ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

DISEÑO Y SIMULACIÓN DE CONTROLADORES ENFOCADOS AL SEGUIMIENTO DE TRAYECTORIA TIPO PID Y SMC CON PREDICTOR DE SMITH PARA UNA PLATAFORMA ROBÓTICA MÓVIL PIONEER 3DX.

TOMO II

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRONICA Y AUTOMATIZACION

JAVIER BOLAÑOS DE LA TORRE

DIRECTOR: Dr. PAULO CÉSAR LEICA ARTEAGA

Distrito Metropolitano de Quito, febrero 2022

CERTIFICACIONES

Yo, Javier Bolaños De La Torre 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.

Sr. Javier Bolaños De La Torre

Certifico que el presente trabajo de integración curricular fue desarrollado por Javier Bolaños De La Torre, bajo mi supervisión.

Dr. Paulo César Leica Arteaga
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.

Sr. Javier Bolaños De La Torre - Estudiante

Dr. Paulo César Leica Arteaga - Director

Sr. Juan Diego Zambrano Torres - Colaborador

AGRADECIMIENTO

Primero agradezco a la Escuela Politécnica Nacional (EPN) por permitirme ser parte de una institución que refleja excelencia, dedicación y trabajo duro, también por ayudarme a conocer profesores y compañeros increíbles que me han ayudado a crecer tanto académica como personalmente, gracias a ellos he podido aprender lecciones de vida que llevaré conmigo a cualquier lugar al que vaya.

Agradezco infinitamente a toda mi familia padres, primos, tíos y amigos los cuáles han confiado en mí y han sido un apoyo incondicional frente a todos los problemas que se me han presentado a lo largo de la carrera, especialmente agradezco a mi madre Magdalena asimismo a mis abuelitos Alicia y Hugo que son y serán las personas más importantes en mi vida, gracias a su guía, amor y consejos he podido hacer frente a cualquier adversidad y me inspiraron a convertirme en una mejor persona.

Quisiera agradecer también al asesor de este trabajo, el profesor Paulo Leica, PhD. quién ha sido un mentor no solamente durante la realización de este trabajo, sino a lo largo de toda la carrera, pues ha sabido aportar conocimientos y consejos que fueron y serán esenciales en el ámbito académico y profesional.

De igual forma, agradezco al profesor Oscar Camacho, PhD. quien fue uno de los primeros maestros que inculcó en mi la idea de la automatización como forma de ayuda y mejora en la calidad de vida de las personas, su creatividad y apoyo fueron una motivación para aprender todo lo relacionado al control.

Agradezco también a Juan Diego quien fue compañero y amigo a lo largo de mi formación profesional además de colaborador en la realización del diseño teórico en este trabajo, su apoyo y entusiasmo fueron invaluables para llegar hasta esta instancia, gracias también a mis amigos Luis, Joe y demás compañeros que colaboraron directa e indirectamente en mi vida universitaria.

Finalmente me gustaría agradecer de forma general a los alumnos, miembros administrativos, profesores, coordinadores, decanos y autoridades de la EPN ya que con su compromiso y trabajo son quienes vuelven a esta Universidad una de las más grandes del país y facilitan el poder llenarse de orgullo al decir “Soy Politécnico”.

ÍNDICE DE CONTENIDO

TOMO II

RESUMEN

En el TOMO I de este trabajo se encuentra el análisis, modelamiento, desarrollo matemático y conceptos necesarios para definir las distintas leyes de control en tiempo continuo y discreto permitiendo implementarlas tanto en CoppeliaSim como en SIMULINK dentro del presente tomo.

El TOMO II se centrará en el desarrollo de una interfaz que permitan simular el comportamiento de una planta robótica móvil como lo es el Pioneer 3DX tanto en SIMULINK como en CoppeliaSim usando un control codificado dentro de MATLAB, esta interfaz deberá permitir observar y obtener gráficas de señales importantes como: error, señal de control y seguimiento de referencias, además de proporcionar datos de los índices de desempeño para evaluar la eficiencia de los controladores. Todo esto con la finalidad de realizar un análisis y comparación que permita determinar el tipo de esquema que mejores resultados obtuvo. Los controles que se usarán para las comparaciones son: Un control de lazo simple con Lyapunov y dos controladores con esquema cascada, con un control para el lazo externo usando Lyapunov y para el lazo interno tanto un PID como un Control por Modos Deslizantes (SMC) con una arquitectura de Predictor de Smith (PS).

Finalmente se presentarán y comentarán los resultados obtenidos en las distintas pruebas realizadas, para así demostrar las ventajas y retos que presenta el control SMC + PS frente a controladores más comunes como el PID.

PALABRAS CLAVE: Pioneer 3DX, Seguimiento de trayectoria, Control por Modos Deslizantes, Predictor de Smith, Interfaz gráfica de usuario.

ABSTRACT

In the VOLUME I of this document it will be possible to find the analysis, modeling, mathematical development and concepts necessary to define the different control laws in continuous and discrete time, allowing them to be implemented in both CoppeliaSim and SIMULINK within this volume.

The VOLUME II focuses on the development of two interfaces that allows to simulate the behavior of a mobile robotic plant like the Pioneer 3DX in both SIMLUINK and CoppeliaSim using a codified control in MATLAB, this applications must allow to observe and obtain the graphs of the most important signals like the error, control signal and the reference tracking, it should also allow to evaluate the efficiency of the controller through the use of performance indices, this with the purpose of carrying out an analysis and comparison to determine which type of control scheme obtain better results. The controllers applied are A simple loop controller based in Lyapunov and two cascade controllers using a Lyapunov design in the external loop and in the internal loop a conventional PID or a Sliding Mode Controller (SMC) with a Smith Predictor (PS) architecture.

Finally, the results obtained in the different test will be exposed and commented to demonstrate the advantages and the challenges of the cascade control scheme with the SMC + PS in the inner loop compared against more commonly used controllers like the PID.

KEYWORDS: Pioneer 3DX, trajectory tracking, Sliding Mode Controller, Smith Predictor, Graphic User Interface.

1 INTRODUCCIÓN

En este trabajo, se realiza la implementación de los modelos y controladores tanto en SIMULINK como en MATLAB con un ambiente de simulación en CoppeliaSim, con el fin de utilizar una plataforma robótica que considere aspectos dinámicos para así evidenciar el desempeño de cada uno de los tres controladores propuestos y comparar sus características usando como planta un modelo aproximado del robot real.

Las trayectorias que se evaluarán para el seguimiento son tres: una trayectoria elíptica, triangular y tipo L, cada una de ellas con su peculiaridad para evaluar el comportamiento del robot ante distintas condiciones.

Además, se plantean pruebas a las que serán sometidos los distintos controladores a evaluarse, de estas pruebas se obtendrán resultados que podrán ser visualizados y analizados desde una interfaz de usuario tanto en SIMULINK como en AppDesigner.

1.1 OBJETIVO GENERAL

Diseñar y simular controladores enfocados al seguimiento de trayectoria tipo PID convencionales, y SMC con predictor de SMITH para una plataforma robótica móvil Pioneer 3DX.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar pruebas experimentales mediante simulación en MATLAB-SIMULINK para el sistema en cascada.
- Implementar una interfaz gráfica que permita probar los controladores propuestos y evidenciar su funcionamiento en el software CoppeliaSim.
- Realizar pruebas a nivel de simulación para evaluar el desempeño de los controladores propuestos, mediante el análisis de índices de desempeño.

1.3 ALCANCE

- Se realizarán pruebas experimentales en MATLAB-SIMULINK para el sistema en cascada.

- Se realizará la transmisión y obtención de datos entre MATLAB y CoppeliaSim, mediante la aplicación de comandos API y la programación interna del Pioneer, con el fin de observar un modelo tridimensional de su funcionamiento.
- Se implementará una interfaz gráfica en MATLAB-SIMULINK y CoppeliaSim, utilizando las herramientas que provee la extensión “AppDesigner” y otras librerías de SIMULINK.
- Se realizará la comparativa de los controladores mediante índices de desempeño tales como ISE e ISCO.

2 METODOLOGÍA

El presente documento se basa en la investigación cuantitativa por métodos descriptivos y experimentales, la recolección de datos se realizó mediante mediciones, observaciones y pruebas de simulación de los controladores implementados.

Se plantea en este trabajo realizar la configuración y programación necesarias para implementar dentro de MATLAB y SIMULINK los controladores, trayectorias e índices de desempeño, de igual forma se revisarán características y tipos de comunicación entre las aplicaciones MATLAB y CoppeliaSim, esto para poder simular en un ambiente tridimensional el efecto y resultados de agregar un controlador de seguimiento a la planta robótica móvil Pioneer 3DX. Finalmente se implementarán dos interfaces de usuario una en AppDesigner y otra dentro de SIMULINK que permitirán el visualizar y obtener datos de forma más sencilla y servirán como base en la adquisición de información para una posterior comparación, análisis y planteamiento de resultados y conclusiones.

2.1 INTERFAZ MATLAB/ SIMULINK

El desarrollo de una interfaz es muy importante ya que será un medio de interacción entre el usuario y la planta controlada, la interfaz permitirá modificar parámetros de simulación, cambiar las trayectorias a seguir y modificar el tipo de control que se usará para el seguimiento así como sus parámetros, al mismo tiempo la interfaz permite visualizar la evolución de ciertas señales importantes lo que puede ser útil para determinar qué tan bueno es el rendimiento y eficiencia del controlador simulado.

La implementación de una interfaz dentro de SIMULINK tiene como finalidad obtener gráficos e índices de rendimiento para los distintos controladores para posteriormente compararlos con lo obtenido en CoppeliaSim.

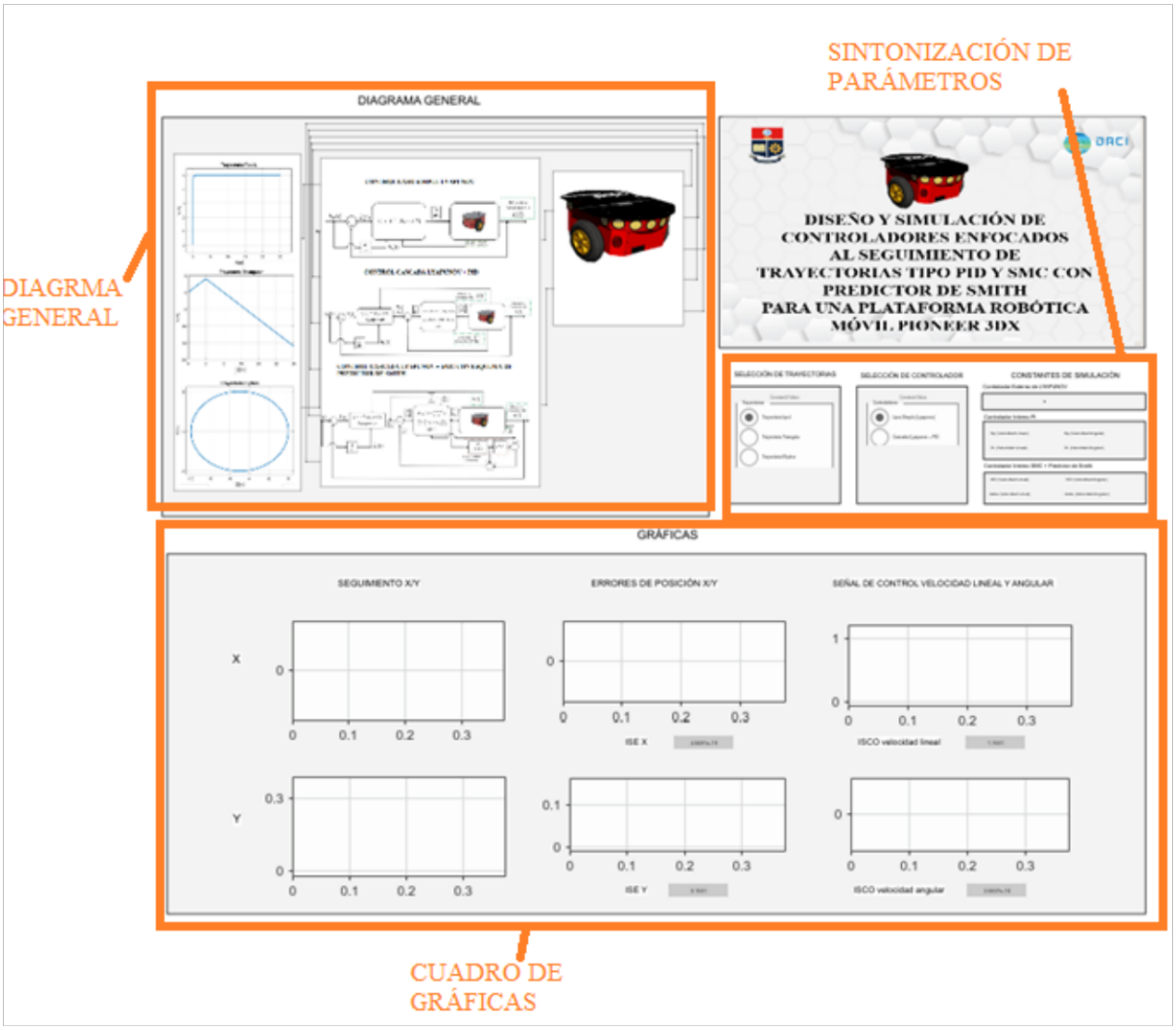


Figura 2.1. Interfaz en Simulink

En la Figura 2.1 se puede observar la interfaz desarrollada, así como una división en 3 bloques: Diagrama General, Parámetros de Simulación y Cuadro de Gráficas.

Parámetros de Simulación: Dentro de los parámetros se puede modificar el tipo de trayectoria, el tipo de controlador a usarse y los parámetros de cada controlador individualmente.

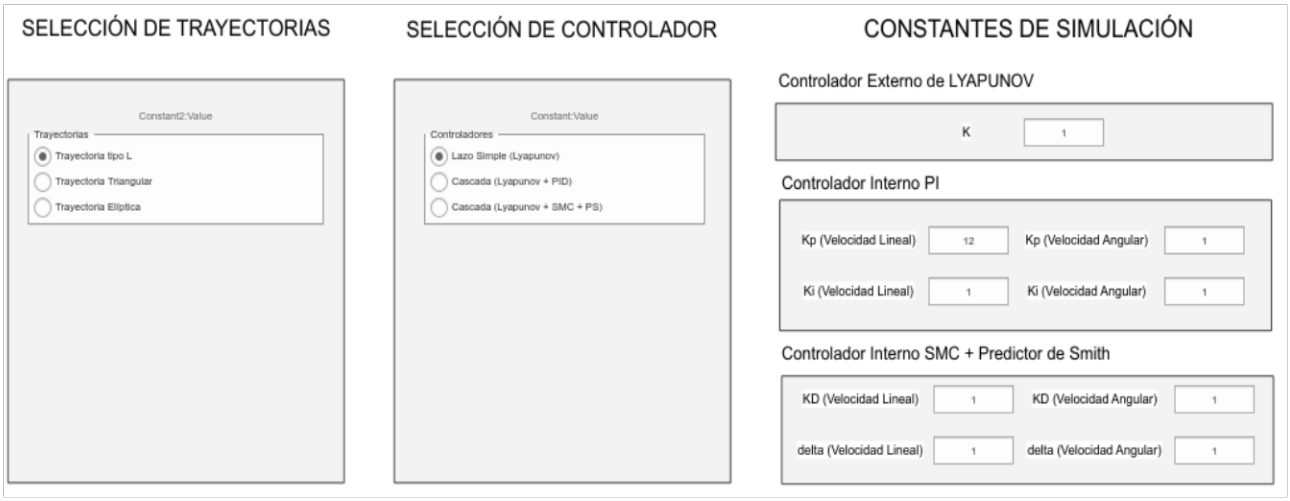


Figura 2.2. Bloque de Parámetros en la Interfaz

Cuadro de Gráficas: En el cuadro de gráficas es posible observar el error de posición XY, la señal de control de velocidades lineal y angular, el seguimiento en XY de la trayectoria y los índices ISE e ISCO.

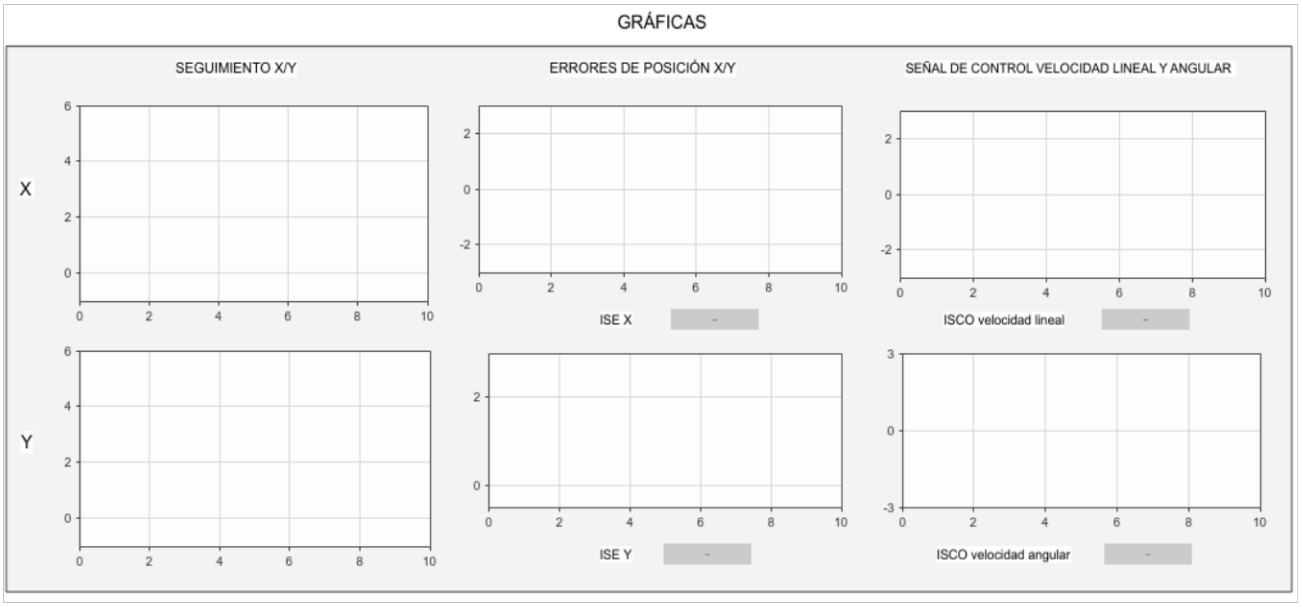


Figura 2.3. Bloque de Gráficas en la interfaz

Diagrama General: El diagrama general presenta un esquema en el que se encuentran: trayectorias, controladores y modelo de la planta.

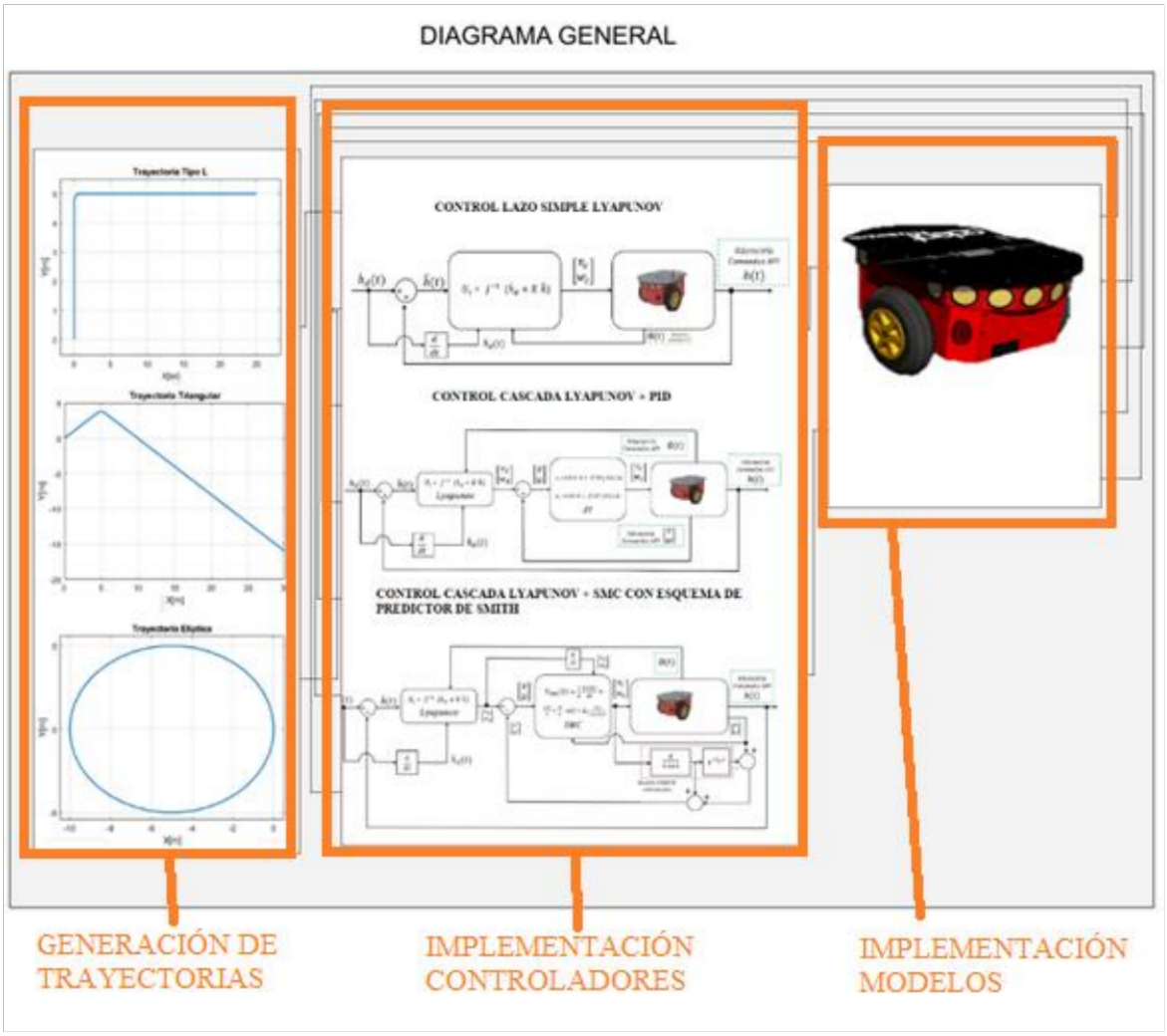


Figura 2.4. Diagrama general de la interfaz

La programación y consideraciones tomadas en cuenta para cada bloque de la Figura 2.4 se lo realizará de forma más detallada a continuación.

2.1.1 GENERACIÓN DE TRAYECTORIAS EN TIEMPO CONTINUO

Para evaluar el seguimiento se proponen 3 trayectorias:

- Trayectoria elíptica es la condición menos forzada para el robot, pues tiene naturaleza curva, lo que se adapta perfectamente a la restricción no holonómica.
- Trayectoria triangular permite evidenciar el cambio de la velocidad angular ante un giro repentino.
- Trayectoria tipo L que permite evidenciar la respuesta del robot ante cambios bruscos, donde la derivada de la referencia es muy alta en un punto específico.

En la Tabla 2.1 se presentan las distintas trayectorias separadas en tramos.

Tabla 2.1. Trayectorias en tiempo continuo

Tipo de Trayectoria	Tramo I para t ≤ 8		Tramo II para t > 8	
	$\mathbf{X(t)}$	$\mathbf{Y(t)}$	$\mathbf{X(t)}$	$\mathbf{Y(t)}$
Elíptica	$5\cos(t) - 5$	$5sin(t)$	$5\cos(t) - 5$	$5sin(t)$
Triangular	t	$0{,}8\ t$	t	$8 - 0{,}8\ t$
Tipo L	0	t	$t - 8$	8

2.1.2 IMPLEMENTACIÓN DE MODELOS EN MATLAB – SIMULINK

En este apartado se realizará la implementación de los diferentes modelos que se desarrollaron en el marco teórico del TOMO I.

2.1.2.1 Implementación del modelo cinemático tipo uniciclo simple

En la Figura 2.5 se implementa como diagrama de bloques el modelo cinemático de un robot móvil tipo uniciclo el cual está dado por las ecuaciones 1.7, 1.8 y 1.9 del subcapítulo 1.4.4.3 que se encuentra en el TOMO I

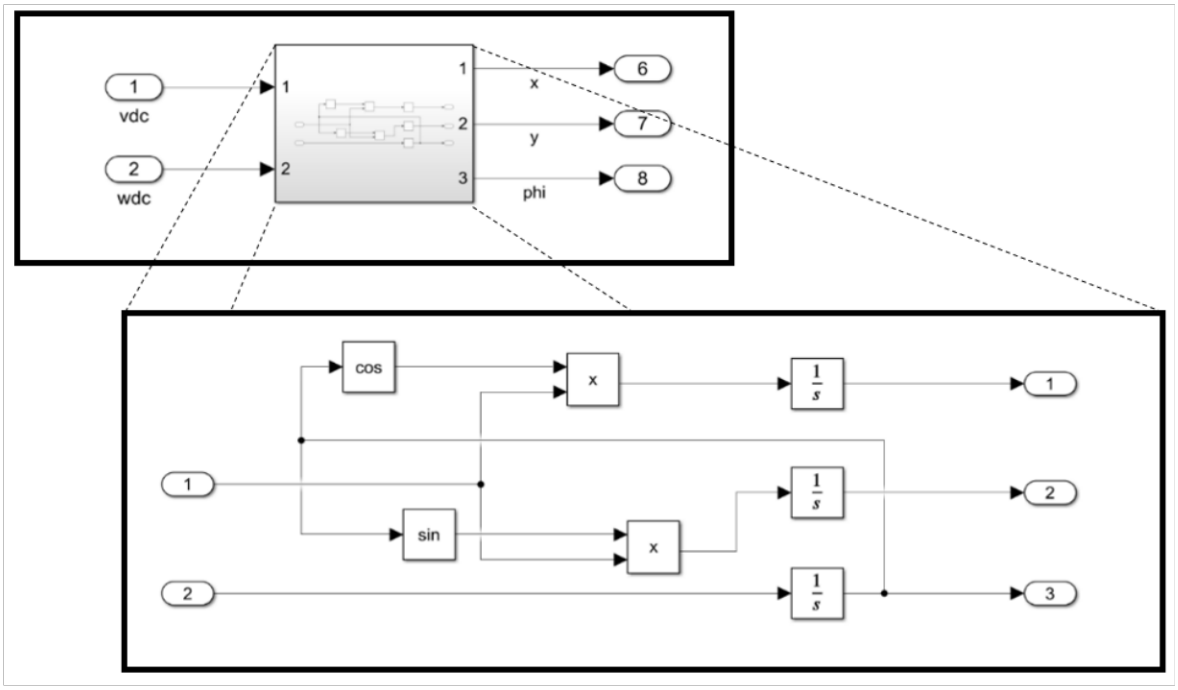


Figura 2.5. Diagrama de bloques del modelo cinemático

2.1.2.2 Implementación del modelo cinemático uniciclo con restricción no holonómica mejorada

En la Figura 2.6 se implementa como diagrama de bloques el modelo cinemático con restricción no holonómica mejorada de un robot móvil tipo uniciclo el cual está dado por la ecuación 1.13 del subcapítulo 1.4.4.4 que se encuentra en el TOMO I

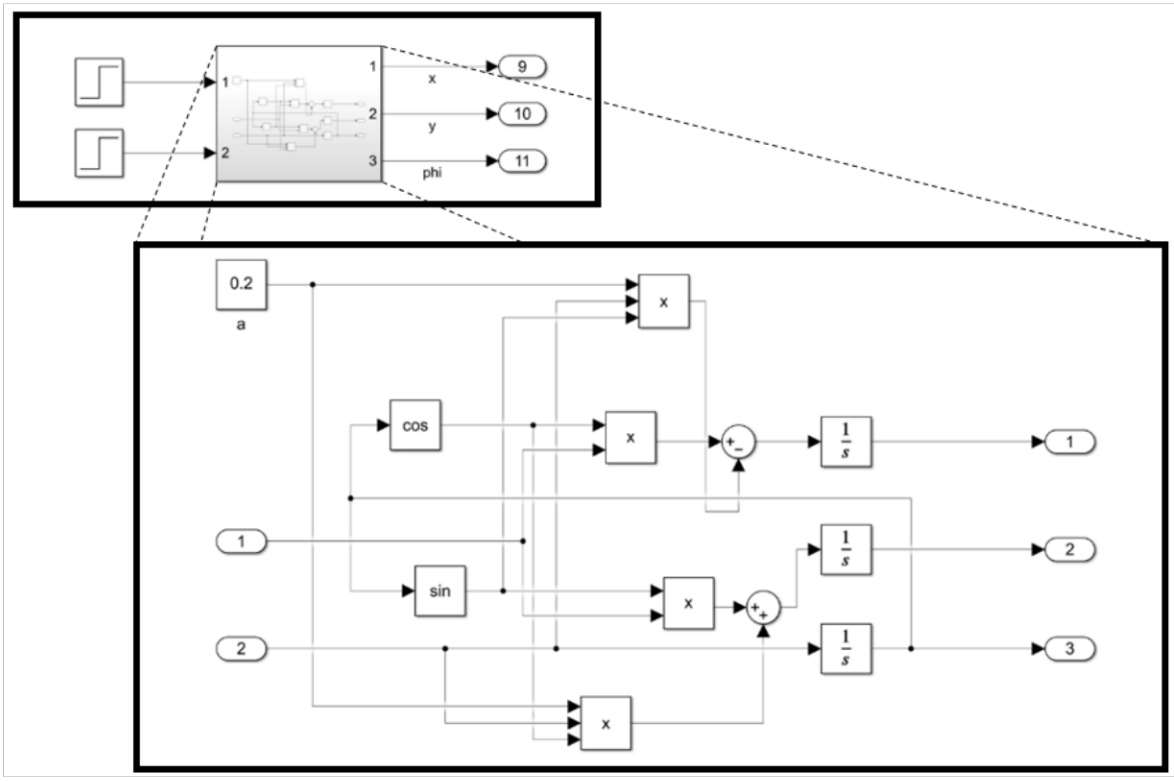


Figura 2.6. Diagrama de bloques con restricción no holonómica mejorada

Este modelo presenta la referencia a una distancia fija del punto de interés del robot, sin embargo, no se considera las características dinámicas.

2.1.2.3 Implementación del modelo dinámico del robot Pioneer 3DX

En la figura 2.7 se muestra el diagrama de bloques del modelo obtenido en la ecuación 1.19, este modelo considera las características constructivas y las fuerzas asociadas al robot real, tal como se detalla en el análisis del marco teórico y se adjunta en el ANEXO I:

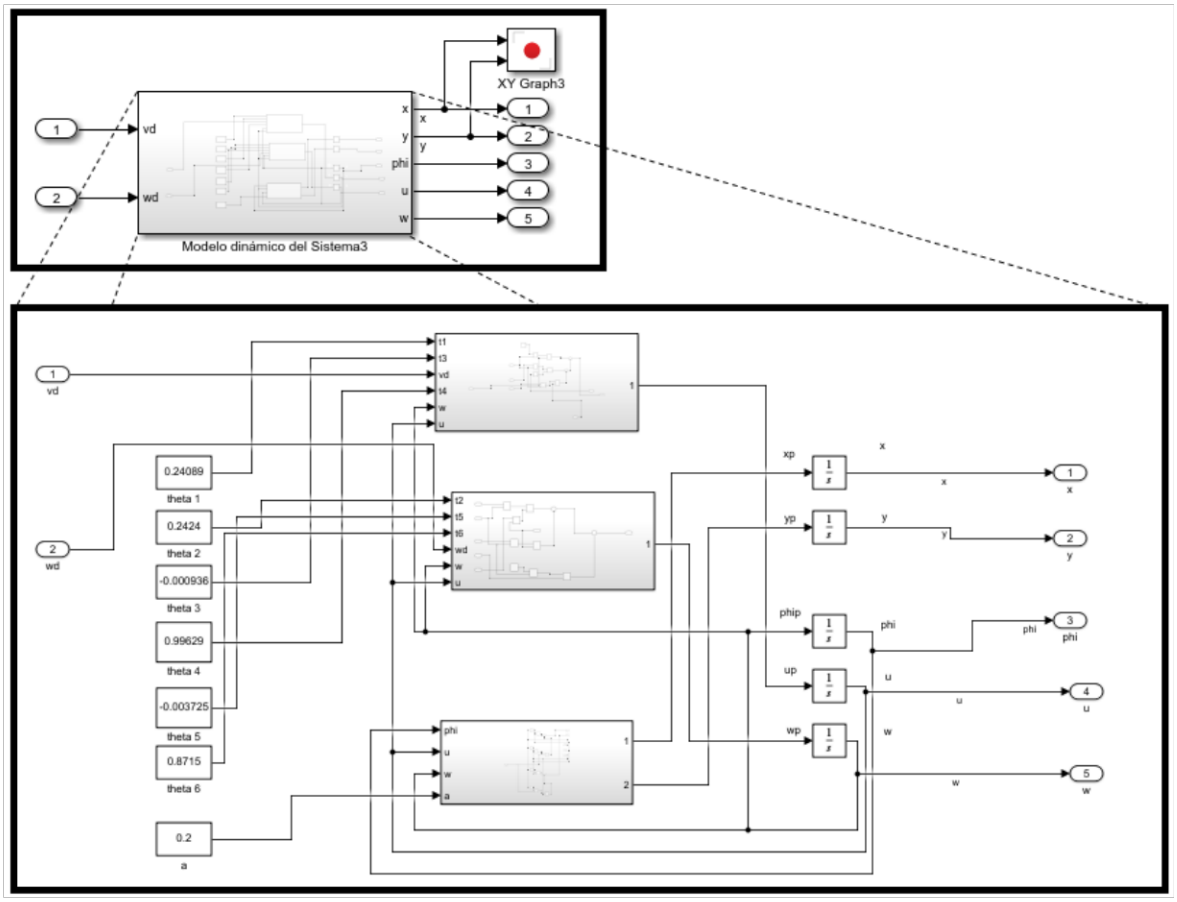


Figura 2.7. Diagrama de bloques modelo dinámico

2.1.3 IMPLEMENTACIÓN DE CONTROLADORES EN MATLAB – SIMULINK

Para la implementación en MATLAB-SIMULINK de los controladores se usarán las ecuaciones encontradas en el TOMO I.

2.1.3.1 Implementación del controlador en lazo simple Lyapunov

El primer controlador a implementarse dentro de SIMULINK es el de Lyapunov en lazo simple mostrado en la Figura 2.8, la acción de control viene dada por la ecuación 2.23 del TOMO I.

Obteniendo el siguiente diagrama de bloques con su respectivo detalle.

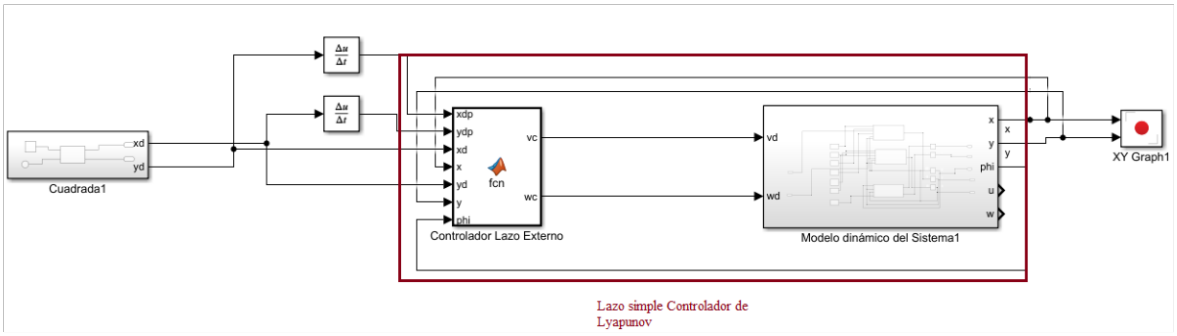


Figura 2.8. Implementación del controlador de Lyapunov en lazo simple

2.1.3.2 Implementación del controlador en cascada Lyapunov + PI

El segundo controlador a implementarse dentro de SIMULINK es un control en cascada con lazo externo de Lyapunov y lazo interno un PI mostrado en la Figura 2.10, la acción de control viene dada por las ecuaciones 2.34 y 2.35 del TOMO I.

Obteniendo el siguiente diagrama de bloques con su respectivo detalle.

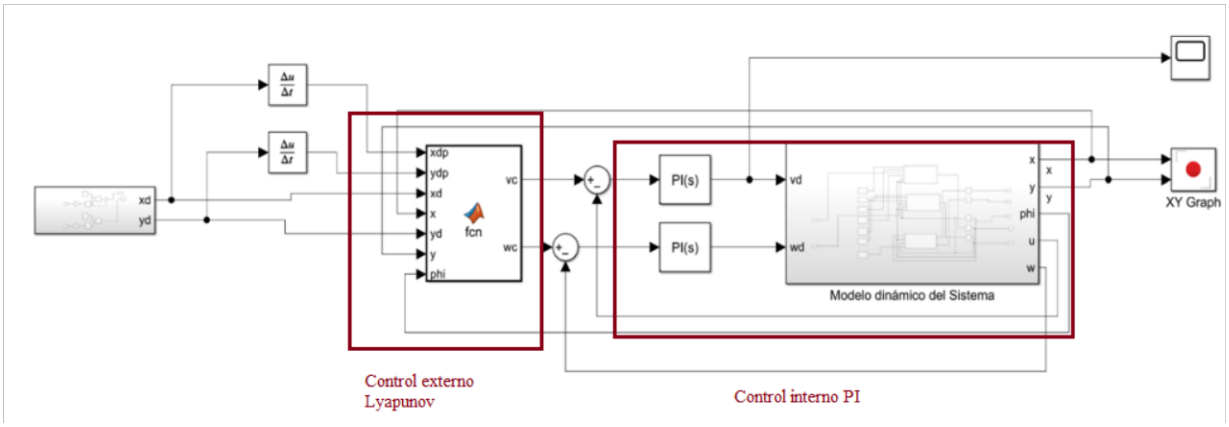


Figura 2.10. Implementación del controlador cascada con control interno PI

2.1.3.3 Implementación del controlador en cascada Lyapunov + SMC con esquema de Predictor de Smith

El segundo controlador a implementarse dentro de SIMULINK es un control en cascada con lazo externo de Lyapunov y lazo interno un SMC con esquema de Predictor de Smith mostrado en la Figura 2.12, la acción de control viene dada por las ecuaciones 2.50 y 2.51 del TOMO I. Obteniendo el siguiente diagrama de bloques:

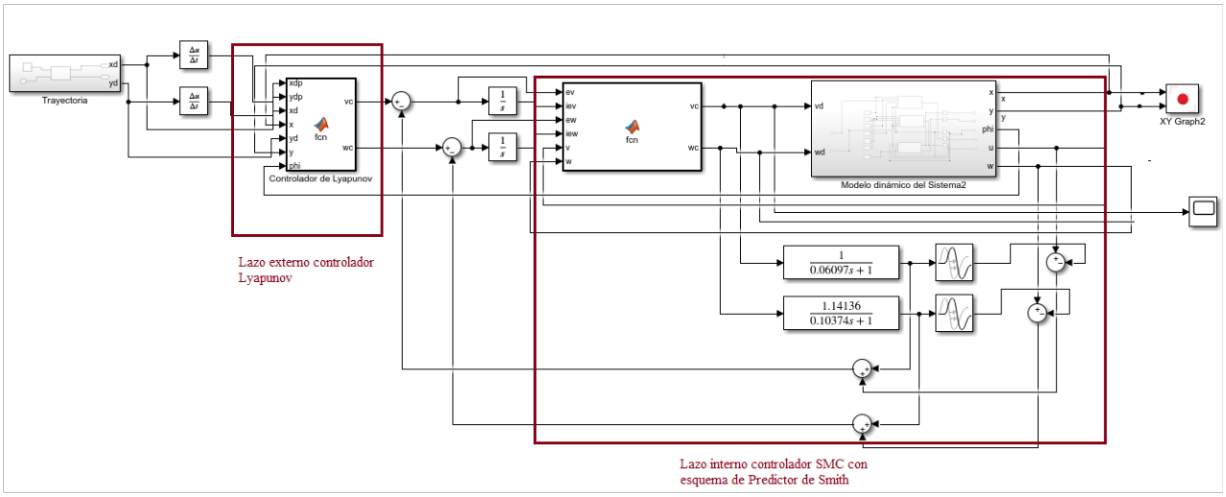


Figura 2.12. Implementación del controlador cascada con control interno SMC y SP

Todos los códigos asociados a los controladores y diagramas de bloques internos que se encuentran dentro de los bloques “subsistemas” se encuentran disponibles en el Anexo III

2.2 INTERFAZ GRÁFICA MATLAB - COPPELIASIM

La interfaz gráfica de usuario o GUI es una aplicación que permitirá realizar un monitoreo, control y comparación de los controladores y señales de respuesta obtenidos en la simulación del seguimiento de trayectoria por parte del robot Pioneer 3DX. Para cumplir con esos objetivos se plantearon cuatro pantallas en las que se destinará una función para cada una.

2.2.1 PANTALLA PRINCIPAL

La pantalla principal tiene la funcionalidad de ser el punto de partida de la simulación, en esta se puede ver el título del trabajo de titulación, los autores y unos botones de ‘Inicio’ y ‘Salir’, esto se puede ver en la Figura 2.13

Figura 2.13. Pantalla Principal

A continuación, se detallará las funciones de los elementos de la pantalla principal:

Controles de Pantalla: Son los botones de minimizar, maximizar y cerrar la ventana

Botones de Control:

- **Inicio:** Este botón permite seguir con la siguiente ventana de simulación.
- **Salir:** Este botón permite salir de la simulación

2.2.2 PANTALLA DE CONTROLADORES

La pantalla de controladores tiene la funcionalidad el elegir el tipo de esquema de control, trayectoria a seguir y las constantes del control que se van a usar para realizar la simulación, dentro de esta pantalla los valores óptimos de simulación están seteados por defecto.

En la pantalla se tiene los cuadros de selección de controlador, selección de trayectoria, modificación de constantes y las flechas de navegación como se puede ver en la Figura 2.14.

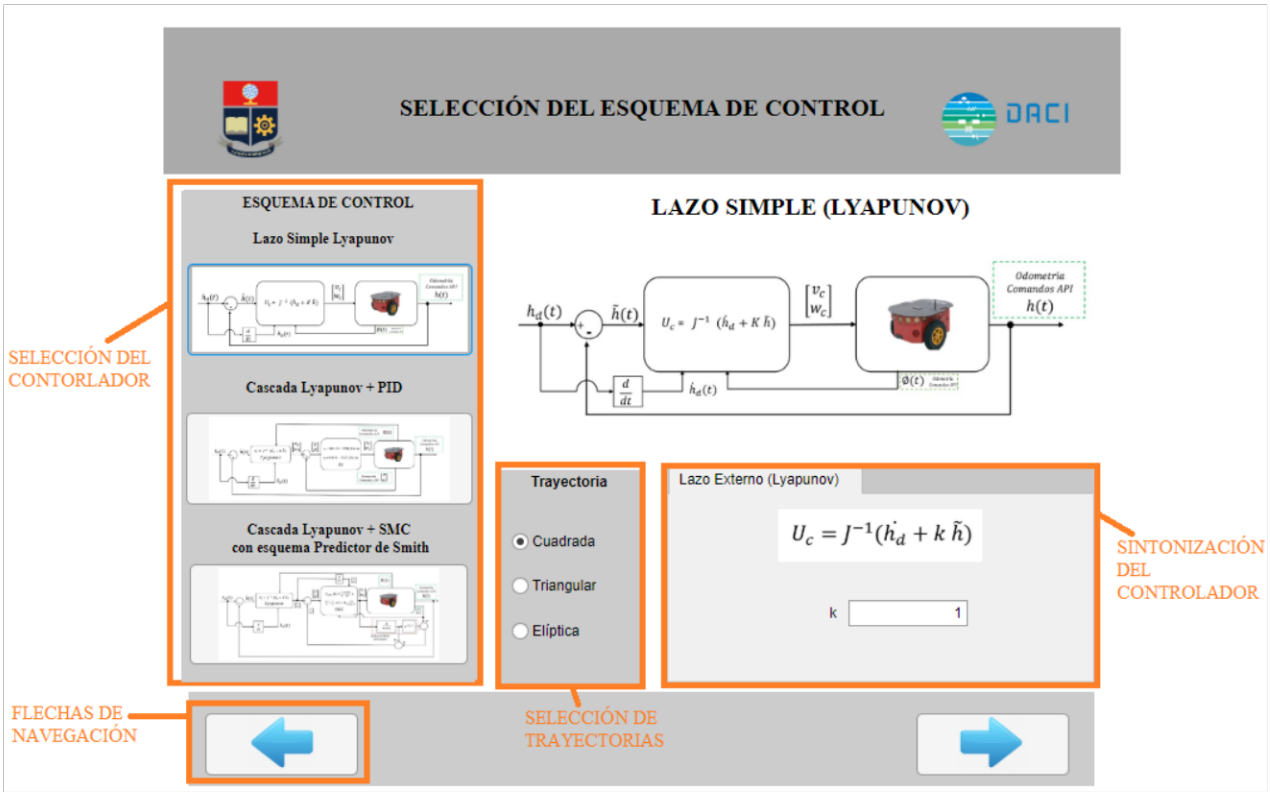


Figura 2.14 Pantalla de Controladores

A continuación, se detallará las funciones de los elementos de la pantalla de controladores:

Selección de Controlador: Permite elegir el tipo de control a simular en orden son:

1. Control en lazo simple (Lyapunov)
2. Control en cascada (Lyapunov + PID)
3. Control en cascada (Lyapunov + SMC con Predictor de Smith)

Selección de Trayectoria: En este cuadro se elige la trayectoria de referencia son 3: (Tipo L, Triangular y Elíptica)

Sintonización del Controlador: Permite cambiar el valor de las constantes que pueden afectar la respuesta del controlador, los parámetros que se pueden sintonizar son los que están en la Figura 2.15 y son:

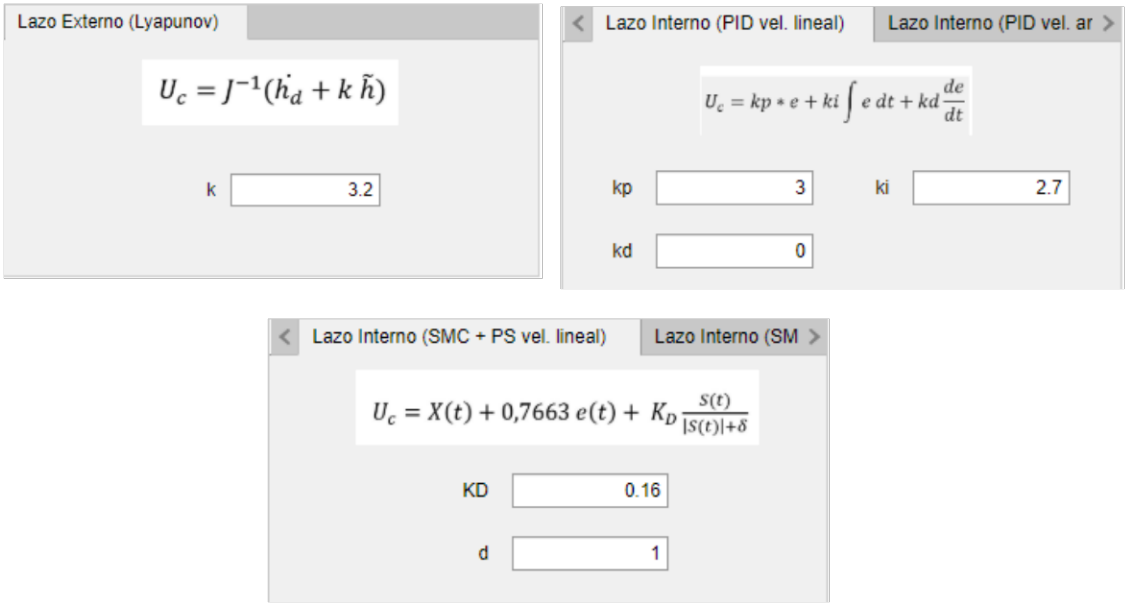


Figura 2.15 Pantalla de Controladores

- **Lyapunov:** La constante K
- **Lyapunov + PID:** La constante K y las constantes proporcional e integral (kp y ki), tanto para la velocidad lineal como la angular
- **Lyapunov + SMC con Predictor de Smith:** La constante K, las constantes K_D y delta tanto para la velocidad lineal como la angular.

Flechas de Navegación: Permite navegar entre la pantalla anterior y siguiente.

2.2.3 PANTALLA DE SIMULACIÓN

La pantalla de simulación tiene la funcionalidad de iniciar la simulación, presentar los datos de error, señal de control y seguimiento obtenido y mostrar los índices de desempeño.

En la pantalla se tiene el cuadro de simulación, el cuadro de gráficas y el de índices de desempeño como se puede ver en la Figura 2.16.

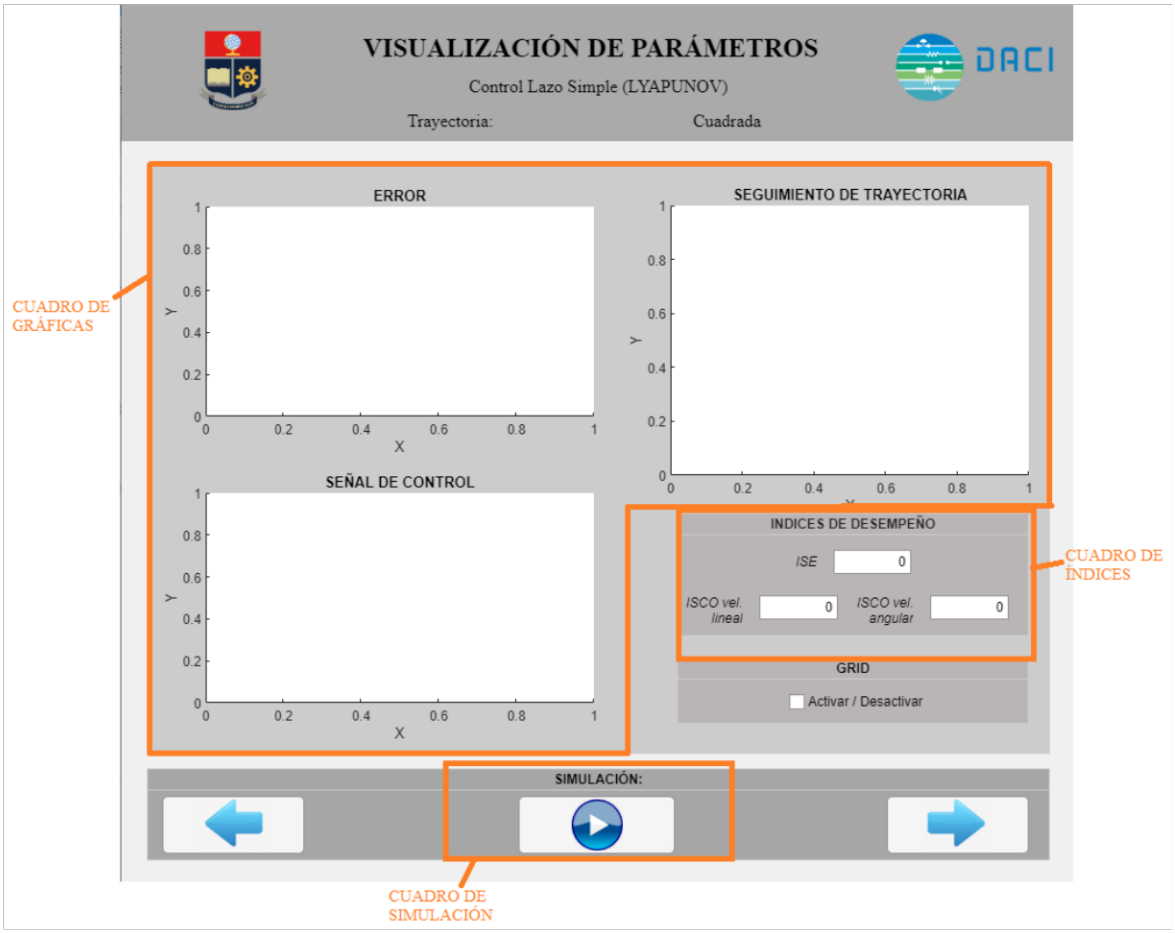


Figura 2.16 Pantalla de Simulación

A continuación, se detallará las funciones de los elementos de la pantalla de simulación:

Cuadro de Simulación: Permite empezar con la simulación, tiene esta forma para poder acoplarse a CoppeliaSim como una barra de simulación como se puede ver en la Figura 2.17

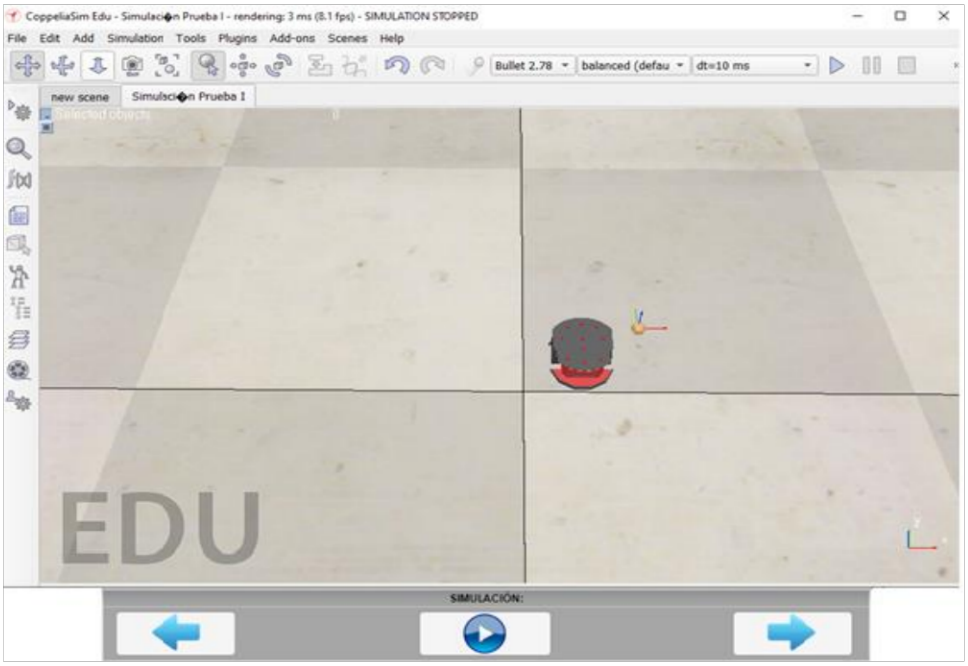


Figura 2.17 Pantalla de Simulación

Cuadro de Gráficas: Permite visualizar cómo evoluciona una señal en el tiempo, las señales que se grafican son:

- **Error:** Se grafica individualmente los errores X y Y
- **Señal de Control:** La señal de control de la velocidad lineal y la velocidad angular.
- **Seguimiento:** Se mostrará la referencia escogida en conjunto con el seguimiento que realizará el robot.

Cuadro de índices: En este se muestra los siguientes valores:

- **ISE:** Se calcula con el cuadrado del módulo del error X y Y
- **ISCO vel. lineal:** Se calcula con el cuadrado de la señal de control de la velocidad lineal
- **ISCO vel. angular:** Se calcula con el cuadrado de la señal de control de la velocidad angular

Flechas de Navegación: Permite navegar entre la pantalla anterior y siguiente.

2.2.4 PANTALLA DE COMPARACIÓN

La pantalla de comparación tiene la funcionalidad de presentar los datos de los 3 tipos de controladores frente a las 3 distintas trayectorias

En la pantalla se tiene los cuadros de elección de controladores a comparar, cuadro de índices de desempeño, botón de regreso y cuadro de gráficas como se puede ver en la Figura 2.18.

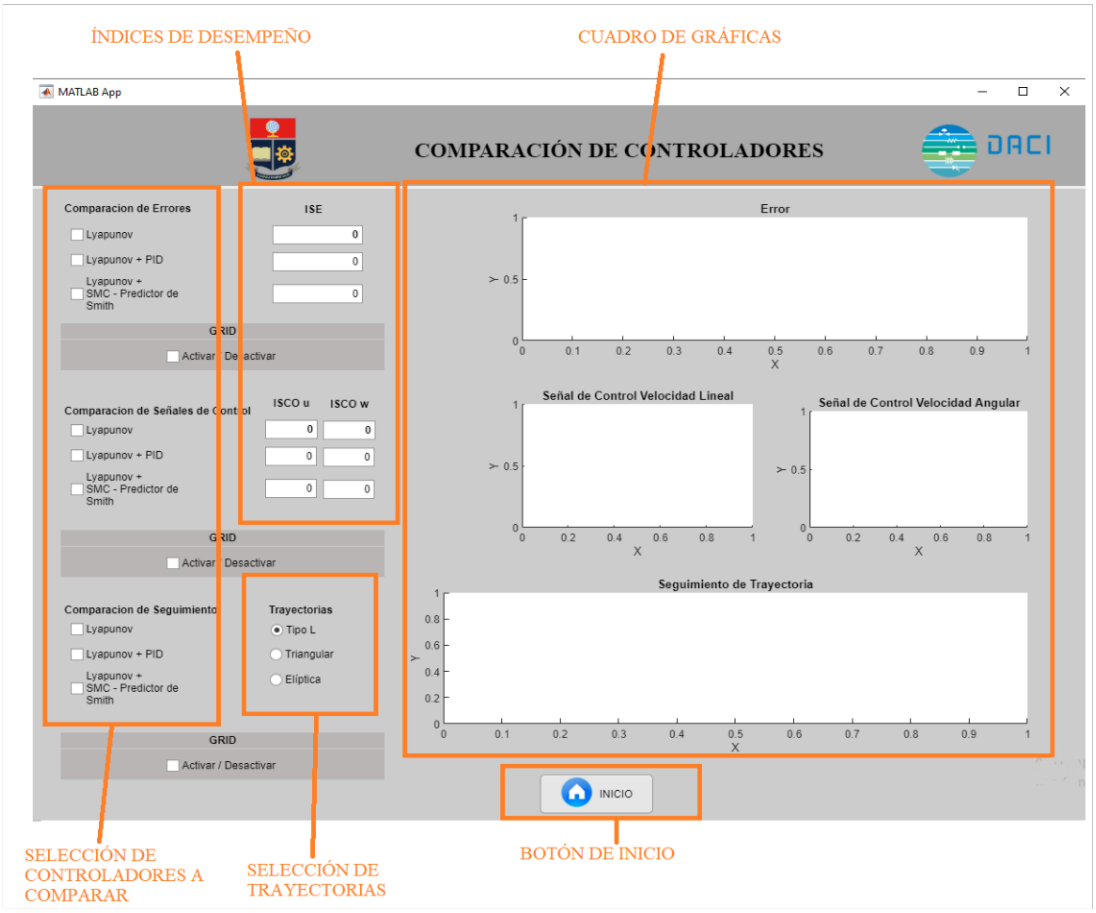


Figura 2.18 Pantalla de Comparación

A continuación, se detallará las funciones de los elementos de la pantalla de simulación:

Cuadro de Elección de Controladores a Comparar: Se tienen casillas de verificación en la que se puede determinar que señales del controlador elegido se graficarán, así como la visualización de su índice de desempeño.

Cuadro de selección de trayectorias: En este cuadro se puede elegir entre los 3 tipos de trayectorias que se detallaron antes, al variar la trayectoria los valores de error, señal de control e índices de desempeño cambian respecto a su nuevo seguimiento.

Cuadro de índices: En estos se mostrarán el valor de ISE que respecta al valor en módulo de los errores en X y Y, los índices ISCO de la velocidad angular y lineal individualmente.

Cuadro de Gráficas: Permite visualizar de forma gráfica tanto el error, señal de control y seguimiento de la planta usando el controlador que se seleccione en el cuadro de elección.

Botón de Regreso: Este botón permite regresar a la pantalla de simulación.

2.3 COMUNICACIÓN ENTRE MATLAB Y COPPELIASIM

La comunicación entre MATLAB y CoppeliaSim es un elemento fundamental para el desarrollo del seguimiento de trayectorias, así como la obtención y presentación de datos. Por defecto esta comunicación se encuentra de forma asincrónica lo que permite a la simulación ser más fluida, aunque no toma en cuenta el progreso del cliente API remoto (MATLAB). Para la aplicación que se desarrollará es necesario que la obtención y transmisión de datos desde un software a otro se realice de forma sincrónica, esto debido a la necesidad de:

- Encontrar un modelo aproximado mediante la curva de reacción.
- Poder tener una misma base de tiempo para graficar curvas de error, señal de control y seguimiento.

Para poder solventar estos requerimientos es necesario programar el tipo de comunicación sincrónica dentro de MATLAB y CoppeliaSim. Para esto se debe determinar.

2.3.1. ESTABLECIMIENTO DE LA COMUNICACIÓN SINCRÓNICA

Para realizar la comunicación sincrónica se debe tener en cuenta cómo actúa el intercambio de datos al usar este modo de operación, se puede revisar en el subcapítulo 1.4.8 y en la Figura 1.16 del Tomo I de este documento, en esa imagen es posible ver de forma general como se desarrollará la programación siguiendo cierto orden y definiendo ciertos parámetros anteriores:

- Se define el tiempo de muestreo en el que trabajarán las aplicaciones y compartirán datos siendo en este caso 10 [ms] para poder tener una respuesta más detallada y fiel de la simulación en CoppeliaSim
- Se define la dirección IP como ‘127.0.0.1’ la cuál es la dirección del servidor (CoppeliaSim) y se define el puerto de comunicación 19997 este debido a que permitía un mejor funcionamiento al simular sincrónicamente. [2]
- Seguido se usarán los comandos API de MATLAB que pueden ser revisados en el subcapítulo 1.4.9.1 en la tabla 1.3 del Tomo I, para cargar las librerías, cerrar cualquier comunicación antes de empezar una nueva e inicializar el clientID con el IP y puerto referido anteriormente.

- Posteriormente se activa el modo de operación sincrónica usando el comando ‘simxSynchronous’.
- Finalmente se inicia la simulación y se realizan las instrucciones necesarias para cada paso de simulación y se envía una señal de disparo con el comando ‘simxSynchronousTrigger’ que permitirá simular el paso de simulación actual y pasar al siguiente. [2]

En la Figura 2.19 se presenta el diagrama de flujo de la comunicación:

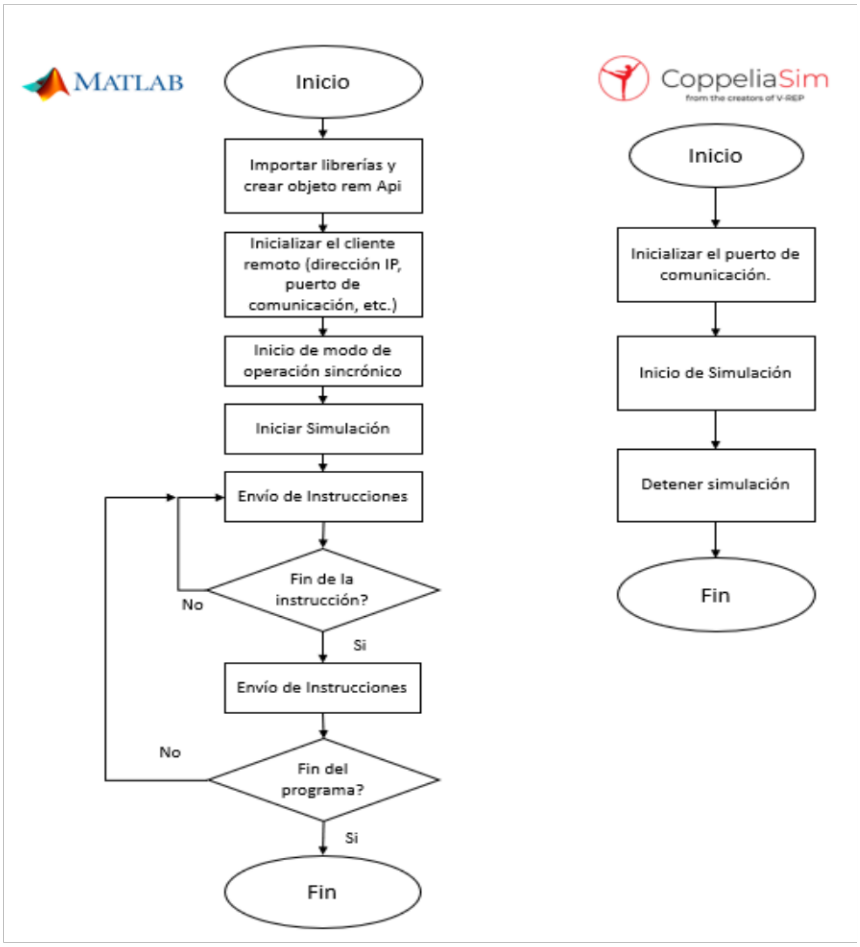


Figura 2.19. Diagrama de flujo de la comunicación sincrónica.

2.4 GENERACIÓN DE TRAYECTORIAS EN TIEMPO DISCRETO

Para poder evaluar la calidad de seguimiento de los controladores es necesario emplear distintos tipos de trayectorias que puedan enfocarse en evaluar condiciones específicas que se requieran del controlador diseñado como:

Para la generación de trayectorias es necesario tener en cuenta el tiempo de simulación requerido, mantener un equilibrio entre poder visualizar todo lo necesario del seguimiento

de trayectoria y que el tiempo de simulación no sea demasiado extenso, por lo que se definió un tiempo de 16 [s] y cada tramo se simulará durante 8 [s].

En este apartado se definirán las trayectorias en tiempo discreto usando un tiempo de muestreo Tm y en la Figura 2.16 Se presenta el diagrama de flujo de las trayectorias.

Tabla 2.2 Funciones del cliente API remoto y descripción [2]

Trayectoria	**Tramo I**		**Tramo II**	
	X(k)	**Y(k)**	**X(k)**	**Y(k)**
Elíptica	5cos(k-1) Tm	5sin(k-1) Tm	5cos(k-1) Tm	5sin(k-1) Tm
Triangular	(k-1) Tm	0.8(k-1) Tm	(k-1) Tm	8 - 0.8(k-1) Tm
Cuadrado	0	(k-1) Tm	(k-1) Tm - 5	5

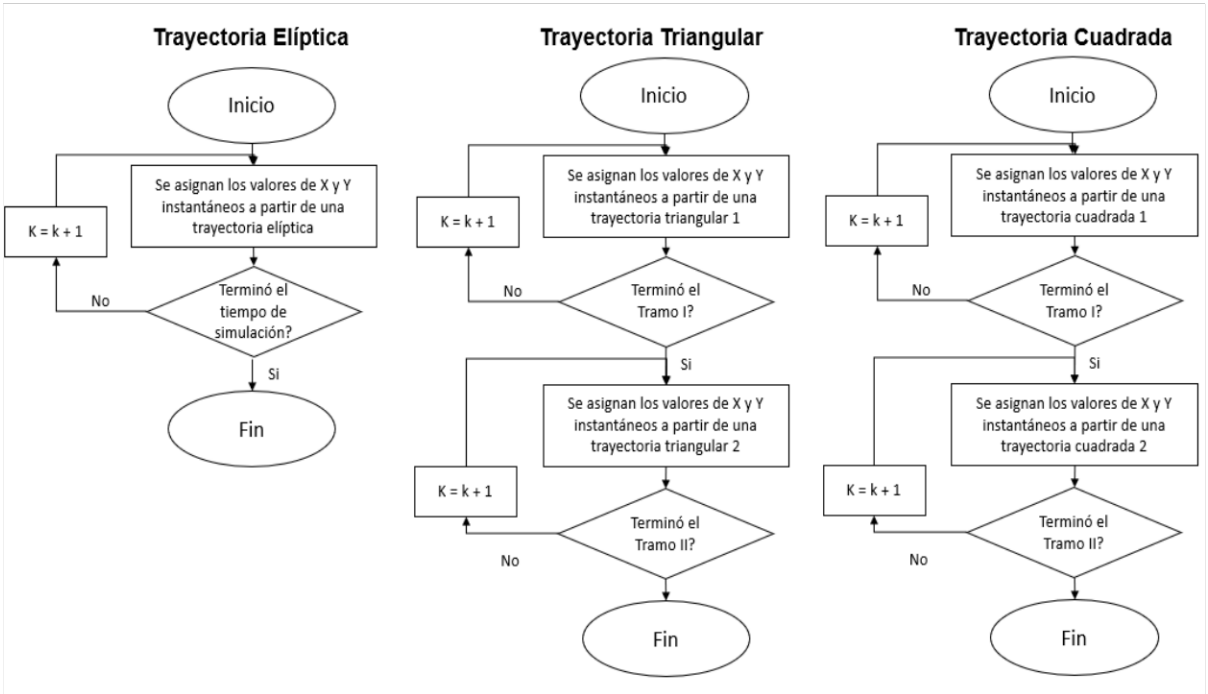


Figura 2.20. Diagrama de flujo de generación de trayectorias.

Las formas de las trayectorias generadas en el plano XY con un tiempo de muestreo de $0{,}01$ segundos se muestran en la figura 2.21 y su código asociado basado en el diagrama de flujo de la Figura 2.22 se encuentra en el ANEXO II:

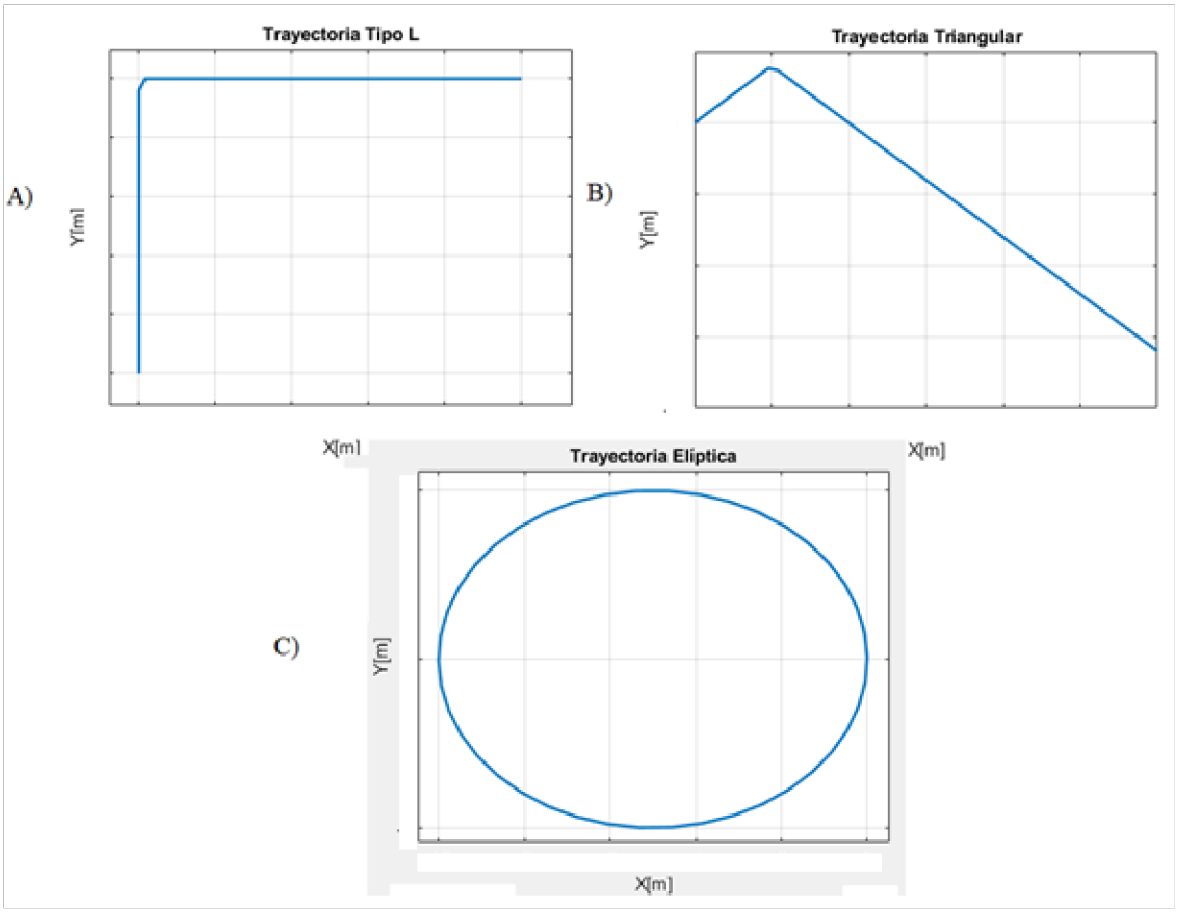


Figura 2.21. Trayectorias: A) Tipo L B) Triangular C) Elíptica

2.5 ESTIMACIÓN DE ÍNDICES DE DESEMPEÑO

Para evaluar y poder realizar la comparación de los controladores implementados para el seguimiento de trayectoria del Pioneer 3DX es necesario usar los índices de desempeño presentados en el subcapítulo 1.4.13 y específicamente los índices ISE (Integral Square Error) e ISCO (Integral Square Control). En la Figura 2.22 se muestra la codificación de estos 2 índices para poderlos implementar en el programa.

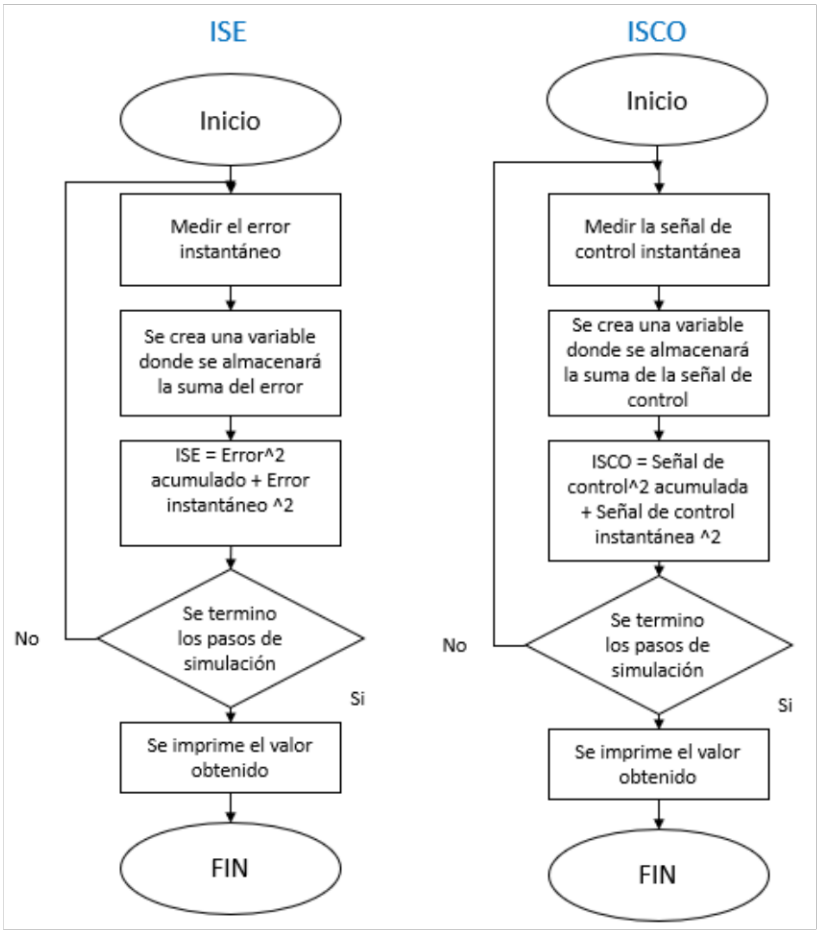


Figura 2.22. Diagrama de flujo de la implementación del ISE e ISCO

2.6 CONVERSIÓN DE VELOCIDADES PARA COPPELIASIM

La salida de los controladores implementados tiene la forma de velocidad lineal v y angular ω y la entrada de la planta en CoppeliaSim tiene la forma de velocidades angulares de la rueda derecha ω_d e izquierda ω_i, debido a esto es necesario realizar una conversión de $(v, \omega) \rightarrow (\omega_d, \omega_i)$.

Tomando las ecuaciones 1.2 y 1.3 del Tomo I se tiene que:

$$v = \frac{R}{L}(\omega_d + \omega_i)$$
$$\frac{Lv}{R} = (\omega_d + \omega_i) \quad (2.1)$$

Usando esta ecuación se despejará la velocidad angular de la rueda derecha y se obtiene

$$\omega_d = \frac{2v}{R} - \omega_i \tag{2.2}$$

Usando la ecuación 1.3 y la relación encontrada en la ecuación 2. 2

$$\omega = \frac{R}{L}(\omega_d - \omega_i)$$
$$\omega = \frac{R}{L}\left(\frac{2v}{R} - \omega_i - \omega_i\right) \tag{2.3}$$

Desarrollando se obtiene que la velocidad angular de la rueda izquierda es:

$$\omega_i = \frac{2v - L\omega}{2R} \tag{2.4}$$

Usando las ecuaciones 2.2 y 2.4 se obtiene que la velocidad angular de la rueda derecha es:

$$\omega_d = \frac{2v}{R} - \frac{2v - L\omega}{2R} \tag{2.5}$$

$$\omega_d = \frac{2v + L\omega}{2R} \tag{2.6}$$

2.7 ALGORITMOS DE CONTROL MATLAB - COPPELIASIM

Los algoritmos discretizados para la implementación dentro del software CoppeliaSim se encuentran en el subcapítulo 2.2 del TOMO I, usando esas expresiones, diagramas de flujo y tomando en cuenta las consideraciones planteadas en los subcapítulos 2.3, 2.4, 2.5 y 2.6 del presente documento se plantea realizar la codificación que se implementará dentro de la interfaz gráfica, el código de implementación de controladores, comunicación, generación de trayectorias y estimación de índices de desempeño se encuentra en el ANEXO V.

3 RESULTADOS CONCLUSIONES Y RECOMENDACIONES

Dentro de este capítulo se desarrollarán las pruebas de funcionamiento de los controladores propuestos para esto se simulará cada uno de ellos usando los distintos tipos de trayectoria planteadas, evaluando en cada uno los índices de desempeño ISE e ISCO para poder realizar una comparación posterior en base a los resultados, además se presentarán las señales de error, señal de control y seguimiento permitiendo observar cómo evoluciona cada una de estas señales. Los parámetros de simulación serán igual para todas las trayectorias usando un tiempo de 16 [s].

Dentro del capítulo se presentarán las simulaciones y datos que permitan realizar una comparación de eficiencia y rendimiento de cada controlador, para esto se pensó en 4 pruebas de funcionamiento las cuales son:

- Prueba I (Comparación modelos): En esta prueba se usarán los 3 modelos del Pioneer 3DX implementados en el subcapítulo 2.1.2 de este documento y el modelo del Pioneer 3DX de CoppeliaSim, a estos modelos se les ingresará una variación de velocidad lineal y angular para obtener respuestas que posteriormente se compararán para determinar cuál modelo de SIMULINK se asemeja más al de CoppeliaSim
- Prueba II (Seguimiento de Trayectoria en SIMULINK): En esta prueba se simulará el Pioneer 3DX en SIMULINK usando el modelo dinámico y los distintos controladores propuestos en conjunto con las trayectorias definidas y se verificará el error, señal de control, seguimiento e índices de desempeño.
- Prueba III (Sintonización de Parámetros en CoppeliaSim): En esta prueba se simulará el Pioneer 3DX en CoppeliaSim y se realizará una variación de parámetros en los controladores para verificar como varia el seguimiento en función de estos cambios y hallar las constantes que minimicen los índices de desempeño.
- Prueba IV (Seguimiento de Trayectoria en CoppeliaSim): En esta prueba se simulará el Pioneer 3DX en CoppeliaSim usando los distintos controladores propuestos en conjunto con las trayectorias definidas y se verificará el error, señal de control, seguimiento e índices de desempeño.

3.1 PRUEBA I: COMPARACIÓN DE MODELOS

Para esta prueba se usarán los 3 modelos implementados en el subcapítulo 2.1.2 y para cada uno de ellos se dará una entrada paso en $t = 0$ de velocidad lineal de amplitud $0{,}6\ m/s$ y una entrada paso en el segundo $t = 5$ de velocidad angular de $0{,}6\ rad/s$ manteniendo la magnitud de la velocidad lineal, esto permitirá obtener una gráfica que represente a cada modelo y se comparará con la gráfica que se obtiene al realizar los mismos cambios dentro del software CoppeliaSim, con la finalidad de verificar que modelo es el que más se asemeja a este último.

3.1.1 MODELO CINEMÁTICO TIPO UNICICLO SIMPLE

En la Figura 3.1 se encuentra la comparación de modelos de CoppeliaSim y el modelo cinemático de un robot móvil tipo uniciclo simple, se nota claramente la diferencia entre este tipo de modelos, debido a que dentro del software CoppeliaSim se realiza un análisis dinámico del elemento a simular.

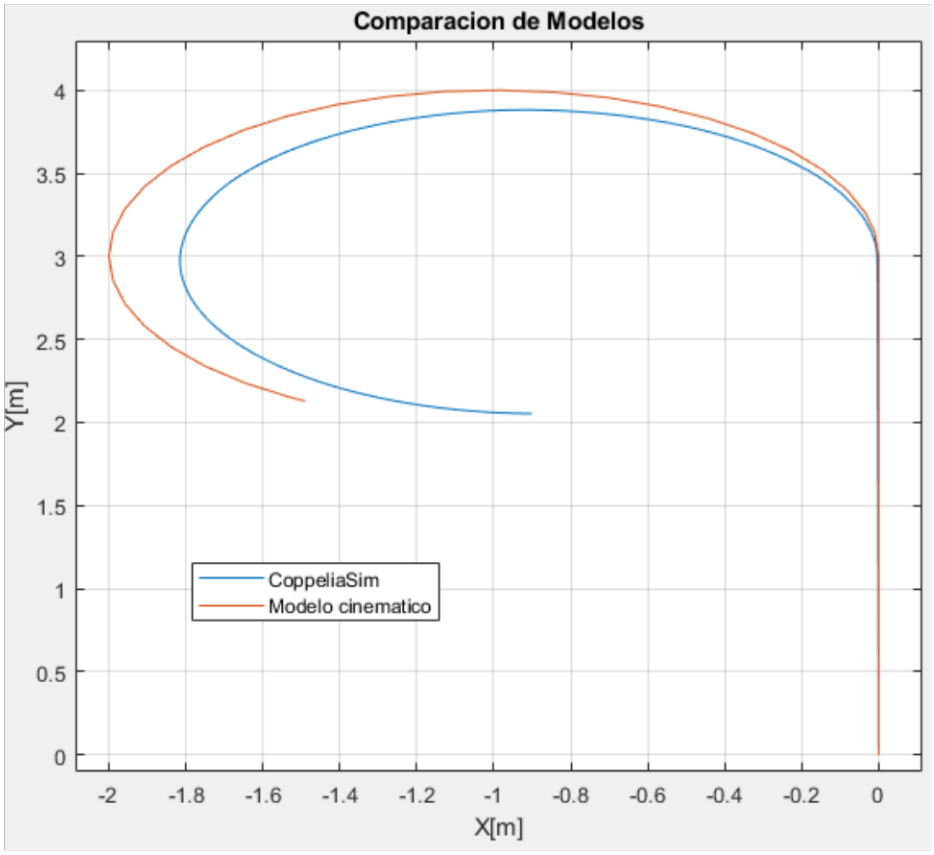


Figura 3.1. Salida del modelo cinemático en Simulink y el modelo de CoppeliaSim

3.1.2 MODELO CINEMÁTICO TIPO UNICICLO CON RESTRICCIÓN NO HOLONÓMICA MEJORADA

En la Figura 3.2 se encuentra la comparación de modelos de CoppeliaSim y el modelo cinemático de un robot móvil tipo uniciclo con restricción no holonómica mejorada, se nota que debido a la distancia a que se da debido a la restricción no holonómica la gráfica se desplaza un poco con respecto a la anterior.

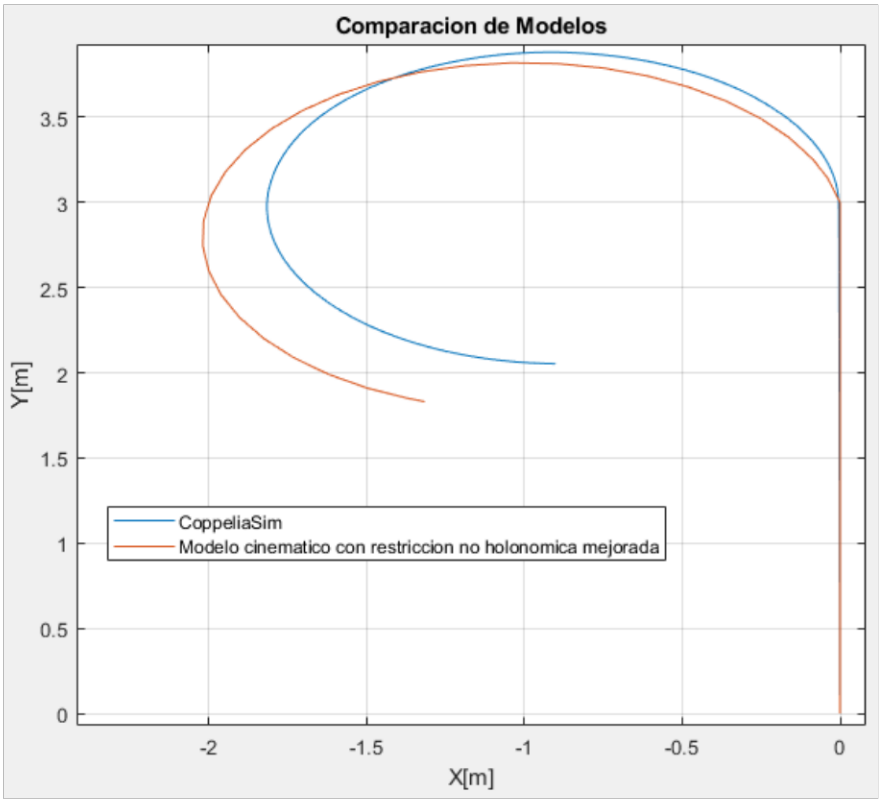


Figura 3.2. Salida del modelo con restricción no holonómica mejorada en Simulink y el modelo de CoppeliaSim

3.1.3 MODELO DINÁMICO

En la Figura 3.3 se encuentra la comparación de modelos de CoppeliaSim y el modelo dinámico del Pioneer 3DX, se nota que al analizar las características mecánicas, eléctricas y dinámicas en general de la planta robótica el modelo que se obtiene es mucho más parecido a la realidad y además el modelo dinámico es el que más se asemeja al modelo dentro de CoppeliaSim debido a que este software trata de analizar todos los aspectos necesarios para que las simulaciones sean un fiel reflejo de la realidad.

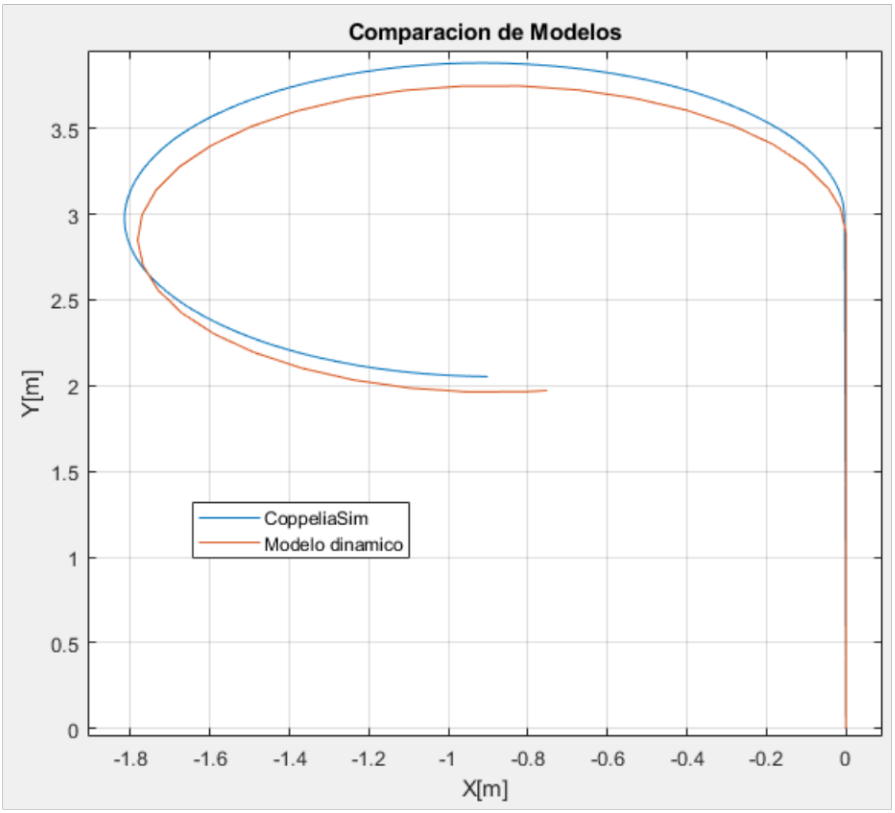


Figura 3.3. Salida del modelo dinámico en Simulink y el modelo de CoppeliaSim

3.2 PRUEBA II: SEGUIMIENTO DE TRAYECTORIAS EN SIMULINK

3.2.1 TRAYECTORIA ELÍPTICA:

Cuando la entrada de referencia es una trayectoria de forma elíptica se tiene los siguientes resultados para el seguimiento:

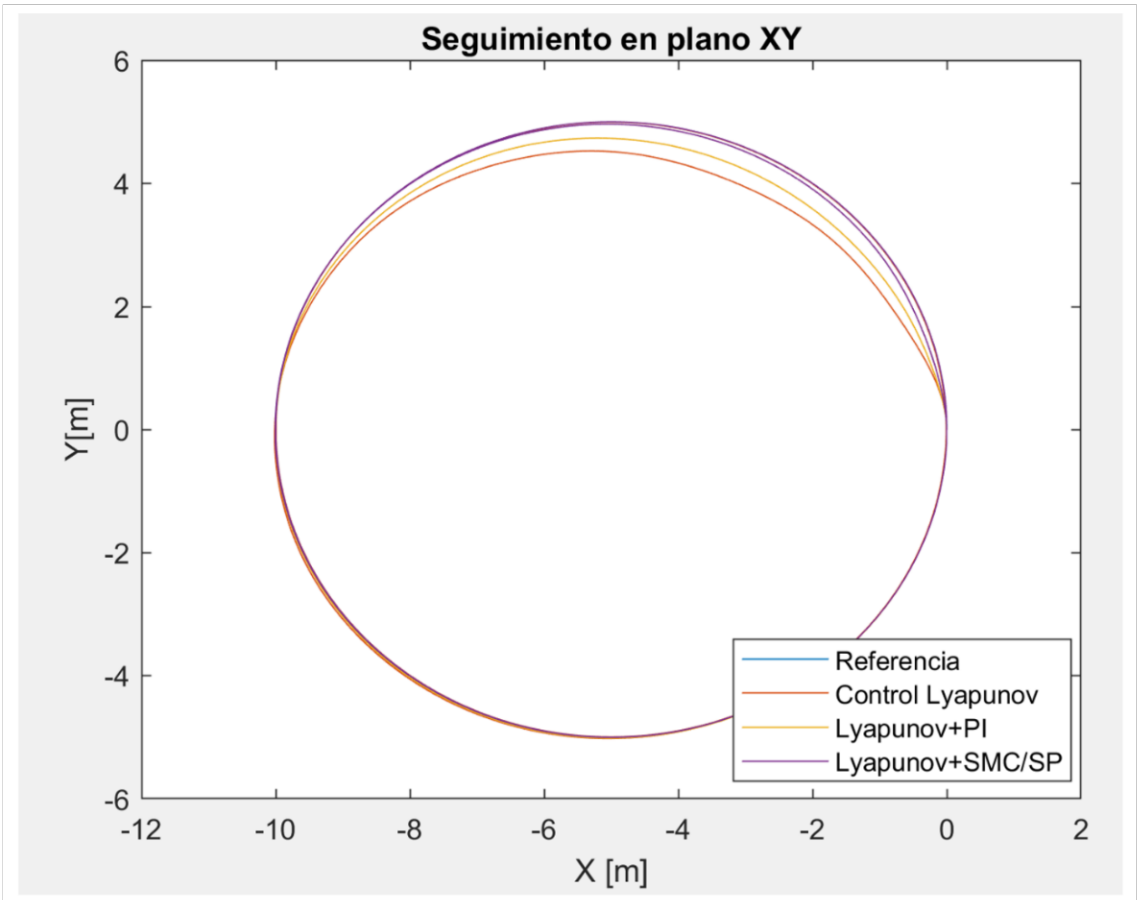
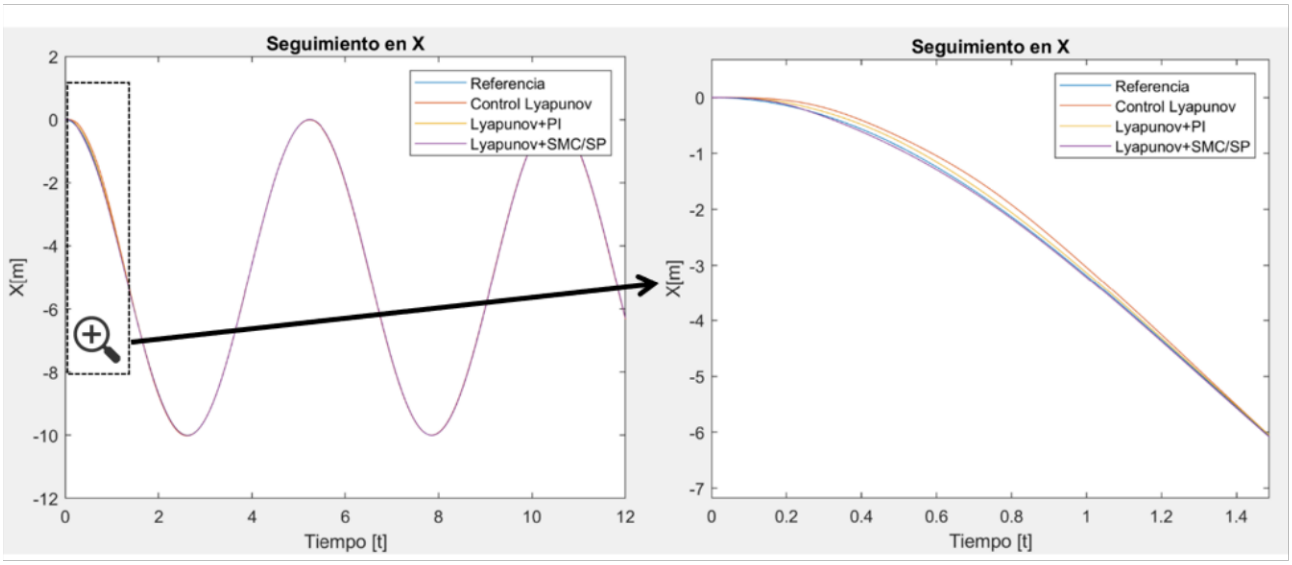


Figura 3.4. Seguimiento en el plano XY para la trayectoria elíptica

En la Figura 3.4 se puede notar como el controlador que mejores resultados presenta es el de Lyapunov + SMC/ PS ya que realiza un seguimiento más aproximado a la referencia, esto se puede apreciar de mejor forma en los seguimientos individuales de XY en las Figuras 3.5 y 3.6, esto permite ver que el control Lyapunov + SMC con PS realiza una corrección del error en menor tiempo tanto en X como en Y.



Se presenta el seguimiento de X en función del tiempo:

Figura 3.5. Seguimiento en X a lo largo del tiempo para la trayectoria elíptica

Con el seguimiento en Y en función del tiempo:

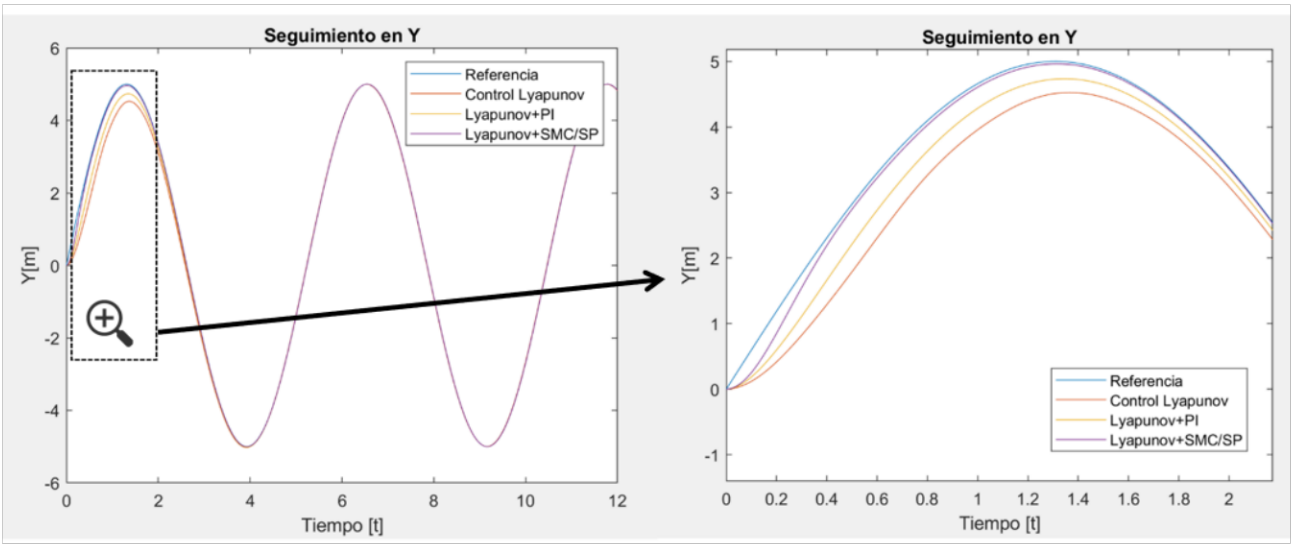


Figura 3.6. Seguimiento en Y a lo largo del tiempo para la trayectoria elíptica

Además, es necesario analizar la evolución del error en XY a lo largo del tiempo teniendo la salida que se presenta en la Figura 3.7 y 3.8 en los que se puede notar que los controladores en cascada permiten reducir el error en X hacia 0 pero el control en lazo simple reduce ese error, aunque le tomará más tiempo volver el error nulo, algo similar sucede con los errores en Y:

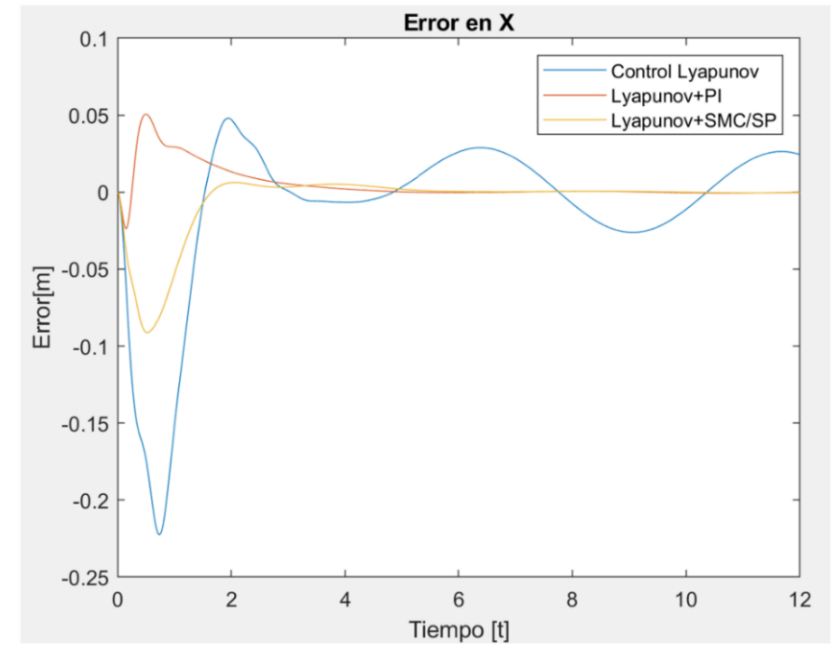


Figura 3.7. Error en X para la trayectoria elíptica

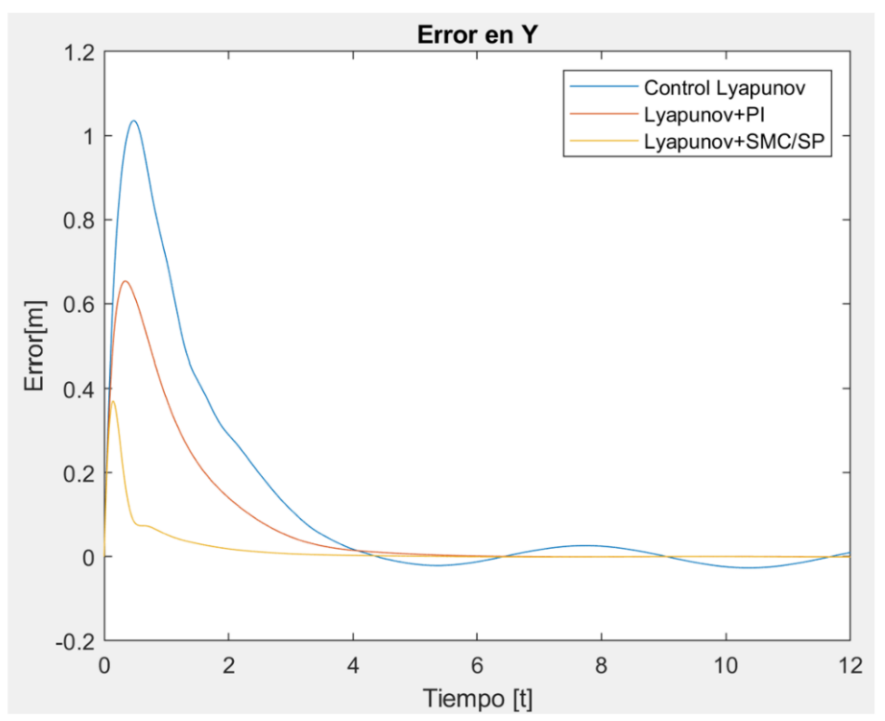


Figura 3.8. Error en Y para la trayectoria elíptica

Es importante analizar la forma de la acción de control, pues es la señal que se suministrará en los elementos finales de control, estas se presentan en las Figuras 3.9 y 3.10, a diferencia del error el controlador que presenta una señal de control de menor magnitud es

el de lazo simple y la señal de control que cargará más al elemento final de control es el control en cascada con Lyapunov + SMC/SP:

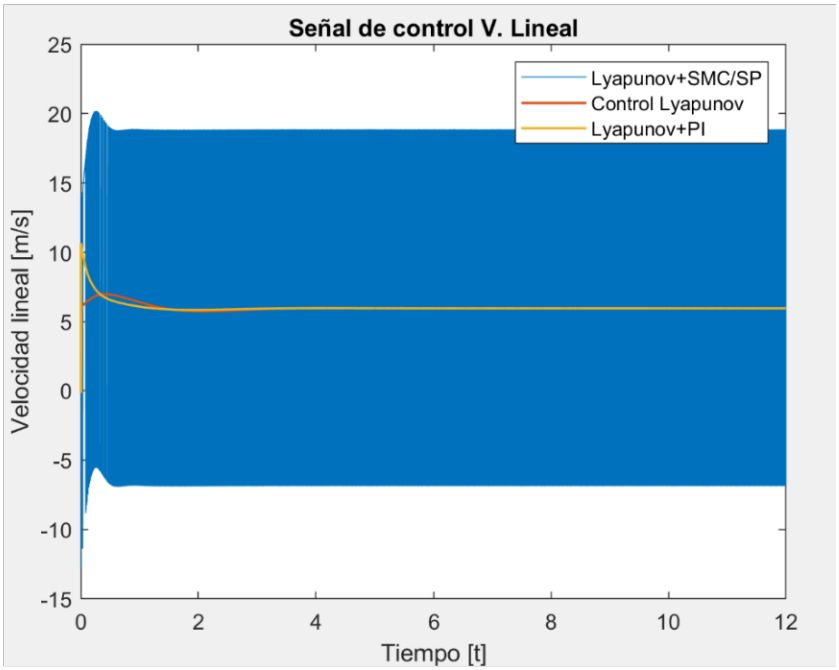


Figura 3.9. Acción de control de la velocidad lineal para la trayectoria elíptica

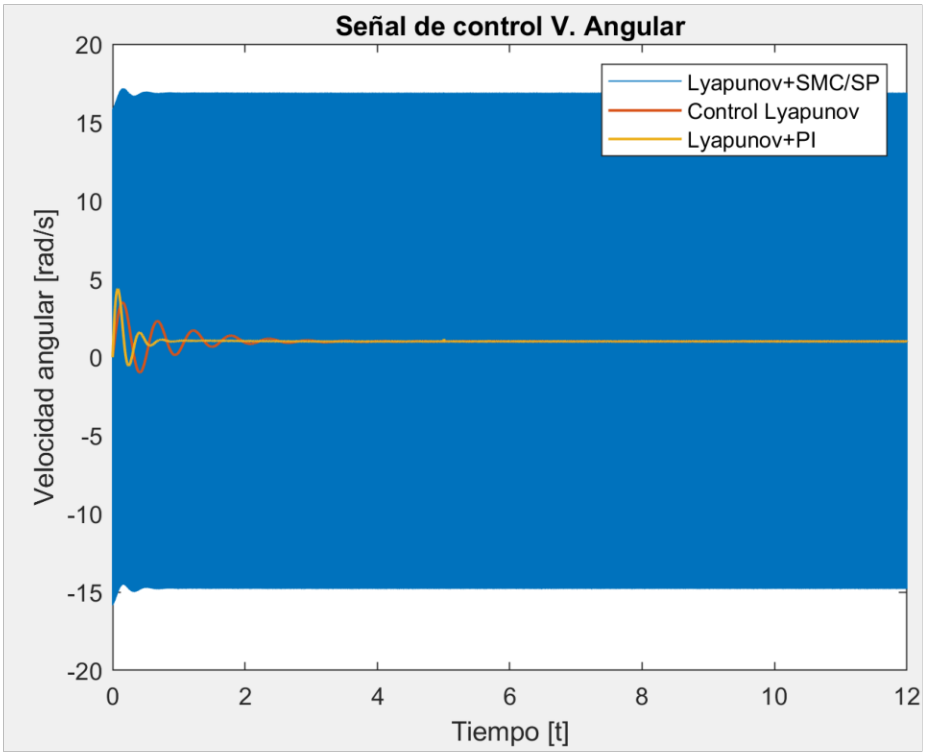


Figura 3.10. Acción de control de la velocidad angular para la trayectoria elíptica

3.2.2 TRAYECTORIA TRIANGULAR:

En la Figura 3.11 se puede notar como el controlador que mejores resultados presenta son los controladores en cascada tanto el PID como el SMC con PS, aunque este último presenta una ligera ventaja ya que realiza un seguimiento más aproximado a la referencia, esto se puede apreciar de mejor forma en los seguimientos individuales de XY en las Figuras 3.12 y 3.13, en los que se nota que ambos controles en cascada tienen un seguimiento bastante similar.

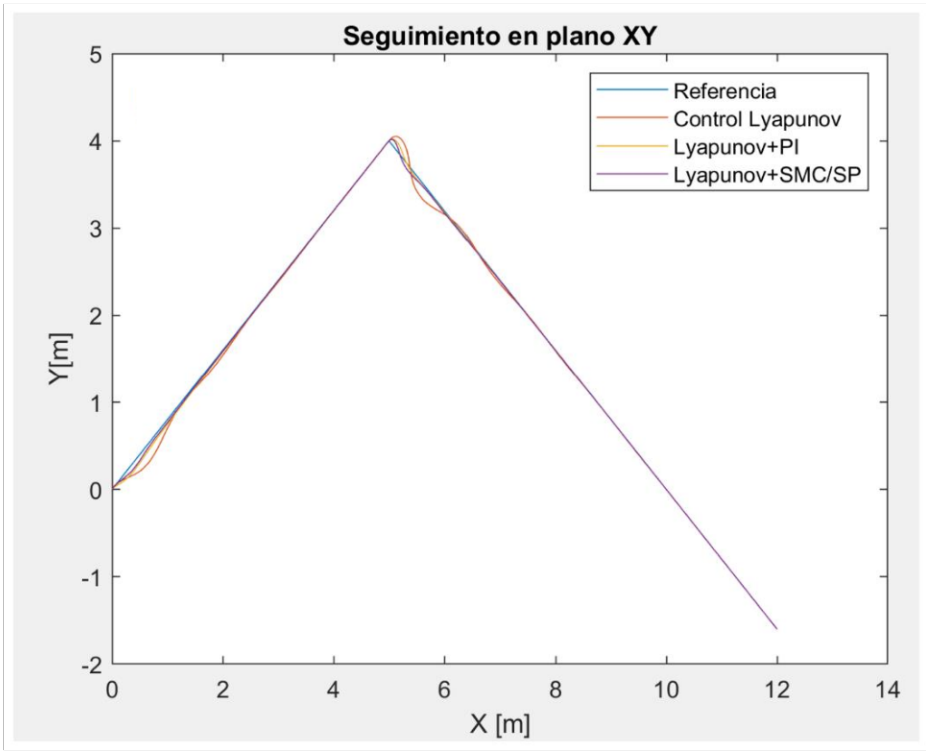


Figura 3.11. Seguimiento en el plano XY para la trayectoria triangular

Con el seguimiento en X a lo largo del tiempo:

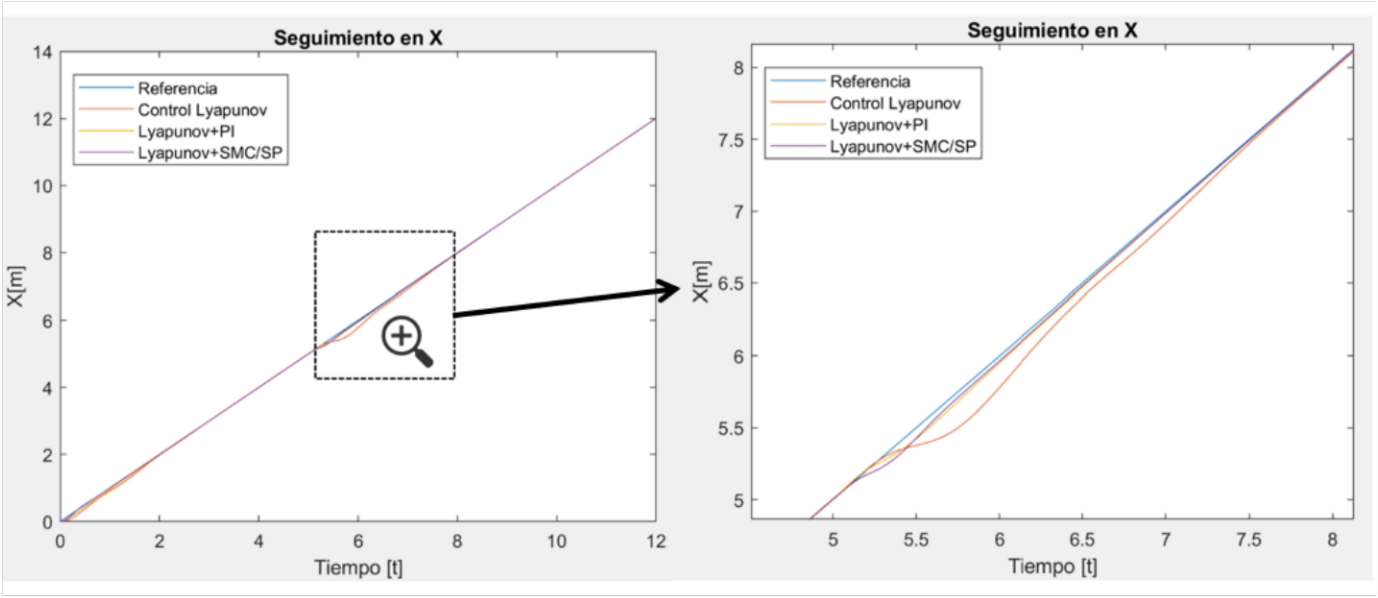


Figura 3.12. Seguimiento en X a lo largo del tiempo para la trayectoria triangular

Con el seguimiento en Y a lo largo del tiempo:

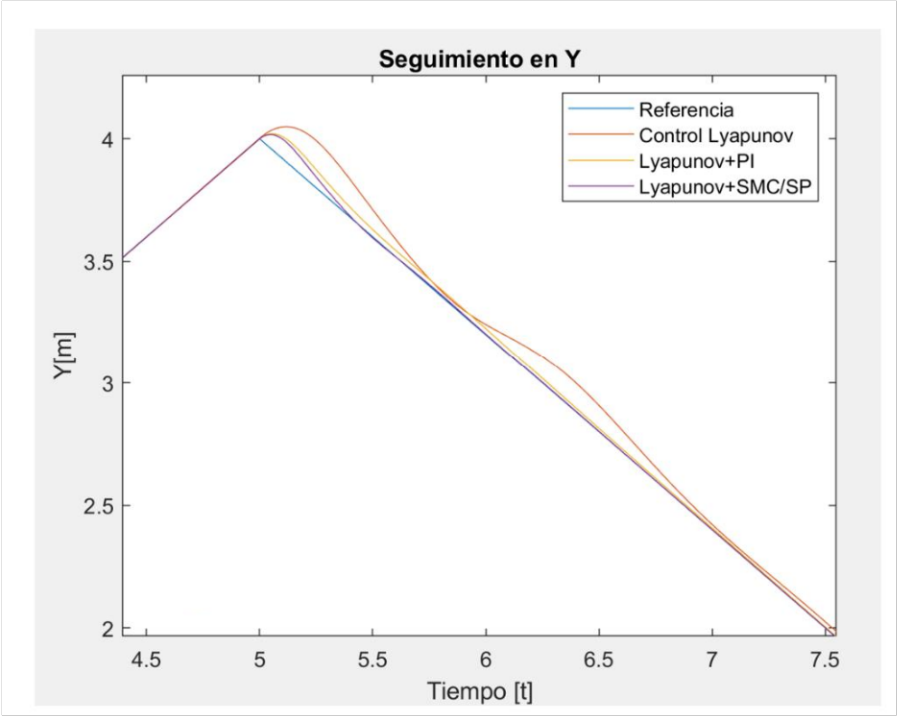


Figura 3.13. Seguimiento en Y a lo largo del tiempo para la trayectoria triangular

Los errores en XY presentes en la Figura 3.14 y 3.15 presentan lo antes mencionado que los controles en cascada presentan seguimientos similares y el control de lazo simple reduce el error a 0 aunque en un tiempo mayor.

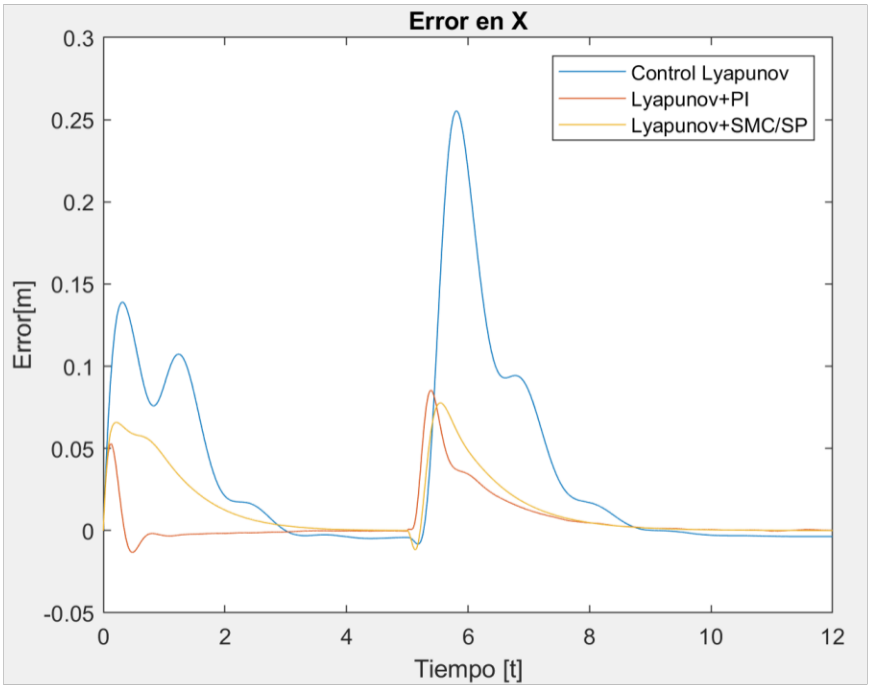


Figura 3.14. Error en X para la trayectoria triangular

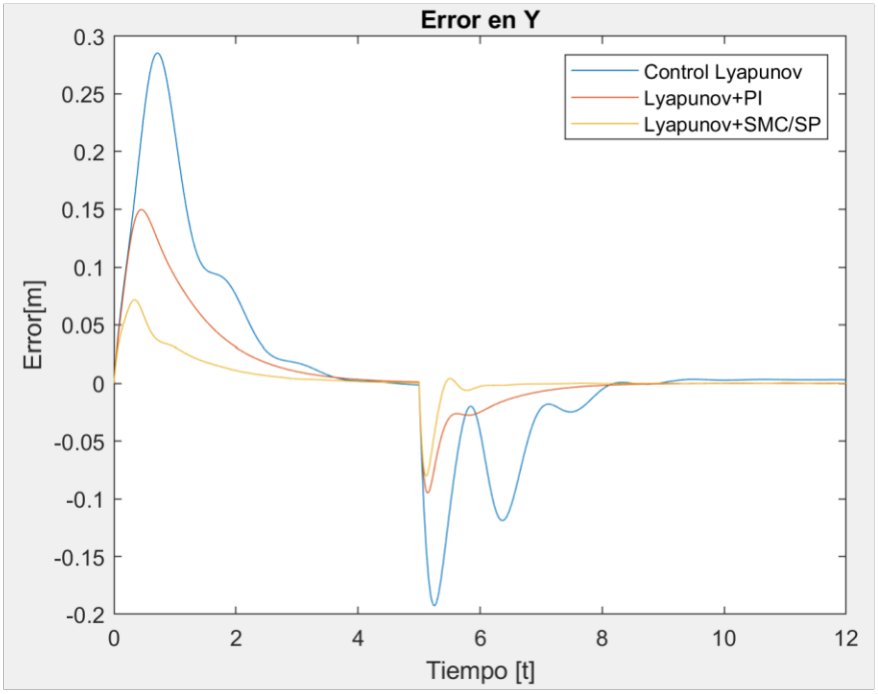


Figura 3.15. Error en Y para la trayectoria triangular

Las señales de control de la Figura 3.16 y 3.17 presentan resultados similares a los obtenidos con la trayectoria elíptica siendo el control de lazo simple el que tiene resultados de menor magnitud a comparación con el SMC con PS que presenta 'chattering'.

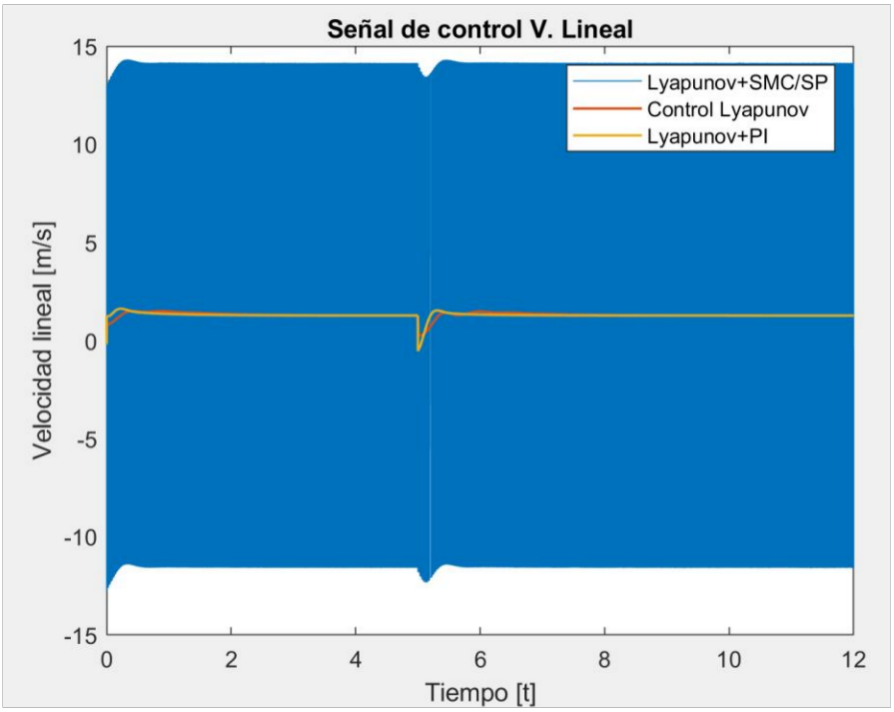


Figura 3.16. Acción de control de la velocidad lineal para la trayectoria triangular

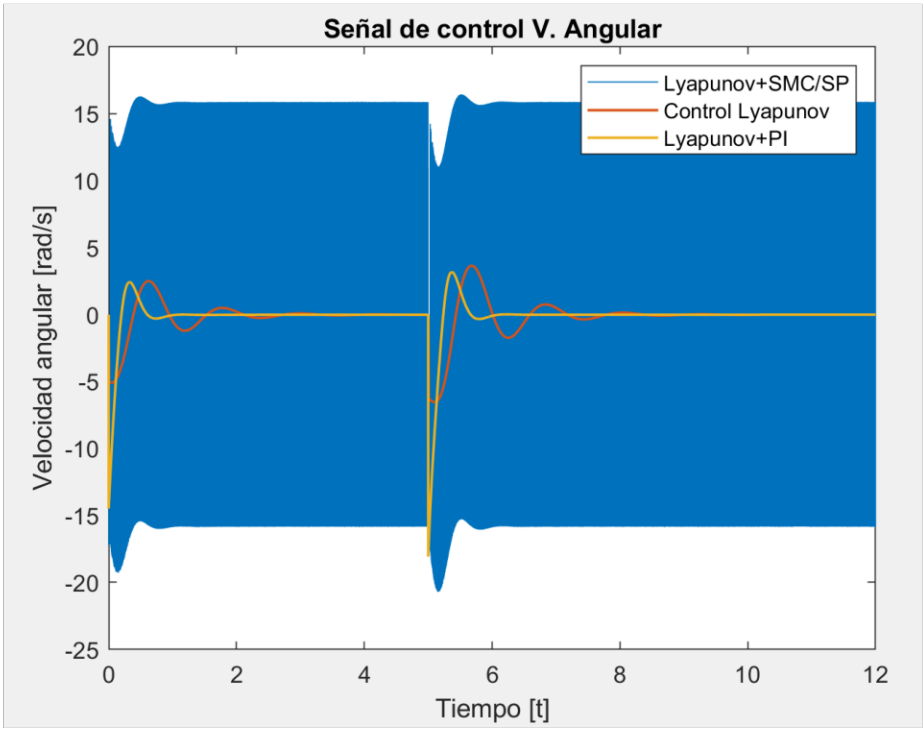


Figura 3.17. Acción de control de la velocidad angular para la trayectoria triangular

3.2.3 TRAYECTORIA TIPO L:

En la Figura 3.18 se puede notar como el controlador que mejores resultados es el control de Lyapunov + SMC con PS que lleva el error a cero de forma bastante rápida, esto se puede apreciar de mejor forma en los seguimientos individuales de XY en las Figuras 3.19 y 3.20.

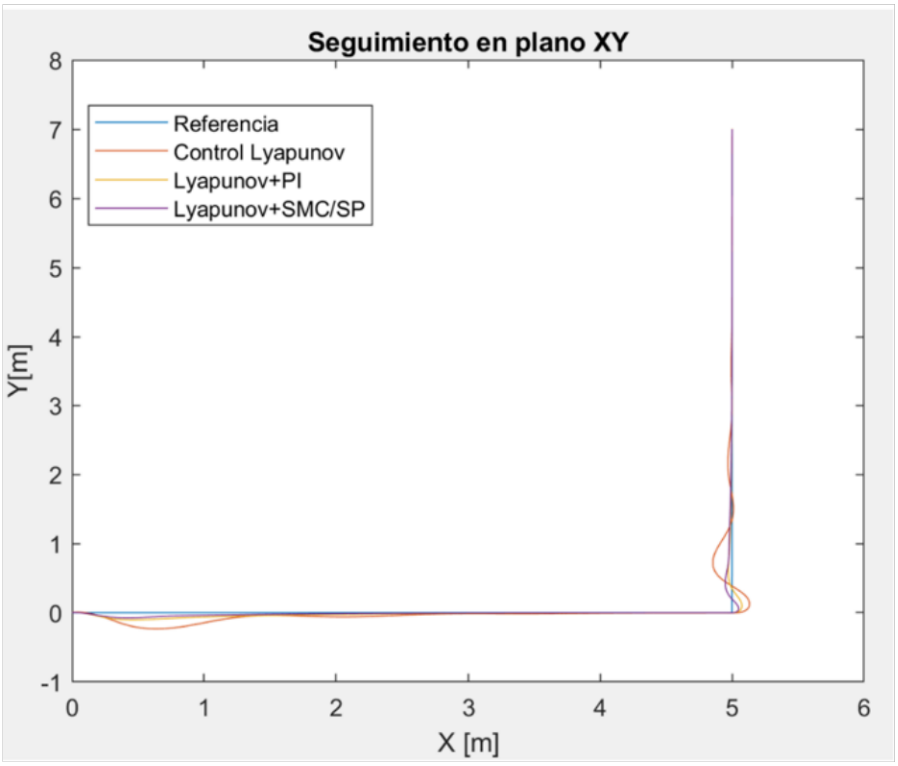


Figura 3.18. Seguimiento en el plano XY para la trayectoria tipo L

Con el seguimiento en X a lo largo del tiempo:

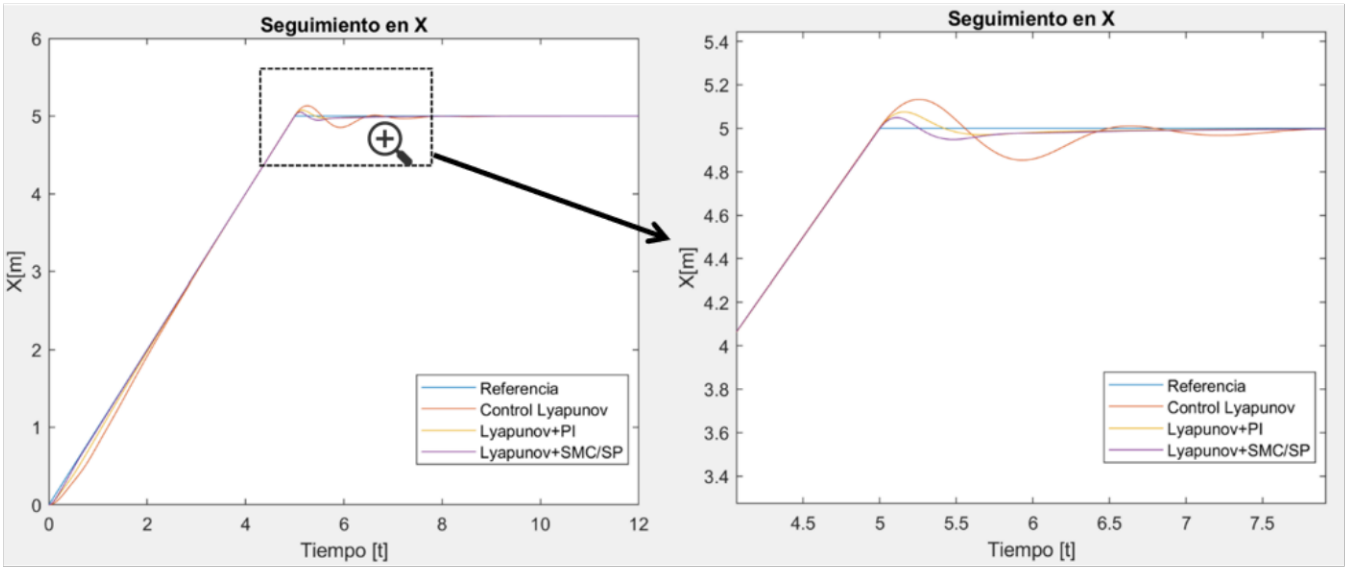


Figura 3.19. Seguimiento en X a lo largo del tiempo para la trayectoria tipo L

Con el seguimiento en Y a lo largo del tiempo:

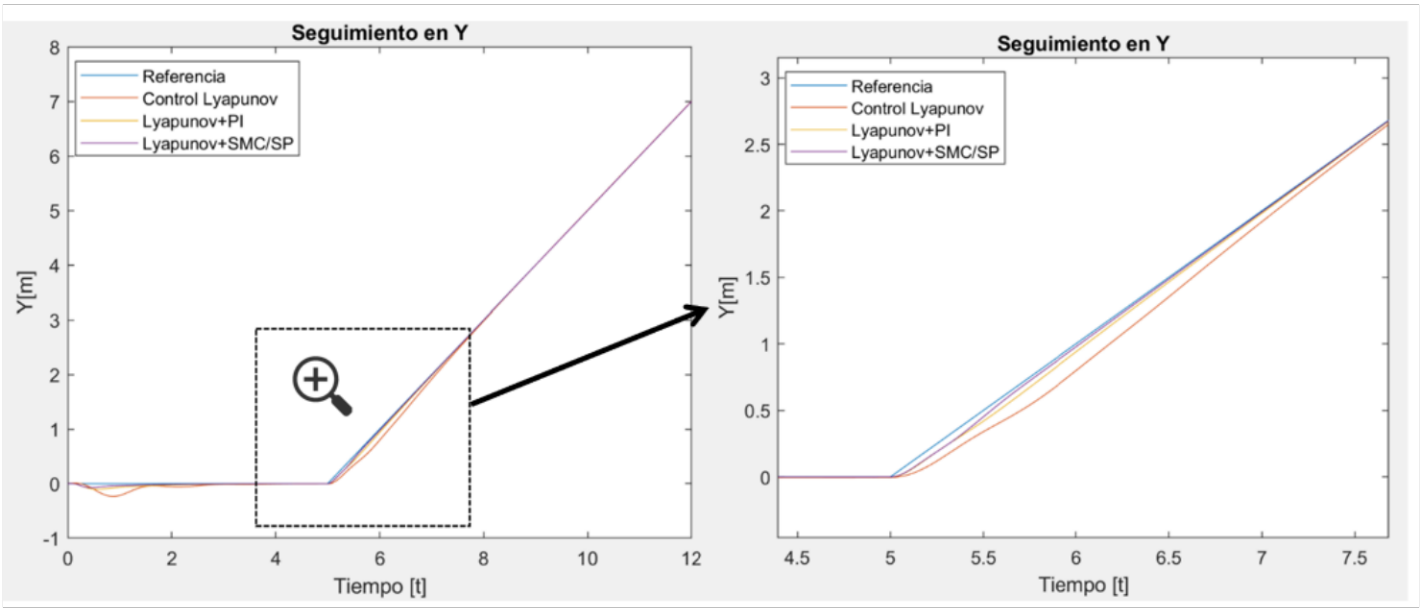


Figura 3.20. Seguimiento en Y a lo largo del tiempo para la trayectoria tipo L

En las Figuras 3.21 y 3.22 se tienen los errores que permiten ver la evolución de estos mostrando que el control que mejor reduce el error en esta trayectoria es el control en cascada con Lyapunov + PID al realizar el seguimiento en X mientras que el control Lyapunov + SMC con PS cuando se habla de seguimiento de Y.

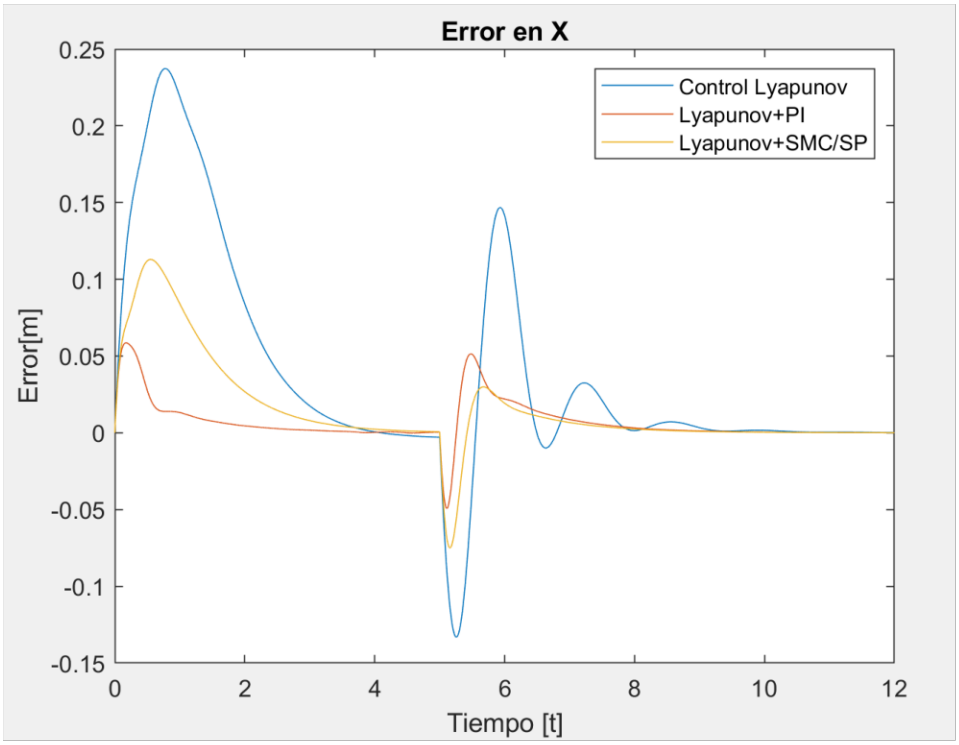


Figura 3.21. Error en X para la trayectoria tipo L

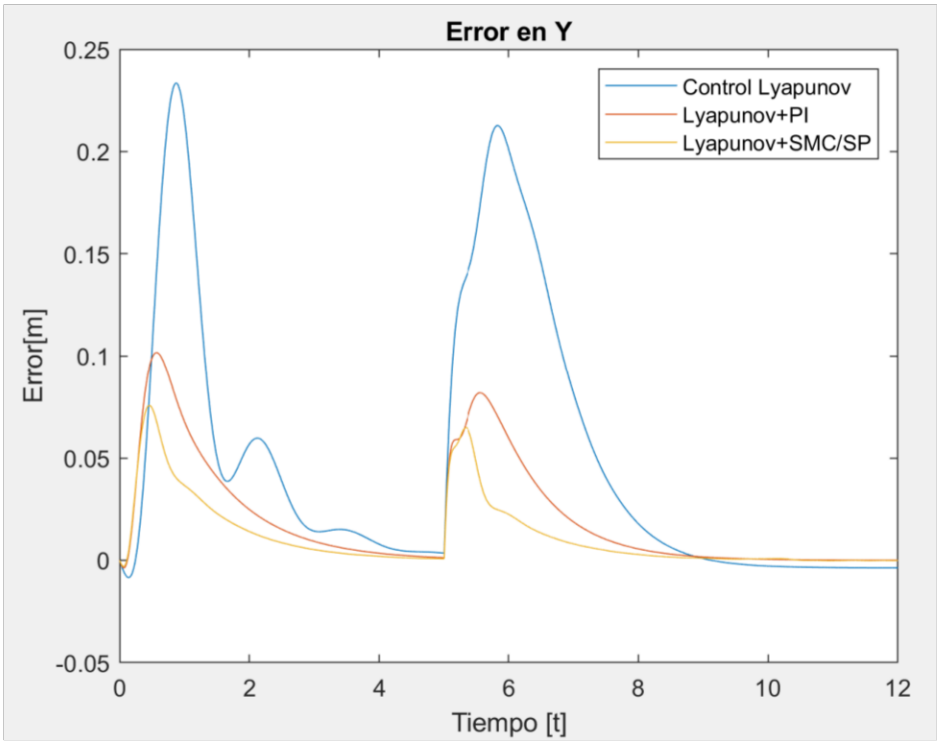


Figura 3.22. Error en Y para la trayectoria tipo L

Es importante analizar la forma de la acción de control de las Figuras 3.22 y 3.23, esto presenta resultados similares a los vistos en las otras trayectorias.

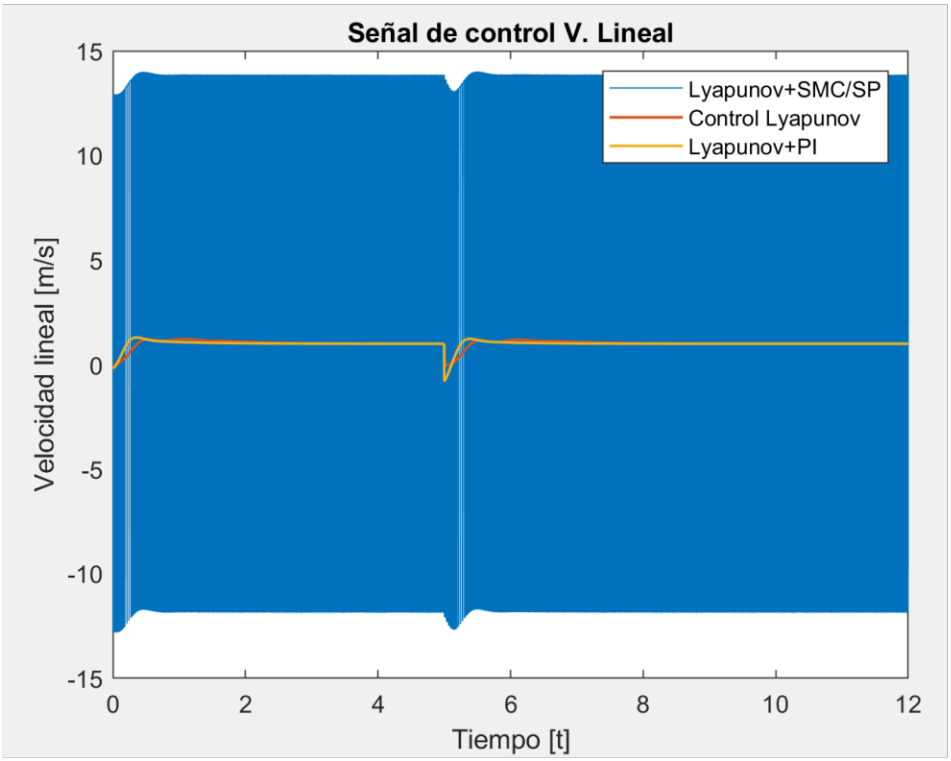


Figura 3.23. Acción de control de la velocidad lineal para la trayectoria tipo L

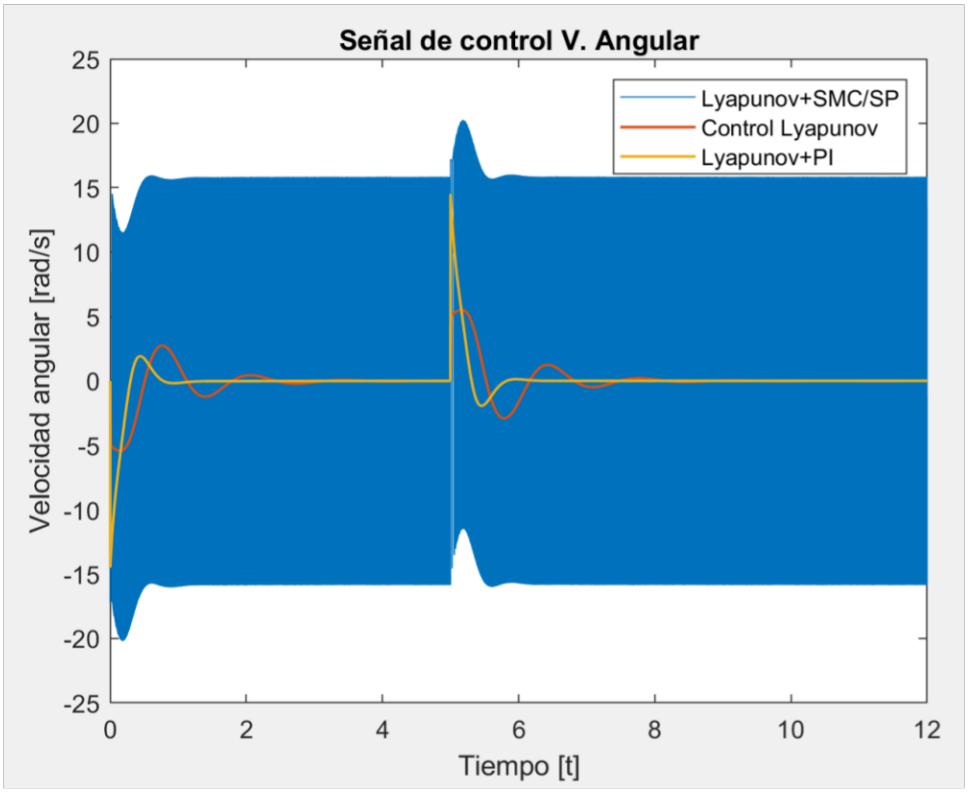


Figura 3.24. Acción de control de la velocidad angular para la trayectoria tipo L

Tabla 3.1. Índices de desempeño SIMULINK

Controlador	**Trayectoria Tipo L**		
	ISE	**ISCOu**	**ISCOw**
Lyapunov	40,70	1590	54704
Lyapunov + PID	15,44	1925	56803
Lyapunov + SMC & PS	2,24	8619	108680
Controlador	**Trayectoria Triangular**		
	ISE	**ISCOu**	**ISCOw**
Lyapunov	693,91	7429,6	20453,03
Lyapunov + PID	167,17	7713,2	39320,18
Lyapunov + SMC & PS	63,12	75061,01	112570,98
Controlador	**Trayectoria Elíptica**		
	ISE	**ISCOu**	**ISCOw**
Lyapunov	1019,4	5607	6422
Lyapunov + PID	222,58	5938	15981
Lyapunov + SMC & PS	97,00	117318	186760

Se presenta los cálculos del índice de desempeño en el ANEXO IV. Se puede notar lo anteriormente expuesto que el controlador que mejor corrigió el error es el Lyapunov +

SMC/PS y el que peor corrigió esto fue el control en lazo simple con Lyapunov, aunque esto se intercambia cuando se presenta la señal de control. Y el que presenta resultados más equilibrados es el control Lyapunov + PID.

3.3 PRUEBA III: SINTONIZACIÓN DE PARÁMETROS EN COPPELIASIM.

Dentro de cada controlador existen sus constantes de sintonización, la modificación en estos valores generará un cambio en el seguimiento, señal de error y señal de control. Debido a esto se vio la necesidad evaluar los distintos controladores propuestos y realizar una comparación de sus índices de desempeño, verificando cuál es la incidencia del cambio de las constantes en sus valores finales. Para realizar las pruebas se tomó un tiempo de 10 [s] para reducir tiempo de simulación y permitir visualizar el comportamiento del controlador.

3.3.1 CONTROL EN LAZO SIMPLE (LYAPUNOV)

En esta sección se presentarán los índices de desempeño del controlador de lazo simple usando un controlador de Lyapunov, la constante K será la que modifique tanto la corrección del error como la señal de control, se evaluarán distintos valores de K para las 3 trayectorias propuestas y se observará como incide la variación en cada índice.

Tabla 3.2. Incidencia de constantes en índices de desempeño del Controlador en lazo simple Lyapunov – Trayectoria Tipo L

	Controlador en lazo simple LYAPUNOV					
	Trayectoria: Tipo L **Tiempo de Simulación:** 16 [s]					
K	**1**	**2**	**2.8**	**3**	**3,2**	**4**
ISEx	41.25	25.21	13.33	17.05	12.97	16.39
ISEy	16.09	8.66	17.17	9.375	7.747	8.174
ISCOu	881.3	866.2	820.01	873.6	892.7	897.3
ISCOw	465.2	521.3	583.2	602.8	633.36	665.5

Tabla 3.3. Incidencia de constantes en índices de desempeño Controlador en lazo simple Lyapunov – Trayectoria Triangular

	Controlador en lazo simple LYAPUNOV					
	Trayectoria: Triangular **Tiempo de Simulación:** 16 [s]					
K	**1**	**2**	**2.8**	**3**	**3.2**	**4**
ISEx	38.13	24.37	19.67	21.93	11.49	21.49
ISEy	25.77	18.39	15.29	19.9	10.51	18.51
ISCOu	840.3	860.43	886.5	902.1	917.9	925.9
ISCOw	270.55	284.11	299.98	310.8	374.7	374.7

Tabla 3.4. Incidencia de constantes en índices de desempeño Controlador en lazo simple Lyapunov – Trayectoria Elíptica

	Controlador en lazo simple LYAPUNOV					
	Trayectoria: Elíptica **Tiempo de Simulación:** 16 [s]					
K	**1**	**2**	**2.8**	**3**	**3.2**	**4**
ISEx	20.21	8.048	0.5318	0.7893	0.573	1.143
ISEy	18.75	8.826	3.284	2.736	2.604	2.451
ISCOu	881.99	899.2	911.02	929.13	956.91	983.5
ISCOw	140.54	154.23	174.8	187.72	193.4	202.5

Para las tablas 3.2, 3.3 y 3.4 se busca los índices que reduzcan el error de seguimiento lo máximo posible teniendo en cuenta que las señales de control no alcancen valores demasiado altos que puedan dañar los elementos finales de control, si es posible que los índices ISE e ISCO sean mínimos en ambos casos se elegirá esos valores para la sintonización.

3.3.2 CONTROL EN CASCADA: LAZO EXTERNO (LYAPUNOV) Y LAZO INTERNO (PID)

En esta sección se presentarán los índices de desempeño del controlador con esquema en cascada usando un controlador externo de Lyapunov y un control interno tipo PID, las constantes K, kp y ki será la que modifique tanto la corrección del error como la señal de control, se evaluarán distintos valores de K, kp y ki para las 3 trayectorias propuestas y se observará como incide la variación en cada índice.

Tabla 3.5. Incidencia de constantes en índices de desempeño Controlador Lyapunov + PID – Trayectoria Tipo L

	Controlador en cascada LYAPUNOV + PID					
	Trayectoria: Tipo L			**Tiempo de Simulación:** 16 [s]		
K	**2.9**	**3**	**3.3**	**3.3**	**3.3**	**3.3**
Kpu	3	3	3	2	2.5	3
Kiu	2.7	2.7	1	2.7	2	2.7
Kpw	12	12	12	9	8	12
Kiw	5	5	2	5	3	5
ISEx	5.716	5.212	0.8552	3.612	3.685	4.506
ISEy	4.321	3.626	7.213	5.12	5.23	3.21
ISCOu	890.7	892,3	870.1	873.6	892.7	897.3
ISCOw	497.3	542.2	4400	2340	1996	665.5

Tabla 3.6. Incidencia de constantes en índices de desempeño Controlador Lyapunov + PID – Trayectoria Triangular

	Controlador en cascada LYAPUNOV + PID					
	Trayectoria: Triangular			**Tiempo de Simulación:** 16 [s]		
K	**1.9**	**1.9**	**2.25**	**2.25**	**2.25**	**3.3**
kpu	1	3.5	1	3.5	3.5	1
Kiu	1	3.2	1	3.2	3.2	1
kpw	1	1	1	1	18	1
kiw	1	1	1	1	12	1
ISEx	42.3	12.787	23.6	5.877	4.51	4.406
ISEy	35.65	16.23	19.28	11.22	2.608	7.097
ISCOu	901.2	1023	929.6	1173	916.4	952.9
ISCOw	307.7	1578	355.9	1847	1,09E+04	574.7

Tabla 3.7. Incidencia de constantes en índices de desempeño Controlador Lyapunov + PID – Trayectoria Elíptica

	Controlador en cascada LYAPUNOV + PID					
	Trayectoria: Elíptica			**Tiempo de Simulación:** 16 [s]		
K	**1.9**	**1.9**	**2.25**	**2.25**	**3.3**	**3.3**
Kpu	1	3.5	1	3.5	3.5	1
Kiu	1	3.2	1	3.2	3.2	1
Kpw	1	3	3	3	18	3
Kiw	1	1	1	1	6	1
ISEx	18.3	3.24	13.63	0.482	12.86	0.3047
ISEy	19.21	4.51	16.27	2.45	9.069	2.278

ISCOu	904.23	1049.2	921.6	1133	896.9	933.5
ISCOw	168.3	54.23	173.1	57.72	6205	179.4

Para las tablas 3.5, 3.6 y 3.7 se busca los índices que reduzcan el error de seguimiento lo máximo posible teniendo en cuenta que las señales de control no alcancen valores demasiado altos que puedan dañar los elementos finales de control, si es posible que los índices ISE e ISCO sean mínimos en ambos casos se elegirá esos valores para la sintonización.

3.3.3 CONTROL EN CASCADA: LAZO EXTERNO (LYAPUNOV) Y LAZO INTERNO (SMC + PREDICTOR DE SMITH)

En esta sección se presentarán los índices de desempeño del controlador con esquema en cascada usando un controlador externo de Lyapunov y un control interno tipo SMC con un esquema de Predictor de Smith, las constantes $K, K_D \; y \; \delta$ será la que modifique tanto la corrección del error como la señal de control, se evaluarán distintos valores de $K, K_D \; y \; \delta$ para las 3 trayectorias propuestas y se observará como incide la variación en cada índice.

Tabla 3.8. Incidencia de constantes en índices de desempeño Controlador Lyapunov + SMC con Predictor de Smith – Trayectoria Tipo L

	Controlador en cascada LYAPUNOV + SMC con Predictor de Smith					
	Trayectoria: Tipo L			**Tiempo de Simulación:** 16 [s]		
K	**1.9**	**2.25**	**2.25**	**2.25**	**2.25**	**2.25**
KDu	0.16	0.16	0.11	0.11	0.11	0.1
δu	1	1	1	1	2	1
KDw	1.56	1.56	1.56	1.38	1.38	1.2
δw	1	1	1	1	2	1
ISEx	27.4	16.88	6.82	3.37	4.78	7.31
ISEy	9.65	4.56	6.98	3.67	4.12	7.83
ISCOu	769.1	764	782.6	787.1	820.7	800
ISCOw	1395	1469	1715	1243	621.9	1057

Tabla 3.9. Incidencia de constantes en índices de desempeño Controlador Lyapunov + SMC con Predictor de Smith – Trayectoria Triangular

	Controlador en cascada LYAPUNOV + SMC con Predictor de Smith					
	Trayectoria: Triangular **Tiempo de Simulación:** 16 [s]					
K	**1.9**	**2.2**	**2.25**	**2.25**	**2.25**	**2.7**
KDu	0.16	0.11	0.16	0.11	0.11	0.16
δu	1	1	1	1	1	1
KDw	1.56	1.38	1.56	1.56	1.38	1.56
δw	1	1	1	1	1	1
ISEx	16.88	12.22	5.32	7.25	7.414	3.79
ISEy	13.54	11.06	4.25	6.96	6.45	3.81
ISCOu	753.6	776.2	762.1	782.7	795.6	770.9
ISCOw	1418	1517	1539	1508	1304	1655

Tabla 3.10. Incidencia de constantes en índices de desempeño Controlador Lyapunov + SMC con Predictor de Smith – Trayectoria Elíptica

	Controlador en cascada LYAPUNOV + SMC con Predictor de Smith					
	Trayectoria: Elíptica **Tiempo de Simulación:** 16 [s]					
K	**1.9**	**2.25**	**3**	**3**	**3.3**	**3.5**
KDu	0.16	0.16	0.11	0.16	0.16	0.16
δu	1	1	1	1	1	1
KDw	1.56	1.56	1.38	1.56	1.56	1.56
δw	1	1	1	1	1	1
ISEx	18.11	4.121	1.73	1.23	0.573	1.702
ISEy	29.34	12.61	7.412	4.23	0.892	6.701
ISCOu	781.4	796.9	830	780.2	808.5	789.4
ISCOw	1079	1070	844.3	2.736	1071	1068

Para las tablas 3.8, 3.9 y 3.10 se busca los índices que reduzcan el error de seguimiento lo máximo posible teniendo en cuenta que las señales de control no alcancen valores demasiado altos que puedan dañar los elementos finales de control, si es posible que los índices ISE e ISCO sean mínimos en ambos casos se elegirá esos valores para la sintonización.

3.4 PRUEBA IV: SEGUIMIENTO DE TRAYECTORIAS EN COPPELIASIM

3.4.1 Trayectoria Tipo L

En la Figura 3.25 se presenta el seguimiento de la trayectoria tipo L de los 3 tipos de controladores dentro del software CoppeliaSim.

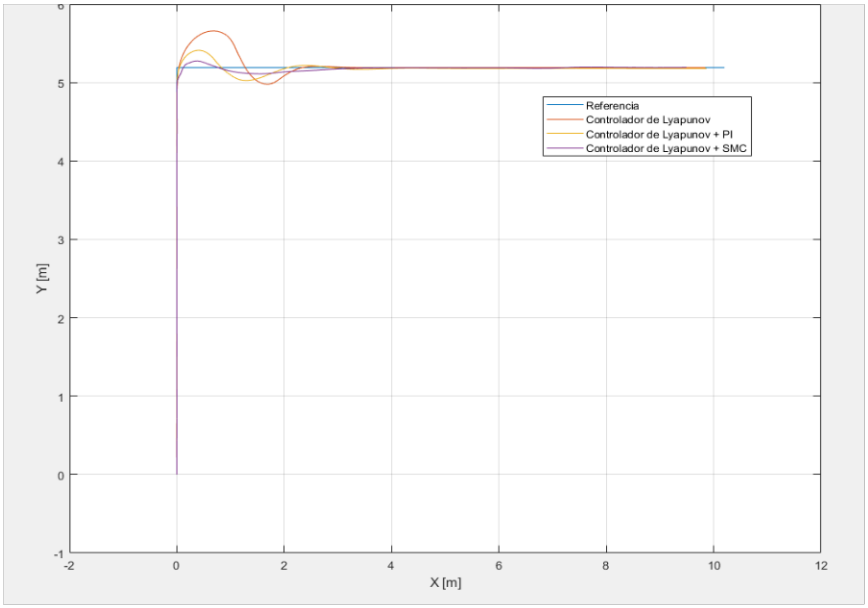


Figura 3.25. Seguimiento de trayectoria tipo L

Usando las señales de error se puede tener una idea de que tan bueno es el rendimiento del controlador, lo que se busca es que el error converja a 0 cuando $t \rightarrow \infty$ además que el tiempo en el que el error llegue a ese valor estable debe ser pequeño.

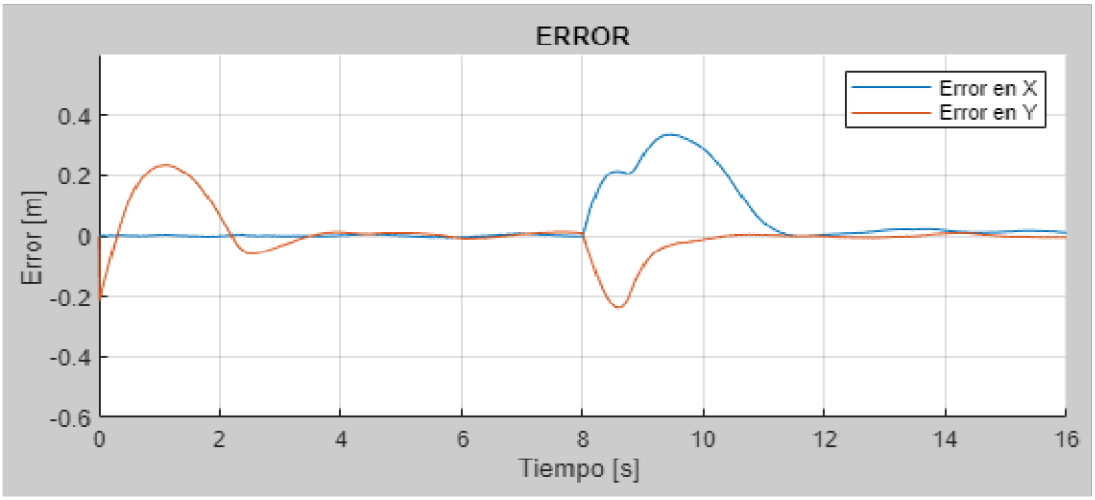


Figura 3.26. Errores Control lazo simple Lyapunov usando una trayectoria tipo L

En la Figura 3.26 se tiene la gráfica de los errores de posición usando un control de lazo simple Lyapunov, en esta gráfica se nota como el cambio de 90 grados de la trayectoria tipo L se da en los 8 segundos, el control simple cumple con la necesidad de llevar el error al valor de 0 dentro del rango de tiempo establecido.

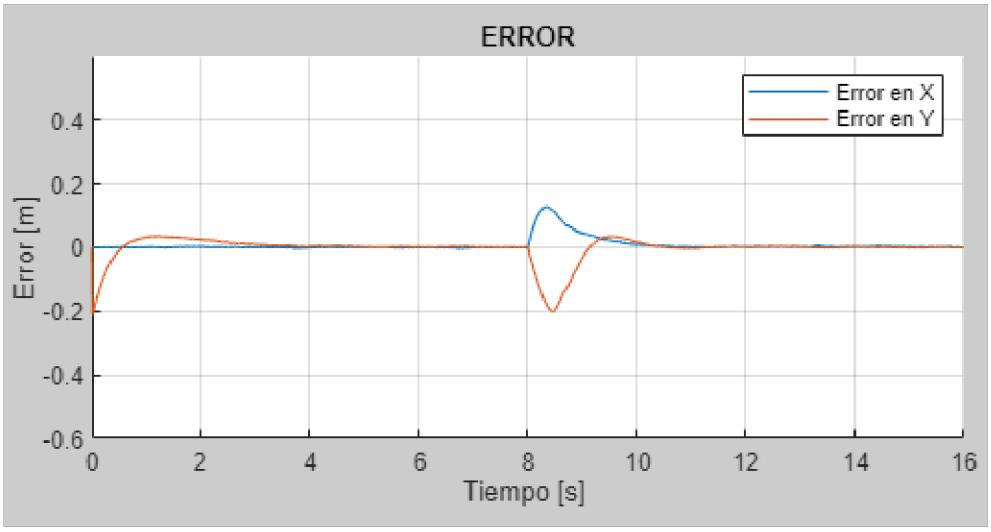


Figura 3.27. Errores Control cascada Lyapunov + PID usando una trayectoria tipo L.

En la Figura 3.27 se tiene la gráfica de los errores de posición, en esta gráfica se nota el cambio a los 8 segundos, el control en cascada Lyapunov + PID hace converger el error a 0 y reduce el tiempo de corrección del error en X, aunque en Y el tiempo que le toma al controlador a llegar a la referencia presenta un aumento con respecto a los errores anteriores.

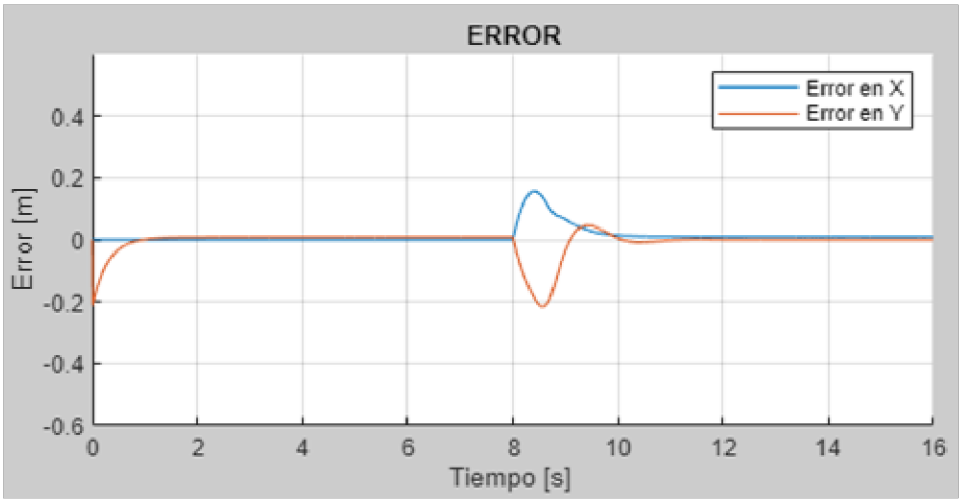


Figura 3.28. Errores Control cascada Lyapunov + SMC con PS usando una trayectoria tipo L

En la Figura 3.28 se tiene la gráfica de los errores de posición, en esta gráfica se nota el cambio a los 8 segundos, el control en cascada con Lyapunov + SMC & PS reduce el error en X de forma suave en un tiempo de aproximadamente 3 [s] y el error de Y en 4 [s].

Aunque el error pueda ayudar a notar que tan bueno es un controlador es necesario analizar otro punto en conjunto con este, la señal de control.

La señal de control es otro indicador de la calidad de un controlador ya que dentro del lazo de control generalmente se tiene un actuador que permite convertir la señal de control en una acción para regular la variable que se desea controlar y generalmente tiene límites de operación, debido a esto las señales de control con valores que superen esos valores límites o que tengan oscilaciones pronunciadas pueden llegar a afectar el actuador, debido a esto es necesario que la señal de control se mantenga en valores que el actuador pueda soportar.

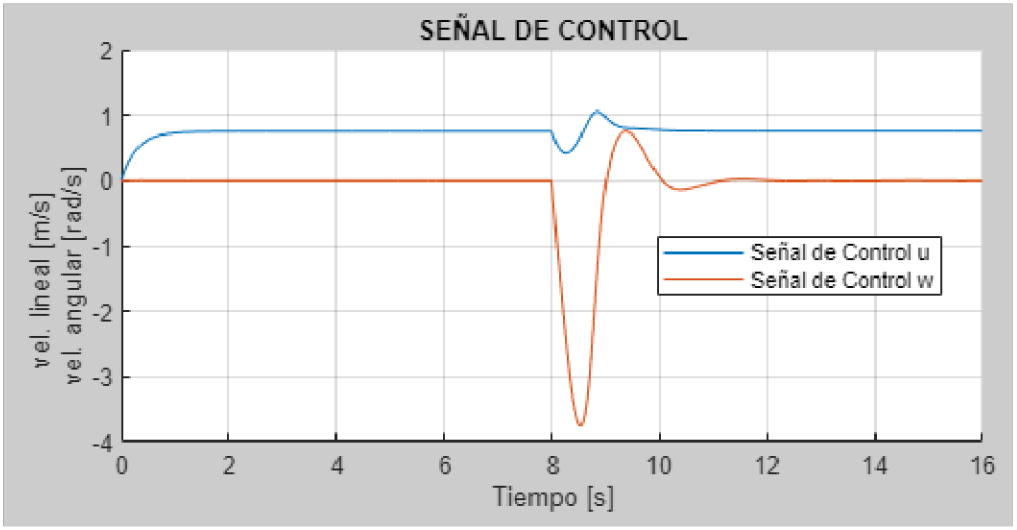


Figura 3.29. Señales de Control lazo simple Lyapunov usando una trayectoria tipo L

En la Figura 3.29 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control de lazo simple Lyapunov, en esta gráfica se nota que la señal de control de la velocidad lineal se mantiene en un aproximado de $0{,}8$ m/s y llega a valor máximo de 1 m/s, aunque la señal de control de la velocidad angular si alcanza valores relativamente de hasta -3 rad/s lo que puede llegar a dañar los motores del Pioneer 3DX.

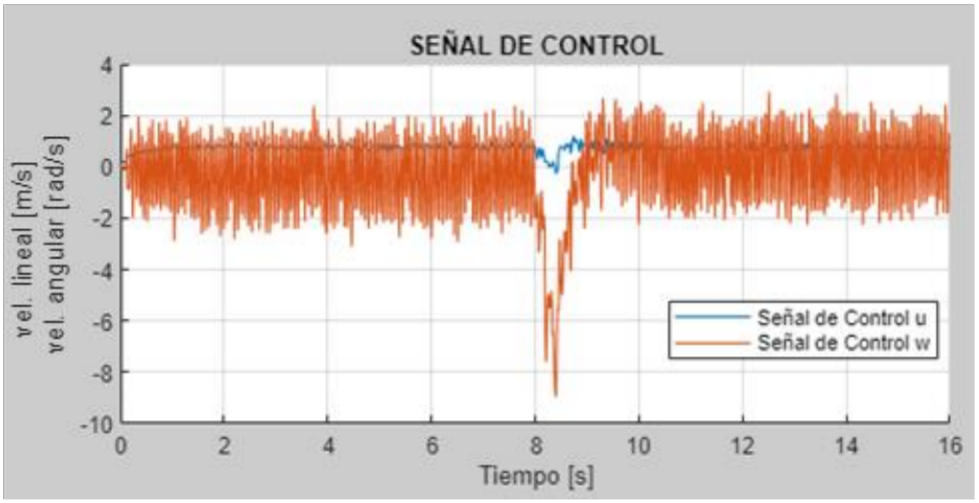


Figura 3.30. Señales de Control cascada Lyapunov + PID usando una trayectoria tipo L.

En la Figura 3.30 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada Lyapunov + PID, en esta gráfica se nota que la señal de control de la velocidad lineal se mantiene en valores similares de entre 0.8 y 1 $\mathrm{m/s}$, aunque la velocidad angular presenta picos que llegan hasta los -8 $\mathrm{rad/s}$.

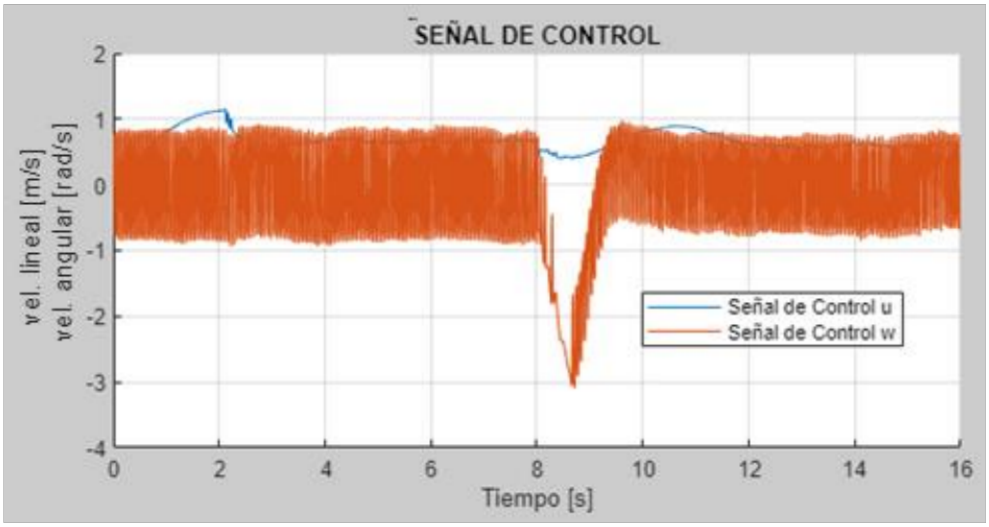


Figura 3.31. Señales de Control cascada Lyapunov + SMC con PS usando una trayectoria tipo L

En la Figura 3.31 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada con Lyapunov + SMC & PS, en esta gráfica se nota que la señal de control de la velocidad lineal tiene un rango de valores parecido a los anteriores controladores y la velocidad angular presenta valores pico de -3 $\mathrm{rad/s}$ siendo menor a los anteriores controladores, aunque existe el problema del chattering debido a la función sigmoide utilizada en la codificación del control SMC.

3.4.2 Trayectoria Triangular

En la Figura 3.32 se presenta el seguimiento de la trayectoria triangular de los 3 tipos de controladores dentro del software CoppeliaSim.

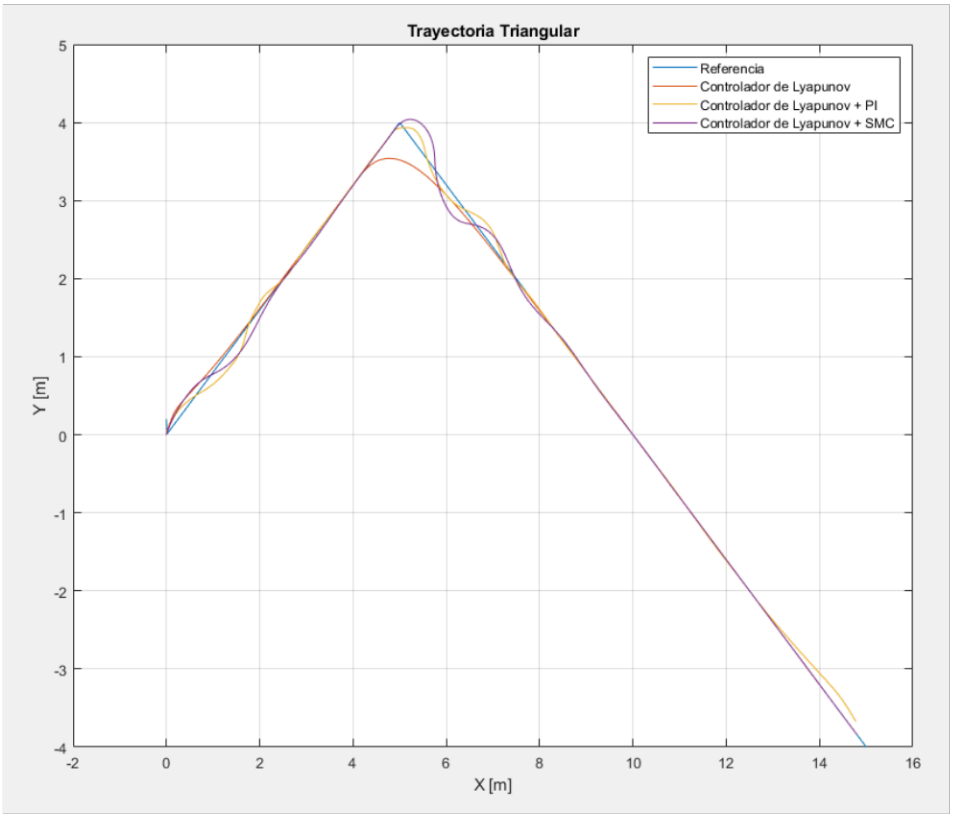


Figura 3.32. Seguimiento de trayectoria triangular

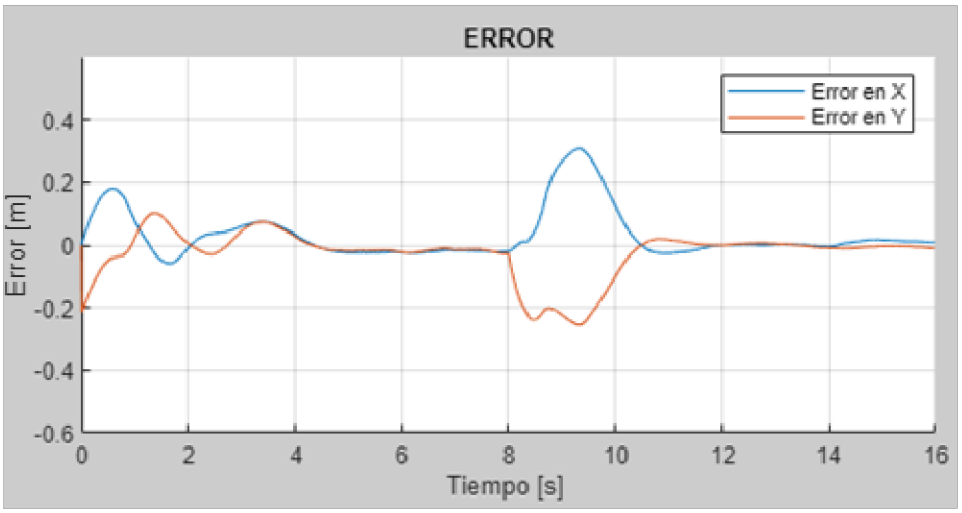


Figura 3.33. Errores Control lazo simple Lyapunov usando una trayectoria Triangular

En la Figura 3.33 se tiene la gráfica de los errores de posición usando un control de lazo simple Lyapunov se nota que a diferencia de la trayectoria tipo L se tiene valores pico de

error al inicio y en el segundo 8 debido a la forma de la trayectoria triangular, los valores convergen a 0 y lo hacen en un tiempo de entre 4 y 5 [s].

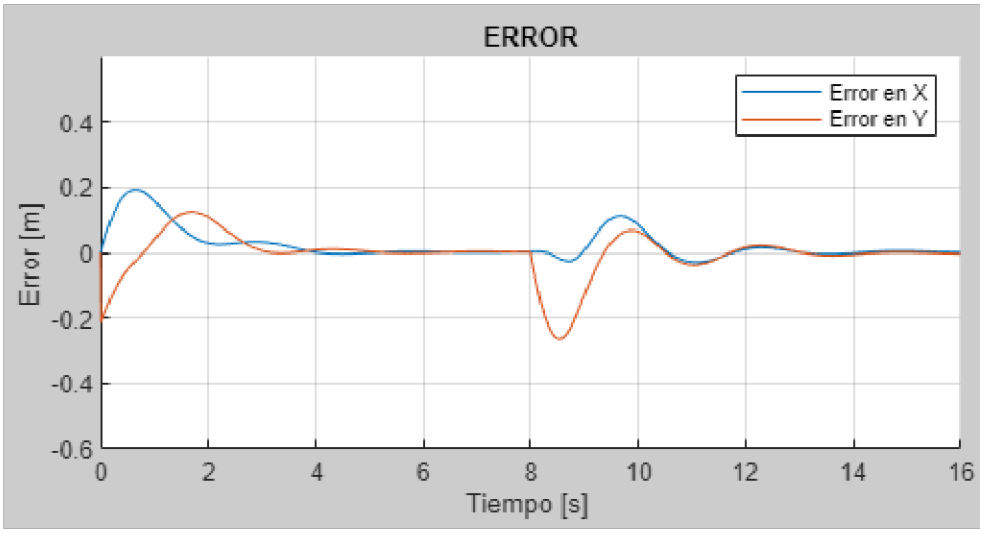


Figura 3.34. Errores Control cascada Lyapunov + PID usando una trayectoria Triangular.

En la Figura 3.34 se tiene la gráfica de los errores de posición usando un control en cascada con Lyapunov + PID, igual que en la figura anterior se alcanza 2 picos al inicio de la trayectoria y en el segundo 8.

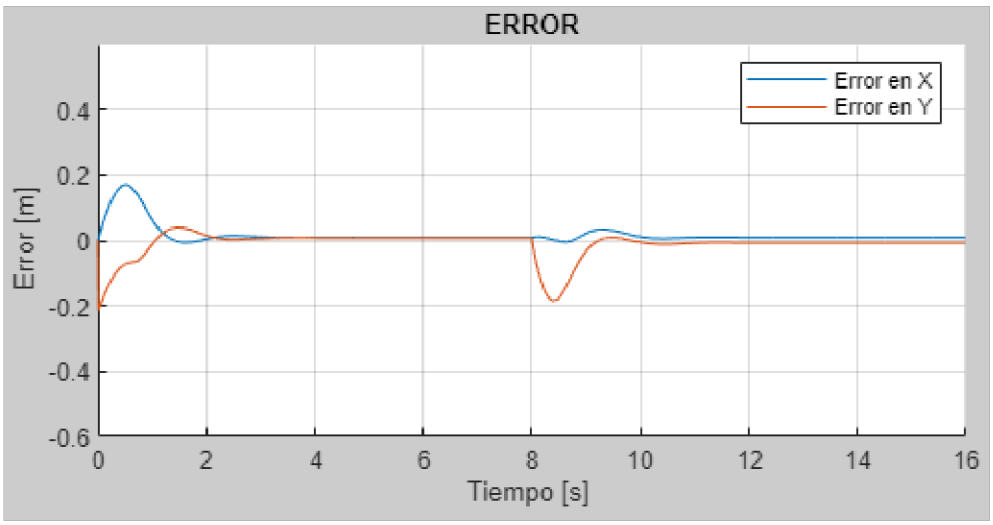


Figura 3.35. Errores Control cascada Lyapunov + SMC con PS usando una trayectoria Triangular

En la Figura 3.35 se tiene la gráfica de los errores de posición usando un control en cascada con Lyapunov + SMC & PS, de igual forma que los anteriores controladores este presenta una reducción del error hasta converger en 0 tanto al inicio como en la mitad del tiempo de simulación.

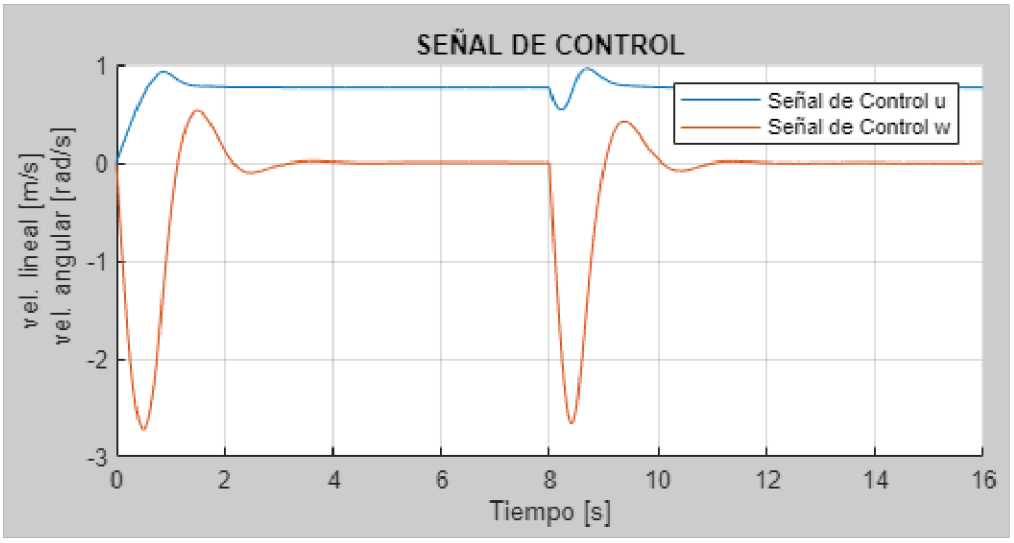


Figura 3. 36. Señales de Control lazo simple Lyapunov usando una trayectoria tipo triangular

En la Figura 3.36 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control de lazo simple con Lyapunov, aquí la señal de control mantiene rangos similares a la trayectoria tipo L con valores de entre 0.8 y $1\ \mathrm{m/s}$ para le velocidad lineal, aunque los valores pico de la señal de control para la velocidad angular se reducen y llegan a valores de aproximadamente -$2.8\ \mathrm{rad/s}$.

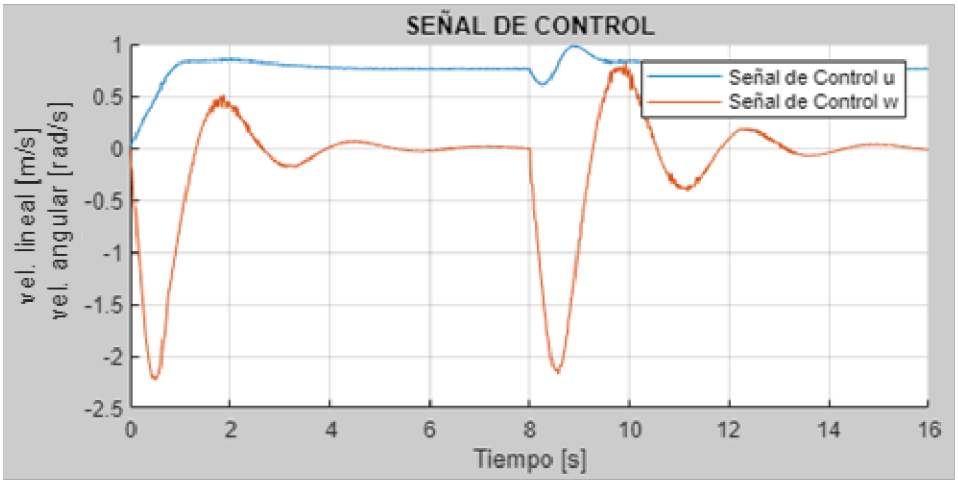


Figura 3.37. Señales de Control cascada Lyapunov + PID usando una trayectoria tipo triangular.

En la Figura 3.37 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada Lyapunov + PID, al igual que el controlador de lazo

simple las señales presentan magnitudes similares, pero el actuador estará sometido durante más tiempo a valores pico lo que puede ser perjudicial.

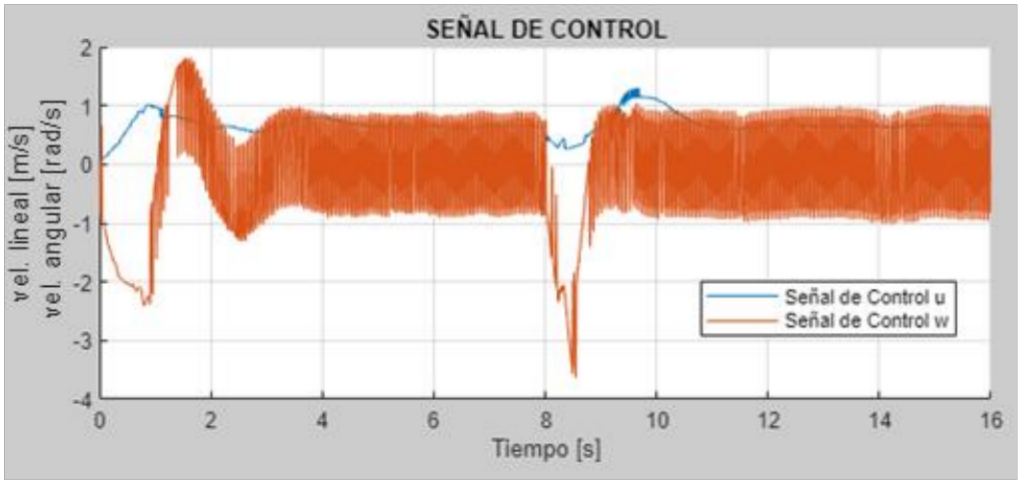


Figura 3.38. Señales de Control cascada Lyapunov + SMC con PS usando una trayectoria tipo triangular

En la Figura 3.38 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada con Lyapunov + SMC & PS, al igual que las anteriores la señal de velocidad lineal los valores que se obtienen son similares, aunque debido al uso del control por modos deslizantes existe el chattering y en la velocidad angular el chattering de forma mayor siendo posible el observar las oscilaciones que tiene esta señal.

3.4.3 Trayectoria Elíptica

En la Figura 3.39 se presenta el seguimiento de la trayectoria triangular de los 3 tipos de controladores dentro del software CoppeliaSim.

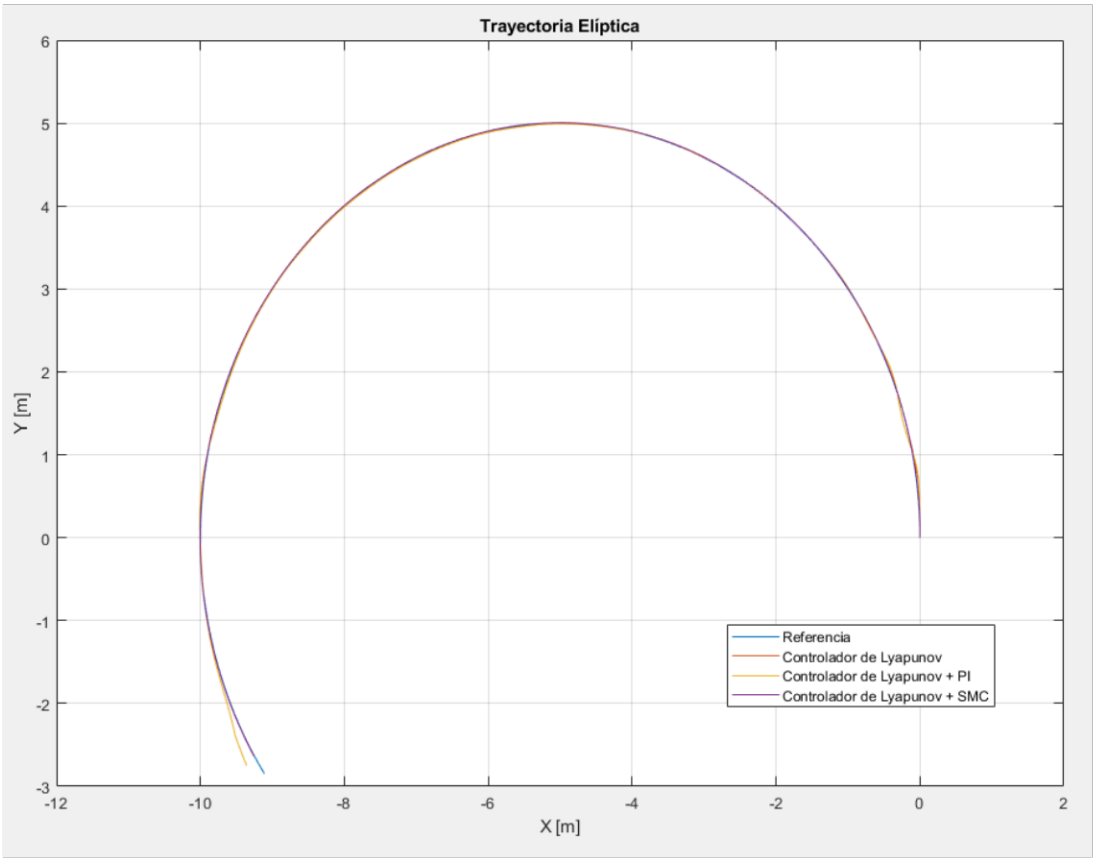


Figura 3.39. Seguimiento de trayectoria elíptica

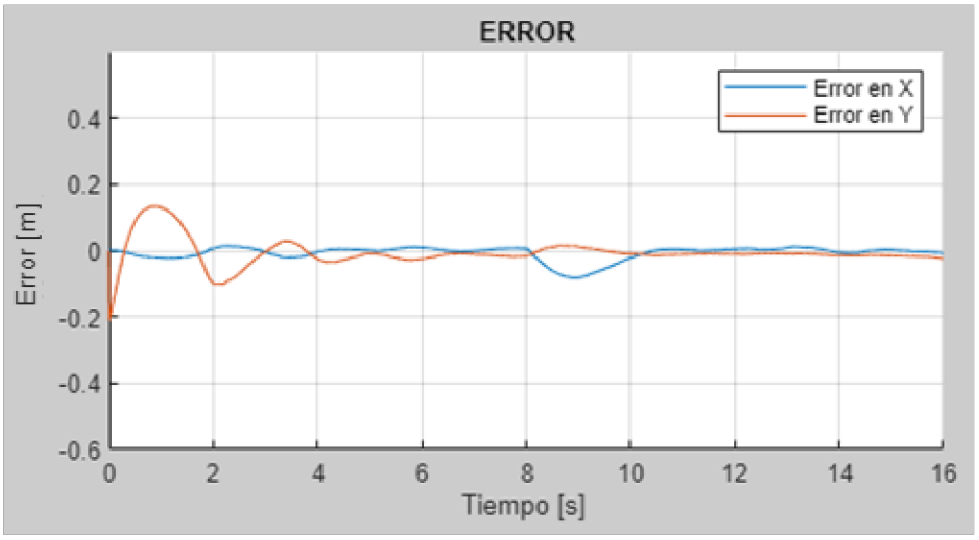


Figura 3.40. Errores Control lazo simple Lyapunov usando una trayectoria Elíptica

En la Figura 3.40 se tiene la gráfica de los errores de posición usando un control de lazo simple Lyapunov, en esta gráfica el error presenta variaciones tanto en x como en y, debido

a la naturaleza de la trayectoria elíptica estos presentan errores de casi 0 en el tiempo de simulación planteado.

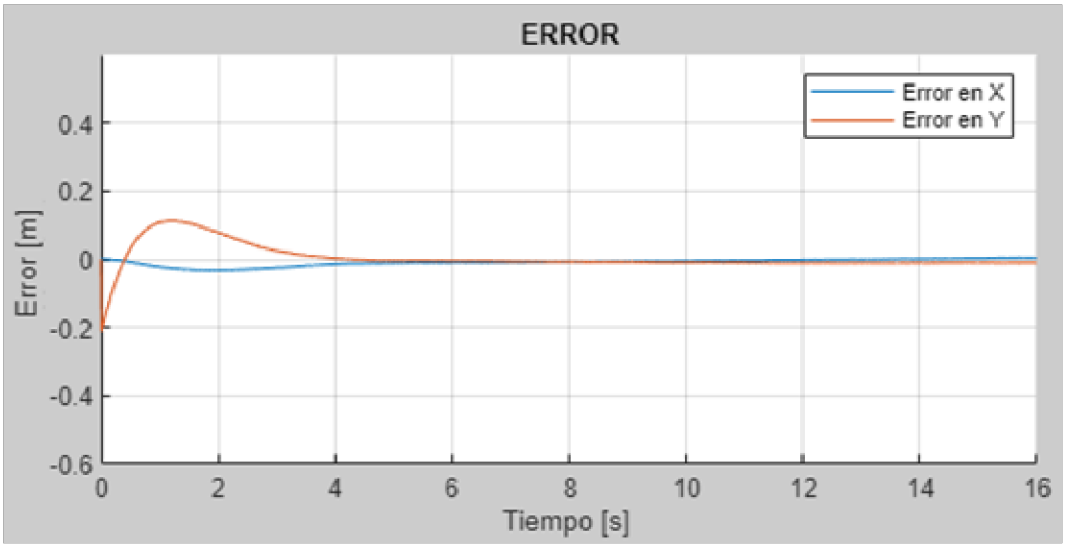


Figura 3.41. Errores Control cascada Lyapunov + PID usando una trayectoria Elíptica.

En la Figura 3.41 se tiene la gráfica de los errores de posición usando un control en cascada Lyapunov + PID, al igual que con el anterior controlador el error presenta errores cercanos a 0 desde el principio.

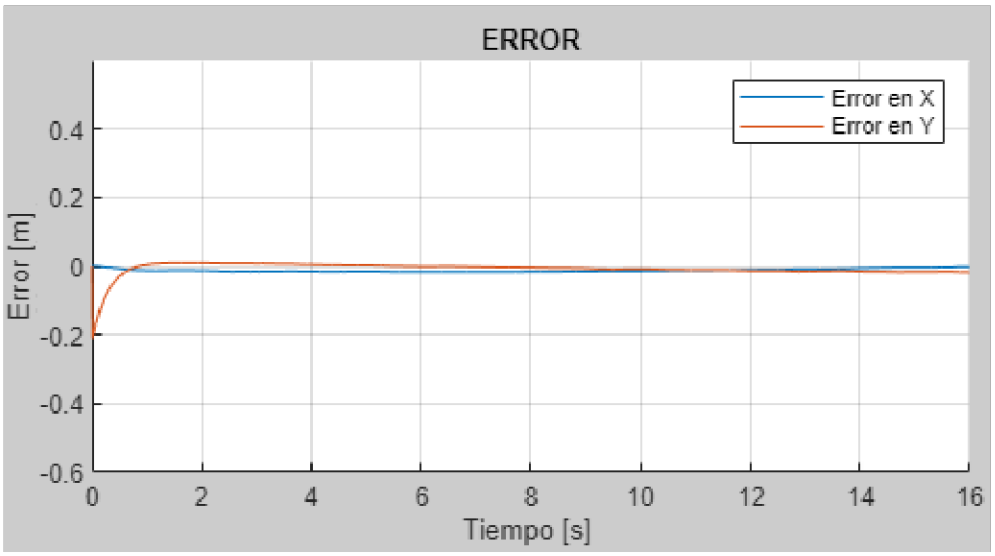


Figura 3.42. Errores Control cascada Lyapunov + SMC con PS usando una trayectoria Elíptica.

En la Figura 3.42 se tiene la gráfica de los errores de posición usando un control en cascada con Lyapunov + SMC & PS, el error converge a 0 aunque este presenta oscilaciones que son visibles a lo largo del tiempo.

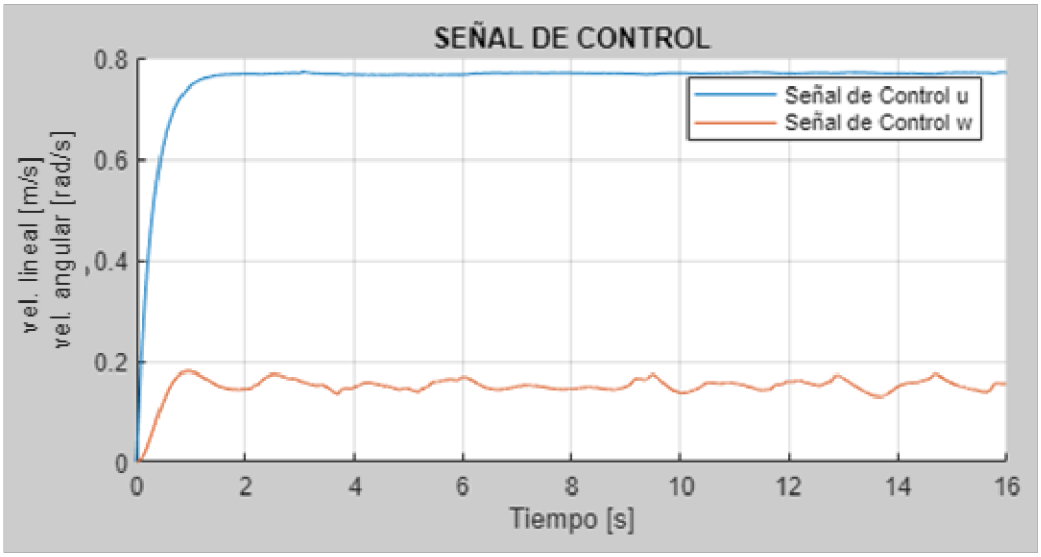


Figura 3.43. Señales de Control lazo simple Lyapunov usando una trayectoria elíptica

En la Figura 3.43 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control de lazo simple Lyapunov, se nota como tanto la señal de control de velocidad lineal como angular presentan valores de $0{,}8\ \mathrm{m/s}$ y de $0{,}2\ \mathrm{rad/s}$ siendo los valores más bajos respecto a las pruebas anteriores.

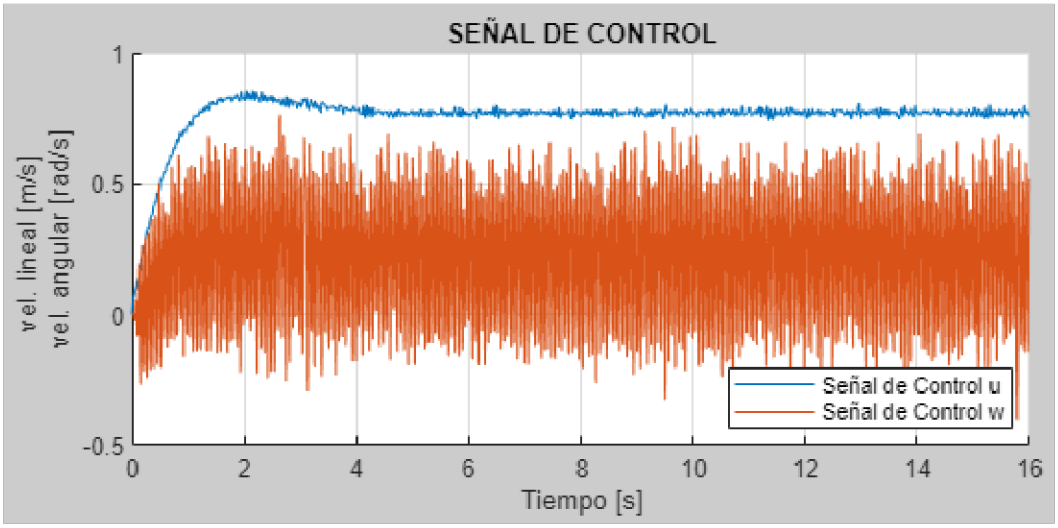


Figura 3.44. Señales de Control cascada Lyapunov + PID usando una trayectoria elíptica.

En la Figura 3.44 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada Lyapunov + PID, se nota que las magnitudes de estas señales se mantienen en valores máximos de velocidad de $0{,}8$, aunque en ambas velocidades, especialmente en la angular la señal de control presenta distorsión y oscilaciones.

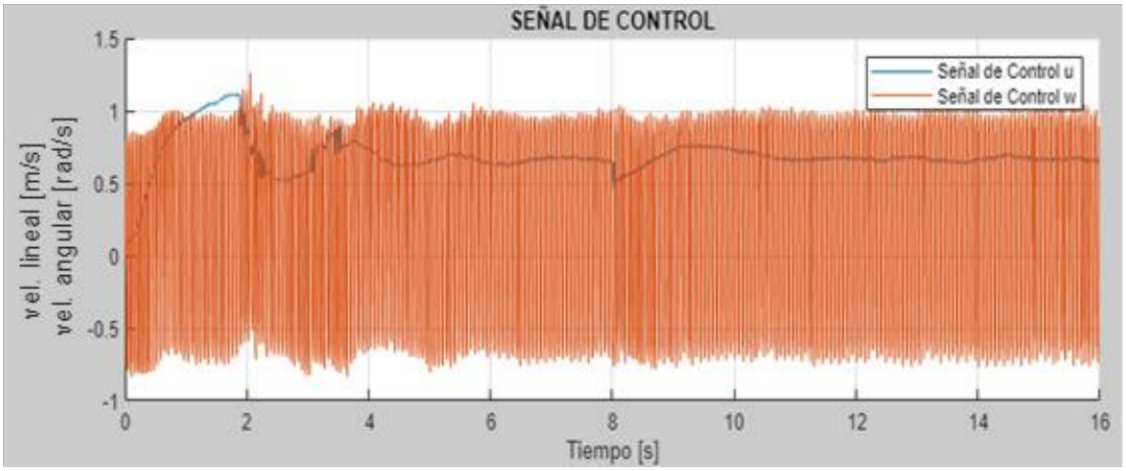


Figura 3.45. Señales de Control Control cascada Lyapunov + SMC con PS usando una trayectoria elíptica

En la Figura 3.45 se tiene la gráfica de las señales de control de las velocidades lineal y angular usando un control en cascada con Lyapunov + SMC & PS, de forma parecida a los anteriores controladores la señal se mantiene por debajo de 1 m/s en velocidad lineal, aunque en la velocidad angular existe chattering.

Tabla 3.11. Índices de desempeño CoppeliaSim

Controlador	**Trayectoria Tipo L**		
	ISE	**ISCOu**	**ISCOw**
Lyapunov	15.701	892.7	633.36
Lyapunov + PID	5.53	897.3	665.5
Lyapunov + SMC & PS	4.98	787.1	1243
Controlador	**Trayectoria Triangular**		
	ISE	**ISCOu**	**ISCOw**
Lyapunov	15.5718	917.9	374.7
Lyapunov + PID	8.35	952.9	379.7
Lyapunov + SMC & PS	6.8	762.1	1539
Controlador	**Trayectoria Elíptica**		

	ISE	ISCOu	ISCOw
Lyapunov	2.66	956.91	193.4
Lyapunov + PID	2.29	933.5	179.4
Lyapunov + SMC & PS	1.06	808.5	1071

Los índices obtenidos se calcularon usando los algoritmos del subcapítulo 2.5 del presente documento, gracias a esos índices se puede notar que al usar el control de Lyapunov + SMC & PS los errores se reducen, aunque la señal de control se verá afectada, que tan alta se vuelva la señal de control dependerá de los parámetros de sintonización del controlador por lo que es necesario que estos tengan valores previamente sintonizados.

3.5 DISCUSIÓN DE RESULTADOS Y HALLAZGOS

Primero se comentará los resultados de la sección 3.1 en la cuál se tiene que el modelo de CoppeliaSim tiene diferencias bastante notables con respecto a los modelos cinemático y cinemático con restricción no holonómica mejorada debido a que estos modelos no toman en cuenta las características dinámicas de la planta, se puede notar al realizar la Prueba I que el modelo que tiene una mayor semejanza es el modelo dinámico.

En este apartado se discutirán los resultados obtenidos en la sección 3.2

Es evidente que para las tres trayectorias es control SMC con esquema de Predictor de Smith en el lazo interno tuvo un mejor desempeño, pues se obtiene el índice ISE menor en los tres casos y se obtiene un seguimiento más preciso a la referencia. Sin embargo, el Chattering presente en la acción de control es de gran amplitud durante toda la trayectoria, se debe tener en cuenta que esta no es la señal que ingresa en el motor DC en forma de voltaje, sino es la referencia de velocidad que alimenta el controlador PD interno que se mencionó en el marco teórico respecto al modelo dinámico. La interpretación de esta señal puede resultar en un trabajo difícil de lograr y se requeriría de filtros para reducir las oscilaciones a alta frecuencia.

Por otro lado, el desempeño del controlador PID en lazo interno no es bajo y la señal de control es una señal continua que fácilmente puede ser suministrada en el elemento final de control, sin embargo, no debería ser utilizada cuando se requiera exactitud en el seguimiento, pues mantiene un error significativo por mucho más tiempo más notablemente en las trayectorias elíptica y triangular.

Por último, el controlador en lazo simple de Lyapunov presenta un desempeño muy bajo, su única ventaja es que presenta el menor índice ISCO, pero su error en estado estable es oscilatorio y no converge a cero en el rango de simulación.

En este apartado se discutirán los resultados obtenidos en la sección 3.3

Sobre las constantes de sintonización y como estos inciden en la respuesta se tiene que:

- En el control de lazo simple con Lyapunov se tiene que la constante afectará directamente con la velocidad, error y señal de control, si K aumenta su valor los errores se reducirán y las señales de control aumentará, aunque hay que tener cierta precaución ya que si el valor de K es demasiado alto este puede llevar a la respuesta a ser inestable disparando los índices de error y control, finamente el tipo de trayectoria los límites de variación de la constante aumentarán o reducirán

- En el control en cascada con lazo externo Lyapunov y control interno con PID se tiene que las constantes K, kp y *ki* afectarán la repuesta del sistema, al igual que el controlador anterior variando K los errores reducirán su valor y las señales de control aumentan, aunque si se sobredimensiona K pueden llegar a la inestabilidad, ahora al variar Kp permite reducir los errores de posición en pequeña medida pero las señales de control aumentan de forma más pronunciada y sobre la constante integral Ki esta reducirá la magnitud de la señal de control aunque aumentará los errores de posición

- En el control en cascada con lazo externo Lyapunov y control interno con SMC usando el Predictor de Smith se tiene que las constantes K, K_D y δ afectarán la repuesta del sistema, al igual que el controlador anterior variando K los errores reducirán su valor y las señales de control aumentan, aunque si se sobredimensiona K pueden llegar a la inestabilidad, ahora al variar K_D permite reducir los errores de posición en pequeña medida pero las señales de control aumentan de forma más pronunciada y δ al variar delta el resultado es disminuir el valor de la señal de control debido a que es el factor de suavizamiento de la función sigmoide.

Ahora al analizar las respuestas de los 3 controladores aplicados para las distintas trayectorias se evidencia claramente que el controlador usando SMC con esquema de Predictor de Smith en el lazo interno presenta un mejor seguimiento de la referencia además de presentar los menores índices ISE, por otro lado las señales de control obtenidas con el indicador ISCO son de las más grandes debido al problema del

‘Chattering’ esto produce oscilaciones indeseadas en la acción de control y podría llegar a afectar a los motores del Pioneer 3DX si no se regula, aunque esto puede ser solucionado con el control PD que tiene el robot internamente.

Sobre los otros 2 esquemas se tiene que el control en cascada con lazo interno tipo PID presenta una respuesta que, aunque no corrige el error con la misma eficiencia que el SMC+PS su índice ISE no es excesivamente grande y presenta un ISCO más bajo y sin el problema de las oscilaciones.

Para el control de lazo simple con Lyapunov se tiene índices ISCO menores, aunque debido a eso los errores pueden llegar a ser más altos.

3.6 CONCLUSIONES

- Se desarrolló una interfaz que permite obtener, visualizar los valores necesarios para determinar la calidad y rendimiento de un controlador, además de una pantalla en la que se puede comparar las mejores respuestas obtenidas para cada controlador en el seguimiento de las 3 trayectorias
- A pesar de que el controlador que mejores respuestas presenta fue el SMC + Predictor de Smith los beneficios frente a los otros 2 controles presentados no fueron muy visibles ya que no se realizaron pruebas con altos retardos ni con perturbaciones representativas.
- Para la elección del tiempo de muestreo de los datos en MATLAB y CoppeliaSim se tomó en cuenta el tiempo de establecimiento en lazo abierto de los modelos tanto de velocidad lineal como de velocidad angular, obteniendo un valor de 10 [ms] para evitar problemas como el aliasing.
- Se pudo realizar un análisis de cómo cambia tanto el seguimiento como los índices de desempeño al variar cada una de las constantes presentes en los controladores, comprobando que dependiendo del tipo de trayectoria estas constantes pueden tener efectos más o menos pronunciados sobre la respuesta.
- Debido al esquema cascada que se realiza en los controladores PID y SMC con Predictor de Smith estos dependerán en gran medida de la respuesta del

compensador de lazo externo de Lyapunov, por lo que los controles funcionarán mejor si se usa una constante K de Lyapunov más grande.

3.7 RECOMENDACIONES

- Para reducir el efecto del "Chattering" presente en la señal de control del controlador SMC+SP se recomienda el uso de filtros a la entrada del controlador PD interno del Pioneer
- Como la finalidad del trabajo se centraba en realizar controles de seguimiento de trayectorias fijas no se tomó en cuenta obstáculos ni algoritmos de evasión, algo que sería interesante de añadir en trabajos futuros.
- Algo que se podría añadir es el uso de saturadores dentro de la codificación en MATLAB como un método de protección extra al PID interno de CoppeliaSim.

4 REFERENCIAS BIBLIOGRÁFICAS – TOMO II

[1] The MathWorks Inc., «mathworks,» MATLAB Version R2021a, [En línea]. Available: https://la.mathworks.com/. [Último acceso: 4 Enero 2022].

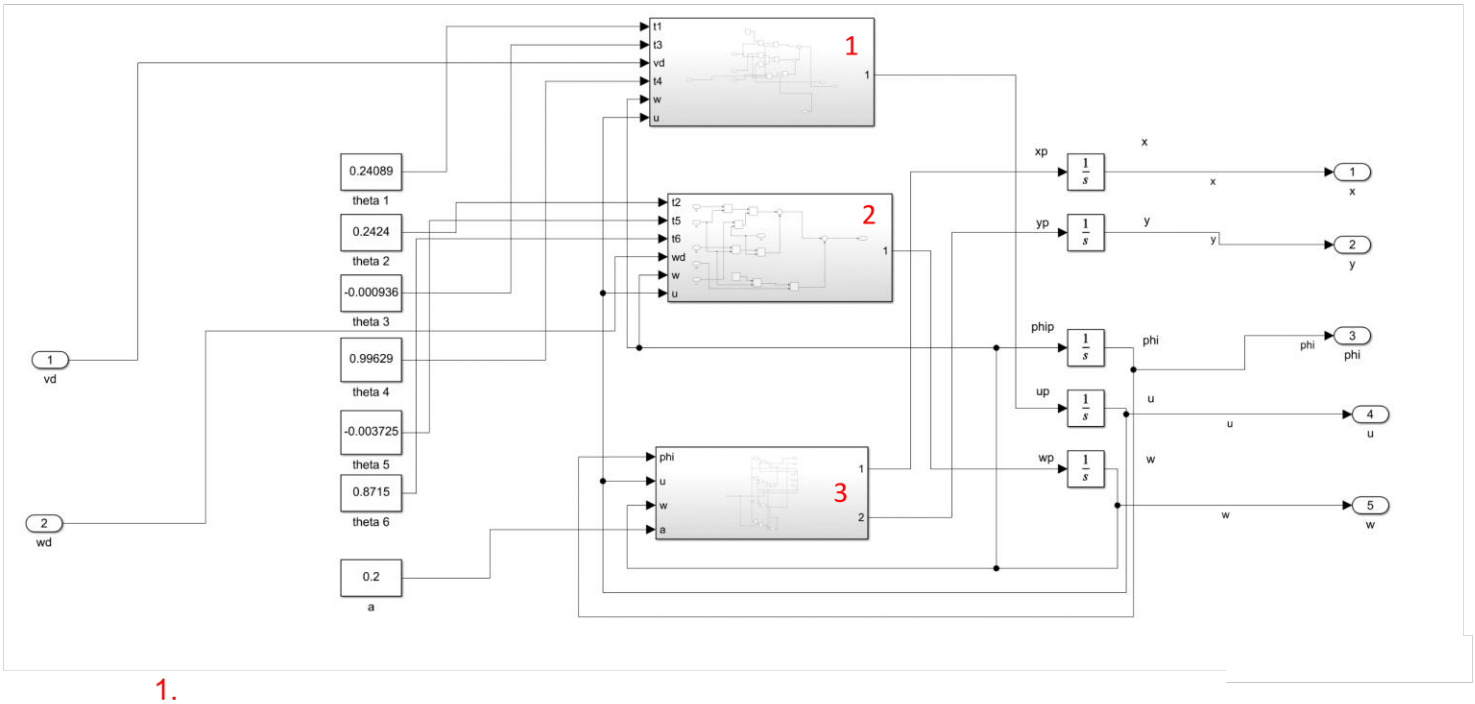
[2] CoppeliaSim, «coppeliarobotics,» CoppeliaSim User Manual, [En línea]. Available: https://www.coppeliarobotics.com/helpFiles/. [Último acceso: 4 Enero 2022].

[3] M. Egerstedt "Control of mobile robots course", Differential Drive robots, Georgia Tech, Georgia, USA, 2013. [Último acceso: 16 Enero 2022].

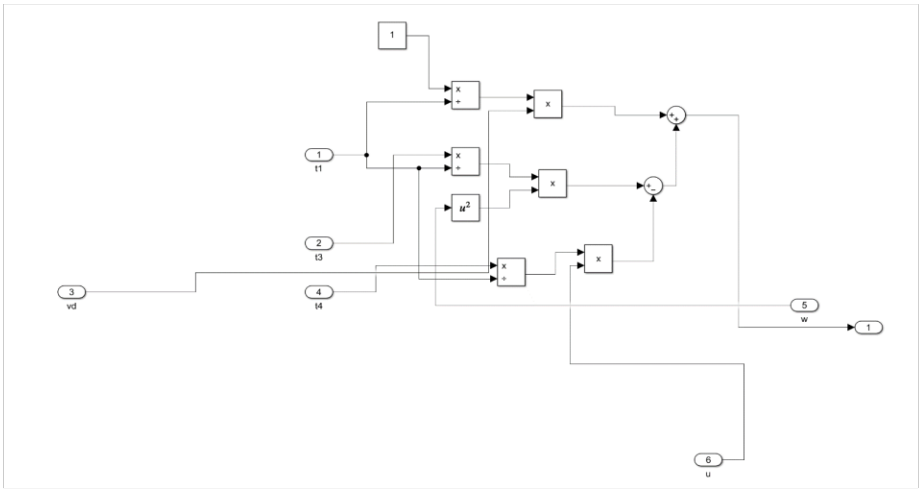
5 ANEXOS

ANEXO I

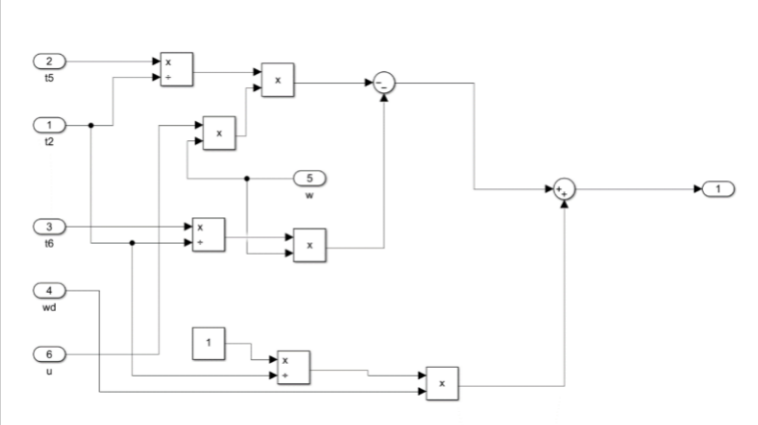
A continuación, se presenta el diagrama de bloques del modelo dinámico:



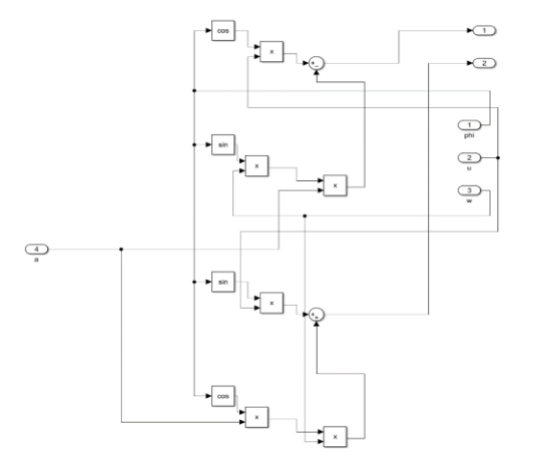
1.



2.



3.



ANEXO II

Generación de Trayectorias

```
clc;
clear all;
t1 = 0:0.01:5;;
t2 = 5.01:0.01:10;
x1 = 5*cos(0.7*t1)-5;
y1 = 5*sin(0.7*t1);
x2 = 5*cos(0.7*t2)-5;
y2 = 5*sin(0.7*t2);
x3 = t1;
y3 = 0.8*x3;
x4 = t2;
y4 = 8-0.8*t2;
x5 = zeros(501);
y5 = t1;
x6 = t2-5;
y6 = 5*ones(500);
xu1 = [x1 x2];
yu1 = [y1 y2];
xu2 = [x3 x4];
yu2 = [y3 y4];
xu3 = [x5(1,:) x6];
yu3 = [y5 y6(1,:)];
plot(xu1,yu1,'b',xu2,yu2,'k',xu3,yu3,'r');
title('Trayectorias');
legend('Epiptica','Triangular','Tipo L');
xlabel('X [m]');
ylabel('Y [m]');
xlim([-10 10]);
ylim([-10 10]);
```

ANEXO III

A continuación, se presenta los códigos para los bloques "MATLAB FUNCTION" presentados en este trabajo:

Controlador por Lyapunov

```
function [vc,wc] = fcn(xdp,ydp,xd,x,yd,y,phi)

a = 0.2;
K = 1;


hdp = [xdp;ydp];


he = [xd-x;yd-y];


J  = [cos(phi), -a*sin(phi); sin(phi), a*cos(phi)];
Ji = inv(J);


Uc = Ji*(hdp+K*he);


vc = Uc(1);
wc = Uc(2);
```

Controlador SMC

```
function [vc,wc] = fcn(ev,iev,ew,iew,v,w)


KDv = 25.72;
KDw = 31.69;
dv = 1;
dw = 1;




Sv = sign(10*ev+30*iev);
Sw = sign(10*ew+30*iew);


vc = v+0.18291*ev+KDv*((Sv)/(abs(Sv)+dv))


wc = 0.8761*w+0.2726*ew+KDw*((Sw)/(abs(Sw)+dw))
```

ANEXO IV

A continuación se presenta el código para la obtención de gráficas y el cálculo del ISE unificado y el ISCO de cada acción de control, teniendo los punteros en simulink que permitan exportar datos a un archivo .m :

Código para resultados

```
plot(out.xd.Data,out.yd.Data,out.xc1.Data,out.yc1.Data,out.xc2.Data,out.yc2.Data,out.xc3
.Data,out.yc3.Data);

legend('Referencia','Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Seguimiento en plano XY');
xlabel('X [m]');
ylabel('Y[m]');

plot(out.xd);
hold
plot(out.xc1);
plot(out.xc2);
plot(out.xc3);
legend('Referencia','Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Seguimiento en X');
xlabel('Tiempo [t]');
ylabel('X[m]');

plot(out.yd);
hold
plot(out.yc1);
plot(out.yc2);
plot(out.yc3);
legend('Referencia','Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Seguimiento en Y');
xlabel('Tiempo [t]');
ylabel('Y[m]');
ex1 = out.xd.Data-out.xc1.Data;
ex2 = out.xd.Data-out.xc2.Data;
ex3 = out.xd.Data-out.xc3.Data;
plot(out.xd.Time,ex1,out.xd.Time,ex3,out.xd.Time,ex2)
legend('Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Error en X');
xlabel('Tiempo [t]');
ylabel('Error[m]');


ey1 = out.yd.Data-out.yc1.Data;
ey2 = out.yd.Data-out.yc2.Data;
ey3 = out.yd.Data-out.yc3.Data;
plot(out.xd.Time,ey1,out.xd.Time,ey2,out.xd.Time,ey3)
legend('Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Error en Y');
xlabel('Tiempo [t]');
ylabel('Error[m]');
```

```
ey1 = out.yd.Data-out.yc1.Data;
ey2 = out.yd.Data-out.yc2.Data;
ey3 = out.yd.Data-out.yc3.Data;
plot(out.xd.Time,ey1,out.xd.Time,ey2,out.xd.Time,ey3)
legend('Control Lyapunov', 'Lyapunov+PI','Lyapunov+SMC/SP')
title('Error en Y');
xlabel('Tiempo [t]');
ylabel('Error[m]');

plot(out.v3);
hold
plot(out.v1,'LineWidth',1.1);
plot(out.v2,'LineWidth',1.1);
legend('Lyapunov+SMC/SP','Control Lyapunov', 'Lyapunov+PI')
title('Señal de control V. Lineal');
xlabel('Tiempo [t]');
ylabel('Velocidad lineal [m/s]');

plot(out.w3);
hold
plot(out.w1,'LineWidth',1.1);
plot(out.w2,'LineWidth',1.1);
legend('Lyapunov+SMC/SP','Control Lyapunov', 'Lyapunov+PI')
title('Señal de control V. Angular');
xlabel('Tiempo [t]');
ylabel('Velocidad angular [rad/s]');

ISE1 = trapz(ex1.^2+ey1.^2)
ISE2 = trapz(ex2.^2+ey2.^2)
ISE3 = trapz(ex3.^2+ey3.^2)
ISCOv1 = trapz((out.v1.Data).^2)
ISCOv2 = trapz((out.v2.Data).^2)
ISCOv3 = trapz((out.v3.Data).^2)
ISCOw1 = trapz((out.w1.Data).^2)
ISCOw2 = trapz((out.w2.Data).^2)
ISCOw3 = trapz((out.w3.Data).^2)
```

ANEXO V

Código controladores

```
% DEFINICION Cliente-Servidor remAPI
vrep = remApi('remoteApi');
vrep.simxFinish(-1);
clientID = vrep.simxStart('127.0.0.1', 19997, true, true, 5000, 5);


% INICIO DE LA COMUNICACION SINCRONICA
vrep.simxSynchronous(clientID,true); % Inicio de comunicación sincrónica
vrep.simxStartSimulation(clientID,vrep.simx_opmode_blocking); % Inicio externo de Coppelia


% VARIABLES USADAS


Tm = 0.01; % Tiempo de muestreo dentro de Coppelia
xref = []; % Referencia de X de la TRAYECTORIA
yref = []; % Referencia de Y de la TRAYECTORIA
yout = []; % Salida Y de la PLANTA
xout = []; % Salida X de la PLANTA
xerror = []; % X Error
yerror = []; % Y Error
phiout = []; % Salida PHI de la PLANTA
vxref = []; % Velocidad X de la TRAYECTORIA
vyref = []; % Velocidad Y de la TRAYECTORIA
uref = []; % Velocidad Lineal Salida del lazo externo
wref = []; % Velocidad Angular Salida del lazo externo
uerror = []; %Error de velocidad lineal
werror = []; %Error de velocidad angular
uinerror = []; %Error de velocidad lineal
winerror = []; %Error de velocidad angular
uout = []; % Velocidad lineal de la PLANTA
wout = []; % Velocidad angular de la PLANTA
u_proporcional = []; % Accion Proporcional de la velocidad lineal
u_integral = []; % Accion Integral de la velocidad lineal
w_proporcional = []; % Accion Proporcional de la velocidad angular
w_integral = []; % Accion Integral de la velocidad angular
u = []; % Velocidad lineal entrada a la PLANTA
w = []; % Velocidad angular entrada a la PLANTA
velocidad_rueda_left = []; % Velocidad de la rueda izquierda
velocidad_rueda_right = []; % Velocidad de la rueda derecha
sv = []; % Superficie deslizante para control en u
sw = []; % Superficie deslizante para control en w
spsv = []; % Salida predictor de Smith velocidad lineal
spsw = [];% Salida predictor de Smith velocidad angular
dRv = [];
dRw = [];
Xv = [];
Xw = [];
Xvd = [];
Xwd = [];
ISE = 0;
ISCOu = 0;
```

```
% CONSTANTES CONTROLADORES
% Constante de Lypunov
switch app.valorEditField.Value
case 0
K = app.kEditField.Value;
app.Label.Text = "Control Lazo Simple (LYAPUNOV)";
case 1
K = app.kpidEditField.Value;
app.Label.Text = "Control Esquema Cascada (LYAPUNOV + PID)";
case 2
K = app.ksmcEditField.Value;
app.Label.Text = "Control Esquema Cascada (LYAPUNOV + SMC con Predictor de Smith)";
end
% Constantes PI
kpu = app.kpuEditField.Value;
kpw = app.kpwEditField.Value;
kiu = app.kiuEditField.Value;
kiw = app.kiwEditField.Value;


% CONSTANTES SMC
KDv = app.KDuEditField.Value;
KDw = app.KDwEditField.Value;
dv = app.duEditField.Value;
dw = app.dwEditField.Value;


% Parametros SP
Kv = 1;
tauv = 0.063245;
Kw = 1.141833;
tauw = 0.09373;


if (clientID>-1)
disp('connected')


% OBTENER LOS HANDLES


[returnCode,left_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.si
mx_opmode_blocking); % MOTOR IZQUIERDO
[returnCode,right_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMotor',vrep.
simx_opmode_blocking); % MOTOR DERECHO
[~,Pioneer]= vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx',vrep.simx_opmode_blocking);
% ROBOT PIONEER
[~,Referencia]= vrep.simxGetObjectHandle(clientID,'Dummy',vrep.simx_opmode_blocking); %
ROBOT PIONEER
```

```
% VALORES INICIALES DEL PIONEER 3DX


xref(1)=0; %Posicion inicial de la trayectoria en x
yref(1)=a; %Posicion inicial de la trayectoria en y
vxref(1)=0; %Derivada de la trayectoria en x inicial
vyref(1)=0; %Derivada de la trayectoria en y inicial


xout(1) = 0; %Posicion en x sensada inicial
yout(1) = 0; %Posicion en y sensada inicial
phiout(1) = 0; %Posicion angular phi sensada inicial


uref(1) = 0; % Velocidad lineal de referencia inicial
wref(1) = 0; % Velocidad angular de referencia inicial
uout(1) = 0; % Velocidad lineal de salida inicial
wout(1) = 0; % Velocidad angular de salida inicial
uerror(1) = 0.2; % Error de velocidad lineal inicial
werror(1) = 0; % Error de velocidad angular inicial


u_proporcional(1) = 0; % Accion Proporcional de la velocidad lineal
u_integral(1) = 0; % Accion Integral de la velocidad lineal
w_proporcional(1) = 0; % Accion Proporcional de la velocidad angular
w_integral(1) = 0; % Accion Integral de la velocidad angular


uinerror(1) = 0;
winerror(1) = 0;


dRv(1) = 0; % DERIVADA DE LA REFERENCIA DE Velocidad
dRw(1) = 0;


Xvd(1) = 0; % SALIDA DEL MODELO CON RETARDO
Xwd(1) = 0; % SALIDA DEL MODELO CON RETARDO


spsv(1) = 0; % VELOCIDAD DE ENTRADA AL COMPARADOR DE LAZO INTERNO
spsw(1) = 0;


Xv(1) = 0; % SALIDA DEL MODELO SIN RETARDO
Xw(1) = 0;


u(1)= 0; % Velocidad lineal inicial hacia la PLANTA
w(1)= 0; % Velocidad angular inicial hacia la PLANTA
```

```
velocidad_rueda_left(1) = (2*u(1)-w(1)*L)/(2*R); % Expresión de velocidad inyectada en
rueda izquierda


[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor,
velocidad_rueda_right(1), vrep.simx_opmode_oneshot); %Envío AC en Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, velocidad_rueda_left(1),
vrep.simx_opmode_oneshot); %Envío AC en Rueda izquierda
[returnCode]=vrep.simxSetObjectPosition(clientID,Referencia, -
1,[xref,yref,0],vrep.simx_opmode_oneshot); %Envío AC en Rueda izquierda
vrep.simxSynchronousTrigger(clientID); %// Trigger para siguiente paso de simulacion




for i= 2:500


%Generacion de trayectoria
switch app.trayectoriaEditField.Value


case 0
%CUADRADA
xref(i) = 0; % Referencia x instantanea
yref(i) = (i-1)*Tm; % Referencia y instantanea
case 1
%TRIANGULAR
xref(i) = (i-1)*Tm; % Referencia x instantanea
yref(i) = 0.8*xref(i); % Referencia y instantanea
case 2
%ELIPSE
xref(i) = 5*cos((i-1)*Tm/4)-5; % Referencia x instantanea
yref(i) = 5*sin((i-1)*Tm/4); % Referencia y instantanea
end




vxref(i) = (Tm/2)*(xref(i)-xref(i-1))-vxref(i-1); % Referencia vx instantanea
vyref(i) = (Tm/2)*(yref(i)-yref(i-1))-vyref(i-1); % Referencia vy instantanea


[returnCode,position]=vrep.simxGetObjectPosition(clientID, Pioneer,-1,
vrep.simx_opmode_blocking); % Obtencion de valores X y Y de la PLANTA
[returnCode,orientation]=vrep.simxGetObjectOrientation(clientID, Pioneer,-1,
vrep.simx_opmode_blocking); % Obtencion de valores X y Y de la PLANTA


xout(i) = position(1);
yout(i) = position(2);
phiout(i) = orientation(3);
```

```
J=[cos(phiout(i)),-a*sin(phiout(i));sin(phiout(i)),a*cos(phiout(i))];


Ji = inv(J);


xerror(i) = xref(i)-xout(i);
yerror(i) = yref(i)-yout(i);


hdp = [vxref(i);vyref(i)];


exp = hdp+K*[xerror(i);yerror(i)];


Uc = Ji*exp;


uref(i) =Uc(1);
wref(i) =Uc(2);


%Generacion de trayectoria
switch app.valorEditField.Value


case 0
u(i) = uref(i);
w(i) = wref(i);


case 1


[returnCode,linear_vel,angular_vel]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking); %Obtencion de las velocidades lineales y angulares de la
PLANTA
uout(i) = sqrt(linear_vel(1)^2 + linear_vel(2)^2);
wout(i) = angular_vel(3);
uerror(i) = uref(i)-uout(i);
werror(i) = wref(i)-wout(i);
u_proporcional(i) = kpu*uerror(i);
w_proporcional(i) = kpw*werror(i);
u_integral(i) = ((kiu*Tm)/2)*(uerror(i)+uerror(i-1))+u_integral(i-1);
w_integral(i) = ((kiw*Tm)/2)*(werror(i)+werror(i-1))+w_integral(i-1);
u(i) = u_proporcional(i) + u_integral(i);
w(i) = w_proporcional(i) + w_integral(i);


case 2
```

```
uout(i) = sqrt(linear_vel(1)^2 + linear_vel(2)^2);
wout(i) = angular_vel(3);


uerror(i) = uref(i-1)-spsv(i-1);
werror(i) = wref(i-1)-spsw(i-1);
uinerror(i) = (Tm/2)*(uerror(i)+uerror(i-1))+uinerror(i-1);
winerror(i) = (Tm/2)*(werror(i)+werror(i-1))+winerror(i-1);
sv(i) = sign(10*uerror(i)+30*uinerror(i)); % SUPERFICIE DESLIZANTE
sw(i) = sign(10*werror(i)+30*winerror(i));
dRv(i) = (2/Tm)*(uref(i)-uref(i-1))-dRv(i-1);
dRw(i) = (2/Tm)*(wref(i)-wref(i-1))-dRw(i-1);
u(i) = 0.0006324*dRv(i)+0.1897*uerror(i)+uout(i)+KDv*((sv(i))/(dv+abs(sv(i))));
w(i) = 0.00082*dRw(i)+0.2462*werror(i)+0.8757*wout(i)+KDw*((sw(i))/(dw+abs(sw(i))));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% PREDICTOR DE SMITH %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Xv(i) = (Kv*Tm*u(i)+Kv*Tm*u(i-1)-(Tm-2*tauv)*Xv(i-1))/(Tm+2*tauv);
Xw(i) = (Kw*Tm*w(i)+Kw*Tm*w(i-1)-(Tm-2*tauw)*Xw(i-1))/(Tm+2*tauw);
if (i<=4)
Xvd(i) = 0;
Xwd(i) = 0;
else
Xvd(i) = Xv(i-4); % RETARDOS PARA EL PREDICTOR DE SMITH
Xwd(i) = Xw(i-4);
end
spsv(i) = uout(i)-Xvd(i)+Xv(i);
spsw(i) = wout(i)-Xwd(i)+Xw(i);


end


[returnCode]=vrep.simxSetObjectPosition(clientID,Referencia, -
1,[xref(i),yref(i),0],vrep.simx_opmode_oneshot); %Envío AC en Rueda izquierda
velocidad_rueda_right(i) = (2*u(i)+w(i)*L)/(2*R); % Expresión de velocidad inyectada en
rueda derecha
velocidad_rueda_left(i) = (2*u(i)-w(i)*L)/(2*R); % Expresión de velocidad inyectada en
rueda izquierda


[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor,
velocidad_rueda_right(i), vrep.simx_opmode_oneshot); %Envío AC en Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, velocidad_rueda_left(i),
vrep.simx_opmode_oneshot); %Envío AC en Rueda izquierda
vrep.simxSynchronousTrigger(clientID); %// Trigger para siguiente paso de simulacion


end
```

```
for i= 501:1500


%Generacion de trayectoria
switch app.trayectoriaEditField.Value


case 0
%CUADRADA
xref(i) = ((i-500)*Tm)+a; % Referencia x instantanea
yref(i) = (4.99); % Referencia y instantanea
case 1
%TRIANGULAR
xref(i) = 5 + (i-500)*Tm; % Referencia x instantanea
yref(i) = 4 - 0.8*(i-500)*Tm; % Referencia y instantanea
case 2
%ELIPSE
xref(i) = 5*cos((i-1)*Tm/4)-5; % Referencia x instantanea
yref(i) = 5*sin((i-1)*Tm/4);
end


vxref(i) = (Tm/2)*(xref(i)-xref(i-1))-vxref(i-1); % Referencia vx instantanea
vyref(i) = (Tm/2)*(yref(i)-yref(i-1))-vyref(i-1); % Referencia vy instantanea


[returnCode,position]=vrep.simxGetObjectPosition(clientID, Pioneer,-1,
vrep.simx_opmode_blocking); % Obtencion de valores X y Y de la PLANTA
[returnCode,orientation]=vrep.simxGetObjectOrientation(clientID, Pioneer,-1,
vrep.simx_opmode_blocking); % Obtencion de valores X y Y de la PLANTA


xout(i) = position(1);
yout(i) = position(2);
phiout(i) = orientation(3);


J=[cos(phiout(i)),-a*sin(phiout(i));sin(phiout(i)),a*cos(phiout(i))];


Ji = inv(J);


xerror(i) = xref(i)-xout(i);
yerror(i) = yref(i)-yout(i);


hdp = [vxref(i);vyref(i)];


exp = hdp+K*[xerror(i);yerror(i)];
```

```
uref(i) =Uc(1);
wref(i) =Uc(2);


%Generacion de trayectoria
switch app.valorEditField.Value


case 0
u(i) = uref(i);
w(i) = wref(i);


case 1


[returnCode,linear_vel,angular_vel]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking); %Obtencion de las velocidades lineales y angulares de la
PLANTA
uout(i) = sqrt(linear_vel(1)^2 + linear_vel(2)^2);
wout(i) = angular_vel(3);
uerror(i) = uref(i)-uout(i);
werror(i) = wref(i)-wout(i);
u_proporcional(i) = kpu*uerror(i);
w_proporcional(i) = kpw*werror(i);
u_integral(i) = ((kiu*Tm)/2)*(uerror(i)+uerror(i-1))+u_integral(i-1);
w_integral(i) = ((kiw*Tm)/2)*(werror(i)+werror(i-1))+w_integral(i-1);
u(i) = u_proporcional(i) + u_integral(i);
w(i) = w_proporcional(i) + w_integral(i);


case 2
[returnCode,linear_vel,angular_vel]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking); %Obtencion de las velocidades lineales y angulares de la
PLANTA
uout(i) = sqrt(linear_vel(1)^2 + linear_vel(2)^2);
wout(i) = angular_vel(3);


uerror(i) = uref(i-1)-spsv(i-1);
werror(i) = wref(i-1)-spsw(i-1);
uinerror(i) = (Tm/2)*(uerror(i)+uerror(i-1))+uinerror(i-1);
winerror(i) = (Tm/2)*(werror(i)+werror(i-1))+winerror(i-1);
sv(i) = sign(10*uerror(i)+30*uinerror(i)); % SUPERFICIE DESLIZANTE
sw(i) = sign(10*werror(i)+30*winerror(i));
dRv(i) = (2/Tm)*(uref(i)-uref(i-1))-dRv(i-1);
dRw(i) = (2/Tm)*(wref(i)-wref(i-1))-dRw(i-1);
u(i) = 0.0006324*dRv(i)+0.1897*uerror(i)+uout(i)+KDv*((sv(i))/(dv+abs(sv(i))));
w(i) = 0.00082*dRw(i)+0.2462*werror(i)+0.8757*wout(i)+KDw*((sw(i))/(dw+abs(sw(i))));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% PREDICTOR DE SMITH %%%%%%%%%%%%%%%%%%
```

```
Xv(i) = (Kv*Tm*u(i)+Kv*Tm*u(i-1)-(Tm-2*tauv)*Xv(i-1))/(Tm+2*tauv);
Xw(i) = (Kw*Tm*w(i)+Kw*Tm*w(i-1)-(Tm-2*tauw)*Xw(i-1))/(Tm+2*tauw);
if (i<=504)
Xvd(i) = 0;
    Xwd(i) = 0;
    else
    Xvd(i) = Xv(i-4); % RETARDOS PARA EL PREDICTOR DE SMITH
    Xwd(i) = Xw(i-4);
    end

    spsv(i) = uout(i)-Xvd(i)+Xv(i);
    spsw(i) = wout(i)-Xwd(i)+Xw(i);


end


velocidad_rueda_right(i) = (2*u(i)+w(i)*L)/(2*R); % Expresión de velocidad inyectada en
rueda derecha
velocidad_rueda_left(i) = (2*u(i)-w(i)*L)/(2*R); % Expresión de velocidad inyectada en
rueda izquierda

[returnCode]=vrep.simxSetObjectPosition(clientID,Referencia, -
1,[xref(i),yref(i),0],vrep.simx_opmode_oneshot);    %Envío AC en Rueda izquierda
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor,
velocidad_rueda_right(i), vrep.simx_opmode_oneshot);   %Envío AC en Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, velocidad_rueda_left(i),
vrep.simx_opmode_oneshot);    %Envío AC en Rueda izquierda
vrep.simxSynchronousTrigger(clientID); %// Trigger para siguiente paso de simulación

end


%Accion de frenado


[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, 0,
vrep.simx_opmode_oneshot);   % SETEAR VELOCIDAD RUEDA DERECHA
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, 0,
vrep.simx_opmode_oneshot);    % SETEAR VELOCIDAD RUEDA IZQUERDA
vrep.simxSynchronousTrigger(clientID);  % Trigger final


vrep.simxStopSimulation(clientID,vrep.simx_opmode_blocking); %Parar Simulacion
disp("Comunicacion con SIMULINK finalizada"); %Mostrar mensaje


t = 0:Tm:(i-1)*Tm;
plot(app.UIAxes_seguimiento,xref,yref,xout,yout);
legend(app.UIAxes_seguimiento,"Referencia","Respuesta del Pioneer P3DX");


Uc = Ji*exp;
```

```
plot(app.UIAxes_error,t,xerror,t,yerror);
legend(app.UIAxes_error,"Error en X","Error en Y");
plot(app.UIAxes_signalctrl,t,u,t,w);
legend(app.UIAxes_signalctrl,"Señal de Control u","Señal de Control w");


eTs = (xerror.^2)+(yerror.^2);
ISE = trapz(eTs);


ISCOu = trapz(u.^2);
ISCOw = trapz(w.^2);


app.ISEEditField_3.Value = ISE;
app.ISCOuEditField.Value = ISCOu;
app.ISCOwEditField_2.Value = ISCOw;
% t = 0:Tm:(i-1)*Tm;
% plot(xref,yref);
% hold
% plot(xout,yout);
save('cc1.mat','t','xref','yref','xout','yout');
end
vrep.simxFinish(-1);
vrep.delete();
```

ANEXO VI

Manual de Usuario

Este documento fue pensado para ayudar con el manejo de la interfaz Gráfica diseñada para el presente trabajo de titulación:" Implementación y comparación de un controlador tipo cascada enfocado al seguimiento de trayectorias con un lazo externo basado en un postulado de Lyapunov y un lazo interno basado tanto en un PID convencional como en un controlador por modos deslizantes que emplea el esquema de predictor de Smith, para una plataforma robótica móvil Pioneer 3dx." Este manual presentará la metodología a seguir para manejar de forma eficiente las ventanas del GUI y poder simular el programa correspondiente en CoppeliaSim.

A1.1 SIMULACIÓN COPPELIASIM

Para poder obtener datos y visualizar el seguimiento es necesario tener CoppeliaSim funcionando y con la capacidad de comunicarse con MATLAB es necesario seguir ciertos pasos:

1. Primero es descargar e Instalar CoppeliaSim, según el sistema operativo (Ubuntu, macOS o Windows).
2. Una vez instalado el programa se dirige a la carpeta de instalación, generalmente se encuentra en la carpeta '*Program files'* con el nombre de *'CoppeliaRobotics'.*
3. Dentro de *CoppeliaRobotics* se debe entrar en las siguientes carpetas *'Programming'* → *'RemoteApiBindings'* → *'MATLAB'.* Una vez dentro de esta carpeta se copia todos los archivos que se ven en la FiguraA1.1.

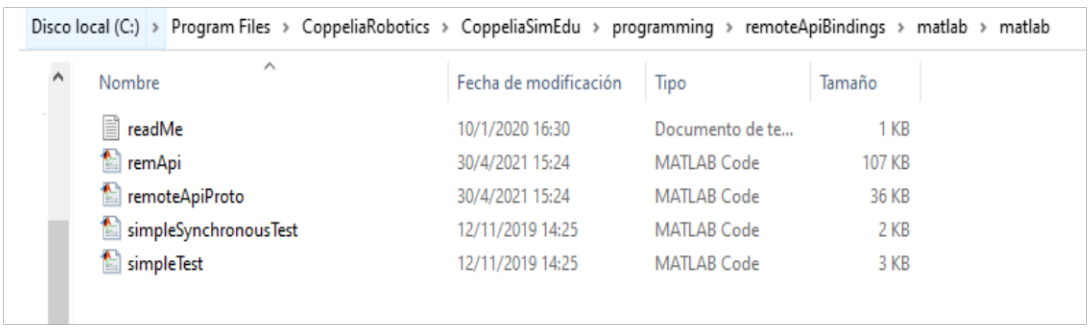


Figura A1.1 Archivos necesarios para comunicar CoppeliaSim parte 1

4. Además de los archivos de la FiguraA1.1 se necesita otro archivo que se encuentra en la carpeta *'Programming'* → *'RemoteApiBindings'* → *'lib'* → *'lib'* dentro de esta carpeta se encuentran los distintos tipos de sistemas operativo, se elige dependiendo el que se esté usando y dentro de la carpeta elegida se encuentra un archivo '.dll' que también es necesario, esto se puede ver en la FiguraA1.2.

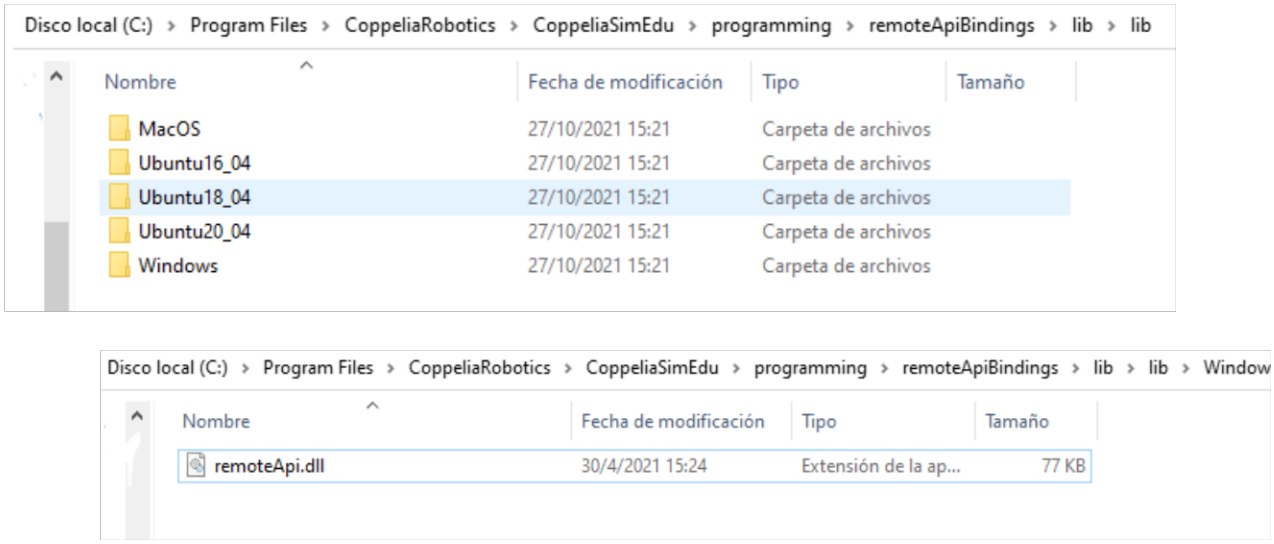


Figura A1.2 Archivos necesarios para comunicar CoppeliaSim parte 2

5. Los archivos tanto de la FiguraA1.1 como de la FiguraA1.2 se copian dentro de una carpeta, esta carpeta debe tener el proyecto de Coppelia que se requiera simular y el archivo de código de MATLAB con el programa.

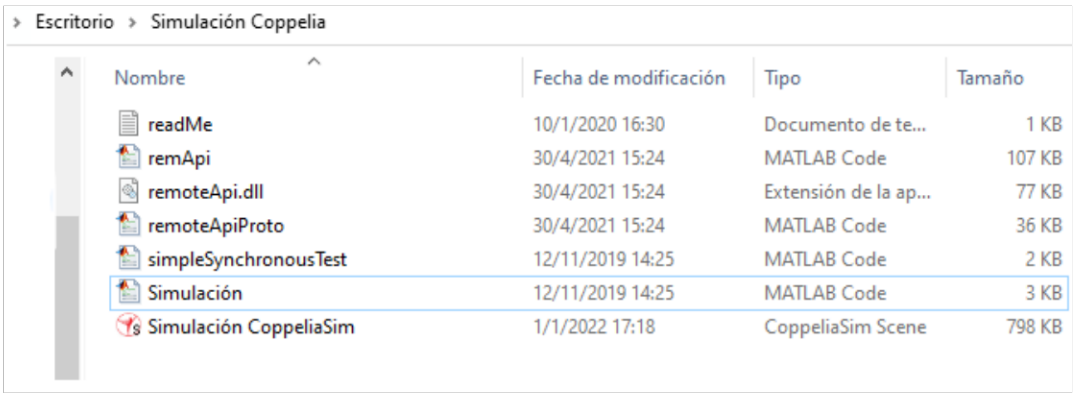


Figura A1.3 Archivos necesarios para comunicar CoppeliaSim

6. Se inicia el archivo de CoppeliaSim que se requiera simular y dentro de un nuevo proyecto se sigue la siguiente secuencia: ‘Add’ → ‘PrimitiveShape’ → ‘Cuboid’, esto creará un cubo que se usará como ayuda para definir el puerto de comunicación, a continuación, se crea un ‘child script’ dentro del cubo como se ve en la FiguraA1.4, el primer menú se despliega con click derecho sobre el objeto ‘Cuboid’.

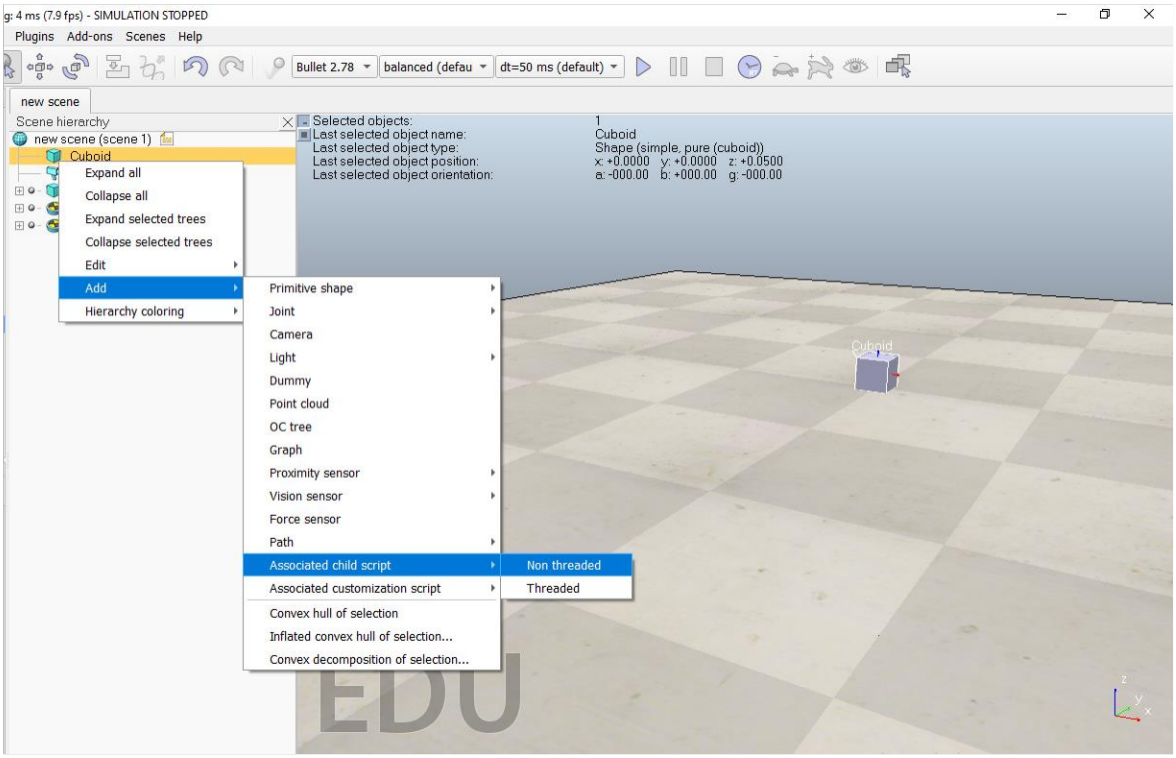


Figura A1.4 Creación de un script para la comunicación

7. Dentro del ‘child script’ del cuboide se tiene ciertas funciones que no son de importancia, lo importante es definir la función

simRemoteApi.start(19999, 1300, false, true)

Dentro de la función de inicialización como en la FiguraA1.5.

Esto permitirá establecer la comunicación desde el lado de CoppeliaSim

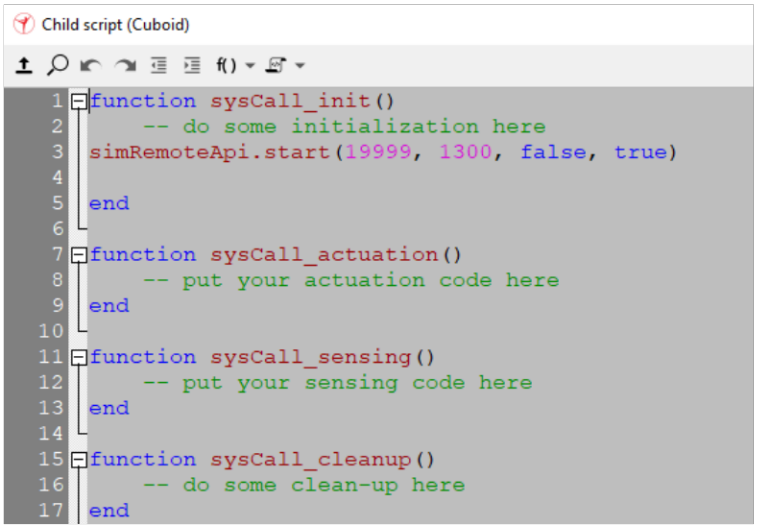


Figura A1.5 Creación de un script para la comunicación