UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

CONTROL DE MÚLTIPLES ROBOTS NO HOLONÓMICOS BAJO UN ESQUEMA LÍDER SEGUIDOR

TESIS PARA OBTENER EL GRADO DE

MAESTRO EN ROBÓTICA

PRESENTA:

ING. EXON FERNANDO VILLALOBOS ÁLVAREZ

DIRECTOR:

DR. OSCAR DAVID RAMÍREZ CÁRDENAS

H. CD. DE HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, NOVIEMBRE 2022

Sinodales:
Dr. José Aníbal Arias Aguilar
Dr. Jesús Linares Flores
Dr. Fermín Hugo Ramírez Leyva
Dr. Ricardo Tapia Herrera

Director de Tesis:
Dr. Oscar David Ramírez Cárdenas

Dedicatoria

Con mucho cariño a Iliana Rodríguez que siempre aligeró la carga en mis hombros, nunca dudó de mí y me mostró que no hay nada que no pueda hacer. Fuiste y sigues siendo mi más grande soporte.

Agradecimientos

A mi director de tesis Oscar Ramírez, que supo guiarme, responder mis dudas y alentarme siempre en el desarrollo de esta tesis.

A mi madre Yamileth Álvarez que supo brindarme su apoyo cuando más lo necesité y que gracias a ello me sentí siempre seguro.

A mi padre Fernando Villalobos, por confiar siempre en mí y hacerme apreciar los frutos de mi esfuerzo.

A mis hermanos Niza Yaa y Getsemani, que siempre me recibieron con una sonrisa y cuidaron de casa, mis padres y de mí.

A Iliana Rodríguez, por su apoyo incondicional, sus consejos y su atención. Tus palabras de aliento me dieron valentía y esas mismas palabras me dieron seguridad cuando más temeroso me sentía, gracias.

A mis tíos Pablo Villalobos y Marlit de Paz que me apoyaron a inicios de mi carrera profesional. Aquel pequeño gesto fue el inicio de este logro.

A todos mis compañeros y amigos que estuvieron conmigo a lo largo de este viaje, compartieron momentos inolvidables y se preocuparon por mi bienestar.

A la Universidad Tecnológica de la Mixteca por las instalaciones, docencia y personal que siempre procuraron un excelente ambiente de investigación.

Finalmente, al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado (Becario No. CVU: 1080813).

A todos ustedes: muchas gracias.
Exon.

Resumen

En este proyecto de investigación se desarrolla una técnica de control cooperativo basada en consenso líder seguidor para un Sistema de Múltiples Agentes (SMA) diseñado en ROS-Gazebo. Los agentes se conforman de robots móviles diferenciales. El controlador se diseña a partir del modelo cinemático con punto de operación c, al cual se le aplica la técnica de linealización por realimentación (planitud diferencial) resultando en un sistema de doble integrador. También, a forma de comparación, se diseña un controlador cooperativo a partir del modelo cinemático con punto de operación h (control clásico h) pues este modelo resulta en un sistema de simples integradores. Por un lado, los resultados a nivel simulación en Matlab-Simulink muestran que el SMA alcanza el consenso líder-seguidor de formación y seguimiento. En este trabajo de tesis no se considera el modelo dinámico. Por otro lado, las simulaciones en ROS-Gazebo consideran las dinámicas de los agentes así como factores físicos como fricción y gravedad. Los resultados sugieren que ambos controladores son capaces de llevar al SMA a un estado de consenso y seguimiento utilizando sólo sus modelos cinemáticos. Tanto en Matlab-Simulink y ROS-Gazebo, se observa una superioridad del control por planitud diferencial en la tarea de seguimiento y una inferioridad en tareas de formación con respecto al control clásico, lo anterior se sustenta mediante los indicadores de desempeño ISE e IAE.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La robótica móvil como rama de la robótica, nace de la necesidad de ampliar el campo de aplicación que anteriormente se limitaba a robots fijos para realizar distintas tareas en la industria [1]. Actualmente, los Robots Móviles Terrestres (RMT) representan un tema básico para adentrarse en el estudio de la robótica móvil[2, 3]. Dentro de esta clase de robots se pueden encontrar distintos métodos de locomoción para su navegación con base en entorno y características del medio: oruga (serpiente) [4], ruedas [5, 6] o patas [7, 8]. Sin embargo, es notable el dominio de la locomoción por ruedas en el desarrollo de tecnología para desplazarse: bicicletas, automóviles y robots móviles. Los RMT se caracterizan por ser holonómicos, es decir, si son capaces de moverse en todas direcciones en el plano al instante (sin restricciones), o bien, no holonómicos, cuyo movilidad es limitada (con restricciones).

Respecto de los robots móviles con ruedas, es posible encontrar configuraciones cinemáticas como Ackerman, triciclo, omnidireccional y diferencial [9]. El Robot Móvil Diferencial (RMD) pertenece al tipo no holonómico, posee solamente dos ruedas con tracción (un motor para cada rueda) y una o dos ruedas de apoyo que pueden ser de tipo castor (o bola). El RMD es utlizado frecuentemente para fines académicos y de investigación debido a que es posible controlarlo utilizando su modelo cinemático, es estable y necesita de menos partes para su construcción. Su tracción diferencial le permite girar alrededor de su propio eje con un menor número de maniobras, en comparación con otras configuraciones que requieren de más componentes. Por lo tanto, necesita de una menor cantidad de energía para desplazarse en el plano [10, 11].

Algunas de las técnicas de control utilizadas en este robot móvil son redes neuronales [12] y modos deslizantes [13] para alcanzar una posición en el plano utilizando las velocidades lineales de cada llanta. En [14] se controla el seguimiento de trayectoria del RMD con las posiciones de cada llanta mediante una combinación de retroalimentación de posición y modos deslizantes. En [15] se obtiene el modelo cinemático de un RMD y se propone un control basado en linealización de entrada-salida. Ortigoza, et al. [16] proponen un controlador jerárquico

(a)

(b)

Figura 1.1: En la naturaleza se pueden encontrar especies que trabajan de forma cooperativa para cumplir una función: a), parvada de patos en migración manteniendo una formación [17], b) abejas de miel construyendo capas en su panal [18].

que consta de tres niveles, donde el nivel medio es la etapa de actuación que es controlado por planitud diferencial.

La atención dada al control de posición y seguimiento de trayectorias de los RMD en el plano, da paso a nuevas disciplinas que permiten utilizar y aprovechar las ventajas antes mencionadas para realizar una tarea cooperativa. Al igual que las hormigas encomiendan subfunciones para el funcionamiento completo del hormiguero, el trabajo en equipo de los RMD puede traer consigo funcionalidades adicionales sin la necesidad de incorporar nuevas propiedades, solamente trabajando de forma coordinada.

1.1. Estado del arte

En las últimas dos décadas se ha visto un incremento en el interés por sistemas coordinados, cooperativos e inteligentes debido a su potencial en la industria [19, 20, 21, 22], comúnmente llamados Sistema de Múltiples Agentes (SMA). Los SMA buscan realizar una tarea en específico aprovechando su flexibilidad y reconfigurabilidad, inspirados en los ejemplos de la naturaleza observable, por ejemplo, el efecto cardumen de los peces, un enjambre de abejas o parvadas de aves como se muestra en la Figura 1.1. El control cooperativo de múltiples robots móviles es un tema de interés actual para las comunidades de control, debido a su aplicación en monitoreo, adquisición de datos y exploración de recursos [23, 24]. En el control cooperativo cada robot móvil es considerado un agente y éstos son partícipes de distintas aplicaciones de SMA: formación artística [25], evasión de depredadores [26], rebaño o enjambre (*flocking*/*swarming*) [27], seguimiento y órbita de objetos [28, 29]. En el trabajo hecho por Gao et al. [30] se presenta la aplicación de patrullaje donde se controla

una formación rotatoria y el seguimiento de un objeto (la referencia). Cada agente conserva la formación y órbita al rededor de éste.

Como anteriormente se mencionó, una de las aplicaciones de SMA es el control de formación, por ejemplo, de una figura geométrica. El control de formación de un SMA se divide en tres esquemas o enfoques: enfoque líder seguidor[31], enfoque de comportamiento[32] y enfoque basado en estructuras virtuales[33]. Debido a su simplicidad y escalabilidad el enfoque líder seguidor es utilizado frecuentemente en diversas aplicaciones ingenieriles [34, 35]. En este enfoque el líder representa la referencia para los demás agentes (seguidores) sobre la cual deben posicionarse [36]. La función del agente líder es guiar al grupo de seguidores por el plano, mientras los seguidores mantienen la formación [37]. La desventaja de un SMA con enfoque líder seguidor se presenta en el caso en el cual el líder pierda conexión con los seguidores o súbitamente es desconectado de la red, dejando a la deriva el sistema completo. Además, presenta una rechazo de perturbaciones pobre [38]. Es posible diseñar al agente líder de forma virtual añadiendo robustez al sistema y reduciendo el error general debido a factores físicos [39]. Sin embargo, esta solución es ajena a los fines de esta investigación, ya que se dependería de un computador que procese el comportamiento del robot móvil líder y simplifica el problema simulando de forma ideal el comportamiento del agente líder.

En el enfoque de comportamiento la ley de control de cada agente es un promedio ponderado del control para cada comportamiento, sin embargo, no es posible predecir o definir el comportamiento global del SMA. Los comportamientos posibles de cada agente pueden ser: evasión de obstáculos, búsqueda o formación. Su principal ventaja es que es un sistema donde cada agente calcula su ley de control (su comportamiento). En el enfoque de estructuras virtuales la formación de los agentes se interpreta como un objeto rígido manteniendo una relación geométrica también rígida durante el movimiento. Por lo tanto, no puede realizar cambios en las formaciones en situaciones como evasión de obstáculos [40]. Por una parte, el enfoque de comportamiento en comparación con en el líder seguidor, es complicado de analizar y comprobar su estabilidad. Por otra, en comparación con el enfoque de estructuras virtuales, el enfoque líder seguidor puede realizar formaciones variables en tiempo de ejecución [41]. Por tales motivos, en esta investigación se selecciona el enfoque líder seguidor para realizar la formación de figuras geométricas de un SMA.

Para que un SMA pueda realizar alguna aplicación como las antes mencionadas es necesario implementarse un sistema de control cooperativo. El control cooperativo para la formación necesita de un intercambio de información para establecer en cada agente un estado específico, es decir, regirse bajo un consenso. Un algoritmo de consenso específica la forma en cómo debe fluir la información entre los agentes y sus vecinos, los cuales a su vez están dentro de la red y relacionados bajo una cierta topología de comunicación [42]: árbol, estrella, anillo, mixto, etc. Esta topología de comunicación está definida por un grafo que representa las conexiones entre agentes. El problema del consenso se puede describir en cómo proponer un

algoritmo que relacione a cada uno de los agentes de tal forma que todos los agentes alcancen un acuerdo en términos de una variable de interés, es decir, que los estados internos de cada agente en el sistema lleguen a un valor común a través de negociación. En un SMA conformado por RMD existe un consenso local en el cual los agentes convergen en una posición definida con una orientación arbitraría [43] y un consenso global hace que cada agente se establezca en una posición y orientación definidas[44].

Lograr que un SMA se establezca en un estado específico de consenso requiere la implementación de un algoritmo de control. Es posible apreciar una gran variedad de algoritmos de control que permiten al SMA converger a un estado estable bajo un consenso predefinido. Entre los algoritmos de control aplicados a SMA se encuentran el Proporcional Integral y Derivativo (PID) adaptativo [45], *Deep-Learning* [46], *Q-Learning* [26], modos deslizantes[47] y *Backsteping* [48]. Ramírez et al. [49] muestran una técnica de control basado en planitud diferencial en un SMA para la formación de RMT tipo péndulo invertido de dos ruedas (diferencial). Los resultados obtenidos de esta investigación se sustentan con una simulación de 5 agentes. Venkatesh et al. [41] realiza la formación de robots no holonómicos tipo Ackerman por medio de planitud diferencial basado en realimentación de forma exacta. Sus resultados para la formación y el seguimiento de trayectoria se muestran a nivel simulación.

Dentro de los SMA se pueden diferenciar dos tipos de arquitecturas de control cooperativo: centralizado[50, 51] y descentralizado[52, 53]. En la arquitectura de control centralizado todos los agentes transmiten y reciben información a un nodo central como si de un bus de datos se tratase. Este nodo central se encarga de calcular cada controlador y dar ordenes de acción a cada agente del sistema. Por tal razón, el sistema centralizado no es escalable en cuanto al número de agentes. Al fallar dicho nodo central el sistema se detiene. En la arquitectura descentralizada se permite a cada agente tener una comunicación con cierto número de agentes (vecinos) y obtener información de ellos, cada agente cuenta con su propio control independiente. La arquitectura de control descentralizado es preferida por su simplicidad, capacidad de manejo de un grupo mayor de agentes, menor cantidad de cálculo computacional y menor uso de recursos de comunicación [54].

En los últimos cinco años es ampliamente utilizada la combinación de una arquitectura de control descentralizada y el enfoque de líder seguidor con agentes no holónomicos como el RMD en el diseño de un SMA para la formación. [44, 55, 56, 57]. Lo anterior debido a las ventajas que esta arquitectura y enfoque ofrecen a los SMA. Restrepo et al. [44] utilizan un concenso global para establecer una formación con condiciones iniciales diferentes de cero y con el agente líder fijo, es decir, sin movimiento. Se presentan resultados a nivel simulación y una implementación en el Sistema Operativo Robótico (ROS, por sus siglas en inglés). Dai et al. [55] estudian la formación y seguimiento de trayectoria bajo limitaciones de comunicación. Asumen que la información transmitida entre los agentes está limitada bajo un radio. En el trabajo previamente aludido, se consideran limitaciones de distancia y

ángulo de giro de cada agente seguidor respecto del agente líder. Una simulación con cuatro agentes muestra los resultados de la formación y seguimiento de trayectoria. En el trabajo presentado por Miao et al. [56] se diseña para cada uno de los agentes seguidores una ley de control usando un estimador de la posición del agente líder. La ley de control propuesta corrobora su funcionamiento para formación con una simulación de cuatro agentes. Además, se presentan resultados experimentales con cuatro agentes. Xiao Yu y Lu lui [57] presentan un la ley de control descentralizada para la formación bajo el enfoque líder seguidor con limitación de velocidad. Los resultados a nivel simulación con seis agentes muestran que la formación de los agentes es alcanzada.

1.2. Planteamiento del problema

La utilización de los RMD en los sistemas multiagentes otorgan ventajas sobre otros tipos de robots móviles debido a su facilidad de construcción, menor consumo energético, menor uso de piezas y estabilidad [10, 11]. Con ello es posible concentrar esfuerzos en la implementación de tareas o funcionalidades que aprovechen la disciplina de un SMA y las características que los RMD ofrecen para realizar tareas en forma conjunta.

Dentro de los enfoques de control de SMA, el líder seguidor para la formación de los agentes en figuras geométricas es uno de los temas de interés en la última década [58, 31, 34, 35, 36, 39, 44, 49, 55, 56, 57, 59]. En los SMA la arquitectura de control descentralizada permite a cada agente ejecutar su propia ley de control bajo un consenso, manejar una cantidad mayor de agentes y utilizar una menor cantidad de recursos de comunicación [54]. Los agentes solo pueden obtener información de sus vecinos para ejecutar sus leyes de control.

Debido a que el funcionamiento del sistema completo consiste en realizar una formación entre los agentes, el problema principal recae en el diseño de un algoritmo de consenso que lleve a los agentes a un estado deseado con respecto del agente líder. Este consenso debe ser capaz de alcanzarse aún cuando el agente líder siga una trayectoria suave, es decir, los agentes seguidores conservan la formación alrededor del líder. La ley de control que permitirá alcanzar el consenso local del sistema, puede basarse en cualquiera de las vistas anteriormente [46, 47, 48, 49]. El modelo cinemático del RMD debe ser considerado en el diseño de la ley de control. De esta manera es garantizado el buen desempeño del controlador contemplando la cinemática del RMD.

En combinación con la arquitectura descentralizada, el enfoque líder seguidor para la formación se ha utilizado en investigaciones recientes en las cuales se presentan resultados a nivel simulación [55, 56, 57, 58] y experimental [56]. En el trabajo presentado por Restrepo et al. [44] se muestran resultados experimentales utilizando ROS. Sin embargo, el agente líder permanece fijo en todo momento. Este entorno ofrece la posibilidad de trabajar con grafos

donde los nodos pueden recibir y transmitir información de sus estados.

Teniendo en cuenta el diseño en *software* de Dibujo Asistido por Computadora (CAD, por sus siglas en inglés) de cada uno de los RMD y lo visto anteriormente se propuso diseñar un consenso descentralizado para un grupo n (superior al promedio de 6 robots) de robots móviles no holonómicos bajo un esquema líder seguidor. Además, la implementación se realizará en ROS con un *software* de simulación especializado.

1.3. Justificación

En vista de las aplicaciones y la versatilidad que los SMA ofrecen en la realización de tareas de forma cooperativa, éstos han sido utilizados en la industria [20]. La propiedad emergente más atractiva de un SMA es la capacidad de dividir el esfuerzo para realizar una tarea. Cada pequeña función encomendada a cada uno de los agentes contribuye al cumplimiento coordinado de un objetivo. Por ejemplo, el transporte de una carga [39] en donde cada agente se posiciona en un punto de apoyo especifico, o el patrullaje de un objeto [30] donde cada agente se posiciona a una distancia y dirección del objeto. El diseñar un SMA con RMD para la formación de figuras geométricas es un precursor de aplicaciones más complejas donde se requiere que cada agente mantenga una posición y dirección.

Actualmente la implementación de SMA compuesto de RMD en *software* especializado basado en ROS, es una forma de experimentación viable y poco abordada para probar el enfoque líder seguidor descentralizado [44, 39]. La razón de ésto es que al igual que la interconexión de este SMA, ROS es interpretado por grafos. Staranowicz y Mariottini [60] en su presentación de *softwares* de simulación robótica, mencionan ésta caracteristica particular de ROS, además de mostrar su versatilidad al llevar una simulación 3D a una implementación real de un control para un RMD.

La simulación en este entorno es una alternativa de bajo riesgo y con resultados considerados como experimentos válidos, debido a que se toma en cuenta la dinámica y cinemática de los agentes robóticos participantes. Además, es posible contemplar comunicación, actuadores, sensores y propiedades del entorno. Con el diseño e implementación en *software* especializado de un SMA compuesto de RMD, la experimentación de enfoques, arquitecturas y técnicas de control no se ven comprometidas por variables exógenas (como la contingencia ocurrida a inicios del año 2019) que las retrasen.

1.4. Hipótesis

Utilizando una ley de control moderno es posible alcanzar un consenso local descentralizado para la formación de figuras geométricas y seguimiento de trayectorias de un SMA conformado por n robots móviles diferenciales bajo el esquema líder seguidor.

1.5. Objetivos

Objetivo general

Implementar el diseño de la formación y seguimiento de trayectorias de un SMA compuesto de n robots móviles diferenciales bajo un esquema líder seguidor en un entorno de simulación especializado basado en ROS.

Objetivos específicos

- Analizar las propiedades del modelo cinemático del robot móvil diferencial de dos ruedas.
- Seleccionar la topología de comunicación (grafo no dirigido) entre los agentes del SMA.
- Diseñar y validar un controlador para el seguimiento de trayectoria del robot móvil diferencial en el plano.
- Diseñar y validar consenso descentralizado para la formación de un SMA.
- Diseñar en *software* asistido por computadora los robots moviles diferenciales.
- Simular el consenso descentralizado en ROS.
- Implementar control de posición y seguimiento de trayectoria de un RMD en *software* de simulación 3D especializado.

1.6. Delimitaciones

- No se considera la evasión de obstáculos.
- No se consideran perturbaciones que cambien súbitamente la posición de uno de los agentes.

1.7. Metodología

En un SMA conformado por RMD el principal objetivo es lograr que cada uno de los agentes converja a un estado deseado por medio de la transmisión de información de variables de interés. Primero, para lograr este objetivo es necesario obtener el modelo cinemático del RMD de dos ruedas utilizando los vectores de velocidad de cada llanta en el plano. A partir del modelo cinemático, se diseña un control no lineal para el posicionamiento y el seguimiento de una trayectoria en el plano.

Segundo, se propone una topología de comunicación representada mediante teoría de grafos, dicha topología será diseñada con base en el enfoque líder seguidor. Del grafo de comunicación se obtiene la matriz de adyacencia, de grado y la matriz Laplaciana para simular el consenso descentralizado de formación y seguimiento de trayectoria conformado por n agentes. Dicho consenso es basado en el promedio de las condiciones iniciales de posición de cada agente.

Tercero, tanto el grafo, el controlador para cada agente y el consenso descentralizado serán programados en ROS para realizar simulaciones de comunicación entre nodos, sensado y actuación de cada agente. Una vez obtenida esta abstracción del SMA en ROS, se diseñará el RMD en un *software* CAD para después ser exportado a un entorno de simulación robótico especializado basado en ROS.

Por último, se realizarán las simulaciones necesarias de formación y seguimiento de trayectoria en el entorno de simulación robótico, para esto se configurarán diferentes figuras geométricas para que los agentes adopten una formación con base en ésta, mientras siguen una trayectoria suave en el plano. Se pretende que la dinámica de formación del SMA conformado por RMD sea visible para el usuario. La metodología que es utilizada se muestra en la Figura 1.2.

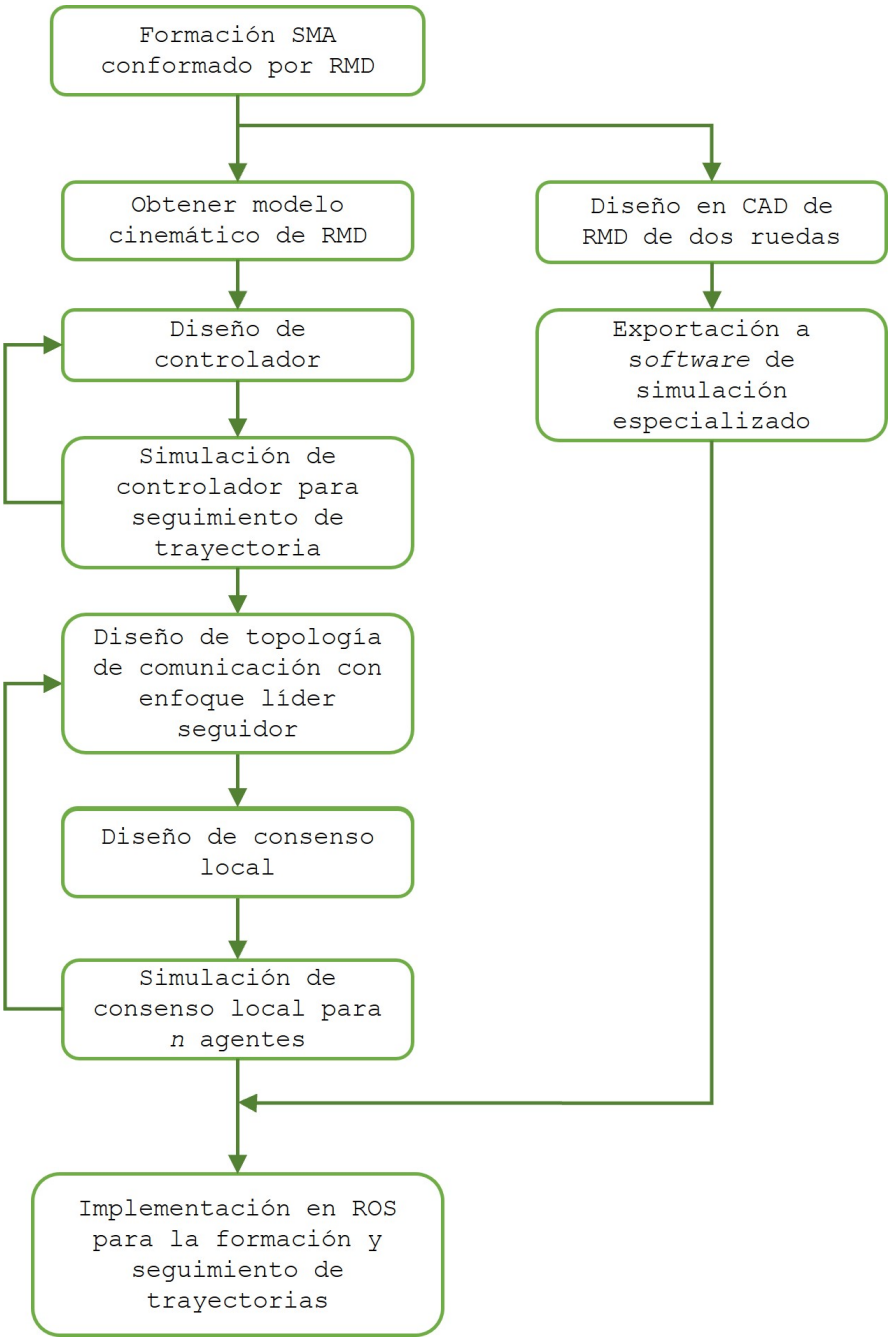


Figura 1.2: Metodología.

Capítulo 2

Marco teórico

En este capítulo se abordan temas específicos, mencionados en el capítulo anterior, para comprender el diseño del algoritmo de control en un SMA y la posterior implementación en *software* especializado. Se definen conceptos básicos de la teoría que representa de forma abstracta y simple un SMA con base en la comunicación entre agentes: un grafo. Después, se introducen los fundamentos de consenso, control no lineal y, por último, fundamentos teóricos del entorno de simulación especializado ROS-Gazebo.

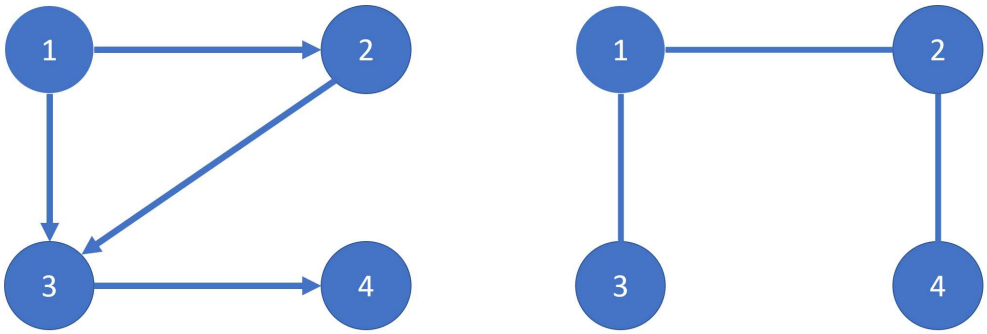
2.1. Teoría de grafos

La teoría de grafos es una disciplina antigua de las matemáticas que tuvo su primeros pasos bajo el problema de los siete puentes de *Königsberg*, resuelto por *Leonard Euler* en el siglo *XVIII* [61]. A partir de ese momento, bajo los criterios y visión de otros autores se han observado aplicaciones basadas en teoría de grafos, por ejemplo, en videojuegos [62] y búsqueda de caminos para conexión de redes [63].

Definición 1 *Un grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consta de $\mathcal{V}_\mathcal{G}$, un conjunto no vacío de vértices (nodos), y de $\mathcal{E}_\mathcal{G}$, un conjunto de pares no ordenados llamados aristas (enlaces). El conjunto $\mathcal{E}_\mathcal{G}$ contiene elementos distintos de $\mathcal{V}_\mathcal{G}$ [64].*

Por un lado, un grafo dirigido contiene aristas como un par ordenado $(v_i, v_j) \in \mathcal{E}_\mathcal{G}$ donde $v_i, v_j \in \mathcal{V}_\mathcal{G}$. Se pueden representar estas aristas como flechas que inician en v_i y terminan en v_j, es decir, una dirección establecida (ver Figura 2.1a). Por otro lado, si $\forall(v_i, v_j) \in \mathcal{E}_\mathcal{G}$ $\exists(v_j, v_i) \in \mathcal{E}_\mathcal{G}$ entonces se le conoce como grafo no dirigido, es decir, una unión sin dirección que es representado como aristas (enlaces) simples (ver Figura 2.1b)[65]. En un grafo no dirigido las aristas sin dirección significan que ambos agentes pueden enviar y recibir información.

Entiéndase a un grafo como una estructura que consta de vértices y conexiones entre ellos: aristas. Un grafo simple consta de vértices unidos por aristas no dirigidas. Cada arista



(a) (b)

Figura 2.1: Grafos de cuatro nodos: a) dirigido y b) no dirigido.

conecta dos vértices y no existen dos para un mismo par de vértices. En la Figura 2.1b se muestra un grafo simple con cuatro vértices y tres aristas. A lo largo de este escrito, se considera al grafo como la representación visual de un SMA, donde un nodo es un agente y una arista representa la comunicación entre dos agentes. Si la arista es una flecha (grafo dirigido), entonces ésta indica la dirección a la cual fluye la información; si es una arista simple (grafo no dirigido), entonces la información fluye en ambos sentidos. En adelante $e_{ij} = (v_i, v_j) \in \mathcal{E}_\mathcal{G}$ representa una arista que incide en los vértices v_i y v_j.

Cada relación que las aristas aportan a cada vértice puede ser descrito bajo las siguientes dos definiciones:

Definición 2 *Se dice que dos vértices v_i y v_j de un grafo no dirigido $\mathcal{G}$ son adyacentes si $e_{ij} \in E$.*

Definición 3 *El grado de un vértice (nodo) de un grafo no dirigido es el número de aristas que inciden en él.*

2.1.1. Representación algebraica de grafos

Utilizar un grafo como representación abstracta de un problema, es útil para observar la perspectiva general de éste. Sin embargo, es posible interpretar un grafo $\mathcal{G}$ en términos de la adyacencia y el grado de cada vértice, lo cual facilita los cálculos y la aplicación de algoritmos. La *matriz de adyacencia* de un grafo $\mathcal{G}$, denotado por $\mathcal{A}_\mathcal{G}$, es una matriz cuadrada de tamaño igual al número de vértices. Ésta almacena las relaciones entre vértices, representadas por $a_{ij} = 1$ si existe adyacencia, es decir, $e_{ij} \in \mathcal{E}_\mathcal{G}$; en caso contrario $a_{ij} = 0$.

Definición 4 *La matriz de adyacencia $\mathcal{A}_\mathcal{G} = [a_{ij}]$ o de conectividad, es definida como [66]:*

$$a_{ij} = \begin{cases} 1 & si \quad e_{ij} \in \mathcal{E}_\mathcal{G} \\ 0 & si \quad e_{ij} \notin \mathcal{E}_\mathcal{G} \end{cases} \tag{2.1}$$

Esta matriz describe específicamente al grafo $\mathcal{G}$, sin embargo, no es única. Dependiendo del orden que se le ha dado al conjunto $\mathcal{V}_\mathcal{G} = \{v_1, v_2, ..., v_n\}$, pueden existir $n!$ matrices de adyecencia posibles para el grafo $\mathcal{G}$ [64]. En el caso de un grafo no dirigido se cumple que $\mathcal{A}_\mathcal{G} = {\mathcal{A}_\mathcal{G}}^T$, es decir, es una matriz simétrica. La *matriz de grado* $\mathcal{D}_\mathcal{G}$ es una matriz diagonal, cuyos elementos representan el grado de cada nodo. El grado del nodo v_i es equivalente a la suma de los elementos de la fila *i-ésima* de la matriz $\mathcal{A}_\mathcal{G}$ [67].

Definición 5 *La matriz de grado (o matriz de valencia [68]) $\mathcal{D}_\mathcal{G} = [d_{ij}]$, es una matriz diagonal tal que,*

$$\mathcal{D}_\mathcal{G} = \begin{cases} d_{ij} = deg(v_i) & si \quad i = j \\ d_{ij} = 0 & si \quad i \neq j \end{cases} \tag{2.2}$$

donde $deg(v_i)$ es el grado del nodo v_i y se define como:

$$deg(v_i) = \sum_{j=1}^{n} a_{ij} \tag{2.3}$$

Estas dos matrices contienen la información de los pares de nodos que tienen comunicación y la cantidad de nodos vecinos (en comunicación) que posee un nodo. La *matriz Laplaciana* $\mathcal{L}_\mathcal{G}$ representa la interconexión de un grafo $\mathcal{G}$, combinando la matriz de adyacencia y de grado.

Definición 6 *La matriz Laplaciana $\mathcal{L}_\mathcal{G}$ es una matriz cuadrada tal que,*

$$\mathcal{L}_\mathcal{G} = \mathcal{D}_\mathcal{G} - \mathcal{A}_\mathcal{G} \tag{2.4}$$

Entre las propiedades de la matriz laplaciana de un grafo no dirigido están: 1) es simétrica, 2) es semidefinida positiva, 3) la suma de los elementos de cada fila es igual a cero, 4) sus valores propios tienen parte real no negativa, 5) el cero es un valor propio [67, 65] cuyo vector propio asociado es $[1_n]$, donde 1_n representa el vector de unos de tamaño n. Considerando el grafo simple mostrado en la Figura 2.1b y la ecuación (2.4), su matriz laplaciana es

$$\mathcal{L}_\mathcal{G} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$

2.2. Algoritmo de consenso

La comunicación entre agentes capaces de transmitir y recibir mensajes a sus vecinos, puede ser representada de forma abstracta mediante un grafo y de forma algebraica con la matriz Laplaciana asociada a éste (sección 2.1). La información que los agentes comparten y obtienen puede ser transmitida por cables, activación de sensores o inhalámbricamente. Esta propiedad de compartir información es una condición necesaria para la existencia del *control cooperativo*.

Algunas de las tecnologías utilizadas para desarrollar tareas del control cooperativo presentadas en el trabajo de R. M. Murray [69], son:

- Enfoques basados en la optimización.
- Campos potenciales.
- Estabilidad de cuerdas.
- Programa lineal entero mixto (MILP, por sus siglas en inglés).
- *Rendezvous*.
- Cobertura.
- Algoritmos de consenso.

Aunque R. M. Murray clasifica el consenso como una tecnología distinta de las que diseñadas para el control de formación, se han citado en el capitulo I investigaciones que emplean esta tecnología de consenso para el control de formación de vehículos autónomos tipo RMD. En esta investigación se selecciona esta tecnología ya que es un primer acercamiento al control cooperativo y puede diseñarse con teoría de grafos.

Definición 7 *Se dice que existe un consenso cuando los estados internos de los agentes de un SMA concuerdan al valor de una variable de interés [70].*

La cooperación de un SMA requiere de la capacidad de los agentes de acceder a la información de interés (crítica), con esto se espera que los agentes se coordinen para realizar una función. La información de la variable de interés, distribuida por la topología de comunicación, debe ser tratada con algoritmos que garanticen que el consenso sea alcanzado.

2.2.1. Consenso promedio

Denotemos $x \in R^n$ como el vector de estados o variable de interés del SMA compuesto de n agentes, $\mathcal{G}$ el grafo asociado a la topología de comunicación del SMA y $\mathcal{L}_\mathcal{G}$ su matriz Laplaciana. El estado x_i y sus derivadas están asociadas al vértice (agente) $v_i \in \mathcal{V}_\mathcal{G}$.

Definición 8 *Un consenso promedio tiene lugar cuando mediante un algoritmo de control, los estados (o variable de interés) de los agentes convergen al promedio de las condiciones iniciales [71], i.e, $c = x_i = x_j, \forall i, j$, tal que*

$$c = \frac{1}{n}\sum_{i=0}^{n} x_i(0) \tag{2.5}$$

Sistema simple integrador

Por un lado, el algoritmo para lograr el consenso promedio (ecuación (2.5)) de un SMA compuesto de n agentes descritos por un *sistema simple integrador* $\dot{x} = u$, está dado por

$$u = -\mathcal{L}_\mathcal{G} x \tag{2.6}$$

el algoritmo de control del *i-ésimo* agente se define como:

$$u_i = -\sum_{j=1}^{n} a_{ij}(x_i - x_j), \quad \text{i=1,...,n} \tag{2.7}$$

donde x_i representa la información del agente actual y x_j de los agentes vecinos. En lazo cerrado el SMA se describe con la siguiente ecuación:

$$\frac{d}{dt}(x) = -\mathcal{L}_\mathcal{G} x \tag{2.8}$$

Sistema doble integrador

Por otro lado, el algoritmo para lograr el consenso promedio de un SMA compuesto de n agentes descritos por un *sistema doble integrador* $\ddot{x} = u$, está dado por

$$u = -\begin{bmatrix}\mathcal{L}_\mathcal{G} & \mathcal{L}_\mathcal{G}\end{bmatrix}\begin{bmatrix}x \\ \dot{x}\end{bmatrix} \tag{2.9}$$

donde el algoritmo de control del *i-ésimo* agente se define como:

$$u_i = -\sum_{j=1}^{n} a_{ij}[(x_i - x_j) + (\dot{x}_i - \dot{x}_j)], \quad \text{i=1,...,n} \tag{2.10}$$

En lazo cerrado el SMA se describe con la siguiente ecuación:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0_n & I_n \\ -\mathcal{L}_\mathcal{G} & -\mathcal{L}_\mathcal{G} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{2.11}$$

donde $0_n \in R^{nxn}$ representa una matriz de ceros, y $I_n \in R^{nxn}$ es la matriz identidad de nxn.

2.2.2. Análisis de estabilidad

En esta sección probaremos la estabilidad del consenso promedio según el criterio de *Lyapunov* [72] para un sistema simple integrador. Sea $x = (x_1, x_2, ..., x_n)^T$ los estados asociados a cada vértice $\mathcal{V}_\mathcal{G} = \{v_1, v_2, ..., v_n\}$, proponemos $V(x)$ una función cuadrática de Lyapunov definida como:

$$V(x) = \frac{1}{2}x^T\mathcal{L}_\mathcal{G}x \tag{2.12}$$

$$V(x) = \frac{1}{2}\sum(x_i - x_j)^2 \tag{2.13}$$

donde la sumatoria es sobre los pares $i < j$ tal que $e_{ij} \in \mathcal{E}_\mathcal{G}$. Esta función indica que tan alejados se encuentran los agentes de sus vecinos. Dado de que $\mathcal{L}_\mathcal{G}$ es una matriz simétrica y semi-definida positiva, $V(x) \geq 0$. La derivada de (2.12) es

$$\dot{V}(x) = \frac{1}{2}(\dot{x}^T\mathcal{L}_\mathcal{G}x + x^T\mathcal{L}_\mathcal{G}\dot{x}) \tag{2.14}$$

$$\dot{V}(x) = x^T\mathcal{L}_\mathcal{G}\dot{x} \tag{2.15}$$

$$\dot{V}(x) = x^T\mathcal{L}_\mathcal{G}u \tag{2.16}$$

Entonces, escogiendo a u según (2.6) y sustituyendo en (2.16) obtenemos la ecuación (2.17), que resulta en $\dot{V}(x) \leq 0$.

$$\dot{V}(x) = -\,x^T(\mathcal{L}_\mathcal{G})^2x \leq 0 \tag{2.17}$$

Sea $x(0) \in R^n$ el vector de estados iniciales de cada agente, y sea $\mathcal{G}$ un grafo no dirigido (comunicación bidireccional), entonces se concluye que el sistema es asintóticamente estable [69, 73] en el punto de equilibrio dado por el consenso promedio (2.5).

2.3. Control no lineal

En esta sección se exponen conceptos básicos de control para diseñar un controlador no lineal de un robot móvil. Por un lado, un modelo no lineal puede ser linealizado alrededor de un punto de equilibrio de forma aproximada [74, 75], para posteriormente diseñar un controlador al rededor de esta aproximación del comportamiento real del sistema. Por otro lado, es posible diseñar un controlador no lineal basado en el modelo del sistema y que proporcione una respuesta de realimentación.

A continuación, se introduce el concepto de modelo no lineal y la técnica de planitud diferencial para el diseño de un algoritmo de control no lineal.

2.3.1. Sistema no lineal

En esta investigación se considera a un *modelo matemático* como un sistema de ecuaciones diferenciales que describen el comportamiento dinámico (modelo dinámico) o cinemático (modelo cinemático) de un sistema físico. El modelado matemático de los sistemas físicos se clasifica en este escrito en dos tipos: lineal y no lineal. A su vez estos se subdividen en sistemas de Una Entra-Una Salida (SISO, por sus siglas en inglés) y Múltiples Entradas-Múltiples Salidas (MIMO, por sus siglas en inglés). Una tercer clasificación puede considerarse: variante e invariantes en el tiempo. Esta investigación se centra en un sistema no lineal, MIMO e invariante en el tiempo. En adelante se denomina como *planta* a un sistema físico con un modelo matemático asociado, independientemente de su clasificación.

En general, un modelo matemático no lineal se representa con el siguiente sistema de ecuaciones

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{2.18}$$

donde x es el estado de la planta, u es la entrada de control, $f(x), g(x)$ pueden ser consideradas como funciones de estado y $h(x)$ funciones de las salidas y [75]. En el siguiente capítulo se verá que el *i-ésimo* agente de SMA tiene dos modelos cinemáticos de la forma 2.18.

2.3.2. Planitud diferencial

En esta sección se aborda la propiedad de *planitud diferencial* de sistemas no lineales, así como la obtención de las salidas planas para sistemas no lineales SISO y MIMO. Cabe adelantar que para este último tipo de sistemas no hay un método sistemático para su determinación. Por último, se expone la relación que existe entre la planitud diferencial y la linealización en forma exacta.

Fundamentos

La *planitud diferencial* de un modelo matemático es una propiedad que permite parametrizar las ecuaciones diferenciales de la salida, entrada y estados, de tal modo que son representadas en términos de un conjunto finito de una así llamada *salida plana* y sus derivadas temporales. El término de planitud diferencial, fue introducido en la década de los $90's$ con las investigaciones de M. Fliess et al. [76, 77, 78]. Se han reportado aplicaciones recientes en transporte [74] y robótica móvil [49, 79].

Como propiedad estructural de un sistema no lineal, la planitud es una herramienta que permite aplicar distintos métodos de control por realimentación una vez parametrizado el sistema: *Backstepping*, linealización por realimentación o pasividad. Dado que el sistema queda en términos de las *salidas planas* entonces la planitud es apropiada para tareas de control en las que se requiera un seguimiento de trayectoria de éstas [78].

Sistemas no lineales SISO

Considerando un sistema SISO no lineal de la forma (2.18) donde $x \in R^n$ y f, g son suaves (continuas), se dice que es *diferencialmente plano* si una salida artificial (salida plana) F que parametriza al sistema de tal forma que el estado, entradas y salida original pueden ser escritas en términos de F y sus derivadas (ecuación 2.19).

$$x = A(F, \dot{F}, \ldots, F^{\gamma}), \quad u = B(F, \dot{F}, \ldots, F^{\gamma+1}) \tag{2.19}$$

donde γ es un entero.

Definición 9 *Un sistema de la forma* (2.18) *es diferencialmente plano si existe una función escalar diferencial del estado, denotado por F, dado por*

$$F = h(x, u, \dot{u}, \ldots, u^{\alpha}) \tag{2.20}$$

donde α es un entero finito, talque el sistema inverso de 2.18 (u como salida y F como entrada), no presenta ninguna dinámica.

En planitud diferencial se habla de una *salida plana* que parametriza a un sistema no lineal, mientras que linealización en forma exacta se menciona un cambio de variables de estado: *difeomorfismo* ($z = T(x)$). Estos términos parecen semejantes, sin embargo notese como la *salida plana* F, a diferencia de T, puede ser función diferencial de la entrada de control u.

Teorema 1 . *Se dice que si un sistema no lineal SISO de la forma 2.18 es linealizable de forma exacta por entrada-salida, entonces, es diferencialmente plano [80].*

Calcular la *salida plana* de un sistema se basa en encontrar una función $\lambda(x)$ tal que cumpla con las condiciones simplificadas para encontrar el difeomorfismo mostradas en [81]. La salida plana F de un sistema no lineal SISO (2.18) es equivalente a calcular $T_1(x) = h(x)$ que linealiza al sistema por *entrada-salida*, es decir, $F = h(x)$.

El vector de estado calculado de la linealización entrada-salida $z = (z_1, z_2, \ldots, z_n)$ está compuesto entonces de F y sus $n-1$ derivadas, por lo tanto, $z(F, \dot{F}, \ldots, F^{n-1})$. Y las variables del sistema completo están parametrizadas según

$$\begin{aligned} x &= T^{-1}(F, \dot{F}, \ldots, F^{n-1}) \\ u &= \alpha(F, \dot{F}, \ldots, F^{n-1}) + \beta(F, \dot{F}, \ldots, F^{n-1})F^n \\ y &= F \end{aligned} \tag{2.21}$$

donde $v = F^n$, dejando ver que el sistema es *plano*.

Sistemas no lineales MIMO

En el caso de los sistemas no lineales de múltiples variables no existe una metodología general para encontrar las salidas planas [80]. Por una parte, los sistemas no lineales MIMO linealizados con un difeomorfismo $T(x)$ son diferencialmente planos. Cumple con el hecho de existir tantas salidas planas F_i como entradas de control. Por otra parte, existen sistemas MIMO que no pueden ser linealizadas con una ley de control *estático* como $u = \alpha(x) + \beta(x)v$, en su lugar, necesitan una ley de control *dinámico* [82] de la forma (2.22) y un difeomorfismo extendido $z = T(x, \xi)$.

$$\begin{aligned} \dot{\xi} &= \gamma(x, \xi) + \delta(x, \xi)v \\ u &= \alpha(x) + \beta(x)v \end{aligned} \tag{2.22}$$

donde ξ es una función del estado. M. Fliess y sus colaboradores han demostrado que para estos sistemas MIMO la linealización exacta por realimentación *dinámica* es equivalente a la planitud diferencial. Estos sistemas son llamados *equivalentes por realimentación endógena* [76, 41]. Considerando a un sistema MIMO que puede linealizarse por realimentación endógena, en virtud de 2.21 entonces ξ está en función de las salidas planas de orden extendido [76].

La realimentación endógena es esencialmente una extensión en el orden de una de las salidas planas, las señales de control u_i deben tener el mismo orden de derivadas de F_i $(i = 1, 2.., m)$, para que la relación entre entradas de control y salidas planas sea *invertible* [80]. Por ejemplo, si en u_j existe $F_i^{(k)}$ y en u_i exsite $F_i^{(k-1)}$, entonces se debe extender el orden de F_i a través de $\xi = \dot{u_i}$.

Definición 10 *Sea un sistema MIMO linealizable por realimentación dinámica, entonces es equivalente a un sistema MIMO plano por realimentación endógena [76].*

En este punto la habilidad, la práctica o empirismo es de utilidad para identificar estos casos y extender la dinámica del controlador. Considerando un sistema MIMO de la forma

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{2.23}$$

donde

$$g(x) = \begin{bmatrix} g_1 & g_2 & \dots & g_m \end{bmatrix}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \quad ; \quad h(x) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}$$

se presentan dos casos particulares que representan las condiciones necesarias y suficientes para la identificación de salidas planas de sistemas no lineales MIMO [80]:

- **Caso 1** ($x, u \in R^n$): este sistema no lineal es *plano* si y solo si $g(x)$ es invertible para x. Las salidas planas son los estados por sí mismos, es decir, $F_i = x_i$ $(i = 1, 2, ..., n)$.
- **Caso 2** ($x \in R^n$ **y** $u \in R^{n-1}$): un sistema de estas características es *plano* si y solo si es *controlable*. Las salidas planas dependen solo de los estados y de las salidas pero no de las derivadas de las entradas.

En este proyecto de investigación, se considera un sistema MIMO que pertenece a la subclase de sistemas planos que son linealizables por realimentación endógena y además pertenece al caso 1 para indentifiación de salidas planas.

2.4. Sistema Operativo Robótico

En esta sección se introducen los conceptos fundamentales relacionados al Sistema Operativo Robótico (ROS, por sus siglas en inglés). Después, se introduce la noción de un paquete y de nodos para la ejecución de programas. También, se abarcan los conceptos básicos de comunicación entre nodos: mensajes y servicios. Por último, se describen las funciones del entorno que permite obtener y transmitir información entre nodos: tópicos.

2.4.1. Fundamentos

ROS es un *framework* libre basado en el sistema operativo Linux, para crear software de robots enfocados a distintas aplicaciones. El proyecto de ROS fue iniciado por Morgan Quigley como parte del proyecto robot STAIR de la universidad de *Standford* en 2007. ROS es una colección de herramientas y librerías con el objetivo de simplificar las tareas de crear comportamiento de robots a través de una gran variedad de plataformas robóticas [83]. También, provee servicios como abstracción de *hardware* y control a bajo nivel.

Algunas de las características más relevantes de ROS son:

- Visualización: la ejecución de programas en ROS puede representarse en forma de *grafo* con nodos y comunicaciones punto a punto en una red de procesos. Esta comunicación es soportada bajo diferentes estilos: a través de *servicios*, *tópicos* o almacenamiento de datos por un *servicio de parámetros*. Además, posee simuladores *open-source* como Rviz y Gazebo para visualizar el comportamiento de robots.
- Multiplataforma: los nodos lanzados a ejecución pueden ser programados en diferentes lenguajes de programación como: *Pyton*, Java, C o C++. La comunicación entre los nodos es independiente del lenguaje de programación en que son descritos, es decir, un nodo programado en *python* puede comunicarse con un nodo programado en C, sin problema alguno.
- Modularidad: se pueden programar diferentes *nodos* para realizar tareas específicas y en caso de fallar alguno de ellos el sistema sigue funcionando. Posee paquetes listos para su utilización como *Moveit* que permite la planificación de trayectoria de manipuladores. La esencia de ROS es no volver a programar desde cero, sino reutilizar y aportar al código desarrollado por la comunidad creciente.

Es posible diferenciar dos forma de entender el entorno de ros [84]: a nivel del sistema de archivos y a nivel de un grafo.

2.4.2. Nivel de sistema de archivos

A continuación se presentan los conceptos que componen el sistema de archivos de ROS.

- Paquetes: son la unidad básica de la organización del *software* en ROS. Un paquete puede contener procesos de ejecución (*nodos*), librerías y archivos de configuración. Es el componente atómico de ROS.
- Meta-paquetes: son paquete especializados que solo sirven para representar o relacionar otros paquetes.

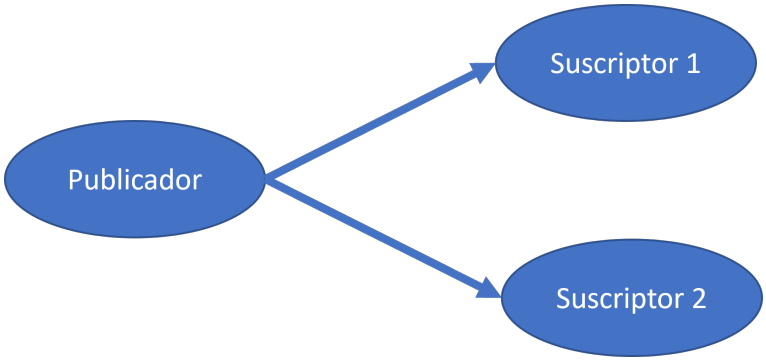


Figura 2.2: Grafo directo de comunicación entre dos nodos.

- Manifiestos de paquetes (*package.xml*): provee *metadatos* acerca de los paquetes, incluyendo su nombre, versión, descripción, información de licencia, dependencias, etc.
- Repositorios: la mayoría de los paquetes tienen mantenimiento usando un Sistema de Control de Versión (VCS, por sus siglas en inglés). Los repositorios son una colección de paquetes con el mismo VCS. Pueden contener incluso solo un paquete.
- Mensajes: son un tipo de dato utilizado para obtener o enviar información.
- Servicio: son otra forma en la que los nodos pueden comunicarse con otros nodos. Permiten enviar un saludos y recibir una respuesta.

Con estos conceptos es posible identificar la composición a nivel sistema de las unidades básicas que componen la comunicación y ejecución de nodos. En la siguiente sección se introduce el siguiente nivel de comprensión de ROS que nos permite visualizar de forma gráfica cada nodo en ejecución así como las comunicaciones entre ellos.

2.4.3. Nivel de grafos

En la literatura a este tema se le conoce como *nivel gráfico de cálculos*, sin embargo, en adelante será conocido como *nivel de grafos*. ROS puede soportar que diferentes programas o cálculos se ejecuten simultáneamente, éstos se comunican al mismo tiempo con otros con la transmisión y recepción de mensajes. Desde el punto de vista de los *grafos*, cada uno de los programas es un nodo (vértice) y cada comunicación (enlace) entre nodos es una arista. Por un lado, aquel nodo que transmite la información se conoce como *publicador*; por otro, aquel que recibe la información se le conoce como suscriptor (ver Figura 2.2).

Los conceptos fundamentales para entender el nivel de grafos son [85]:

- Nodo: es un programa ejecutable que realiza un calculo. Éstos pueden comunicarse con otros para realizar tareas complejas.
- Maestro: provee un nombre de registro y búsqueda para el nodo. Sin el maestro, los nodos no podrían encontrarse unos con otros, no podrían enviar y recibir mensajes, o invocar servicios.
- Servicio de parámetros: permite que los datos de ejecución sean guardados en una locación central. Es parte del *Maestro*.
- Mensaje: son utilizados para que los nodos se comuniquen con otros. Pueden ser constituidos por tipos primitivos (enteros, puntos flotantes, lógicos) o compuestos.
- Tópicos: son una forma sencilla de transmitir información de un nodo a otro. Éstos funcionan de forma independiente y son asíncronos.
- Servicios: proveen una forma de comunicación sincróna. Éstos actúan en forma de respuesta a una llamada de un nodo, con lo cual el otro debe dar una respuesta.
- *Bags*: son un formato para salvar y reproducir datos de mensajes de ROS. Son un mecanismo importante para salvar datos como los sensores.

Notese la fuerte relación que existe entre ROS y la teoría de grafos, esta relación denota la capacidad de ROS de interpretar a un SMA robótico y diseñarlo a nivel de grafos. Para más información acerca de estos conceptos y de su programación o modificación véase [84, 83].

Con los conceptos anteriores descritos de forma básica podemos introducir la siguiente sección sobre el simulador robótico 3D Gazebo basado en ROS que es la base de esta investigación para visualizar el comportamiento del SMA en un entorno especializado.

2.4.4. Gazebo

En esta sección introducimos el simulador seleccionado para hacer las pruebas del consenso para la formación de un SMA. Aunque las simulaciones matemáticas son una forma eficiente de obtener resultados a nivel simulación, usualmente no se pueden considerar factores físicos del propio robot y del ambiente: fricción, gravedad o colisiones. ROS nos ofrece un simulador en tres dimensiones (3-D) que puede tomar en consideración estos aspectos y además es interpretado por sus principios (paquetes, nodos, tópicos y servicios). Una opción justificable en el diseño de un SMA robótico para su formación ya que ademas de la navegación en el plano (2-D), es posible observar el comportamiento del SMA ante factores físicos.

En Gazebo es posible utilizar un motor de dinámica abierto el cual posibilita realizar una simulación con robots relativamente simples que producen un comportamiento físico

aceptable[83].

ROS se relaciona fuertemente con Gazebo por medio del paquete *gazebo_ros*. Este paquete proporciona un módulo de complemento que permite la comunicación bidireccional entre Gazebo y ROS. Los sensores simulados y los datos de física pueden transmitir desde Gazebo a ROS, y los comandos del actuador pueden transmitir desde ROS a Gazebo. Cada robot simulado en el entorno de Gazebo tendrá asociado un nodo en el nivel de grafos de ROS, en consecuencia este nodo del robot puede compartir información con otros nodos a través de sus tópicos. Además, otros nodos pueden modificar su estado mediante la publicación en los tópicos del robot.

Modelado de robots URDF

El Formato de Descripción Robótica Unificada (URDF, por sus siglas en inglés) es un archivo en formato XML usado en ROS para describir todos los elementos de un robot. Es un formato estandarizado de ROS que nos da la posibilidad de diseñar y programar las relaciones entre los componentes de un robot [84]: *link* y *joint*. Por un lado un *link* puede entenderse como un eslabón de un brazo manipulador o una llanta de un robot RMD, estos representan las partes básicas de un robot y contiene la información de color, tamaño y propiedades dinámicas. Por otro lado, un *joint* representa una unión o un par cinemático entre dos *links*: revoluta, prismático, fijo, etc.

A pesar de esta posibilidad de diseñar el modelo de un robot, el formato URDF está limitado, por las restricciones del entorno de diseño, a solo poder utilizar figuras geométricas simples. Sin embargo, existe la posibilidad de exportar un diseño en *software* de Diseño Asistido por Computadora (CAD, por sus siglas en inglés) al entorno de ROS Gazebo, de tal forma que se exporten diseños más complejos. En la Figura 2.3 se observa un RMD diseñado en CAD y exportado a formato URDF. Éste conserva las propiedades dinámicas como: matriz de inercia y masa. En caso de requerirse información adicional o ahondar en este tema puede consultarse [84].

El codigo de descripción URDF puede ser mejorado conviertiendo a formato *xacro*. Xacro es una versión mejorada del URDF que permite exportar otros codigos URDF, crear macros y reutilizarlos Además, en xacro se pueden utilizar herraminetas de programación para hacer operaciones matematicas simples, declarar varaiables y condionales. En el capítulo 5 se expondrá el proceso de diseño y exportación del RMD, además de su programación para una comuniación desde Gazebo y ROS.

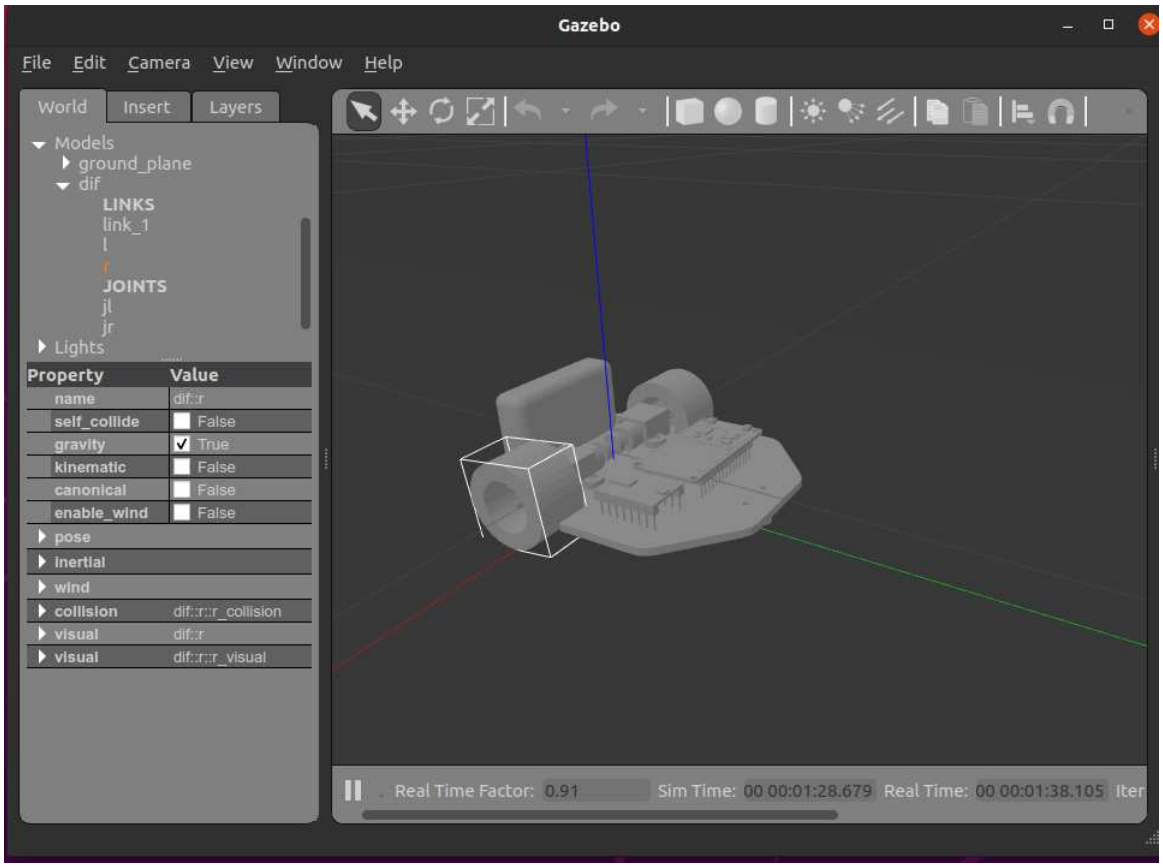


Figura 2.3: Diseño de RMD exportado a formato URDF.

Capítulo 3

Robot Móvil Diferencial

En este capítulo se muestran dos modelos cinemáticos asociados al RMD y los algoritmos de control clásico para regulación y seguimiento de trayectoria en el plano. Se desarrolla un control no lineal con base en el modelo cinemático utilizando la técnica de control visto en el capítulo anterior: planitud diferencial o linealización por realimentación dinámica. Por último, se realizan simulaciones matemáticas donde se muestra la acción de los sistemas de control.

3.1. Modelo cinemático

Cada modelo matemático asociado a un sistema tiene una referencia sobre la cual es medida de forma cuantitativa una variable de interés o el estado completo. El modelo cinemático asociado al RMD describe su posición y velocidad en el plano. Antes de conocer el estado (posiciones y velocidades) es necesario seleccionar un punto sobre el robot móvil sobre el cual se obtendrá su modelo cinemático. En general es posible diferenciar dos casos para el modelo cinemático del RMD: punto de operación *'c'* y *'h'*.

Considerando la imagen del RMD mostrado en la Figura 3.1, se observa por un lado, que el punto *'c'* está sobre la línea perpendicular a la velocidad de las llantas y sobre el centro de giro del robot móvil. Por otro lado, el punto *'h'* puede ser cualquier otro punto sobre el RMD diferente de *'c'*. En la práctica es común que no siempre se desee conocer el estado al rededor de un punto central, debido a espacio de componentes o disposición de los mismos. Por ejemplo, el robot KUKA *youbot* [86] cuyo brazo manipulador montado sobre un robot móvil omnidireccional no se encuentra en el centro de acción de las llantas o del propio robot.

Como consideraciones previas a la presentación de los modelos cinemáticos, tomaremos el hecho de que el RMD es completamente rígido, no cuenta con partes flexibles y que no existe deslizamiento de las ruedas con el plano 2D.

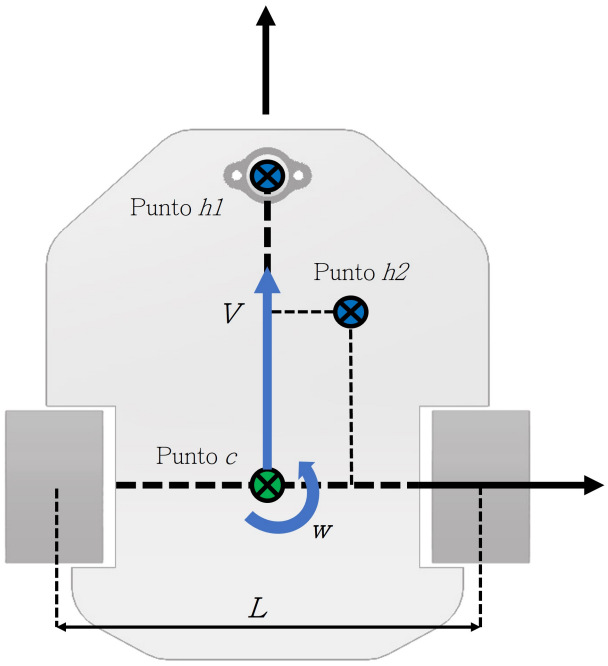


Figura 3.1: Robot móvil diferencial puntos h y c.

Por último, se presenta la nomenclatura que será utilizada en adelante en los modelos cinemáticos:

L Distancia entre el centro de masas de las llantas (mts).

h Distancia del punto c perpendicular al eje de giro de las llantas (mts).

V Velocidad lineal del RMD (mts/seg).

w Velocidad angular del RMD (rad/seg).

θ Ángulo de inclinación del robot (rad).

x_c Posición del punto c en X (mts).

y_c Posición del punto c en Y (mts).

3.1.1. Modelo en *c*

Sea el diagrama de cuerpo libre del RMD en el plano 2D de la Figura 3.2 y considerando el punto c (punto en color verde)como el punto del cual deseamos obtener su posición,

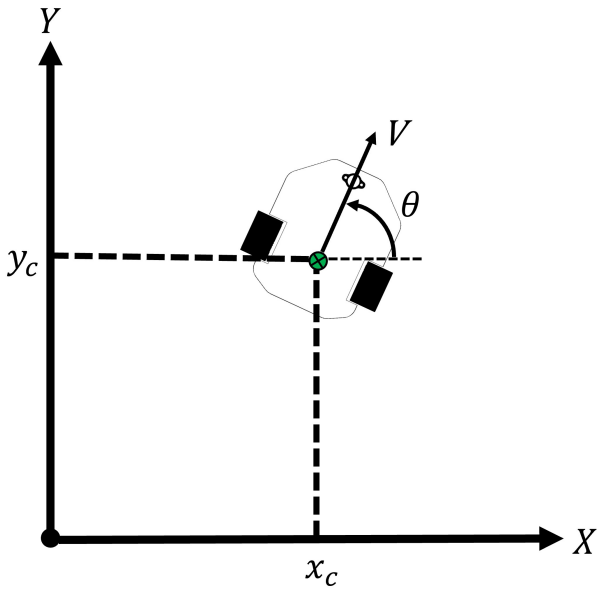


Figura 3.2: Diagrama de cuerpo libre del RMD punto c.

velocidad y orientación. Primero, observe que las llantas del RMD son paralelas entre sí. Segundo, dependiendo del sentido de giro y magnitud de cada una de las llantas se puede obtener una velocidad lineal V y una velocidad angular w que se refleja en el punto c. Tercero, se establecen V y w son las entradas al sistema y (x, y) las posiciones en el plano 2D. Cuarto, las ecuaciones que describen la posición y orientación del RMD en punto c, por simple inspección, para unas entradas de control (V, w) arbitrarias son:

$$\begin{aligned} \dot{x}_c &= V\ cos(\theta) \\ \dot{y}_c &= V\ sin(\theta) \\ \dot{\theta} &= w \end{aligned} \tag{3.1}$$

donde θ es medido con respecto al eje horizontal X. Resultando en un sistema MIMO no lineal de tres ecuaciones con dos entradas (velocidad lineal V y angular w) y dos salidas (posiciones x y y). Según [58, 55], despejando la entrada de control V de las primeras dos ecuaciones del sistema 3.1, e igualando tenemos

$$\dot{x}_c\ sin(\theta) - \dot{y}_c\ cos(\theta) = 0 \tag{3.2}$$

La ecuación 3.2 describe la *restricción noholonómica* del modelo en c del RMD. Significa que la velocidad lineal del robot es perpendicular al eje de giro del las llantas. En otras palabras, la velocidad lineal del robot en dirección al eje de giro de las llantas siempre es cero. Por último, se hace notar que este modelo contempla la capacidad de cambiar la orientación del RMD de forma instantánea.

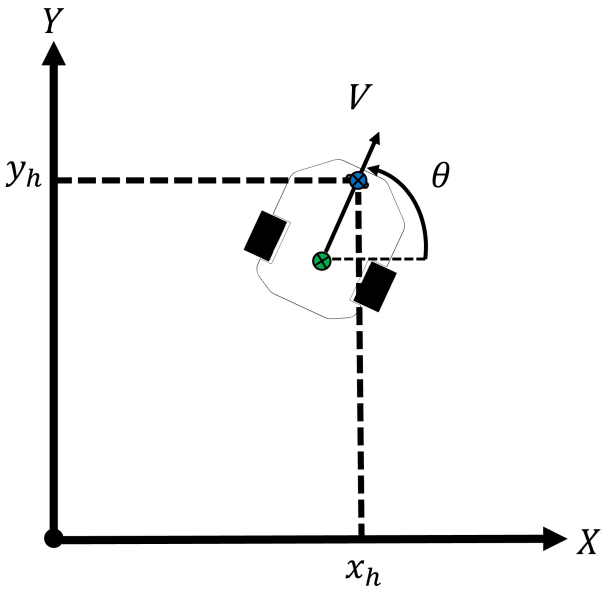


Figura 3.3: Diagrama de cuerpo libre del RMD punto h.

3.1.2. Modelo en h

Sea el diagrama de cuerpo libre mostrado en la Figura 3.3 y considerando que el punto h (punto en azul) es aquel del cual deseamos conocer su estado. Primero, observese que el punto c puede considerarse como un pivote de h, por lo tanto la posición de h se describe como:

$$\begin{aligned} x_h &= x_c + h\, cos(\theta) \\ y_h &= y_c + h\, sin(\theta) \end{aligned} \tag{3.3}$$

Segundo, derivando la posición para obtener las velocidades cartesianas

$$\begin{aligned} \dot{x}_h =& \dot{x}_c - h\, sin(\theta)\, w \\ \dot{y}_h =& \dot{y}_c + h\, cos(\theta)\, w \\ \dot{\theta} =& w \end{aligned} \tag{3.4}$$

Tercero, sustituyendo 3.1 en 3.4

$$\begin{aligned} \dot{x}_h =& V\, cos(\theta) - h\, sin(\theta)\, w \\ \dot{y}_h =& V\, sin(\theta) + h\, cos(\theta)\, w \\ \dot{\theta} =& w \end{aligned} \tag{3.5}$$

Resultando en el modelo cinemático del RMD en h. Por último, se expresa en su forma matricial (ecuación 3.6) donde se puede observar que la matriz jacobiana de velocidades (J) tiene un determinante distinto de cero. Esta particularidad permite realizar una cinemática inversa en el diseño de controladores, lo cual no es posible en el modelo en c.

$$\begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} = \underbrace{\begin{bmatrix} cos(\theta) & -h\ sin(\theta) \\ sin(\theta) & h\ cos(\theta) \end{bmatrix}}_{J} \begin{bmatrix} V \\ w \end{bmatrix} \tag{3.6}$$

3.2. Diseño de sistemas de control

En esta sección se presentan dos controladores para el modelo cinemático con punto de opeación $\bullet$ y un controlador para el modelo cinemático con punto de operación h. El segundo controlador para el modelo cinemático en c se basa en la linealización por realimentación dinámica o planitud diferencial bajo la misma técnica mostrada en [41]. Lo anterior se logra reescribiendo al sistema en términos de las así llamadas *salidas planas*.

3.2.1. Control cinemático en c

Partiendo de las ecuaciones 3.1, se puede sustituir las derivadas de más alto orden del sistema por variables de control auxiliar, es decir, sustituir el estado por variables auxiliares de control. De esta forma la dinámica del controlador puede ser despejada y dejada en términos del error. Primero, las variables auxiliares utilizadas son:

$$\begin{aligned} r_1 &= -\ k_1\ (x_c - x_{ref}) \\ r_2 &= -\ k_1\ (y_c - y_{ref}) \\ r_3 &= -\ k_2\ (\theta - \theta_{ref}) \end{aligned} \tag{3.7}$$

donde $k_1, k_2 \in R$ son ganancias proporcionales positivas y $(x_{ref}, y_{ref}, \theta_{ref})^T \in R^3$ es el estado de referencia al que se desea llegar. Segundo, se sustituye 3.7 en estado de las ecuaciones 3.1, resultando en

$$\begin{aligned} r_1 &= V\ cos(\theta) \\ r_2 &= V\ sin(\theta) \\ r_3 &= w \end{aligned} \tag{3.8}$$

Tercero, despejando las entradas de control obtenemos

$$\begin{aligned} V =& \sqrt{r_1^2 + r_2^2} \\ w =& r_3 \end{aligned} \tag{3.9}$$

En este momento se debe escoger una coordenada en el plano (x_{ref}, y_{ref}) como la posición deseada, con esto la entrada de control V queda cubierta; sin embargo, la señal de control w depende de θ_{ref} la cual se diseña como

$$\theta_{ref} = tan^{-1}\left(\frac{r_2}{r_1}\right) \tag{3.10}$$

La orientación con la que llegue el robot al punto deseado en el plano 2D dependerá de las ganancias asociadas a las variables de control auxiliar. El controlador 3.9 y las variables auxiliares 3.7 solucionan el problema de control de *regulación* en el plano 2D. Para realizar un *control de seguimiento* es necesario proporcionar las derivadas de las referencias deseadas en las variables auxiliares como se muestra a continuación:

$$\begin{aligned} r_1 =& \dot{x}_{ref} - k_1\,(x_c - x_{ref}) \\ r_2 =& \dot{y}_{ref} - k_1\,(y_c - y_{ref}) \\ r_3 =& \dot{\theta}_{ref} - k_2\,(\theta - \theta_{ref}) \end{aligned} \tag{3.11}$$

donde

$$\dot{\theta}_{ref} = \frac{\dot{r}_2\, r_1 - \dot{r}_1\, r_2}{r_1^2 + r_2^2} \tag{3.12}$$

3.2.2. Control cinemático en h

Del modelo cinemático en h presentado en 3.6, se observa que es de la forma $B = Ax$, por la tanto, puede resolverse la cinemática inversa para despejar las leyes de control V y w. Primero, se comprueba que la matriz jacobiana de velocidades J posee inversa obteniendo su determinante

$$det[J] = \begin{vmatrix} cos(\theta) & -h\,sin(\theta) \\ sin(\theta) & h\,cos(\theta) \end{vmatrix} = h \neq 0 \tag{3.13}$$

Segundo, se procede a obtener J^{-1} mediante la técnica de cofactores vista en [87], resultando en

$$J^{-1} = \frac{1}{h}\begin{bmatrix} h\,cos(\theta) & h\,sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \tag{3.14}$$

la cinemática inversa resulta en

$$\begin{bmatrix} V \\ w \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x}_h \\ \dot{y}_h \end{bmatrix} \tag{3.15}$$

Tercero, similarmente al controlador en c, se proponen las variables auxiliares para las derivadas de mayor orden

$$\begin{aligned} r_x &= -k_p \left(x_h - x_{ref}\right) \\ r_y &= -k_p \left(y_h - y_{ref}\right) \end{aligned} \tag{3.16}$$

donde $k_p \in R$ es una ganancia proporcional positiva. Cuarto, sustituyendo 3.16 y 3.14 en 3.15, obtenemos

$$\begin{aligned} V &= r_x \, cos(\theta) + r_y \, sin(\theta) \\ w &= -\frac{r_x}{h} \, sin(\theta) + \frac{r_x}{h} \, cos(\theta) \end{aligned} \tag{3.17}$$

En este controlador no es necesario establecer una variable auxiliar r_3 que se encargue de regular la orientación θ (como en el punto c), ya que en 3.17 la velocidad angular está en términos del error de posición del RMD. Naturalmente, las variables auxiliares 3.16 solo resuelven el problema de regulación de posición en el plano 2D. Para realizar un seguimiento de trayectoria en el plano 2D se proponen como

$$\begin{aligned} r_x &= \dot{x}_{ref} - k_p \left(x_h - x_{ref}\right) \\ r_y &= \dot{y}_{ref} - k_p \left(y_h - y_{ref}\right) \end{aligned} \tag{3.18}$$

3.2.3. Control por linealización en forma exacta

Esta técnica de control se basa en la parametrización de cada una de las variables del sistema (estados, entradas y salidas) en términos de las salidas planas x_c, y_c y sus derivadas. Esta parametrización es equivalente a la linealización en forma exacta por realimentación dinámica, ya que necesita de incrementar el orden de una de las variables de control. Para desarrollar este controlador nos basaremos en el modelo cinemático con el punto de operación en c

$$\begin{aligned} \dot{x}_c &= V \, cos(\theta) \\ \dot{y}_c &= V \, sin(\theta) \\ \dot{\theta} &= w \end{aligned}$$

Considerando las primeras dos ecuaciones de modelo cinemático anterior, podemos expresar el angulo d giro del RMD mediante:

$$\begin{aligned} \frac{\dot{y}_c}{\dot{x}_c} &= tan(\theta) \\ \theta &= tan^{-1}\left(\frac{\dot{y}_c}{\dot{x}_c}\right) \end{aligned} \tag{3.19}$$

derivando con respecto del tiempo

$$\dot{\theta} = \frac{\ddot{y}_c\,\dot{x}_c - \ddot{x}_c\,\dot{y}_c}{\dot{x}_c^2 + \dot{y}_c^2} \tag{3.20}$$

Sabiendo que $\dot{\theta} = w$, entonces la velocidad angular está escrita en términos de las velocidades y aceleraciones en el plano. Elevando al cuadrado las mismas dos ecuaciones usadas en el paso anterior y sumando obtenemos

$$\dot{x}_c^2 + \dot{y}_c^2 = V^2\left(cos^2(\theta) + sin^2(\theta)\right)$$

despejando la señal de control de velocidad lineal

$$V = \sqrt{\dot{x}_c^2 + \dot{y}_c^2} \tag{3.21}$$

Las leyes de control 3.20 y 3.21 no son capaces de linealizar al sistema 3.1 por realimentación (al sustituir las entradas de control no se obtiene un sistema canónico de integradores). Es decir, este sistema no es linealizable por *realimentación estática* según [88]. Por esta razón, se recurre a la técnica de *realimentación dinámica*, que es en esencia un aumento de orden en la dinámica de las entradas de control, lo que resulta en un sistema de mayor orden que puede ser linealizable [89]. Ahora bien, observe que la entrada de control V no posee las aceleraciones de las posiciones (x_c, y_c) como en w. Por lo tanto, se agrega un integrador a la dinámica del controlador 3.21, tal que

$$\begin{aligned} V &= \int v\,dt \\ v = \dot{V} &= \frac{\ddot{y}_c\,\dot{x}_c + \ddot{x}_c\,\dot{y}_c}{\sqrt{\dot{x}_c^2 + \dot{y}_c^2}} \end{aligned} \tag{3.22}$$

donde v es la nueva ley de control de orden superior, que está en términos de la velocidad y aceleración de las posiciones (x_c, y_c).

Ahora bien, por un lado, considerando el modelo cinemático con punto de operación en c, su derivada (orden extendido) resulta en

$$\begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \end{bmatrix} = \begin{bmatrix} cos(\theta) & -V\ sin(\theta) \\ sin(\theta) & V\ cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{3.23}$$

donde se aprecia una matriz invertible, un caso similar al visto en la ecuación 3.6. Despejando las entradas de control

$$\begin{bmatrix} v \\ w \end{bmatrix} = \underbrace{\frac{1}{V} \begin{bmatrix} V\ cos(\theta) & V\ sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}}_{J_{L1}} \begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \end{bmatrix} \tag{3.24}$$

Por otro lado, escribiendo las ecuaciones 3.20 y 3.22 en forma matricial

$$\begin{bmatrix} v \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\dot{x}_c}{\sqrt{\dot{x}_c^2+\dot{y}_c^2}} & \frac{\dot{y}_c}{\sqrt{\dot{x}_c^2+\dot{y}_c^2}} \\ \frac{-\dot{y}_c}{\dot{x}_c^2+\dot{y}_c^2} & \frac{\dot{x}_c}{\dot{x}_c^2+\dot{y}_c^2} \end{bmatrix}}_{J_{L2}} \begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \end{bmatrix} \tag{3.25}$$

Se declaran variables de control auxiliar que sustituirán a las derivadas de más alto orden $(\ddot{x}_c, \ddot{y}_c)$ de los controladores (3.24) y (3.25)

$$\begin{aligned} u_x &= \ddot{x}_{ref} - \gamma_1 \left(\dot{x_c} - \dot{x}_{ref}\right) - \gamma_0 \left(x_c - x_{ref}\right) \\ u_y &= \ddot{y}_{ref} - \gamma_1 \left(\dot{y_c} - \dot{y}_{ref}\right) - \gamma_0 \left(y_c - y_{ref}\right) \end{aligned} \tag{3.26}$$

donde $\gamma_0, \gamma_1 \in R$ son definidos con un polinomio *Hurwits* de segundo orden, tal que

$$\begin{aligned} \gamma_0 &= {w_n}^2 \\ \gamma_1 &= 2\,\zeta\,w_n \end{aligned} \tag{3.27}$$

con $w_n, \zeta \in R$ mayores que cero.

Suposición 1 *Si al sustituir 3.20 y 3.22 en el modelo de orden superior 3.23 resulta en un sistema en su forma canónica de integradores, entonces las leyes de control 3.20 y 3.22 linealizan al sistema 3.1 en forma exacta por realimentación dinámica.*

La suposición 1 es cierta, si y solo si

$$J_{L1} = J_{L2} \tag{3.28}$$

La ecuación (3.28) deriva en otras 4 ecuaciones (una por cada elemento de las matrices de dos por dos) y éstas a su vez encuentran solución con las igualdades del modelo en c. Esto

significa que la ecuación (3.25) es equivalente al modelo inverso (3.23) y linealiza en forma exacta al modelo cinemático en c, pues al sustituir (3.24) en (3.23) se obtiene

$$\begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{3.29}$$

Se demuestra entonces que la suposición 1 es cierta.

Cada una de las variables del modelo 3.1 puede ser descrito en términos de las posiciones (x_c, y_c) y sus primeras dos derivadas. Invocando el teorema 1 y según [76, 41] se concluye que el modelo cinemático con punto de operación en c es diferencialmente plano con las salidas planas x_c y y_c.

Por último, el modelo resultante (3.29) presenta la forma ideal de un agente en un SMA para la aplicación del consenso promedio (2.9). Este tema se abordará completamente en el capítulo 4.

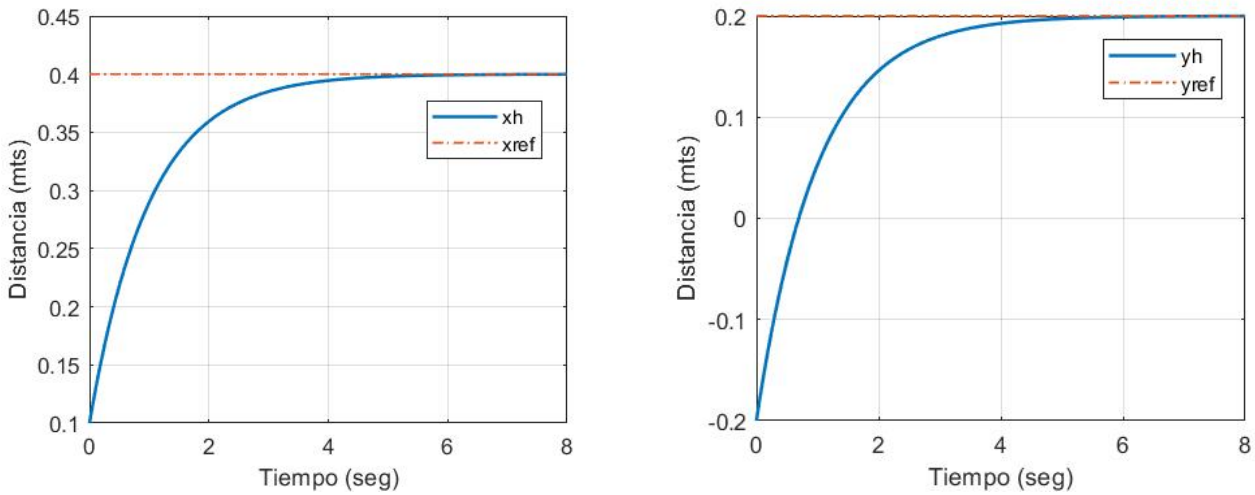
3.3. Simulaciones

En esta sección se presentan resultados a nivel simulación en el entorno de *Matlab Simulink*® para cada uno de los controladores desarrollados en la sección anterior. Los tres controladores del RMD son simulados con condiciones iniciales similares y los mismos objetivos de regulación y seguimiento de posición, para observar su comportamiento característico. Por último, se muestra la respuesta del controlador por linealización en forma exacta o planitud diferencial, ante los requerimientos de regulación y seguimiento de trayectorias para el modelo cinemático con punto de operación c.

3.3.1. Regulación de posición

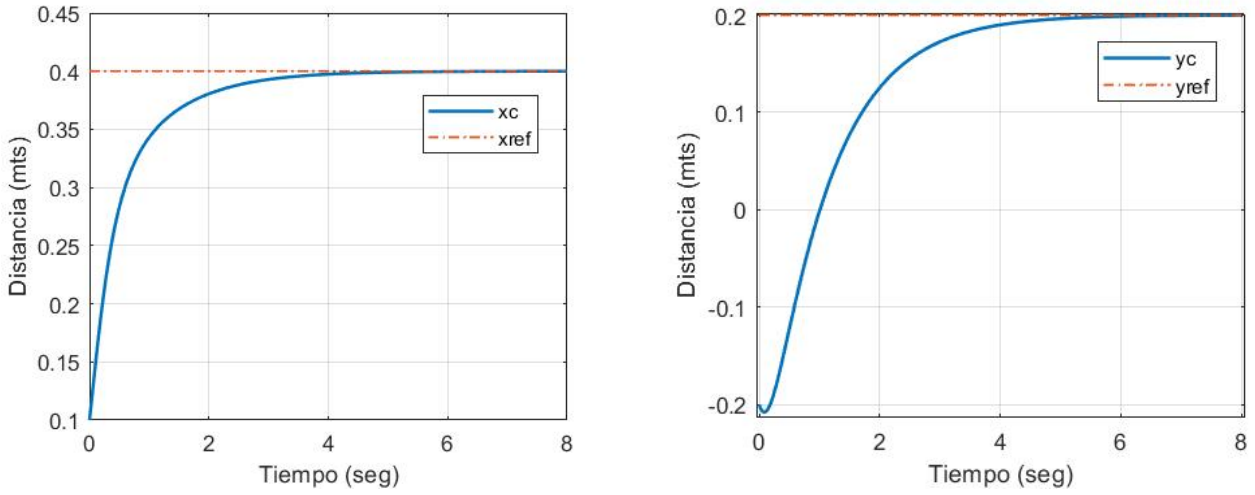
Es necesario poder diferenciar entre los dos controladores diseñados para el modelo en c, para esto la primer técnica mostrada en la sección 3.2.1 se conocerá en adelante como *control clásico c* y la segunda técnica se conocerá como planitud diferencial. Naturalmente, el controlador único para el modelo en h también se conocerá como *control clásico h*. Como últimas consideraciones previas a los resultados se enuncian los parámetros de simulación configurados en Matlab®: 1) paso fijo, 2) tolerancia de $1E{-}4$ y 3) *solver Runge Kutta*.

Los resultados para la regulación de posición en el plano utilizando en control clásico en h (3.17), con $h = 0{,}06\,m$ y $k_p = 1$, se presentan en las Figuras 3.4 y 3.6a. Las condiciones iniciales son: $x_h(0) = 0{,}1\,m$, $y_h(0) = -0{,}2\,m$ y $\theta(0) = -\pi/8$. Los puntos deseados son: $x_{ref} = 0{,}4\,m$ y $y_{ref} = 0{,}2\,m$. En la Figura 3.6a se representa el movimiento del RMD en el



(a) (b)

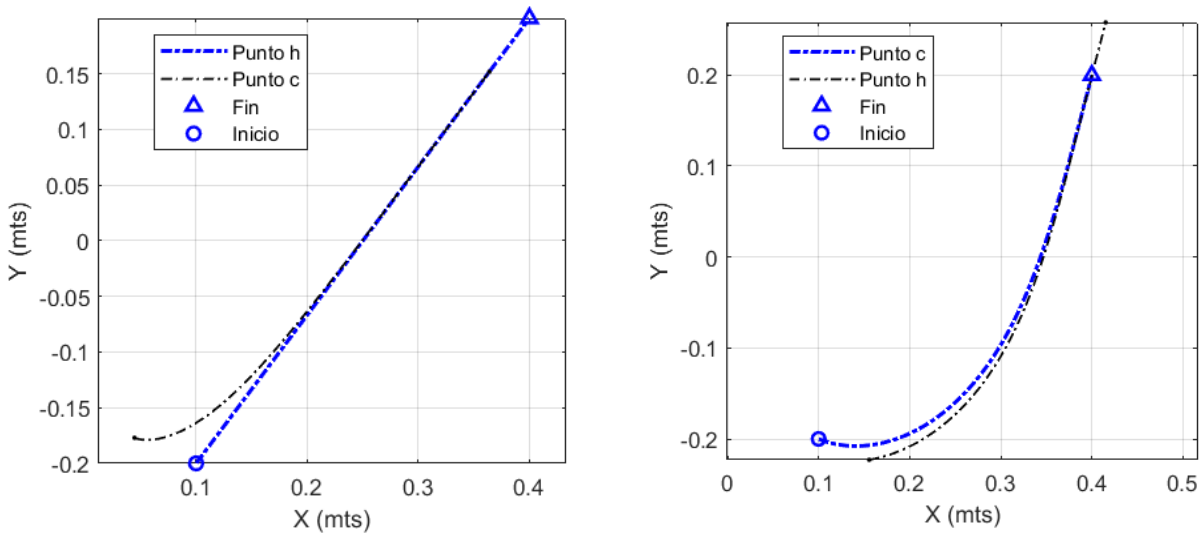
Figura 3.4: Respuesta de regulación del control clásico h: a) x_h y b) y_h.



(a) (b)

Figura 3.5: Respuesta de regulación del control clásico c: a) x_c y b)y_c.

plano. Con la finalidad de comprender el movimiento que el RMD realiza se muestran las trayectorias de los puntos c (línea negra punteada) y h (línea azul punteada), sin embargo, se debe tener en cuenta que el control se hace sobre el punto h y que el punto c solo se muestra con fines representativos de las maniobras que realiza el RMD. El punto h traza una recta que une las posiciones inicial (círculo azul) y final (triángulo azul), mientras que el punto c se comporta de tal forma que se garantice esta trayectoria.



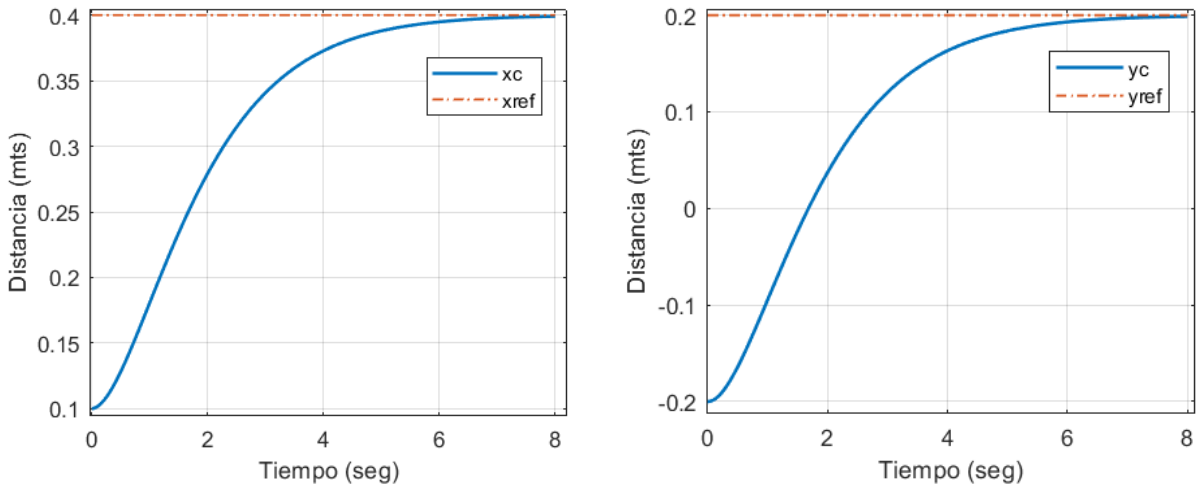
(a) (b)

Figura 3.6: Regulación de posición en el plano del RMD: a) control clásico en h y b) control clásico en c.

Un punto importante a recalcar del controlador clásico h es que aumentar la ganancia k_1 no cambia la forma de la respuesta, es decir, seguirá siendo una recta. Este fenómeno ocurre debido a que las ganancias del los controladores auxiliares (3.16) son iguales. Naturalmente, si las ganancias fueran distintas el RMD se comportaría diferente: con una trayectoria curva.

Los resultados al implementar del control clásico en c se presentan en las Figuras 3.5 y 3.6b. Las condiciones iniciales y las referencias deseadas son las mismas, ésto con la finalidad de comparar las respuestas entre las técnicas de control. Las ganancias de este controlador son: $k_1 = 1$ y $k_2 = 4$. En la Figura 3.6b se muestra la posición del RMD en plano, se muestra la trayectoria del punto c (línea azul punteada) y del punto h (línea negra punteada). Nuevamente, se aclara que la inclusión del punto h solo es representativo y que la acción de control es sobre el punto c.

La trayectoria del punto c se asemeja a una parábola. Debido a que cuenta con una ganancia específica para el control de orientación se permite ajustar su velocidad angular (w) independientemente de la velocidad lineal (V) inyectada. Al realizar diversas simulaciones se encuentra que la ganancia k_2 debe ser por lo menor dos veces mayor a k_1, esto es debido a que la velocidad angular del RMD debe ser mayor que su velocidad lineal o de lo contrario podría girar indefinidamente sin alcanzar su objetivo. Alternativamente, si se desea un comportamiento similar al visto en la Figura 3.6a se debe cumplir con $k_2 \gg k_1$.



(a) (b)

Figura 3.7: Respuesta de regulación por planitud diferencial: a) x_c y b) y_c.

Los resultados de simulación al implementar la ley de control por planitud diferencial se muestran en las Figuras 3.7 y 3.8. Las ganancias se sintonizaron con $w_n = 1$ y $\zeta = 1$, tal que $\gamma_0 = 1$ y $\gamma_1 = 2$, con lo cual se espera un respuesta amortiguada. Dado que la entrada de control v requiere de un integrador es necesario seleccionar una condición inicial de la velocidad lineal distinta de cero ($V(0) \neq 0$). Con base en las simulaciones realizadas y el controlador (3.24), por un lado, si la condición inicial es relativamente pequeña provoca una velocidad angular muy grande y la orientación del RMD crece rápidamente. Por otro lado, si la velocidad inicial es relativamente grande se tendrá un comportamiento similar al del controlador clásico c (ver Figura 3.6b), ya que la velocidad lineal inicial mueve al RMD hacia adelante antes de orientarse correctamente.

Por lo anterior, se escoge la velocidad inicial $V(0) = 0.01$. La respuesta de posición en el plano se muestra en la Figura 3.8. En esta Figura se observa como el RMD se orienta rápidamente en dirección del punto deseado y después comienza a moverse. Como resultado el punto c traza una línea recta. Con el controlador por planitud diferencial el RMD manifiesta un comportamiento contrario al control clásico h.

En la Figura 3.9 se muestran los ángulos de inclinación del RMD debidos a la acción de los tres controladores. En esta se muestra el valor ideal del ángulo θ (línea negra punteada), este valor resulta de la línea que une al valor inicial y final con respecto del eje horizontal. El control por planitud diferencial establece el ángulo del RMD al valor de ángulo ideal en un tiempo menor. El control clásico en c establece la orientación del robot en un valor distinto

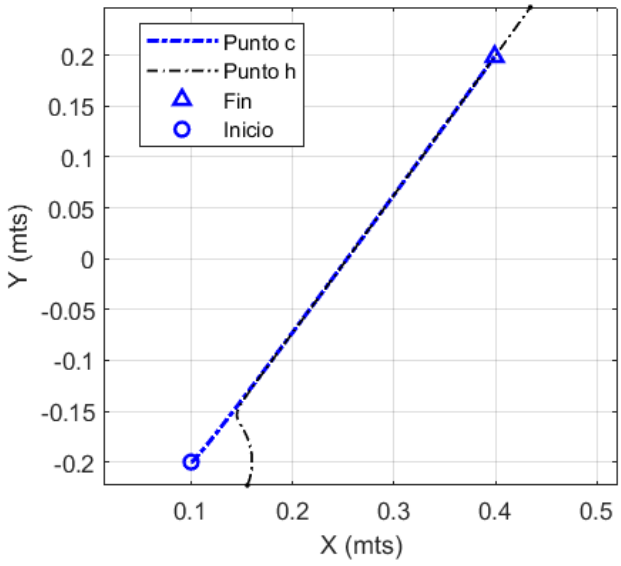


Figura 3.8: Regulación de posición en el plano del RMD por acción de planitud diferencial.

del ideal.

Una vez que se han mostrado los resultados de regulación de posición de cada uno de los controladores, podemos notar que: 1) el control por planitud diferencial es más lento: 8 segundos. Se atribuye este efecto a la extensión de orden del sistema (3.1) y a que las variables auxiliares contemplan las derivadas de velocidad del RMD, a diferencia de los controladores clásicos en h y c que solo contemplan posición conservando el orden de sus modelos cinemáticos. 2) El control clásico c necesita de ganancias mayores para trazar una trayectoria recta del punto c. 3) En el control clásico h el punto c realiza maniobras para que el punto h mantenga una trayectoria recta que une el punto inicial y final en el plano. 4) Por último, con el control por planitud diferencial, conservando las mismas ganancias, el punto c traza una recta que une el punto final e inicial, es decir, el robot aprovecha su capacidad de cambiar su orientación al instante con un menor número de maniobras, como se mencionó en el capítulo 1.

3.3.2. Seguimiento de trayectoria

Los controladores presentados anteriormente pueden diseñarse para el seguimiento de trayectoria, utilizando las variables auxiliares (3.16), (3.7) y (3.11). En los controladores clásicos c y h solo es necesario agregar la primer derivada de la trayectoria deseada (velocidad), mientras que para el controlador por planitud es necesario considerar hasta la segunda derivada (aceleración) de la posición deseada.

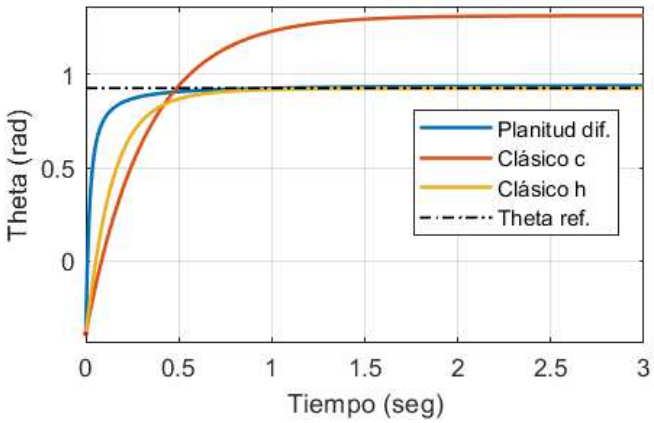


Figura 3.9: θ por acción de control de regulación.

La trayectoria deseada que se utiliza en las simulaciones es un polinomio *Bézier* de noveno orden. Lo anterior, es porque se busca un arranque y llegada suave del RMD en un intervalo de tiempo determinado. Esta trayectoria es asignada a la variable de posición deseada x_{ref}. La variable de referencia y_{ref} y sus derivadas serán obtenidas en función de x_{ref} de tal forma que el RMD traze siempre una recta entre la posición inicial y final. Considerando x_{ini}, y_{ini}, t_{ini} como los valores iniciales de cada variable y x_{fin}, y_{fin}, t_{fin} como los valores finales, el polinomio *Bézier* que describe a x_{ref} se define de la siguiente forma [90]:

Sea

$$\tau = \frac{t - t_{ini}}{t_{fin} - t_{ini}}$$

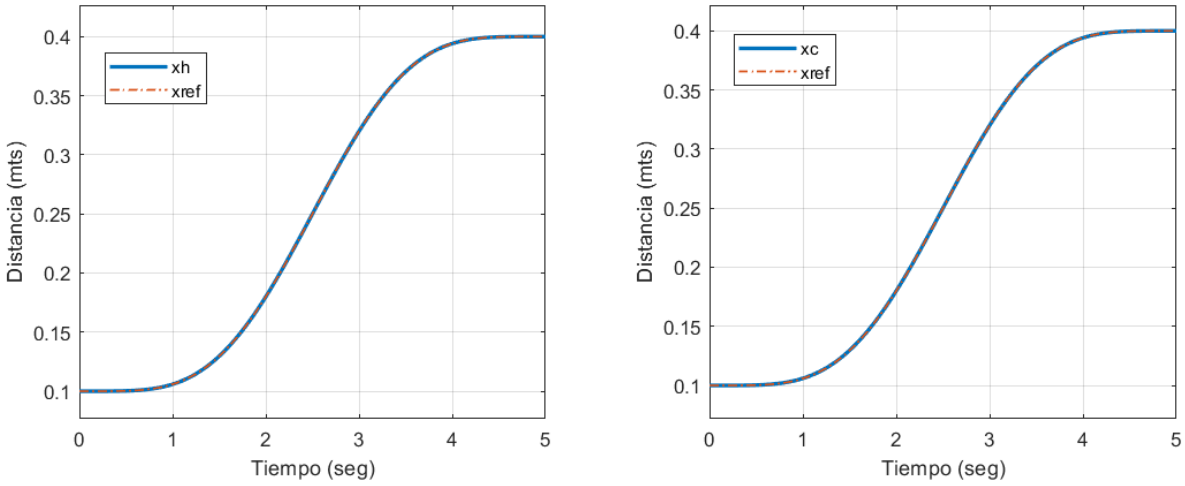
entonces

$$x_{ref} = \begin{cases} x_{ini} & \forall t \ < \ t_{ini} \\ x_{fin} & \forall t \ > \ t_{fin} \\ x_{ini} + p(\tau)(x_{fin} - x_{ini}) & \forall\, t_{ini} \leq t \leq t_{fin} \end{cases} \tag{3.30}$$

donde

$$p(\tau) = \tau^5(126 - 420\tau + 540\tau^2 - 315^3 + 70\tau^4)$$

Con lo cual la trayectoria de referencia y_{ref} se define como



(a) (b)

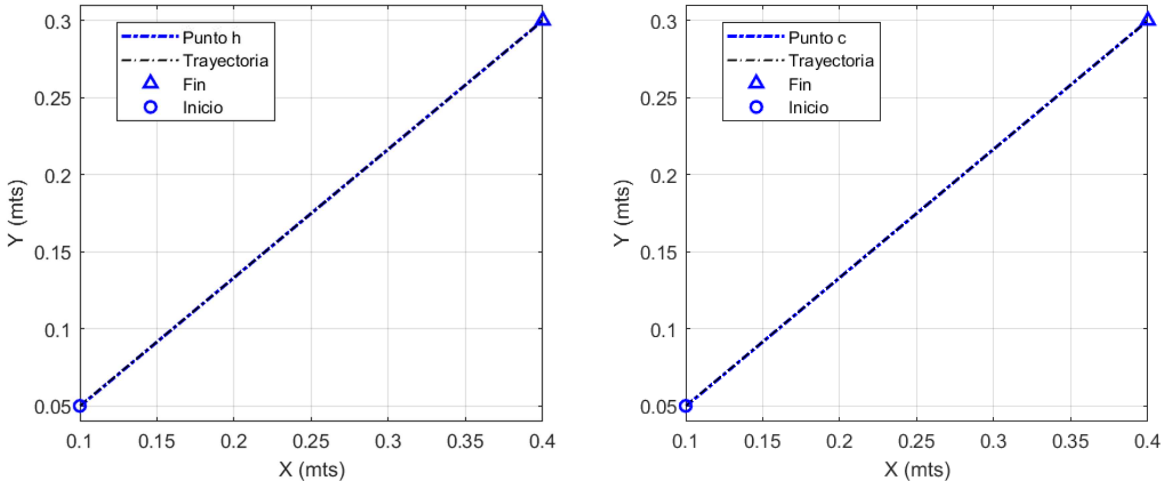
Figura 3.10: Respuesta al seguimiento de x_{ref} por control clásico: a) h y b) c.

$$y_{ref} = \left(\frac{y_{fin} - y_{ini}}{x_{fin} - x_{ini}} \right) (x_{ref} - x_{ini}) + y_{ini} \tag{3.31}$$

Se seleccionan los siguientes valores para el diseño de la trayectoria deseada: $x_{ini} = 0{,}1\,m$, $x_{fin} = 0{,}4\,m$, $y_{ini} = 0{,}05\,m$, $y_{fin} = 0{,}3\,m$, $t_{ini} = 0\,s$ y $t_{fin} = 5\,s$. Las condiciones iniciales de posición en el plano del RMD, para cada modelo, son exactamente x_{ini} y y_{ini}. La configuración de simulación en *Matlab-Simulink*® es la misma que en la sección anterior.

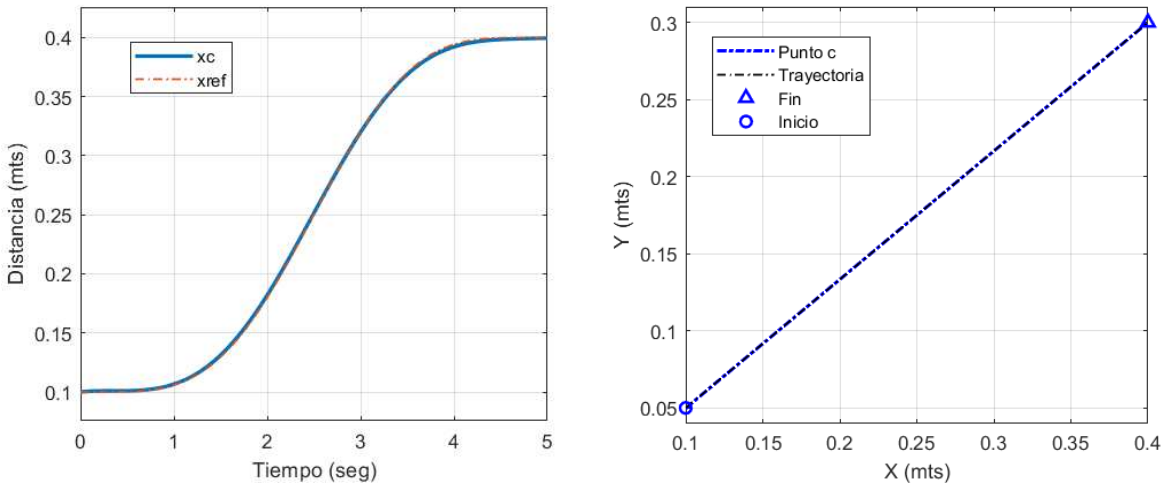
Por un lado, los resultados al seguimiento de trayectoria deseada x_{ref}, de los controladores clásicos h y c, se muestran en la Figura 3.10a y 3.10b respectivamente. Las ganancias de los controladores son: $k_p = 2$, $k_1 = 2$ y $k_2 = 4$. En las Figuras 3.11a y 3.11b, se muestran las respuestas de posición del RMD utilizando el controlador clásico en h y en c respectivamente. Las referencia deseada es seguida fielmente por ambos controladores tal que se logra el desplazamiento deseado (recta que une punto inicial y final) en un tiempo determinado.

Por otro lado, los resultados al seguimiento de trayectoria del controlador por planitud diferencial se muestra en la Figura 3.12. Las ganancias son sintonizadas con $z = 1$ y $w_n = 3$. Naturalmente, se establece la condición inicial de velocidad lineal $V(0) = 0{,}01$. En la Figura 3.12a se muestra el seguimiento de la referencia x_{ref}; en la Figura 3.12b, la posición del RMD en el plano. Este controlador también cumple con el objetivo del seguimiento de trayectoria deseada.



(a) (b)

Figura 3.11: Seguimiento de posición del RMD por control clásico: a)h y b) en c.



(a) (b)

Figura 3.12: Seguimiento de trayectoria por acción de planitud diferencial: a) x_c y b) posición del RMD.

En la Figura 3.13 se observa θ por acción de cada controlador de seguimiento. En este caso todos los controladores llevan al RMD a un valor estado de inclinación.

Presentados los resultados de seguimiento de trayectoria para los tres controladores, se puede notar que: 1) el control por planitud diferencial establece el RMD al valor de inclinación

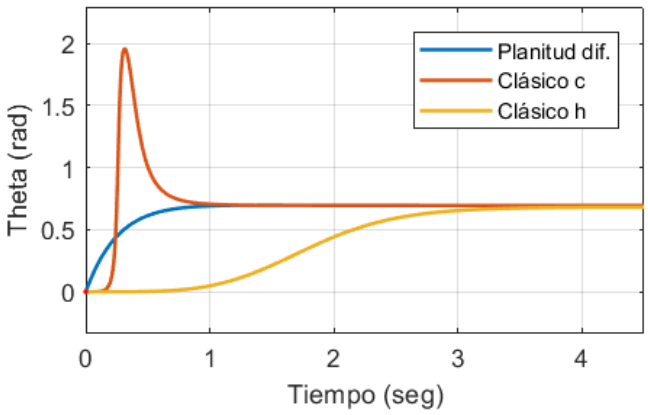


Figura 3.13: θ por acción de control de seguimiento.

ideal en un menor tiempo en comparación con el control clásico h; y 2) con una menor oscilación en comparación con el control clásico c. En cuanto a la respuesta en el plano no se identifica una diferencia notable en su comportamiento. Sin embargo, cabe mencionar que el controlador por planitud diferencial requiere de una $w_n = 3$, mayor a la de los controladores clásicos: $w_n = k_p = k_1 = 2$. Esto se atribuye a la extensión en el orden del controlador.

3.4. Diseño asistido por computadora

En esta sección se abordan los componentes electrónicos y mecánicos que componen al robot móvil, así como el Diseño Asistido por Computadora (CAD, por sus siglas en inglés) del RMD. El diseño preliminar fue desarrollado en el *software* Solidworks® ya que permite exportar los archivos a un formato URDF compatible con ROS-Gazebo. Por último, se muestran las vistas del ensamble y diseño preliminar del chasis.

Consideraciones previas

Primero, es de interés el hecho de poder incorporar todos los circuitos y componentes necesarios en una Placa de Circuito Impreso (PCB, por su siglas en inglés) o, en su defecto, una placa de prototipado rápido (preperforada) que sirva de base. La dimensión de esta placa es de $10x10\,cm$. Esto para mantener un diseño de tamaño reducido y utilizar dispositivos electrónicos pequeños ya que la aplicabilidad futura se orienta a dimensiones pequeñas: seguidor de línea o resuelve laberintos. Se espera obtener un diseño que no supere las siguientes medidas: $10x12x10\,cm$.

Segundo y último, con la finalidad de obtener un diseño de medidas realistas, las medidas

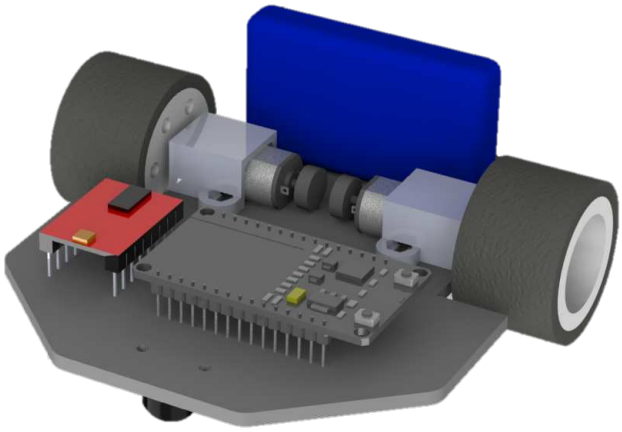


Figura 3.14: Vista isometrica de la distribución de los componentes electrónicos y mecánicos.

de los componentes electrónicos y mecánicos se basan en sus modelos comerciales.

3.4.1. Componentes electrónicos y mecánicos

Los componentes electrónicos y mecánicos que forman parte del diseño se listan a continuación seguidos de una breve justificación.

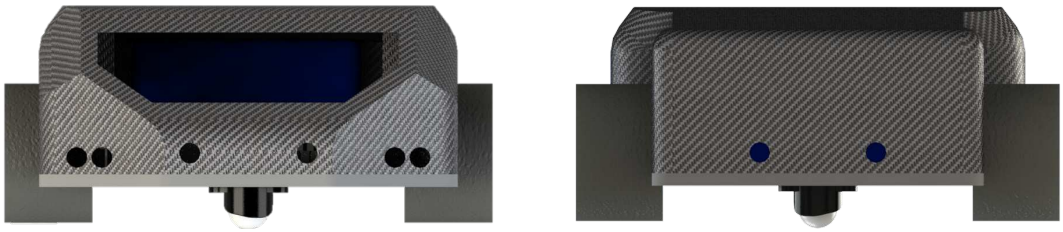
- Microcontrolador Esp32 WROOM 32: posee módulos de comunicación serial inhalambrica *Bluetooth* y *Wifi*. Permite enfocar el control de este robot por medio de una interfaz gráfica o establecer comunicación con otro dispositivo inalámbrico.
- Micro reductor con encoder de 105 rpm a 6 Volts: se espera conseguir una velocidad lineal superior a $25\,cm/s$. Estos motores poseen una resolución de 3 pulsos por revolución en el eje primario, con el factor de reducción adecuado, el eje de salida ofrece 870 pulsos por revolución. El RMD no necesita de un par alto ya que su aplicabilidad se restringe al desplazamiento en el plano.
- Batería LiPo de 7.4 Volts 500 mAh: suficiente para alimentar todo el sistema incluyendo los motores de 6 volts cuyo consumo máximo de corriente es $160\,mA$.
- Puente H TB6612: puente H dual con capacidad de 1 amperios por salida.
- Regulador de 5 volts LM7805: se utiliza para alimentar el microcontrolador y los demás componentes electrónicos, además de servir como protección.
- Leds, transistores, diodos y resistencias: para el diseño de un sensor de nivel de batería baja.

- *Switch* de dos estados: habilita la alimentación de la batería Lipo.
- Abrazaderas: su función es sujetar los motores a la placa PCB o preperforada.
- Rueda de castor: punto de apoyo para el RMD.
- Llantas de un diámetro de 32 mm. El material es caucho o semejante, para garantizar buen un agarre con el suelo. Los rines de aluminio se sujetan con tornillos de presión al eje de los motores.

En la Figura 3.14 se muestra la distribución grafica de los componentes sobre la base. Se utilizan *headers* para conectar el microcontrolador, el driver y las salidas de los encoders. Los motores, batería y rueda de castor son ensamblados por medio de tornillos. La batería se monta superficialmente. Por último, el resto de componentes (resistores, leds, etc) serán soldados a la placa.

3.4.2. Vistas del diseño

En esta sección se presentan las vistas del diseño renderizado en Solidworks®. En la Figura 3.15 se muestran las vistas frontal y trasera respectivamente. En estas dos se observan orificios en la parte baja que se pretenden utilizar (en una implementación futura) como las salidas y entradas de los sensores de proximidad.



(a) (b)

Figura 3.15: Vistas del RMD: a) frontal y b) trasera.

El orificio de la ventana frontal se deja como paso del flujo de aire con salida en los espacios de las llantas (ver Figura 3.15a), sirviendo como disipación. El chasis se diseña a partir de la base donde se ubican todos los componentes eléctricos. Por último, la vista isométrica se presenta en la Figura 3.16.

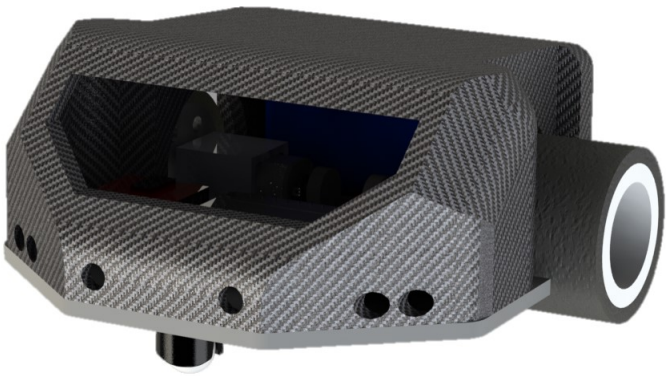


Figura 3.16: Vista isométrica del RMD.

Capítulo 4

Sistema Multiagente

En este capítulo se desarrolla el control cooperativo de un SMA compuesto de RMD. La topología de comunicación bidireccional de los agentes es representada mediante un grafo no dirigido, el cual es interpretado mediante la matriz Laplaciana asociada para implementar un control basado en consenso promedio sobre el controlador por planitud diferencial y control clásico en h. Posteriormente, son diseñados desfases de posición en el plano para cada agentes para realizar la formación de figuras o configuraciones diversas. Por último, se muestran resultados de simulaciones en el entorno matemático Matlab Simulink®.

4.1. Topología comunicación

Un conjunto de RMDs en el plano capaz de cumplir tareas o funciones de forma cooperativa debe transmitir y recibir información de las variables de estado de cada robot. La topología de comunicación representa el flujo de la información a través de una red de agentes capaces de interconectarse mediante un protocolo. Es posible representar la topología de comunicación entre agentes mediante un grafo. Un grafo no dirigido y una topología tipo árbol es un esquema que facilita el escalamiento, no presenta ciclos en el flujo de información y posee propiedades de estabilidad [70].

La Figura 4.1 muestra el grafo no dirigido en conjunto con una topología de árbol utilizado, el cual representa un SMA compuesto de $n = 10$ agentes, donde el agente A_i representa al i-ésimo RMD. Este grafo se compone de tres niveles, o generaciones de agentes, comenzando con el agente A_1 (líder), seguidos de los agentes A_2, A_3, A_4 (nivel intermedio) y terminando con los agentes A_5, A_6, A_7, A_8, A_9, A_{10} (nivel exterior). Extender el número de agentes requiere de seguir la misma lógica de comunicación mostrada en el grafo, de tal forma que cada agente (con excepción del nivel exterior) posea un grado de comunicación de tres, es decir, $d_i = 3$ $(i = 1, 2, 3, 4)$.

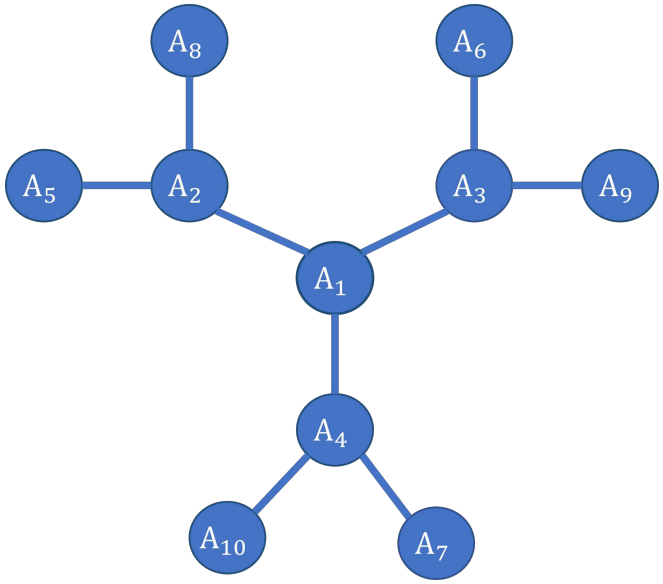


Figura 4.1: Grafo no dirigido de SMA compuesto de diez agentes RMD.

4.2. Diseño de control cooperativo

En el capítulo 3 se abordaron las técnicas de control de posición en el plano para un sólo RMD. En particular, el control por planitud diferencial transforman al modelo cinemático en c en un sistema de doble integrador (ver ecuación (3.29)) que posee la forma necesaria para implementar un consenso de segundo orden. En este punto el control por planitud diferencial actúa como un control externo y el consenso como un control interno.

Por un lado, es notable que el control clásico en h también permite la implementación de un consenso de primer orden, con lo cual se extiende su participación en esta investigación. Por otro lado, el control clásico en c no posee alguna particularidad en cuanto a control por consenso (ver capítulo 3) y se descarta de pruebas posteriores.

Lema 1 *Debido a que el control clásico h transforma al modelo cinemático h en un sistema de integradores simples, se puede entender que el control clásico en h linealiza el modelo cinemático h de forma exacta por realimentación estática.*

Con base en lo anterior, es posible realizar el consenso de un SMA compuesto de RMD que utilizan el modelo cinemático en h y en c: aplicando las leyes de control (3.17) y (3.24). La diferencia en el diseño radica en el orden del consenso, pues el control clásico en h necesita de un consenso promedio de primer orden, por otro lado, el control por planitud diferencial, un consenso de segundo orden. Considerando el grafo de la Figura 4.1, la matriz laplaciana asociada a éste, es

$$\mathcal{L} = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 3 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 3 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.2.1. Consenso promedio de RMDs

El objetivo del consenso promedio es que los agentes lleguen a una posición común en el plano, es decir, que coincidan sus valores en los ejes x y y (el promedio de sus condiciones iniciales). Por esta razón, es necesario desarrollar dos algoritmos de consenso para cada eje coordenado. La orientación de los RMDs será determinada por las propias leyes de control como se observó en las simulaciones del capítulo anterior. En esta sección el agente líder no funge como tal, ya que sólo se está empleando el consenso promedio y solamente se interpreta como un agente que interactúa con los demás.

Desarrollar los algoritmos de consenso requiere de la matriz laplaciana (4.1) y variables de interés (posiciones) de los agentes que conforman al SMA. Se considera $X = [x_1, x_2, \ldots, x_n]^T$ y $Y = [y_1, y_2, \ldots, y_n]^T$, donde x_i y y_i $(i = 1, 2, \ldots, 10)$ son las posiciones del i-ésimo agente en el plano. Se debe tener en cuenta que x_i puede referirse tanto a x_c como a x_h del i-ésimo agente, esto queda implícito dependiendo del controlador al que se haga referencia.

Consenso promedio modelo cinemático en h

Considere un SMA compuesto de múltiples RMD con modelo cinemático en h al cual se le aplican las leyes de control (3.17). Entonces, las variables de control auxiliar (3.16) se reescriben, según el consenso promedio de primer orden (2.8), como

$$\begin{bmatrix} r_{x\,1} \\ r_{x\,2} \\ \vdots \\ r_{x\,i} \end{bmatrix} = -\mathcal{L}\lambda_p X \; ; \quad \begin{bmatrix} r_{y\,1} \\ r_{y\,2} \\ \vdots \\ r_{y\,i} \end{bmatrix} = -\mathcal{L}\lambda_p Y \tag{4.2}$$

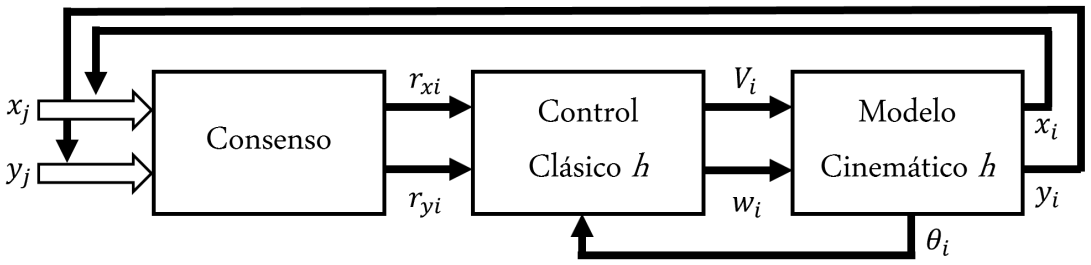


Figura 4.2: Diagrama de bloques del control en lazo cerrado del i-ésimo agente con modelo en h

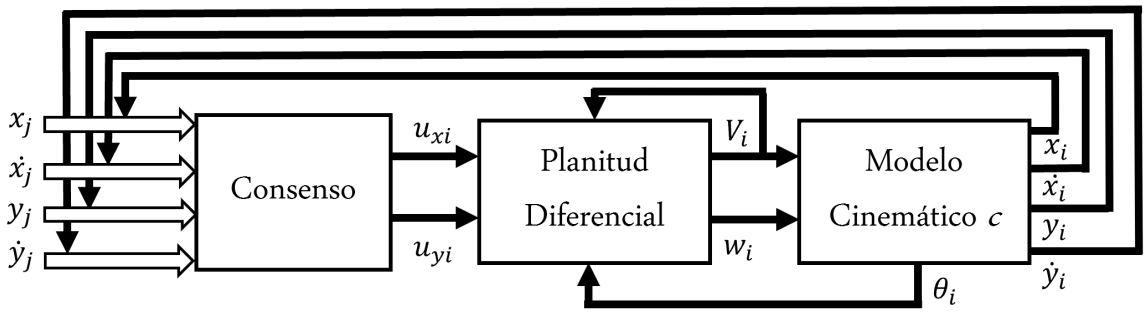


Figura 4.3: Diagrama de bloques del control en lazo cerrado del i-ésimo agente con modelo en c

donde $r_{xi}, r_{yi} \in \mathbb{R}s$ $(i = 1, 2, ..., 10)$ son la variables de control auxiliar del i-ésimo agente, $\lambda_p \in R$ es una ganancia positiva, x_j y y_j son las variables de estado de los agentes vecinos. Como ya se ha mencionado antes, esto es posible debido a que el modelo de cada agente se transforma en un sistema lineal de simple integrador. En la Figura 4.2, se muestra el diagrama de bloques del control cooperativo en lazo cerrado del SMA para el *i-ésimo* agente con modelo en h.

Consenso promedio modelo cinemático en c

Ahora bien, considere el caso de un SMA compuesto de RMD con modelo cinemático en c al cual se le aplican las leyes de control por planitud diferencial (3.24). Entonces, debido a que el modelo de cada agente se transforma en un sistema lineal de doble integrador, las variables de control auxiliar se reescriben, según el consenso promedio de segundo orden (2.9), como

$$\begin{bmatrix} u_{x1} \\ u_{x2} \\ \vdots \\ u_{xi} \end{bmatrix} = -[\mathcal{L} \;\; \mathcal{L}] \begin{bmatrix} \lambda_0 X \\ \lambda_1 \dot{X} \end{bmatrix} \; ; \quad \begin{bmatrix} u_{y1} \\ u_{y2} \\ \vdots \\ u_{yi} \end{bmatrix} = -[\mathcal{L} \;\; \mathcal{L}] \begin{bmatrix} \lambda_0 Y \\ \lambda_1 \dot{Y} \end{bmatrix} \tag{4.3}$$

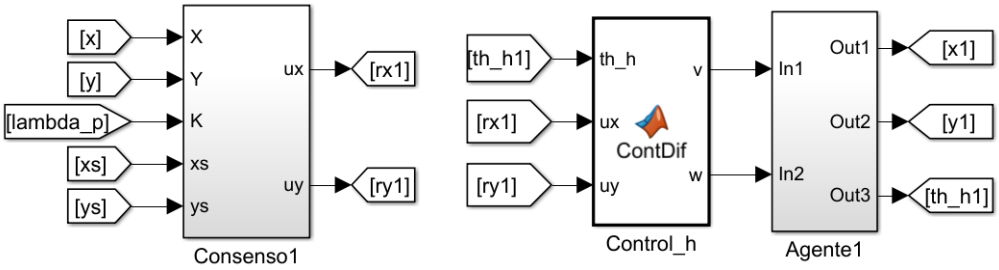


Figura 4.4: Bloques de simulación del agente A_1 para consenso promedio por control clásico en h.

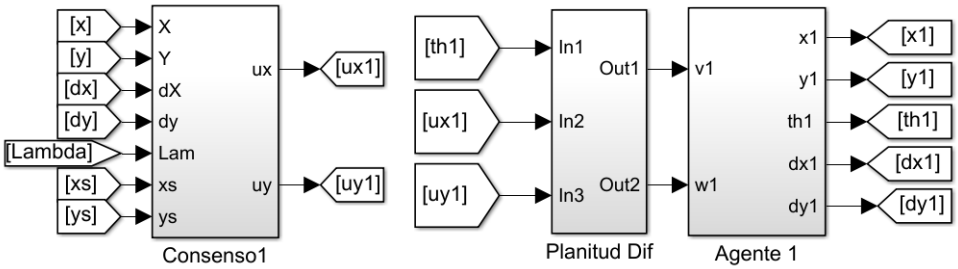


Figura 4.5: Bloques de simulación del agente A_1 para consenso promedio por planitud diferencial.

donde $u_{xi}, u_{yi} \in R$ $(i = 1, 2, ..., 10)$ son las variables de control auxiliar del i-ésimo agente y $\lambda_0, \lambda_1 \in R$ son ganancias sintonizadas con un polinomio *Hurwitz* de segundo orden. En la Figura 4.3, se muestra el diagrama de bloques del control cooperativo en lazo cerrado del SMA para el *i-ésimo* agente con modelo en c. La forma de las ecuaciones (4.3) poseen factores proporcionales y derivativos; las ecuaciones (4.2), sólo proporcionales.

Observe como los vectores X, Y y sus derivadas entran al bloque de consenso, sin embargo, dentro de éste se selecciona la información de los agentes vecinos (x_j, y_j)según la matriz laplaciana $\mathcal{L}$ que representa al grafo mostrado en la Figura 4.1. A continuación, se explica como se logra la selección de información de los vecinos.

Simulaciones de consenso promedio

Con base en los diagramas de bloques de las Figuras 4.2 y 4.3, las ecuaciones de los modelos cinemáticos, controladores cinemáticos y las ecuaciones de consenso, se programan utilizando funciones definidas por el usuario en el entorno Matlab-Simulink®. Se crean entonces tres bloques por agente en el entorno: de consenso, de control y modelo del agente.

En la Figura 4.4 se observan los tres bloques del agente A_1 para realizar un *consenso pro-*

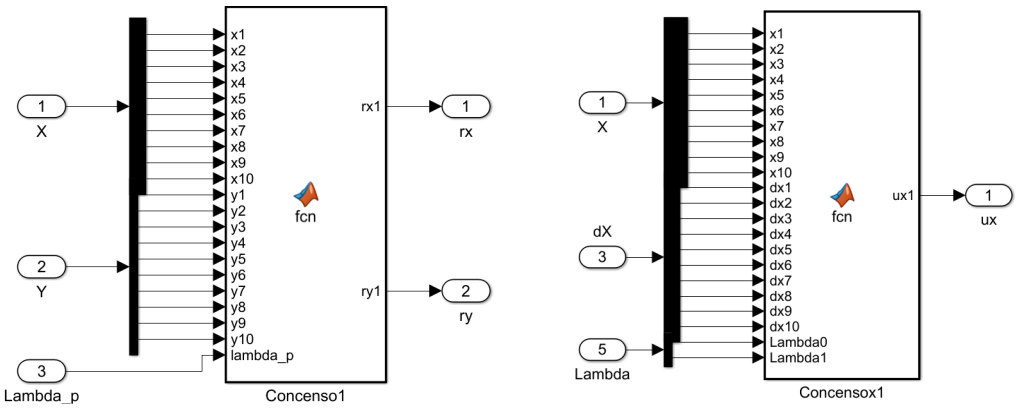


Figura 4.6: Bloque definido por usuario: a) Consenso en x y y, control clásico h. b) Consenso en x, planitud diferencial.

medio por control clásico en h. El resto de los agentes se diseñan igual, la única diferencia es la programación interna del bloque de consenso que le permite seleccionar la información. Dentro del bloque de consenso se encuentra la función definida por usuario de la Figura 4.6a. Los vectores de estado X y Y entran a la función, sin embargo, dentro de ésta se programa la primer fila de la matriz laplaciana $\mathcal{L}$ que al multiplicarse por los vectores de estado dan como resultado los controles auxiliares r_{x1} y r_{y1}, es decir

$$r_{x\,1} = -\mathcal{L}_{(1,:)}\lambda_p X \tag{4.4}$$

$$r_{y\,1} = -\mathcal{L}_{(1,:)}\lambda_p Y \tag{4.5}$$

donde $\mathcal{L}_{(1,:)}$ representa la fila uno de la matriz laplaciana $\mathcal{L}$. La misma lógica de programación es seguida para el i-ésimo agente: se programa la i-ésima fila y se multiplica por los vectores de estado. Como resultado se obtiene una forma rápida de escalar el SMA: 1) programar un función de consenso como el de la Figura 4.6a con tantos estados x_i como se necesite; 2) copiar tantas veces se requiera y 3) programar la i-ésima fila de la matriz Laplaciana para cada agente.

En la Figura 4.5 se observan los bloques del agente A_1 para el *consenso promedio por planitud diferencial*. En la Figura 4.6b se muestra la función definida por usuario para el consenso en el eje x. Naturalmente, debe existir un bloque similar para el consenso en el eje y. El funcionamiento es el mismo que el descrito anteriormente. Entonces, las variables de control auxiliares u_{x1} y u_{y1}, son calculadas como

$$u_{x1} = -[\mathcal{L}_{(1,:)} \;\; \mathcal{L}_{(1,:)}] \begin{bmatrix} \lambda_0 X \\ \lambda_1 \dot{X} \end{bmatrix} \tag{4.6}$$

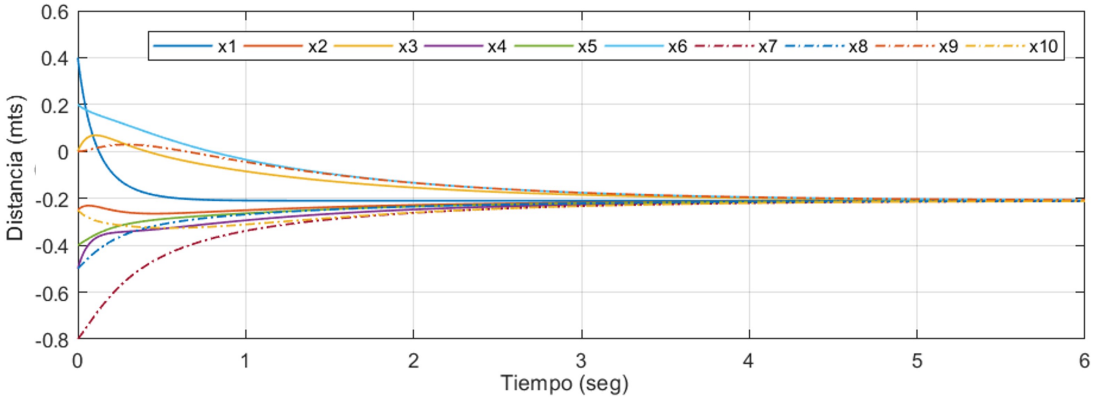


Figura 4.7: Consenso promedio por control clásico h en eje x

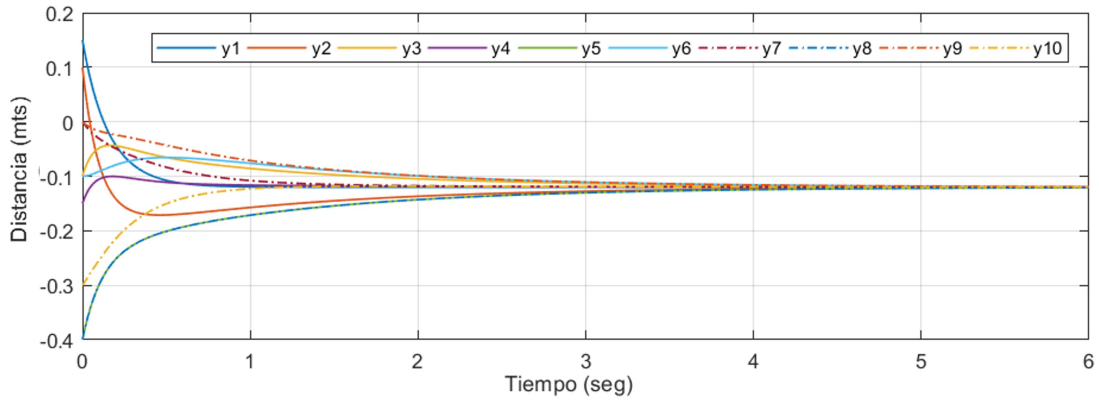


Figura 4.8: Consenso promedio por control clásico h en eje y

$$u_{y1} = -[\mathcal{L}_{(1,:)} \;\; \mathcal{L}_{(1,:)}] \begin{bmatrix} \lambda_0 Y \\ \lambda_1 \dot{Y} \end{bmatrix} \tag{4.7}$$

Los parámetros de simulación son: 1) paso fijo, 2) *solver* Runge Kuta y 3) tolerancia de $1\,E-4$. Las condiciones iniciales de posición de los agentes son: $X(0) = [0.4, -0.25, 0, -0.5, -0.4, 0.2, -0.8, -0.5, 0, -0.25]^T$, $Y(0) = [0.15, 0{,}1, -0.1, -0.15, -0.4, -0.1, 0, -0.4, 0, -0.3]^T$, $\dot{X}(0) = 0_{(10,1)}$ y $\dot{Y}(0) = 0_{(10,1)}$.

Por un lado, los resultados de consenso promedio por control clásico h se muestran en las Figuras 4.7 y 4.8. En éstas se muestra el desplazamiento de cada uno de los agentes en el eje x y en el eje y respectivamente. La ganancia de las ecuaciones (4.2) se selecciona con un valor de $\lambda_p = 3$.

Por otro lado, los resultados de consenso promedio por planitud diferencial se muestran en las Figuras 4.9 y 4.10. Las ganancias de las ecuaciones (4.3) son $\lambda_0 = 9$ y $\lambda_1 = 6$. Natu-

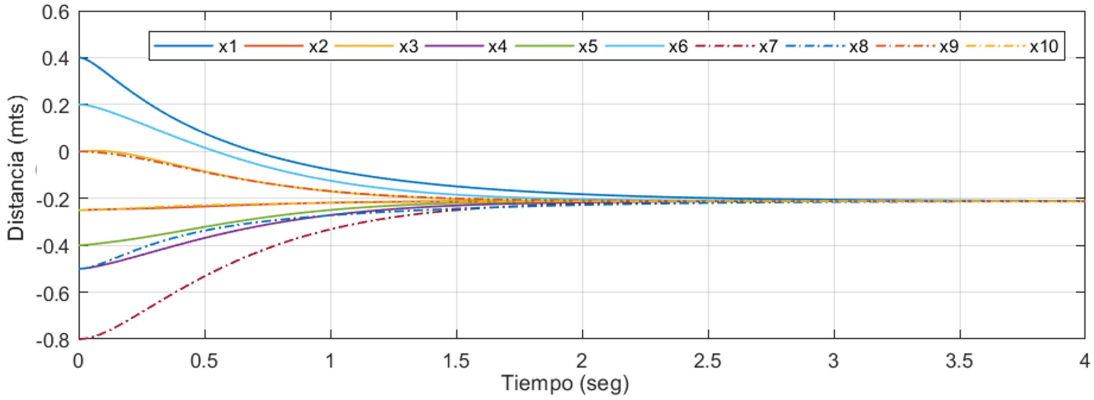


Figura 4.9: Consenso promedio por planitud diferencial en eje x

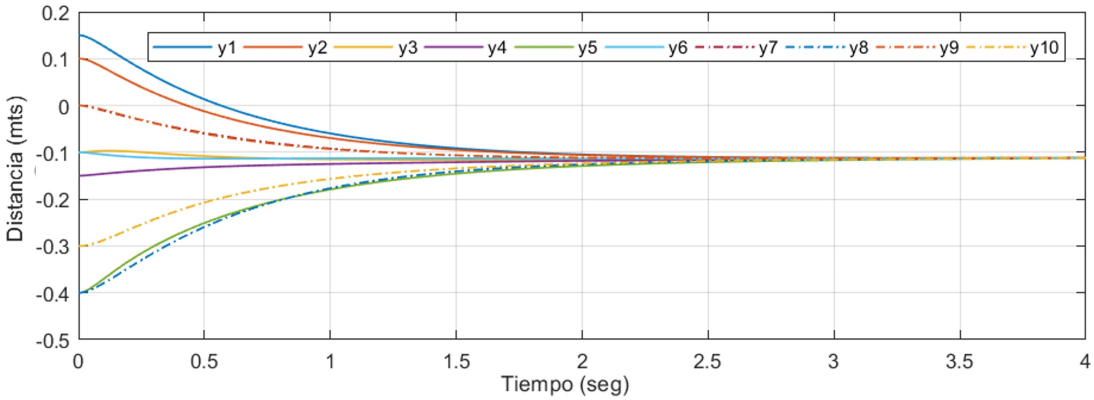


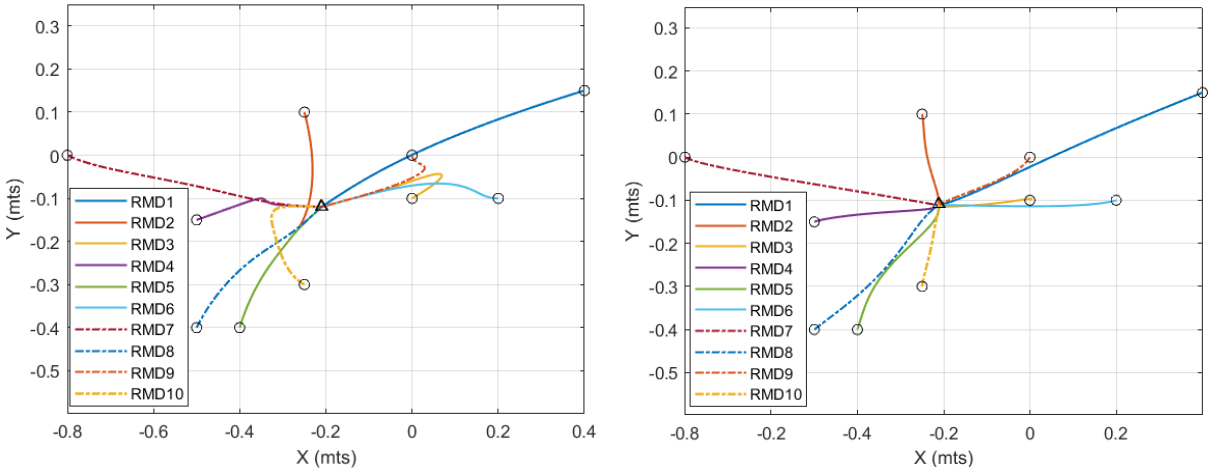
Figura 4.10: Consenso promedio por planitud diferencial en eje y

ralmente, $V(0) = 0.01$ para cada uno de los agentes.

En la Figura 4.11 se muestra la respuesta del consenso promedio para ambos controladores en el plano. Los agentes llegan a un valor común en un tiempo finito. Este valor se calcula utilizando la ecuación (2.5), es decir, $\bar{x} = -0.21$ y $\bar{y} = -0.12$. Es notable que los RMD de la Figura 4.11b se desplazan de sus posiciones iniciales (círculos negros) a su destino (triángulo negro) con menos maniobras en comparación con la Figura 4.11a. Esto debido a que se aprovecha la capacidad de orientar en cualquier dirección al instante (ver sección 3.3.1).

De las ecuaciones y simulaciones anteriores es importante mencionar que:

- Estos consensos no pueden extenderse de forma derivativa, es decir, el orden del consenso no puede superar al orden del sistema. Por ejemplo, el consenso (4.2) no puede extenderse para tener la forma del consenso (4.3), en caso contrario el consenso falla.



(a) (b)

Figura 4.11: Consenso promedio de diez agentes por: a) control clásico h. b) planitud diferencial.

- Es posible incorporar un factor integral a los consensos (4.2) y (4.3) para corregir el error, debido a perturbaciones. Sin embargo, las perturbaciones cambian el valor final del consenso, es decir, el consenso converge a otro valor distinto del promedio.

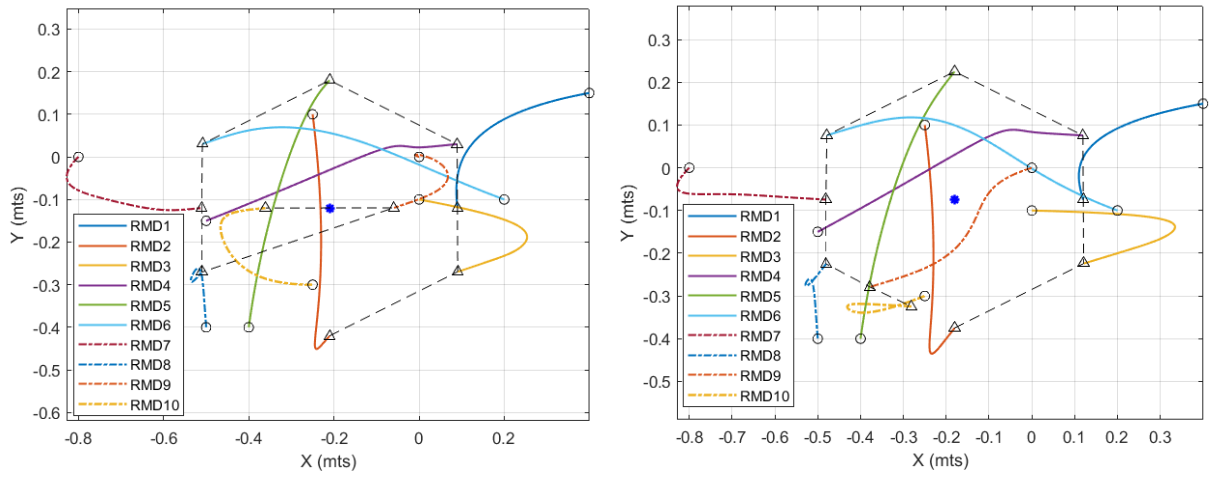
- En las ecuaciones auxiliares (4.3) no pueden ser puramente proporcionales, en caso contrario el consenso falla. De lo anterior podemos establecer que

  Lema 2 *El orden del consenso debe poseer el mismo orden del sistema.*

- En la Figura 4.9 y 4.10 consenso promedio es alcanzado pero súbitamente comienza a moverse, es decir, todos los agentes se desplazan conservando el mismo valor. Este fenómeno se atribuye a la condición inicial de velocidad vista en la sección 3.3.1, pues no cumple con los requerimientos del consenso promedio. La interpretación de este suceso es que se alcanza un consenso de velocidad lineal V, provocando que después de alcanzar un valor común de posición éste se modifique y no permanezca estático. En otras palabras este comportamiento no es el de un consenso promedio, sino un *consenso de posición y velocidad lineal*.

4.2.2. Formación de RMD

Realizar la formación de un SMA consiste en establecer distancias (desfases) entre los agentes de tal manera que cada uno se establezca en una posición individual. Estas separaciones se realizan en relación al punto de consenso que se alcanzaría, es decir, **la referencia**



(a) (b)

Figura 4.12: Formación hexagonal con suma de desfases: a) igual a cero. b) distinto de cero.

o centro de la formación es el consenso promedio. En esta investigación se posiciona de forma cartesiana a cada agente, es decir, especificar una distancia en x y y. Sea $l_x = [l_{x1}, l_{x2}, \ldots, l_{xn}]^t$ y $l_y = [l_{y1}, l_{y2}, \ldots, l_{yn}]^t$ vectores de desfase de posición, donde l_{xi} y l_{yi} $(i = 1, 2, \ldots, 10)$ son los desfases del i-ésimo agente en el eje x y eje y respectivamente. Entonces se reescriben los vectores de estado como

$$X_l = X - l_x \tag{4.8}$$

$$Y_l = Y - l_y \tag{4.9}$$

donde X_l y Y_l son los *vectores de estado para formación*. Este cambio de variables puede alterar, o no, el consenso promedio: 1) si la suma de todos los desfases tanto en x $(\sum l_{xi})$ en y $(\sum l_{yi})$ son iguales a cero, entonces el consenso promedio (2.5) es alcanzado; 2) si es diferente de cero, el consenso promedio en cualquiera o ambos ejes se modifica de acuerdo a las siguientes ecuaciones

$$\begin{aligned} \bar{x} &= \frac{1}{n}\sum_{i=0}^{n} x_i(0) - \frac{1}{n}\sum_{i=0}^{n} l_{xi} \\ \bar{y} &= \frac{1}{n}\sum_{i=0}^{n} y_i(0) - \frac{1}{n}\sum_{i=0}^{n} l_{yi} \end{aligned} \tag{4.10}$$

La Figura 4.12 muestra los dos casos de formación en el plano para un SMA de diez agentes para una figura hexagonal. Como ejemplo, el SMA está compuesto de RMDs con modelo cinemático en h. Las condiciones iniciales son las mismas que en la sección anterior. Los desfases para el primer de caso de formación (Figura 4.12a) son: $l_x = [0.3, 0, 0.3, 0.3, 0, -0.3, -0.3,$

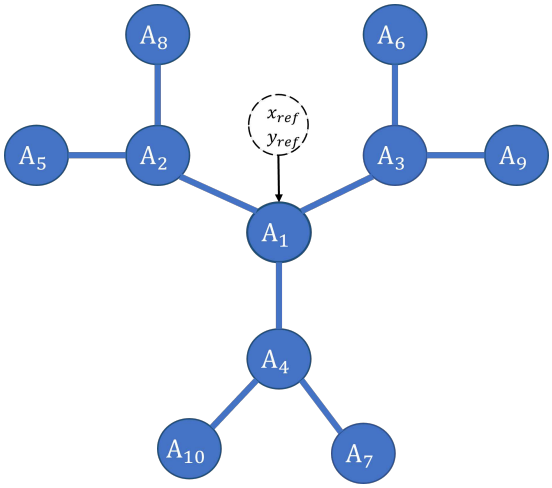


Figura 4.13: Grafo de SMA bajo esquema líder-seguidor.

$-0.3, 0.15, -0.15]^t$ y $l_y = [0, -0.3, -0.15, 0.15, 0.3, 0.15, 0, -0.15, 0, 0]^t$. La formación mantiene el consenso promedio ($\bar{x} = -0.21, \bar{y} = -0.12$) como su centro (punto azul) pues los desfases suman cero.

Para el segundo caso de formación: $l_x = [0.3, 0, 0.3, 0.3, 0, -0.3, -0.3, -0.3,\ 0.15, -0.15]$ y $l_y = [0, -0.3, -0.15, 0.15, 0.3,\ 0.15, 0, -0.15, -0.205, -0.25]$. Observe en la Figura 4.12b como el punto de consenso (punto azul) es alterado cuando la sumatoria de los desfases es distinta de cero y, según la ecuación (4.10), se dirige a $\bar{x} = -0.018$ y $\bar{y} = -0.0745$. En ambos casos se habla de un *consenso de formación*: promedio de condiciones iniciales y desfases.

4.2.3. Esquema Líder-Seguidor

En este punto, las ecuaciones consideran al SMA de diez agentes RMD como un sistema que llega a un acuerdo de variables de interés, con base en sus condiciones iniciales y suma de desfases. El enfoque Líder-Seguidor propone la existencia de un agente líder que funge como guía de los agentes seguidores. Este agente comparte información de su posición a un grupo selecto de agentes seguidores y es el único que recibe información de las referencias deseadas x_{ref} y y_{ref}[91] o marco de referencia virtual [70] (círculo punteado de la Figura 4.13). En este trabajo se selecciona al nodo A_1 como el agente líder y los nodos restantes son seguidores.

Las variables de control auxiliares son reescritas solamente para el RMD líder, pues ahora recibe información que afecta su posición en el plano, además del consenso. La variables de control auxiliar del agente líder con modelo en h son

$$r_{x\,1} = -\mathcal{L}_{(1,:)}\lambda_p X_l + r_x \quad ; \quad r_{y\,1} = -\mathcal{L}_{(1,:)}\lambda_p Y_l + r_y \tag{4.11}$$

Las variables de control auxiliar del agente líder con modelo en c son

$$u_{x1} = -[\mathcal{L}_{(1,:)} \;\; \mathcal{L}_{(1,:)}] \begin{bmatrix} \lambda_0 X_l \\ \lambda_1 \dot{X}_l \end{bmatrix} + u_x \quad ; \quad u_{y1} = -[\mathcal{L}_{(1,:)} \;\; \mathcal{L}_{(1,:)}] \begin{bmatrix} \lambda_0 Y_l \\ \lambda_1 \dot{Y}_l \end{bmatrix} + u_y \tag{4.12}$$

donde (r_x, r_y) y (u_x, u_y) son las ecuaciones (3.18) y (3.26) respectivamente. Estas dos ecuaciones son controles de seguimiento en el plano para un RMD (véase el capítulo 3), donde las referencias a seguir, x_{ref} y y_{ref}, serán las posiciones o trayectorias deseadas para el agente líder. Note que el agente líder, debido a los términos (r_x, r_y) o (u_x, u_y), se desplazará hacia las referencias deseadas de posición alterando el consenso. Al aplicar las leyes de control (4.11) y (4.12), el consenso promedio (2.5) y el consenso de formación (4.10) dejan de existir para reemplazarse por un *consenso Líder-seguidor*. Lo anterior significa que: 1) si el SMA intenta realizar un consenso promedio, todos los agentes irán a la posición del agente líder ignorando las condiciones iniciales; 2), si el SMA intenta realizar un consenso de formación, el centro de la formación (o referencia) será el agente líder restando sus desfases de posición l_{x1} y l_{y1}.

4.3. Simulaciones

En esta sección se presentan los resultados de simulación del consenso bajo el esquema líder-seguidor. Los controladores cinemáticos utilizados son: clásico en h y planitud diferencial. Utilizando las funciones definidas por usuario se implementan las ecuaciones para el consenso líder-seguidor. Por una parte, al bloque de consenso de la Figura 4.4, se programan las ecuaciones (3.18) y se inyecta hasta la primer derivada de las posiciones deseadas en el plano. Por otra, al bloque de consenso de la Figura 4.5, se programan las ecuaciones (3.26) y se inyecta hasta la segunda derivada de posiciones deseadas en el plano. Por último, se compara la respuesta de ambos controladores ante la regulación de posición y seguimiento de trayectoria en el plano.

4.3.1. Indicadores de desempeño

Para evaluar el desempeño de ambos controladores, se calcula la integral del error cuadrático (ISE, por sus siglas en inglés) y la integral del error absoluto (IAE, por sus siglas en inglés). Se busca un valor mínimo en ambos indicadores.

Las componentes del indicador ISE del i-ésimo agente se define como:

$$ISE_{xi} = \int (x_{ref} - x_i + l_{xi})^2 dt \; ; \quad ISA_{yi} = \int (y_{ref} - y_i + l_{yi})^2 dt \tag{4.13}$$

Las componentes del indicador IAE del i-ésimo agente se define como:

$$IAE_{xi} = \int |x_{ref} - x_i + l_{xi}|dt \; ; \quad IAE_{yi} = \int |y_{ref} - y_i + l_{yi}|dt \tag{4.14}$$

Las componentes del indicador ISE del SMA se define como:

$$ISE_X = \sum_{i=1}^{N} ISE_{xi} \; ; \quad ISE_Y = \sum_{i=1}^{N} ISE_{yi} \tag{4.15}$$

Las componentes del indicador IAE del SMA se define como:

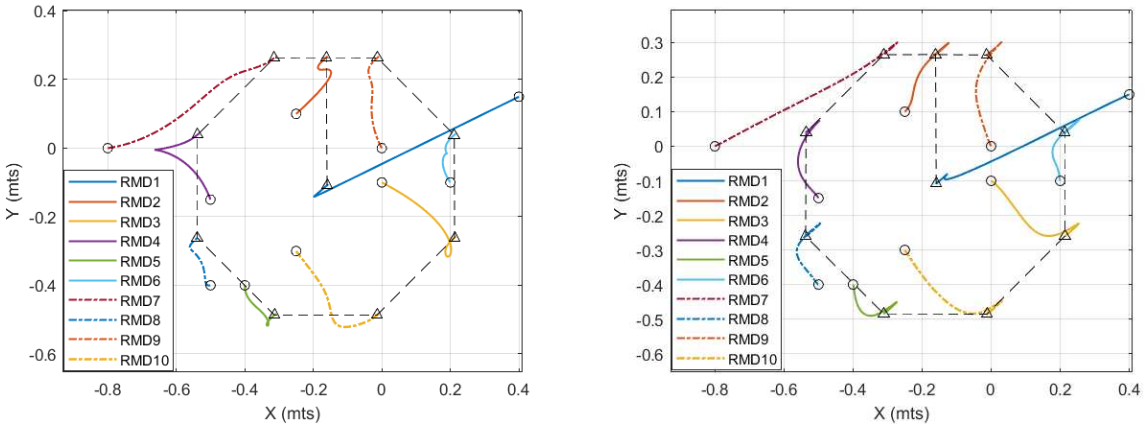
$$IAE_X = \sum_{i=1}^{N} IAE_{xi} \; ; \quad IAE_Y = \sum_{i=1}^{N} IAE_{yi} \tag{4.16}$$

La magnitud de los indicadores ISE e IAE del SMA se define como:

$$ISE = \sqrt{(ISE_X)^2 + (ISE_Y)^2} \; ; \quad IAE = \sqrt{(IAE_X)^2 + (IAE_Y)^2} \tag{4.17}$$

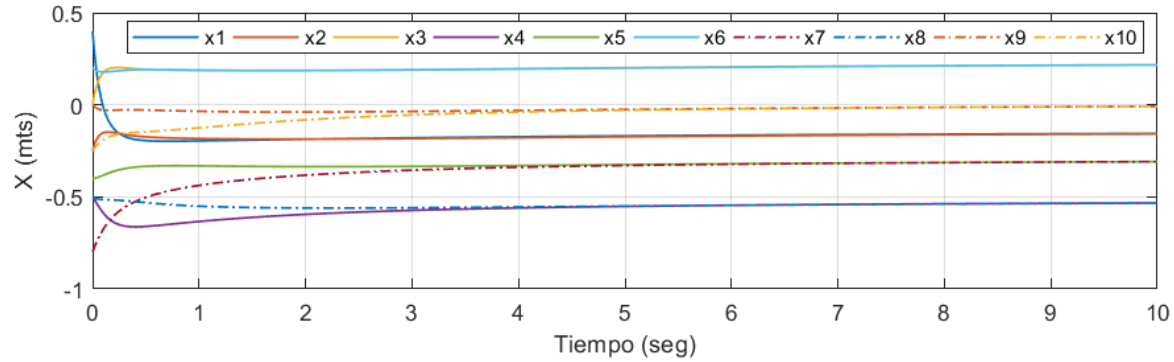
4.3.2. Regulación de posición del SMA

Los parámetros de simulación y las condiciones iniciales de los agentes, son los mismos que los propuestos en la sección 4.2.1. Los desfases de posición para la formación de una figura octagonal son: $l_x = [0, 0, 0.375, -0.375, -0.15, 0.375, -0.15, -0.375, 0.15, 0.15]^t$ y

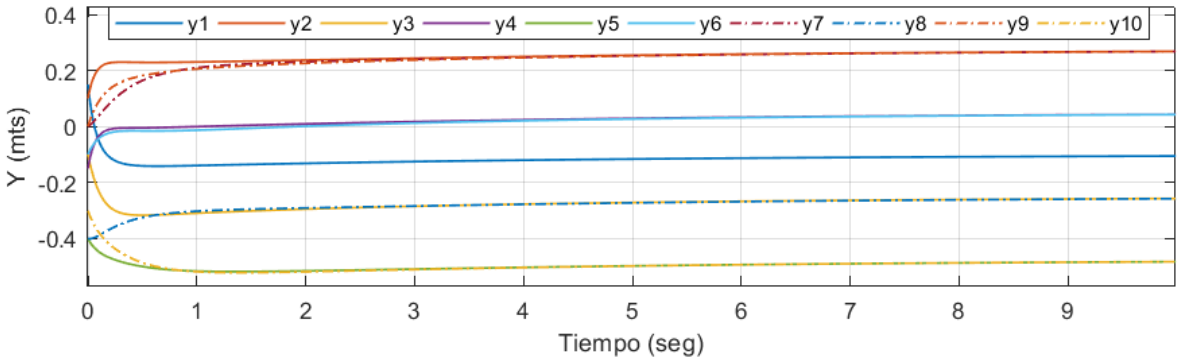


(a) (b)

Figura 4.14: Formación octagonal con regulación líder-seguidor: a) control clásico h. b) planitud diferencial.



(a)



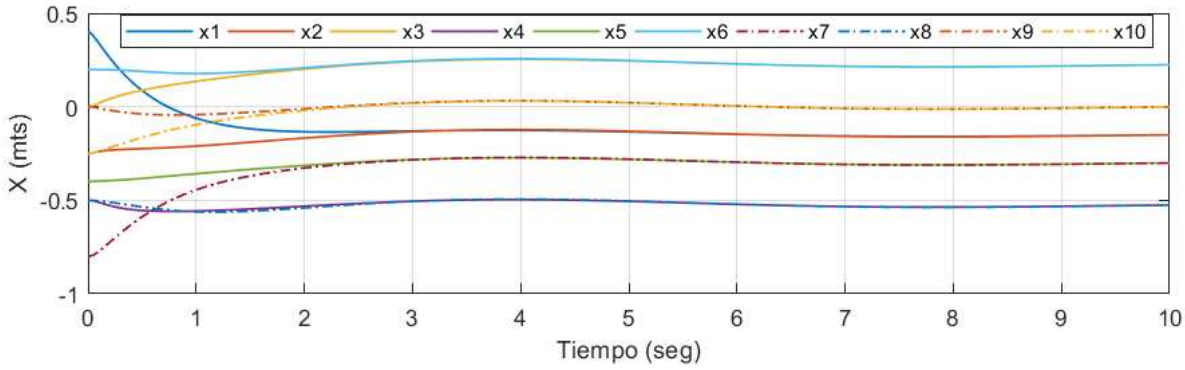
(b)

Figura 4.15: Formación y regulación por control clásico h: a) eje x. b) eje y.

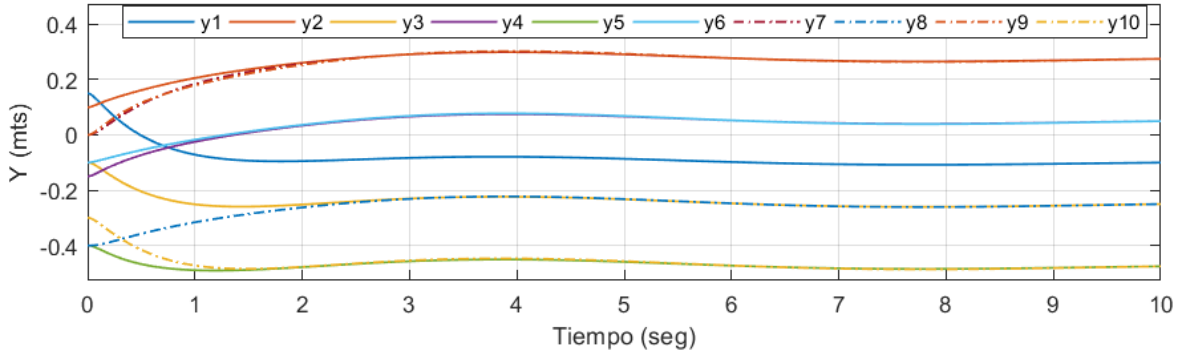
$l_y = [0, 0.375, -0.15, 0.15, -0.375, 0.15, 0.375, -0.15, 0.375, -0.375]^t$. Se seleccionan las posiciones deseadas $x_{ref} = -0.15$ y $y_{ref} = -0.1$. Debido a que se trata de una regulación, las derivadas sucesivas de las posiciones deseadas son cero.

En la Figura 4.14a se observa la respuesta de posición en el plano del SMA bajo el consenso líder-seguidor por acción del control clásico h. Las gráficas de posición en los ejes coordenados se observa en la Figura 4.15. Las ganancias utilizadas son $\lambda_p = 3$ y $k_p = 3$. El agente líder presenta un sobreimpulso antes de establecerse en la posición deseada, ésto es debido a la interferencia del consenso. Los agentes seguidores también presentan un ligero sobreimpulso al llegar a los desfases deseados. El agente líder es el centro de la formación ya que sus desfases son $l_x = 0$ y $l_y = 0$, es decir, $(x_1, y_1) = (x_{ref}, y_{ref})$.

Los resultados de posición del SMA bajo el consenso líder seguidor por acción del control por planitud diferencial, se muestra en la Figura 4.14b. Las ganancias utilizadas son $\lambda_0 = 9$, $\lambda_1 = 6$ y $\gamma_0 = 9$, $\gamma_1 = 6$. Este consenso muestra un comportamiento oscilatorio del agente líder al llegar a la referencia, además se observa un sobreimpulso en los agentes



(a)



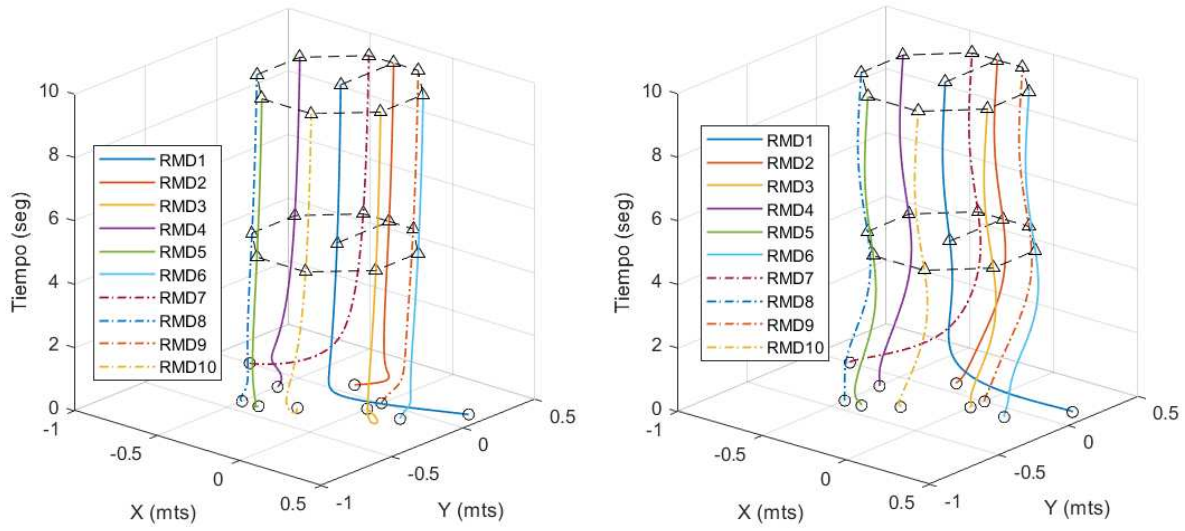
(b)

Figura 4.16: Formación y regulación por planitud diferencial: a) eje x. b) eje y.

seguidores al momento de llegar a las posiciones deseadas. Se observan menos maniobras del RMD1, RMD2 y RMD3 para poder establecer la formación (ver Figura 4.17).

Con la finalidad de observar la evolución del fenómeno de formación en el plano, se introducen dos gráficas (ver Figura 4.17), que muestran la posición en el plano del SMA contra tiempo. La Figura 4.17a muestra la evolución del consenso líder-seguidor por control clásico h; la Figura 4.17b, por planitud diferencial. En esta última gráfica es posible ver que el control por planitud diferencial provoca un sobreimpulso en dirección al tercer cuadrante, para luego establecerse en la posición deseada. Mientras tanto, en el control clásico h, sólo se inclina suavemente hacia la posición final.

Las magnitudes de los indicadores de desempeño para el control clásico h son: ISE=0.2521 e IAE=4.17. Las magnitudes de los indicadores de desempeño para el control por planitud diferencial son: ISE=0.3573 e IAE=3.43. Lo cual sugiere una superioridad por parte del control clásico h en cuanto al indicador ISE.

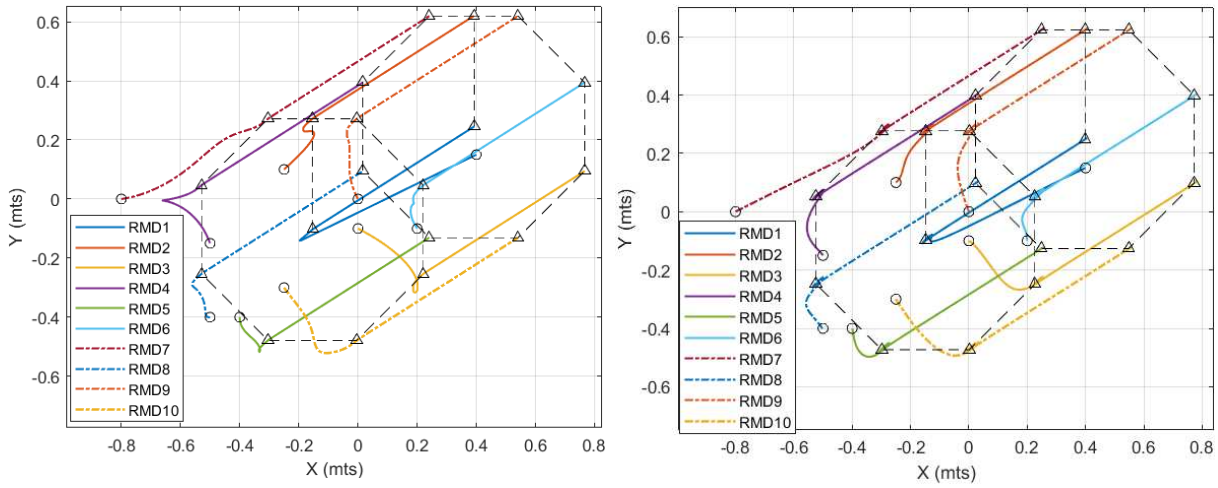


(a) (b)

Figura 4.17: Formación y regulación del SMA contra el tiempo: a) control clásico *h*. b) planitud diferencial.

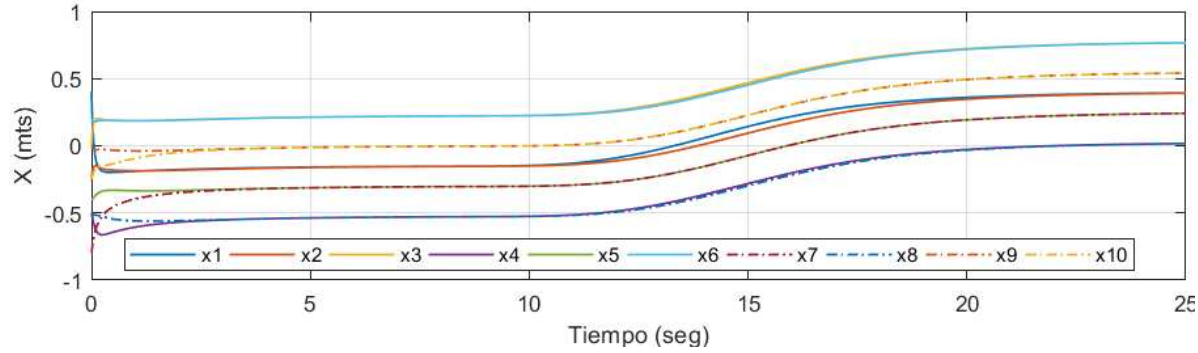
4.3.3. Seguimiento de trayectoria del SMA

Una vez alcanzada la formación en una posición específica, es deseable que ésta se pueda conservar mientras el agente líder la guía a través de una trayectoria en el plano. Para lograr ésto, es necesaria la información de derivadas sucesivas de las posiciones deseadas. Por lo

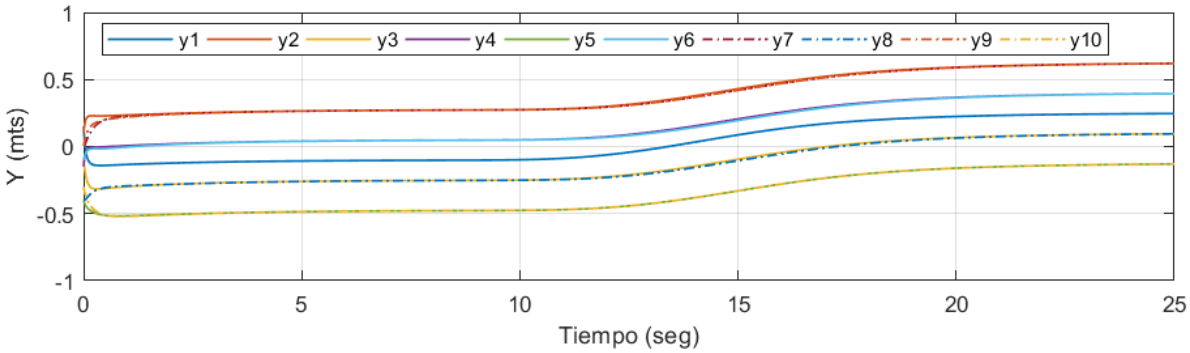


(a) (b)

Figura 4.18: Formación y seguimiento líder-seguidor: a) control clásico *h*. b) planitud diferencial.



(a)



(b)

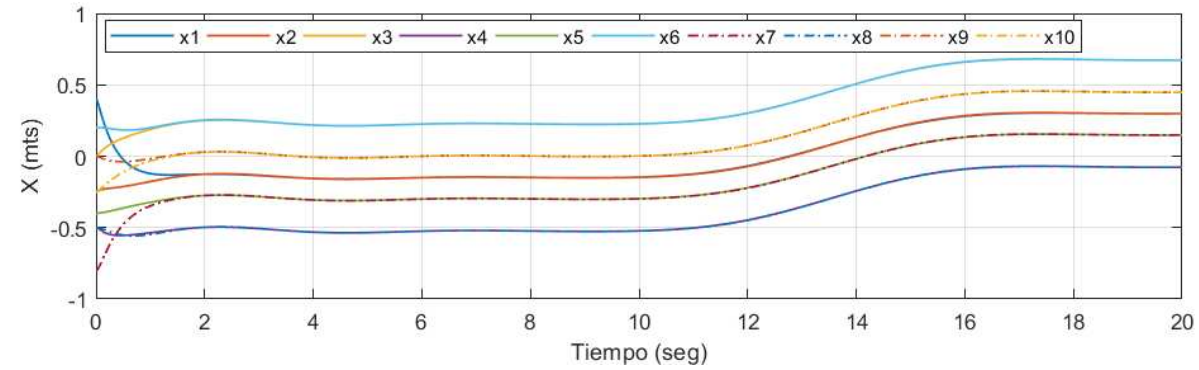
Figura 4.19: Formación y seguimiento por control clásico h: a) eje x. b) eje y.

tanto, se escoge una trayectoria suave: una curva de *Bézier* de la forma vista en la sección 3.3.2, es decir, se espera que el SMA trace una línea recta suave en el plano que se una los puntos deseados inicial y final.

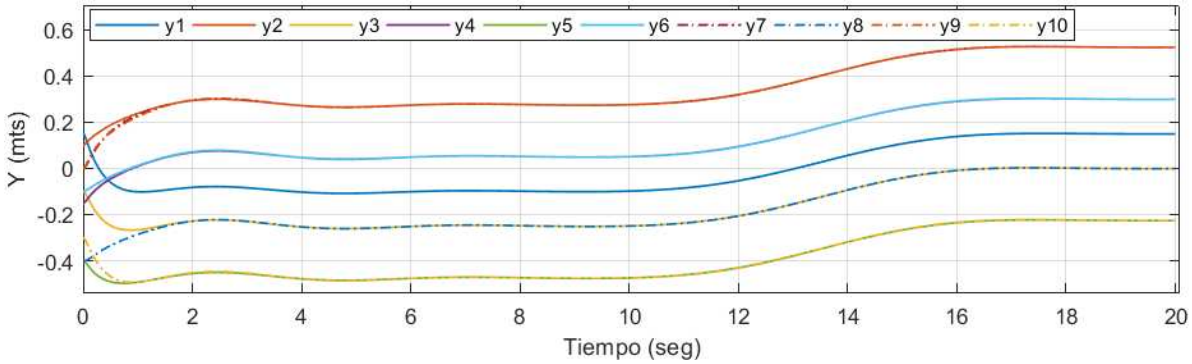
Las posición inicial de la curva son $x_{ini} = -0.15$ y $y_{ini} = -0.1$; la posición final, $x_{fin} = 0.4$ y $y_{fin} = 0.25$. La formación es la misma que en la sección anterior. La simulación completa se compone de dos etapas: 1) $(0 \leq t < 8)$, el SMA realiza su formación mientras se regula hacia la posición deseada inicial; y 2) $(8 \leq t < 20)$, el SMA comienza a seguir la trayectoria, es decir, el SMA llega suavemente a la posición final deseada en 12 segundos.

En la Figura 4.18a se observan las posiciones en el plano de los agentes por acción del control clásico h. Las ganancias utilizadas son $\lambda_p = 5$ y $k_p = 5$. Las posiciones en los ejes coordenados se muestra en la Figura 4.19. La evolución de la formación y seguimiento de trayectoria, se observa en la Figura 4.21a.

En la Figura 4.18b se observan las posiciones en el plano de los agentes por acción de planitud diferencial. Las ganancias seleccionadas para esta tarea de seguimiento son $\lambda_0 = 25$, $\lambda_1 = 10$ y $\gamma_0 = 25$, $\gamma_1 = 10$. Las posiciones en los ejes coordenados se muestra en la Figura 4.20. La



(a)



(b)

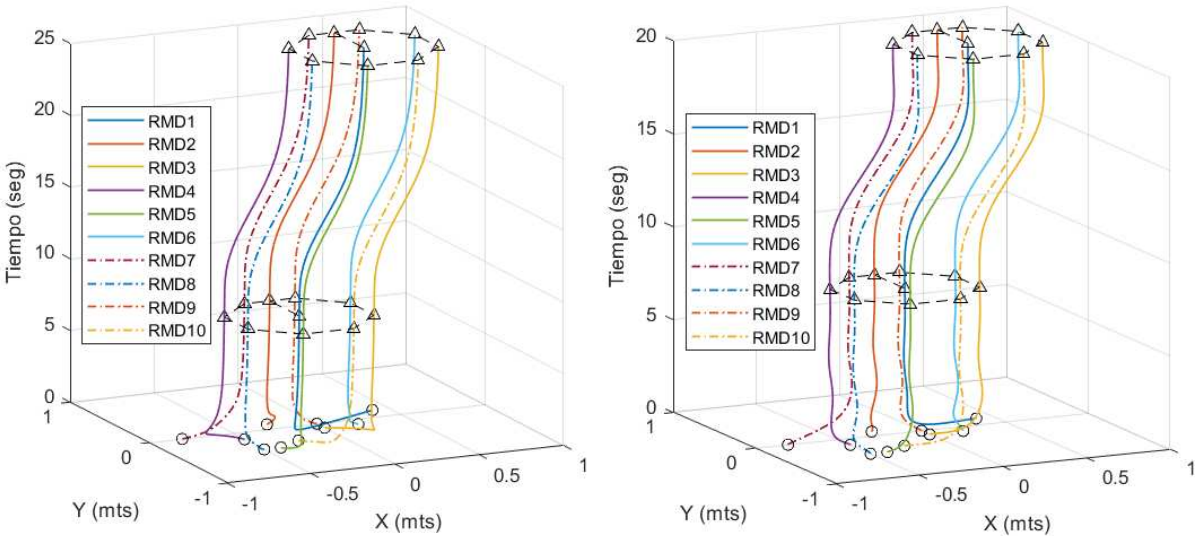
Figura 4.20: Formación y seguimiento por planitud diferencial: a) eje x. b) eje y.

evolución de la formación y seguimiento de trayectoria, se observa en la Figura 4.21b.

Con base en las respuestas de esta sección y la anterior, se concluye en los siguientes puntos:

- Las magnitudes de los indicadores de desempeño durante la formación ($0 < t \leq 8$) del control clásico h son: ISE=0.1521 y IAE=2.65. Las magnitudes de los indicadores del control planitud diferencial son: ISE=0.2138 e IAE=2.14. Nuevamente, el control clásico h es superior en cuanto al indicador ISE y el control por planitud diferencial es superior en cuanto al indicador IAE.

- Ambos controladores presentan errores en la tarea de seguimiento (ver Figura 4.22). Las magnitudes de los indicadores de desempeño durante la formación ($8 < t \leq 20$) del control clásico h son: ISE=1.99 e IAE=14.455. Las magnitudes de los indicadores del control planitud diferencial son: ISE=0.0233 e IAE=1.512. Con lo cual el control por planitud diferencial es superior en la tarea de seguimiento. Al aumentar la frecuencia natural w_n, el SMA reduce su error de seguimiento. Sin embargo, el aumentar la ganancia produce efectos no deseados en la aplicación experimental.



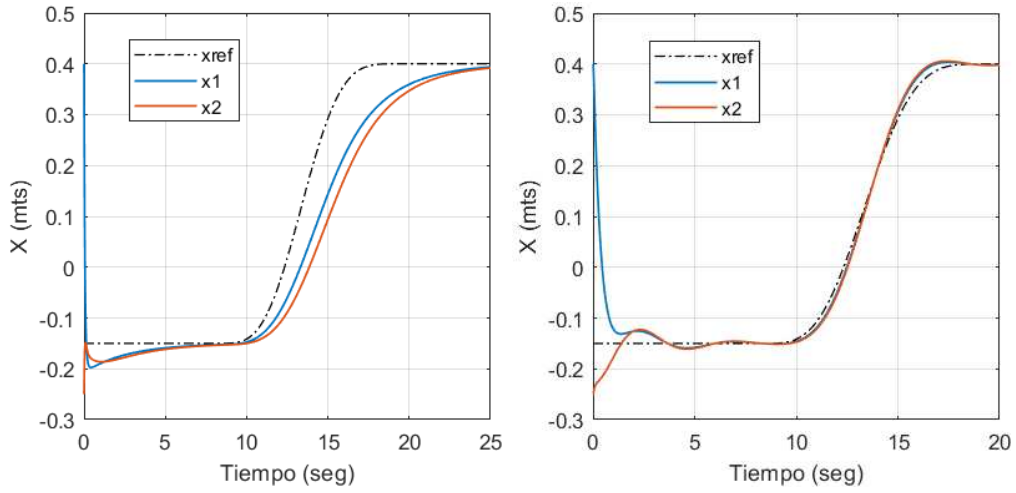
(a) (b)

Figura 4.21: Formación y seguimiento del SMA contra el tiempo: a) control clásico h. b) planitud diferencial.

- El hecho de que el SMA, con ambos controladores, no pueda seguir fielmente la trayectoria deseada se atribuye al algoritmo de consenso, pues éste representa una perturbación para las componentes r_x, r_y, u_x y u_y, destinadas al seguimiento de la trayectoria de referencia. El agente líder intenta guiar al SMA a la referencia deseada mientras negocia la formación con sus seguidores. En cuanto el error de formación (o de consenso) sea inferior al de seguimiento, en ese momento el SMA comenzará a seguir la trayectoria deseada.

- Se nota la existencia de un retraso en el seguimiento entre el agente líder y el agente A_2 (RMD2) (ver Figura 4.22). Debido a lo mencionado en el punto anterior y a la forma del grafo seleccionado 4.1, el SMA se sujeta al flujo de información desde el agente líder hasta las ramas exteriores, como un fenómeno de tráfico en carretera. Por lo tanto, los agentes de la primer rama (A_2, A_3, A_4) presentan un retraso con el respecto al líder, y los agentes de la segunda rama $(A_5, A_6, A_7, A_8, A_9, A_{10})$ presentan un retraso con respecto de la primer rama.

Considerando lo mencionado anteriormente, en las tareas de regulación de posición del SMA, el control clásico h presenta un comportamiento superior en cuanto a error de regulación y establecimiento de la formación. Sin embargo, en la tarea de seguimiento, el control por planitud diferencial es superior pues presenta un mejor ajuste al polinomio deseado y un estableciendo dentro del tiempo requerido.



(a) (b)

Figura 4.22: Seguimiento en x, agentes A_1 y A_2: a) control clásico h. b) planitud diferencial

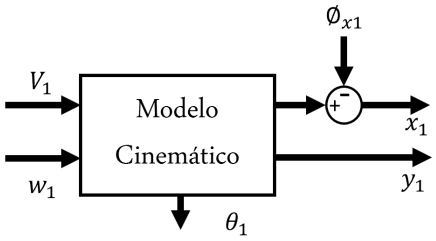


Figura 4.23: Diagrama de bloques de perturbación.

4.3.4. Perturbaciones

En este apartado se abarca de forma breve la respuesta del SMA ante perturbaciones de posición. En particular se desea observar cómo responde el sistema completo, cuando el agente líder recibe una perturbación en su posición. Se considera que el SMA realiza la misma tarea de seguimiento que en la sección anterior y que el agente líder percibe un desplazamiento súbito de posición en el eje x debido a una perturbación (ver Figura 4.23). Sea $\phi_{x1}(t)$ la perturbación acotada del agente A_1 en el eje x, definida como

$$\phi_{x1}(t) = \begin{cases} 0 & \forall t < 4 \\ 0.1 & \forall t \geq 4 \end{cases} \tag{4.18}$$

entonces, en la Figura 4.24 se muestran la respuesta del consenso líder-seguidor por control clásico h, ante una perturbación; en la Figura 4.25, por planitud diferencial. La recuperación del agente líder es más rápida en el control clásico h, debido a que una vez perturbado el agente se realiza una tarea de regulación, donde este controlador es más rápido. También, se

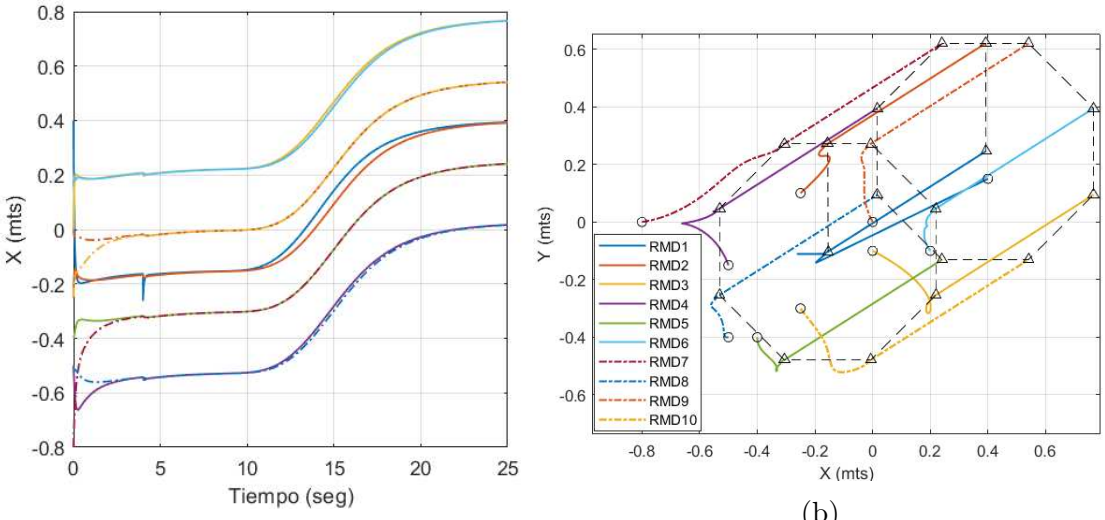
nota cómo los agentes de la Figura 4.24a perciben la perturbación del agente líder y responden con un sobreimpulso en sentido a la perturbación. En la Figura 4.25a el sobreimpulso de los seguidores es menos pronunciado. Con las pruebas anteriores y considerando que el consenso se calcula de forma recurrente, se hacen las siguiente observaciones:

1.- Los sobreimpulsos de los agentes son debidos a que la perturbación es interpretada como una nueva *condición inicial de consenso* (CIC) que se activa en $t = 4$ segundos. Esta CIC será estado de los agentes en el instante de perturbación. Por lo tanto, lo que se observa no es más que un ajuste del centro de formación debido a la negociación de la nueva CIC. Después de este ajuste, el agente líder se regula hacia la referencia de posición y guía a la formación hacia su posición deseada. Con base en lo anterior se plantea la siguiente suposición:

   Suposición 2 *Sea $\phi_i(t)$ una perturbación de posición de la forma* (4.18) *sobre el i-ésimo agente de un SMA bajo consenso de formación, entonces, el centro de la formación cambia bajo las ecuaciones 4.10 considerando la nueva CIC y los desfases de posición.*

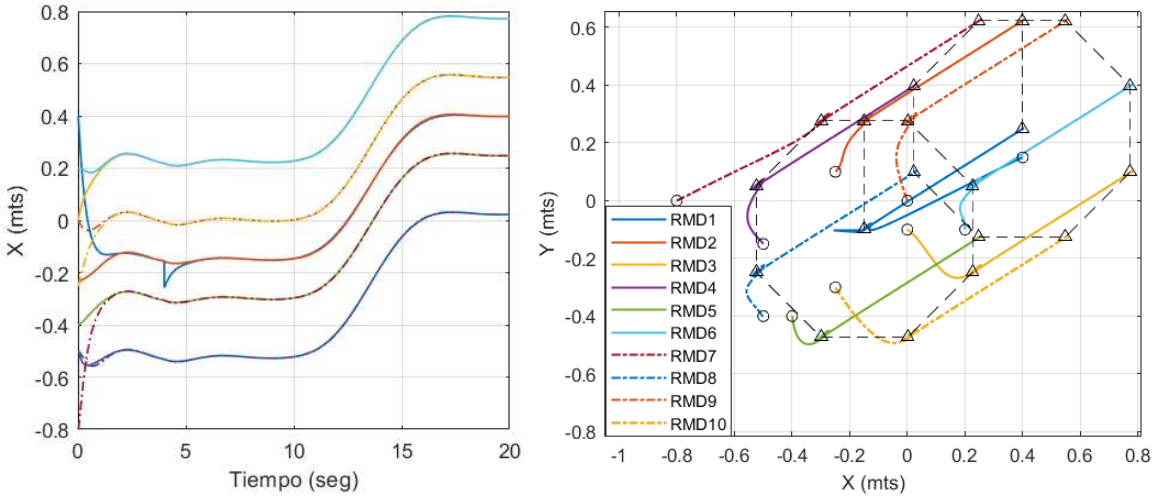
   Naturalmente, perturbaciones cuyos comportamientos sean más complejos que la ecuación (4.18), conllevan al ajuste del consenso de formación dependiendo de la forma de la perturbación $\phi(t)$.

2.- Es posible considerar una perturbación como un impulso de duración y amplitud acotada, así los cambios de posición provocados por la perturbación serán negociados por el SMA cuando ésta ocurra. Cualquier perturbación que reciba alguno de los agentes,



(a) (b)

Figura 4.24: Perturbación del SMA con control clásico h: a) eje x. b) posición en el plano.

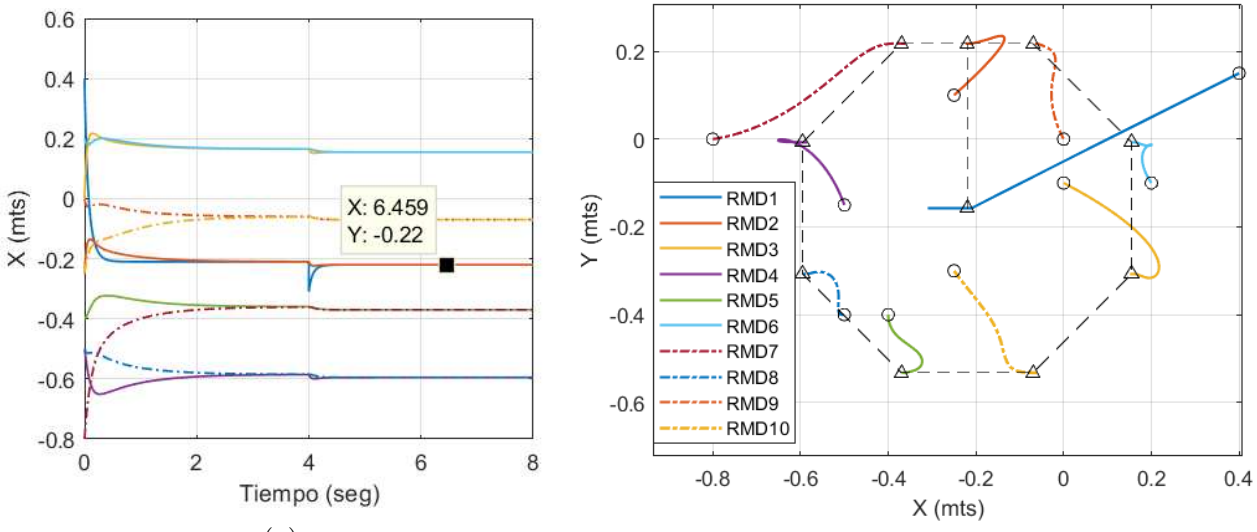


(a) (b)

Figura 4.25: Perturbación del SMA con planitud diferencial: a) eje x. b) Posición en el plano.

se interpreta como una CIC. En algún instante de tiempo la nueva información de posición (CIC) será transmitida y el consenso de formación se verá alterado.

3.- En consecuencia del punto anterior, si se considera a l_x y l_y como CIC realizadas en $t = 0$ para cada uno de los agentes, entonces, es posible negociar los desfases en distintos instantes de tiempo y lograr una así llamada ***formación cambiante***.



(a) (b)

Figura 4.26: Perturbación del consenso de formación bajo control clásico en h: a) eje x. b) posición en el plano.

4.- Alternativamente, es posible diseñar estos desfases de forma suave como la ecuación (3.30) para controlar la trayectoria de la formación.

En sentido de dar sustento al punto dos y la suposición 2, se presenta la respuesta de sólo el consenso de formación ante una perturbación de la forma (4.18) y con amplitud de 0,2 (ver Figura 4.26), es decir, se omite el esquema líder seguidor que regula hacia la posición deseada la formación. En el instante en el que la perturbación es inyectada, la CIC en el eje x es $X(t = 4) = [-0.31, -0.21, 0.165, -0.585, -0.36,\ 0.165, -0.36, -0.585, -0.06, -0.06]$, por lo tanto, utilizando la ecuación (4.10) y dado que los desfases son los mismos, $\bar{x} = -0.22$. Se demuestra así la suposición 2.

Capítulo 5

Implementación en ROS-Gazebo

Como consideración previa al inicio de este capítulo, se recomienda ampliamente al lector consultar los tutoriales disponibles en https://wiki.ros.org/ROS/Tutorials y en http://wiki.ros.org/sw_urdf_exporter/Tutorials/Export%20an%20Assembly, que contienen la información necesaria para comprender los conceptos básicos ROS y proceso detallado de exportación a formato URDF detallado. Esta información es vital para comprender sin dificultades la implementación del consenso en ROS-Gazebo. La distribución de Linux utilizada es Ubuntu 20 y la versión de ROS es noetic.

5.1. Exportación del modelo CAD

Como se mencionó en la última sección del capítulo dos, SolidWorks® posee un *pluging* capaz de exportar un diseño CAD a formato URDF y así importarlo en ROS-Gazebo para su visualización. Naturalmente, es posible describir el diseño en URDF desde ROS, sin embargo, las herramientas de diseño del entorno son limitadas.

Antes de exportar el diseño, el origen o cero del robot debe coincidir con el del ensamble y la orientación del ensamble debe ser la deseada, de otra manera el robot se invocará con desfases de posición y ángulo respecto del origen. Para ésto se recomienda crear un eje de coordenadas en el punto cero del robot y después hacerlo coincidente con el origen del ensamble. En esencia el *pluging* devuelve un paquete donde se encuentran dos archivos importantes (además de *packahge.xml* y *CMakeLists.txt*): un archivo de descripción URDF y un archivo de lanzamiento a ejecución. A continuación se describe a grandes rasgos la exportación del modelo CAD:

1. Descargar e instalar el *pluging* disponible en http://wiki.ros.org/sw_urdf_exporter.
2. Se ejecuta el pluging y seleccionan los cuerpos (links), tipo de par cinemático (joint) y sistema coordenado de referencia (origen del robot). La base del robot es la carcasa y se

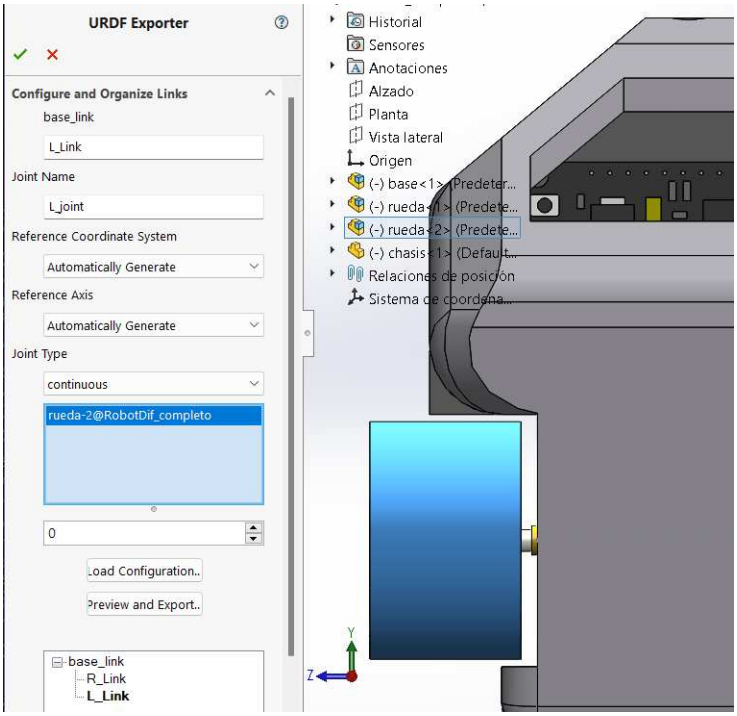


Figura 5.1: Configuración de la llanta izquierda para exportación URDF.

llama *base_link*; las llantas, son dos links hijos que se extienden de la base , son llamados *R_link* y *L_link*. Los joint seleccionados son de tipo continuo, *R_joint* y *L_joint*. Una vez configurado el robot se selecciona la opción *Preview and Export*. En la Figura 5.1 se muestra una captura de la configuración de una de las llantas del RMD.

3. Se corroboran los datos de cada cuerpo o link y se exporta el diseño seleccionando la opción *Export URDF and Meshes*.

4. El resultado de este proceso es una carpeta con el contenido mostrados en la Figura 5.2 donde se aprecian cinco carpetas y tres archivos. En la carpeta llamada urdf, se encuentra el archivo que describe nuestro modelo CAD. En la carpeta *meshes* se encuentran los archivos STL de la carcasa y llantas. En la carpeta *launch* se encuentra el archivo de ejecución del nodo que permite visualizar el robot. Las carpetas restantes no requieren de más atención. El código URDF del RMD se presenta en el Apéndice B.

5. En Linux, se crea un espacio de trabajo (carpeta) que llamaremos *SMA*, dentro de ésta se crea una carpeta con nombre *src* y se ejecuta el comando **$ catkin_make**, como se muestra en las siguientes lineas de código

```
$ mkdir -p ~/SMA/src
```

Figura 5.2: Carpetas de paquete exportado.

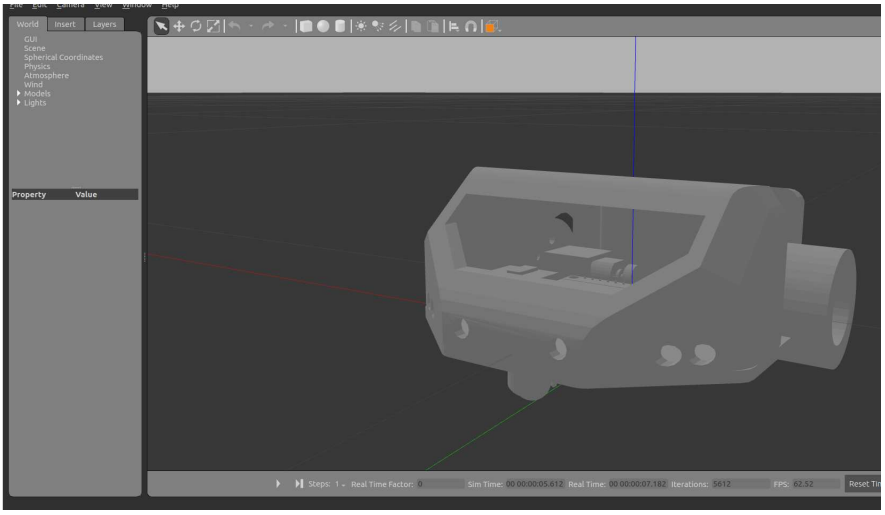


Figura 5.3: RMD en entorno de simulación Gazebo.

```
$ cd ~/SMA/
$ catkin_make
```

Por último, para visualizar el RMD en Gazebo (ver Figura 5.3), se pegan las carpetas obtenidas de la exportación en la carpeta *scr* y se ejecutan los siguientes comandos

```
$ cd ~/SMA/
$ catkin_make
$ source devel/setup.bash
$ cd ~/SMA/src/RMD/launch/
$ roslaunch gazebo.launch
```

El codigo URDF es convertido a formato xacro. En este nuevo código se crean macros de la matriz de inercia para las llantas reduciendo la extensión del código un 8.6 %. EL archivo xacro se muestra en el apéndice B2.

5.2. Programación de nodos

El grafo de la Figura 4.1 posee en total diez agentes, lo que en ROS sería equivalente a programar diez nodos. De ahora en adelante adelante a un agente del SMA se le conocerá como nodo. Los enlaces de comunicación se implementan con suscripciones y publicaciones a tópicos. Cada tópico es publicado por el correspondiente nodo y suscrito por sus nodos vecinos.

Como paso previo a la programación de los nodos, se modifican los archivos package.xml y CMakelist.txt del paquete RMD (ver Figura 5.2), agregando paquetes y dependencias necesarias como *geometry_msgs, roscpp, rviz* y *tf*. El archivo package.xml se reemplaza con el siguiente código:

```
<package format="2">
  <name>RMD</name>
  <version>1.0.0</version>
  <author>TODO</author>
  <maintainer email="TODO@email.com" />
  <license>BSD</license>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>geometry_msgs</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>tf</build_depend>
  <build_depend>urdf</build_depend>
  <build_depend>xacro</build_depend>
  <build_export_depend>geometry_msgs</build_export_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>tf</build_export_depend>
  <build_export_depend>urdf</build_export_depend>
  <build_export_depend>xacro</build_export_depend>
  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>tf</exec_depend>
  <exec_depend>urdf</exec_depend>
  <exec_depend>xacro</exec_depend>
  <depend>roslaunch</depend>
  <depend>robot_state_publisher</depend>
  <depend>joint_state_publisher_gui</depend>
```

```
  <depend>gazebo</depend>
  <export>
    <architecture_independent />
  </export>
</package>
```

el archivo CMakelist.txt, con el siguiente:

```
cmake_minimum_required(VERSION 3.0.2)
project(RMD)
find_package(catkin REQUIRED COMPONENTS
  geometry_msgs
  roscpp
  tf
  urdf
  xacro)
catkin_package()
find_package(roslaunch)
foreach(dir config launch meshes urdf)
install(DIRECTORY ${dir}/
DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}/${dir})
endforeach(dir)
```

Para que estas modificaciones se consoliden, en una terminal abierta desde la carpeta SMA, ejecutamos el comando catkin_make.

5.2.1. Nodo agente

El i-ésimo nodo hace referencia a su propio archivo de descripción en formato xacro, es decir, diez archivos de descripción llamados RMDi.xacro ($i = 1, 2, \ldots, 10$) son creados en la carpeta urdf. Después, se agrega un plugin para controlar al RMD. A continuación, se muestra bloque de código del plugin para al primer archivo xacro (RMD1.xacro):

```
  <gazebo>
   <plugin name="differential1" filename="libgazebo_ros_diff_drive.so">
     <rosDebugLevel>Debug</rosDebugLevel>
     <publishWheelTF>false</publishWheelTF>
     <robotNamespace>/1</robotNamespace>
     <publishTf>1</publishTf>
     <publishWheelJointState>false</publishWheelJointState>
     <alwaysOn>true</alwaysOn>
     <updateRate>50</updateRate>
```

```
      <leftJoint>L_joint</leftJoint>
      <rightJoint>R_joint</rightJoint>
      <wheelSeparation>0.092</wheelSeparation>
      <wheelDiameter>0.032</wheelDiameter>
      <broadcastTF>1</broadcastTF>
      <wheelTorque>0.1</wheelTorque>
      <wheelAcceleration>1</wheelAcceleration>
      <commandTopic>cmd_vel1</commandTopic>
      <odometryFrame>odom1</odometryFrame>
      <odometryTopic>odom1</odometryTopic>
      <robotBaseFrame>base_link</robotBaseFrame>
    </plugin>
  </gazebo>
```

Naturalmente, se debe modificar el nombre del plugin, espacio de robot y tópicos para cada archivo xacro restante. Este plugin existe en el entorno de Gazebo, permite definir los tópicos donde se enviarán los comandos de control y de donde se obtendrá la posición y orientación de cada nodo RMD. Por un lado, el tópico *cmd_vel1* recibe la información de velocidad lineal y angular. Por otro, en el tópico *odom1* se publica la información de posición y orientación a un frecuencia de 50 Hz.

El archivo launch que lanza a ejecución los diez nodos (archivos RMDi.xacro) se guarda como *sma.launch* en la carpeta launch. El código se encuentra en el apéndice C. Este archivo inicializa a cada robot con las condiciones iniciales $X(0) = [3,3,3,3,3,3,2,1,0,-1]^T$ y $Y(0) = [0,1,2,3,4,5,5,5,5,5]^T$. Para lanzar a ejecución los diez nodos escribimos en una terminal las siguientes sentencias

```
$ cd ~/SMA/
$ source devel/setup.bash
$ cd ~/SMA/src/RMD/launch/
$ roslaunch sma.launch
```

En la Figura 5.4 se muestra una vista isométrica de los diez RMD distribuidos en un mundo vacío. Los robots poseen la misma orientación ($\theta_i = \pi/2$), sin embargo, es posible cambiar ésta configurando las condiciones iniciales desde el archivo de ejecución utilizando el comando **$ rostopic list** se observan los tópicos activos durante la ejecución de Gazebo (ver Figura 5.5). Cada nodo RMD posee dos tópicos: cmd_vel y odom. En este punto solo existen nodos independientes en el mismo entorno de simulación, en la siguiente sección se programan las suscripciones y publicaciones a los tópicos para intercomunicar los nodos de acuerdo al grafo propuesto.

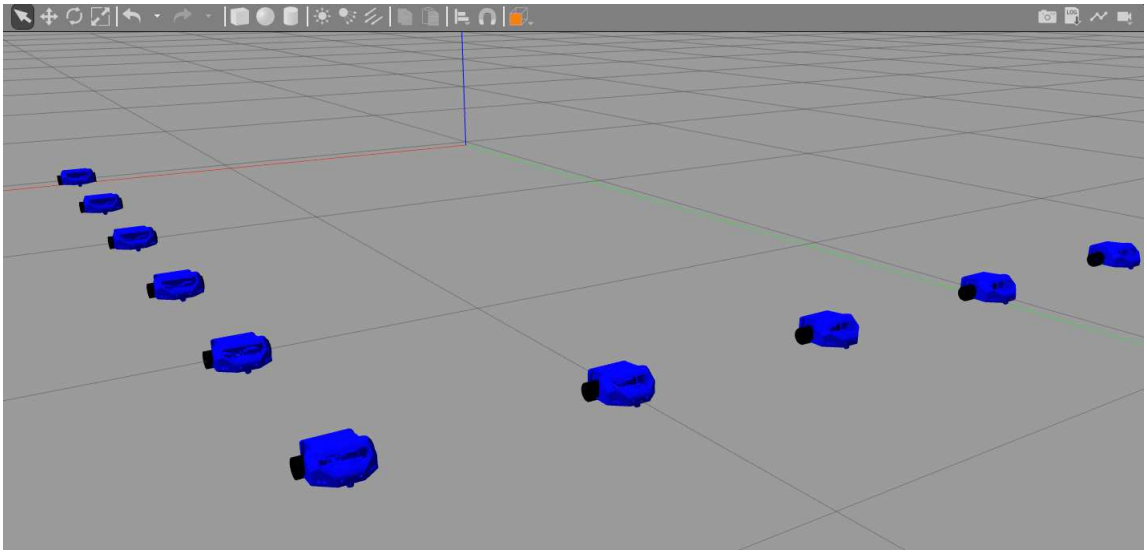


Figura 5.4: Diez nodos RMD lanzados a ejecucioń en Gazebo

```
exon@exon-pc:~/SMA/src/RMD/launch$ rostopic list
/1/cmd_vel1
/1/odom1
/10/cmd_vel10
/10/odom10
/2/cmd_vel2
/2/odom2
/3/cmd_vel3
/3/odom3
/4/cmd_vel4
/4/odom4
/5/cmd_vel5
/5/odom5
/6/cmd_vel6
/6/odom6
/7/cmd_vel7
/7/odom7
/8/cmd_vel8
/8/odom8
/9/cmd_vel9
/9/odom9
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
```

Figura 5.5: Tópicos de los diez nodos

5.2.2. Nodo de control

Para lanzar a ejecución las leyes de control propuestas en el capitulo 4, es necesario programarlas en un archivo. El lenguaje de programación seleccionado es Python. Dentro de la carpeta RMD, se crea una carpeta llamada *scrips* en donde se guardan los diez archivos de control clásico en h y diez archivos de control por planitud diferencial. Las leyes de control clásico h (4.2) son llamados controlhi.py (con $i = 1, 2, ..., 10$); las de control por planitud diferencial (4.3), controlci.py (con $i = 1, 2..., 10$). Por un lado, el acceso de información se realiza suscribiendo a cada agente al respectivo tópico de posición del cual require información, por ejemplo, el nodo de control 1 se suscribe a los tópicos /1/odom1, /3/odom2, /3/odom/3 y /4/odom4 (ver Figura 4.13 para ver el esquema de comunicación). Lo anterior se realiza con las siguientes sentencias

```
rospy.Subscriber("/1/odom1", Odometry, callback1)
rospy.Subscriber("/2/odom2", Odometry, callback2)
rospy.Subscriber("/3/odom3", Odometry, callback3)
rospy.Subscriber("/4/odom4", Odometry, callback4)
```

Cada vez que un tópico de posición actualiza su valor, las funciones *callback* son llamadas y obtienen la información del tópico almacenándola en la variable local *Odometry*. La posición del robot se obtiene con las sentencias

```
x = data.pose.pose.position.x
y = data.pose.pose.position.y
```

Debe tomarse en consideración que las posiciones obtenidas son con respecto al sistema coordenado con el que se exporta el robot. En este caso el origen es el punto de operación c. Para calcular el punto de operación h se suman las componentes $hcos(\theta)$ y $hsin(\theta)$ respectivamente. Un punto importante es que la orientación de cada RMD se obtiene en forma de cuaternión, por lo tanto la orientación θ se calcula como

```
c1  = data.pose.pose.orientation.z
c2  = data.pose.pose.orientation.w
th = 2*math.atan2(c1,c2)
    if th <0:
        th = 2*math.pi + th
    th = th + math.pi/2
```

donde t_h devuelve un valor en el rango 0 a 2π radianes. Por otro lado, cada i-ésimo nodo de control debe publicar en el i-ésimo tópico de velocidad, por ejemplo, para actualizar las entradas de control del tópico /1/cmd_vel1, se configura al nodo de control 1 como publicador utilizando la siguiente sentencia

```
pub = rospy.Publisher('/1/cmd_vel1', Twist, queue_size=5)
```

Una vez que se obtiene información de los agentes vecinos se procede a calcular las leyes de control. A continuación se muestran las funciones que se encargan de calcular las leyes de control:

a) Control clásico h

```
def control():
    global V,w
    r1=-k1*(x1h-x2h)-k1*(x1h-x3h)-k1*(x1h-x4h)
    r2=-k1*(y1h-y2h)-k1*(y1h-y3h)-k1*(y1h-y4h)
    V = r1*math.cos(th1) + r2*math.sin(th1)
    w = -r1*math.sin(th1)/h + r2*math.cos(th1)/h
```

b) Planitud diferencial

```
def control():
    global V1,V2,x1c2,y1c2,x2c2,y2c2,x3c2,y3c2,x4c2,y4c2,w
    # calculo de velocidades
    dx1c = (x1c1 - x1c2)/delta
    dy1c = (y1c1 - y1c2)/delta
    dx2c = (x2c1 - x2c2)/delta
    dy2c = (y2c1 - y2c2)/delta
    dx3c = (x3c1 - x3c2)/delta
    dy3c = (y3c1 - y3c2)/delta
    dx4c = (x4c1 - x4c2)/delta
    dy4c = (y4c1 - y4c2)/delta

    # calculo de desfases
    x1=x1c1-lx1
    x2=x2c1-lx2
    x3=x3c1-lx3
    x4=x4c1-lx4
    y1=y1c1-ly1
    y2=y2c1-ly2
    y3=y3c1-ly3
    y4=y4c1-ly4

    # calculo de control auxiliar
    r11 = -k0*(x1-1)-k1*(dx1c-0)-k0*(x1-x2)-k1*(dx1c-dx2c)
    r1 = r11 -k0*(x1-x3)-k1*(dx1c-dx3c)-k0*(x1-x4)-k1*(dx1c-dx4c)
    # calculo de control auxiliar
    r22 = -k0*(y1-1)-k1*(dy1c-0)-k0*(y1-y2)-k1*(dy1c-dy2c)
    r2 = r22-k0*(y1-y3)-k1*(dy1c-dy3c)-k0*(y1-y4)-k1*(dy1c-dy4c)
```

```
# calculo de velocidades
V1 = (r1*math.cos(th1) + r2*math.sin(th1))*delta + V2
w = -r1*math.sin(th1)/V1 + r2*math.cos(th1)/V1
# actualizar valor anterior de velocidad lineal
V2=V1
```

El cálculo de las derivadas para la ley de control por planitud diferencial se implementa con la aproximación discreta

$$\dot{x}[k] = \frac{x[k] - x[k-1]}{\Delta t} \tag{5.1}$$

La frecuencia de ejecución de los nodos de control se configura a 100 Hz. En el apéndice D1 y D2 se muestra el archivo python completo del primer nodo de control clásico h y por planitud diferencial respectivamente. Los archivos para lanzamiento a ejecución de los nodos de control se encuentran en los apéndices D3 Y D4, en éstos a cada nodo se envía información de los desfases, frecuencia de ejecución y valor de las ganancias. El proyecto completo de ROS-Gazebo se puede consultar en el repositorio de GitHub https://github.com/ExonVillalobos/SMA.git.

5.3. Simulaciones

Para ejecutar esta simulación,

- Primero, se lanza a ejecución el SMA

  ```
  $ cd ~/SMA/
  $ source devel/setup.bash
  $ roslaunch RMD sma.launch
  ```

  Las condiciones iniciales de cada agente son $X(0) = [1.6, 2.4, 0.8, 0.8, 2.4, 2.4, 1.6, 0.8, 0.8, 2.4]$ y $Y(0) = [0, 1.6, 2.4, 0, 0.8, 2.4, 2.4, 1.6, 0.8, 0]$. Estas condiciones iniciales son arbitrarias y pueden modificarse.

- Segundo, en otra terminal se lanzan a ejecución los controladores requeridos

  ```
  $ cd ~/SMA/
  $ source devel/setup.bash
  $ roslaunch RMD controlh.launch
  ```

  En una nueva terminal se utiliza el comando **$rqt _graph** que permite observar los nodos y tópicos lanzados a ejecución. El grafo que resulta se muestra en la Figura 5.6.

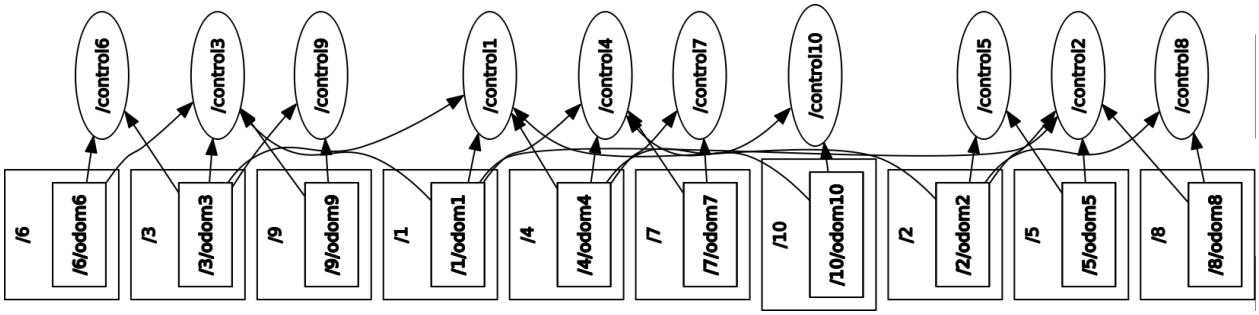


Figura 5.6: Grafo del SMA implementado en ROS-Gazebo

En éste tanto el espacio de trabajo que alberga los tópicos */i* y el nodo de control */controli* ($i = 1, 2, ..., 10$), conforman a un agente. Observe que cada agente se comunica en base al grafo propuesto (ver Figura 4.1). Es importante mencionar que se omitieron otros tópicos (*/i/cmd_veli*) que dificultan la comprensión a simple vista del grafo.

En los archivos de ejecución (*controlc.launch* y *controlh.launch*) los desfases de posición son $l_x = [0, 1.2, -0.4, -0.4, 1.2, 0.8, 0.4, -1.2, -1.2, 0.4]$ y $l_y = [0, 0.4, 1.2, -1.2, -0{,}4, 0.8, 1.2, 0.4, -0.4, -1.2]$. Estos desfases formaran una figura octagonal. Se establecen de esta forma para evitar colisiones, pues al unir con una línea posiciones iniciales y desfases ninguna se intersecta.

- Tercero, utilizando archivos de texto se almacenan el tiempo y las posiciones de los agentes. El agente líder almacena su propia información y la de los agentes 2, 3 y 4 (*ah1234.txt* y *ac1234.txt*); el agente 2, almacena la de los agente 5 y 8 (*ah258.txt* y *ac258.txt*); el agente 3, almacena la de los agente 6 y 9 (*ah369.txt* y *ac369.txt*); el agente 2, almacena la de los agente 7 y 10 (*ah4710.txt* y *ac4710.txt*).

- Cuarto y último, los archivos con la información de posición del SMA son exportados y graficados en Matlab®.

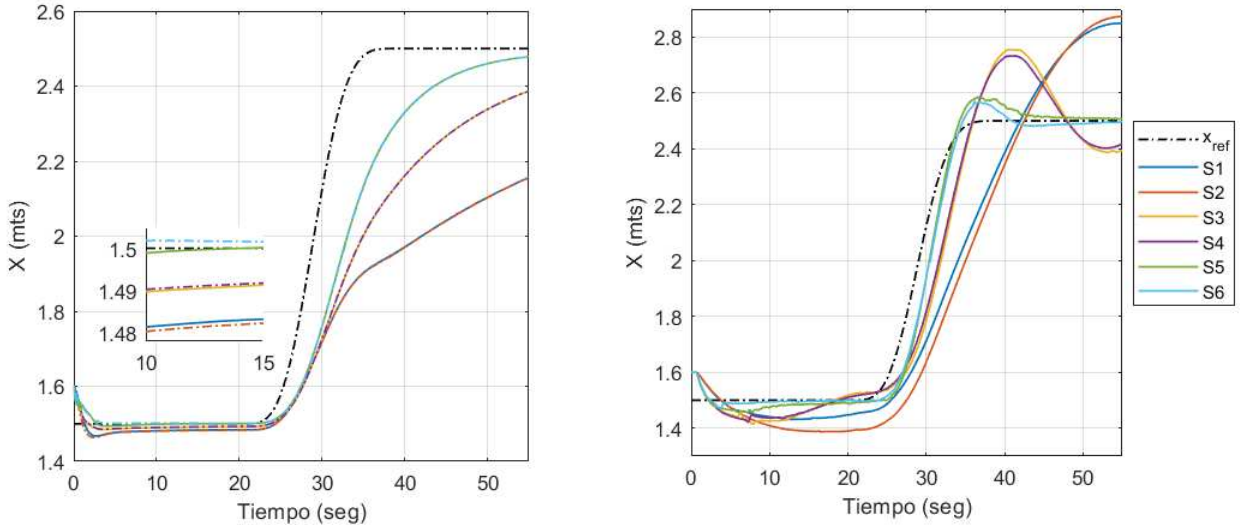
El experimento consiste en que el SMA realize una formación cuyo centro esté descrito por una trayectoria *Bézier* (ecuación (3.30)). Los parámetros de la trayectoria son: $x_{ini} = 1{,}5, y_{ini} = 1, x_{fin} = 2{,}5, y_{fin} = 2, t_{ini} = 20$ y $t_{fin} = 40$. El tiempo total de simulación es de 55 segundos. Se realizan seis simulaciones variando ganancias y frecuencia de ejecución F_s. Los valores asignados se muestran en la Tabla 5.1. En el consenso bajo esquema líder seguidor por acción del control clásico h las ganancias proporcionales son $k_p = \lambda_p = w_n$. En el consenso bajo esquema líder seguidor por acción del control clásico h las ganancias son $\lambda_1 = \gamma_1 = 2\zeta w_n$ y $\lambda_0 = \gamma_0 = w_n^2$.

Tabla 5.1: Parámetros de control por simulación.

Parámetro	Simulación					
	S1	S2	S3	S4	S5	S6
w_n	0.4	0.4	1	1	2	2
ζ	1	1	1	1	1	1
F_s	200	400	200	400	200	400

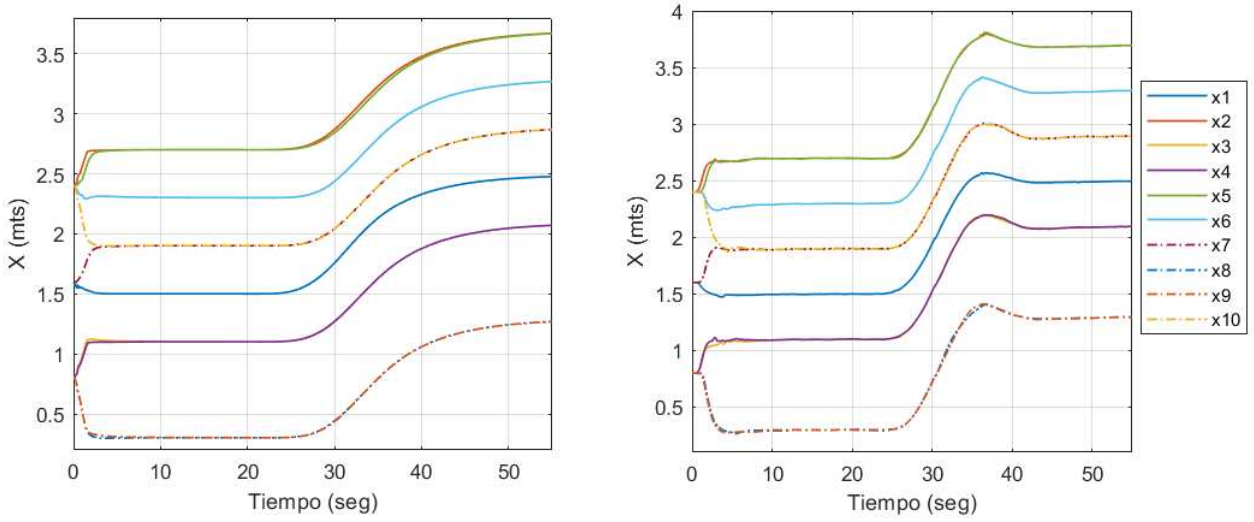
En la Figura 5.7 se observan las respuestas de posición en el eje x de los agentes líderes (RMD_1) en cada simulación realizada. Se observa una mejora conforme se aumenta F_s y w_n. Durante la etapa de formación y regulación ($0 < t \leq 20$), se establece como límite $w_n = 2$, puesto que se observa que los RMD se levantan sobre sus ruedas. Este comportamiento realista es debido a la ganancia relativamente alta, se puede mejorar diseñando una trayectoria suave desde cada posición inicial hacia el valor deseado de desfase.

En la Figura 5.8 se observan las posiciones de los agentes en el eje x. Estas posiciones son obtenidas de la simulación S6. El control por planitud diferencial (Figura 5.8b), se apega mejor al polinomio deseado, pero presenta un sobre impulso de 6,9 % al final del seguimiento. En el control clásico h (Figura 5.8a), no se presenta oscilaciones en la formación y regulación, pero si un retraso considerable al seguir la trayectoria. En la Figura 5.9 se observan las posiciones de los agentes en el plano contra el tiempo. Estos resultados concuerdan con los



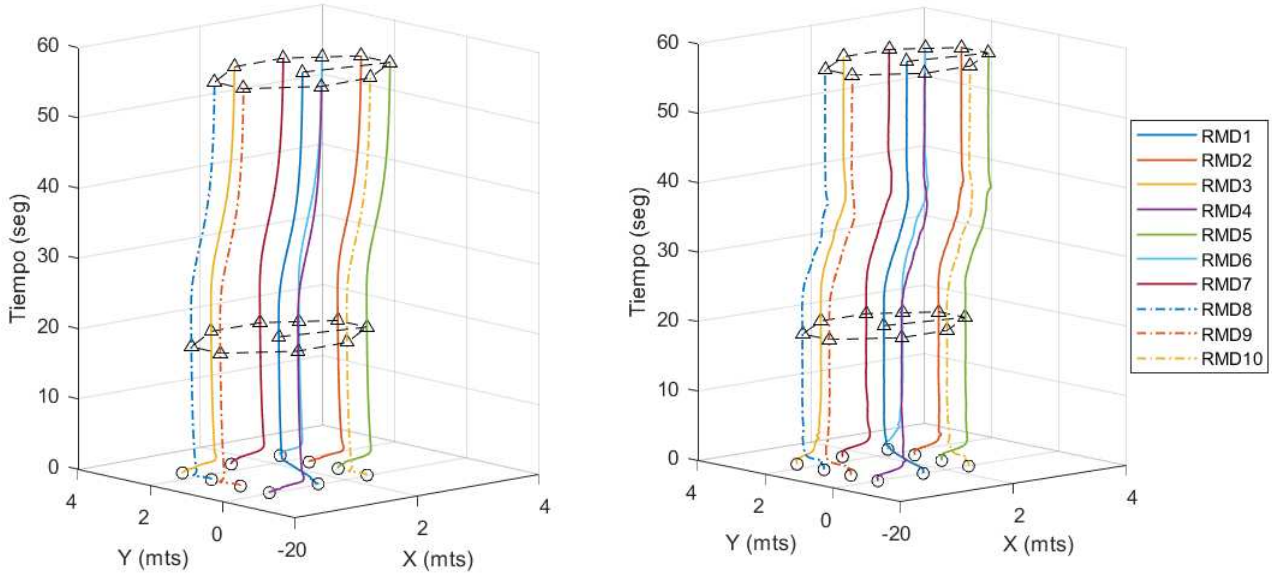
(a) (b)

Figura 5.7: Posición en el eje x del agente líder en cada simulación: a) control clásico h. b) planitud diferencial.



(a) (b)

Figura 5.8: Posición en el eje x del agente líder simulación S6: a) control clásico h. b) planitud diferencial.



(a) (b)

Figura 5.9: Posición del SMA contra tiempo en simulación S6: a) planitud diferencial. b) control clásico h.

obtenidos en el capitulo 4.

5.3.1. Análisis de resultados

Para evaluar el desempeño de ambos controladores, se dividirá cada simulación en dos etapas: 1) Regulación-formación ($0 < t \leq 20$) y 2) seguimiento-establecimiento ($20 < t \leq 55$). Ambas etapas se evaluaran con los indicadores ISE e IAE.

En la Tabla 5.2 se muestran las componentes de los indicadores en la etapa de regulación y formación. En la Tabla 5.3 se muestran las magnitudes de los indicadores en esta misma etapa. En el control clásico h la magnitud del indicador ISE con es inferior al de planitud diferencial en cada simulación. Evidenciando su superioridad en esta etapa en base a este indicador. Sin embargo, se observa que la tendencia decreciente de ambos indicadores en control clásico h se interrumpe cuando $w_n = 2$, al contrario de planitud diferencial. Lo anterior indica una tolerancia a valores relativamente altos de ganancia por parte del control por planitud diferencial.

Tabla 5.2: Componentes de indicadores en regulación y formación ($0 < t \leq 20$).

Simulación	Control clásico h				Planitud diferencial			
	ISE_X	ISE_Y	IAE_X	IAE_Y	ISE_X	ISE_Y	IAE_X	IAE_Y
S1	1.759	4.048	11.24	22.86	6.676	6.904	29.5	26.58
S2	1.668	4.084	11.04	23.57	7.481	7.355	34.05	28.83
S3	1.105	1.892	6.882	6.978	3.417	2.996	18.17	9.544
S4	1.095	1.915	6.783	6.55	3.12	3.083	17.72	10.15
S5	0.8829	3.302	4.213	17.44	2.263	2.888	11.28	9.988
S6	0.8972	2.799	3.864	15.32	2.001	3.023	7.489	10.49

Tabla 5.3: Magnitud de indicadores en regulación y formación ($0 < t \leq 20$).

Simulación	Control clásico h		Planitud diferencial	
	ISE	IAE	ISE	IAE
S1	4.414	25.474	9.604	39.708
S2	4.411	26.027	10.491	44.616
S3	2.191	9.801	4.544	20.524
S4	2.206	9.429	4.386	20.421
S5	3.418	17.942	3.669	15.066
S6	2.939	15.800	3.625	12.889

En la Tabla 5.4 se muestran las componentes de los indicadores en la etapa de seguimiento y establecimiento. En la Tabla 5.5 se muestran las magnitudes de los indicadores en esta misma etapa. En ambos indicadores el control por planitud es superior al presentar valores inferiores. Por lo tanto, en tareas de seguimiento el control por planitud es óptimo en cuanto a error.

Tabla 5.4: Componentes de indicadores en seguimiento y establecimiento ($20 < t \leq 55$).

Simulación	Control clásico h				Planitud diferencial			
	ISE_X	ISE_Y	IAE_X	IAE_Y	ISE_X	ISE_Y	IAE_X	IAE_Y
S1	113.441	93.892	178.26	161.74	46.104	45.196	108.6	101.82
S2	114.132	92.826	178.86	160.93	59.989	47.725	126.35	103.87
S3	53.295	51.888	115.818	113.722	17.713	16.394	64.27	61.676
S4	53.065	52.255	115.617	114.15	9.12	16.607	57.08	62.39
S5	25.1271	25.438	70.077	71.63	3.129	2.985	22.88	22.552
S6	34.9328	25.201	69.856	71.19	2.686	2.869	20.281	21.28

Tabla 5.5: Magnitud de indicadores en seguimiento y establecimiento ($20 < t \leq 55$).

Simulación	Control clásico h		Planitud diferencial	
	ISE	IAE	ISE	IAE
S1	147.257	240.700	64.562	148.867
S2	147.115	240.602	76.657	163.564
S3	74.382	162.316	24.135	89.076
S4	74.475	162.473	18.946	84.561
S5	35.756	100.208	4.324	32.126
S6	43.074	99.739	3.930	29.397

5.3.2. Formación de figuras geométricas

Para finalizar este capítulo, se realiza una simulación del SMA siguiendo una trayectoria Leminiscata y realizando diversas formaciones de figuras geométricas. El control implementado es planitud diferencial. De la sección anterior se selecciona las mismas condiciones iniciales y los parámetros de control $w_n = 2$, $\zeta = 1$ y $F_s = 400$. Las figuras geométricas a realizar son: octágono, triángulo y hexágono. Los desfases para realizar esta simulación se muestran en la Tabla 5.6. Estos desfases son transmitidos a todos los agentes y luego seleccionados por cada agente.

La primer formación (octágono) se realiza en ($0 < t \leq 35$). La segunda formación (triángulo) se realiza en ($35 < t \leq 70$). La tercera formación se realiza en $70 < t \leq 95$. La trayectoria

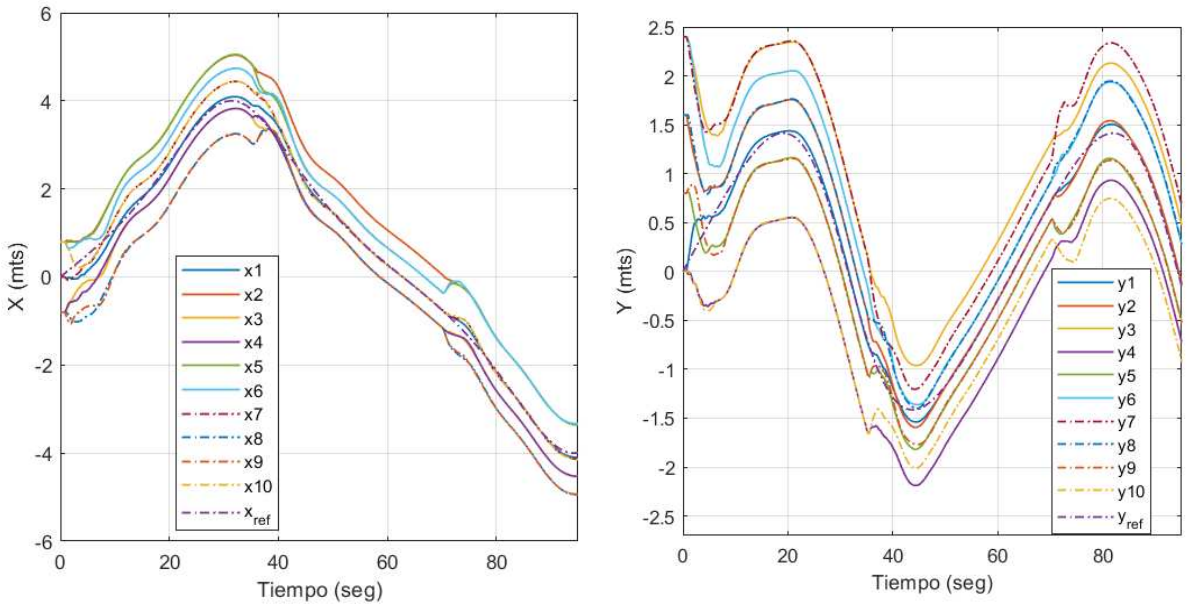
Tabla 5.6: Desfases en cada agente para la formación de figuras en el plano.

Figuras geométricas	Desfases
Octágono	l_x =[0,0.9,-0.3,-0.3,0.9,0.6,0.3,-0.9,-0.9,0.3] l_y =[0,0.3,0.9,-0.9,-0.3,0.6,0.9,0.3,-0.3,-0.9]
Triangulo	l_x =[0,0.8,-0.4,-0.4,0.4,0.4,0,-0.4,-0.4,0] l_y =[0,0,0.6,-0.6,-0.2,0.2,0.4,0.2,-0.2,-0.4]
Hexágono	l_x =[0,0.8,-0.4,-0.4,0.8,0.8,0,-0.8,-0.8,0] l_y =[0,0,0.6,-0.6,-0.4,0.4,0.8,0.4,-0.4,-0.8]

deseada (leminiscata) comienza desde $t = 0$ hasta $t = 95$ segundos, es decir, cada formación se realiza mientras el agente líder sigue la trayectoria deseada. En la Figura 5.10 se observan las posiciones de los agentes en cada eje. En la figura 5.11 se observa el movimiento en el plano del SMA.

En la Figura 5.12 se observa la evolución de los formación contra tiempo. Note que el instante de inicio de cada formación se genera error de seguimiento en el agente líder. Esto se debe a que el término de control de formación genera un error superior al de seguimiento. Una vez que la formación se alcanza parcialmente, el error de seguimiento vuelve a tener jerarquía y el agente líder guía al SMA. Evitar este comportamiento natural del consenso líder-seguidor, implicaría (como se menciono anteriormente) diseñar desfases suaves en cada agente.

El vídeo de las simulaciones de formación y seguimiento se encuentra disponible en la siguiente url: https://youtu.be/wh8d_hAz3bM.



(a) (b)

Figura 5.10: Posición de los agentes en cada eje ante la formación y seguimiento de leminiscata: a) eje x. b) eje y.

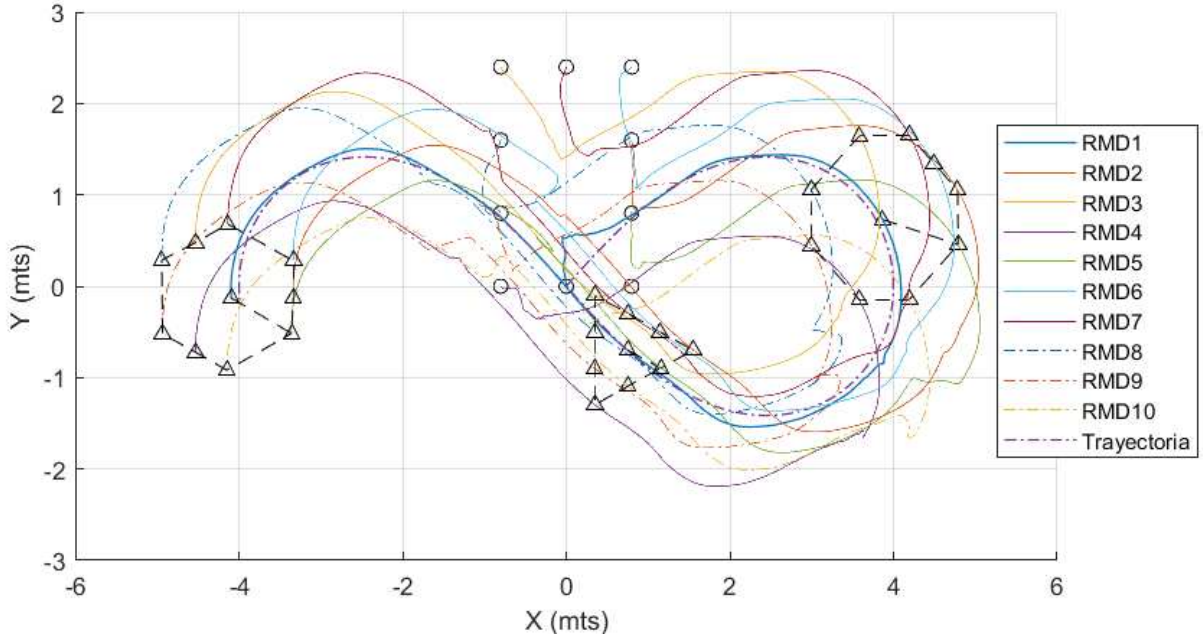


Figura 5.11: Posición en el plano de SMA ante la formación y seguimiento de leminiscata.

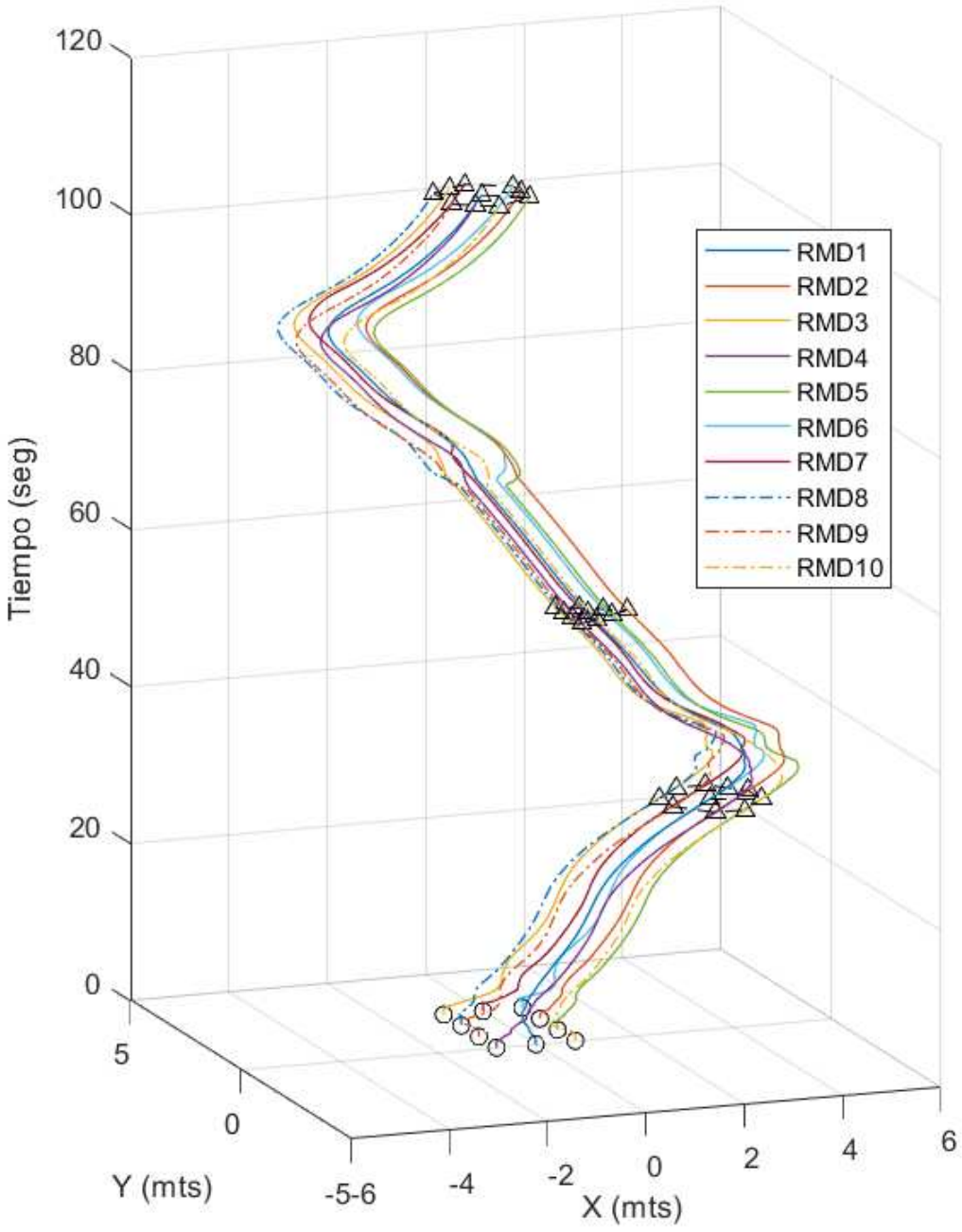


Figura 5.12: Formación y seguimiento de leminiscata contra tiempo.

Capítulo 6

Conclusiones

En este proyecto de investigación se extendieron las propiedades y capacidades de movilidad del RMD (menor consumo energético y orientación instantánea) en el diseño de un SMA compuesto de diez agentes para la formación y seguimiento de trayectorias. La implementación se realizó en un entorno capaz de diseñar un SMA mediante la programación de nodos independientes y su topología de comunicación: ROS-Gazebo.

Se estudiaron dos controladores cinemáticos en diferentes puntos de operación (en c y en h) y se propone un tercer controlador basado en realimentación dinámica o planitud diferencial. Los tres cumplen satisfactoriamente con la tarea de seguimiento y regulación, solo distinguiéndose al establecer la orientación deseada del RMD: el control por planitud diferencial es óptimo en cuanto a maniobrabilidad pues aprovecha la capacidad del RMD de girar sobre su propio eje.

Por un lado, el modelo cinemático en h se propuso plano por realimentación estática; por otro lado, el modelo en c es plano por realimentación dinámica. Estas técnicas optimizan el modelo cinemático y permitieron el diseño de leyes de control cooperativo a través del consenso promedio y la representación algebraica de los grafos.

Los controladores cooperativos se implementaron en dos entornos de simulación: Matlab-Simulink y ROS-Gazebo. Este último permitió visualizar el modelo 3D y emular la dinámica de los diez agentes además de poder considerar la frecuencia de comunicación y frecuencia de operación de los controladores. Con esto se observó el comportamiento ante la dinámica de los agentes, con distintas ganancias y frecuencias. Las simulaciones permitieron comprender los efectos de lo controladores, además de que se pueden tomar las consideraciones previas a una implementación experimental.

Las respuestas en ambos simuladores son las mismas en esencia, es decir, las técnicas de control cooperativo pueden llevar al SMA al estado de consenso aun sin considerar la dinámi-

ca en su diseño. El cambio de formaciones, o inclusive la formación desde las condiciones iniciales, requiere de un tiempo de establecimiento antes comenzar a seguir una trayectoria deseada. De otra forma el agente líder presentará error como si de una perturbación se tratase. En ambos entornos de simulación los resultados demuestran una superioridad del control clásico en h en las tareas de formación, mientas que el control por planitud diferencial es superior en la tarea de seguimiento y establecimiento.

6.1. Trabajo a futuro

Para evitar las perturbaciones durante la formación o cambio de formaciones, se puede diseñar todos los desfases de posición de forma suave (polinomio Bézier o un filtro).

Las colisiones entre los agentes se podrían evitar con: 1) una rutina de asignación minimizando la distancia entre el agente y el desfase deseado. 2) Diseñar un componente de control para evasión de obstáculos.

En caso de finalizar un nodo agente o de control, el resto de agentes vecinos almacenan el último valor de posición, esto puede perjudicar el seguimiento y formación. Se propone diseñar un función que inhiba la participación de este agente en caso de perder comunicación.

Implementar de forma experimental la ley de control por planitud diferencial en un RMD. Después, con un grupo de N RMD comprobar los comportamientos de las leyes de control cooperativo.

Bibliografía

[1] A. O. Baturone, *Robótica: manipuladores y robots móviles*. Marcombo, 2005.

[2] V. Borrero López, "Desarrollo de programas y experimentación de navegación de un robot móvil terrestre," 2020.

[3] J. L. Paniagua Jaramillo *et al.*, "Diseño e implementación de un sistema de control que permita integrarse con diferentes tipos de robot móviles terrestres," B.S. thesis, Universidad Autónoma de Occidente, 2014.

[4] A. Maity, S. Mandal, S. Mazumder, and S. Ghosh, "Serpentine robot: An overview of current status & prospect," in *14th National conference on machines and mechanisms*, 2009, p. 275.

[5] J. C. Alexander and J. H. Maddocks, "On the kinematics of wheeled mobile robots," *The International Journal of Robotics Research*, vol. 8, no. 5, pp. 15–27, 1989.

[6] P. F. Muir, *Modeling and control of wheeled mobile robots*. Citeseer, 1988.

[7] E. Calle, I. Ávila, and J. Zambrano, "Diseño e implementación de un robot móvil cuadrúpedo," *Revista Tecnológica-ESPOL*, vol. 20, no. 1, 2007.

[8] D. J. Todd, *Walking machines: an introduction to legged robots*. Springer Science & Business Media, 2013.

[9] V. R. B. Sotelo, J. R. G. Sánchez, and R. S. Ortigoza, "Robots móviles: Evolución y estado del arte," *Polibits*, no. 35, pp. 12–17, 2007.

[10] R. S. Ortigoza, E. R. Ramos-Silvestre, and R. Morales-Guerrero, "Modelado, simulación y construcción de un robot móvil de ruedas tipo diferencial," *Latin-American Journal of Physics Education*, vol. 4, no. 3, p. 39, 2010.

[11] M. G. Bekker, "Introduction to terrain-vehicle systems. part i: The terrain. part ii: The vehicle," MICHIGAN UNIV ANN ARBOR, Tech. Rep., 1969.

[12] G. Fernández, L. H. R. González, and M. B. López, "Implementación de un controlador de posición y movimiento de un robot móvil diferencial," *Tecnura: Tecnología y Cultura Afirmando el Conocimiento*, vol. 20, no. 48, pp. 123–136, 2016.

[13] P. Niño-Suárez, E. Aranda-Bricaire, and M. Velasco-Villa, "Control mediante modos deslizantes en tiempo discreto para el seguimiento de trayectorias de un robot móvil," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 4, no. 4, pp. 31–38, 2007.

[14] C. O. C. Aviles, A. M. Díaz, and A. R. Angeles, "Control de sincronización de un robot móvil diferencial," 2010.

[15] M. A. M. Vilchis, E. A. P. Flores, M. M. Melchor, V. M. H. Guzmán, and R. S. Ortigoza, "Modelado y control de un robot móvil tipo newt en la tarea de seguimiento de trayectoria," *Télématique*, vol. 7, no. 2, pp. 129–145, 2008.

[16] R. Silva-Ortigoza, G. Silva-Ortigoza, V. M. Hernández-Guzmán, V. R. Barrientos-Sotelo, J. M. Albarrán-Jiménez, and V. M. Silva-Garcia, "Trajectory tracking in a mobile robot without using velocity measurements for control of wheels," *IEEE Latin America Transactions*, vol. 6, no. 7, pp. 598–607, 2008.

[17] M. A. Zimmer, "Formaciǿn de aves migratorias," https://pixabay.com/es/photos/formaci%C3%B3n-pajaros-migratorios-gansos-508038/, 2014.

[18] P. Worcestershire, "Abejas de miel," https://pixabay.com/es/photos/abejas-miel-abejas-de-miel-panal-345628/, 2014.

[19] A. Villalonga Jaén, F. Castaño Romero, R. Haber, G. Beruvides, and J. Arenas, "El control de sistemas ciberfísicos industriales: revisión y primera aproximación," in *XXXIX Jornadas de Automática*. Área de Ingeniería de Sistemas y Automática, Universidad de Extremadura, 2018, pp. 916–923.

[20] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158–168, 2016.

[21] J. R. C. Fernández and V. J. P. Pérez, "Entornos de sistemas multiagente y ciberfisicos en la ciberdefensa," *Revista SIC: ciberseguridad, seguridad de la información y privacidad*, vol. 111, pp. 100–102, 2014.

[22] V. J. B. Navarro and A. G. Boggino, "Aplicaciones industriales de los sistemas multiagente," 2003.

[23] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2012.

[24] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[25] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4512–4517.

[26] R. Konda, H. M. La, and J. Zhang, "Decentralized function approximated q-learning in multi-robot systems for predator avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6342–6349, 2020.

[27] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[28] Z. Chen and H.-T. Zhang, "No-beacon collective circular motion of jointly connected multi-agents," *Automatica*, vol. 47, no. 9, pp. 1929–1937, 2011.

[29] N. Ceccarelli, M. Di Marco, A. Garulli, and A. Giannitrapani, "Collective circular motion of multi-vehicle systems," *Automatica*, vol. 44, no. 12, pp. 3025–3035, 2008.

[30] S. Gao, R. Song, and Y. Li, "Cooperative control of multiple nonholonomic robots for escorting and patrolling mission based on vector field," *IEEE Access*, vol. 6, pp. 41 883–41 891, 2018.

[31] K. Shojaei, "Leader–follower formation control of underactuated autonomous marine surface vehicles with limited torque," *Ocean Engineering*, vol. 105, pp. 196–205, 2015.

[32] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[33] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 916–923, 2018.

[34] R. Cui, S. S. Ge, B. V. E. How, and Y. S. Choo, "Leader–follower formation control of underactuated autonomous underwater vehicles," *Ocean Engineering*, vol. 37, no. 17-18, pp. 1491–1502, 2010.

[35] S. He, M. Wang, S.-L. Dai, and F. Luo, "Leader–follower formation control of usvs with prescribed performance and collision avoidance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 572–581, 2018.

[36] A. Loria, J. Dasdemir, and N. A. Jarquin, "Leader–follower formation and tracking control of mobile robots along straight paths," *IEEE transactions on control systems technology*, vol. 24, no. 2, pp. 727–732, 2015.

[37] M. A. M. Villa, L. E. S. Guzman, A. R. Guevara, and N. F. V. Toledo, "Decentralized cooperative control technique for non-holonomic mobile formation," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*. IEEE, 2014, pp. 1–5.

[38] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE transactions on automatic control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[39] L. Guzman, D. R. A. Flórez, M. M. Villa, and C. Rojas, "Sistema de transporte cooperativo desarrollado para un grupo de robots móviles no-holonómicos usando el método líder virtual," *Memorias*, 2015.

[40] W. Ren and R. Beard, "Virtual structure based spacecraft formation control with formation feedback," in *AIAA Guidance, Navigation, and control conference and exhibit*, 2002, p. 4963.

[41] C. Venkatesh, S. K. Surve, and N. Singh, "Flatness based formation control of non holonomic vehicle," in *International Conference on Advances in Computing, Communication and Control*. Springer, 2011, pp. 486–493.

[42] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[43] L. I. G. Calandín, "Modelado cinemático y control de robots móviles con ruedas," Ph.D. dissertation, Universidad Politécnica de Valencia, 2006.

[44] E. Restrepo, A. Loria, I. Sarras, and J. Marzat, "Leader-follower consensus of unicycles with communication range constraints via smooth time-invariant feedback," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 737–742, 2020.

[45] D. Shen, W. Sun, and Z. Sun, "Adaptive pid formation control of nonholonomic robots without leader's velocity information," *ISA transactions*, vol. 53, no. 2, pp. 474–480, 2014.

[46] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning dqn algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177–183, 2020.

[47] M. Defoort, T. Floquet, A. Kokosy, and W. Perruquetti, "Sliding-mode formation control for cooperative autonomous mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 11, pp. 3944–3953, 2008.

[48] Y.-Y. Chen and Y.-P. Tian, "A backstepping design for directed formation control of three-coleader agents in the plane," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 19, no. 7, pp. 729–745, 2009.

[49] O. Ramírez-Cárdenas, J. Guerrero-Castellanos, and J. Linares-Flores, "Control de formación de múltiples robots móviles de auto-balanceo vía una comunicación disparada por eventos," 2019.

[50] M. F. S. Rabelo, A. S. Brandão, and M. Sarcinelli-Filho, "Centralized control for an heterogeneous line formation using virtual structure approach," in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*. IEEE, 2018, pp. 135–140.

[51] C. De La Cruz and R. Carelli, "Dynamic modeling and centralized formation control of mobile robots," in *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*. IEEE, 2006, pp. 3880–3885.

[52] J. Russell Carpenter, "Decentralized control of satellite formations," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 12, no. 2-3, pp. 141–161, 2002.

[53] R. A. Brooks and J. H. Connell, "Asynchronous distributed control system for a mobile robot," in *Mobile Robots I*, vol. 727. International Society for Optics and Photonics, 1987, pp. 77–84.

[54] J. Shamma, *Cooperative control of distributed multi-agent systems*. John Wiley & Sons, 2008.

[55] S.-L. Dai, S. He, X. Chen, and X. Jin, "Adaptive leader–follower formation control of nonholonomic mobile robots with prescribed transient and steady-state performance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3662–3671, 2019.

[56] Z. Miao, Y.-H. Liu, Y. Wang, G. Yi, and R. Fierro, "Distributed estimation and control for leader-following formations of nonholonomic mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1946–1954, 2018.

[57] X. Yu and L. Liu, "Distributed formation control of nonholonomic vehicles subject to velocity constraints," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1289–1298, 2016.

[58] L. Z. Xian, M. N. Mahyuddin, and G. Herrmann, "Leader-following distributed control of multiple nonholonomic wheeled mobile robots," in *TENCON 2015-2015 IEEE Region 10 Conference*. IEEE, 2015, pp. 1–6.

[59] C. A. V. León, "Control inteligente aplicado a robótica colaborativa," 2016.

[60] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, 2011, pp. 1–8.

[61] L. Euler, "Leonhard euler and the königsberg bridges," *Scientific American*, vol. 189, no. 1, pp. 66–72, 1953.

[62] E. Novo and A. M. Alonso, "Aplicaciones de la teoría de grafos a algunos juegos de estrategia," *Suma*, vol. 46, pp. 31–35, 2004.

[63] A. Recuero, "Aplicaciones de la teoría de grafos: búsqueda de caminos en una red y análisis de su conectividad," *Informes de la construcción*, vol. 46, no. 433, pp. 33–45, 1994.

[64] B. Básica and K. Rosen, "Matemática discreta y sus aplicaciones," *McGraw-Hill*, 2004.

[65] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE transactions on automatic control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[66] J. A. Ávila Cadena, "Diseño e implementación de estrategias de consenso líder-seguidor para un sistema multi-agentes." 2019.

[67] I. Grave, Y. Tang, and M. Sen, "Control cooperativo de sistemas multi-agentes y sus aplicaciones," Repositorio Institucional de la UNAM, 2010.

[68] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[69] R. M. Murray, "Recent research in cooperative control of multivehicle systems," 2007.

[70] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008, vol. 27, no. 2.

[71] J. Sánchez-Santana, J. Guerrero-Castellanos, M. Villarreal-Cervantes, and S. Ramırez-Martınez, "Control distribuido y disparado por eventos para la formación de robots móviles tipo (3, 0)," in *Congreso Nacional de Control Automático*, vol. 2017, 2018.

[72] H. K. Khalil, "Lyapunov stability," *Control Systems, Robotics and AutomatioN–Volume XII: Nonlinear, Distributed, and Time Delay Systems-I*, p. 115, 2009.

[73] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," 2003.

[74] G. Curiel Olivares *et al.*, "Diseño y construcción del sistema de tracción eléctrica basado en motores de cd sin escobillas para un vehículo de dos ruedas," *REPOSITORIO NACIONAL CONACYT*, 2019.

[75] Q. Lu, Y. Sun, and S. Mei, *Nonlinear control systems and power system dynamics*. Springer Science & Business Media, 2001, vol. 10.

[76] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "On differentially flat nonlinear systems," in *Nonlinear Control Systems Design 1992*. Elsevier, 1993, pp. 159–163.

[77] ——, "Flatness and defect of non-linear systems: introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[78] ——, "A lie-backlund approach to equivalence and flatness of nonlinear systems," *IEEE Transactions on automatic control*, vol. 44, no. 5, pp. 922–937, 1999.

[79] A. Abadi, A. El Amraoui, H. Mekki, and N. Ramdani, "Robust tracking control of quadrotor based on flatness and active disturbance rejection control," *IET Control Theory & Applications*, vol. 14, no. 8, pp. 1057–1068, 2020.

[80] H. Sira-Ramirez and S. K. Agrawal, *Differentially flat systems*. Crc Press, 2004.

[81] H. J. Marquez, *Nonlinear control systems: analysis and design*. John Wiley Hobokenˆ eN. JNJ, 2003, vol. 161.

[82] M. A. Henson and D. E. Seborg, *Nonlinear process control*. Prentice Hall PTR Upper Saddle River, New Jersey, 1997.

[83] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System*. .º'Reilly Media, Inc.", 2015.

[84] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System*. Packt Publishing Ltd, 2018.

[85] K. Besseghieur, W. Kaczmarek, J. Panasiuk, and P. Prusaczyk, "Formation control of mobile robots under ros," *Engineering Mechanics*, vol. 23, pp. 138–41, 2017.

[86] R. Bischoff, U. Huggenberger, and E. Prassler, "Kuka youbot-a mobile manipulator for research and education," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.

[87] S. I. Grossman, *Álgebra lineal*. McGraw Hill Educación, 2008.

[88] A. Isidori, E. Sontag, and M. Thoma, *Nonlinear control systems*. Springer, 1995, vol. 3.

[89] J. Chiasson, "A new approach to dynamic feedback linearization control of an induction motor," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 391–397, 1998.

[90] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[91] S. Knorn, Z. Chen, and R. H. Middleton, "Overview: Collective control of multiagent systems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 4, pp. 334–347, 2015.

Apéndice A

Trabajo publicado

Durante el desarrollo de este proyecto de investigación, los conceptos de consenso en SMA compuestos de distintos sistemas electromecánicos fueron implementados en los cursos de control no lineal. Como resultado publicó un artículo en la *23ª Conferencia Internacional del IEEE sobre Tecnología Industrial*. Este artículo lleva por título: Control de Velocidad de un Sistema de Múltiples PMSM basado en Consenso Líder-Seguidor. A continuación se muestra el certificado de presentación y la primer página del artículo.

Exon Fernando Villalobos Álvarez

Successfully participated as an author and presenter at ICIT 2022, held online 22-25 August 2022.

Speed Tracking Control for Multi-PMSM Based on Leader-follower Consensus

Presenter: Exon Villalobos

Co-authors: Jesús Linares Flores, Oscar David Ramírez Cárdenas

PROF. MARIUSZ MALINOWSKI
IES PRESIDENT

PROF. VALERIY VYATKIN ICIT 2022
GENERAL CHAIR

PROF. LUIS GOMES ICIT 2022
GENERAL CHAIR

PROF. MO-YUEN CHOW ICIT 2022
GENERAL CHAIR

PROF. XINPING GUAN ICIT 2022
GENERAL CHAIR

Speed Tracking Control for Multi-PMSM Based on Leader-follower Consensus.

1st Exon-F. Villalobos-Álvarez
Universidad Tecnológica de la Mixteca
División de estudios de Posgrado
Huajuapan de León, Oaxaca, México
exonfer@gmail.com

2nd Jesús Linares-Flores
Universidad Tecnológica de la Mixteca
Instituto de Electrónica y Mecatrónica
Huajuapan de León, Oaxaca, México
jlinares@mixteco.utm.mx

3rd Oscar-D. Ramírez-Cárdenas
Universidad Tecnológica de la Mixteca
División de estudios de Posgrado
Huajuapan de León, Oaxaca, México
odramirez@mixteco.utm.mx

***Abstract*—This article deals with the design and implementation of a collaborative and decentralized control for synchronizing the angular velocity tracking of a group of spatially distributed permanent magnet synchronous (PMSM) motors. Via feedback linearization, acting as an internal loop, the dynamics of the PMSM rewrites on two subsystems, a subsystem of second-order integrator and a subsystem of the first-order integrator, which is considered an agent. Then, a decentralized collaborative control strategy is proposed, which solves the problem of leader-follower consensus for the multi-agent system and thus speed synchronization. Further, in this article is proposed a velocity profiles formation among agents. The communication topology between agents models using an undirected and connected graph. The decentralized control law incorporates a desired reference trajectory of angular velocity, which employs a Bezier polynomial. The i-th agent transmits the angular velocity trajectory information to its neighbor. The Matlab-Simulink/PSIM simulation results show excellent performance for angular velocity consensus for regulation and tracking tasks.**

***Index Terms*—Permanent magnet synchronous motor (PMSM), feedback linearization, cooperative control, leader-follower, consensous.**

I. Introduction

In recent years, Cyber-Physical Systems (CFS), also called CPS (Cyber–Physical Systems), have attracted the attention of scientists and engineers in the area of automatic control, communications, automotive, integrated systems, and microgrids [1]–[3]. CFS's represent a fusion of computational entities and physical elements, which interconnect by communication networks. This merger imposes tremendous challenges, generating innovations and technological advances in Industry 4.0 and IoT (Internet of Things) [3]. The objective of these new paradigms is to ensure real-time, closed-loop monitoring and control of a wide variety of processes to save energy, size, and weight [4]. On the other hand, multi–agent systems (MAS) are a particular case of CFS and consist of a network of dynamic systems that interact and communicate to achieve a goal in a group and collaborative way. Allows the development of tasks that would be impossible to carry out individually [5] [6]. Examples of MAS find in the networks of uncrewed aerial vehicles [7], mobile ground robots [8] , among others. Distributed or decentralized cooperative control for the control of MAS has aroused great interest because they provide an attractive solution to the regulation of large-scale systems, reducing the complexity in formulating the problem and reducing the computational load required for the implementation of algorithms in real–time [9], [10]. Among the different problems of cooperative control is consensus or synchronization. Of particular interest is manipulating a group of Agents having a leader whose behavior is independent of the other Agents. The problem in question is known as the leader–follower consensus [11].

There is a growing demand for electromechanical systems that work in a synchronized way for highly complex applications; paper manufacturing machines, assembly lines, robotic systems, among others [12]. In addition, applications where actuators that support high loads require, such as tunnel boring machines [13], vehicle traction [14], to name a few, should be considered. In these cases, a single-motor would not be able to complete the objective. Adopting a multi-motor approach has several advantages, notably lengthening the life of motors, reducing stress on power electronic devices due to high currents, and being fault-tolerant. For this reason, high load torque multi-motor systems have recently resorted to sophisticated torque distribution and speed synchronization control schemes [15]. Synchronization strategies for multi-motor systems reported, such as synchronization based on adjacent coupling control [12], the relative coupling synchronization method [16], the master scheme -slave [17], cross-coupling strategy [18], coupling ring synchronization scheme [13] and virtual axis synchronization [19]. In these cases, the goal is to make the tracking and timing error converge to zero between any adjacent motors and motors.

These synchronization strategies combined with speed control techniques are generally applied to motors, ranging from PID controllers [20] to neural networks and fuzzy control [21]. For nonlinear control schemes based on sliding modes and control strategies based on Lyapunov's second method [12], [22], output feedback controllers that include the use of asymptotic observers of variable structure [23], and even optimal control [24]. These schemes have generated incredible results from a theoretical and practical perspective. However, all of them are centralized control strategies. They don't consider the case where the motors distribute spatially or the control loop is closed by a communication network.

Apéndice B

Códigos de descripción del RMD

A continuación se presentan los códigos de descripción URDF y Xacro del robot móvil diferencial diseñado en software CAD. Éstos permiten la representación del modelo físico en el software de simulación Gazebo. Estos archivos se pueden encontrar en el siguiente enlace https://github.com/ExonVillalobos/SMA/tree/main/RMD/urdf.

B.1. Código URDF

```
<?xml version="1.0" encoding="utf-8"?>
<robot
  name="RMD">
  <link
    name="base_link">
    <inertial>
      <origin
        xyz="7.8882E-06 0.010304 0.015319"
        rpy="0 0 0" />
      <mass
        value="0.16545" />
      <inertia
        ixx="0.000116"
        ixy="-1.3693E-10"
        ixz="5.0864E-09"
        iyy="0.00012082"
        iyz="4.8963E-06"
        izz="0.00020689" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
```

```
      <mesh
        filename="package://RMD/meshes/base_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.75294 0.75294 0.75294 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://RMD/meshes/base_link.STL" />
    </geometry>
  </collision>
</link>
<link
  name="R_Link">
  <inertial>
    <origin
      xyz="-0.0095 4.099E-17 2.3506E-16"
      rpy="0 0 0" />
    <mass
      value="0.015656" />
    <inertia
      ixx="2.513E-06"
      ixy="-4.5727E-21"
      ixz="-1.3954E-21"
      iyy="1.8577E-06"
      iyz="2.1487E-22"
      izz="1.8577E-06" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://RMD/meshes/R_Link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.75294 0.75294 0.75294 1" />
    </material>
  </visual>
```

```
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/R_Link.STL" />
      </geometry>
    </collision>
  </link>
  <joint
    name="R_joint"
    type="continuous">
    <origin
      xyz="0.0373 0 0.005"
      rpy="0 0 -3.1416" />
    <parent
      link="base_link" />
    <child
      link="R_Link" />
    <axis
      xyz="-1 0 0" />
  </joint>
  <link
    name="L_Link">
    <inertial>
      <origin
        xyz="-0.0095 6.5919E-17 2.8189E-18"
        rpy="0 0 0" />
      <mass
        value="0.015656" />
      <inertia
        ixx="2.513E-06"
        ixy="-4.5703E-21"
        ixz="9.9263E-22"
        iyy="1.8577E-06"
        iyz="3.9705E-22"
        izz="1.8577E-06" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/L_Link.STL" />
      </geometry>
      <material
        name="">
```

```
        <color
          rgba="0.75294 0.75294 0.75294 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/L_Link.STL" />
      </geometry>
    </collision>
  </link>
  <joint
    name="L_joint"
    type="continuous">
    <origin
      xyz="-0.0373 0 0.005"
      rpy="0 0 0" />
    <parent
      link="base_link" />
    <child
      link="L_Link" />
    <axis
      xyz="-1 0 0" />
  </joint>
</robot>
```

B.2. Código xacro

```
<?xml version="1.0"?>
<robot name="RMD" xmlns:xacro="http://www.ros.org/wiki/xacro">
     <xacro:macro name="MatInerciaRueda" params="masa">
      <inertial>
        <origin
        xyz="-0.0095 0 0"
        rpy="0 0 0" />
       <mass value="${masa}" />
        <inertia
        ixx="2.513E-06"
        ixy="-4.5727E-21"
        ixz="-1.3954E-21"
        iyy="1.8577E-06"
        iyz="2.1487E-22"
        izz="1.8577E-06" />
      </inertial>
   </xacro:macro>
```

```
  <link
    name="base_link">
    <inertial>
      <origin
        xyz="1.0792E-06 0.0019022 0.0062789"
        rpy="0 0 0" />
      <mass
        value="0.085922" />
      <inertia
      ixx="2.9758E-05"
      ixy="-3.8502E-10"
      ixz="5.1049E-09"
      iyy="2.4055E-05"
      iyz="3.3956E-10"
      izz="4.9281E-05" />
    </inertial>
    <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/base_link.STL" />
      </geometry>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/base_link.STL" />
      </geometry>
    </collision>
  </link>
  <link
    name="R_link">
<xacro:MatInerciaRueda masa="0.015656"/>
        <visual>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://RMD/meshes/R_Link.STL" />
      </geometry>
      <material
        name="">
        <color
```

```
        rgba="0.75294 0.75294 0.75294 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://RMD/meshes/R_Link.STL" />
    </geometry>
  </collision>
</link>
<joint
  name="R_joint"
  type="continuous">
  <origin
    xyz="0.0373 0 0.005"
    rpy="0 0 -3.1416" />
  <parent
    link="base_link" />
  <child
    link="R_link" />
  <axis
    xyz="-1 0 0" />
</joint>
<link
  name="L_link">
  <xacro:MatInerciaRueda masa="0.015656"/>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://RMD/meshes/L_Link.STL" />
    </geometry>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://RMD/meshes/L_Link.STL" />
    </geometry>
  </collision>
</link>
<joint
```

```
    name="L_joint"
    type="continuous">
    <origin
      xyz="-0.0373 0 0.005"
      rpy="0 0 0" />
    <parent
      link="base_link" />
    <child
      link="L_link" />
    <axis
      xyz="-1 0 0" />
  </joint>
  <gazebo reference="R_link">
    <material>Gazebo/Black</material>
  </gazebo>
  <gazebo reference="L_link">
    <material>Gazebo/Black</material>
  </gazebo>
  <gazebo reference="base_link">
    <material>Gazebo/Blue</material>
  </gazebo>
</robot>
```

Apéndice C

Archivo de ejecución sma.launch

En este apéndice se presentan el código de lanzamiento a ejecución del SMA que consta de diez nodos o agentes RMD. Este archivo se puede encontrar en el siguiente enlace https://github.com/ExonVillalobos/SMA/tree/main/RMD/launch.

```
<launch>
<arg name="paused" default="false"/>
<arg name="use_sim_time" default="true"/>
<arg name="gui" default="true"/>
<arg name="headless" default="false"/>
<arg name="debug" default="false"/>
<!-- We resume the logic in empty_world.launch -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
<arg name="debug" value="$(arg debug)" />
<arg name="gui" value="$(arg gui)" />
<arg name="paused" value="$(arg paused)"/>
<arg name="use_sim_time" value="$(arg use_sim_time)"/>
<arg name="headless" value="$(arg headless)"/>
</include>

<arg name="x1" default="1.6"/>
<arg name="y1" default="0"/>
<arg name="z1" default="0.012"/>
<arg name="x5" default="2.4"/>
<arg name="y5" default="0.8"/>
<arg name="z5" default="0.012"/>
<arg name="x2" default="2.4"/>
<arg name="y2" default="1.6"/>
<arg name="z2" default="0.012"/>
<arg name="x6" default="2.4"/>
<arg name="y6" default="2.4"/>
<arg name="z6" default="0.012"/>
<arg name="x7" default="1.6"/>
<arg name="y7" default="2.4"/>
<arg name="z7" default="0.012"/>
```

```
<arg name="x3" default="0.8"/>
<arg name="y3" default="2.4"/>
<arg name="z3" default="0.012"/>
<arg name="x8" default="0.8"/>
<arg name="y8" default="1.6"/>
<arg name="z8" default="0.012"/>
<arg name="x9" default="0.8"/>
<arg name="y9" default="0.8"/>
<arg name="z9" default="0.012"/>
<arg name="x4" default="0.8"/>
<arg name="y4" default="0"/>
<arg name="z4" default="0.012"/>
<arg name="x10" default="2.4"/>
<arg name="y10" default="0"/>
<arg name="z10" default="0.012"/>

<param name="robot1" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD1.xacro'" />
<param name="robot2" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD2.xacro'" />
<param name="robot3" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD3.xacro'" />
<param name="robot4" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD4.xacro'" />
<param name="robot5" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD5.xacro'" />
<param name="robot6" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD6.xacro'" />
<param name="robot7" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD7.xacro'" />
<param name="robot8" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD8.xacro'" />
<param name="robot9" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD9.xacro'" />
<param name="robot10" command="$(find xacro)/xacro '$(find RMD)/urdf/RMD10.xacro'"/>

<node name="urdf_spawner1" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif1 -param robot1 -x $(arg x1)
-y $(arg y1) -z $(arg z1)">
</node>

<node name="urdf_spawner2" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif2 -param robot2 -x $(arg x2)
-y $(arg y2) -z $(arg z2)">
</node>

<node name="urdf_spawner3" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif3 -param robot3 -x $(arg x3)
-y $(arg y3) -z $(arg z3)"/>

<node name="urdf_spawner4" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif4 -param robot4 -x $(arg x4)
-y $(arg y4) -z $(arg z4)"/>
```

```
<node name="urdf_spawner5" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif5 -param robot5 -x $(arg x5)
-y $(arg y5) -z $(arg z5)"/>

<node name="urdf_spawner6" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif6 -param robot6 -x $(arg x6)
-y $(arg y6) -z $(arg z6)"/>

<node name="urdf_spawner7" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif7 -param robot7 -x $(arg x7)
-y $(arg y7) -z $(arg z7)"/>

<node name="urdf_spawner8" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif8 -param robot8 -x $(arg x8)
-y $(arg y8) -z $(arg z8)"/>

<node name="urdf_spawner9" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif9 -param robot9 -x $(arg x9)
-y $(arg y9) -z $(arg z9)"/>

<node name="urdf_spawner10" pkg="gazebo_ros"
type="spawn_model" respawn="false" output="screen"
args="-urdf -model dif10 -param robot10 -x $(arg x10)
-y $(arg y10) -z $(arg z10)"/>-->
</launch>
```

Apéndice D

Código de leyes de control

En este apéndice se presentan los códigos Phyton de los dos controladores (clásico h y planitud diferencial) del agente líder. En el siguiente enlace se pueden consultar estos archivoshttps://github.com/ExonVillalobos/SMA/tree/main/RMD/scripts. También se muestran los códigos para lanzar a ejecución (*.launch*) los diez nodos de control. Estos archivos se pueden encontrar en el enlace del apéndice anterior.

D.1. Control clásico en h (controlh1.py)

```
import rospy
import math
import ast
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
#Mensajes de inicialización y fin de nodo
msg = """
En linea control 1
"""
e = """
EStoy fuera control 1
"""
# Lectura de ganancias y frecuencia de ejecución para el control
k1=rospy.get_param("control1/kp")
hz=rospy.get_param("control1/hz")
delta=1/hz
#Lectura de desfases entre agentes
lx=ast.literal_eval(rospy.get_param("/control1/lx"))
ly=ast.literal_eval(rospy.get_param("/control1/ly"))
lx1=lx[0]
ly1=ly[0]
lx2=lx[1]
ly2=ly[1]
```

```
lx3=lx[2]
ly3=ly[2]
lx4=lx[3]
ly4=ly[3]

# Condiciones iniciales de leyes de control
V=0.0
w=0.0

#Variables para cada agente
x1c=0;y1c=0;th1=0;x1h=0;y1h=0
x2c=0;y2c=0;th2=0;x2h=0;y2h=0
x3c=0;y3c=0;th3=0;x3h=0;y3h=0
x4c=0;y4c=0;th4=0;x4h=0;y4h=0

#Parametros de robot móvil
h=0.064
L=0.092

#Parametros de trayectoria deseada
t=0;xd=0;xd1=0;yd=0;dxd=0;dyd=0
xini=1.5;yini=1
xfin=2.5;yfin=2
tini=20;tfin=40

# Funciones para obtener posicion y ángulo de agentes
def callback1(data):
    global x1c,y1c,y1h,x1h,th1
    #print(data)
    x1c = data.pose.pose.position.x
    y1c = data.pose.pose.position.y
    #Calculo de orientación en radianes
    c1  = data.pose.pose.orientation.z
    c2  = data.pose.pose.orientation.w
    th1 = 2*math.atan2(c1,c2)
    if th1 <0:
        th1 = 2*math.pi + th1
    th1 = th1 + math.pi/2
    #Calculo de punto de operación h menos desfase
    x1h = x1c + h*math.cos(th1) - lx1
    y1h = y1c + h*math.sin(th1) - ly1

def callback2(data):
    global x2c,y2c,y2h,x2h,th2
    #print(data)
    x2c = data.pose.pose.position.x
    y2c = data.pose.pose.position.y
    c1  = data.pose.pose.orientation.z
    c2  = data.pose.pose.orientation.w
```

```
    th2 = 2*math.atan2(c1,c2)
    if th2 <0:
        th2 = 2*math.pi + th2
    th2 = th2 + math.pi/2
    x2h = x2c + h*math.cos(th2) - lx2
    y2h = y2c + h*math.sin(th2) - ly2

def callback3(data):
    global x3c,y3c,y3h,x3h,th3
    #print(data)
    x3c = data.pose.pose.position.x
    y3c = data.pose.pose.position.y
    c1  = data.pose.pose.orientation.z
    c2  = data.pose.pose.orientation.w
    th3 = 2*math.atan2(c1,c2)
    if th3 <0:
        th3 = 2*math.pi + th3
    th3 = th3 + math.pi/2
    x3h = x3c + h*math.cos(th3) - lx3
    y3h = y3c + h*math.sin(th3) - ly3

def callback4(data):
    global x4c,y4c,y4h,x4h,th4
    #print(data)
    x4c = data.pose.pose.position.x
    y4c = data.pose.pose.position.y
    c1  = data.pose.pose.orientation.z
    c2  = data.pose.pose.orientation.w
    th4 = 2*math.atan2(c1,c2)
    if th4 <0:
        th4=2*math.pi + th4
    th4 = th4 + math.pi/2
    x4h = x4c + h*math.cos(th4) - lx4
    y4h = y4c + h*math.sin(th4) - ly4

# Control clásico h
def control():
    global V,w
    r1=dxd-k1*(x1h-xd)-k1*(x1h-x2h)-k1*(x1h-x3h)-k1*(x1h-x4h)
    r2=dyd-k1*(y1h-yd)-k1*(y1h-y2h)-k1*(y1h-y3h)-k1*(y1h-y4h)
    #print(r1)
    #print(r2)
    V = r1*math.cos(th1) + r2*math.sin(th1)
    w = -r1*math.sin(th1)/h + r2*math.cos(th1)/h
    #Saturación de salidas en función del robot real
    if (V>0.4):
        V=0.4
    if (V<-0.4):
        V=-0.4
```

```
    if (w>8.69):
        w=8.68
    if (w<-8.69):
        w=-8.68

# Funcion de trayectoria deseada
def trayectoria():
    global t,xd,yd,dxd,dyd,xd1
    time=t*delta
    tau= (time - tini)/(tfin-tini)
    poli=pow(tau,5)*(126 - 420*tau + 540*pow(tau,2) - 315*pow(tau,3) + 70*pow(tau,4))
    if time < tini:
        xd=xini
        xd1=xini
    if time > tfin:
        xd=xfin
    if (time >= tini)&(time <= tfin):
        xd= xini + poli*(xfin-xini)

    dxd=(xd-xd1)/delta
    yd = (yfin-yini)/(xfin-xini)*(xd - xini) + yini
    dyd= (yfin-yini)/(xfin-xini)*dxd
    xd1=xd
    t=t+1

if __name__=="__main__":
    #inicialización de nodo
    rospy.init_node('Control1',anonymous=True)
    #Publicar en tópico
    pub = rospy.Publisher('/1/cmd_vel1', Twist, queue_size=5)
    #Suscripciones a topicos de agentes vecinos
    rospy.Subscriber("/1/odom1", Odometry, callback1)
    rospy.Subscriber("/2/odom2", Odometry, callback2)
    rospy.Subscriber("/3/odom3", Odometry, callback3)
    rospy.Subscriber("/4/odom4", Odometry, callback4)
    rate = rospy.Rate(hz) # Frecuencia de ejecución en Hz
    #Escirtura de datos en archivo .txt
    archivo = open("/home/exon/SMA/src/RMD/scripts/plotear/ah1234.txt","w")
    # bloque de detección de errores
    try:
        print (msg)
        while not rospy.is_shutdown():
            trayectoria()
            control()
            twist = Twist()
            twist.linear.x = V; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = w
            pub.publish(twist)
            archivo.write(str((t*delta))+" "+str((x1h))+" "+str((y1h))+" "+str((th1))+" ")
```

```
            archivo.write(str((x2h)) + " " + str((y2h)) + " " + str((th2)) + " ")
            archivo.write(str((x3h)) + " " + str((y3h)) + " " + str((th3)) + " ")
            archivo.write(str((x4h)) + " " + str((y4h)) + " " + str((th4)))
            archivo.write("\n")
            rate.sleep()
        archivo.close()

    except rospy.ROSInterruptException:
        twist = Twist()
        twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
        pub.publish(twist)
        print(e)
        pass

    finally:
        twist = Twist()
        twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
        pub.publish(twist)
```

D.2. Planitud diferencial (controlc1.py)

```
import rospy
import math
import time
import ast
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
#Mensajes de inicialización y fin de nodo
msg = """
En linea control 1
"""
e = """
EStoy fuera control 1
"""
# Lectura de ganancias y frecuencia de ejecución para el control
k0=rospy.get_param("control1/kp")
k1=rospy.get_param("control1/kd")
hz=rospy.get_param("control1/hz")
delta=1/hz

#Lectura de tiempos para cada formación
tiempos=ast.literal_eval(rospy.get_param("/control1/tiempos"))
t1=tiempos[0]
t2=tiempos[1]

# Lectura de desfases entre agentes
```

```
lx=ast.literal_eval(rospy.get_param("/control1/lx"))
ly=ast.literal_eval(rospy.get_param("/control1/ly"))
lx1=lx[0][0]
ly1=ly[0][0]
lx2=lx[0][1]
ly2=ly[0][1]
lx3=lx[0][2]
ly3=ly[0][2]
lx4=lx[0][3]
ly4=ly[0][3]

#Condiciones iniciales de leyes de control
V1=0
V2=0.01
w=0.0

#Variables para cada agente
x1c1=0;dx1c=0;y1c1=0;x1c2=0;y1c2=0;th1=0
x2c1=0;y2c1=0;x2c2=0;y2c2=0
x3c1=0;y3c1=0;x3c2=0;y3c2=0
x4c1=0;y4c1=0;x4c2=0;y4c2=0

#Parametros de robot móvil
L=0.092

#Parametros de trayectoria deseada
xd=0;xd1=0yd=0;yd1=0;dxd=0;dxd1=0
dyd=0;dyd1=0;ddxd=0;ddyd=0
xini=1.5;yini=1
xfin=2.5;yfin=2
tini=20;tfin=40
t=0

# Funciones para obtener posicion y ángulo de agentes
def callback1(data):
    global x1c1,y1c1,th1
    #posiciones X y Y en punto c
    x1c1 = data.pose.pose.position.x
    y1c1 = data.pose.pose.position.y
    #Calculo de orientación en radianes
    c1  = data.pose.pose.orientation.z
    c2  = data.pose.pose.orientation.w
    th1 = 2*math.atan2(c1,c2)
    if th1 <0:
        th1 = 2*math.pi + th1
    th1 = th1 + math.pi/2

def callback2(data):
    global x2c1,y2c1
```

```
    #print(data)
    x2c1 = data.pose.pose.position.x
    y2c1 = data.pose.pose.position.y

def callback3(data):
    global x3c1,y3c1
    #print(data)
    x3c1 = data.pose.pose.position.x
    y3c1 = data.pose.pose.position.y

def callback4(data):
    global x4c1,y4c1
    #print(data)
    x4c1 = data.pose.pose.position.x
    y4c1 = data.pose.pose.position.y

def muestras():
    global x1c2,y1c2,x2c2,y2c2,x3c2,y3c2,x4c2,y4c2
    # actualización de muestras anteriores
    x1c2 = x1c1
    y1c2 = y1c1

    x2c2 = x2c1
    y2c2 = y2c1

    x3c2 = x3c1
    y3c2 = y3c1

    x4c2 = x4c1
    y4c2 = y4c1

#Función que cambia el valor de los desfases
def desfases():
    global lx1,ly1,lx2,ly2,lx3,ly3,lx4,ly4
    if t>t1*hz:
        lx1=lx[1][0]
        ly1=ly[1][0]

        lx2=lx[1][1]
        ly2=ly[1][1]

        lx3=lx[1][2]
        ly3=ly[1][2]

        lx4=lx[1][3]
        ly4=ly[1][3]
    if t>t2*hz:
        lx1=lx[2][0]
        ly1=ly[2][0]
```

```
        lx2=lx[2][1]
        ly2=ly[2][1]

        lx3=lx[2][2]
        ly3=ly[2][2]

        lx4=lx[2][3]
        ly4=ly[2][3]

#Planitud diferencial
def control():
    global V1,V2,w
    # calculo de velocidades
    dx1c = (x1c1 - x1c2)/delta
    dy1c = (y1c1 - y1c2)/delta

    dx2c = (x2c1 - x2c2)/delta
    dy2c = (y2c1 - y2c2)/delta

    dx3c = (x3c1 - x3c2)/delta
    dy3c = (y3c1 - y3c2)/delta

    dx4c = (x4c1 - x4c2)/delta
    dy4c = (y4c1 - y4c2)/delta
    # calculo de desfases
    x1=x1c1-lx1
    x2=x2c1-lx2
    x3=x3c1-lx3
    x4=x4c1-lx4
    y1=y1c1-ly1
    y2=y2c1-ly2
    y3=y3c1-ly3
    y4=y4c1-ly4
    # calculo de control auxiliar
    r11 =ddxd - k0*(x1-xd) - k1*(dx1c-dxd) - k0*(x1 - x2) - k1*(dx1c - dx2c)
    r1 = r11 - k0*(x1 - x3) - k1*(dx1c - dx3c) - k0*(x1 - x4) - k1*(dx1c - dx4c)
    # calculo de control auxiliar
    r22 =ddyd - k0*(y1-yd) - k1*(dy1c-dyd) - k0*(y1 - y2) - k1*(dy1c - dy2c)
    r2 = r22 - k0*(y1 - y3) - k1*(dy1c - dy3c) - k0*(y1 - y4) - k1*(dy1c - dy4c)
    # calculo de velocidades

    V1 = (r1*math.cos(th1) + r2*math.sin(th1))*delta + V2
    w = (-r1*math.sin(th1) + r2*math.cos(th1))/V1
    # actualizar valor anterior de velocidad lineal
    if (V1>0.4):
        V1=0.4
    if (V1<-0.4):
        V1=-0.4
```

```
    if (w>8.69):
        w=8.68
    if (w<-8.69):
        w=-8.68
    V2=V1

# Funcion de trayectoria deseada: curva Bézier
def trayectoria():
    global t,xd,yd,dxd,dyd,ddxd,ddyd,xd1,dxd1
    time=t*delta
    tau= (time - tini)/(tfin-tini)
    poli=pow(tau,5)*(126 - 420*tau + 540*pow(tau,2) - 315*pow(tau,3) + 70*pow(tau,4))
    if time < tini:
        xd=xini
        xd1=xini
    if time > tfin:
        xd=xfin
    if (time >= tini)&(time <= tfin):
        xd= xini + poli*(xfin-xini)
    dxd=(xd-xd1)/delta
    ddxd=(dxd-dxd1)/delta
    yd = (yfin-yini)/(xfin-xini)*(xd - xini) + yini
    dyd= (yfin-yini)/(xfin-xini)*dxd
    ddyd=(yfin-yini)/(xfin-xini)*ddxd
    xd1=xd
    dxd1=dxd
    t=t+1

# Funcion de trayectoria deseada: Leminiscata
def leminiscata():
    global t,xd,yd,dxd,dyd,ddxd,ddyd,xd1,dxd1,yd1,dyd1
    time2=t*delta
    time=0.05*t*delta
    xd=4*math.sin(time)/(1 + math.cos(time)*math.cos(time))
    yd=4*math.sin(time)*math.cos(time)/(1 + math.cos(time)*math.cos(time))
    dxd=(xd-xd1)/delta
    ddxd=(dxd-dxd1)/delta
    dyd=(yd-yd1)/delta
    ddyd=(dyd-dyd1)/delta
    xd1=xd
    dxd1=dxd
    yd1=yd
    dyd1=dyd
    t=t+1

if __name__=="__main__":

    rospy.init_node('Control1',anonymous=True)
    pub = rospy.Publisher('/1/cmd_vel1', Twist, queue_size=5)
```

```
rospy.Subscriber("/1/odom1", Odometry, callback1)
rospy.Subscriber("/2/odom2", Odometry, callback2)
rospy.Subscriber("/3/odom3", Odometry, callback3)
rospy.Subscriber("/4/odom4", Odometry, callback4)
rate = rospy.Rate(hz) # Frecuencia de ejecución en Hz
archivo = open("/home/exon/SMA/src/RMD/scripts/plotear/ac1234.txt","w")

try:
    print (msg)
    print (delta)
    while not rospy.is_shutdown():
        leminiscata()
        #print("\n--------Control1--------")
        if t > hz*0.1:
            control()
        desfases()
        muestras()
        twist = Twist()
        twist.linear.x = V1; twist.linear.y = 0; twist.linear.z = 0
        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = w
        pub.publish(twist)
        archivo.write(str((t*delta))+" "+str((x1c1))+" "+str((y1c1))+" ")
        archivo.write(str((x2c1)) + " " + str((y2c1)) + " ")
        archivo.write(str((x3c1)) + " " + str((y3c1))  + " ")
        archivo.write(str((x4c1)) + " " + str((y4c1)))
        archivo.write("\n")
        rate.sleep()
    archivo.close()

except rospy.ROSInterruptException:
    twist = Twist()
    twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
    twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
    pub.publish(twist)
    print(e)
    pass

finally:
    twist = Twist()
    twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
    twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
    pub.publish(twist)
```

D.3. Lanzamiento de nodos de control por control clásico h (controlh.launch)

```
<launch>
  <arg name="kp" default="2"/>
  <arg name="hz" default="400"/>

  <arg name="lx" default="[0,0.9,-0.3,-0.3,0.9,0.6,0.3,-0.9,-0.9,0.3]"/>
  <arg name="ly" default="[0,0.3,0.9,-0.9,-0.3,0.6,0.9,0.3,-0.3,-0.9]"/>

  <node pkg="RMD" type="controlh1.py" name="control1"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
-->
  </node>
  <node pkg="RMD" type="controlh2.py" name="control2"  output="screen">
        <param name='kp' value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh3.py" name="control3"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh4.py" name="control4"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh5.py" name="control5"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh6.py" name="control6"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
```

```
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh7.py" name="control7"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh8.py" name="control8"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh9.py" name="control9"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>

  <node pkg="RMD" type="controlh10.py" name="control10"  output="screen">
        <param name="kp" value="$(arg kp)" type="double" />
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="hz" value="$(arg hz)" type="double"/>
   </node>-->

</launch>
```

D.4. Lanzamiento de nodos de control por planitud diferencial (controlc.launch)

```
<launch>
  <arg name="kp" default="4"/>
  <arg name="kd" default="4"/>
  <arg name="hz" default="400"/>
  <arg name="tiempos" default="[50,70]"/>

  <arg name="lx" default="[0,0.9,-0.3,-0.3,0.9,0.6,0.3,-0.9,-0.9,0.3],
                          [0,0.8,-0.4,-0.4,0.4,0.4,0,-0.4,-0.4,0],
                          [0,0.8,-0.4,-0.4,0.8,0.8,0,-0.8,-0.8,0]"/>
```

```
  <arg name="ly" default="[0,0.3,0.9,-0.9,-0.3,0.6,0.9,0.3,-0.3,-0.9],
                          [0,0,0.6,-0.6,-0.2,0.2,0.4,0.2,-0.2,-0.4],
                          [0,0,0.6,-0.6,-0.4,0.4,0.8,0.4,-0.4,-0.8]"/>

  <node pkg="RMD" type="controlc1.py" name="control1"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

<node pkg="RMD" type="controlc2.py" name="control2"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc3.py" name="control3"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc4.py" name="control4"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc5.py" name="control5"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>
```

```
  <node pkg="RMD" type="controlc6.py" name="control6"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc7.py" name="control7"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc8.py" name="control8"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc9.py" name="control9"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>

  <node pkg="RMD" type="controlc10.py" name="control10"  output="screen">
        <param name="hz" value="$(arg hz)" type="double"/>
        <param name="kp" value="$(arg kp)" type="double"/>
        <param name="kd" value="$(arg kd)" type="double"/>
        <param name="lx" value="$(arg lx)" type="string" />
        <param name="ly" value="$(arg ly)" type="string" />
        <param name="tiempos" value="$(arg tiempos)" type="string" />
  </node>-->
</launch>
```