado el otro.

3.2.3 RESUMEN DE RESULTADOS

Los resultados del proyecto se muestran en los valores obtenidos por el controlador de trayectoria diseñado para cada una de las trayectorias propuestas.

En la trayectoria 1:

Tabla 3.3. Índice ISE para el error de posición y velocidad en la trayectoria 1.

Error	Eje	ISE
Error de posición	X	0.002824
	Z	0.00782
Error de velocidad	X	0.004906
	Z	0.0037

En la trayectoria 1 los índices de resultado ISE e ISCO de la Tabla 3.3 y la Tabla 3.4 muestran el rendimiento del controlador de trayectoria para la trayectoria 1, estos resultados presentan un Índice ISE bajo por lo que se puede intuir que el controlador tiene un buen desempeño.

Tabla 3.4. Índice ISCO de las acciones de control aplicadas al robot en la trayectoria 1

Articulación	ISCO
q11	0.417
q21	0.3814
q31	0.3418
q42	0.3900

En la Tabla 3.4 muestra los índices ISCO para la trayectoria circular, precisamente estos son los más bajos en comparación a las demás trayectorias debido que esta es la más simple de todas ya que no presenta cambios severos en el sentido de su trayectoria como se puede evidenciar con las otras dos trayectorias.

Trayectoria 2:

Tabla 3.5. Índice ISE para el error de posición y velocidad en la trayectoria 2.

	Eje	ISE
Error de posición	X	0.0003666
	Z	0.001438
Error de velocidad	X	0.00298
	Z	0.0008503

Tabla 3.6. Índice ISCO de las acciones de control aplicadas al robot en la trayectoria 2

Articulación	ISCO
q11	0.4219
q21	0.3821
q31	0.3322
q42	0.3866

En la trayectoria 2 los índices ISE mostrados en la Tabla 3.5 para la posición de la plataforma móvil son bajos, mostrando de esta manera que el controlador colocado tiene poco error y se podría considerar como satisfactoria el desempeño del controlador de trayectoria.

De igual forma se muestran los resultados de los índices ISCO en la Tabla 3.6, donde se observa que la acción de control no es muy severa, es decir, su acción de control se encuentra como aceptable, ya que no muestra pulsos excesivos en la acción de control.

Trayectoria 3:

Tabla 3.7. Índice ISE para el error de posición y velocidad en la trayectoria 3

	Eje	ISE
Error de posición	X	0.0006267
	Z	0.001591
Error de velocidad	X	0.00336
	Z	0.001113

Tabla 3.8. Índice ISCO de las acciones de control aplicadas al robot en la trayectoria 3.

Articulación	ISCO
q11	0.5216
q21	0.454
q31	0.48
q42	0.5275

Los índices ISE e ISCO mostrados en las Tablas 3.7 y 3.8, muestran un desempeño aceptable del controlador tanto en el eje z como en el eje x, que se ve verifica en la gráfica de los errores de la Figura 3.20.

3.3 CONCLUSIONES

- El robot 3UPS+RPU es un robot paralelo que fue pensado para realizar la etapa inicial de rehabilitación de rodilla, donde la trayectoria de la plataforma móvil es fundamental para realizar tareas y movimientos necesarios para rehabilitación.
- La creación de modelos virtual de un robot paralelo en el software de CoppeliaSim puede tomar dos caminos distintos, en el primer caso el modelo a desarrollarse puede ser cinemático, para este modelo de robot se puede usar el plugin de cinemática mediante los cuales se cierran las cadenas cinemáticas; mientras que en el segundo caso, el modelo dinámico, las cadenas cinemáticas pueden cerrarse por medio de enlaces dinámicos, mismo que es usado en este trabajo.
- En el modelo dinámico creado en CoppeliaSim las características dinámicas se las puede establecer al configurar los objetos dinámicos del mismo robot, dentro de esta configuración se pude establecer momentos de inercia, masa y centros de masa.
- Una de las grandes ventajas en CoppeliaSim es su flexibilidad de comunicación con aplicaciones externas, específicamente, con MATLAB-Simulink se pude comunicar por medio de una API remota legacy la cual brinda la posibilidad de realizar simulación síncrona con CoppeliaSim.
- La comunicación entre CoppeliaSim y MATLAB por medio de una API necesita de las librerías presentes en la carpeta de instalación de CoppeliaSim, por medio de estas la API establece la comunicación con MATLAB pudiendo configurarse en distintos aspectos según la aplicación, como son el modo de comunicación que a su vez puede ser síncrona o asíncrona, el inicio o fin de la simulación en CoppeliaSim, la aplicación y lectura de fuerzas, posición o velocidad de las referencias u objetos usados en el modelo del robot virtual implementado.
- La generación de trayectoria de este proyecto especifica en todo momento a lo largo del tiempo la posición y velocidad, parámetros más que importantes a la hora de controlarla trayectoria de un robot.

- El esquema de control creado en este proyecto necesita de la cinemática inversa del robot 3UPS+RPU, además del Jacobiano de velocidades, el cual es el responsable de relacionar la velocidad cartesiana de la plataforma móvil con la velocidad de las articulaciones.

- Los errores de posición obtenidos por el controlador de trayectoria son bajos, mientras que se nota un error aceptable en la velocidad, específicamente en el eje z, donde el error de velocidad permanece casi constante, sin embargo, al ser pequeño no representa problema con el seguimiento de trayectoria, este error es debido al tipo de trayectoria usado, el cual es una función polinomial que al generar la trayectoria toma en cuenta la aceración en la referencia.

3.4 RECOMENDACIONES

- En la comunicación CoppeliaSim-MATLAB por medio de una API remota se debe considerar el uso de la función de obtención de señales para obtener el valor de la velocidad de cada uno de los eslabones debido a que no existe una función específica que retorne la velocidad de las articulaciones dentro de la librería de las API remotas.

- En la creación del modelo cinemático virtual del robot se recomienda orientar las piezas geométricas apartándolas del modelo completo, esto debido a las funciones limitadas que posee CoppeliaSim en el manejo de referencias de los objetos que componen el robot, de esta manera se pueden rotar los ejes de la pieza que se quiere mover girar sus ejes de referencia.

- Realizar la comunicación MATLAB-CoppeliaSim por los métodos que utilicen menos recursos computacionales de tal manera que se tenga una simulación ágil en CoppeliaSim, para este fin se pude usar la función S-Function de nivel 1 en MATLAB para comunicarse con CoppeliaSim ya que esta función es mucho más rápida en comparación con otros métodos como la S-Function de nivel 2.

- El robot 3UPS+RPU físico presenta zonas donde para una determinada extensión de los cilindros se hallan más de una posición para la plataforma móvil, por lo que queda para posteriores trabajos evitar estas zonas al generar la trayectoria para el robot paralelo.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] A. L. G. MORRELL *et al.*, “The history of robotic surgery and its evolution: when illusion becomes reality,” Revista do Colégio Brasileiro de Cirurgiões, vol. 48, 2021, doi: 10.1590/0100-6991e-20202798.

[2] Z. Yaniv and L. Joskowicz, “Precise robot-assisted guide positioning for distal locking of intramedullary nails,” IEEE Trans Med Imaging, vol. 24, no. 5, 2005, doi: 10.1109/TMI.2005.844922.

[3] S. B. Brotzman and K. E. Wilk, Rehabilitación Ortopédica Clínica, Elsevier. Madrid, 2005.

[4] A. Elizabeth et al., “Parallel robot for knee rehabilitation: Reduced order dynamic linear model, mechanical assembly and control system architecture,” Periodicals of Engineering and Natural Sciences, vol. 9, no. 1, pp. 194–215, Feb. 2021.

[5] P. Araujo-Gómez, M. Díaz-Rodriguez, V. Mata, A. Valera, and A. Page, “Design of a 3-UPS-RPU Parallel Robot for Knee Diagnosis and Rehabilitation,” in ROMANSY 21 - Robot Design, Dynamics and Control, 2016, pp. 303–310.

[6] Coppelia Robotics AG, “messaging/interfaces/connectivity,” 2020. https://www.coppeliarobotics.com/helpFiles/en/meansOfCommunication.htm (último acceso: Nov. 29, 2021).

[7] L. Fernández, L. Sotomayor, and I. Zambrano, “ANÁLISIS CINEMÁTICO INVERSO Y DIRECTO DEL ROBOT PARALELO ,” Quito, 2016. Último acceso: Nov. 29, 2021. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/16868

[8] R. Navarrete, G. Sánchez, and I. Zambrano, “DISEÑO, CONSTRUCCIÓN, ENSAMBLE Y PRUEBAS DE UN ROBOT PARALELO 3UPS + 1RPU PARA REHABILITACIÓN DE RODILLA,” Quito, 2019. Último acceso: Nov. 29, 2021. [En línea]. Disponible: http://bibdigital.epn.edu.ec/handle/15000/20582

[9] D. Nedosseikine, “Modelado cinemático y dinámico de un robot paralelo con arquitectura 3UPS-RPU. Aplicación en la simulación de robots de rehabilitación de miembro inferior,” Sep. 2021, Último acceso: Nov. 28, 2021. [En línea]. Disponible: https://riunet.upv.es:443/handle/10251/172609

[10] K. Duarte Barón and C. Borrás Pinilla, “Generalidades de robots paralelos,” Visión electrónica, vol. 10, no. 1, pp. 102–112, Jun. 2016, doi: 10.14483/22484728.11711.

[11] F. Valero, M. Díaz-Rodríguez, M. Vallés, A. Besa, E. J. Bernabeu, and Á. Valera, “Reconfiguration of a parallel kinematic manipulator with 2T2R motions for avoiding singularities through minimizing actuator forces,” Mechatronics, vol. 69, p. 102382, Aug. 2020, doi: 10.1016/j.mechatronics.2020.102382.

[12] P. Araujo-Gómez, V. Mata, M. Díaz-Rodríguez, Á. Valera, and A. Page, “Design and Kinematic Analysis of a Novel 3UPS/RPU Parallel Kinematic Mechanism With 2T2R Motion for Knee Diagnosis and Rehabilitation Tasks,” Journal of Mechanisms and Robotics, vol. 9, Sep. 2017, doi: 10.1115/1.4037800.

[13] The MathWorks, “Uso de MATLAB con otros lenguajes de programación - MATLAB & Simulink.” https://la.mathworks.com/products/matlab/matlab-and-other-programming-languages.html?s_tid=srchtitle_lenguaje_2 (último acceso: Dec. 26, 2021).

[14] The MathWorks, “Simulación y diseño basado en modelos con Simulink - MATLAB & Simulink,” 2021. https://la.mathworks.com/products/simulink.html?s_tid=srchtitle_simulink_2 (último acceso: Dec. 26, 2021).

[15] Coppelia Robotics, “User Manual to Coppelia,” 2017.

[16] Coppelia Robotics, “Remote API functions (Matlab),” 2017. https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm (último acceso: Dec. 13, 2021).

[17] H. F. R. Bravo, L. M. R. Rivera, A. P. Buchelli, and J. Barco, “Development and control of virtual plants in a co-simulation environment,” 4th IEEE Colombian Conference on Automatic Control: Automatic Control as Key Support of Industrial Productivity, CCAC 2019 - Proceedings, Oct. 2019, doi: 10.1109/CCAC.2019.8921186.

[18] J. J. Craig, *ROBOTICA*, 3rd ed. México: Pearson Educación, 2006.

[19] O. Camacho, A. Rosales, and F. Rivas, Control de Procesos, 1st ed. Quito: EPN Editorial, 2020.

[20] L. A. Fernández Yánez and L. F. Sotomayor Reinoso, “Análisis cinemático inverso y directo del robot paralelo,” Quito, 2016., Quito, 2016. Accessed: Feb. 14, 2022. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/16868

5 ANEXOS

Para el completo entendimiento del desarrollo de este trabajo se presenta los siguientes anexos:

- Anexo I. Manual de usuario.
- Anexo II. Configuración de las articulaciones.
- Anexo III. Ubicación de los elementos que componen el robot 3UPS+RPU.
- Anexo IV. Bloque de comunicación Simulink-CoppeliaSim.
- Anexo V. Ecuaciones de velocidad
- Anexo VI. Sintonización de controlador interno.
- Anexo VII. Sintonización del controlador de trayectoria
- Anexo VIII. Enlaces

ANEXO I

MANUAL DE USUARIO

A continuación, se presenta el manual de usuario para simular el controlador de trayectoria para el robot paralelo virtual 3UPS+RPU, en este manual se presenta la operación de todos los archivos necesarios para correr el controlador con las trayectorias creadas para el robot.

Para lograr correr el controlador satisfactoriamente es necesario que al abrir la carpeta de Archivos_de_simulacion encuentre los siguientes archivos:

Archivos para comunicación:

- Controlador_trayectoria.slx
- remApi.m
- remoteApi.dll
- remoteApiProto.m
- UPSRPU_SIMtoVREP.m

Interfaz en MATLAB:

- Interfaz3UPS1RPU.mlapp
- Posición.mlapp
- Escudo_EPN.png

Modelo en CoppeliaSim:

- 3UPS_1RPU_V4B0_dinámico.ttt

Control de trayectoria:

- Control_trayectoria.slx

Al tener estos archivos presentes, el primer paso es abrir la interfaz de usuario el cual se muestra en la Figura I.1, en esta se muestra la pantalla inicial de la interfaz. Por medio de esta pantalla se accede al controlador de trayectoria que se muestra en la Figura I.2.

Figura I.1. Pantalla inicial de la interfaz gráfica.

En la pantalla de la Figura I.2, el paso siguiente y el más importante es abrir el modelo virtual mediante el botón 5, al presionar se abrirá el archivo de CoppeliaSim mostrado en

la Figura I.3 donde para mejorar la vista del robot se puede ocular las ventanas del Model browser y Scene hierechary, también mostradas en la Figura I.3.

Luego de estos pasos iniciales se debe seleccionar la trayectoria en la zona 1 de la pantalla en la Figura I.2, con la trayectoria seleccionada se comanda la simulación con los botones RUN y STOP en la zona 2 de la pantalla. Finalizada la simulación se puede graficar la trayectoria en el espacio xz que se muestra en la zona 6 de la pantalla.

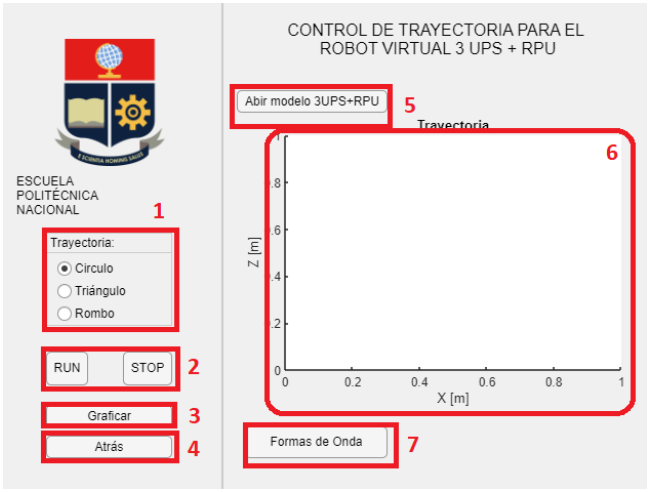


Figura I.2. Pantalla de manejo del controlador de trayectoria.

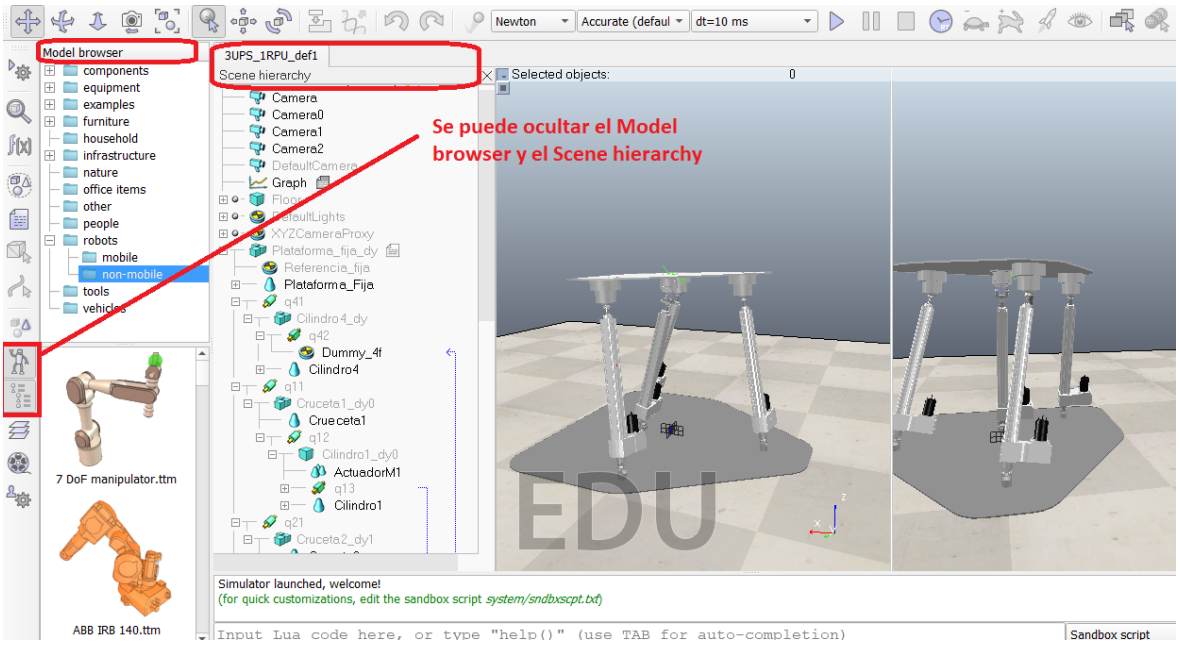


Figura I.3. Modelo virtual del robot 3UPS+RPU en CoppeliaSim.

Para obtener las gráficas de la posición y la velocidad con respecto al tiempo en los ejes x y z se selecciona el botón formas de ondas de la Figura I.2 el cual abre la pantalla de la Figura I.4, esta pantalla muestra las gráficas de posición velocidad y error en los ejes x y z seleccionando los respectivos botones mostrados a su izquierda.

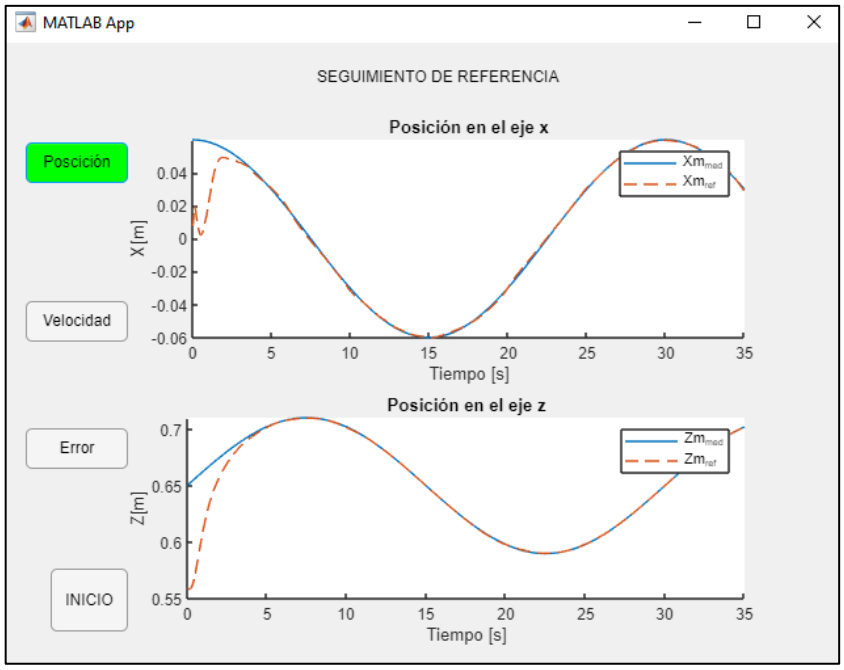


Figura I.4. Pantalla de gráfico de posición.

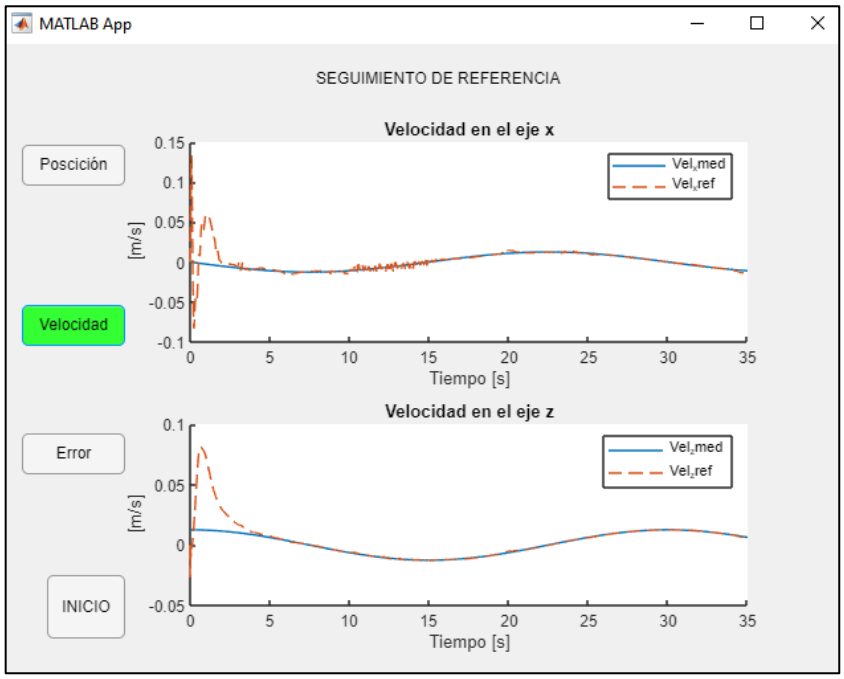


Figura I.5. Pantalla de gráfico de velocidad.

Las pantallas de las Figuras I.4, I.5 e I.6 son los gráficos de las señales que se pueden obtener de la simulación de una trayectoria seleccionada, cada una de estas mostrará una señal de referencia y la señal de salida del robot virtual 3UPS+RPU proveniente de CoppeliaSim.

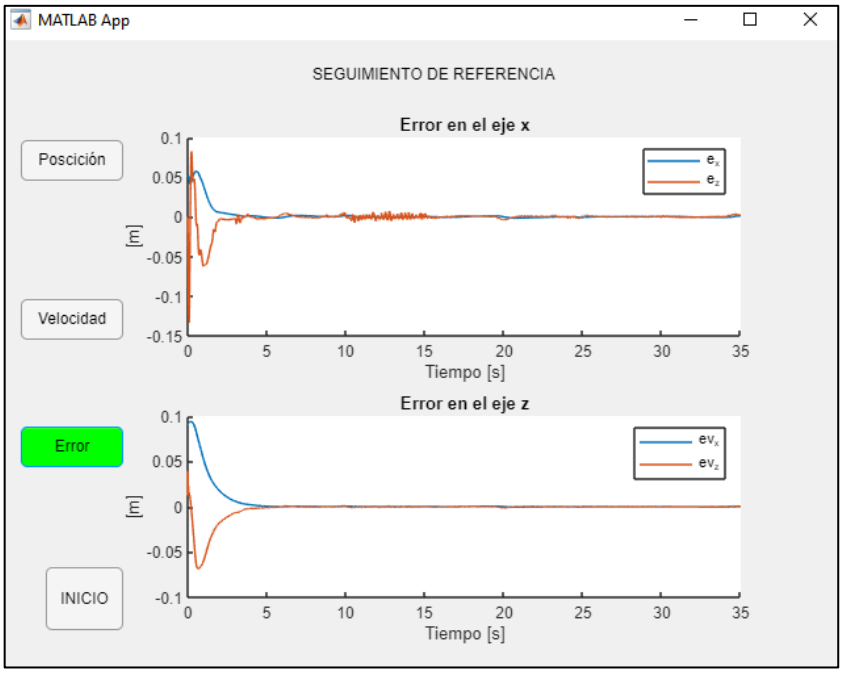


Figura I.6. Pantalla de gráfico de velocidad.

Como paso adicional el botón INICIO abre la pantalla de INICIO de la aplicación que se mostró en la Figura I.1.

ANEXO II

CONFIGURACIÓN DE LAS ARTICULACIONES

Tabla II.1. Configuración de las articulaciones para el modo dinámico en CoppeliaSim

Clasificación	Tipo	Simbología	Configuración (modo)	Configuración Adicional
Dependientes	Revolución	q_{11}	Cinemática inversa	Modo híbrido
	Revolución	q_{12}	Cinemática inversa	Modo híbrido
	Esférica	q_{24}	Par-Fuerza	
	Revolución	q_{21}	Cinemática inversa	Modo híbrido
	Revolución	q_{22}	Cinemática inversa	Modo híbrido
	Esférica	q_{24}	Par-Fuerza	
	Revolución	q_{31}	Cinemática inversa	Modo híbrido
	Revolución	q_{32}	Cinemática inversa	Modo híbrido
	Esférica	q_{34}	Par-Fuerza	
	Revolución	q_{41}	Cinemática inversa	Modo híbrido
	Revolución	q_{43}	Cinemática inversa	Modo híbrido
	Revolución	q_{44}	Cinemática inversa	Modo híbrido
Independientes	Prismática	q_{13}	Par-Fuerza	Habilitación de motor y PID
	Prismática	q_{23}	Par-Fuerza	Habilitación de motor y PID
	Prismática	q_{33}	Par-Fuerza	Habilitación de motor y PID
	Prismática	q_{34}	Par-Fuerza	Habilitación de motor y PID

Esta tabla muestra cada una de las configuraciones que se aplicó a las distintas articulaciones que se utilizó al construir el robot paralelo 3UPS+RPU en CoppeliaSim.

ANEXO III

UBICACIÓN DE LOS ELEMENTOS QUE COMPONEN EL ROBOT 3UPS+RPU

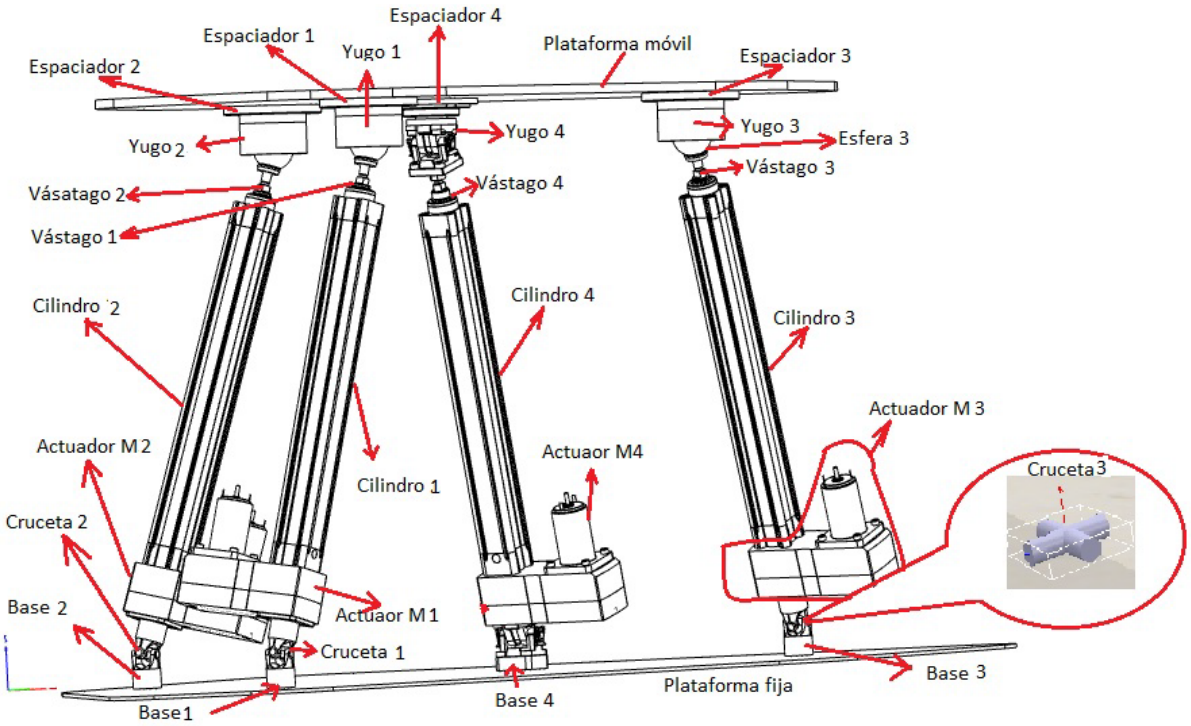


Figura III.1. Elementos que conforman el robot 3UPS+RPU.

Los elementos mostrados en la Figura II.1 son los nombres con los que se puede encontrar en el modelo creado en CoppeliaSim.

ANEXO IV

BLOQUE DE COMUNICACIÓN Simulink-CoppeliaSim.

Esta primera función recibe los datos desde el motor de simulación de Simulink, recibe variables que son necesarios para la operación del bloque, como el tiempos (t), x, y la variable flag, el cual permite ejecutar cada función del bloque dependiendo el estado de la simulación, que puede ser inicio de la simulación, paso de simulación y también fin de la simulación.

```
function [sys,x0,str,ts] = UPSRPU_SIMtoVREP(t,x,u,flag) %x=[q1 dq1 q2
dq2];u=tol, (t,x,u,flag)

switch flag
    case 0
        [sys,x0,str,ts]=mdlInitializeSizes;
    case 3
        sys=mdlOutputs(t,u);
    case {2,4}
        sys=[];
    case 9,
    sys=mdlTerminate(t,x,u);
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end
```

Esta función solo se ejecuta una vez al inicio de la simulación, por lo que en la función mdlInitializeSizes se inicializa variables para el bloque, como son las variables de entrada, variables de salida, entre las más importantes, además para el caso de la comunicación Simulink-CoppeliaSim, se abre la librería para utilizar la API remota para la comunicación, así como también establece comunicación con el programa CoppeliaSim. Una vez comunicado con CoppeliaSim, obtenemos los Handles o manejadores de los dummies para poder recibir información en cada paso de simulación.

```
function [sys,x0,str,ts]=mdlInitializeSizes % Establece las condiciones.
global sim clientID loopt1 h ref_movil
sizes = simsizes;
sizes.NumOutputs     = 12; %output is tol=[tau1 tau2].
sizes.NumInputs      = 4;  %q=[q13 q23 q33 q42]
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
% %about ts
% %The valid sample time pairs for a Level-2 MATLAB S-function are
% [0 offset]                            % Continuous sample time
% [discrete_sample_time_period, offset] % Discrete sample time
% [-1, 0]                               % Inherited sample time
% [-2, 0]                               % Variable sample time
% %where variable names in italics indicate that a real value is
required.
```

```
% %When using a continuous sample time,
% %an offset of 1 indicates the output is fixed in minor integration time
steps.
% %An offset of 0 indicates the output changes at every minor integration
time step.
x0  = [];
str = [];
%ts  = [0 0];
ts  = [0 1];

h=[0,0,0,0];
sim=remApi('remoteApi');
sim.simxFinish(-1); % just in case, close all opened connections
clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5); %establece la
conexion con VREP
%obtención de los handles
[r,h(1)] =
sim.simxGetObjectHandle(clientID,'q13',sim.simx_opmode_blocking);
[r,h(2)] =
sim.simxGetObjectHandle(clientID,'q23',sim.simx_opmode_blocking);
[r,h(3)] =
sim.simxGetObjectHandle(clientID,'q33',sim.simx_opmode_blocking);
[r,h(4)] =
sim.simxGetObjectHandle(clientID,'q42',sim.simx_opmode_blocking);
[r,ref_movil]=sim.simxGetObjectHandle(clientID,'Ref_IK',sim.simx_opmode_b
locking);

sim.simxSynchronous(clientID,true);% activa el modo síncrono para la
comunicación
r=sim.simxStartSimulation(clientID,sim.simx_opmode_blocking); % inicia la
simulación en VREP
    while r~=sim.simx_return_ok % espera hasta que termine de iniciar la
simulación
        Tm=0;
    end
loopt1=0;
```

En la función `mdlOutputs` se establece las variables que se tendrán a la salida del bloque y se envía las acciones de control hacia CoppeliaSim.

```
function sys=mdlOutputs(t,u) %salidas del bloque de comunicacion
global sim clientID loopt1 h ref_movil %Position Orientation
Vel_signal=['jointVel1';'jointVel2';'jointVel3';'jointVel4'];
if loopt1==0
    if (clientID>-1)
        disp('Connected to remote API server');
    else
        disp('Failed connecting to remote API server');
    end
    %sim.delete(); % call the destructor!
    loopt1 = loopt1+1;
end

%get q_des from input.
q13=u(1);
q23=u(2);
q33=u(3);
q42=u(4);
```

```
q=[q13;q23;q33;q42];
joint_pos1=q;
Position=zeros(1,3);
Orientation=zeros(1,3);
VP=zeros(1,3);
VO=zeros(1,3);
qp=[0 0 0 0];
qpos=[0 0 0 0];
if (clientID>-1)
    %Obtencion de la posicion y orientacion de la plataforma movil.
    [r,Position] = sim.simxGetObjectPosition(clientID,ref_movil,-
1,sim.simx_opmode_blocking);
%    Pz=Position{2};
    [r,Orientation] = sim.simxGetObjectOrientation(clientID,ref_movil,-
1,sim.simx_opmode_blocking);
%    Oz=Orientation{2};
%Aplicacion de la accion de control y obtencion de Posicion y velocidad
de
%los actuadores.
    for i = 1: length(h)


sim.simxSetJointTargetPosition(clientID,h(i),u(i),sim.simx_opmode_oneshot
);

    end
    sim.simxSynchronousTrigger(clientID); % envia el trigger para
realizar un paso de simulación en VREP
    %Obtencion de las velocidades de los actuadores:
    for i=1:4
            [r,qpos(i)]=sim.simxGetJointPosition(clientID,
h(i),sim.simx_opmode_streaming);
    end

        %obtencion de las velocidades de los joints q13,q23,q33,q42:

[r,qp(1)]=sim.simxGetFloatSignal(clientID,Vel_signal(1,:),sim.simx_opmode
_streaming);

[r,qp(2)]=sim.simxGetFloatSignal(clientID,Vel_signal(2,:),sim.simx_opmode
_streaming);

[r,qp(3)]=sim.simxGetFloatSignal(clientID,Vel_signal(3,:),sim.simx_opmode
_streaming);

[r,qp(4)]=sim.simxGetFloatSignal(clientID,'T_simulation',sim.simx_opmode_
streaming);
%    [r,h(7)] =
sim.simxGetObjectHandle(clientID,'q32',sim.simx_opmode_blocking);
%Obtención de la velocidad de la plataforma movil:

[r,VP,VO]=sim.simxGetObjectVelocity(clientID,ref_movil,sim.simx_opmode_st
reaming);
else
    disp('Failed connecting to remote API server');
end


%disp('loopt ended');
```

```
sys(1)=double(Position(1)); %S-file output1
sys(2)=double(Position(3)-0.045); %S-file output2
sys(3)=double(Orientation(2));
sys(4)=double(Orientation(3));
sys(5)=double(qpos(1));
sys(6)=double(qpos(2));
sys(7)=double(qpos(3));
sys(8)=double(qp(4));
sys(9)=double(VP(1));
sys(10)=double(VP(3));
sys(11)=double(VO(2));
sys(12)=double(VO(3));

function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
```

Finalmente en la función mdlTerminate se finaliza con la comunicación, eliminando el canal de comunicación con CoppeliaSim y deteniendo la simulación en CoppeliaSim.

```
function sys=mdlTerminate(t,x,u) % finaliza la comunicación con VREP.
global sim clientID
 sim.simxStopSimulation(clientID,sim.simx_opmode_oneshot)
 sim.simxAddStatusbarMessage(clientID,'Comunicación con matlab
finalizada',sim.simx_opmode_blocking);
 sim.simxFinish(-1);
 disp('Comunicación vrep finalizada')
 sim.delete();
 %clear all;
 clc
sys = [];

% end mdlGetTimeOfNextVarHit
```

ANEXO V

ECUACIONES DE VELOCIDAD

$$\cos(q_{11})\sin(q_{12})\,dq_{13} - q_{13}\sin(q_{11})\sin(q_{12})\,dq_{11} + q_{13}\cos(q_{11})\cos(q_{12})\,dq_{12} - dX_m - rsin(\theta)\cos(\psi)\,d\theta - rcos(\theta)\sin(\psi)\,d\psi = 0$$

$$-\cos(q_{12})\,dq_{13} + q_{13}\sin(q_{12})\,dq_{12} - rsin(\theta)sin(\psi)d\theta + rcos(\theta)\cos(\psi)\,d\psi = 0$$

$$\sin(q_{11})\sin(q_{12})\,dq_{13} + q_{13}\cos(q_{11})\sin(q_{12})\,dq_{11} + q_{13}\sin(q_{11})\cos(q_{12})\,dq_{12} - dZ_m - rcos(\theta)d\theta = 0$$

$$\cos(q_{21})\sin(q_{22})\,dq_{23} - q_{23}\sin(q_{21})\sin(q_{22})\,dq_{21} + q_{23}\cos(q_{21})\cos(q_{22})\,dq_{22} - dX_m + rcos(\mathrm{B}_{2m})\sin(\theta)\cos(\psi)\,d\theta + (rsin(\beta_{2\mathrm{m}})\cos(\psi) + r\cos(\beta_{2m})\cos(\theta)\sin(\psi))d\psi = 0$$

$$-\cos(q_{22})\,dq_{23} + q_{23}\sin(q_{22})\,dq_{22} + rcos(\beta_{2m})\sin(\theta)\cos(\psi)\,d\theta + (rsin(\beta_{2m})\sin(\psi) + rcos(\beta_{2m})\cos(\theta)\sin(\psi))d\psi = 0$$

$$\sin(q_{21})\sin(q_{22})\,dq_{23} + q_{23}\cos(q_{21})\sin(q_{22})\,dq_{21} + q_{23}\sin(q_{21})\cos(q_{22})\,dq_{22} - dZ_m - rcos(\beta_{2m})cos(\theta)d\theta = 0$$

$$\cos(q_{31})\sin(q_{32})\,dq_{33} + q_{33}\sin(q_{31})\sin(q_{32})\,dq_{31} + q_{33}\cos(q_{31})\cos(q_{32})\,dq_{32} - dX_m + rcos(\beta_{3m})\sin(\theta)\cos(\psi)\,d\theta + (-rsin(\beta_{3m})\cos(\psi) + rcos(\beta_{3m})\cos(\theta)\sin(\psi))d\psi = 0$$

$$-\cos(q_{23})\,dq_{33} + q_{33}\sin(q_{32})\,dq_{32} + rcos(\beta_{3m})\sin(\theta)\sin(\psi)\,d\theta + (-rcos(\beta_{3m})\cos(\theta)\cos(\psi) - rsin(\beta_{3m})\sin(\psi))d\psi = 0$$

$$\sin(q_{31})\sin(q_{32})\,dq_{33} + q_{33}\cos(q_{31})\sin(q_{32})\,dq_{31} + q_{33}\sin(q_{31})\cos(q_{32})\,dq_{32} - dZ_m + rcos(\beta_{3m})\cos(\theta)\,d\theta = 0$$

$$q_{42}\cos(q_{41})\,dq_{41} + \sin(q_{41})\,dq_{42} + dX_m = 0$$

$$q_{42}\sin(q_{41}) - \cos(q_{41})\,dq_{42} + dZ_m = 0$$

Donde:

dX_m= Velocidad de la plataforma móvil en el eje x.

dZ_m=Velocidad de la plataforma móvil den el eje z.

$d\theta$=Velocidad angular respecto al eje y.

$d\psi$=Velocidad angular respecto al eje z.

dq_{11}=Velocidad de la primera articulación rotacional de la pata 1.

dq_{12}=Velocidad de la segunda articulación rotacional de la pata 1.

dq_{13}=Velocidad de la articulación prismática de la pata 1.

dq_{21}=Velocidad de la primera articulación rotacional de la pata 2.

dq_{22}=Velocidad de la segunda articulación rotacional de la pata 2.

dq_{23}=Velocidad de la articulación prismática de la pata 2.

dq_{31}=Velocidad de la primera articulación rotacional de la pata 3.

dq_{32}=Velocidad de la segunda articulación rotacional de la pata 3.

dq_{33}=Velocidad de la articulación prismática de la pata 3.

dq_{41}=Velocidad de la primera articulación rotacional de la pata 4.

dq_{42}=Velocidad de la articulación prismática de la pata 4.

ANEXO VI

SINTONIZACIÓN DE CONTROLADOR INTERNO

CoppeliaSim ofrece una opción de control PID, dentro de las propiedades dinámicas, para cada una de las articulaciones añadidos en su entorno de simulación. Para la sintonización de las articulaciones independientes del robot 3UPS+RPU se varían los valores de KP, Ki y Kd del controlador PID hasta conseguir alcanzar la referencia.

Durante la sintonización del controlador se obtuvieron los valores mostrados en la Tabla VI.1.

Tabla VI.2. Pruebas para sintonización del controlador interno de posición

N° Prueba	Articulación	Kp	Kd	Ki	ITSE
1	q_{13}	0.01	0	0	0.1011
1	q_{23}	0.01	0	0	0.1018
1	q_{33}	0.01	0	0	0.09454
1	q_{42}	0.01	0	0	0.09361
2	q_{13}	0.05	0	0	0.09235
2	q_{23}	0.05	0	0	0.09238
2	q_{33}	0.05	0	0	0.09242
2	q_{42}	0.05	0	0	0.0923
3	q_{13}	0.05	0	0	0.09236
3	q_{23}	0.05	0	0.001	0.0924
3	q_{33}	0.05	0	0.001	0.0926
3	q_{42}	0.05	0	0.01	0.0923
4	q_{13}	0.05	0	0.04	0.09257
4	q_{23}	0.05	0	0.004	0.0958
4	q_{33}	0.05	0	0.004	0.09244
4	q_{42}	0.05	0	0.04	0.0923
5	q_{13}	0.051	0.001	0	0.09229
5	q_{23}	0.051	0.001	0.001	0.0923
5	q_{33}	0.048	0.001	0.001	0.09227
5	q_{42}	0.05	0.001	0.01	0.09226

En la primera prueba se obtienen valores de índices isco relativamente altos, por lo que se empieza variando únicamente el valor de Kp en un rango de 0.01, estos resultados se muestran en la Figura VI.1.

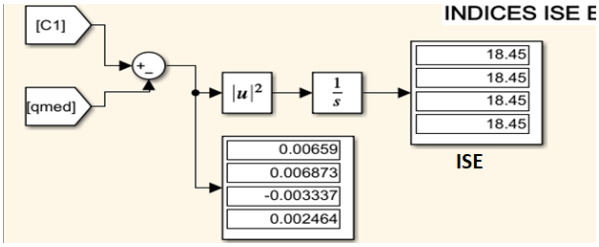


Figura VI.1. Índices ISE obtenidos en la prueba N° 1.

Para KP de 0.01, se obtienen una forma de onda muy alejada de la referencia que se muestra en la Figura VI.2.

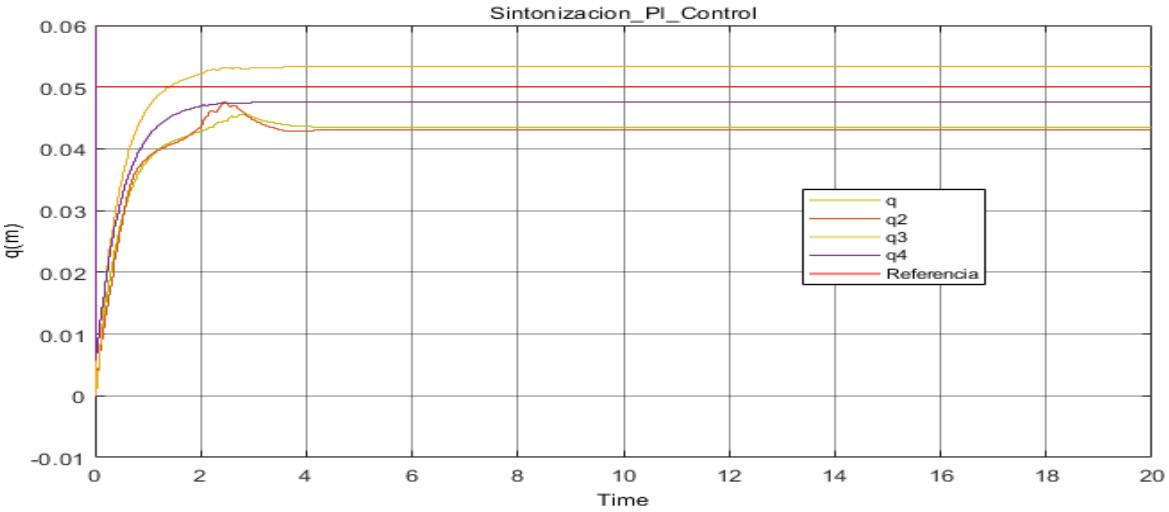


Figura VI.2. Respuesta de los cilindros para la prueba N° 1.

Como referencia se optó por un valor de 0.05 para todos los cilindros.

Se fue variando el Kp hasta llegar a un valor en el que casi todos los cilindros se acercan bastante a la referencia obteniendo la forma de onda de la Figura VI.3, mediante este valor es con el que se acerca más a la referencia, a partir de este valor se empezó a considerar valores de Ki para alcanzar el valor de la referencia.

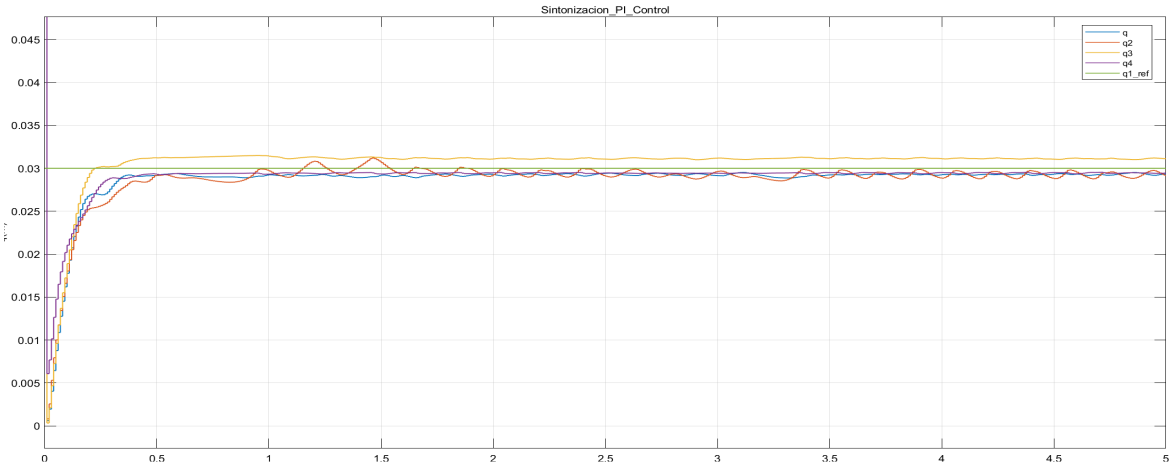


Figura VI.3. Respuesta de los cilindros para la prueba N° 2.

Para K=0.1, este valor de Kp es muy alto por lo que no se establece en la referencia y la señal se vuelve inestable, como se muestra en la Figura VI.4 , esta prueba nos indica hasta donde podemos aumentar el valor Kp, para generalizar este valor para todos los cilindros se estableció Kp en 0.05.

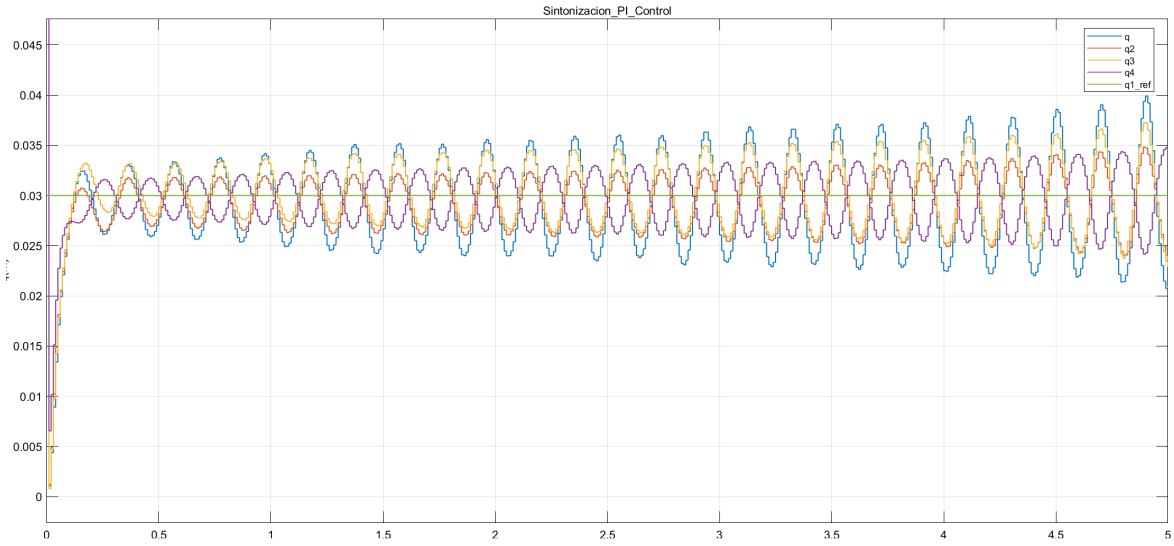


Figura VI.4. Respuesta de los cilindros para un valor de Kp=0.1.

Corrección de la pata 2.

Se tomó principal énfasis en la pata o cilindro 2, ya que este es el que más se aleja de la referencia, una vez alcanzado la referencia en esta pata o cilindro este mismo Ki se colocó en las otras patas teniendo resultados satisfactorios, como se muestra en la Figura VI.5 mismos que corresponden a la prueba N° 3 de la Tabla VI.1, donde se observa como los cilindros alcanzan la referencia.

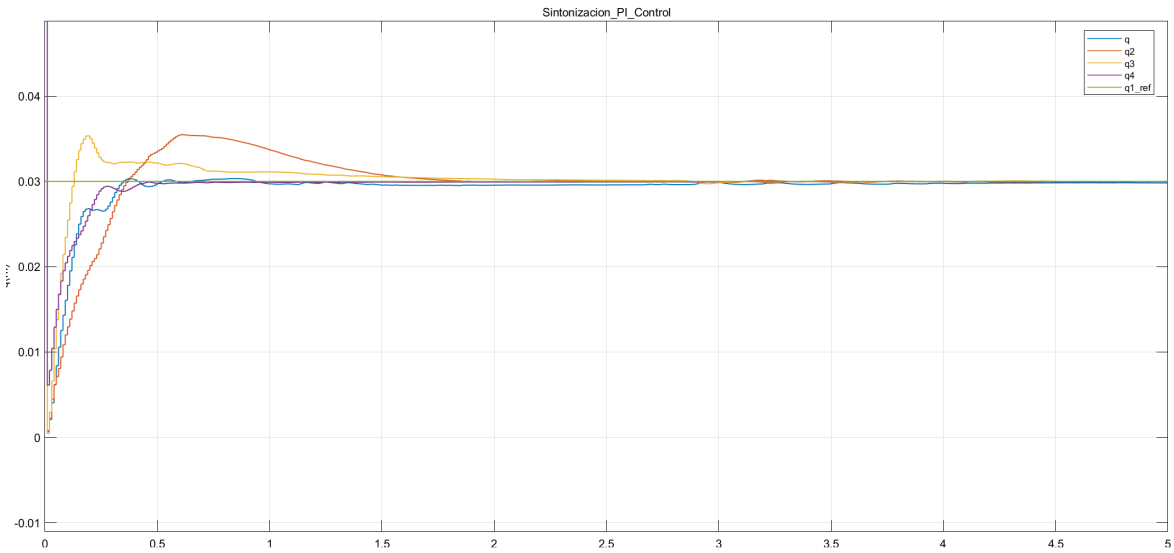


Figura VI.5. Sintonización de los valores Ki para el controlador interno.

El problema de la prueba N° 3 de la Tabla VI.1 es el sobrepico inicial, por lo que se añadió un componente derivativo, el Kd, este permitió reducir el sobrepico como se observa en la Figura VI.6

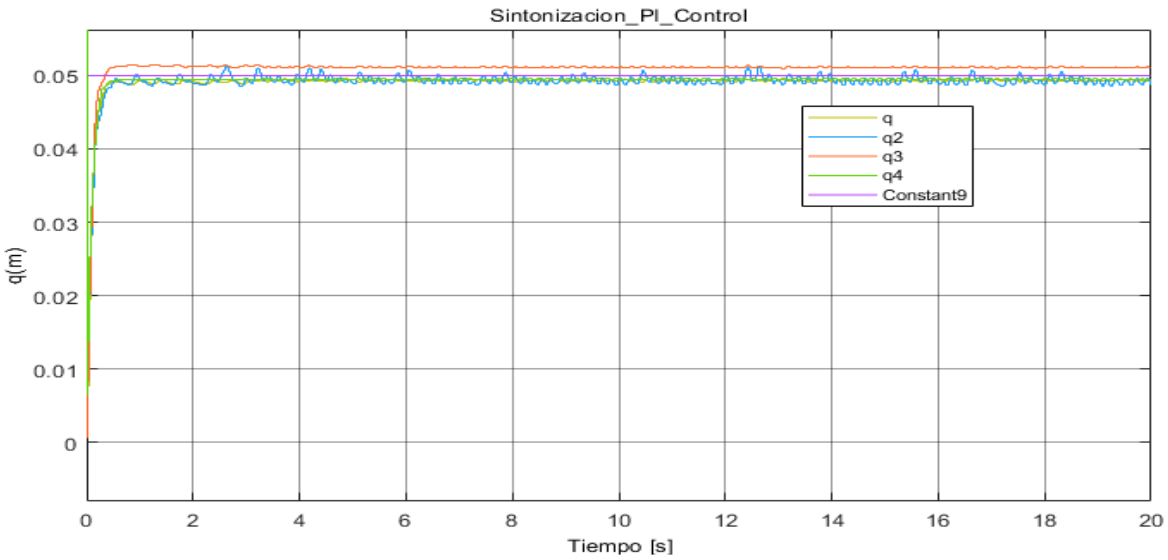


Figura VI.6. Sintonización de los valores de Kd para los controladores internos.

Última sintonización:

Esta sintonización es la denominada prueba 5 en la Tabla VI.1, en esta prueba ya se establecen los cilindros cerca de la referencia, esto también se ve evidente al obtener los índices ISE , donde se ve como el valor ha reducido ,al observar el error, también se puede evidenciar esto en la Figura VI.7 y en la Figura VI.8. Este cambio no se nota mucho en el valor ya que el error de posición que se redujo es pequeño en comparación con el tiempo de simulación.

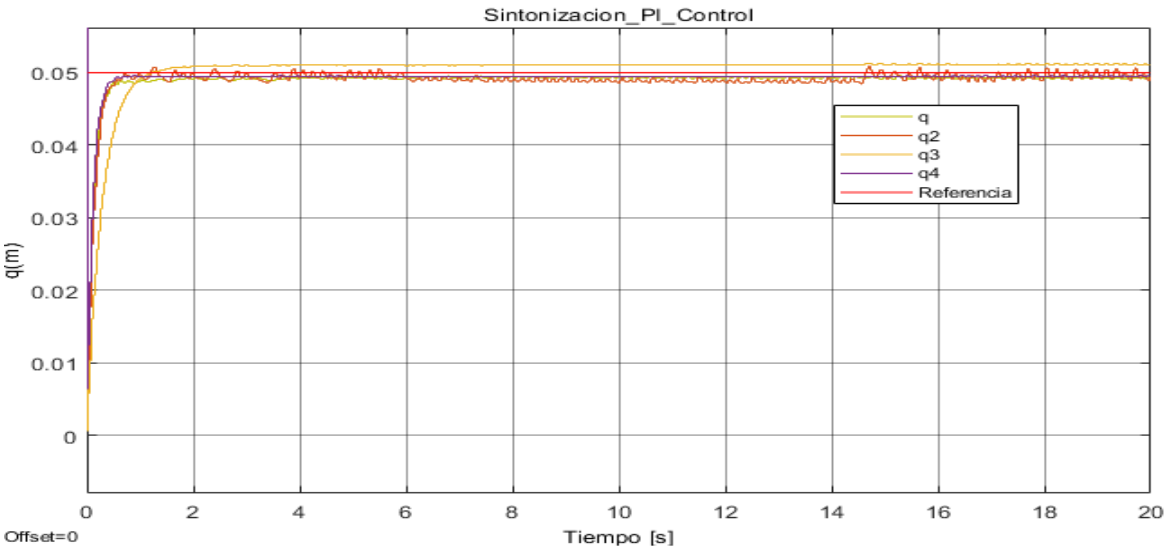


Figura VI.7. Respuesta de los cilindros para la prueba N° 5.

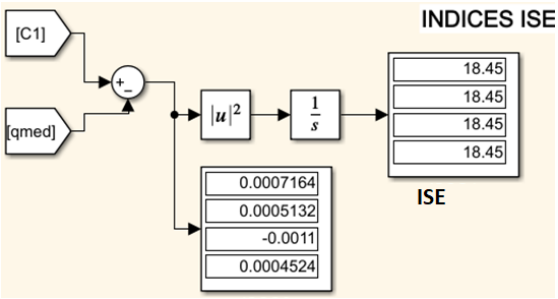


Figura VI.8. Valores de los índices ISE para la prueba N°5.

ANEXO VII

SINTONIZACIÓN DEL CONTROLADOR DE TRAYECTORIA

Para obtener una planta para el controlador, se observó cómo se comporta el modelo virtual enviando una señal con distintos pasos a uno de los ejes manteniendo un valor constante en el otro eje, como se muestra en la Figura VII.1, el valor de posición de los cilindros es enviado al modelo virtual en CoppeliaSim.

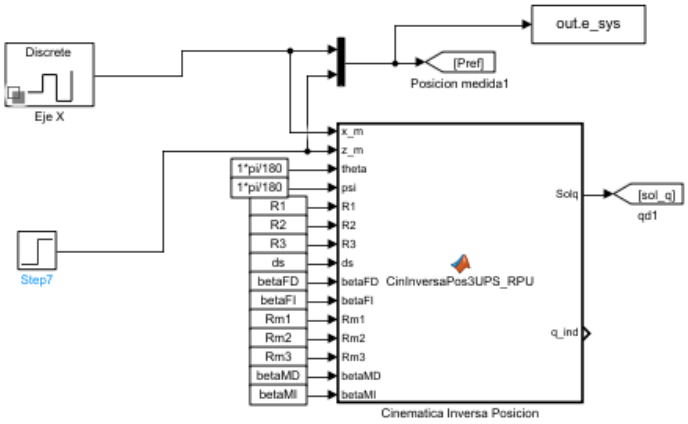


Figura VII.1. Operación realizada para enviar un valor constante en el eje Z y una función variable en el eje X.

Eje X:

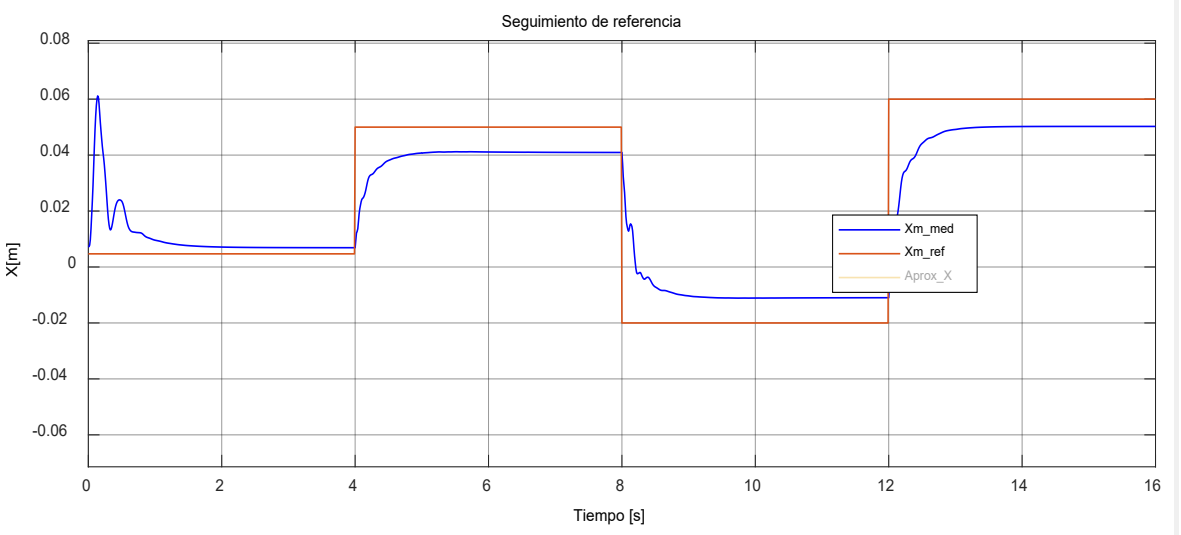


Figura VII.2. Respuesta del robot en el eje X ante la señal de referencia.

El resultado de la simulación en la Figura VII.1 es guardada con el fin exportarla al sistema de identificación de MATLAB, más conocido como IDENT, para obtener una planta con el que podamos hallar los parámetros para el controlador de trayectoria, como se muestra en la Figura VII.3.

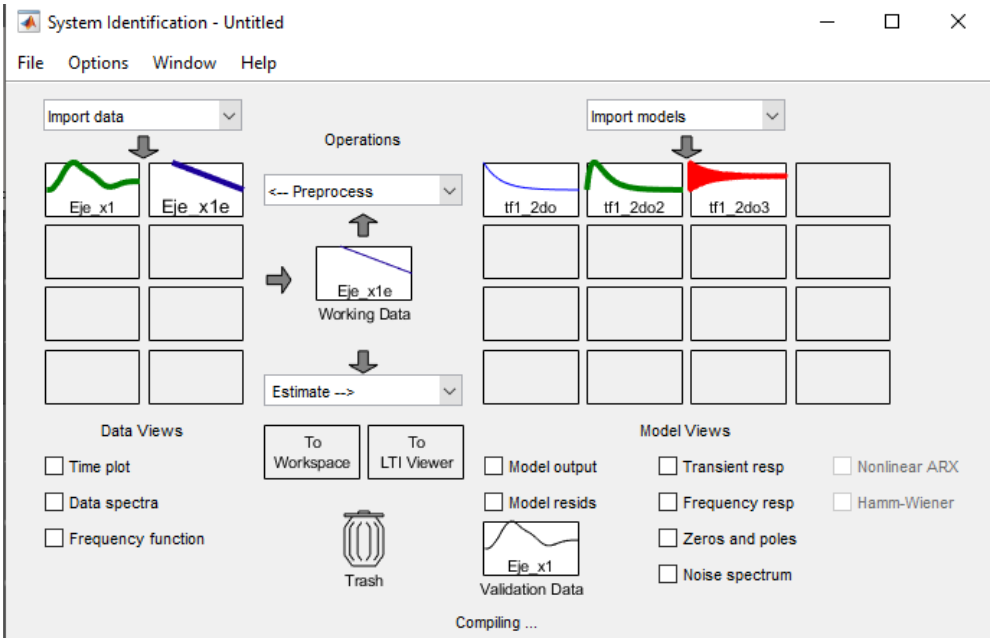


Figura VII.3. Manejo de datos obtenidos a partir del proceso de la Figura VII.1.

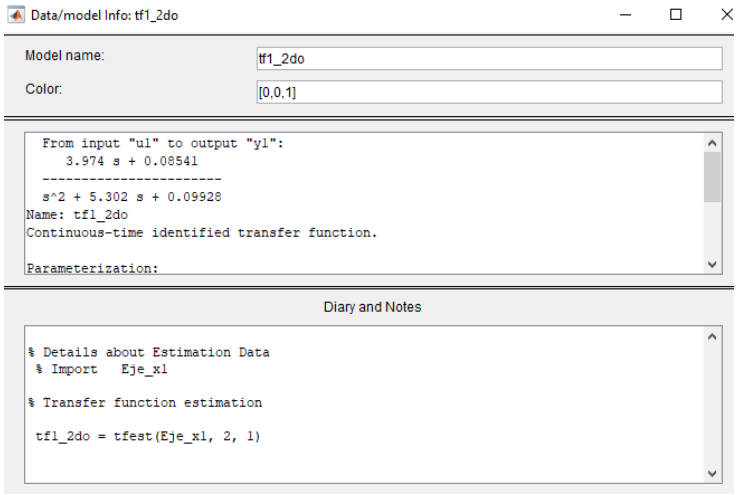


Figura VII.4. Función de transferencia de la función aproximada hallada en el eje X.

La función que mejor respuesta tubo fue la función de transferencia de tercer grado, esto se muestra en la Figura VII.4 por medio del ingreso de datos en la aplicación de MATLAB ident, como se muestra en la Figura VII.3

Función de transferencia obtenida de la Ecuación (VII.1)

$$G_x = \frac{3.974s + 0.08541}{s^2 + 5.302s + 0.09928} \qquad \text{(VII.1)}$$

A partir de la Función G_x se obtiene la respuesta del robot leída en el eje x, misma que se muestra en la Figura VII.5.

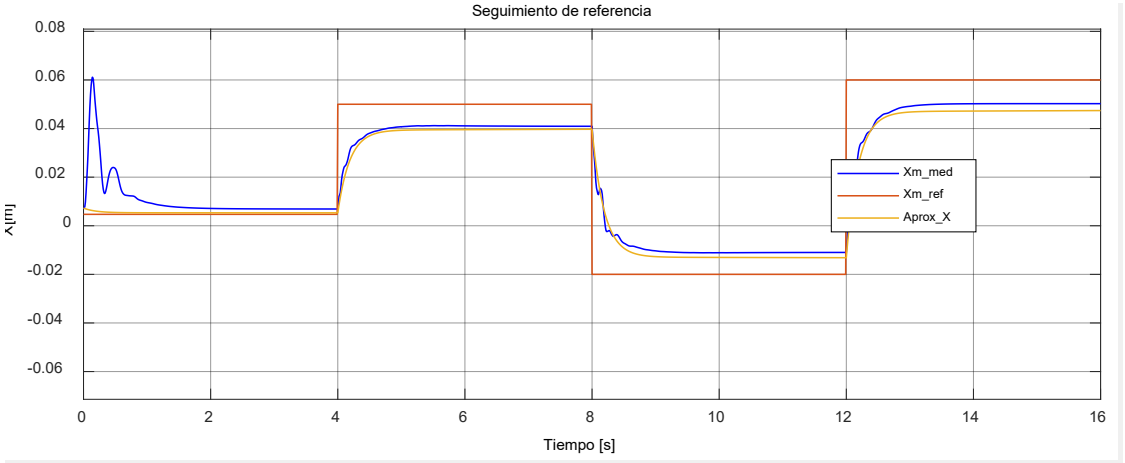


Figura VII.5. Respuesta del sistema aproximado hallado en el eje X.

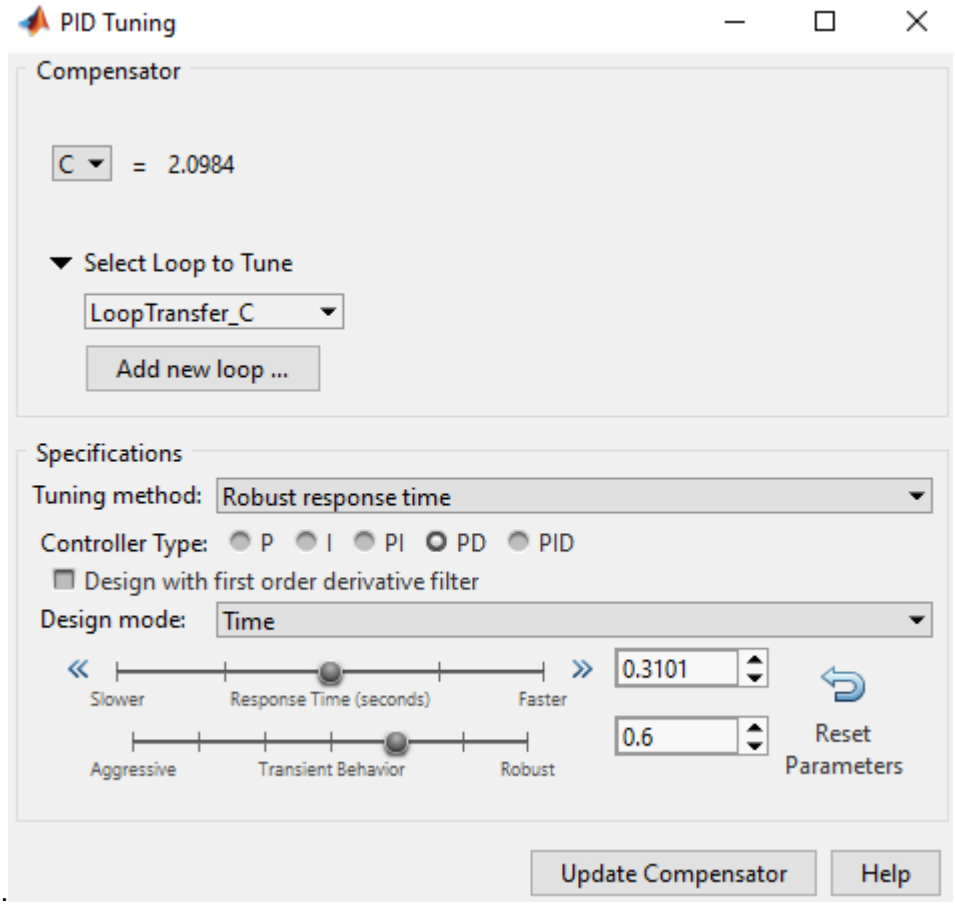


Figura VII.6. Sintonización de los parámetros del PID para la función de transferencia en el eje X.

El mismo procedimiento realizado con X se realiza con el eje Z, es decir se envía una señal en Z manteniendo constante X para luego con la ayuda de Ident de MATLAB obtener una función.

Eje z:

Se envió la señal de referencia que se muestra en la Figura VII.7 de la misma forma que se mostró en la Figura VII.

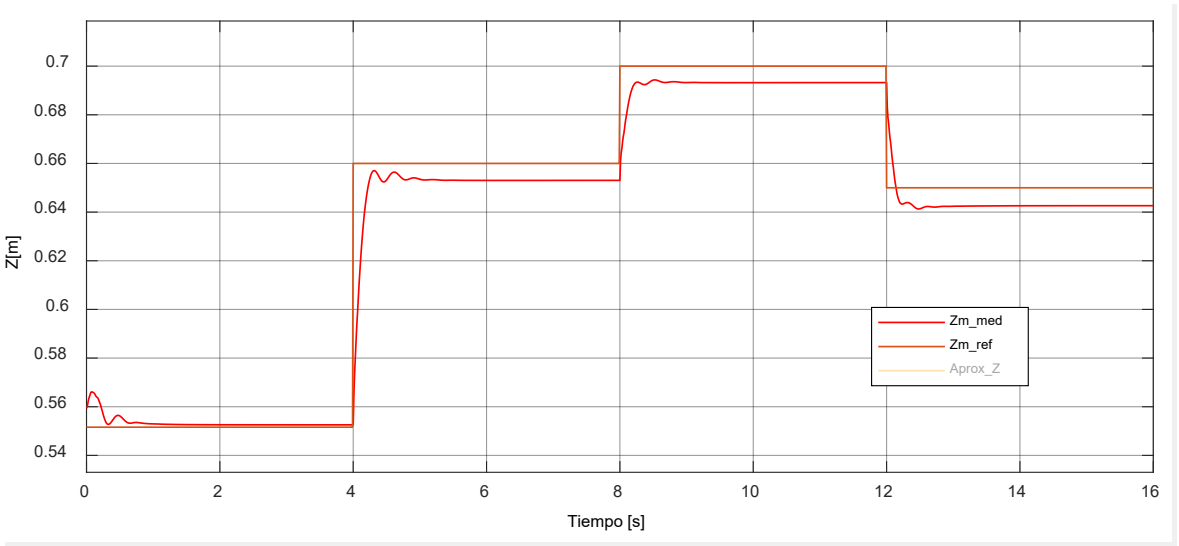


Figura VII.7. Respuesta del robot en el eje Y ante la señal de referencia

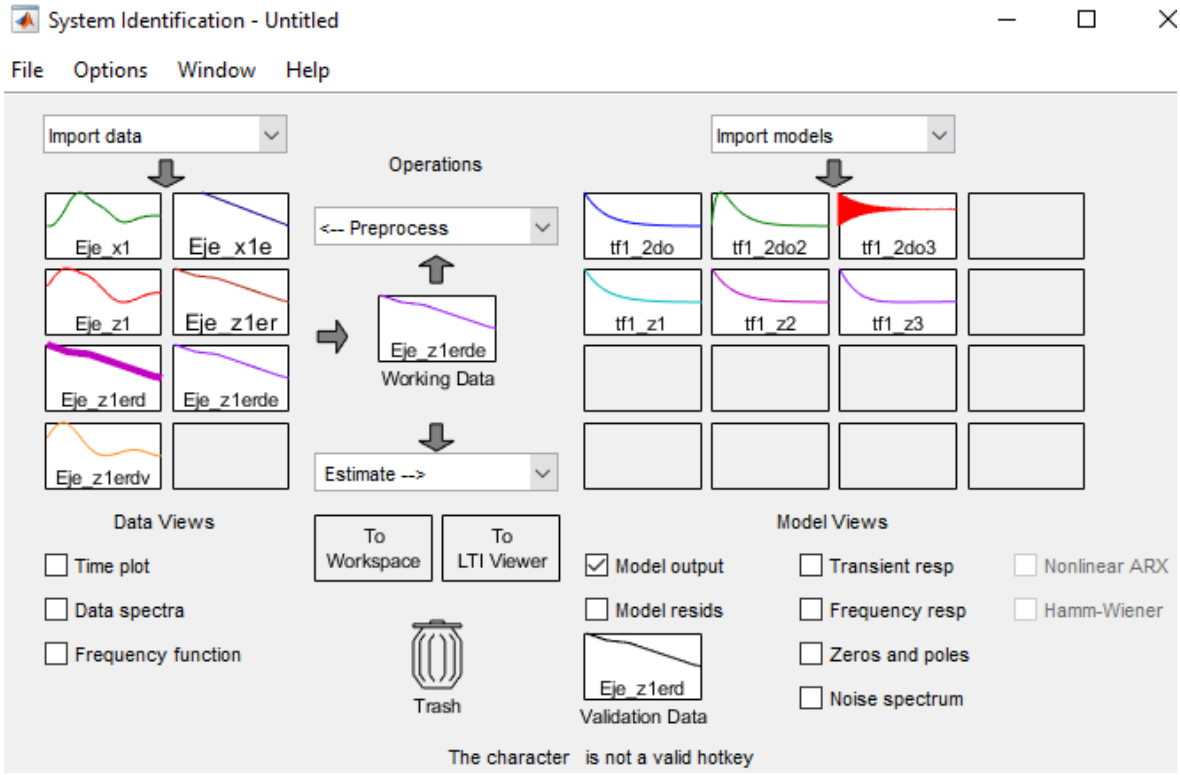


Figura VII.8. Manejo de datos obtenidos a partir del proceso de la Figura VII.7.

Los datos obtenidos de posición obtenidos de la Figura VII.9 se envía a la aplicación del IDENT en la Figura VII.8, aquí se pude obtener una función aproximada, para esto se seleccionó la función de transferencia de tercer orden, ya que es la que mejor respuesta presentó, esta respuesta se muestra en la Figura VII.9.

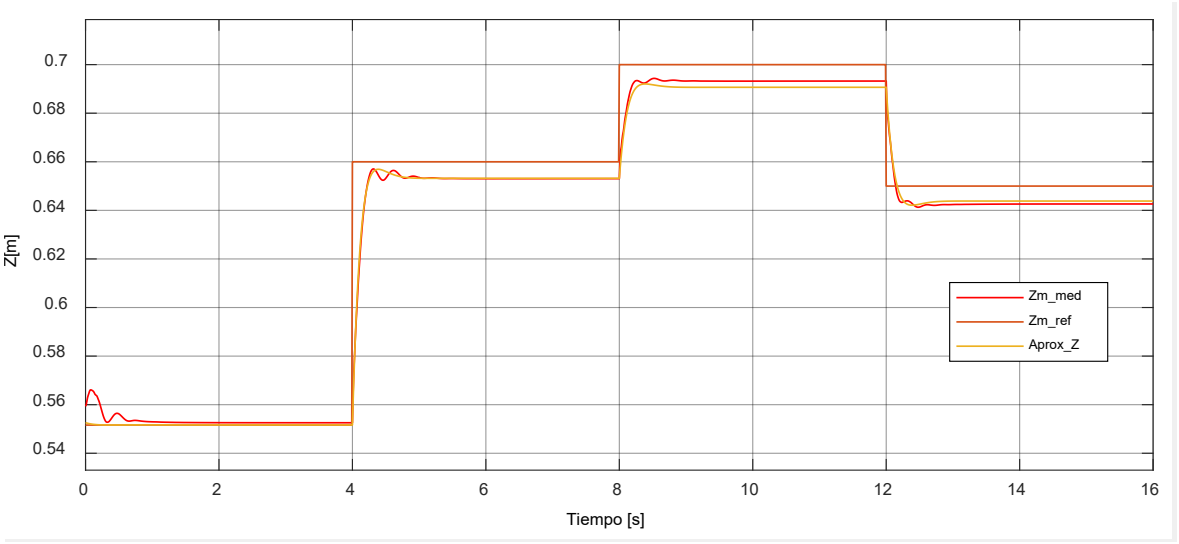


Figura VII.9. . Respuesta del sistema aproximado hallado en el eje Z.

La respuesta de la función aproximada hallada mediante IDENT de MATLAB se muestra en la Figura VII.9, aquí se observa una aproximación aceptable, ya que este se acerca bastante a la señal de la respuesta del sistema de la Ecuación (VII.2)

$$G_z = \frac{8.92s + 61.55}{s^2 + 14.18s + 65.71} \qquad \text{(VII.2)}$$

Finalmente, con la aproximación del sistema obtenido se ingresa este sistema al Sisotool de MATLAB, por medio de esto se hallan los parámetros del controlador.

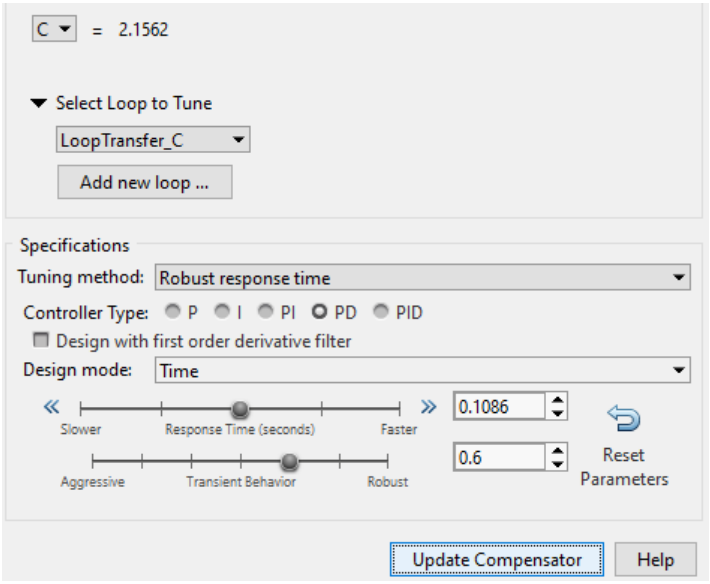


Figura VII.10. Sintonización de los parámetros del PID para la función de transferencia en el eje X

ANEXO VIII

ENLACES

Manual de usuario extendido: Manual_de_usuario_3UPSRPU.docx

Video: Prueba de las trayectorias creadas y mostradas en este documento, triángulo, círculo y rombo.

Trayectoria circular: https://youtu.be/uPldltaaYA0

Trayectoria triangular: https://youtu.be/yXU1ozv0yFk

Trayectoria rombo: https://youtu.be/DGfFNzuvQQE