ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA 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 I

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

JUAN DIEGO ZAMBRANO TORRES

DIRECTOR: Dr. PAULO CÉSAR LEICA ARTEAGA

Distrito Metropolitano de Quito, febrero 2022.

CERTIFICACIONES

Yo, Juan Diego Zambrano Torres 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. Juan Diego Zambrano Torres.

Certifico que el presente trabajo de integración curricular fue desarrollado por el Sr. Juan Diego Zambrano Torres, 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 los productos resultantes 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. Juan Diego Zambrano Torres - Estudiante

Dr. Paulo César Leica Arteaga - Director

Sr. Javier Bolaños de la Torre - Colaborador

DEDICATORIA

Ante la tumba recién abierta de mi bisabuela

Olguita María Ordoñez Lucero, mujer luchadora y

perseverante, este trabajo lo dedico a su memoria.

AGRADECIMIENTO

Me considero afortunado de haber sido parte de la Escuela Politécnica Nacional (EPN) y del Departamento de Automatización y Control Industrial durante mi pregrado, agradezco a tan noble institución por permitirme participar en la creación de nuevas teorías y tecnologías que expanden los límites de la ciencia e ingeniería.

Quisiera ofrecer mis más sinceros agradecimientos a mis padres, por su sacrificio, apoyo e inspiración. Su guía y enseñanza me ha brindado la sabiduría necesaria para superar todos los obstáculos. A mis hermanas, quienes me han brindado su compañía y apoyo durante toda mi vida. También a mi tío Wilson, quien ha sabido brindarme su apoyo y desafiarme a descubrir todo lo que la educación y la vida puede ofrecer.

He sido afortunado de contar con el apoyo del Dr. Paulo Leica, su dirección, conocimientos y orientación han sido invaluables para lograr los objetivos establecidos en este trabajo y su entusiasmo ha convertido todo este tiempo en un agradable viaje.

Agradezco también al Dr. Oscar Camacho, quién hace cinco años me convenció del potencial que tiene el control y la automatización para mejorar la vida de las personas y desde entonces, su apertura a nuevas ideas me ha confirmado repetidamente que fue una de las mejores decisiones que he tomado en mi vida.

A Javier, por ser el mejor amigo y colaborador durante este proyecto. Por sus ganas de trabajar y su desarrollo en la simulación de los resultados obtenidos en este trabajo. A Luis y Joe por su apoyo y compañía, gracias muchachos por su amistad.

A todos los docentes del Departamento: Dr. Andrés Rosales, Dra. Jackeline Abad, Dr. Patricio Cruz, MSc. Yadira Bravo, Dra. Silvana Gamboa, Dr. Geovanny Chávez, MSc. María Trujillo, Dr. Marcelo Pozo, MBA. Ana Rodas, MSc. Nelson Sotomayor y demás profesores, quienes han sabido transmitirme su conocimiento y experiencias para el desarrollo de este trabajo. Además, me han permitido conocer el mundo de la docencia y su impacto en la vida de las personas. Por último, un agradecimiento a los Honorables miembros del Consejo Politécnico, a la rectora, Dra. Florinella Muñoz quien lo preside, al personal docente y administrativo de la EPN y sobre todo a los estudiantes, por su apoyo y por haberme permitido ayudar en el mejoramiento de nuestra Alma mater.

Gracias infinitas
Juan Diego

ÍNDICE DE CONTENIDO

TOMO I

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA……..III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO..... V
RESUMEN..... VIII
ABSTRACT..... IX

1.INTRODUCCIÓN……1
1.1 Objetivo general.....2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco teórico 3
1.4.1 Introducción a la robótica móvil…….…….. 3
1.4.2 Odometría……..…….. 3
1.4.3 Plataforma robótica móvil Pioneer 3DX..……..…….. 4
1.4.4 Modelo cinemático del robot móvil..……..…….. 5
1.4.4.1 Restricción no holonómica……..……..5
1.4.4.2 Modelo cinemático de un robot de tracción diferencial..…..... 6
1.4.4.3 Modelo uniciclo de un robot móvil…..……. 7
1.4.4.4 Modelo uniciclo con restricción no holonómica mejorada……. 7
1.4.5 Modelo dinámico del robot móvil Pioneer 3DX..……..…….. 8
1.4.5.1 Consideraciones mecánicas y eléctricas……...9
1.4.5.2 Modelo dinámico del sistema…….…… 10
1.4.6 Métodos de Discretización.....11
1.4.6.1 Discretización por Tustin…….…….. 13
1.4.7 Software de simulación CoppeliaSim .....……...……..…….15
1.4.8 Tipos de Comunicación “remote API”.....16
1.4.8.1 Comunicación Asincrónica.....16

1.4.8.2 Comunicación Sincrónica 17
1.4.9 Software de Simulación MATLAB-SIMULINK 18
1.4.9.1 Funciones remApi en Matlab 19
1.4.9.2 AppDesigner 21
1.4.10 Condiciones de estabilidad según Lyapunov 21
1.4.11 Control por modos deslizantes (SMC) 22
1.4.12 Predictor de Smith 24
1.4.13 Índices de Desempeño 24
1.4.13.1 ISE 25
1.4.13.2 ISCO 25

2 METODOLOGÍA 26
2.1 Identificación y validación 26
2.1.1 Identificación de la velocidad lineal en el robot de CoppeliaSim 26
2.1.2 Identificación de la velocidad angular en el robot de CoppeliaSim 29
2.1.3 Validación de los modelos 31
2.2 Topología de los controladores 31
2.2.1 Topología controlador en lazo simple 32
2.2.2 Topología controlador en cascada 33
2.3 Diseño del controlador basado en un postulado de Lyapunov 34
2.4 Diseño de controladores PI para el lazo interno de velocidades 36
2.5 Diseño de controladores SMC+SP 39
2.5.1 Diseño parte continua SMC 39
2.5.2 Diseño parte discontinua SMC 41
2.5.3 Acción de control para velocidad lineal 42
2.5.4 Acción de control para velocidad angular 42
2.5.5 Esquema del Predictor de Smith 43
2.6 Discretización de Controladores 43
2.6.1 Tiempo de muestreo 44
2.6.2 Control Lazo Simple Lyapunov 45
2.6.3 Control cascada: Lazo interno PI 47
2.6.4 Control cascada: Lazo interno SMC + SP 48

2.6.4.1 Discretización modelo de primer orden V. lineal.....51
2.6.4.2 Discretización modelo de primer orden V. angular 52
2.6.4.3 Consideraciones de tiempo muerto.....53

3 CONCLUSIONES Y RECOMENDACIONES……55
3.1 Conclusiones……..……..55
3.2 Recomendaciones……..……56

4 REFERENCIAS BIBLIOGRÁFICAS TOMO I.....57

5 ANEXOS……59
ANEXO I……..……...59
ANEXO II…… ……..……62

RESUMEN

El presente trabajo de integración curricular presenta el diseño de un controlador tipo cascada con un lazo externo basado en un postulado de Lyapunov, mientras que en el lazo interno se propone un Controlador por Modos Deslizantes (SMC) basado en un esquema tipo Predictor de Smith (SP), esto con el fin de obtener los beneficios de robustez y alcanzabilidad del SMC y la compensación del retardo que ofrece el SP, pues se evidenció un tiempo de subida lento y un tiempo muerto significativo en la curva de respuesta de la velocidad. Además, se implementa un lazo interno tipo PI, con el fin de realizar la comparativa de los beneficios en el desempeño.

Se pretende que este trabajo sirva de guía para la implementación en cualquier plataforma real o software de simulación, pues las leyes de control y sus respectivas ecuaciones en diferencia son obtenidas en este documento luego de un proceso de diseño bien fundamentado. El diseño está orientado a la aplicación en CoppeliaSim y SIMULINK que se lo realizará en el Tomo II, pues la curva de reacción responderá a las características constructivas y el entorno de dicho simulador.

En este tomo se realiza una revisión bibliográfica de los tópicos necesarios para el desarrollo de este trabajo, se diseña los controladores propuestos en tiempo continuo y se discretiza las leyes de control obtenidas, con el fin de obtener expresiones que sirvan como punto de partida para la implementación de los controladores en el Tomo II.

PALABRAS CLAVE: Control por modos deslizantes, Lyapunov, Predictor de Smith, Pioneer 3DX, Coppelia Sim.

ABSTRACT

The present work presents the design of a cascade controller with an external loop based on a Lyapunov function, meanwhile in the internal loop, a Sliding Mode Controller (SMC) is proposed, based on a Smith Predictor (SP) scheme, to obtain the benefits of robustness and achievability of the SMC and the delay compensation offered by the SP, because a slow rise time and a significant dead time were evidenced in the speed response. Also, a PID controller is implemented in the internal loop, in order to compare the performance benefits of the first controller.

This work is intended to serve as a guide for implementation in any real platform or simulation software. Since the control laws and their respective difference equations are obtained after a well-founded design process. The design is oriented to its application in CoppeliaSim, and SIMULINK developed on the Volume II of this work, due to the reaction curve will respond to the construction characteristics and the environment of this simulator.

This Volume is oriented to carry out a bibliographic review of the necessary topics for the development of this work, design the proposed controllers in a continuous time, then the control laws are discretized in order to serve as a starting point for the implementation in Volume II.

KEYWORDS: Sliding Mode Controller, Lyapunov, Smith Predictor, Pioneer 3DX, Coppelia Sim.

1 INTRODUCCIÓN

El seguimiento de trayectoria en robots móviles es un campo que está en expansión y ha sido hasta hoy en día, un área activa para la investigación académica y aplicaciones industriales. Sus aplicaciones se centran en soporte y cooperación con humanos en actividades como transporte de objetos en oficinas, hospitales, hoteles, restaurantes y actividades domésticas como se muestra en la figura 1.1. Por otro lado, en aplicaciones militares han sido utilizados para exploración en ambientes hostiles y de difícil acceso [1].

Figura 1.1. Aplicaciones de soporte y cooperación con robótica móvil

Los controladores de seguimiento de trayectoria que típicamente se han venido implementando en el robot móvil Pioneer 3DX son basados en métodos numéricos (Euler o Trapezoidal) [2], control PI y controladores por modos deslizantes convencionales basados en superficies PD-Like [3]. Una de las problemáticas presentes en el control de trayectoria es el retardo en la respuesta, debido a la comunicación y el modelo dinámico propio del robot, lo que genera errores en el seguimiento de la referencia e inclusive, dependiendo de la magnitud del retardo, se puede llevar el sistema a la inestabilidad. En algunas aplicaciones, como el transporte de cargas, se requieren respuestas suaves y rápidas de parte del robot, pues un cambio brusco o con vibraciones comprometería la integridad de la carga y el seguimiento adecuado de la trayectoria. El problema radica entonces en buscar un equilibrio de rapidez, estabilidad, robustez y precisión.

Este documento consiste en un análisis teórico sobre los fundamentos de robótica móvil, modelamiento de robots de tracción diferencial y técnicas de control avanzadas. Luego, se busca establecer una metodología de identificación y diseño para los controladores propuestos orientados al modelo del robot móvil Pioneer 3DX.

En los siguientes capítulos se diseña un controlador tipo cascada con un lazo externo que permita la obtención de la referencia de velocidad para el lazo interno, el cual se basa en un esquema de Predictor de Smith (SP) con naturaleza deslizante (SMC) orientado a la implementación en el Software CoppeliaSim. Dicho software es una plataforma que permite establecer eventos con diferentes robots considerando sus características constructivas y su dinámica de forma sincrónica con MATLAB, para evitar problemas de tiempo en la comunicación [4].

Se busca entonces, obtener una ley de control eficiente que compense los retardos y mejore el desempeño, a la vez que conserve robustez ante perturbaciones en el robot móvil Pioneer 3DX, como punto de partida para la implementación en sistemas embebidos o simuladores, actividad a desarrollarse en el Tomo II de este trabajo.

1.1 OBJETIVO GENERAL

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

1.2 OBJETIVOS ESPECÍFICOS

- Estudiar y determinar el modelo cinemático y dinámico de un robot móvil tipo uniciclo.
- Diseñar e implementar controladores tipo PID y SMC con predictor de Smith en un esquema cascada para una plataforma robótica móvil.

1.3 ALCANCE

- Se realizará una revisión bibliográfica y se implementará el modelo cinemático de un robot uniciclo.
- Se realizará una revisión bibliográfica y se implementará el modelo dinámico del robot tipo uniciclo Pioneer 3DX.
- Se realizará una revisión bibliográfica, se diseñará y se implementará un controlador para el lazo externo del esquema cascada usando un controlador basado en Lyapunov.
- Se realizará una revisión bibliográfica, se diseñará y se implementará un primer controlador para el lazo interno del esquema cascada usando un esquema tipo PID.
- Se realizará una revisión bibliográfica, se diseñará y se implementará un segundo controlador para el lazo interno tipo SMC usando un esquema de predictor de Smith.

1.4 MARCO TEÓRICO

En el siguiente apartado, se pretende dar una introducción a la robótica móvil, la definición de algunos términos relacionados, el modelamiento cinemático de un robot de tracción diferencial con restricción no holonómica, el modelo dinámico orientado al robot Pioneer 3DX y algunos aspectos básicos sobre las técnicas de control y discretización utilizadas en este trabajo con sus características y ventajas asociadas.

1.4.1 INTRODUCCIÓN A LA ROBÓTICA MÓVIL.

Un robot móvil se define como cualquier sistema capaz de desplazarse por sí mismo a través de un ambiente cualquiera, mediante el control de su trayectoria y el uso de sensores para la identificación de su ambiente. Es decir, un robot móvil puede ser aquel sistema que posee ruedas, piernas o incluso aquel que vuele o nade como se muestra en la figura 1.2

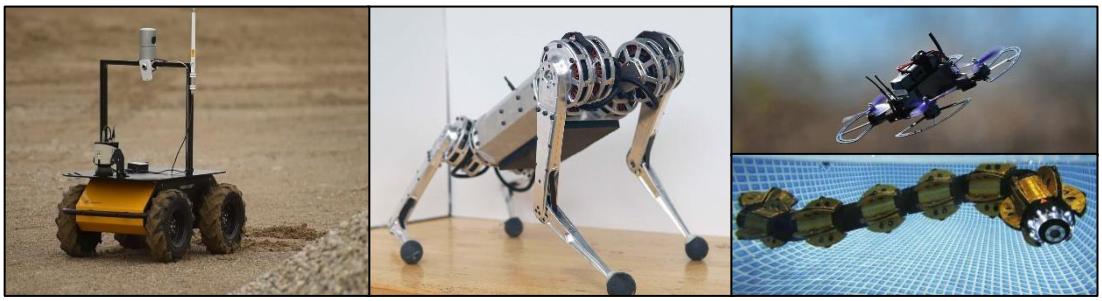


Figura 1.2. Robots móviles.

Un robot móvil debe conocer su posición, hacia donde va y como lo conseguirá. Para ello, debe tomar medidas, modelar el ambiente y planear un camino y estrategias para alcanzar su objetivo [5]. El sensado de las condiciones internas como su orientación, velocidad en los motores, estado de la batería, etc. se conocen como habilidades propioceptivas, mientras que el sensado de sus condiciones externas como cercanía de obstáculos, temperatura, humedad, etc. se conoce como habilidades exteroceptivas.

1.4.2 ODOMETRÍA

La Odometría se encarga de obtener la posición relativa de un robot respecto a un punto de referencia, mediante el sensado y estimaciones matemáticas [6]. En este trabajo, la obtención de su posición y velocidad se logrará mediante comandos API respecto a la referencia establecida en Coppelia Sim y se detallará en la metodología.

1.4.3 PLATAFORMA ROBÓTICA MÓVIL PIONEER 3DX

El robot Pioneer 3DX es un robot de tracción diferencial como se muestra en el diagrama de la figura 1.3, en este diagrama también se distingue la rueda loca (Caster Wheel), la cual tiene la finalidad de soporte y rotación pasiva, permitiendo un movimiento firme basado en las decisiones de las dos ruedas controladas.

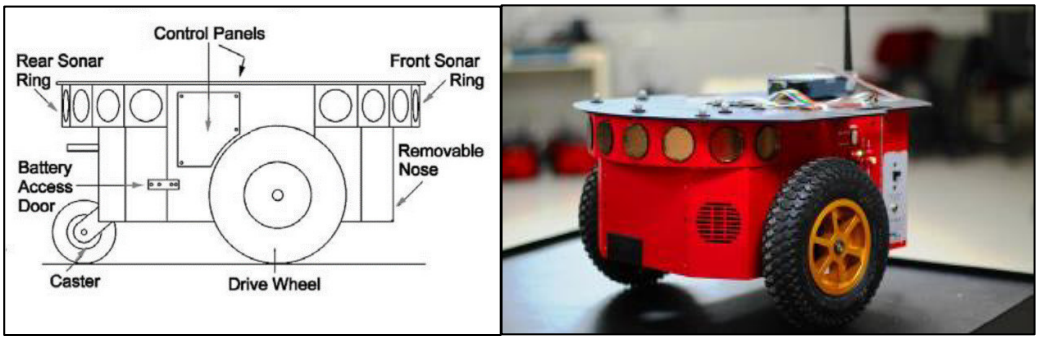


Figura 1.3. Diagrama Pioneer 3DX.

El robot en su modelo base posee algunos sensores y actuadores, listados en la Tabla 1.1:

Tabla 1.1. Sensores y actuadores del Pioneer 3DX

Descripción	Sensor	Actuador	Elementos
Sensores ultrasónicos frontales	X		8
Sensores ultrasónicos traseros	X		8
Encoders 76.600 cont/rev	X		2
Motores DC reversibles		X	2
Driver para motores		X	2

El robot también posee diferentes expansiones que añaden sensores y actuadores, lo que significa un aumento en sus prestaciones tales como manipulación de objetos, visión y mapeo e incluso como un parlante móvil tal como se muestra en la figura 1.4. [7]:

Figura 1.4. Diferentes prestaciones del robot Pioneer 3DX.

1.4.4 MODELO CINEMÁTICO DEL ROBOT MÓVIL

El modelo cinemático de un robot móvil de tracción diferencial se encuentra bien definido por la descripción de su trayectoria, por elementos tales como posición, velocidad y aceleración, considerando al robot como una partícula, sin tomar en cuenta efectos de fuerzas asociadas, momentos, inercia, trabajo y otras propiedades dinámicas propias del entorno y las características constructivas del robot.

1.4.4.1 Restricción no holonómica

Los robots móviles a ruedas como el Pioneer 3DX tienen la peculiaridad de poseer una restricción no holonómica, esto debido a que sus ecuaciones asociadas describen movimientos rotacionales, pero no todas las posibles restricciones traslacionales.

No existe forma en que el robot realice un movimiento únicamente lateral, como se muestra en la figura 1.5:

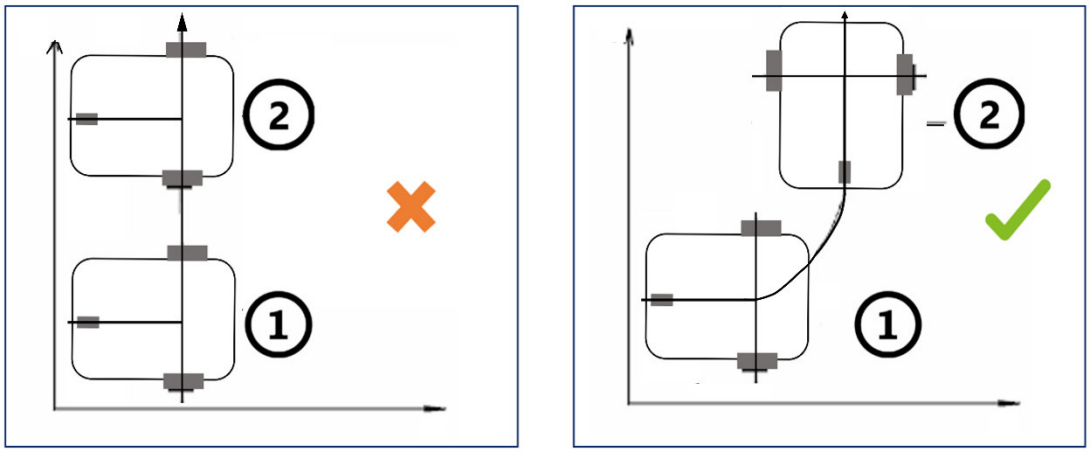


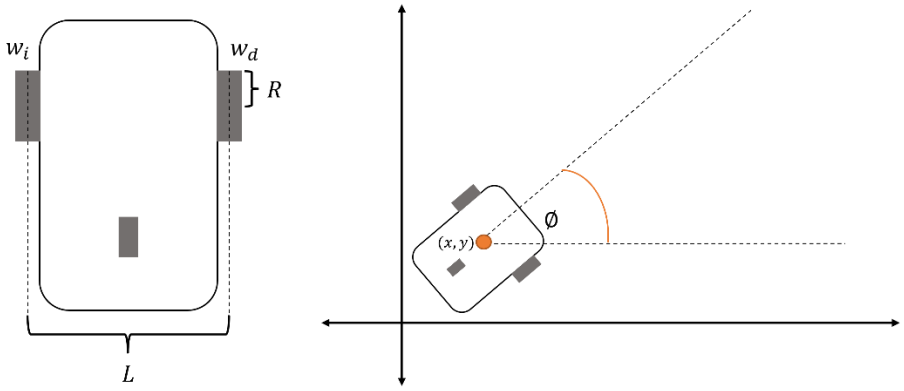
Figura 1.5 Restricción no holonómica en robots móviles.

La solución que se propone ente este problema es colocar el punto de interés a una distancia "a" del centro definido por el eje que une las dos ruedas. Esta condición se la conoce como "Restricción no holonómica mejorada" y presenta algunas ventajas de control, pues permite llevar a cabo la acción con anticipación y evita tomar decisiones que no pueden llevarse a cabo debido a la restricción antes mencionada [8].

Una vez definida la restricción no holonómica se puede proponer el modelo cinemático y algunas expresiones útiles en el desarrollo de este trabajo en los siguientes apartados.

1.4.4.2 Modelo cinemático de un robot de tracción diferencial

En la figura 1.6 se muestra un robot de tracción diferencial, el cual tiene ruedas de un radio R, una separación entre ellas de longitud L y las respectivas velocidades asociadas a cada una de las ruedas w_d y w_i. Además, se encuentran en un plano XY que define su posición y un ángulo ∅ que define su orientación:



a) b)

Figura 1.6 Parámetros de posición y velocidad del robot diferencial.

La velocidad traslacional - angular (w_t) depende de una expresión basada en el promedio de las velocidades angulares de cada rueda (w_d y w_i), un claro ejemplo es cuando una de ellas tiene velocidad nula, entonces se traslada a la mitad de velocidad que cuando dos de ellas se mueven. Por lo tanto, se puede definir como:

$$w_t = \frac{w_d + w_i}{2} \tag{1.1}$$

Entonces la velocidad lineal (v) puede definirse en base al radio de la rueda (R) como:

$$v = R\ \frac{w_d + w_i}{2} \tag{1.2}$$

Por otro lado, la velocidad rotacional angular (w) de todo el robot crece linealmente con el radio de la rueda y se reduce linealmente por la distancia entre ambas (L) [6], todo esto al cambio entre las velocidades, es decir su diferencia, por lo que:

$$w = \frac{R}{L}(w_d - w_i) \tag{1.3}$$

Mediante el análisis cinemático de la figura 1.6 b), y la descomposición de la velocidad lineal en su componente horizontal y vertical se puede obtener las expresiones de las derivadas asociadas a las variables de posición ($\dot{X}$, $\dot{Y}$, $\dot{\emptyset}$), obteniendo las siguientes ecuaciones diferenciales [9]:

$$\dot{X} = \frac{R}{2}\,(w_d + w_i)\ \cos(\emptyset) \tag{1.4}$$

$$\dot{Y} = \frac{R}{2}\,(w_d + w_i)\ \sin(\emptyset) \tag{1.5}$$

$$\dot{\emptyset} = \frac{R}{L}\,(w_d - w_i) \tag{1.6}$$

1.4.4.3 Modelo uniciclo de un robot móvil

Cuando se trabaja en robótica móvil, es mejor simplificar las señales de control, de tal forma que las expresiones (ecuaciones diferenciales) sean mucho más sencillas y faciliten el diseño de los controladores. Así entonces, el modelo uniciclo de un robot móvil puede definirse por su velocidad lineal y su velocidad angular, reemplazando la ecuación 1.2 y 1.3 en las ecuaciones 1.4, 1.5 y 1.6 se tiene [9]:

$$\dot{X} = v\ \cos(\emptyset) \tag{1.7}$$

$$\dot{Y} = v\ \sin(\emptyset) \tag{1.8}$$

$$\dot{\emptyset} = w \tag{1.9}$$

En la figura 1.7 se puede observar las entradas y salidas de este modelo:

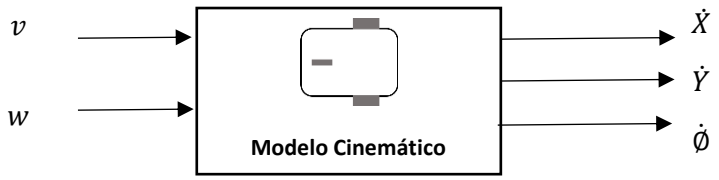


Figura 1.7 Entradas y salidas en el modelo uniciclo.

Nótese que las velocidades y posiciones de salida pueden ser obtenidas mediante el sensado y odometría, para ser utilizadas adecuadamente como realimentación para nuestro lazo de control.

1.4.4.4 Modelo uniciclo con restricción no holonómica mejorada

Como se vio en el subcapítulo 1.4.4.1 referente a la restricción no holonómica, los robots de este tipo deben tener un punto de referencia alejado (P_c) a una distancia (a) del eje de

acción de las ruedas donde se encuentra el punto original (P_o), así entonces analicemos el diagrama de la figura 1.8:

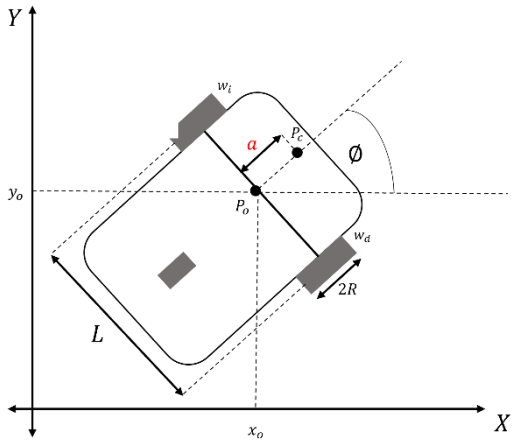


Figura 1.8 Diagrama de robot con restricción no holonómica mejorada.

Tomando en cuenta que esta distancia permite generar movimientos traslacionales en base a la velocidad angular de manera anticipada, se puede definir nuevas ecuaciones para el modelo cinemático [8], donde:

$$\dot{X} = v\cos(\emptyset) - a\,w\,\sin(\emptyset) \tag{1.10}$$

$$\dot{Y} = v\,\sin(\emptyset) + a\,w\,cos(\emptyset) \tag{1.11}$$

$$\dot{\emptyset} = w \tag{1.12}$$

Las ecuaciones anteriores se pueden expresar de forma matricial como se muestra en la ecuación 1.13, esta expresión la necesitaremos al momento de diseñar el controlador del lazo externo donde se distingue el Jacobiano del modelo como [8]:

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\emptyset}\end{bmatrix} = \begin{bmatrix}\cos(\emptyset) & -a\sin(\emptyset)\\ \sin(\emptyset) & a\cos(\emptyset)\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ w\end{bmatrix} \tag{1.13}$$

1.4.5 MODELO DINÁMICO DEL ROBOT MÓVIL PIONEER 3DX

El modelo dinámico es un conjunto de ecuaciones matemáticas que describen el comportamiento físico real de un cuerpo en específico, en este caso el robot Pionner 3DX. Los parámetros constructivos y el entorno del robot juegan un papel fundamental en este modelo pues se considera todos los aspectos de mecánica clásica.

1.4.5.1 Consideraciones mecánicas y eléctricas

De la Cruz y Carelli [10] en su trabajo se enfocan en el modelamiento de un robot móvil de dos ruedas activas con rueda loca, este modelo incluye un controlador PD que se encarga de traducir las referencias de velocidad lineal y angular en voltaje para los motores de las ruedas activas, teniendo esto en cuenta, se propone el siguiente diagrama de cuerpo libre:

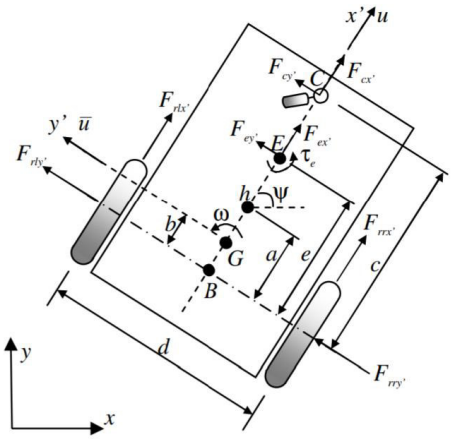


Figura 1.9 Diagrama de cuerpo libre de un robot de tracción diferencial.

Donde, G es el centro de masa, B es el punto medio en el eje de las ruedas, h es un vector con las coordenadas para el seguimiento, $u\ y\ \bar{u}$ son las velocidades longitudinales y laterales del centro de masa, w y $\emptyset$ son la velocidad angular y el ángulo de orientación respectivamente, a, b, c, d y e son las distancias constructivas propias de cada robot, F_{rrx} , F_{rry}, F_{rlx} , F_{rly} , F_{cx} , F_{cy}, F_{ex} , F_{ey} son las fuerzas longitudinales y laterales de las 2 ruedas (derecha e izquierda), de la rueda loca (Castor Wheel) y las realizadas en E, donde realmente ocurre la rotación respectivamente. Además, τ_e es el momento realizado en este punto (E) e I_z es el momento de inercia sobre el eje vertical localizado en el centro de masa.

En base al diagrama de cuerpo libre de la figura 1.9 y aplicando la segunda ley de Newton, que relaciona la fuerza y aceleración se tiene:

$$\sum\nolimits_{FX} = F_{rrx} + F_{rlx} + \ F_{cx} + F_{ex}$$
$$\sum\nolimits_{FY} = F_{rry} + F_{rly} + \ F_{cy} + F_{ey} \qquad (1.14)$$

El momento entonces puede ser descrito como:

$$\sum_M = \ I_z\dot{w} = \frac{d}{2}\ (F_{rrx} - F_{rlx}) - b\big(F_{rly} - F_{rry}\big) + (e-b)F_{ey} + (c-b)F_{cy} + \ \tau_e \qquad (1.15)$$

Por otro lado, Zhang en su trabajo [11] propone las siguientes expresiones para la velocidad en un robot móvil de tracción diferencial cuando se considera deslizamiento:

$$u = \frac{1}{2}\left[r(w_r + w_l) + (u_r + u_l)\right]$$

$$w = \frac{1}{d}\left[r(w_r - w_l) + (u_r - u_l)\right] \tag{1.16}$$

$$\bar{u} = \frac{b}{d}\left[r(w_r - w_l) + (u_r - u_l)\right] + \bar{u}^s$$

Donde r es el radio de las ruedas, $w_r\ y\ w_l$ son las velocidades angulares de cada llanta, $u_r\ y\ u_l$ son las velocidades lineales de deslizamiento y $\bar{u}^s$ es la velocidad de deslizamiento lateral de las ruedas [12].

Con respecto a la parte eléctrica, los torques generados en los motores de las ruedas ($\tau_r\ y\ \tau_l$) pueden ser deducidos de manera más sencilla si se desprecia el voltaje en las inductancias, donde k_a es la constante de torque, k_b es la constante de voltaje, R_a es la resistencia eléctrica de la armadura y $V_r\ y\ V_l$ son los voltajes que se suministran a las ruedas. Teniendo las siguientes expresiones:

$$\tau_r = \frac{k_a\,(V_r - k_b w_r)}{R_a} \tag{1.17}$$

$$\tau_l = \frac{k_a\,(V_l - k_b w_l)}{R_a}$$

Los robots comerciales, como el Pioneer 3DX poseen controladores internos, en este caso un P-D, con el fin de traducir la referencia de velocidades ($u_{ref} y\ w_{ref}$) (entradas) en voltaje para los motores de las ruedas, por lo tanto, se propone la siguiente ecuación [12]:

$$\begin{bmatrix} V_u \\ V_w \end{bmatrix} = \begin{bmatrix} K_{PT}\left(u_{ref} - u\right) - K_{DT}\,\dot{u} \\ K_{PR}\left(w_{ref} - w\right) - K_{DR}\,\dot{w} \end{bmatrix} \tag{1.18}$$

1.4.5.2 Modelo dinámico del sistema

Así entonces, unificando y desarrollando las ecuaciones 1.14, 1.15, 1.16, 1.17 y 1.18 se obtiene el modelo dinámico descrito por la siguiente ecuación [12]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \\ \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} u\cos(\emptyset) - a\,w\sin(\emptyset) \\ u\sin(\emptyset) + a\,w\cos(\emptyset) \\ w \\ \frac{\theta_3}{\theta_1}w^2 - \frac{\theta_4}{\theta_1}u \\ -\frac{\theta_5}{\theta_2}uw - \frac{\theta_6}{\theta_2}w \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix} \begin{bmatrix} u_{ref} \\ w_{ref} \end{bmatrix} \tag{1.19}$$

El vector θ son constantes definidas por los parámetros constructivos, parámetros de sintonización y distancias definidas del robot. Rosales y Scaglia [12] en su trabajo determinan el vector para el robot Pioneer 3DX en base a pruebas experimentales, obteniendo los valores mostrados en la tabla 1.2:

Tabla 1.2 Constantes del modelo dinámico para el robot móvil Pioneer 3DX

	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$
Valor	0,24089	0,2424	-0,000936	0,99629	-0,003725	0,8915

Nótese que se utilizará un modelo dinámico sin incertidumbres, pues esta información no tiene efecto relevante en el simulador al que se encuentra orientado este trabajo.

1.4.6 MÉTODOS DE DISCRETIZACIÓN

Dentro de la teoría de control es muy común el uso de funciones de transferencia (FT) que se generan a partir de la ecuación diferencial de cierto modelo en tiempo continuo para luego usar la transformada de Laplace y llevarla a la variable compleja s, obteniendo una nueva función $F(s)$, de esta forma, el desarrollo de los modelos se simplifica ya que los mismos se pueden expresar en forma de polinomios.

Ahora, para poder digitalizar este tipo de modelos y trabajarlos dentro de un computador es necesario muestrear los valores que puedan representar la curva de forma fiel a la realidad, esto debido a la complejidad y carga computacional que tomaría el obtener todos los valores en tiempo continuo de una señal, además de ser una tarea imposible.

Debido a esto es necesaria la discretización y al hablar de discretizar una señal es necesario tener en cuenta el tiempo en el que esta se muestrea ya que la mala elección de este tiempo (T_m) puede incurrir en errores como la aparición del 'aliasing', que es la pérdida de información por muestrear de forma "lenta".

Existen varias formas de discretizar una ecuación diferencial y llevarla a una en ecuación en diferencias, así como también pasar de una función de transferencia continua en el dominio s a una en la variable Z (discretizada) y a partir de esta crear la ecuación en diferencias, que es la expresión que permite establecer algoritmos con facilidad.

Dentro de los métodos de discretización se encuentran el Euler backward, Euler forward y Tustin o trapezoidal, cada uno de ellos con su peculiaridad en el mapeo de polos y ceros, como se detalla a continuación:

A continuación, en el siguiente análisis se muestran las gráficas del mapeo según el tipo de discretización, es decir la equivalencia discreta de las zonas en las que se pueden encontrar los polos en el plano s (Continuo) reflejada en el plano z (Discreto). La zona sombreada en el plano s representa el semiplano izquierdo, lugar donde los polos siempre definen una respuesta continua estable, mientras que la zona sombreada en el plano z representa su equivalencia según el método que se utilice, es necesario recordar que en el plano z, los polos son estables solo si se encuentran dentro del círculo unitario:

Euler Forward: Para este método de discretización al realizar la gráfica de mapeo de polos se puede notar que dentro del semiplano izquierdo del plano s en el cuál son repuestas con polos estables, reflejan una equivalencia en el semiplano izquierdo del plano z si se traslada el eje a la unidad, es necesario que los polos se encuentren en un círculo de diámetro $\frac{2}{T_m}$ para reflejar polos estables discretos, por lo que algunos polos estables continuos pueden reflejar polos inestables y encontrarse fuera del círculo unitario en su equivalencia discreta [13].

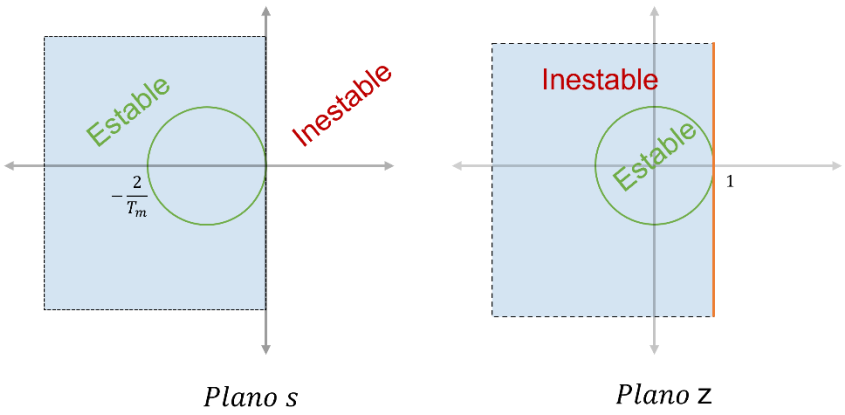


Figura 1.10. Mapeo de polos plano s a plano z - Método Euler forward

Euler backward: Para este método de discretización al revisar la Figura 1.11 la cuál es la gráfica del mapeo de polos, se puede notar que dentro del semiplano izquierdo de s todos los polos discretizados serán estables dentro del círculo pequeño, aunque los polos que se encuentren en el círculo de $(0$ a $+2/T)$ en el plano s y son inestables serán mapeados como estables, por lo que una función continua inestable puede reflejar una función discreta estable [13].

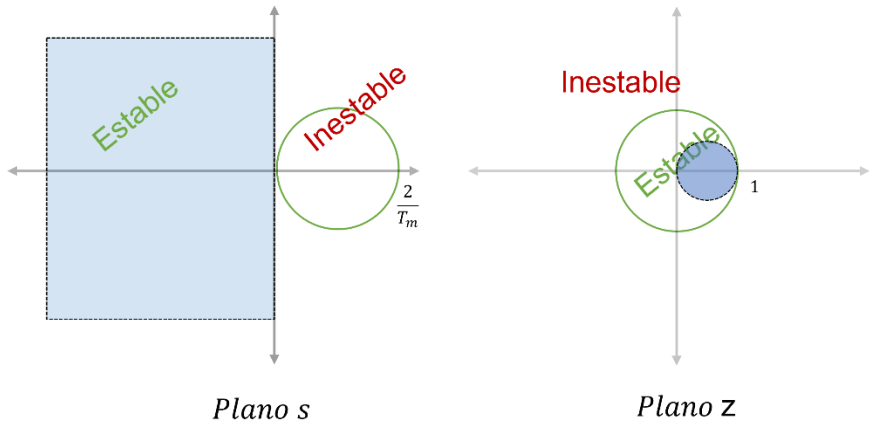


Figura 1.11. Mapeo de polos plano s a plano z - Método Euler backward

Tustin: Para este método de discretización al revisar la Figura 1.12 la cuál es la gráfica del mapeo de polos, se puede notar que únicamente los valores del semiplano izquierdo son estables tanto en el plano s como en el plano z, lo que indica que todos los polos estables mapeados del plano s entregarán polos estables en el plano z. [13]

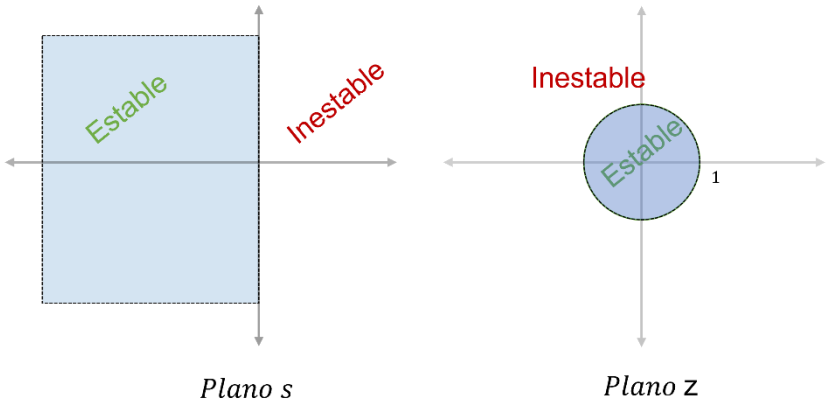


Figura 1.12. Mapeo de polos plano s a plano z – Método de Tustin

1.4.6.1 Discretización por el método de Tustin

Acorde con el análisis realizado anteriormente se procede a definir la discretización mediante el método de Tustin, pues es la mejor opción debido a que preserva su respuesta en frecuencia y la estabilidad de nuestras FT. Este método se basa en la aproximación trapezoidal de la integral que se muestra en la figura 1.13.

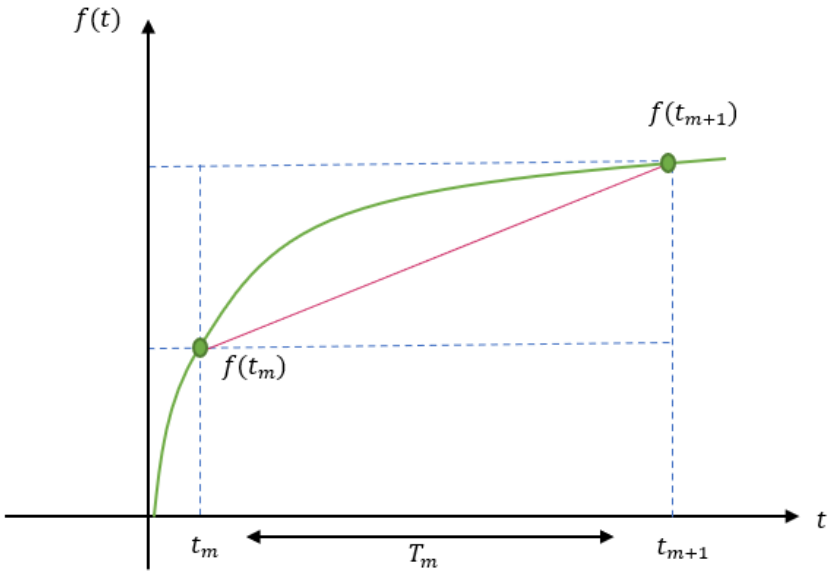


Figura 1.13. Regla del trapecio para definir la integral.

Se define una nueva variable 'y' como la integral de la función $f(t)$, tal que en un intervalo definido se tiene:

$$\Delta y(t) = \int_{t_m}^{t_{m+1}} f(t)\, dt \tag{1.20}$$

Usando la regla del trapecio se obtiene que la integral de la aproximación $u(t)$ es el área del trapecio definido por:

$$\Delta y(t) = \int_{t_m}^{t_{m+1}} u(t)\, dt = (t_{m+1} - t_m)\left(\frac{f(t_{m+1}) + f(t_m)}{2}\right) \tag{1.21}$$

Como se puede ver en la Figura 1.13 ($t_{m+1} - t_m = T_m$), el cuál es el tiempo de muestreo, además la variación en y estará definida en la aproximación por:

$$y(t_{m+1}) - y(t_m) \tag{1.22}$$

Por lo que:

$$y(t_{m+1}) - y(t_m) = T_m\left(\frac{f(t_{m+1}) + f(t_m)}{2}\right) \tag{1.23}$$

Se puede discretizar la ecuación 1.23 para poder encontrar la transformada Z del método de Tustin, tomando como tiempo de muestreo T_m y el tiempo actual k como t_{m+1} , realizando la sustitución:

$$y[k] - y[k-1] = \frac{T_m}{2}(f[k] + f[k-1]) \tag{1.24}$$

Para poder llevar la ecuación 1.24 al plano z es necesario usar ciertas propiedades de la transformada Z, una de estas propiedades es la expresión para $x[k-k_0] \rightarrow z^{-k_0} * X(z)$ por lo que se puede escribir como:

$$Y(z)\,[\,(1-z^{-1})] = \frac{T_m}{2}\,F(z)[\,(1+z^{-1})]$$

$$\frac{Y(z)}{F(z)} = \frac{T_m}{2}\,\frac{1+z^{-1}}{1-z^{-1}} \tag{1.25}$$

Desarrollando la ecuación anterior se tiene:

$$Y(z) = \frac{T_m}{2}\,\frac{z+1}{z-1}\,F(z) \tag{1.26}$$

Recordemos que Y se definió como la integral de F en la ecuación 1.20, por lo que en tiempo continuo se tiene la siguiente equivalencia:

$$Y(s) = \frac{1}{s}\,F(s) \tag{1.27}$$

Al observar las equivalencias en la ecuación 1.26 y 1.27 se tiene que:

$$\frac{1}{s} = \frac{T_m}{2}\,\frac{z+1}{z-1} \tag{1.28}$$

Por lo tanto, el operador s en este método de discretización será:

$$s = \frac{2}{T_m}\,\frac{z-1}{z+1} \tag{1.29}$$

1.4.7 Software de simulación Coppelia Sim

Este trabajo está orientado a una posterior implementación en el Software Coppelia Sim, el cuál es un simulador creado para la aplicación física de un robot, sin disponer de la máquina real. Sus aplicaciones se enfocan al desarrollo de algoritmos, simulaciones de automatización industrial, monitoreo remoto, entre otros. En la figura 1.14 se muestra el prototipo del robot Pioneer 3DX disponible en este entorno de trabajo:

Figura 1.14 Robot Pioneer 3DX en software CoppeliaSim (LOGO)

Es necesario conocer que esta plataforma permite establecer conexión sincrónica con MATLAB, programa donde puede desarrollarse el algoritmo de control y utilizar al robot virtual como planta, obteniendo una salida confiable y muy cercana a la realidad, pues nuestro robot virtual considera todas sus características dinámicas. Los comandos utilizados para dicho propósito son los comandos API que permiten obtener y configurar parámetros en CoppeliaSim [4].

1.4.8 TIPOS DE COMUNICACIÓN "REMOTE API"

El software CoppeliaSim puede comunicarse con otras aplicaciones como ROS o Matlab, en este caso la conexión será entre CoppeliaSim y Matlab mediante el uso de un cliente API remoto, este API remoto es una aplicación que funcionará dentro de otro proceso u otra máquina permitiendo enviar y recibir datos o comandos hacia CoppeliaSim. Dentro del modo de operación en el que puede trabajar el API remoto están la operación asincrónica y sincrónica.

1.4.8.1 Comunicación Asincrónica "remote API"

El primer modo de operación es el modo asincrónico, su principal característica es que al enviar o recibir datos de un software a otro se lo hace de forma separada ya que la simulación dentro de CoppeliaSim no considera el tiempo de simulación del cliente API.

El orden de simulación al usar la comunicación asincrónica empieza estableciendo la conexión entre el cliente y el servidor, a partir de esto se inicia la simulación dentro de ambas aplicaciones y en este punto se puede enviar o recibir datos desde el cliente API iniciando las acciones del objeto al que se apunta, el objeto o robot se mantendrá siguiendo la instrucción anterior hasta la llegada de un nuevo comando como se muestra en la figura 1.15:

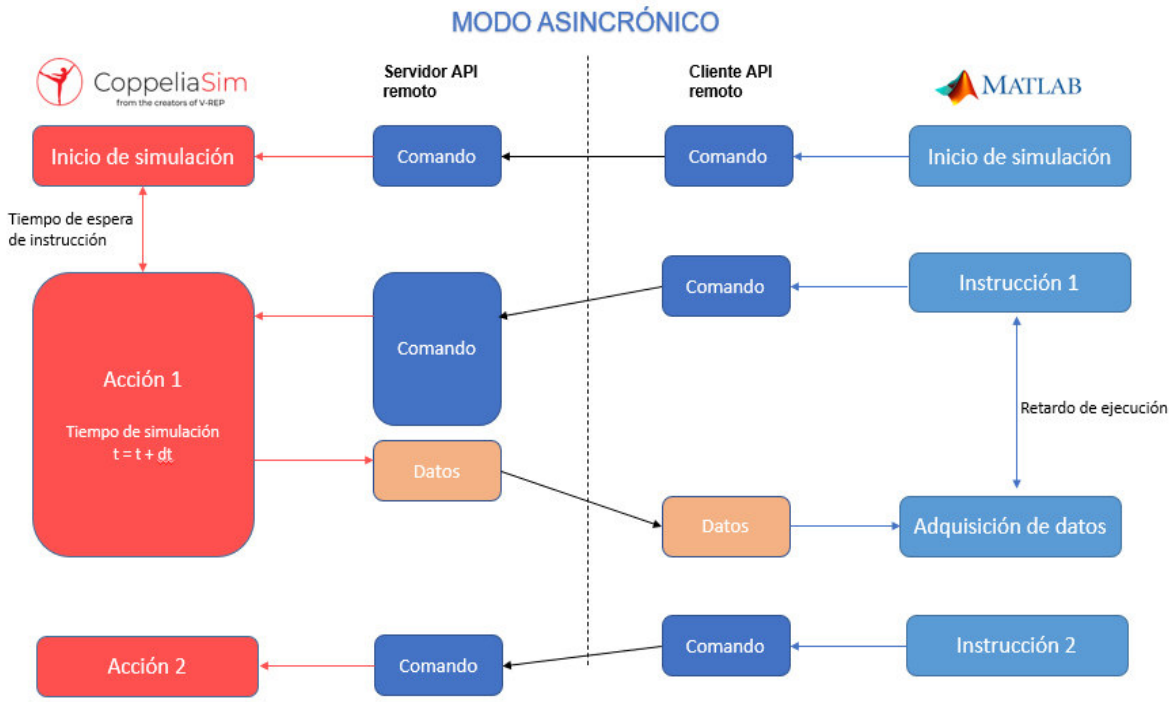


Figura 1.15. Modo de operación asincrónico

1.4.8.2 Comunicación Sincrónica "remote API"

El segundo modo de operación es el sincrónico, cuya principal característica es el de simular de forma paralela el cliente API remoto y CoppeliaSim haciendo que el progreso de ambas aplicaciones se sincronice permitiendo obtener gráficas y respuestas que usen una base de tiempo más cercana a la realidad. La sincronización consiste en controlar desde el lado del cliente el progreso del servidor remoto.

El orden de simulación usando la comunicación sincrónica empieza de igual manera estableciendo la conexión entre el cliente y el servidor, luego con una señal de inicio ("Trigger") ambas aplicaciones empezarán su progreso, a diferencia de la comunicación asincrónica cada comando se ejecutará en el servidor remoto como un paso de simulación a la vez y para realizar el siguiente paso de simulación necesita una señal de disparo (Trigger) desde el cliente remoto, todo esto se puede ver en la Figura 1.16. [4]:

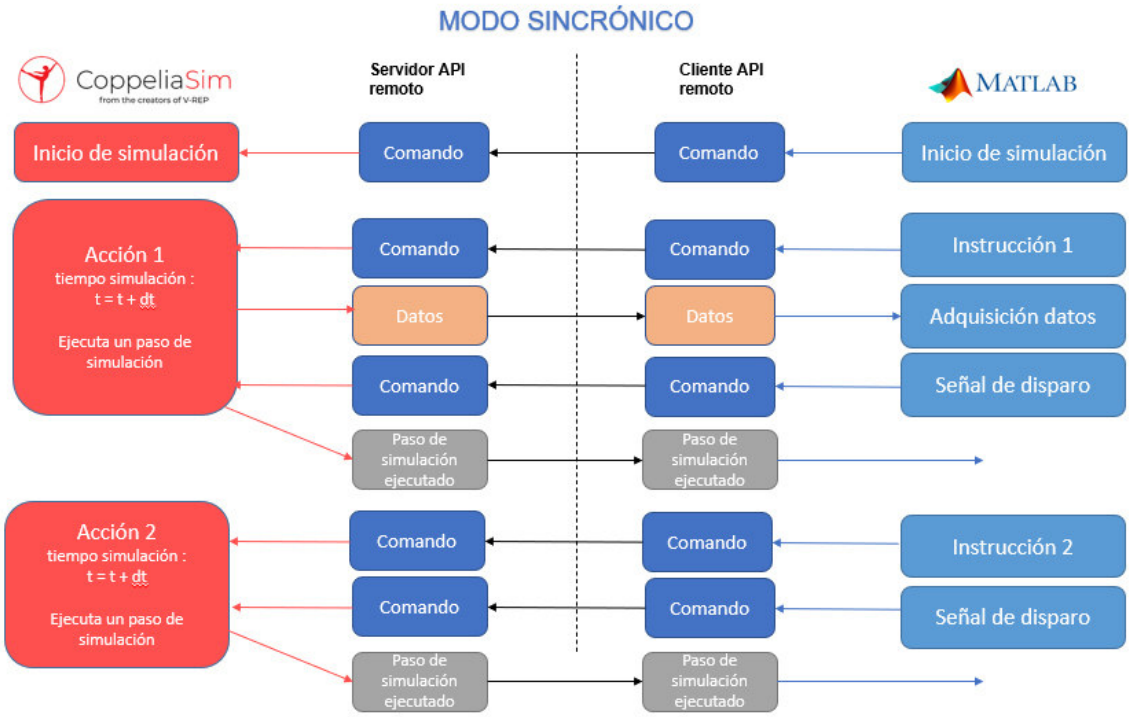


Figura 1.16. Modo de operación sincrónico

1.4.9 SOFTWARE DE SIMULACIÓN MATLAB-SIMULINK

Matlab (**Mat**rix **Lab**oratory) es un software matemático que se especializa en el trabajo numérico usando arreglos y matrices, este tiene un lenguaje de programación que permite realizar operaciones matemáticas y aritméticas, crear variables, definir distintos tipos de datos, visualizar información en gráficas y adicionalmente cuenta con una gran variedad de funciones y comandos que facilitan y permiten implementar desarrollos matemáticos más complejos sin tener que usar demasiadas líneas de código.

Es un programa que puede ser usado para distintos fines ya que posee una gran variedad de “toolbox” , los cuales son aplicaciones que contienen funciones especializadas para ciertas tareas como por ejemplo: Análisis de señales, Sistemas de Control, Simulación de Robots, etc.

Por otro lado, SIMULINK es una extensión de Matlab, esta extensión crea un entorno para programación visual, basada en diagramas de bloques que permiten validar modelos físicos de manera más sencilla y obtener señales de manera más intuitiva. En la figura 1.17 se muestra el logo de este software perteneciente a MATHWORKS, corporación propietaria.

Figura 1.17. Logo asociado al software MATLAB

1.4.9.1 Funciones REMOTE API para Matlab

Dentro de la comunicación entre CoppeliaSim y Matlab existen funciones que permiten enviar instrucciones del cliente remoto hacia el servidor, generalmente para enviar o recibir datos. Estas funciones actúan sobre la comunicación y modifican las condiciones de simulación. Las más importantes para el desarrollo de este trabajo se presentan en la Tabla 1.3 [18]:

Tabla 1.3 Funciones del cliente API remoto y descripción [18]

Función (*tipo de dato* nombre)	**Descripción**
remApi('remoteApi')	Esta función crea el objeto y carga las librerías
simxStart (*string* Dirección de Conexión, *number* Puerto de Conexión, *boolean* Esperar Conexión, *boolean* No volver a conectar una vez desconectado, *number* Tiempo out in en milisegundos, *number* Ciclos de comunicación en ms)	Esta función inicia el hilo de comunicación entre el servidor y el cliente solo se puede tener una comunicación por IP y puerto, la dirección de conexión es el IP del servidor, el puerto de conexión el número de puerto al que se conectará, los booleanos generalmente son 'true' para ambos casos, el tiempo OutIn es 5000 y los ciclos de comunicación son las veces que los datos se envían y reciben se recomienda 5 **Ej:** simxStart ('127.0.0.1',19997, true, true, 5000, 5); **Dentro de CoppeliaSim** Se debe incluir la función 'simRemoteApi.start(19999, 1300, false, true)'
simxFinish(*number* clientID)	Esta función permite finalizar el hilo de comunicación de cierto cliente remoto, el argumento es un número y se usa el valor (-1) para cerrar a comunicación. **Ej:** simxFinish (-1);

simxSynchronous(*number* clientID, *boolean* activación)	Activa o Desactiva el modo de operación sincrónico del cliente remoto que se defina y se encuentre conectado. **Ej:** simxSynchronous(clientID, true);
simxStartSimulation(*number* clientID, *number* Modo de Operación)	Permite iniciar la simulación o activar una simulación pausada.
simxPauseSimulation(*number* clientID, *number* Modo de Operación)	Pausa la simulación que se esté ejecutando.
simxStopSimulation(*number* clientID, *number* Modo de Operación)	Detiene la simulación que se esté ejecutando.
simxGetObjectHandle(*number* clientID, *string* Nombre del objeto *number* Modo de Operación)	Regresa el "handle" del objeto que se requiera usando su nombre. **Ej:** simxGetObjectHandle(clientID,'Pioneer',vrep.simx_opmode_blocking);
simxSynchronousTrigger(*number* clientID)	Envía una señal de disparo hacia el servidor, solo funciona cuando se esté trabajando en operación sincrónica.
simxSetJointTargetVelocity(*number* clientID, *number* Handle de la articulación, *number* Velocidad, *number* Modo de Operación)	Esta instrucción setea la velocidad de una articulación del robot, para poder hacer esto es necesario primero obtener el "handle" de la articulación a la que se apunta. **Dentro de CoppeliaSim** Se debe activar el modo torque/forcé de la articulación que se planee manipular.
simxGetObjectVelocity(*number* clientID, *number* Handle del objeto, *number* Modo de Operación)	Esta instrucción obtiene la velocidad de un objeto, para poder hacer esto es necesario primero obtener el handle del objeto al que se apunta. Al usar este comando se obtiene 2 arreglos 1 de velocidades lineales en sentido de los ejes x, y, z y otra angular tomando como eje de rotación los mismos ejes.
simxGetObjectPosition(*number* clientID, *number* Handle del objeto, *number* Posición relativa a que referencia, *number* Modo de Operación)	Obtiene la posición de un objeto, para esto es necesario obtener previamente el "handle" del objeto al que se apunta. Se obtiene un vector de posición en x, y, z. Para especificar la referencia que tendrá la posición se puede usar (-1) y se obtiene la posición absoluta respecto a 0,0.
simxGetObjectOrientation(*number* clientID, *number* Handle del objeto, *number* Posición relativa a una	Regresa la orientación en ángulos de Euler (indican la rotación de un cuerpo) de un objeto.

referencia definida, *number* Modo de Operación)	Para especificar la referencia que tendrá la posición se puede usar (-1) y se obtiene la posición absoluta respecto a 0,0 u obteniendo el "frame" de referencia de otro objeto.
simxSetObjectOrientation(*number* clientID, *number* Handle del objeto, *number* Posición relativa a que referencia, *array* Ángulos de Euler, *number* Modo de Operación)	Setea la orientación de un objeto en base a las coordenadas especificadas, es necesario obtener el handle del objeto primero.

1.4.9.2 AppDesigner

AppDesigner es una aplicación dentro de Matlab que permite crear y diseñar interfaces gráficas de usuario de forma sencilla y con resultados profesionales, este software facilita la tarea de agregar componentes dentro de la pantalla y asociar un código a estos mediante "callbacks" los cuales son funciones internas que permitirán realizar cierta acción al haber actuado sobre el componente codificado. AppDesigner proporciona ayudas para alinear, separar y centrar los componentes de la interfaz. Además, tiene la posibilidad de exportar la aplicación creada para el uso de cualquier usuario [18].

1.4.10 CONDICIONES DE ESTABILIDAD SEGÚN LYAPUNOV

Al momento de diseñar controladores, se presenta la opción de hacerlo mediante una función o postulado de Lyapunov, una función de Lyapunov $V(x)$ estable cumple tres condiciones en sistemas autónomos, lineales e invariantes en el tiempo:

$$V(0) = 0 \tag{1.30}$$

$$V(x) > 0\,, \forall\, x \neq 0 \tag{1.31}$$

$$\dot{V}(x) < 0\,, \forall\, x \neq 0 \tag{1.32}$$

Esta función además debe ser continuamente diferenciable y su ilustración cuando x es de dimensión 2, se muestra en la figura 1.18 [13]:

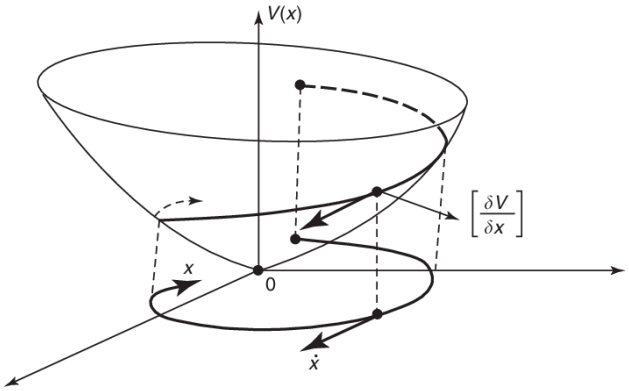


Figura 1.18 Curva de una función de Lyapunov

Nótese que cualquier punto, lleva a que en algún momento la variable x tenga el valor de cero, esto quiere decir que ante cualquier condición o evento en el que empiece x, la curva forzará a que siempre su valor converja a cero después de un tiempo.

Los postulados de Lyapunov son interesantes si se considera X (de simple o varias variables) como los errores del sistema, llevando al diseño de un controlador siempre estable y con un eventual seguimiento de referencia, pues $X \to 0\,, \text{cuando}\;\; t \to \infty$.

1.4.11 CONTROL POR MODOS DESLIZANTES (SMC)

El control SMC es una técnica de control no lineal que brinda propiedades como robustez, precisión y sintonización sencilla [14]. Este controlador permite llevar el sistema a estados específicos definidos por una superficie deslizante, una vez se ha alcanzado este estado, el controlador lo mantiene muy cerca a la referencia.

La acción de control de un controlador SMC (U_{SMC}) es la suma de la acción continua (U_{Con}) y de la acción discontinua (U_{Dis}), como se muestra en la ecuación 1.33:

$$U_{SMC} = U_{Con} + U_{Dis} \tag{1.33}$$

El diseño involucra dos partes, una de ellas que se encarga de la parte continua, que tiene relación con el modelo del sistema y la superficie deslizante. La otra parte se encarga del switcheo para la alcanzabilidad y es conocida como la parte discontinua. En la figura 1.19 se muestra la acción de cada una de estas partes con el fin de mantener el estado deseado sin ser sensible ante perturbaciones de una magnitud cualquiera.

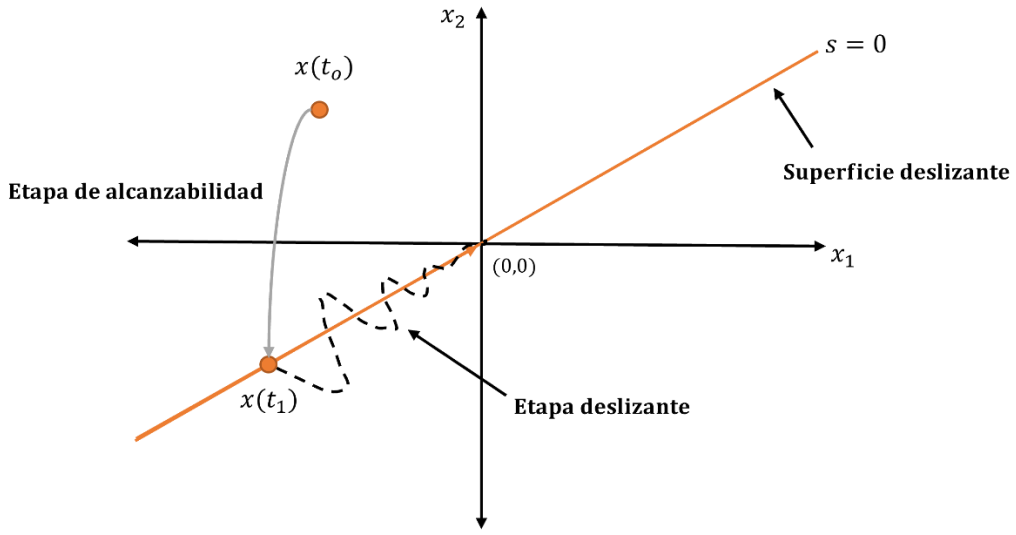


Figura 1.19. Superficie deslizante y acciones del SMC

Al ver la figura 1.19, nótese que se empieza en un punto inicial de error cualquiera $x(t_o)$, es ahí donde la acción discontinua de alcanzabilidad que tiene relación con la función signo actúa para llegar al punto $x(t_1)$, el cual se encuentra cerca de la superficie deslizante, donde la acción continua empieza su rol para llevar todo el sistema a un error nulo $(0,0)$.

En la metodología se establece una técnica de diseño para sustituir la función signo por la función sigmoide y de esta forma reducir el impacto de "Chattering" [14]. El Chattering es un switcheo a alta frecuencia que ocurre al momento de encontrarse cerca de la superficie deslizante, ya que la función signo sigue actuando como se muestra en la figura 1.20:

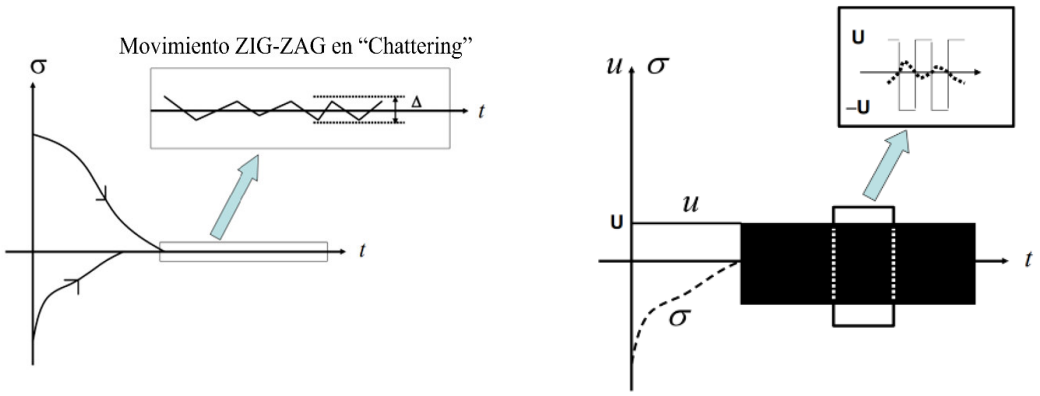


Figura 1.20. Chattering presente en el control SMC

1.4.12 PREDICTOR DE SMITH

Como se muestra en la Figura 1.21, el predictor de Smith es un controlador basado en modelo que tiene dos bucles. En el bucle interno que utiliza el modelo de proceso sin el tiempo muerto para predecir la salida ya que se retroalimenta al controlador principal $G_c\ (s)$ para generar la señal de control apropiada, con el fin de que la salida del proceso rastree el punto de ajuste o señal de referencia r sin la interferencia del tiempo muerto [15].

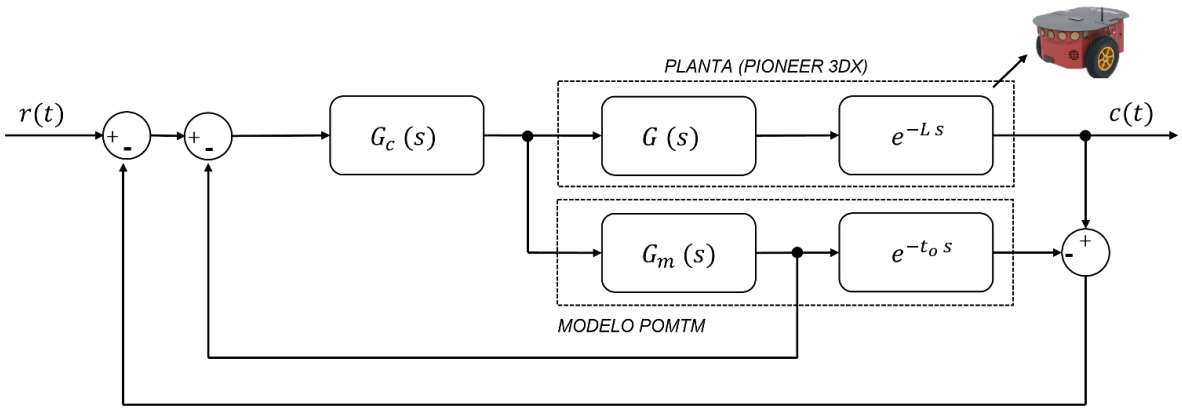


Figura 1.21. Esquema de un Predictor de Smith

Como este lazo no contiene el tiempo muerto, la ganancia del controlador puede ser sintonizada para tener una alcanzabilidad más rápida. Los efectos de las perturbaciones y los errores de modelado son corregidos en el lazo interno. Nótese que se requiere el modelo de primer orden $G_m\ (s)$ más tiempo muerto ($e^{-t_o\,s}$) , con sus siglas POMTM.

En el diagrama se distingue además la planta real del robot Pioneer representada por su modelo $G(s)$ y el tiempo muerto real de la planta (L).

Este trabajo estará orientado al diseño de un controlador en cascada, el lazo externo será un controlador basado en una función de Lyapunov que asegure la estabilidad y brinde una referencia de velocidad adecuada al controlador interno que tendrá el esquema de Predictor de Smith de la figura 1.21 donde el controlador principal $G_c\ (s)$ será un controlador por modos deslizantes SMC basado en una superficie deslizante tipo PI como lo recomienda Sarabia en su trabajo [3].

1.4.13 ÍNDICES DE DESEMPEÑO

Cuando se requiere realizar la comparativa de controladores, se utiliza los llamados índices de desempeño como indicadores de la efectividad y la suavidad que ejerce la acción de

control en los elementos finales de control de la planta. Los índices de desempeño más importantes son el índice ISE y el índice ISCO:

1.4.13.1 ISE

El índice de la integral del error cuadrático (ISE) permite medir el desempeño del controlador al momento de seguir la referencia en todo el sistema. No toma en cuenta si el sistema está adelantado o detrás de la referencia, pues cuantifica también los errores negativos al estar elevado al cuadrado [16]. La integral permite establecer el valor acumulado en cada medición.

A menor ISE se tiene un controlador de mejor desempeño, pues esto significará que su error no estará presente durante mucho tiempo en un intervalo de tiempo definido, su expresión viene definida como:

$$ISE = \int_0^t [e(t)]^2 \, dt \tag{1.34}$$

En este trabajo se tendrá acceso a los errores en $\boldsymbol{x}$ e $\boldsymbol{y}$, por lo que se obtiene una expresión que unifique para tener un solo índice comparador:

$$e(t) = \sqrt{(e_x)^2 + (e_y)^2} \tag{1.35}$$

Así entonces, reemplazando en la ecuación 1.34 se obtiene la expresión para un intervalo de tiempo $\boldsymbol{t}$:

$$ISE = \int_0^t \left[(e_x)^2 + (e_y)^2\right] dt \tag{1.36}$$

1.4.13.2 ISCO

El índice de la integral a la salida del controlador cuadrática (ISCO) permite medir la amplitud acumulada de la señal de control. Esto permite analizar cuál es el controlador que fatiga de mayor manera al elemento final de control [16].

A menor ISCO se tiene un controlador de mejor calidad, pues esto significaría que el elemento final de control no se somete a una señal muy amplia durante mucho tiempo, su expresión en un intervalo de tiempo $\boldsymbol{t}$ viene definida como:

$$ISCO = \int_0^t [CO(t)]^2 \, dt \tag{1.37}$$

Donde $\boldsymbol{CO(t)}$ es la señal a la salida del controlador que se suministra en la planta.

2 METODOLOGÍA

El presente trabajo de integración curricular emplea la investigación descriptiva con un enfoque cuantitativo, los datos fueron recolectados mediante mediciones y observaciones de las respuestas obtenidas en cada uno de los controladores tanto en MATLAB-SIMULINK como en el software CoppeliaSim en el Tomo II de este trabajo.

Se utilizó el método deductivo, partiendo de los conceptos generales establecidos en el marco teórico y definiendo una secuencia que permita replicar el componente al que se orienta este trabajo.

La información aplicada en este documento se basa en trabajos anteriores cuyo enfoque es el control en lazo cerrado del robot móvil Pioneer 3DX, artículos científicos enfocados al modelo, libros y otros recursos de control automático y digital que sirvan de herramienta para evidenciar el desempeño del controlador propuesto.

En este apartado se detalla la identificación de las curvas de reacción asociadas a las velocidades en el software CoppeliaSim, se realizará el diseño y discretización del controlador de posición basado en un postulado de Lyapunov, del controlador interno de velocidad PI y del controlador interno por modos deslizantes (SMC) con Predictor de Smith (SMC + SP) en la unificación del esquema.

2.1 Identificación y validación

La identificación es la técnica necesaria para deducir modelos matemáticos de un sistema a partir de una entrada bien definida y su salida, sin la necesidad de conocer los aspectos dinámicos de la planta, es decir, con la identificación se puede obtener una expresión matemática capaz de representar a un sistema de manera fiel de una forma más sencilla.

Cabe destacar que la identificación debe venir acompañada de la validación cuando se trabaja en una máquina o planta real, esto incluye aplicar el conocimiento previo, los datos experimentales y la experiencia usando el modelo para verificar si el resultado obtenido es efectivo.

2.1.1 Identificación de la velocidad lineal en el robot Pioneer 3DX disponible en el software CoppeliaSim

Este trabajo está orientado al diseño de controladores para el robot móvil Pioneer 3DX disponible en el software CoppeliaSim. Sin embargo, este Tomo no se enfoca en la implementación, por lo que la forma de conexión, envío y recepción de datos entre MATLAB y CoppeliaSim, así como los comandos utilizados y procedimiento asociado son descritos en el Tomo II. En el ANEXO I se encuentran los códigos implementados en esta sección.

Para la identificación, se realizará la aproximación a un modelo de primer orden más tiempo muerto (POMTM), el cual tiene la siguiente forma en el dominio de la frecuencia (Laplace):

$$\frac{X(s)}{U(s)} = \frac{K}{\tau s+1}\ e^{-t_o\, s} \tag{2.1}$$

Donde $\boldsymbol{K}$ es la ganancia determinada como la relación entre la variación de la salida respecto a una variación de la entrada en estado estable, $\boldsymbol{t_o}$ el tiempo muerto, es decir el tiempo que tarda el sistema en variar su estado anterior a partir de la estimulación recibida en la entrada y $\boldsymbol{\tau}$ la constante de tiempo definida como el tiempo necesario para alcanzar el $\boldsymbol{63,2\%}$ del valor en estado estable a partir del término del tiempo muerto en la figura 2.1 se diferencian estos valores que definen el modelo de POMTM.

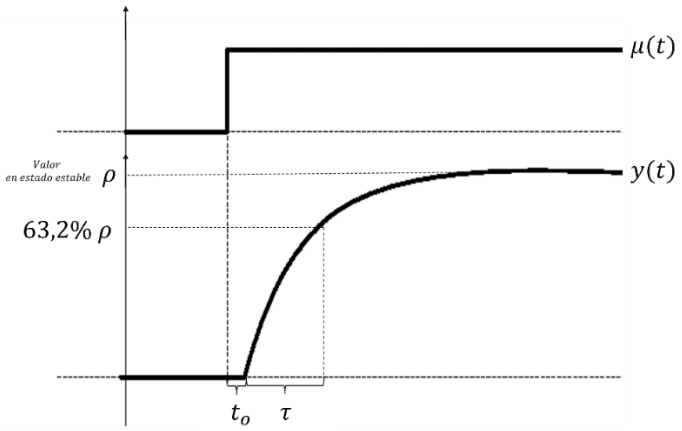


Figura 2.1. Parámetros de un modelo POMTM

En este caso ante una entrada $\mu(t)$ se tiene una salida $y(t)$, nótese que no se conoce el modelo propio del sistema de la figura 2.1. ni su naturaleza dinámica. Sin embargo, se lo puede aproximar perfectamente a un sistema de POMTM, lo que facilita el diseño de su controlador y permite un desarrollo matemático más simplificado.

La ganancia K está bien definida por la variación de la entrada $\Delta\mu$ y la variación de la salida $\Delta Y(t \to \infty)$ en estado estable, teniendo:

$$K = \frac{\Delta Y(t\to\infty)}{\Delta\, \mu} \tag{2.2}$$

Alfaro [17] en su trabajo realiza la comparativa de varias propuestas para la identificación de un modelo de POMTM, llegando a determinar unos valores óptimos que reducen la variación entre la salida real y el modelo. Los porcentajes de respuesta para las medidas de los tiempos y las constantes para encontrar los parámetros se muestran en la tabla 2.1:

Tabla 2.1. Valores de Alfaro para la identificación de un modelo POMTM

Método	$\%\rho_1\ (t_1)$	$\%\rho_2\ (t_2)$	A	B	C	D
Alfaro	25%	75%	-0,910	0,910	1,262	-0,262

Donde:

$$\tau = At_1 + Bt_2 \tag{2.3}$$

$$t_o = Ct_1 + Dt_2 \tag{2.4}$$

A continuación, se inyecta una señal paso de amplitud $0{,}4$ en la velocidad lineal, luego de un tiempo se incrementa hasta $0{,}6$, lo que permite evidenciar la curva de respuesta cuando el robot está en funcionamiento luego del arranque. En la figura 2.2 se muestra la señal de control y la respuesta de la velocidad lineal, donde se ha realizado un "zoom" en el área limitada por la linea inter puntada, con el fin de analizar a detalle la curva de reacción:

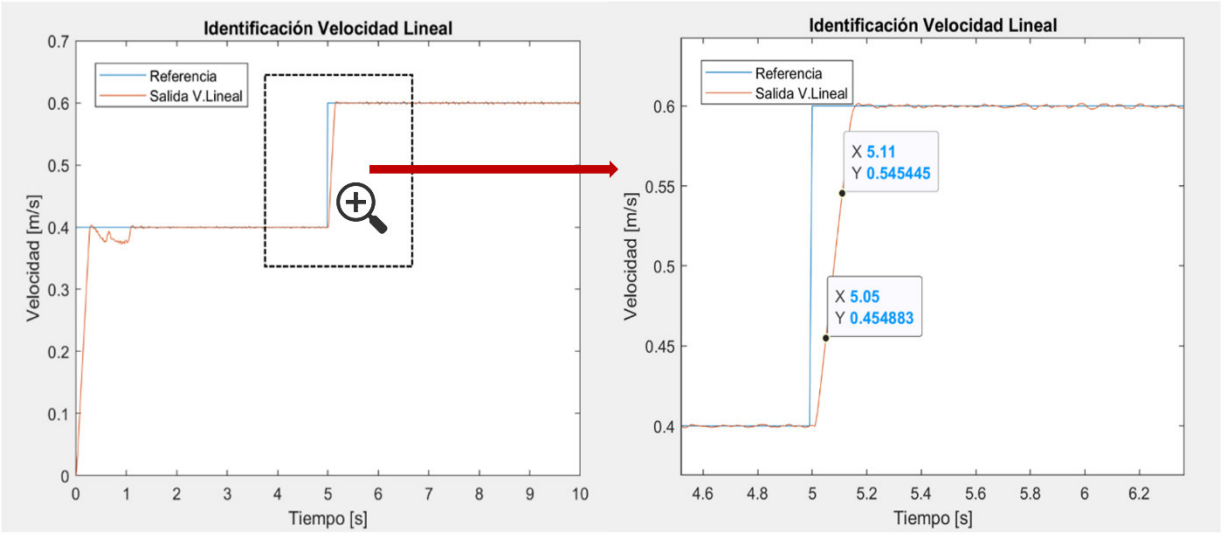


Figura 2.2. Curva de respuesta de la velocidad lineal

La comunicación se la realiza con un tiempo de muestreo de 0,01 segundos, determinados en el capítulo 2.6 de este trabajo, esto con el fin de tener una señal confiable, pues es el menor tiempo que permite establecer Coppelia como paso de simulación.

Es evidente que la ganancia en estado estable para el caso de esta velocidad es la unidad, pues en base a la ecuación 2.2 se tiene:

$$K = \frac{\Delta v(t \to \infty)}{\Delta U} = \frac{0{,}6-0{,}4}{0{,}6-0{,}4} = \frac{0{,}2}{0{,}2} = 1 \quad (2.5)$$

Nótese que la variación en la entrada (ΔU) es de 0,2 m/s, pues se realiza un cambio de 0,4 m/s a 0,6 m/s a los 5 segundos de simulación en la señal de color azul, obteniendo una variación en estado estable del mismo valor en la salida pues se evidencia el cambio de 0,4m/s a 0,6 m/s, pues cuando el tiempo tiende a infinito la señal naranja (velocidad lineal) tiene una variación idéntica a la de la entrada.

En la figura 2.2 se muestran los puntos de interés más próximos al 25% y 75% del valor en estado estable que define Alfaro ($0{,}45\ m/s\ y\ 0{,}55\ m/s$) en $5{,}05\ y\ 5{,}11$ segundos respectivamente, teniendo los tiempos de $5{,}047$ y $5{,}114$ segundos al interpolar los puntos más cercanos y restando el tiempo en que la señal cambia de referencia ($5\ seg$) se tiene:

$$t_1 = 0{,}047 \qquad t_2 = 0{,}114$$

Entonces en base a las ecuaciones 2.3 y 2.4 se tiene:

$$\tau = -0{,}910 * 0{,}047 + 0{,}910 * 0{,}114 = \ 0{,}06097 \quad (2.6)$$

$$t_o = \ \ 1{,}262 * 0{,}047 - 0{,}262 * 0{,}114 \ = \ 0{,}030291 \quad (2.7)$$

Teniendo esto en cuenta el modelo de POMTM para la salida de velocidad lineal $v(s)$ ante una entrada $U\ (s)$ viene definida por la siguiente ecuación:

$$\frac{v(s)}{U(s)} = \frac{1}{0{,}0609\ s+1}\ e^{-0{,}030291\ s} \quad (2.8)$$

2.1.2 Identificación de la velocidad angular en el robot Pioneer 3DX disponible en el software CoppeliaSim

Siguiendo un procedimiento muy similar al de la velocidad lineal se obtiene el modelo de POMTM de la velocidad angular, pero esta vez el experimento varía, pues se inyecta una señal de velocidad lineal en un principio para vencer el arranque y mientras se encuentra en movimiento se varía la velocidad angular en dos señales pasos de diferente magnitud, una de ellas a $0{,}4\ rad/s$ y otra a $1\ rad/s$ en $0\ y\ 5$ segundos de simulación respectivamente. En la figura 2.3 se encuentra esta curva de reacción.

Nótese que esta vez la ganancia no es la unidad y que se tiene diferentes valores en los tiempos, lo que implicará una variación de la constante de tiempo y el tiempo muerto. La curva de repuesta con "zoom" en la zona de interés se muestra en la figura 2.3:

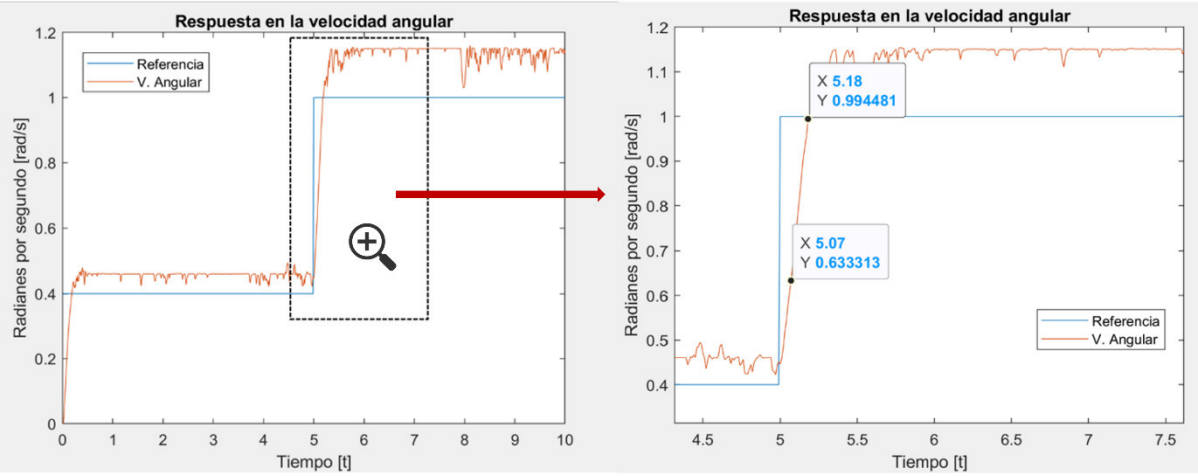


Figura 2.3. Curva de respuesta de la velocidad angular

En base al procedimiento anterior se tiene:

$$K = \frac{\Delta w(t\to\infty)}{\Delta W} = \frac{1{,}12144-0{,}442624}{1-0{,}4} = 1{,}14136 \qquad (2.9)$$

Los tiempos interpolados en los que se obtiene el 25% y el 75% de la variación en estado estable son $5{,}062$ y $5{,}176$ segundos respectivamente y restando el tiempo en que la señal cambia de referencia ($5\ seg$) se tiene:

$$t_1 = 0{,}062 \qquad t_2 = 0{,}176$$

$$\tau = -0{,}910 * 0{,}062 + 0{,}910 * 0{,}176 = \ 0{,}10374 \qquad (2.10)$$

$$t_o = \ \ 1{,}262 * 0{,}062 - 0{,}262 * 0{,}176 \ = \ 0{,}03213 \qquad (2.11)$$

Por lo que su modelo de POMTM para la salida de velocidad angular $w(s)$ ante una entrada $W(s)$ es:

$$\frac{w(s)}{W(s)} = \frac{1.14136}{0{,}10374\ s+1}\ e^{-0{,}03213\ s} \qquad (2.12)$$

2.1.3 Validación de los modelos

En la figura 2.4 se muestran las referencias, salidas reales y salida del modelo ante una señal paso, nótese que las aproximaciones son muy precisas y por lo tanto el modelo es una representación fiel del sistema. Estas expresiones serán de gran utilidad al momento de diseñar los controladores en el lazo interno.

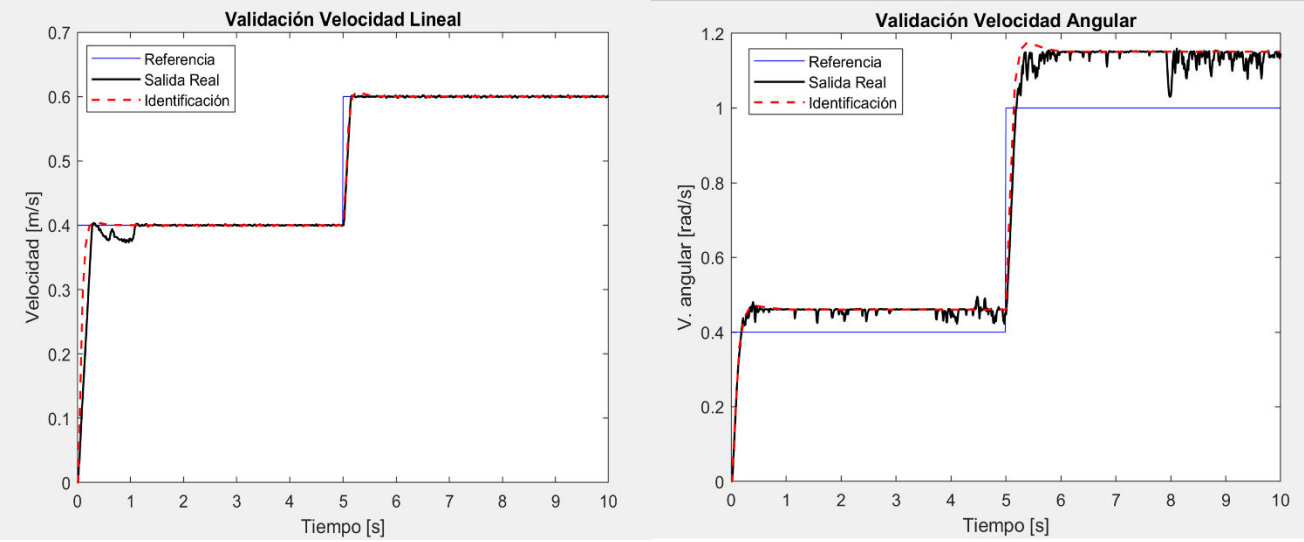


Figura 2.4. Validación de los modelos

La identificación se realiza únicamente de las velocidades que serán controladas en el lazo interno, pues ya se conoce el modelo para la posición definida en el modelo cinemático del robot que se controla en el lazo externo.

En la tabla 2.2 se muestra el índice ISE para evidenciar el error de modelado entre la salida real y la salida de la identificación, nótese que los valores son muy bajos, lo que indica un proceso de validación satisfactorio.

Tabla 2.2. Índice ISE para error de modelado

Índice	Modelo Velocidad Lineal	Modelo Velocidad Angular
ISE	0,2247	0,3272

2.2 Topología de los controladores

Cuando se tiene un mismo proceso en el que se involucran variables rápidas y variables lentas, se propone una topología tipo cascada, esto con el fin de actuar de manera más agresiva en el lazo interno con la variable veloz y que la variable tardía no se vea afectada por los cambios de referencia y perturbaciones presentes en el lazo interno.

En este trabajo las variables rápidas serán las velocidades (lineal y angular), mientras que las variables lentas serán la posición y orientación del robot Pioneer 3DX. En la figura 2.5 se muestra las diferencias de un controlador en lazo simple con un controlador tipo cascada para un proceso genérico.

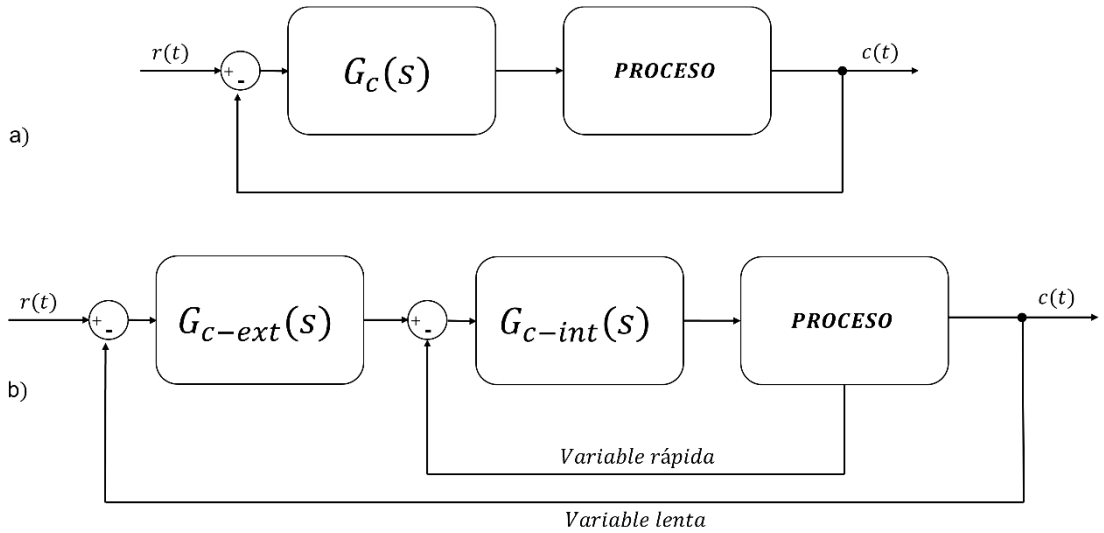


Figura 2.5. a) Control en lazo simple b) Control en cascada

Nótese que en la topología tipo cascada, la salida del controlador externo ($\boldsymbol{G_{c-ext}(s)}$) sirve como referencia del lazo interno donde actúa su controlador ($\boldsymbol{G_{c-int}(s)}$), por lo que se debe buscar que a la salida del lazo externo se tenga la misma variable a controlar del lazo interno.

En este trabajo se busca un controlador de lazo externo que permita traducir el error de posición (Variable lenta a controlar en el lazo externo) en referencias de velocidad angular y lineal (Variables rápidas a controlar en el lazo interno).

2.2.1 Topología del controlador en lazo simple:

Para empezar el diseño se requiere de un bloque que permita traducir la referencia de posición definida por la trayectoria $\boldsymbol{h_d}$ en velocidades ($\boldsymbol{v_c}$ $\boldsymbol{y}$ $\boldsymbol{w_c}$) que serán suministradas como señales de control en el robot. Este controlador puede funcionar por sí solo y estará basado en una función de Lyapunov, su topología inicial se muestra en la figura 2.6:

Nótese que las velocidades y posiciones actuales ($\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{h}$) pueden ser obtenidas mediante odometría, con esto se puede obtener el error de posición ($\tilde{\boldsymbol{h}}$) como la diferencia entre $\boldsymbol{h_d}$ y $\boldsymbol{h}$.

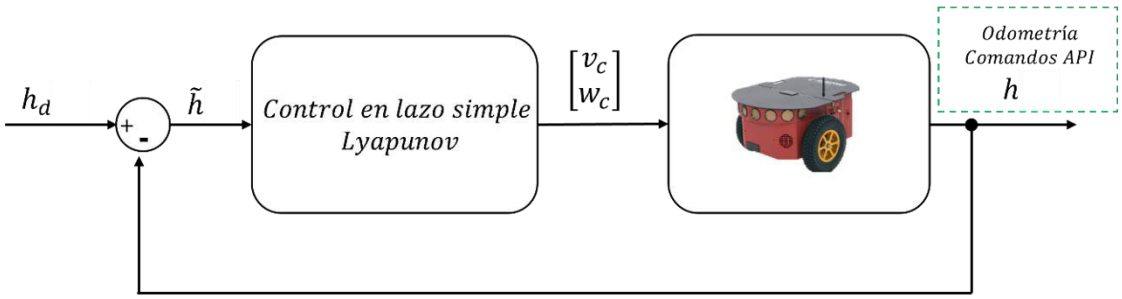


Figura 2.6. Topología del controlador en lazo simple

2.2.2 Topología del controlador en cascada:

Para mejorar los problemas del tiempo muerto y la velocidad de respuesta que brinda el robot Pioneer 3DX, se propone un controlador interno de velocidad. En este trabajo se implementará un control interno PI y un control interno con SMC más predictor de Smith (SMC+SP), la topología general en cascada se muestra en la figura 2.7.

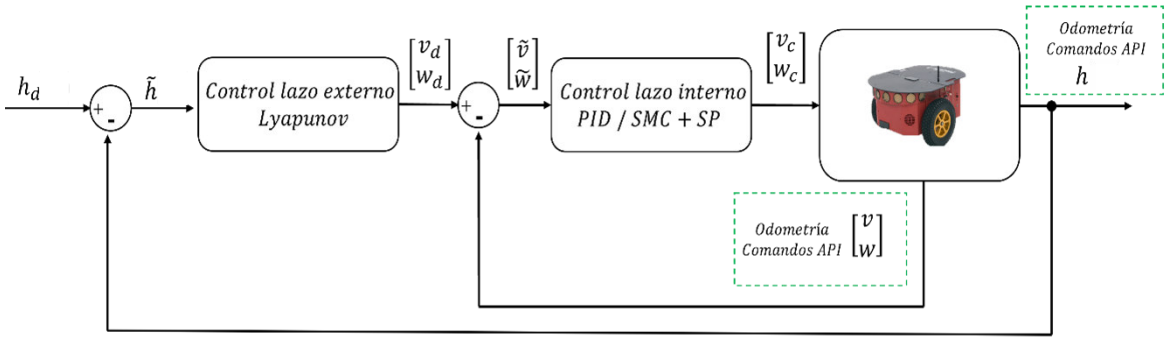


Figura 2.7. Topología general del controlador en cascada

En este esquema se tiene una entrada de posición h_d definida por la trayectoria, luego se calcula el error de posición $\tilde{h}$ como la diferencia entre h_d y la posición actual h obtenida por odometría. Se utiliza el control externo para proveer una velocidad de referencia ($v_d \; y \; w_d$), que servirá al lazo interno para nuevamente calcular el error de velocidad ($\tilde{v} \; y \; \tilde{w}$) y en base a su naturaleza (PI o SMC+SP) se obtiene unas acciones de control ($v_c \; y \; w_c$) que mejoran el desempeño del sistema.

Nótese que los diagramas anteriores son topologías genéricas para visualizar la idea de un control en lazo simple y un control en cascada. En este trabajo la comparativa se realizará de un controlador en lazo simple basado en Lyapunov, uno en cascada con PI interno y uno en cascada con SMC+SP en el lazo interno, estos dos últimos con el mismo

lazo externo basado en Lyapunov. Sus esquemas detallados se encuentran en los siguientes apartados.

2.3 Diseño del controlador basado en un postulado de Lyapunov para el lazo externo

Para el diseño de este controlador, se requiere definir algunas expresiones para entender que tipos de señales son útiles en el cálculo de la acción de control. Para empezar, definamos la expresión matricial del error de posición $\tilde{h}$, tanto para x como en y, esto debido a que la idea de trabajar con una función de Lyapunov es que el error tienda a cero, donde h_d es la posición deseada:

$$\tilde{h} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} = h_d - h \tag{2.13}$$

Ahora es necesario proponer la función de Lyapunov con el fin de encontrar la ley de control, tomando en cuenta la forma matricial que representa el cuadrado del error, entonces se propone la siguiente candidata de Lyapunov:

$$V(\tilde{h}) = \frac{1}{2}\, \tilde{h}^T \; \tilde{h} \tag{2.14}$$

La ecuación 2.14 satisface la primera y segunda condición del teorema de Lyapunov para sistemas continuos autónomos [13], pues cuando el error se evalúa en cero se tiene una $V(\tilde{h})$ nula y cuando se evalúa para cualquier valor diferente de cero $V(\tilde{h})$ siempre será positiva.

Apliquemos la regla de la cadena para obtener la derivada de la ecuación 2.14:

$$\dot{V} = \frac{1}{2}\, \dot{\tilde{h}}^T \; \tilde{h} \quad + \frac{1}{2}\, \tilde{h}^T \; \dot{\tilde{h}} \tag{2.15}$$

Tomando en cuenta que $\tilde{h}$ es un vector de 2x1 definido en la ecuación 2.13 es evidente que el producto $\dot{\tilde{h}}^T \; \tilde{h}$, da como resultado un escalar, por lo tanto, se puede definir las siguientes igualdades, la transpuesta de un escalar es el mismo valor:

$$\dot{\tilde{h}}^T \; \tilde{h} \; = \; (\, \dot{\tilde{h}}^T \; \tilde{h}\,)^{\,T} \tag{2.16}$$

Aplicando propiedades de las matrices, se distribuye el operador transpuesto:

$$\dot{\tilde{h}}^T \ \tilde{h} \ = (\ \tilde{h}^T \left(\dot{\tilde{h}}^T \right)^T) \tag{2.17}$$

Por último, se simplifica la expresión tomando en cuenta que al transponer una matriz ya transpuesta se obtiene la matriz original:

$$\dot{\tilde{h}}^T \ \tilde{h} \ = \tilde{h}^T \ \dot{\tilde{h}} \tag{2.18}$$

Con la igualdad de la ecuación 2.18 y unificando con la 2.15 se tiene:

$$\dot{V} = \frac{1}{2} \ \tilde{h}^T \ \dot{\tilde{h}} \ + \frac{1}{2} \ \tilde{h}^T \ \dot{\tilde{h}} \ = \ \ \tilde{h}^T \ \dot{\tilde{h}} \tag{2.19}$$

Derivando la definición del error de posición de la ecuación 2.13 y reemplazándola en la ecuación 2.19 se tiene:

$$\dot{V} = \ \tilde{h}^T \ (\dot{h}_d - \dot{h}) \tag{2.20}$$

Nótese que se busca una expresión que relacione el modelo obtenido en el marco teórico en la ecuación 1.13 sin la fila correspondiente al ángulo ∅, pues la orientación define la posición deseada y de esta manera simplifica la expresión de $\dot{\emptyset}$, así entonces se tiene la definición de Jacobiano y de la acción de control como:

$$\dot{h} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\emptyset) & -a \sin(\emptyset) \\ \sin(\emptyset) & a \cos(\emptyset) \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \ = J \ U \tag{2.21}$$

Reemplazando esta igualdad en la ecuación 2.20 se tiene:

$$\dot{V} = \ \tilde{h}^T \ \left(\dot{h}_d - JU\right) \tag{2.22}$$

La condición de estabilidad de Lyapunov, disponible en la ecuación 1.22 requiere que su derivada sea menor a cero, es por esto que se propone la siguiente acción de control:

$$U_c = \ J^{-1} \ (\dot{h}_d + K \ \tilde{h}) \tag{2.23}$$

Donde K es un parámetro para igualar las unidades y sintonizar el controlador, desarrollando se tiene:

$$\dot{V} = \ \tilde{h}^T \ (\dot{h}_d - J \ \ J^{-1} \ (\dot{h}_d + K \ \tilde{h}))$$

$$\dot{V} = \ \tilde{h}^T \ (\dot{h}_d - I \ (\dot{h}_d + K \ \tilde{h}))$$

$$\dot{V} = \ \tilde{h}^T \ \left(\dot{h}_d - \dot{h}_d - K \ \tilde{h}\right) = \ - K \ \tilde{h}^T \ \tilde{h} \quad , \forall \ K > 0 \tag{2.24}$$

La ecuación 2.24 satisface la tercera condición pues su derivada siempre será negativa para todo K definida positiva, pues $\dot{V} < 0$, por lo tanto $\tilde{h} \to 0$, cuando $t \to \infty$.

Lo interesante de este análisis radica en la obtención de la ley de control descrita en la ecuación 2.23, además, se determina la estabilidad del controlador y se obtienen las velocidades que servirán más adelante como referencia del lazo interno.

Se puede obtener fácilmente el jacobiano inverso, si se toma en cuenta un valor a de $0{,}2$ como la distancia en metros a la que se encuentra el punto de referencia por delante del robot:

$$J^{-1} = \begin{bmatrix} \cos(\emptyset) & -a\sin(\emptyset) \\ \sin(\emptyset) & a\cos(\emptyset) \end{bmatrix}^{-1} = \frac{1}{det|J|} \begin{pmatrix} a\cos(\emptyset) & a\sin(\emptyset) \\ -\sin(\emptyset) & \cos(\emptyset) \end{pmatrix}$$

$$J^{-1} = \frac{1}{a\,cos^2(\emptyset) + a\,sin^2(\emptyset)} \begin{pmatrix} a\cos(\emptyset) & a\sin(\emptyset) \\ -\sin(\emptyset) & \cos(\emptyset) \end{pmatrix} = \begin{pmatrix} \cos(\emptyset) & \sin(\emptyset) \\ \frac{-\sin(\emptyset)}{0{,}2} & \frac{\cos(\emptyset)}{0{,}2} \end{pmatrix} \quad (2.25)$$

Por lo tanto, la acción de control viene definida como:

$$U_c = \begin{bmatrix} v_c \\ w_c \end{bmatrix} = \begin{bmatrix} \cos(\emptyset) & \sin(\emptyset) \\ \frac{-\sin(\emptyset)}{0{,}2} & \frac{\cos(\emptyset)}{0{,}2} \end{bmatrix} \left(\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} + K \begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} \right) \quad (2.26)$$

Teniendo esto en cuenta, se establece el diagrama de bloques correspondiente a este controlador para una topología de lazo simple en la figura 2.8:

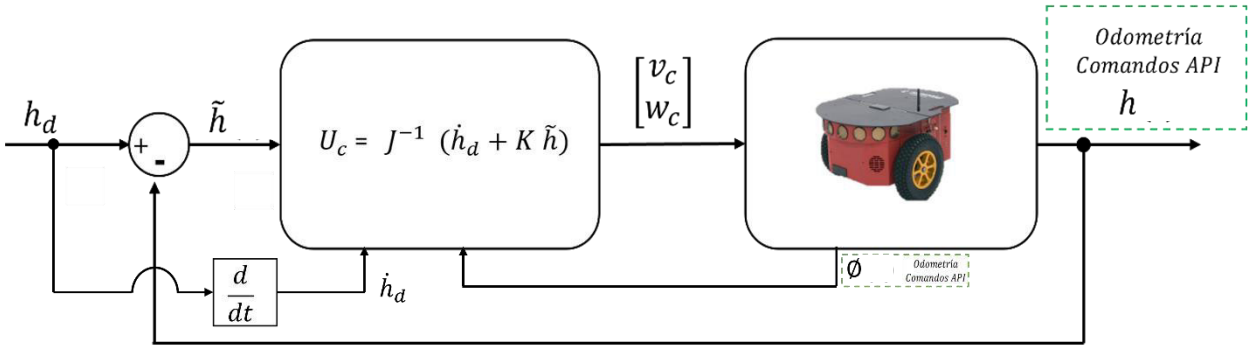


Figura 2.8. Diagrama de bloques para controlador en lazo simple por Lyapunov

2.4 Diseño de los controladores PI para el lazo interno de velocidades

Por lo general, un controlador PI tiene la habilidad de generar velocidad estable en sistemas robóticos móviles [6]. De hecho, Sarabia en su trabajo [3] recomienda el uso de

este tipo de controlador (PI) con el fin de mejorar la respuesta y evitar que la parte derivativa pueda llevar el sistema a la inestabilidad. La acción de control de un controlador PI está definida por la siguiente ecuación:

$$U_c = K_p\ e(t) + K_i \int_0^t e(t)\, dt \tag{2.27}$$

Donde $\boldsymbol{e(t)}$ es el error de la variable a controlar, que en este caso serían los errores de las velocidades lineales y angulares ($\tilde{\boldsymbol{v}}\ \boldsymbol{y}\ \tilde{\boldsymbol{w}}$), mientras que $\boldsymbol{K_p}\ \boldsymbol{y}\ \boldsymbol{K_i}$ son constantes de sintonización, pues se tiene controladores independentes, cada uno para un tipo de velocidad.

Para una adecuada sintonización de los controladores, se empleará el método en lazo abierto de Ziegler Nichols en base a los valores determinados en la identificación. En la tabla 2.3 se muestran las expresiones según el tipo de controlador en función de los parámetros de un sistema de POMTM:

Tabla 2.3. Método en lazo abierto de Ziegler Nichols

Controlador	$\boldsymbol{K_p}$	$\boldsymbol{\tau_i}$	$\boldsymbol{\tau_d}$
P	$\frac{\tau}{t_o}$	∞	0
PI	$0{,}9\frac{\tau}{t_o}$	$\frac{t_o}{0{,}3}$	0
PID	$1{,}2\ \frac{\tau}{t_o}$	$2\ t_o$	$0{,}5\ t_o$

En base a las ecuaciones 2.5, 2.6 y 2.7 y considerando el diseño de un controlador tipo PI se tiene la siguiente sintonización para el controlador de velocidad lineal ($\boldsymbol{v}$):

$$K_p = 0{,}9\ \cdot \frac{0{,}06097}{0{,}030291} = 1{,}811 \tag{2.28}$$

$$\tau_i = \frac{0{,}030291}{0{,}3} = 0{,}10097 \tag{2.29}$$

$$K_i = \frac{K_{pv}}{\tau_{iv}} = \frac{1{,}811}{0{,}10097} = 17{,}93 \tag{2.30}$$

Donde $\boldsymbol{K_p}$ es la ganancia proporcional, $\boldsymbol{\tau_i}$ es la constante de tiempo integral y $\boldsymbol{K_i}$ es la ganancia integral del controlador PI asociado a la velocidad lineal.

Así mismo, se realiza la sintonización para la velocidad angular ($\boldsymbol{w}$), teniendo:

$$K_p = 0{,}9\ \cdot \frac{0{,}10374}{0{,}03213} = 2{,}90 \tag{2.31}$$

$$\tau_i = \frac{0{,}03213}{0{,}3} = 0{,}1071 \tag{2.32}$$

$$K_i = \frac{K_{pw}}{\tau_{iw}} = \frac{2{,}90}{0{,}1071} = 27{,}07 \tag{2.33}$$

Donde $\boldsymbol{K_p}$ es la ganancia proporcional, $\boldsymbol{\tau_i}$ es la constante de tiempo integral y $\boldsymbol{K_i}$ es la ganancia integral del controlador PI asociado a la velocidad angular.

En base a esto y a la ecuación 2.27, se puede definir las acciones de control para cada una de las velocidades ($\boldsymbol{v_c}$ $\boldsymbol{y}$ $\boldsymbol{w_c}$), recordando que son controladores independientes, pues cada uno de ellos está enfocado a una curva de reacción diferente.

$$v_c = 1{,}811\ \tilde{v} + 17{,}93 \int \tilde{v}\ dt \tag{2.34}$$

$$w_c = 2{,}90\ \tilde{w} + 27{,}07 \int \tilde{w}\ dt \tag{2.35}$$

Donde los errores de velocidad ($\tilde{\boldsymbol{v}}$ $\boldsymbol{y}$ $\tilde{\boldsymbol{w}}$) son la diferencia entre la referencia deseada ($\boldsymbol{v_d}$ $\boldsymbol{y}$ $\boldsymbol{w_d}$, obtenidas como salida del lazo externo) menos el valor actual de cada una de las velocidades ($\boldsymbol{v}$ $\boldsymbol{y}$ $\boldsymbol{w}$), obtenidas por odometría:

El diagrama de bloques del controlador cascada con lazo interno tipo PI, se presenta en la figura 2.9:

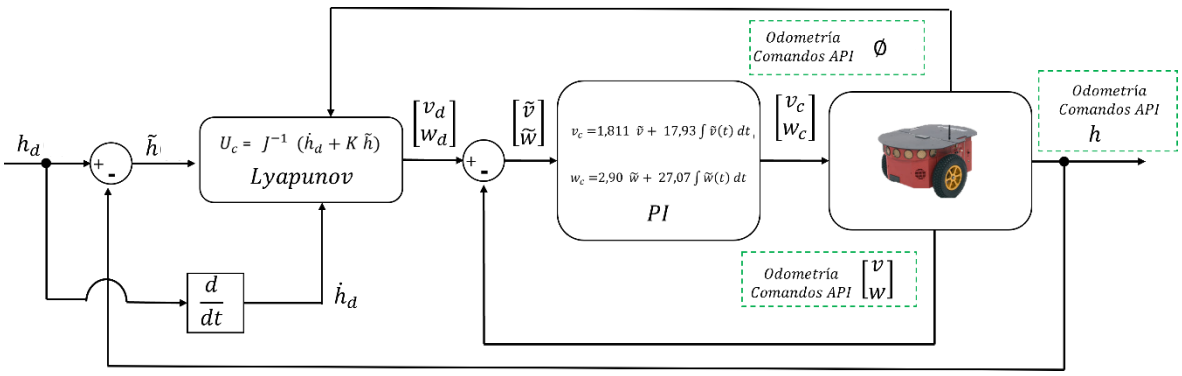


Figura 2.9. Diagrama de bloques para controlador en cascada con lazo externo basado en un postulado de Lyapunov y lazo interno basado en un controlador PI.

Los controladores PID y sus variaciones como el PI, son de fácil implementación y sintonización, además de ser muy comunes en la industria. Esta es la razón de su implementación en este trabajo, pues se busca analizar las ventajas y los beneficios del controlador SMC+SP respecto al PI.

2.5 Diseño del controlador SMC con esquema de Predictor de Smith

En la figura 2.10 se muestra el esquema del lazo interno que controlará las velocidades, nótese que en base al diagrama de la figura 1.21, cuyo controlador es por modos deslizantes (SMC), este controlador posee una acción continua $\boldsymbol{U_{Con}}$ basada en la superficie deslizante y una acción discontinua $\boldsymbol{U_{Dis}}$, basada en la función signo que otorga un "switcheo" que permite la alcanzabilidad. La suma de estas acciones de control forma la acción de control total de un SMC ($\boldsymbol{U_{SMC}}$).

En los siguientes apartados se diseña un controlador SMC para la velocidad lineal y otro para la velocidad angular, con controladores y esquemas de Predictor de Smith independientes.

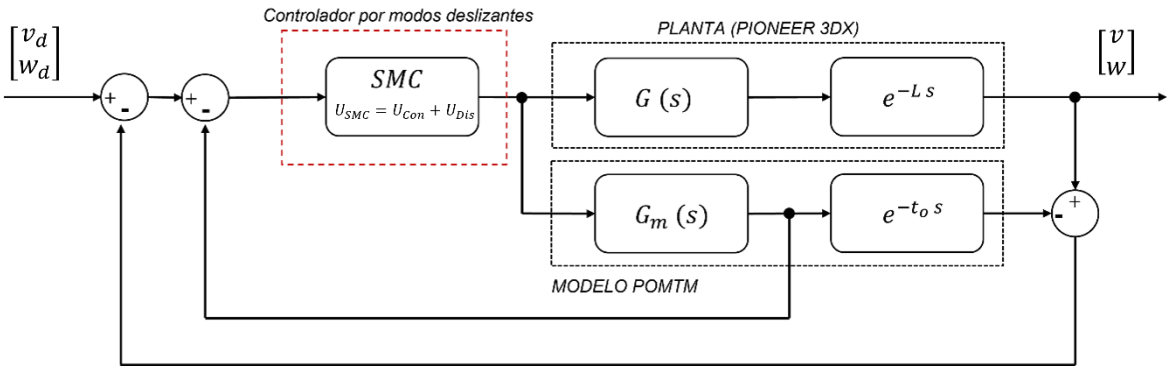


Figura 2.10. Lazo únicamente interno para el control de velocidades

2.5.1 Diseño parte continua SMC

Para el diseño de los controladores de lazo interno, se propone una superficie deslizante tipo PI, tal como recomienda Sarabia en su trabajo [3], para esto se utiliza el modelo de primer orden sin retardo, pues este se compensará con el esquema de Predictor de Smith.

Para el diseño del controlador de velocidad se tiene el modelo de primer orden de la forma:

$$\frac{v(s)}{U(s)} = \frac{K}{\tau\, s+1} \tag{2.36}$$

Donde τ es la constante de tiempo de la curva de reacción genérica, K es la ganancia en lazo abierto y $U(s)$ es la entrada.

Desarrollando las expresiones en el dominio del tiempo se tiene:

$$\tau\, s\, v + v = KU$$

$$\tau \frac{dv}{dt} + v = KU$$

$$\frac{d\,v}{dt} = \frac{K\,U}{\tau} - \frac{v}{\tau} \tag{2.37}$$

La superficie deslizante $S(t)$ es de tipo PI y viene definida por la siguiente ecuación, donde $\lambda_o\ y\ \lambda_1$ son parámetros de sintonización de la superficie deslizante y $e(t)$ el error genérico:

$$S(t) = \lambda_1 e(t) + \lambda_o \int e(t)\, dt \tag{2.38}$$

Es necesario mantener un error de cero todo el tiempo, para esto se define la derivada de la superficie como un valor nulo el momento en que la referencia se ha alcanzado, por lo que:

$$\frac{dS(t)}{dt} = 0 \tag{2.39}$$

Al derivar la ecuación 2.38 y reemplazar en la ecuación 2.39 se tiene la siguiente igualdad:

$$\lambda_1 \frac{d\,e(t)}{dt} + \lambda_o\, e(t) = 0 \tag{2.40}$$

Reemplazando la definición del error $e(t) = R(t) - v(t)$, donde $R(t)$ es la referencia de velocidad (v_d), $v(t)$ la velocidad instantánea (v) y $e(t)$ el error en este caso de velocidad lineal $\tilde{v}$, todo esto reemplazando y utilizando la propiedad de linealidad, se tiene:

$$\lambda_1 \left[\frac{d\,v_d}{dt} - \frac{d\,v}{dt}\right] + \lambda_o\, \tilde{v} = 0 \tag{2.41}$$

Al momento de diseñar un control por modos deslizantes, Camacho [16] en su trabajo demuestra que eliminar la derivada de la referencia no tiene impacto en el desempeño del controlador. Sin embargo, recordemos que el lazo externo puede darnos señales de referencia no constantes, incluso se evidenció curvas parecidas a la función radical del tiempo, es por esto que no se simplifica la derivada de la referencia.

Reemplazando la igualdad de la ecuación 2.37 en la 2.41 se tiene:

$$\lambda_1 \left[\frac{d\,v_d}{dt} - \left(\frac{KU}{\tau} - \frac{v}{\tau}\right)\right] + \lambda_o\, \tilde{v} = 0 \tag{2.42}$$

$$\lambda_1 \frac{d\,v_d}{dt} - \lambda_1 \frac{KU}{\tau} + \lambda_1 \frac{v}{\tau} + \lambda_o\, \tilde{v} = 0 \tag{2.43}$$

Obteniendo la siguiente ley de control para la parte continua al despejar U:

$$U = \frac{\tau}{K}\frac{d\,v_d}{dt} + \frac{v}{K} + \frac{\lambda_o\,\tau}{\lambda_1 \cdot K}\,\tilde{v} \tag{2.44}$$

Como se utilizó el modelo de primer orden, siempre se va a tener una respuesta sobre amortiguada, por lo que se escoge los valores que presentan una respuesta adecuada de forma heurística, en este caso:

$$\lambda_1 = 10 \qquad \lambda_o = 30 \tag{2.45}$$

Así entonces la acción de control de la parte continua (U_{con}) queda definida como:

$$U_{Con} = \frac{\tau}{K}\frac{d\,v_d}{dt} + \frac{v}{K} + \frac{3\,\tau}{K}\,\tilde{v} \tag{2.46}$$

2.5.2 Diseño parte discontinua SMC

Para la parte discontinua se emplearía la función signo, donde su argumento es la superficie deslizante, sin embargo, se utiliza la función sigmoide con el fin de reducir el impacto del chattering, como se define en 2.48 se añade una constante K_D que modifica la amplitud de la señal. El parámetro δ se encarga de modificar la función signo y suavizarla como se muestra en la figura 2.11. propuesta por Camacho, Rosales y Rivas [16].

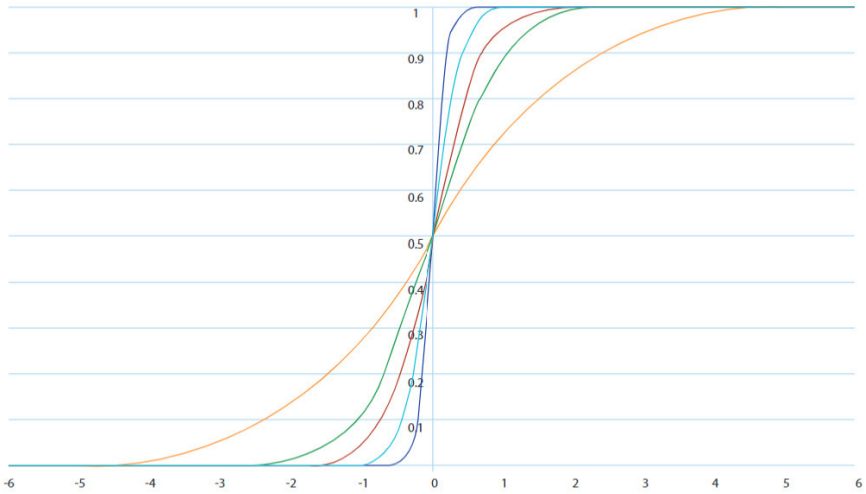


Figura 2.11. Aproximación de la función signo con la función sigmoide

En base a las constantes definidas anteriormente se tiene:

$$S(t) = sign[10\ \ e(t) + 30 \int e(t)\,dt] \tag{2.47}$$

La parte discontinua (U_{Dis}) viene dada por la definición de la función sigmoide donde K_D es la amplitud sintonizable, $S(t)$ la superficie deslizante definida en la ecuación 2.47 y δ un parámetro de sintonización para suavizar mediante la función sigmoide:

$$U_{Dis} = K_D \frac{S(t)}{|S(t)|+\delta} \quad (2.48)$$

Este cambio en la función signo disminuye el efecto de alcanzabilidad, lo que implica una disminución en el desempeño, sin embargo, reduce las oscilaciones de alta frecuencia que podrían dañar al elemento final de control.

Unificando la parte continua y la discontinua disponibles en las ecuaciones 2.46 y 2.48 se obtiene la ley de control U_{SMC} en base a una superficie PI, teniendo en cuenta la definición de la superficie deslizante disponible en la ecuación 2.47:

$$U_{SMC} = \frac{\tau}{K}\frac{d\,v_d}{dt} + \frac{v}{K} + \frac{3\tau}{K}\ \tilde{v} + K_D \frac{S}{|S|+\delta} \quad (2.49)$$

2.5.3 Acción de control para velocidad lineal

En base a la ecuación 2.49 y con los valores encontrados en 2.5, 2.6 y 2.7 se obtiene la acción de control para la velocidad lineal v_c definida por:

$$v_c = 0{,}06097\ \dot{v}_d + v + 0{,}18291\ \tilde{v} + K_D \frac{S(t)}{|S(t)|+\delta} \quad (2.50)$$

Es necesario tomar en cuenta que $K_D\ y\ \delta$ son parámetros de sintonización, v es la velocidad lineal instantánea, $\tilde{v}$ el error de velocidad lineal y $\dot{v}_d$ la derivada de la referencia obtenida como salida del lazo externo para la velocidad lineal.

2.5.4 Acción de control para velocidad angular

En base a la ecuación 2.46, 2.49 y con los valores encontrados en las ecuaciones 2.9, 2.10 y 2.11 se obtiene la acción de control para la velocidad angular definida por:

$$w_C = 0{,}0908\ \dot{w}_d + 0{,}8761w + 0{,}2726\ \tilde{w} + K_D \frac{S}{|S|+\delta} \quad (2.51)$$

Donde w es la velocidad angular instantánea, $\tilde{w}$ el error de velocidad angular y $\dot{w}_d$ la derivada de la referencia obtenida en la salida del lazo externo para la velocidad angular.

2.5.5 Esquema del Predictor de Smith

El esquema de predictor de Smith es un compensador de tiempo muerto, su denominación de predictor ocurre debido a que realimenta la salida sin retardo en base a modelo, una de sus limitaciones es precisamente eso, que se requiere un modelo muy exacto para obtener resultados favorables.

El diagrama de bloques de este esquema se encuentra en la figura 1.21, teniendo esto en cuenta se propone el diagrama del controlador final donde ya se ha acoplado al lazo externo en la figura 2.12:

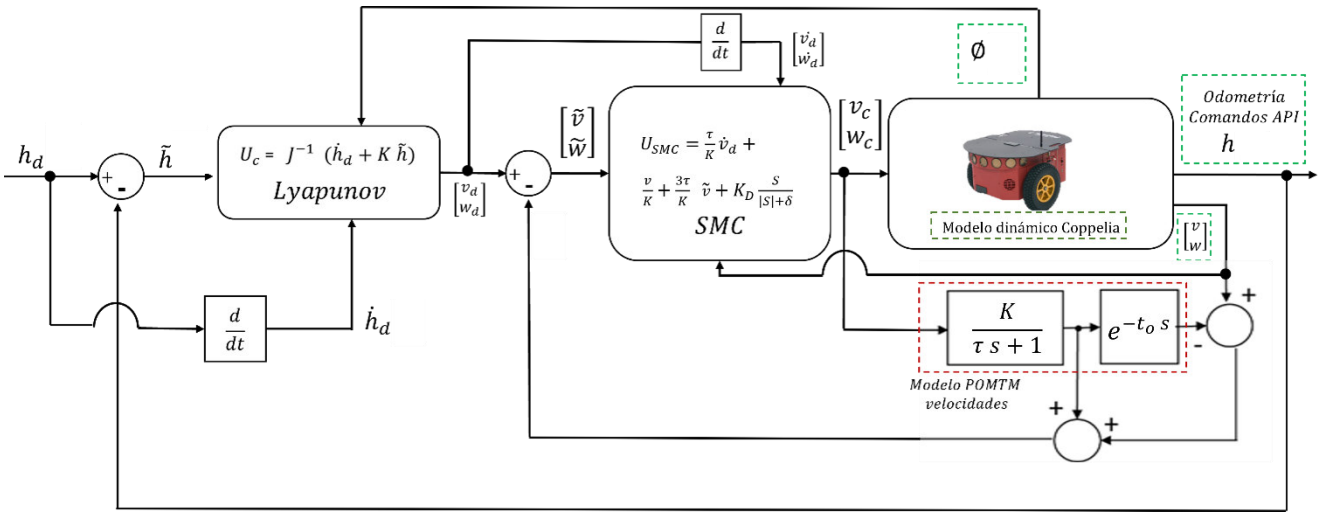


Figura 2.12. Diagrama de bloques para controlador en cascada con lazo externo basado en un postulado de Lyapunov y lazo interno basado en un controlador SMC + SP.

2.6 DISCRETIZACIÓN DE CONTROLADORES

Para la discretización de los controladores se usarán las leyes de control desarrolladas en los apartados anteriores y el método de discretización de Tustin desarrollado en el apartado 1.4.6. donde se requiere utilizar el valor del tiempo de muestreo $\boldsymbol{T_m}$.

La idea es encontrar las ecuaciones en diferencias que permitan implementar el código en el Tomo II de este trabajo, donde el algoritmo de control se comunicará de manera sincrónica con el simulador.

2.6.1 Tiempo de muestreo

Como se muestra en la figura 2.13, las señales deben ser muestreadas, con el fin de ser digitalizadas, para esto se debe escoger un tiempo de muestreo T_m adecuado. En la literatura se puede encontrar el siguiente criterio [8]:

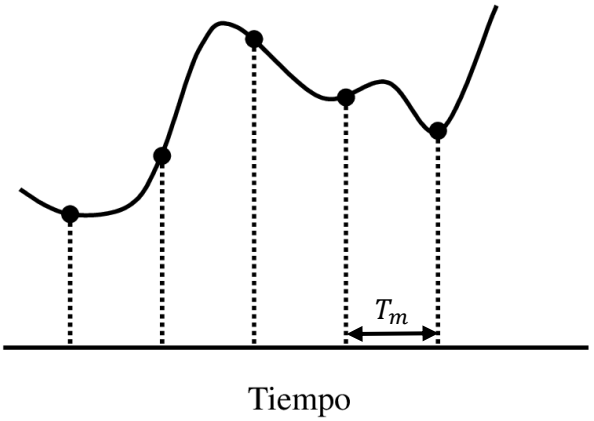


Figura 2.13. Discretización de una señal continua

- Una señal en lazo abierto, que, a partir de su tiempo de establecimiento (t_s), se debe muestrear de 30 a 50 veces.

  Dándonos la siguiente relación, en el caso más utilizado:

$$30\, T_m = t_s \tag{2.52}$$

Tomando en cuenta que vamos a partir de la constante de tiempo τ de las velocidades en lazo abierto y la definición de tiempo de establecimiento para un sistema de primer orden al 98%, se tiene:

$$30\, T_m = 4\, \tau \tag{2.53}$$

$$T_m = \frac{2}{15}\, \tau \tag{2.54}$$

Para el caso de la velocidad lineal en base al valor encontrado en la ecuación 2.6 se tiene el siguiente tiempo de muestreo T_{mv} :

$$T_{mv} = \frac{2}{15}\ (0{,}06097) = 8{,}129\ \ ms \tag{2.55}$$

Para la velocidad angular en base al valor encontrado en la ecuación 2.10 se tiene el siguiente tiempo de muestreo T_{mw} :

$$T_{mw} = \frac{2}{15}\ (0{,}10374) = 13{,}832\ ms \tag{2.56}$$

Además, tomando en cuenta que el software CoppeliaSim permite pasos de simulación de $0{,}01$ y $0{,}05$ segundos, es evidente que la elección más acertada para representar de manera fiel las señales involucradas en el lazo interno (Variables más rápidas) sería la siguiente:

$$T_m = 10\ ms \tag{2.57}$$

2.6.2 CONTROL EN LAZO SIMPLE (LYAPUNOV)

Primero se discretizará el control en lazo simple obtenido en el apartado 2.3, donde partiremos de la ley de control obtenida en la ecuación 2.26:

$$U_c = \begin{bmatrix} v_c \\ w_c \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ \dfrac{-\sin(\phi)}{0.2} & \dfrac{\cos(\phi)}{0.2} \end{bmatrix} \left(\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \end{bmatrix} + K \begin{bmatrix} x_d - x \\ y_d - y \end{bmatrix} \right) \tag{2.58}$$

Para discretizar la ecuación anterior es necesario la derivada de los valores deseados de posición $(\dot{x}_d, \dot{y}_d)$ por lo que se usará el equivalente discreto de la derivada usando el método de Tustin. Aplicando la transformada de Laplace se tiene:

$$L\{\dot{x}_d\} = s\ \ X_d(s) \tag{2.59}$$

El siguiente paso es remplazar s por su equivalente de Tustin descrito en la ecuación 1.29 y definir una nueva variable discreta $X_{dp}\ (z)$:

$$X_{dp}(z) = \frac{2}{T_m}\ \ \frac{z-1}{z+1}\ X_d(z) \tag{2.60}$$

$$X_{dp}(z)\quad (1 + z^{-1}) = X_d(z)\ (1 - z^{-1})\ \frac{2}{T_m} \tag{2.61}$$

Usando las propiedades de la transformada Z para pasar a tiempo discreto k, se modifica la ecuación 2.61 obteniendo la ecuación en diferencias de la derivada $X_{dp}[k]$

$$X_{dp}\,[k] = \frac{2}{T_m}(X_d[k] - X_d[k-1]) - \; X_{dp}\,[k-1] \tag{2.62}$$

Usando el desarrollo previo también se puede definir la derivada $Y_{dp}[k]$ de la misma manera obteniendo:

$$Y_{dp}[k] = \frac{2}{T_m}(Y_d[k] - Y_d[k-1]) - \; Y_{dp}[k-1] \tag{2.63}$$

Usando las ecuaciones 2.62 y 2.63 se define la ley de control discreta del control de lazo simple como:

$$\begin{bmatrix} v_c\,[k] \\ w_c\,[k] \end{bmatrix} = \begin{bmatrix} \cos(\phi[k]) & \sin(\phi[k]) \\ \dfrac{-\sin(\phi[k])}{0.2} & \dfrac{\cos(\phi[k])}{0.2} \end{bmatrix} \left(\begin{bmatrix} \dfrac{2}{T_m}(X_d[k] - X_d[k-1]) - \; X_{dp}[k-1] \\ \dfrac{2}{T_m}(Y_d[k] - Y_d[k-1]) - \; Y_{dp}[k-1] \end{bmatrix} + K \begin{bmatrix} x_d[k] - x[k] \\ y_d[k] - y[k] \end{bmatrix} \right) \tag{2.64}$$

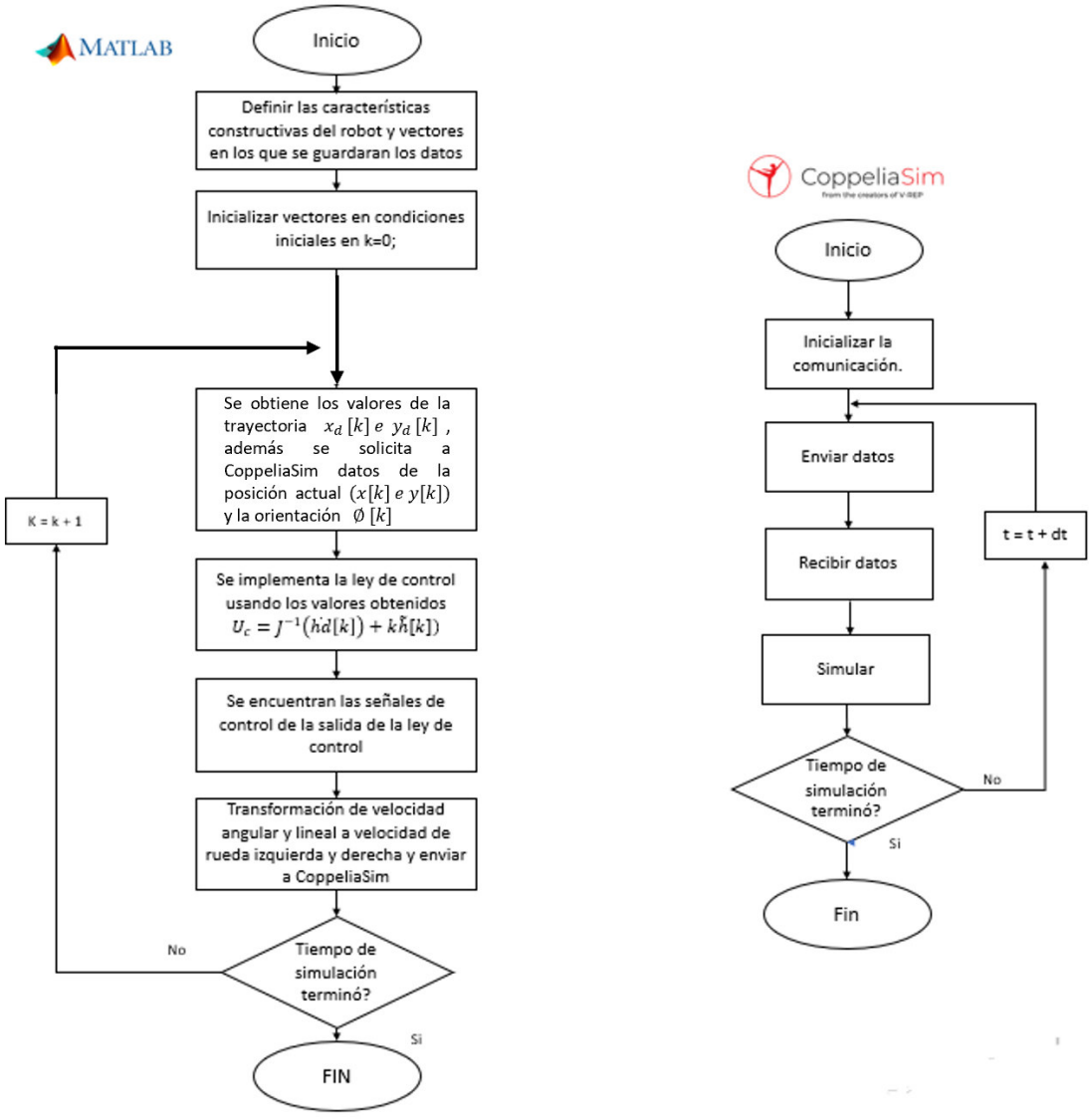


Figura 2.14. Diagrama de flujo del control en lazo simple (LYAPUNOV)

En la figura 2.14 se muestra el diagrama de flujo para la Implementación de esta ley de control. Nótese que se llevan a cabo de manera sincrónica un algoritmo para MATLAB y otro para CoppeliaSim.

El algoritmo de MATLAB se encarga de adquirir datos y procesar con la información del muestreo anterior una nueva ley de control para el tiempo $k+1$, utilizando la expresión 2.64. Una vez se tenga el valor de la señal de control en ese instante se envía al programa CoppeliaSim mediante comandos API y se entrega una señal de disparo o "Trigger", que es la encargada de avisar al simulador que tiene que procesar y avanzar un paso de simulación.

2.6.3 CONTROL EN CASCADA: LAZO INTERNO (PI)

Para la discretización del control de lazo interno PI partiremos de la ley de control diseñada en el apartado 2.4, de la cual se tiene que $\tilde{v} = v_d - v$ y $\tilde{w} = w_d - w$, con las siguientes leyes de control asociadas en tiempo continuo ($v_c \; y \; w_c$):

$$\begin{aligned} v_c &= 1.811\tilde{v} + 17.93 \int \tilde{v}\, dt \\ w_c &= 2.9\, \tilde{w} + 27.07 \int \tilde{w}\, dt \end{aligned} \tag{2.65}$$

Para discretizar la ecuación anterior es necesario la integral del error de velocidad tanto lineal como angular por lo que se usará el equivalente a la integral usando el método de Tustin. Primero se definirá la integral del error de velocidad en tiempo continuo como:

$$\int \tilde{v}\,(t)\, dt = \frac{1}{s}\tilde{v}(s) \tag{2.66}$$

El siguiente paso es remplazar s por su equivalente de Tustin descrito en la ecuación 1.29, definiendo la integral del error en su equivalente discreto $\tilde{V}_i(z)$:

$$\tilde{V}_i(z) = \tilde{V}(z)\ \frac{T_m}{2}\frac{(1+z^{-1})}{(1-z^{-1})} \tag{2.67}$$

$$\tilde{V}_i(z)\quad (1-z^{-1}) = \tilde{V}(z)\ \ (1+z^{-1})\ \ \frac{T_m}{2} \tag{2.68}$$

Usando las propiedades de la transformada Z para pasar a tiempo discreto k se modifica la ecuación 2.68 obteniendo la ecuación en diferencias de la integral del error, la cual es representada por $\tilde{V}_i[k]$:

$$\tilde{V}_i[k] = \frac{T_m}{2}(\tilde{v}[k] + \tilde{v}[k-1]) + \tilde{V}_i[k-1] \tag{2.69}$$

Usando el desarrollo previo, también se puede definir la integral del error de w en el dominio z, siendo $\widetilde{W}_i\ (z)$, obteniendo la siguiente expresión:

$$\widetilde{W}_i[k] = \frac{T_m}{2}(\tilde{w}[k] + \tilde{w}[k-1]) + \widetilde{W}_i\ [k-1] \tag{2.70}$$

Usando las ecuaciones 2.69 y 2.70 se define la ley de control discreta del control PI para la velocidad angular y lineal.

$$\begin{aligned} v_c[k] &= 1.811\tilde{v}[k] + 17.93\left(\frac{T_m}{2}(\tilde{v}[k] + \tilde{v}[k-1]) + \tilde{V}_i[k-1]\right) \\ w_c\ [k] &= 2.9\ \tilde{w}[k] + 27.07\left(\frac{T_m}{2}(\tilde{w}[k] + \tilde{w}[k-1]) + \widetilde{W}_i\ [k-1]\right) \end{aligned} \tag{2.71}$$

En la figura 2.15 se muestra el diagrama de flujo para la implementación del algoritmo asociado a este control en cascada.

2.6.4 CONTROL EN CASCADA: LAZO INTERNO (SMC + SP)

En este apartado únicamente se discretizará el control SMC + Predictor de Smith del lazo interno además de los modelos de primer orden asociados a la velocidad lineal y angular que se requieren discretizar para la Implementación del SP

Para la discretización del control de lazo interno partiremos de la ley de control diseñada en el subcapítulo 2.5 en las ecuaciones 2.50 y 2.51:

$$\begin{aligned} v_c &= 0{,}06097\ \dot{v}_d + v + 0{,}18291\ \tilde{v} + K_{Dv}\frac{S(t)}{|S(t)| + \delta} \\ w_c &= 0{,}0908\ \dot{w}_d + 0{,}876\ w + 0{,}2726\ \tilde{w} + K_{Dw}\frac{S(t)}{|S(t)| + \delta} \end{aligned} \tag{2.72}$$

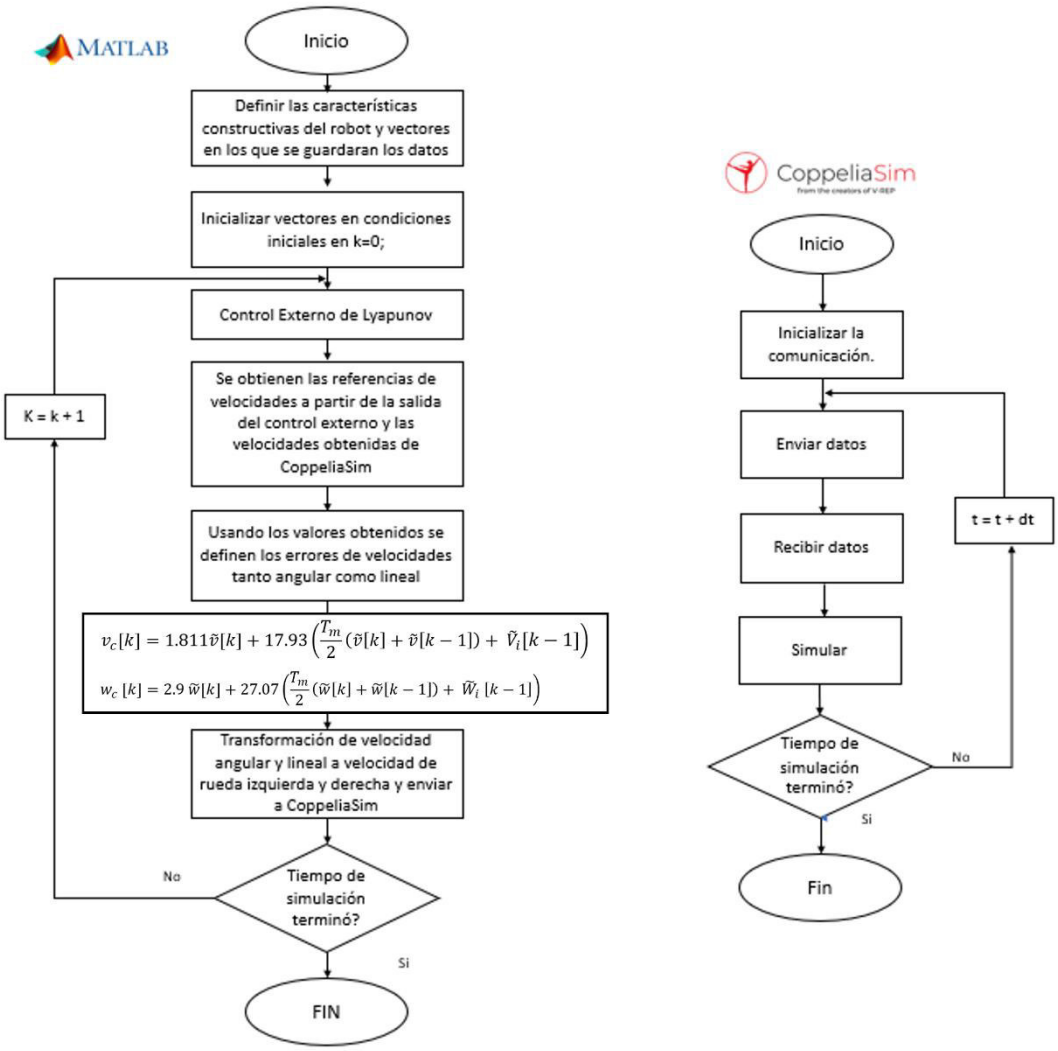


Figura 2.15. Diagrama de flujo del Control en cascada (Lyapunov + PI)

Siendo:

v_c la acción de control de la velocidad lineal

w_c la acción de control de la velocidad angular

v_d la referencia de velocidad lineal

w_d la referencia de velocidad angular

$\tilde{v}$ error de velocidad lineal

$\tilde{w}$ error de velocidad angular

K_{Dv} la ganancia de la parte discontinua asociada a la velocidad lineal

K_{Dw} la ganancia de la parte discontinua asociada a la velocidad angular

$S(t)$ la superficie deslizante

δ término de suavizamiento para sigmoide

Por facilidad se separará las señales de control de la ecuación 2.72 en U_{Con} la cual es la parte continua y U_{Dis} la cual es la parte discontinua. Será también necesario definir específicamente para la velocidad lineal y angular como U_{cv}, U_{Dv}, U_{cw} y U_{Dw} respectivamente.

$$v_c = \overbrace{0{,}06097\,\dot{v}_d + v + 0{,}18291\,\tilde{v}}^{U_{cv}} + \overbrace{K_{Dv}\frac{S(t)}{|S(t)| + \delta}}^{U_{Dv}}$$

$$w_c = \underbrace{0{,}0908\,\dot{w}_d + 0{,}876\,w + 0{,}2726\,\tilde{w}}_{U_{cw}} + \underbrace{K_{Dw}\frac{S(t)}{|S(t)| + \delta}}_{U_{Dw}} \tag{2.73}$$

Debido a que la acción de control involucra la superficie deslizante, se procede a realizar el proceso de discretización de esta, utilizando el método de Tustin como en el apartado anterior, teniendo la siguiente ecuación en diferencias para un error genérico $e(t)$:

$$S(t) = sign\left[10e(t) + 30\int e(t)\,dt\right] \tag{2.74}$$

Definimos el equivalente discreto de la integral del error genérico $e_i[k]$ como en el apartado anterior:

$$e_i\,[k] = \left(\frac{T_m}{2}(e[k] + e[k-1]) + e_i[k-1]\right)$$

Teniendo la siguiente ecuación en diferencias de la superficie deslizante: (2.75)

$$S[k] = sign\left[10\,e[k] + 30\left(\frac{2}{T_m}(e[k] + e[k-1])\right) + e_i[k-1])\right]$$

Para discretizar las derivadas e integrales de la ecuación 2.73 de la parte continua se usarán los mismos procedimientos, reemplazando el operador s por su equivalente de Tustin y transformándolo en una ecuación en diferencias obteniendo:

$$U_{Cv}[k] = 0.06097\left(\frac{2}{T_m}(v_d[k] - v_d[k-1]) - \dot{v}_d[k-1]\right) + v[k] + 0.18291\ \tilde{v}[k] \quad (2.76)$$

$$U_{C\omega}[k] = 0.0908\left(\frac{2}{T_m}(w_d[k] - w_d[k-1]) - \dot{w}_d[k-1]\right) + 0.8761w[k] + 0.2726\ \tilde{w}[k] \quad (2.77)$$

Tomando en cuenta que ya se ha discretizado la superficie deslizante en la ecuación 2.75, se realiza el proceso de discretización para la parte discontinua:

$$U_{Dv}[k] = K_{Dv}\ \frac{S[k]}{S[k] + \delta_v} \quad (2.78)$$

$$U_{D\omega}[k] = K_{D\omega}\ \frac{S[k]}{S[k] + \delta_\omega} \quad (2.79)$$

Usando las ecuaciones 2.76, 2.77, 2.78 y 2.79 se define la ley de control discreta del control SMC para la velocidad angular ($w_c\ [k]$) y lineal ($v_c\ [k]$) como:

$$v_c[k] = U_{Cv}[k] + U_{Dv}[k] \quad (2.80)$$

$$w_c[k] = U_{C\omega}[k] + U_{D\omega}[k]$$

2.6.4.1 Discretización del Modelo de primer orden de la velocidad lineal

Para el diseño del Predictor de Smith, se requiere el modelo discretizado. De igual forma que las derivadas e integrales vistas anteriormente se realizará la discretización de los modelos de primer orden más tiempo muerto con el método de Tustin por lo que se realizará a partir del modelo de la velocidad lineal.

$$G_m(s)^- = \frac{v_m(s)}{U(s)} = \frac{1}{0.0609s + 1} \quad (2.81)$$

Siendo:

$v_m(s)$ la velocidad lineal actual de modelo

$U(s)$ la entrada

Para la discretización se usará el modelo sin retardo $G_m(s)^-$ ya que al trabajar en el plano z los retardos pueden ser expresados en base al tiempo de muestreo únicamente.

Usando el método de Tustin se procede a cambiar todas las $s \rightarrow \frac{2}{T_m}\frac{(1-z^{-1})}{(1+z^{-1})}$ obteniendo el siguiente modelo en el dominio de Z.

$$G_m(z)^{\text{-}} = \frac{v_m(z)}{U(z)} = \frac{1}{0.0609\left(\frac{2}{T_m}\frac{(1-z^{-1})}{(1+z^{-1})}\right)+1} \tag{2.82}$$

$$\frac{v_m(z)}{U(z)} = \frac{T_m\,(1+z^{-1})}{0.0609 * 2\,(1-z^{-1}) + T_m * (1+z^{-1})} \tag{2.83}$$

Finalmente, para hallar la expresión necesaria, se despeja $v_m(z)$ en base a la entrada discretizada $U(z)$ y se la pasa una ecuación en diferencias, donde:

$$v_m[k] = \frac{T_m(U[k]+U[k-1]) - (T_m - 0.1218)\;\; v_m[k-1]}{T_m + 0.1218} \tag{2.84}$$

2.6.4.2 Discretización del Modelo de primer orden de la velocidad angular

De acuerdo con el razonamiento anterior se procede a seguir el mismo procedimiento con el modelo de la velocidad angular:

$$G_m(s)^{\text{-}} = \frac{w_m(s)}{W(s)} = \frac{1.14136}{0.10374s+1} \tag{2.85}$$

Siendo:

$w_m(s)$ la velocidad angular actual del modelo

$W(s)$ la entrada

Usando el método de Tustin se procede a cambiar todas las $s \rightarrow \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}$ obteniendo el siguiente modelo en Z.

$$G_m(z)^{\text{-}} = \frac{w_m(z)}{W(z)} = \frac{1.14136}{0.10374\left(\frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}\right)+1} \tag{2.86}$$

$$\frac{w_m(z)}{W(z)} = \frac{1.14136\, T_m * (1 + z^{-1})}{0.10374 * 2 * (1 - z^{-1}) + T_m * (1 + z^{-1})} \tag{2.87}$$

Finalmente, para hallar la expresión necesaria, se despeja $w_m(z)$ en base a la entrada discretizada $W(z)$ y se la pasa una ecuación en diferencias, donde:

$$w_m[k] = \frac{1.14136\, T_m\, (W[k] + W[k-1]) - (T_m - 0.20748)\, w_m[k-1]}{(T_m + 0.20748)} \tag{2.88}$$

2.6.4.3 Consideraciones del tiempo muerto

Es necesario tomar en cuenta tanto la parte del modelo sin retardo $G_m(s)^-$ como la parte de retardo $G_m(s)^+$ el cuál consisten únicamente del término $e^{-t_o\, s}$ y debido a la importancia que tiene su uso el esquema de Predictor de Smith no es posible despreciarlo por lo que se deberá añadir la discretización de este dentro del algoritmo:

$$G_m(s)^+ = \ e^{-t_o\, s} \tag{2.89}$$

La transformada Z de la función de retardo se puede expresar como:

$$G_m(z)^+ = \ z^{-t_m} \tag{2.90}$$

Donde t_m es el entero inmediato superior a la expresión:

$$t_m = \ \frac{t_o}{T_m} = \frac{0{,}030291}{0{,}01} \approx 4 \tag{2.91}$$

Esto significa que se requerirá tener una salida de cero en los primeros t_m (4) pasos de muestreo antes de tener la salida prevista para el paso 1 del modelo. En la figura 2.16 se muestra el diagrama de flujo del algoritmo para el control en cascada con lazo interno basado en SMC+SP.

En el ANEXO II, se tiene el diagrama de bloques del controlador final discretizado, donde se encuentra este diseño acoplado al lazo externo. Este diagrama servirá como punto de partida en el Tomo II para implementar la comunicación y envío de la señal de control adecuada al simulador en CoppeliaSim.

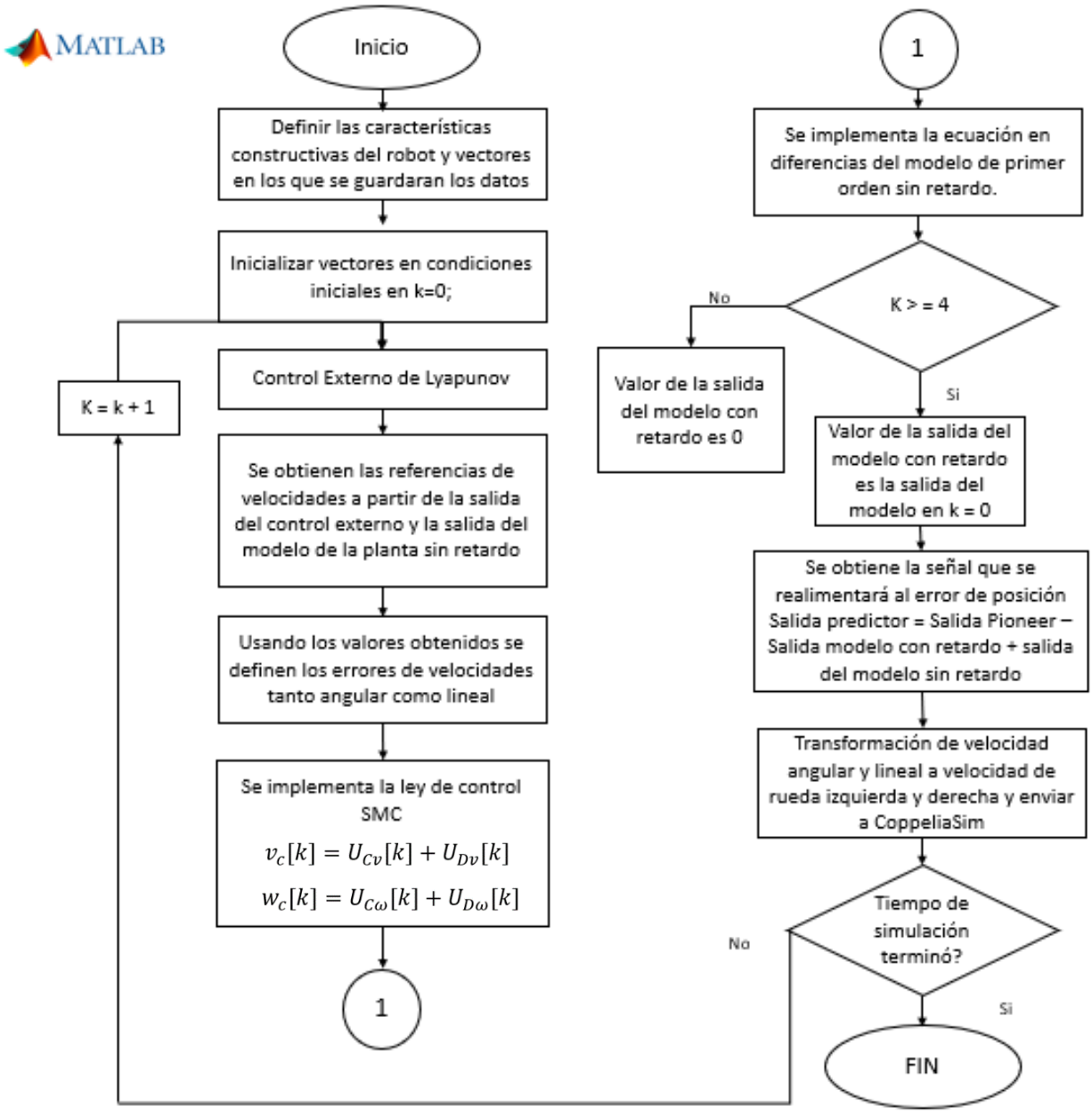


Figura 2.16. Diagrama de flujo de Control en cascada (Lyapunov + SMC + SP)

3 CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES

- En este trabajo se determinaron los modelos cinemáticos y el modelo dinámico de un robot móvil, pues se obtuvieron expresiones en forma de ecuaciones diferenciales que dependían de las coordenadas, orientación, velocidades y características constructivas en un robot tipo uniciclo.

- Se pudo establecer el procedimiento para el diseño de un controlador basado en un candidato de Lyapunov que permita asegurar la estabilidad en una región de operación específica en sistemas autónomos. El procedimiento demanda generar un desarrollo matemático que permita proponer una ley de control que cumpla ciertas condiciones de estabilidad y seguimiento de referencia. Es importante destacar que este método puede ser aplicado para sistemas no lineales donde exista más de una variable a controlar.

- Utilizando comandos API y señales tipo paso entre MATLAB y CoppeliaSim, se pudo identificar y validar el modelo de primer orden más tiempo muerto (POMTM) del Pioneer 3DX disponible en CoppeliaSim, donde las salidas fueron las velocidades lineal y angular del robot. En este trabajo se obtuvo errores de modelado muy bajos reflejados por un índice ISE en base a la salida real del simulador y la salida del modelo.

- En este trabajo se diseñó un controlador tipo cascada, con un lazo externo basado en Lyapunov y un lazo interno de naturaleza PI, la sintonización del controlador PI pudo ser obtenida mediante el método de Ziegler Nichols en lazo abierto.

- Los beneficios que se buscan obtener del controlador SMC+SP son la robustez y seguimiento que aporta el control por modos deslizantes y la compensación de retardos que brinda el Predictor de Smith.

- La desventaja en el diseño del controlador SMC+SP es que se requiere de un modelo muy preciso que efectivamente realimente una señal apropiada y compense el retardo de la planta real.

- Se evidenció que la selección de un tiempo de muestreo adecuado es de gran importancia al momento de diseñar y discretizar controladores, pues permite evitar el “aliasing”, fenómeno que ocurre cuando se toma muestras de manera muy lenta, perdiendo información y reflejando de manera poco fiel a las señales involucradas.

3.2 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.
- Es recomendable que la distancia a considerar para mejorar la restricción no holonómica (a) se encuentre en un valor entre 0,1 y 0,3 metros, pues acorde a las dimensiones del robot Pioneer 3DX, es un valor adecuado para calcular con anticipación sus movimientos traslacionales.
- Es ideal siempre trabajar en modo sincrónico al momento de la comunicación entre MATLAB Y CoppeliaSim, pues no se requiere considerar tiempos de procesamiento del algoritmo o del simulador ya que el disparo generará siempre un paso de simulación fijo y no continuará hasta que este se cumpla por completo.

4 REFERENCIAS BIBLIOGRÁFICAS - TOMO I

[1] Spyros G. Tzafestas, “Introduction to mobile robot control,” vol. 1, Mobile Robots application, Ed., 2nd ed. New York, NY, USA: ELSEVIER, 2013, pp. 11-18. [Online]. Available: https://books.google.com.ec/books/about/Introduction_to_Mobile_Robot_Contr

[2] L. Capito and P. Proaño, “Seguimiento de trayectorias mediante cuatro técnicas de control utilizando una plataforma robótica Pioneer 3DX y el sistema operativo robótico ROS”, Escuela Politécnica Nacional, Quito, 2015

[3] B. Sarabia, “Diseño y simulación de tres técnicas de control clásicas y robustas aplicadas al seguimiento de trayectorias ante la presencia de retardos fijos para la plataforma robótica Pioneer 3DX”, Escuela Politécnica Nacional, Quito, 2017.

[4]. Coppelia Robotics “Remote API functions (MATLAB)” Package Version 2.0, Jan. 09, 2021. [Online]. Available: https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm

[5] B. Siliciano, A. De Luca, “Advances in control of articulated and mobile robots,” vol. 1, Mobile Robots application, Ed., 3rd ed. Milano, Italy: Springer, 2004, pp. 06-11. [Online]. Available: https://books.google.com.ec/books/about/Introduction_to_Mobile_Robot_Contr

[6] C. Fischer, “SLAM for Pedestrians and Ultrasonic Landmarks in Emergency Response Scenarios,” Science Direct., vol. 13, no. 1, pp. 01-03, Oct. 2014.

[7] Addept Mobile Robots “Pioneer 3DX Manual” Package RevA, Jan. 09, 2021. [Online]. Available: https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf

[8] A. De Luca, G. Oriolo “Modeling and control of nonholonomic mechanical systems”, vol. 1, Controllability, Ed., 1st ed. Roma, Italy: Roma University, 2011, pp. 278-291. [Online]. Available: https://www.diag.uniroma1.it/~labrob/pub/papers/CISM95.pdf

[9] M. Egerstedt “Control of mobile robots course”, Differential Drive robots, Georgia Tech, Georgia, USA, 2013.

[10] De La Cruz, C., & Carelli, R. (2006). “Dynamic Modeling and Centralized Formation Control of Mobile Robots”. IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics.

[11] Y. Zhang, D. Hong, J. H. Chung, and S. A. Velinsky, “Dynamic Model Based Robust Tracking Control of a Differentially Steered Wheeled Mobile Robot,” Proceedings of the American Control Conference, Philadelphia, Pennsylvania, June 1998, pp. 850-855.

[12] A. Rosales, G. Scaglia. (2007). “Control dinámico mediante métodos numéricos para robots móviles tipo uniciclo”. Instituto de Automática (INAUT). Universidad Nacional de San Juan, Argentina.

[13] Z. Dayeta, “Lyapunov’s Second Method for Estimating Region of Asymptotic Stability”, Etiopy: Haramaya University (1994)

[14] O. Camacho, “A new approach to design and tune sliding mode controllers for chemical processes”, Tampa: University of Florida, (1996).

[15] F. de la Cruz, O. Camacho “Controlador de Modos Deslizantes basado en Predictor de Smith y Modelo de Segundo Orden para Procesos con Elevado Retardo”, Revista Politécnica, Vol. 21 (2015).

[16] O. Camacho, A. Rosales, F. Rivas “Control de Procesos”, Ed., 1st ed. Quito, Ecuador: Escuela Politécnica Nacional, 2015, pp. 114-176.

[17] VM. Alfaro, “Identificación de modelos de orden reducido a partir de la curva de reacción del proceso”, Departamento de Automática, Escuela de Ingeniería Eléctrica Universidad de Costa Rica, 2060 Costa Rica.

5 ANEXOS

ANEXO I

Identificación velocidad lineal

```
clear all;
clc;

vrep = remApi('remoteApi');
vrep.simxFinish(-1);
clientID = vrep.simxStart('127.0.0.1', 19997, true, true, 5000, 5);
vrep.simxSynchronous(clientID,true); %// Enable the synchronous mode (Blocking
function call)
vrep.simxStartSimulation(clientID,vrep.simx_opmode_blocking);
velx=[];
vely=[];
Tm = 0.01;
R = 0.195/2; % Radio de las ruedas
L = 0.381; % Longitud entre las ruedas
if (clientID>-1)
disp('connected')
%Handle
[returnCode,left_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vr
ep.simx_opmode_blocking);
[returnCode,right_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMotor',
vrep.simx_opmode_blocking); %Code to Access Motors
[~,Pioneer]=
vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx',vrep.simx_opmode_blocking);


% The first simulation step waits for a trigger before being executed

for i= 1:500
[returnCode,vlineal,vangular]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking);
vrep.simxSynchronousTrigger(clientID); %// Trigger next simulation step
t(i) = i*Tm;
ref(i) = 0.4;
uc(i) = 0.4;
wc(i) = 0;
rv(i) = (2*uc(i)+wc(i)*L)/(2*R);
lv(i) = (2*uc(i)-wc(i)*L)/(2*R);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, rv(i),
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor,  lv(i),
vrep.simx_opmode_oneshot);    %Rueda izquierda
velx(i) = vlineal(1);
vely(i) = vlineal(2);
vel(i) = sqrt((velx(i))^2+(vely(i))^2);

end
```

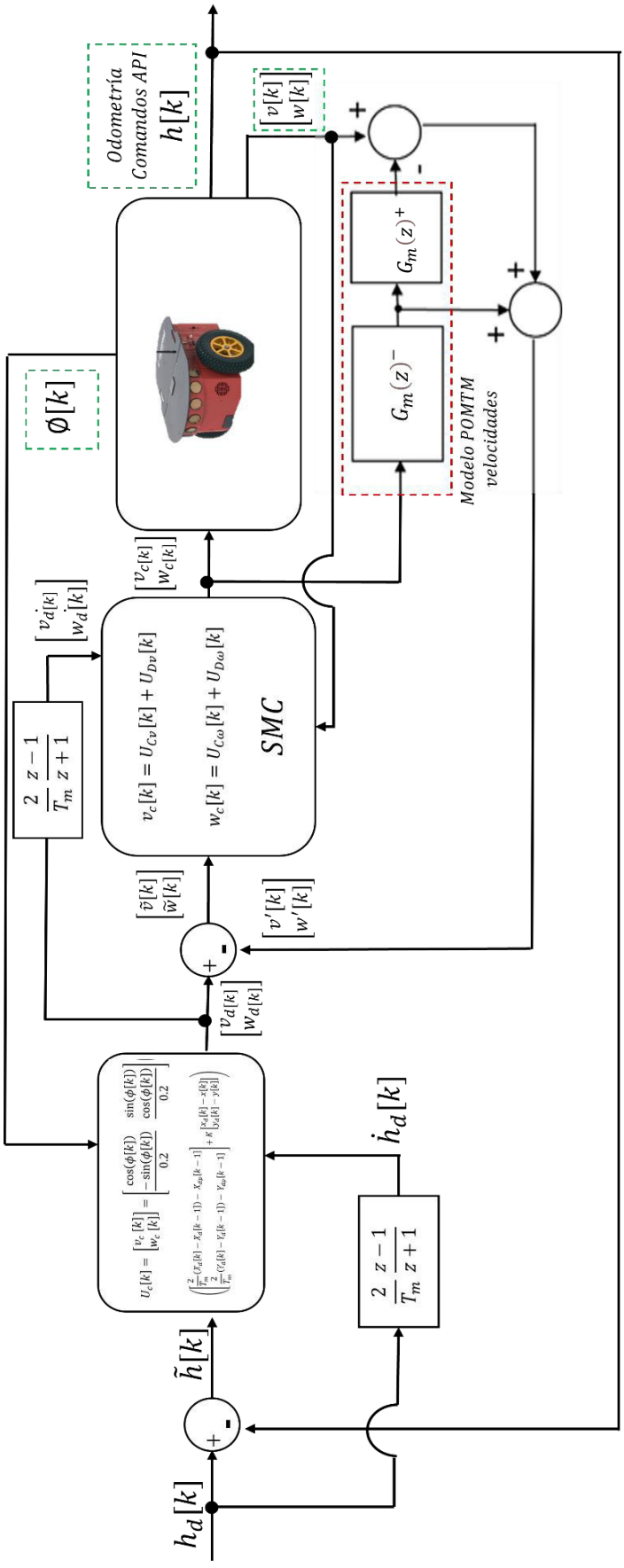
```
for i= 500:1000
[returnCode,vlineal,vangular]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking);
vrep.simxSynchronousTrigger(clientID); %// Trigger next simulation step (Blocking
function call)
t(i) = i*Tm;
ref(i) = 0.6;
uc(i) = 0.6;
wc(i) = 0;
rv(i) = (2*uc(i)+wc(i)*L)/(2*R);
lv(i) = (2*uc(i)-wc(i)*L)/(2*R);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, rv(i),
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor,  lv(i),
vrep.simx_opmode_oneshot);    %Rueda izquierda
velx(i) = vlineal(1);
vely(i) = vlineal(2);
vel(i) = sqrt((velx(i))^2+(vely(i))^2);
end
plot(t,ref,t,vel);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, 0,
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, 0,
vrep.simx_opmode_oneshot);    %Rueda izquierda
vrep.simxSynchronousTrigger(clientID);
vrep.simxStopSimulation(clientID,vrep.simx_opmode_blocking);
disp("Comunicacion con SIMULINK finalizada");
end
vrep.simxFinish(-1);
vrep.delete();
```

Identificación velocidad angular

```
clear all;
clc;
vrep = remApi('remoteApi');
vrep.simxFinish(-1);
clientID = vrep.simxStart('127.0.0.1', 19997, true, true, 5000, 5);
vrep.simxSynchronous(clientID,true); %// Enable the synchronous mode (Blocking function
call)
vrep.simxStartSimulation(clientID,vrep.simx_opmode_blocking);
velang=[];
Tm = 0.01;
R = 0.195/2; % Radio de las ruedas
L = 0.381; % Longitud entre las ruedas
if (clientID>-1)
disp('connected')
%Handle
[returnCode,left_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.
simx_opmode_blocking);
[returnCode,right_motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMotor',vre
p.simx_opmode_blocking); %Code to Access Motors
[~,Pioneer]=
vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx',vrep.simx_opmode_blocking);
```

```
% The first simulation step waits for a trigger before being executed
for i= 1:500
[returnCode,vlineal,vangular]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking);
vrep.simxSynchronousTrigger(clientID);
t(i) = i*Tm;
ref(i) = 0.4;
uc(i) = 0;
wc(i) = 0.4;
rv(i) = (2*uc(i)+wc(i)*L)/(2*R);
lv(i) = (2*uc(i)-wc(i)*L)/(2*R);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, rv(i),
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor,  lv(i),
vrep.simx_opmode_oneshot);    %Rueda izquierda
velang(i) = vangular(3);
end
for i= 500:1000
[returnCode,vlineal,vangular]=vrep.simxGetObjectVelocity(clientID, Pioneer,
vrep.simx_opmode_blocking);
vrep.simxSynchronousTrigger(clientID);
t(i) = i*Tm;
ref(i) = 1;
uc(i) = 0;
wc(i) = 1;
rv(i) = (2*uc(i)+wc(i)*L)/(2*R);
lv(i) = (2*uc(i)-wc(i)*L)/(2*R);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, rv(i),
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor,  lv(i),
vrep.simx_opmode_oneshot);    %Rueda izquierda
velang(i) = vangular(3);
end
plot(t,ref,t,velang);
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_motor, 0,
vrep.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_motor, 0,
vrep.simx_opmode_oneshot);    %Rueda izquierda
vrep.simxSynchronousTrigger(clientID);
vrep.simxStopSimulation(clientID,vrep.simx_opmode_blocking);
disp("Comunicacion con SIMULINK finalizada");
end
vrep.simxFinish(-1);
vrep.delete();
```

ANEXO II


Odometría
Comandos API
$h[k]$
$\phi[k]$
$\begin{bmatrix} v[k] \\ w[k] \end{bmatrix}$
$G_m(z)^+$
$G_m(z)^-$
Modelo POMTM
velocidades
$\begin{bmatrix} v_c[k] \\ w_c[k] \end{bmatrix}$
$\begin{bmatrix} v_d[k] \\ w_d[k] \end{bmatrix}$
$\frac{2}{T_m}\frac{z-1}{z+1}$
$v_c[k] = U_{Cv}[k] + U_{Dv}[k]$
$w_c[k] = U_{C\omega}[k] + U_{D\omega}[k]$
SMC
$\begin{bmatrix} \tilde{v}[k] \\ \tilde{w}[k] \end{bmatrix}$
$\begin{bmatrix} v'[k] \\ w'[k] \end{bmatrix}$
$\begin{bmatrix} v_d[k] \\ w_d[k] \end{bmatrix}$
$U_c[k] = \begin{bmatrix} v_c[k] \\ w_c[k] \end{bmatrix} = \begin{bmatrix} \cos(\phi[k]) & \sin(\phi[k]) \\ -\frac{\sin(\phi[k])}{0.2} & \frac{\cos(\phi[k])}{0.2} \end{bmatrix} \begin{pmatrix} \frac{T_s^2}{T_m}(X_d[k] - X_{ab}[k-1]) - X_{ab}[k-1] \\ \frac{T_s^2}{T_m}(Y_d[k] - Y_{ab}[k-1]) - Y_{ab}[k-1] \end{pmatrix} + K \begin{bmatrix} X_d[k] - x[k] \\ Y_d[k] - y[k] \end{bmatrix}$
$\tilde{h}[k]$
$h_d[k]$
$\dot{h}_d[k]$
$\frac{2}{T_m}\frac{z-1}{z+1}$