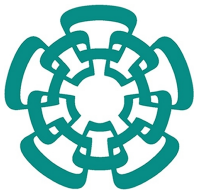


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN MECATRÓNICA

Problema de consenso en redes de agentes con retardo

Tesis que presenta:
Josue Heras Godínez

Para obtener el grado de:
Maestro en Ciencias

En la especialidad de:
Ingeniería Eléctrica

Director de Tesis:
Dr. Martín Velasco Villa

México, DF. Febrero,2015.

Dedicatoria

A Dios, por que es la fuerza que me sostiene y que me hace seguir esforzándome día a día.

A la memoria de mi papá, Hipólito Heras. Aún sigue siendo un gran ejemplo para mi.

A mi mamá, Ana Godínez, por su infinito amor y apoyo incondicional.

A mis hermanos, Poan e Ivan, por que no solo son mis hermanos, sino también mis mejores amigos.

Paz y Bien

Agradecimientos

- ♣ A Dios por darme la vida y las ganas de salir adelante, por guiar mi caminar y permitirme llegar hasta donde me encuentro, por ser mi fortaleza, mis esperanzas y el amor reflejado en mi familia.
- ♣ A mi mamá, que por su amor y ternura siempre ha estado conmigo en las buenas y en las malas, por que me ha dado el tesoro mas grande que jamas pudiera desear, el amor a Dios. Gracias mamá, te amo!
- ♣ A mis hermanos, par de locos que amo con todo mi corazón, y cuyo apoyo incondicional me ha ayudado a superar las pruebas mas difíciles de mi vida.
- ♣ A mi papá, que en paz descanse, por que ha sido mi ejemplo a seguir de esfuerzo y perseverancia, de humildad y caridad. Te amo apa!
- ♣ A mi asesor de tesis, el Doctor Martín Velasco Villa por sus enseñanzas, por su apoyo, su tiempo, su gran paciencia, pero sobre todo, por darme la oportunidad de crecer profesionalmente. Fue todo un honor y placer trabajar a su lado.
- ♣ A Jesus, Miguel, Edgar, Marcos, Ruben y todos mis compañeros de maestría, por que mejores compañeros no pude tener, todos los momentos compartidos, tareas que juntos realizamos y pláticas eternas que nos aventamos, hicieron que el estudio fuera un aventura en lugar de una labor pesada.
- ♣ A Tlapa y Cesar, por ser esos grandes amigos que todos desean y pocos tenemos la fortuna de tener.
- ♣ A mis hermanos Jufranos, por que todas la vivencias con ustedes son lecciones de vida, además de únicas.
- ♣ A los Doctores de la sección de Mecatrónica por permitirme realizar mi maestría y transmitirme su conocimiento.
- ♣ Al CINVESTAV por la oportunidad de estudiar aquí y al Consejo Nacional de Ciencia y Tecnología (CONACYT), por haberme otorgado una beca para realizar mis estudios de maestría.
- ♣ A mi tía Lucila Godínez y a toda mi familia, por su apoyo y cariño. Gracias Totales!

Resumen

Se presenta el estudio del problema de consenso en redes de agentes de primer orden considerando que en la comunicación entre los nodos existe un retardo de tiempo asociado. Estudios reportados anteriormente muestran que el retardo máximo en la comunicación que pueden soportar una red depende de la topología de la misma. En este documento se propone un esquema de control basado en la predicción de estados futuros que permite aumentar el retardo de tiempo en la comunicación, lo que significa, que no depende de la topología de la red. Se presenta el desarrollo teórico del problema, se realizan simulaciones numéricas y experimentales que muestran la efectividad de la solución propuesta.

Se realiza de forma complementaria una solución al problema de consenso en sistemas multiagente definidos por doble integrador y sujetos a retardos de tiempo debido a la comunicación. A partir de los resultados reportados en la literatura, la solución planteada se basa en el mismo esquema predictor-observador; en el que la consideración de valores futuros de los estados de cada agente, permite proponer una solución al problema con un tiempo de retardo mayor al de las aproximaciones reportadas actualmente. El resultado se demuestra formalmente, y es complementado con simulaciones numéricas, estableciendo nuevas cotas de retardo máximo que muestran a su vez el adecuado comportamiento del esquema de control propuesto.

Abstract

In this work, we discuss consensus problem for networks of first order agents considering communication time-delay. Previous works show that communication maximum delay that a network can support depends on the itself topology. A control scheme based on prediction of future states which increases communication time-delay, which means that it does not depend on the network topology, it is proposed in this research. The theoretical development of the problem is present, numerical and experimental simulations show effectiveness of the proposed solution.

Is performed in a complementary manner a solution to consensus multi-agent systems problem defined by double integrator and subject to communication time-delay. From results reported in literature, the proposed solution is based on the same predictor–observer scheme; where, consideration of future values of the states of each agent, allows proposed a solution with time-delay higher than approximations currently reported. The result is formally demonstrated, and is complemented with numerical simulations, setting new standards of time-delay that show the appropriate behavior of proposed control scheme.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Robótica Móvil

En los últimos años, el área de la robótica está alcanzado un nuevo nivel de aplicación donde las necesidades o tareas a ejecutar cada vez son más complejas tanto en habilidad como en la interacción con otros dispositivos, ambientes e incluso con seres humanos, campos donde los robots hoy en día no han alcanzado un nivel totalmente satisfactorio llegando a tener deficiencias en tareas que para los humanos son muy sencillas tales como caminar, correr, manipular objeto frágiles y de diferentes pesos, interactuar y desplazarse con múltiples objetos, agentes o en ambientes variables, entre otras acciones [1]. Sin embargo, todos estos problemas que hoy en día prevalecen, motivan a industrias, instituciones de investigación o académicas, gobiernos e incluso aficionados a estos temas a buscar soluciones viables y que además son incentivados por el constante incremento y abaratamiento en los procesadores computacionales, cámaras para visión artificial, sensores, componentes electrónicos y otras herramientas que cada vez nos acercan más al comportamiento real de las cosas. Dentro de todo este contexto, actualmente se está desarrollando el campo de estudio de la robótica móvil [2], rama que en la última década ha logrado un gran número de avances teóricos que han permitido la inclusión de ésta en el ambiente aplicado, principalmente en la industria de procesos, servicios [3] y en los últimos años en varias aplicaciones militares sobresaliendo su utilización en vigilancia [4].

La robótica móvil se considera actualmente un área de la tecnología avanzada manejadora de problemas de alta complejidad. Sus productos se constituyen en aplicaciones de las áreas de control, programación, inteligencia artificial, percepción e instrumentación, y sirven de base para el avance en diversos campos de la industria, aportando soluciones tecnológicas innovadoras orientadas al desarrollo de mejores robots y a la ampliación del abanico de aplicaciones disponibles.

El desarrollo de la robótica móvil surge fundamentalmente como respuesta a la necesidad de incrementar el área de trabajo de los robots manipuladores, los cuales, se encuentran fijos en un lugar de trabajo determinado. La motivación surge también de las diversas aplicaciones como la exploración, transporte de materiales, recopilación de información o mantenimiento de instalaciones en entornos complicados o inconvenientes para el hombre. La tendencia es a lograr la autonomía de un robot móvil, basada, por ejemplo, en un sistema de navegación; sin embargo, pueden incluirse además, tareas de planificación de trayectorias y el diseño de sistemas de control necesarios para lograr los objetivos deseados.

Una definición de robot móvil viene de la teoría clásica de robots [5]:"Un robot móvil es un vehículo de propulsión autónoma y movimiento (re) programado por medio del control automático para realizar una tarea especifica". Estos robots están equipados con ruedas, patas u orugas para realizar el movimiento autónomo, esta acción lo logra por medio de actuadores los cuales, por lo general, están controlados por una computadora [1], [6], [7].

Se trata de que el robot tenga la suficiente inteligencia como para reaccionar y tomar decisiones basándose en observaciones de su entorno, sin suponer que este entorno es perfectamente conocido. La autonomía de un robot móvil se basa en el sistema de navegación automática. En estos sistemas se incluyen tareas de planificación, percepción y control.

Tradicionalmente los robots móviles operaban en ambientes estructurados. Los sistemas se programaban para realizar siempre las mismas labores en circunstancias perfectamente definidas a priori, así por ejemplo, los vehículos iban automáticamente de un punto a otro por una trayectoria fijada con anterioridad, de la cual no podían o debían desviarse porque estaban preparados sólo para operar en esas situaciones. De acuerdo a esto, se desarrollaron metodologías y algoritmos concretos que fallaban en cuanto se presentara un evento desconocido o ajeno a lo previsto. Actualmente las tendencias son diferentes, los sistemas son autónomos, están proyectados para actuar en circunstancias diferentes de lo normal, incluso se investigan modelos de aprendizaje cada vez más sofisticados que permitan operar con el menor número de datos conocidos a priori. Para obtener información del entorno los robots se valen de los sensores, estos proporcionan un sistema de aprendizaje del estado del mundo con posibilidades de actualización continua. El uso de sensores en un sistema inteligente es de hecho la aceptación de la imposibilidad de conocer a priori el estado del mundo exterior, pues es impráctico almacenar toda la cantidad de conocimiento que ello conllevaría, y menos en un entorno cambiante de modo continuo. La facultad de los sistemas para utilizar la información procedente de múltiples sensores es uno de los factores que les habilita para tener cierta capacidad de inteligencia pues les permite interactuar con el entorno y operar en circunstancias diferentes sin el completo control de un operador humano.

Los robots móviles necesitan mecanismos de locomoción que los hagan capaces de moverse ilimitadamente en su ambiente. La manera de moverse es muy variada y por ello la selección de la forma de locomoción es un aspecto muy importante en el diseño del robot móvil. Esto hace que los robots móviles estén provistos de patas, ruedas u orugas que puedan ser programadas para desplazarse de acuerdo a su programación. Los robots móviles procesan la información que reciben a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes. También se utilizan para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos o terrestres, desactivación de bombas, en áreas radioactivas o volcánicas [8], [9], [10].

El problema básico de un robot móvil es el de navegación, es decir, moverse de un lugar a otro auxiliado de sensores, planeación y control. El problema de navegación (Figura 1.1) es simplemente encontrar una trayectoria desde un inicio (O) a un objetivo (F).

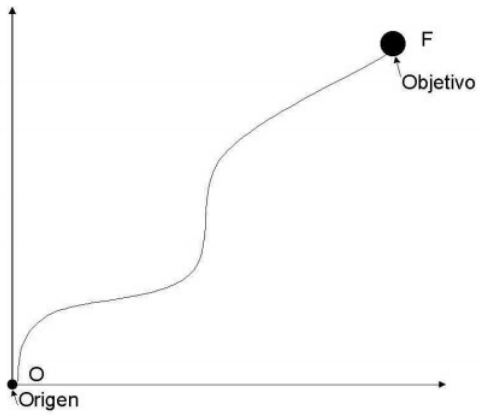


Figura 1.1: Trayectoria de un robot móvil.

1.1.1. Tipos de Robots Móviles

Los robots móviles, de acuerdo a su locomoción, se pueden clasificar en tres categorías: Robots con patas, orugas y ruedas. en este trabajo se enfocara a los robots móviles con ruedas.

La mayoría de los robots móviles con locomoción mediante ruedas pertenecen a la clase de sistemas denominados no holónomos. Los sistemas no holónomos se caracterizan por satisfacer restricciones en las velocidades que no son integrables. Una parte muy importante que se debe considerar en la construcción y diseño de robots móviles propulsados por ruedas es precisamente el tipo de ruedas a considerar. Las ruedas que se utilizan en su construcción se clasifican fundamentalmente en ruedas convencionales

y ruedas omnidireccionales. Las ruedas omnidireccionales pueden girar en cualquier sentido sin necesidad de orientarse. Las ruedas convencionales son del tipo de ruedas utilizadas en la construcción de los automóviles y sólo pueden girar en el sentido de su orientación; es decir, la componente de velocidad ortogonal al plano de la rueda es cero (restricción no holónoma). Las ruedas convencionales, a su vez, se clasifican en ruedas fijas y ruedas orientables. Las ruedas fijas son aquellas cuyo plano forma un ángulo constante con el cuerpo del robot, por el contrario, en las ruedas orientables este ángulo es variable.

Los robots móviles propulsados por ruedas se pueden dividir en cinco categorías de acuerdo al número y tipo de grados de libertad. Estos grados de libertad se dividen en dos tipos, grado de movilidad (δ_m) y grado de direccionabilidad (δ_s). El grado de movilidad del robot es un grado de libertad asociado a una variable de velocidad del robot. El grado de direccionabilidad es un grado de libertad asociado a una variable de dirección del robot. Los grados de movilidad y direccionabilidad del robot dependen de los tipos de ruedas que se utilicen en su diseño. El número total de grados de libertad de un robot móvil propulsado por ruedas es igual a la suma de los grados de movilidad y direccionabilidad. Un robot móvil práctico que se mueve sobre un plano debe tener como mínimo dos grados de libertad y como máximo tres, esto se expresa de manera matemática como,

$$2 \leq \delta_m + \delta_s \leq 3.$$

Un sólo grado de libertad en el robot solo le permitiría moverse en un círculo o sobre una línea recta ($\delta_m = 1; \delta_s = 0$) o bien no tendría propulsión y solo podríamos controlar la orientación de sus ruedas convencionales ($\delta_m = 0; \delta_s = 1$). Considerando lo anterior es posible definir tres categorías diferentes de robots de acuerdo al grado de movilidad y direccionabilidad con que cuenten. Considerando la notación ($\delta_m; \delta_s$) los robots móviles propulsados por ruedas se dividen en:

- *Tipo (3; 0)* : Robot móvil con tres ruedas omnidireccionales (ver Figura 1.2a. Las velocidades lineales u_1 y u_2 son dos grados de movilidad y la velocidad angular u_3 es el tercer grado de movilidad.

- *Tipo (2; 0)*: Robot móvil que utilizan dos o más ruedas fijas sobre el mismo eje, de las cuales sólo dos son actuadas, como se muestra en la Figura 1.2b. Sus grados de movilidad son: la velocidad lineal u_1 y la velocidad angular u_2. La direccionabilidad de este tipo de robots se logra al tener dos velocidades diferentes asociadas a dos ruedas convencionales fijas.

- *Tipo (2; 1)* : Utiliza ruedas orientables, como se muestra en la Figura 1.2c Tiene dos grados de movilidad: la velocidad lineal u_1 y la velocidad angular u_2 , y un grado de direccionabilidad que es la velocidad angular de la dirección de las ruedas orientables u_3 .

- *Tipo (1; 1)*: Consta de una o más ruedas fijas sobre el mismo eje y al menos una rueda orientable fuera del eje de las ruedas fijas, como se muestra en la Figura 1.2d. Este robot tiene un grado de movilidad que es la velocidad lineal u_1 (o v_p) y un grado de direccionabilidad que es la velocidad angular de la dirección de las ruedas orientables u_2. Nótese que este tipo de robots tienen la configuración de los automóviles convencionales.
- *Tipo (1; 2)*: Robot con al menos dos ruedas orientables, como se muestra en la Figura 1.2e. Tiene un grado de movilidad que es la velocidad lineal u_1 (ó v_p) y dos grados de direccionabilidad, cada uno de ellos asociado a la orientación de una rueda orientable.

En general, el modelo matemático de un robot móvil con ruedas es no lineal y pertenece a la clase de sistemas denominados no holónomos, es decir, se caracterizan por satisfacer restricciones no integrables que involucran las velocidades.

Un aspecto fundamental que debe considerarse en el estudio, construcción, diseño y control de robots móviles propulsados por ruedas es el tipo, disposición, configuración y características de las mismas. En cada caso, se asume que el contacto entre la rueda y el piso se reduce a un punto en el plano.

1.2. Sistemas Multiagentes

Una de las tantas aplicaciones de la robótica móvil es el estudio de coordinación distribuida de redes de agentes dinámicos, cuya atención ha recibido una mayor atención dentro de la comunidad de control en los últimos años. Esto es en gran parte a las amplias aplicaciones de sistemas multiagentes en muchas áreas incluyendo el control cooperativo de vehículos aéreos no tripulados [11] (UVA's por sus siglas en ingles), control de formación [12], [13], seguimiento de líder [14], redes de sensores distribuidos, alineación de satélites, y control de la congestión en redes de comunicación [15] (Figura 1.3).

En todos estos casos el objetivo es controlar un grupo de agentes conectados a través de una red de comunicación. Más precisamente, en lugar de controlar el movimiento de cada agente en torno a un punto de referencia dado, el objetivo es entender como hacer que los agentes se organicen y se coordinen en formaciones en movimiento. Este problema se vuelve aún más difícil en virtud de los protocolos de comunicación parciales, es decir, cuando cada agente intercambia información solo con algunos otros.

Según [16], una de las ventajas que se obtienen del comportamiento colectivo de grupos de robots es que las tareas pueden ser más complejas que las de un simple robot puede realizar. Además, diseñar y construir un grupo de robots puede ser más fácil, barato, flexible y con mayor tolerancia a fallas [17].

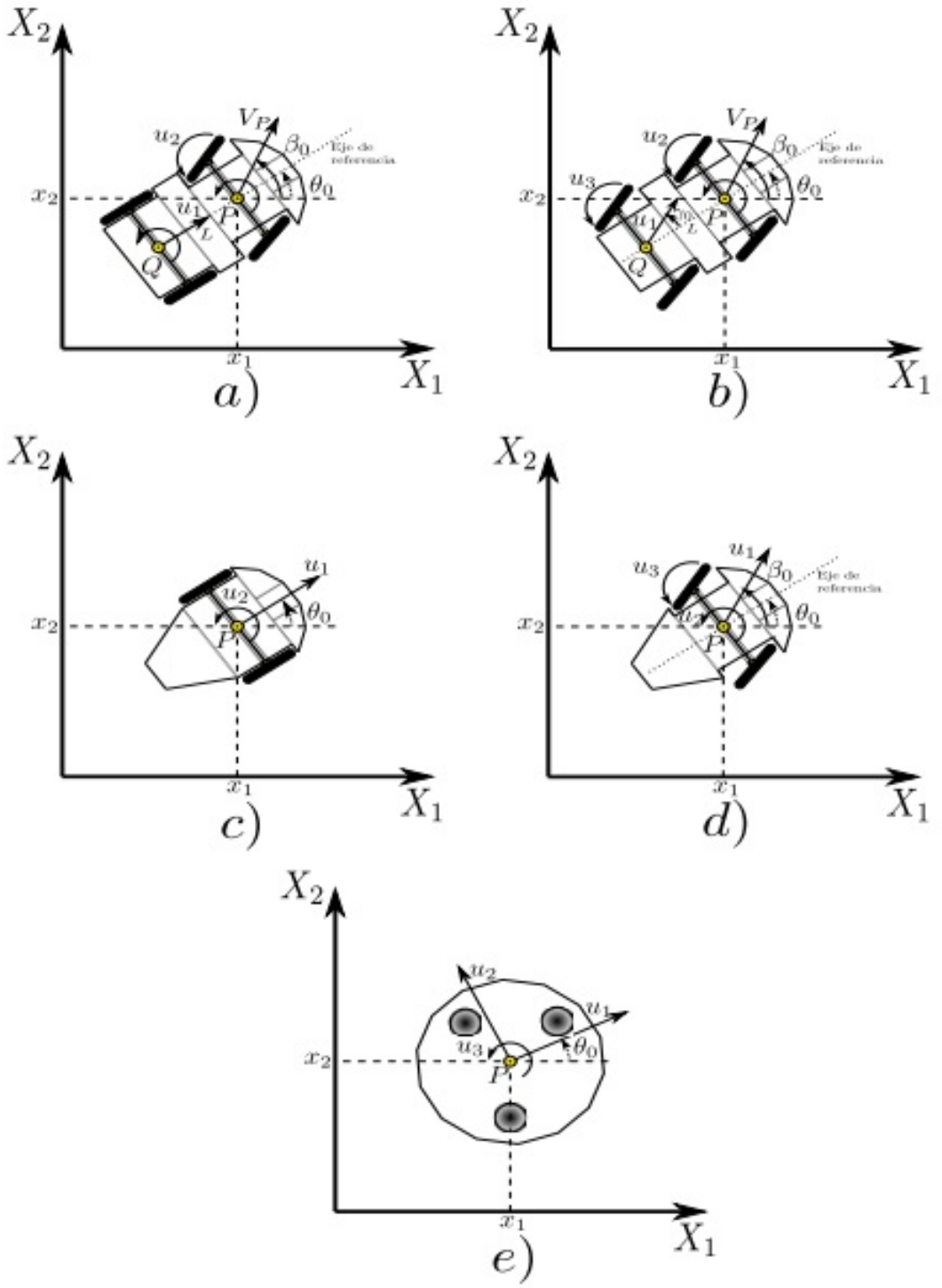


Figura 1.2: Tipos de robots móviles prácticos, a) tipo (3,0), b) tipo (2,0), c) tipo (2,1), d) tipo (1,1), e) tipo (1,2).

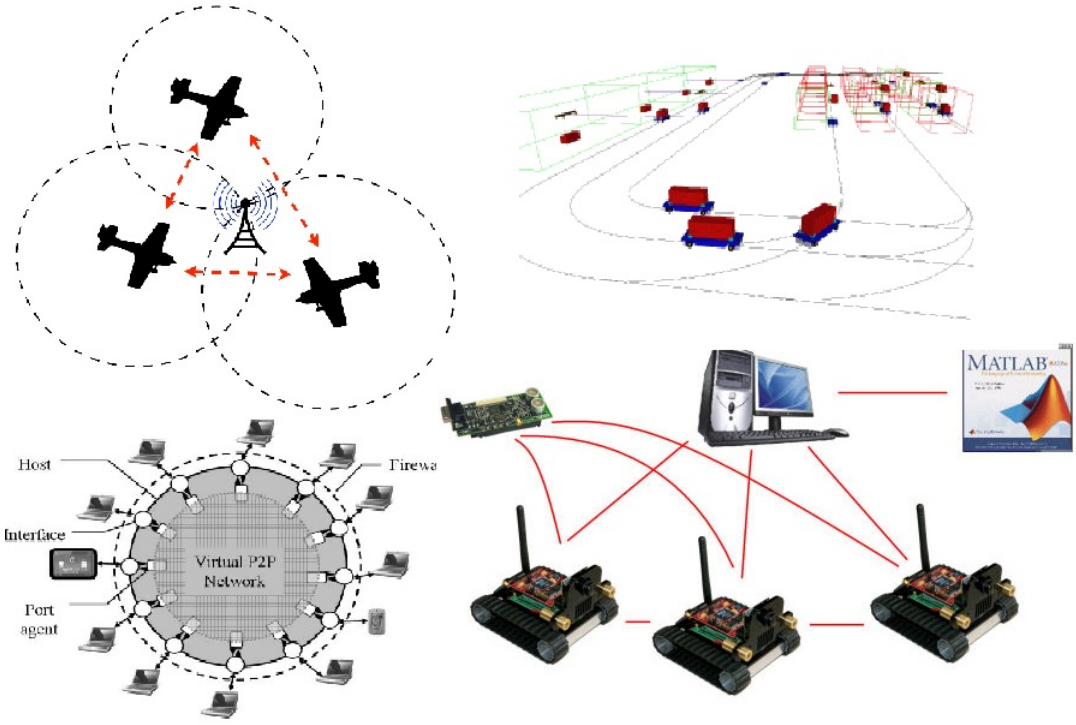


Figura 1.3: Ejemplos de sistemas multiagentes.

El comportamiento cooperativo de un sistema multiagente se define como la asociación de varios agentes para obtener un beneficio mutuo. Otra definición es la de buscar formas de interacción para realizar una tarea de manera conjunta y así lograr un resultado progresivo tal como incrementar la productividad o ahorrar tiempo.

Un problema importante que aparece con frecuencia en el contexto de coordinación de sistemas multiagentes es el convenio del grupo o problema de consenso [18]. El objetivo principal consiste en controlar y coordinar un grupo de agentes conectados a través de una red de comunicación. Este problema tiene una larga historia en el campo de la ciencia de la computación, particularmente en la teoría de autómatas y computación distribuida. En muchas aplicaciones que se relacionan con sistemas multiagentes, los elementos del grupo necesitan estar en común acuerdo con respecto a algún parámetro de interés, como una posición especifica, la velocidad de cada agente, una formación de interés entre otras opciones.

Como un ejemplo, en una red de vehículos aéreos, una forma de consenso es representada por la alineación que ocurre cuando todos los vehículos se mueven asintóticamente con la misma velocidad. En artículos anteriormente citados ([19], [20], [21], [22], [23]), el problema de consenso ha sido estudiado bajo una variedad de suposiciones en la topología de la red (fija/conmutada), el protocolo de comunicación (bidireccional o

no), diferentes requerimientos de desempeño (por ejemplo, evasión de colisión, evasión de obstáculos, etc) y el esquema de control adoptado (también denominado como protocolo de consenso).

En [23] se muestra la importancia del Laplaciano de la red, una matriz bien conocida en la teoría de grafos [24], como parte de la solución del problema de consenso cuando se considera un grupo de vehículos comunicados bajo condiciones ideales y donde las dinámicas de los agentes son lineales. Este problema también ha sido abordado por diferentes autores ([19], [18], [25], [26]) buscando garantizar que los agentes alcancen asintóticamente un consenso, es decir, que estén de acuerdo en un valor común de una cantidad de interés.

En [23] y [27], el *"Laplaciano de la gráfica"* es utilizado para el cumplimiento de la tarea de estabilización de la formación para un grupo de agentes con dinámica lineal. Este método particular para la estabilización de la formación aun no ha sido extendida a sistemas con dinámica no lineal que no son de retroalimentados linealizable. Un caso especial de este planteamiento es la conocida arquitectura *líder-seguidor* y ha sido ampliamente usada por numerosos investigadores [21], [28].

Un reto adicional que se presenta dentro de la coordinación de agentes, implica alcanzar un consenso aún ante la presencia de retardos de tiempo en la comunicación entre los agentes. Estos retardos que se encuentran presentes frecuentemente en problemas de coordinación de agentes, pueden originarse en fenómenos como el transporte de material, lazos de reciclo o bien en la aproximación de sistemas de alto orden por medio de sistemas de orden reducido con retardo de tiempo ([29]). Diversas soluciones han sido propuestas para tratar tiempos de retardo en procesos estables. En el caso lineal, cuando el tiempo de retardo afecta a la señal de entrada (o de salida) del sistema, un enfoque común es eliminar el efecto de la señal de retraso mediante una adecuada retroalimentación ([30], [31]).

El retardo de tiempos es un fenómeno frecuentemente encontrado en los sistemas dinámicos, algunos ejemplos son en reactores nucleares, procesos químicos, en procesos controlados de forma remota remota (vía Internet). Las causas de los retardos son diversas, pueden ser parte de la naturaleza de un proceso (fenómeno de transporte), o debido al tiempo de medición de una variable o una señal de transmisión. En algunos sistemas el tamaño del retardo es tal que sus efectos se pueden despreciar, sin embargo hay otros en los cuales su magnitud es tal que no es posible omitirlo. Una de las razones principales para la popularidad del tema en el medio de la investigación es que sigue siendo un problema abierto en cuanto a las condiciones necesarias y suficientes de estabilidad para este problema al cerrar el lazo de control.

1.2.1. Trabajo reportado en la literatura

La coordinación distribuida de redes de agentes dinámicos ha atraído el interés de muchos investigadores, dando pie al estudio de los diferentes comportamientos, problemas, formas de control, entre otros, de los sistemas multiagentes o multivehículos. Cada autor aborda de manera diferente o particular un problema o mejora que pueden presentar estos sistemas al ser controlados.

En el trabajo [32] se estudia protocolos de consenso lineales y no lineales para redes de agentes dinámicos que les permite llegar a un acuerdo de forma cooperativa y distribuida. Se consideran los casos con tiempos de retardo y canales que tienen efectos de filtración. Se encuentra un valor máximo de tiempo de retardo que puede ser tolerado en la red.

En [19] se discute el problema de consenso para redes de agentes dinámicos con topologías fijas o variantes. Se analizan 3 casos: 1)redes directas con topología fija; 2) redes directas con topologías variantes; y 3) redes no directas con tiempos de retardo en la comunicación y topologías fijas. Presentan dos protocolos de consenso para redes con y sin tiempos de retardo y proveen un análisis de convergencia en los tres casos.

Los autores en [33] proveen un tutorial acerca de la información de consenso para el control cooperativo de multivehículos. Resultados teóricos con respecto a la búsqueda de consenso en topologías de comunicación bajo tiempo invariante y dinámicamente variante, son resumidos en este trabajo. Se describen algunas aplicaciones específicas de algoritmos de consenso para la coordinación de multivehículos.

Un estudio del problema de consenso promedio en redes de agentes con topología variante y tiempos de retardo, se realiza en [20]. Se hace un análisis de estabilidad basada en una función propuesta de Lyapunov-Krasovskii.

Autores en [34] presentan unos resultados que sugieren que los sistemas multiagentes que interactúan a través del tiempo, con reglas de agentes vecinos mas cercanos, pueden sincronizar sus estados independientemente del valor del retardo en las comunicaciones. Realizan un análisis en tiempo discreto y se basan en las propiedades de las matrices no negativas.

En [35] se considera la estabilidad robusta y control de sistemas con tiempos inciertos de retardo. Condiciones suficientes de estabilidad son derivadas del uso del "small μ theorem". Además el control robusto de sistemas con retardos inciertos pueden ser estudiados por la combinación de su criterio de estabilidad y las técnicas estándar de la síntesis μ.

Para el caso donde se consideran agentes con dinámicas de doble integrados, en [36] se investiga el control por consenso distribuido para este tipo de sistemas. Se analizan dos tipos de redes: directas con topología fija e indirecta con topología fija pero con tiempo de retardo. Se muestran condiciones suficientes y necesarias para garantizar el

consenso. Se obtiene el tiempo mas grade de retardo que soporta el sistema, el cual esta dado por el eigenvalor mayor del Laplaciano de la topología.

Protocolos de consenso de segundo orden que tienen en cuenta los movimientos de la información de los estados, es estudiado en [18]. Se extiende los protocolos de primer orden vistos en la literatura. Se realiza un análisis de las ganancias de la ley de control, debido a su importancia para estabilidad del sistema coordinado de segundo orden.

1.3. Justificación del trabajo

La motivación principal de este trabajo es el estudio de problema de consenso implementado en sistemas multiagentes, los cuales presentan retardo en la red de comunicación, y dado el análisis en [19] y [36] para obtener el máximo valor de retardo para sistemas de primer y segundo orden, respectivamente, sin que este afecte a la estabilidad de formación, la tarea es poder aumentar dicho valor de retardo al implementar al sistema un esquema *predictor-observador*, es decir, usando este control, el retardo máximo no se debe ver limitado por el eigenvalor más grande que presenta la matriz Laplaciana de la topología de la red, sino que pueda aumentar según parámetros de diseño utilizados en el esquema predictor-observador.

1.4. Objetivos

- En este trabajo se propone abordar el problema de estimación de valores futuros de los estados para ser utilizados en el control de robots móviles sujetos a retardos de tiempo en las señales de entrada.
- Se propone un esquema predictor-observador el cual esta basado en la idea original del predictor Smith, el cual incluirá una etapa adicional de estabilización para la señal de error de estimación.
- El esquema se aplicará inicialmente en un sistema lineal (robot móvil puntual) para después considerar la representación no lineal de robots tipo uniciclo.
- Se tiene como hipótesis de trabajo que el retardo es una consecuencia de un canal de comunicación.
- Simulaciones numéricas y análisis de los errores para estabilidad.
- Programación, diseño e implementación de la plataforma experimental (Optitrack, programación en *C* y Robots García).
- Ejecución y validación de las leyes de control en la plataforma experimental.

1.5. Organización de la tesis

El resto del documento se organiza de la siguiente manera.

En el Capítulo 2, se muestra las bases teóricas para el desarrollo del presente trabajo. Se presenta una breve reseña del algoritmo de consenso, teoría de grafos y se muestran las propiedades de la matriz Laplaciana.

En el Capítulo 3, se realiza el planteamiento del problema para sistemas de primer y segundo orden, haciendo referencia a resultados con y sin retardo de tiempo. Se muestran el desarrollo de los resultados obtenidos por diferentes autores. Se realizan simulaciones como validación de los resultados de los autores.

En el Capítulo 4, se presenta el esquema predictor-observador propuesto y se detalla la solución basada en estados futuros que resuelve el problema, se muestra la estabilidad de los sistemas con la implementación del esquema predictor-observador. Se complementa con resultados numéricos y simulaciones.

En el Capítulo 5, Se reportan los resultados experimentales. Se utiliza un robot móvil tipo uniciclo (2.0) y se aplican las leyes de control propuestas. Se describe la plataforma experimental y se muestran las pruebas y resultados físicos como forma de validación del desarrollo teórico.

En el Capítulo 6 se describen las conclusiones sobre la tesis desarrollada, así como algunas perspectivas y sugerencias para el trabajo a futuro.

Finalmente se presentan los anexos, los cuales son los programas escritos en Lenguaje C++ con los que se realizaron los cálculos matemáticos para aplicar la ley de control en el sistema de agentes físicos.

Capítulo 2

Fundamentos Teóricos

2.1. Teoría de Grafos

La teoría de grafos (también llamada teoría de las gráficas) es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos [24]. Supóngase que un grupo de agentes interactua uno con otro a través de una red de comunicación o sensores o una combinación de ambos.

Un grafo $G = (N, E)$ es una colección de puntos llamados vértices N, unidos por líneas llamadas aristas E. Cada arista une dos vértices. Como ejemplo, el grafo mostrado en

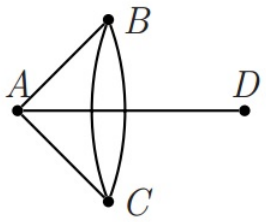


Figura 2.1: Ejemplo de Grafo.

la Figura 2.1, el conjunto de vértices está formado por los puntos

$$N = \{A, B, C, D\}$$

y las aristas

$$E = \{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, C\}\}.$$

Las aristas no tiene que ser líneas rectas, pueden ser arcos, segmentos curvos, etc. El caso en que una arista conecte a un vértice con él mismo también está permitido, y en

este caso le llamamos *lazo*. También esta permitido el caso en que más de una arista conecta los dos mismos vértices. A estas aristas se le llama aristas múltiples.

En el ejemplo anterior no hay lazos, pero sí dos aristas múltiples que unen los vértices B y C.

En algunos casos es útil que las aristas tengan alguna dirección. En este caso por ejemplo (Figura 2.2), la arista que une los puntos X y Y se pueden denotar por $\overrightarrow{\{X, Y\}}$ o solamente por (X, Y).

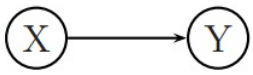


Figura 2.2: Ejemplo de Grafo con dirección.

2.1.1. Tipos de grafos y definiciones

Es natural modelar el cambio de información entre vehículos en un grafo directo o indirecto. Considere que un equipo consiste de n agentes.

Un grafo *directo o dirigido* es un par (N, E), donde $N = \{1, ..., n\}$ es un conjunto finito no vació de nodos y $E \subset N \times N$ es un conjunto de pares ordenados de nodos, llamado *arista*. La arista $(i, j) \in E$ denota que el vehículo j puede obtener información del vehículo i, pero no necesariamente viceversa. Aristas que conectan nodos consigo mismos, es decir, $(i, i) \in E$ están permitidos. Para la arista (i, j), i es el vértice o nodo *padre* y j es el vértice o nodo *hijo*.

En contraste a un grafo directo, los pares de nodos de un *grafo no directo o no dirigido* están en desorden, donde la arista (i, j) denotan que los vehículos i y j pueden obtener información uno del otro. Nótese que un grafo no directo puede ser visto como un caso especial de un grafo directo, donde una arista (i, j) en un grafo no directo corresponde a las aristas (i, j) y (j, i) en el grafo directo. La unión de una colección de grafos es un grafo cuyo conjunto de vértices y aristas son la unión de los conjuntos de vértices y aristas de los grafos en la colección.

Un *Camino directo* es una secuencia de aristas en un grafo directo de la forma $(i_1, i_2), (i_2, i_3), ...$ Un *camino no directo* en un grafo no directo es definido análogamente. En un grafo directo, un *ciclo* es un camino directo que empieza y finaliza en el mismo nodo. La misma arista (i, i) denota un ciclo de longitud 1. Un grafo directo es *fuertemente conectado* si existe un camino directo desde cada nodo a cualquier otro nodo. Un grafo no directo es *conectado* si existe un camino no directo entre cada par

de nodos distintos. Un *árbol arraigado directo* es un grafo directo en donde todos los vértices tienen exactamente un padre a excepción de un solo nodo, llamado *la raíz*, el cual no tiene padre y que tiene un camino directo hacia cualquier otro vértice. Tenga en cuenta que un árbol arraigado directo no tiene ciclo ya que cada arista está orientada lejos de la raíz. En el caso de grafos no directos, un *árbol* es un grafo en el que cada par de vértices esta conectado por exactamente un camino no dirigido.

Un *subgrafo* (N_1, E_1) de (N, E) es un grafo tal que $N_1 \subset N$ y $E_1 \subset E \cap (N_1 \times N_1)$. Un *árbol arraigado directo de expansión* (N_1, E_1) es un árbol arraigado directo y $N_1 = N$. Un *árbol no directo de expansión* de un grafo no directo es definido análogamente. El grafo (N, E) tiene o contiene un árbol arraigado directo de expansión si un árbol arraigado directo de expansión es un subgrafo de (N, E). Nótese que el grafo directo (N, E) tiene por lo menos un vértice con un camino directo a todos los demás vértices. En el caso de grafos no directos, la existencia de un árbol no directo de expansión es equivalente a estar conectado. Sin embargo, en el caso de grafos directos, la existencia de un árbol arraigado directo de expansión es una condición muy débil que el estar fuertemente conectado.

La Figura 2.3 muestra un grupo directo que contiene mas de un árbol arraigado directo de expansión, pero no es fuertemente conectado. Los vértices 1 y 2 son ambos raíces del árbol arraigado directo de expansión por lo que ambos tienen un camino directo a todos los otros vértices. Sin embargo, el grafo no es fuertemente conectado ya que cada vértice $3, 4, 5$ y 6 no tienen caminos directos a todos los otros vértices.

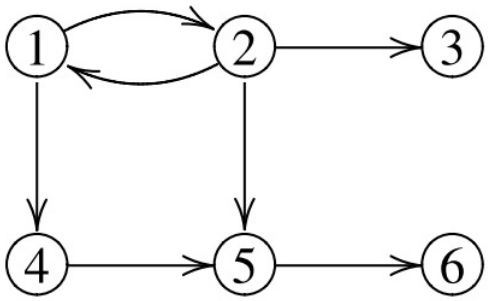


Figura 2.3: Comunicación de un grafo directo entre seis vehículos.
Una flecha desde el vértice i al vértice j indica que el vehículo j recibe información del vehículo i.

2.1.2. Teoría matricial: Matriz de Adyacencia y Laplaciana

Hay muchas matrices que están naturalmente asociadas con los grafos, tales como la *matriz de adyacencia* o la *matriz Laplaciana*. Uno de los principales problemas del

álgebra de teoría de grafos es determinar si las propiedades de los grafos son reflejadas en las propiedades algebraicas de tales matrices.

La matriz de adyacencia $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ de un grafo directo con un conjunto de vértices $N = \{1, ..., n\}$ es definida tal que a_{ij} tiene ponderación positiva si $(j, i) \in E$, mientras que $a_{ij} = 0$ si $(j, i) \notin E$. Nótese que todos los grafos están ponderados. Si la ponderación es irrelevante, entonces a_{ij} es igual a 1 para todo $(j, i) \in E$. Un grafo es *balanceado* si $\sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ji}$ para todo i. Para un grafo no dirigido, A es simétrica y por lo tanto el grafo no dirigido es balanceado.

Definamos la matriz Laplaciana $L = [l_{ij}] \in \mathbb{R}^{n \times n}$ de un grafo directo como $l_{ii} = \sum_{j \neq i} a_{ij}$ y $l_{ij} = -a_{ij}$ para todo $i \neq j$. Notar que si $(j, i) \notin E$ entonces $l_{ij} = -a_{ij} = 0$. La matriz Laplaciana satisface,

$$l_{ij} \leq 0, \ i \neq j, \qquad \sum_{j=1}^{n} l_{ij} = 0 \qquad i = 1, ..., n.$$

Propiedades de la matriz Laplaciana

- Para un grafo no directo, L es simétrica. Sin embargo, para un grafo directo, L no es necesariamente simétrica.
- En ambos casos, directo y no directo, 0 es un eigenvalor de L con el eigenvector asociado $\mathbf{1} \triangleq [1, ..., 1]^T$, el vector columna de dimensión $n \times 1$ con entradas unitarias.
- Notar que L es diagonalmente dominante y no contiene entradas negativas en su diagonal.
- Si seguimos el teorema del disco de Gershgorin ([37]) que, para un grafo no directo, todos los eigenvalores no cero de L son positivos (L es semidefinida positiva), mientras, para un grafo directo, todos los eigenvalores no cero de L tienen parte real positiva. Por lo tanto, todos los eigenvalores de $-L$ tienen parte real negativa.
- Para un grafo no directo, 0 es un simple eigenvalor de L si y solo si el grafo no directo es conectado ([38]). Para un grafo directo, 0 es un simple eigenvalor de L si el grafo directo es fuertemente conectado, aunque a la inversa no se cumple.
- Para un grafo no directo, permítase que $\lambda_1(L)$ sea el eigenvalor mas pequeño de L con $\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_n(L)$ por lo tanto $\lambda_1(L) = 0$.
- Para un grafo no directo, $\lambda_2(L)$ es la *conectividad algebraica*, la cual es positiva si y solo si el grafo no directo es conectado ([38]).
- La conectividad algebraica cuantifica la velocidad del algoritmo de consenso [39].

- Dada una matriz $S = [s_{ij}] \in \mathbb{R}^{n\times n}$, el grafo directo de S, denotado por $\Gamma(S)$ es el grafo directo con el conjunto de vértices $N = \{1, ..., n\}$ tal que hay una arista en $\Gamma(S)$ desde j hasta i si y solo si $s_{ij} \neq 0$ ([37]). En otras palabras, las entradas de la matriz de adyacencia satisface que $a_{ij} > 0$ si $S_{ij} \neq 0$ y $a_{ij} = 0$ si $s_{ij} = 0$.

2.2. Algoritmo de Consenso

Cuando múltiples vehículos acuerdan en un valor o variable de interés, se dice que ellos han llegado a un acuerdo o *Consenso*. La información de consenso garantiza que el vehículo que comparte información sobre una red, tiene una vista coherente de la información que es esencial para la tarea de coordinación. Para lograr consenso, debe de ser compartida una variable de interés, llamada *estado de información*, así como un algoritmo apropiado de negociación para alcanzar consenso en el valor o variable, llamado *algoritmo de consenso*.

Ejemplos incluyen una representación local del centro y estructura de la formación, el tiempo de encuentro, la longitud del perímetro monitoreado, la dirección del movimiento de un escuadrón multivehículos, y la probabilidad de que un objetivo militar haya sido destruido. Por necesidad, los algoritmos de consenso son diseñados para ser distribuidos, asumiendo solo interacción entre vehículos de vecino a vecino. Los vehículos actualizan su estado de información basados en los estados de información de sus vecinos. El objetivo es diseñar una ley actualizable tal que los estados de información de todos los vehículos en la red, converjan a un valor en común.

La idea básica del algoritmo de consenso es imponer una dinámica similar en la información de los estados de cada agente. Si la comunicación de la red entre los vehículos permite una comunicación continua o si el ancho de banda de comunicación es suficientemente grande, entonces la actualización de la información del estado de cada vehículo es modelada por una ecuación diferencial. Para este caso, la actualización de la información escalar de cada estado sera modelada utilizando una ecuación diferencial de primer orden.

2.2.1. Algoritmo de consenso para la dinámica de un solo integrador

La comunicación de la topología del grupo de agentes puede ser representada por un grafo directo [18]. El algoritmo de consenso en tiempo continuo más común ([19], [23], [40]) esta dado por:

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}(t)(x_i(t) - x_j(t)) \tag{2.1}$$

donde $a_{ij}(t)$ es la entrada (i, j) de la matriz de adyacencia asociada a la gráfica de comunicación en tiempo t y x_i es la información del estado de i-ésimo vehículo. Ajustando $a_{ij} = 0$ denota el hecho que el vehículo i no puede recibir información del vehículo j. Una consecuencia de (2.1) es que la información del estado $x_i(t)$ del vehículo i es dirigida hacia la información de los estados de sus vecinos.

Mientras que (2.1) asegura que la información de los estados del equipo lleguen a una acuerdo, no dicta un valor especifico en común. Por ejemplo, considere un problema de encuentro cooperativo donde la tarea de un grupo de vehículos sea arribar simultáneamente a una posición conocida específica. Ya que el tiempo de encuentro no es dado y probablemente necesitara ser ajustado en respuesta a los problemas u obstáculos en el ambiente, el equipo necesitara llegar a un consenso en el tiempo de encuentro. Para hacer esto cada vehículo primero crea una variable de información x_i que represente el valor del tiempo de encuentro del i-ésimo vehículo. Para inicializar su información del estado, cada vehículo determina un tiempo en el cual es capaz de encontrarse con el equipo y ajustar $x_i(0)$ a ese valor. Cada miembro del equipo entonces se comunica con sus vecinos y negocia un tiempo de arribo en equipo usando el algoritmo de consenso (2.1). Luego los controladores incorporados, maniobran cada vehículo para encontrarse en tiempo de arribo negociado. Cuando las condiciones ambientales cambian, individualmente cada vehículo debe resetear la información de estado y retomar el proceso de negociación.

Nótese que (2.1) no permite la especificación de una información del estado deseado. En [33] se muestra que si la comunicación en la topología es fija y las ganancias a_{ij} son invariantes en tiempo, entonces el valor común asintótico es una combinación lineal de la información inicial de los estados. En general es posible garantizar únicamente que el valor común es una combinación convexa de la información inicial de los estados.

El algoritmo de consenso (2.1) es escrito en forma matricial como,

$$\dot{x}(t) = -L(t)x(t) \tag{2.2}$$

donde $x = [x_1, ..., x_n]^T$ es la información del estado y $L(t) = [l_{ij}(t)] \in \mathbb{R}^{n\times n}$ es la Laplaciana del grafo de comunicación. El consenso es alcanzado para un equipo de vehículos si, para todo $x_i(0)$ y todo $i, j = 1, ..., n$, $|x_i(t) - x_j(t)| \to 0$ cuando $t \to \infty$.

2.2.2. Algoritmo de consenso para la dinámica doble integrador

El algoritmo de consenso (2.1) es extendido en varias formas en la literatura. Por ejemplo, en [41] una entrada externa es incorporada en (2.1) de tal forma que la información del estado sigue una entrada variante en el tiempo. En [42], condiciones necesarias y suficientes son dadas, tal que, una colección de sistemas es controlado por un equipo líder.

El algoritmo de consenso de un solo integrador dado por (2.1) ha sido extendido a la dinámica doble integrador en [43] y [44] para modelar de la manera mas natural la evolución de los fenómenos físicos, tal como un helicóptero coaxial MAV que puede ser controlado a través de maniobras suaves con un modelo de doble integrador desacoplado. Para la dinámica doble integrador el algoritmo de consenso esta dado por,

$$\ddot{x}_i = -\sum_{j=1}^{n} a_{ij}(t)\left[(x_i - x_j) + \gamma(\dot{x}_i - \dot{x}_j)\right] \tag{2.3}$$

donde $\gamma > 0$ denota la fuerza de acoplamiento entre las derivadas de los estados, y ambos x_i y $\dot{x}_i$ son transmitidos entre los miembros del equipo. Es mostrado en [43] que tanto la comunicación de la topología y la fuerza de acoplamiento γ afectan la busque de consenso en el caso general de intercambio de información directa. Para alcanzar consenso, la comunicación directa de la topología debe de tener un árbol arraigado directo de expansión y γ debe ser suficientemente grande.

Capítulo 3

Planteamiento del problema

Hasta el momento, pocos han sido los trabajos que consideran el problema de consenso cuando la comunicación es afectada por un retardo de tiempo. En particular, asumiendo que los agentes se comportan como integradores y el retardo de comunicación es constante y uniforme en el tiempo (es decir, tienen el mismo valor en todos los canales de comunicación), un análisis del retardo máximo que puede ser tolerado sin comprometer el consenso ha sido realizado en [19] y [45].

Específicamente, el protocolo adoptado por [19], para sistemas de un solo integrador, es capaz de garantizar un consenso promedio en el que los estados de cada agente convergen asintóticamente a una constante dependiente de las condiciones iniciales en lugar de una constante arbitraria. Además, los autores proveen una fórmula explícita para el retardo más grande en la transmisión, demostrando que el tiempo de retardo máximo que puede ser tolerado por una red de integradores aplicando un protocolo lineal de consenso, es inversamente proporcional al valor propio máximo de la matriz Laplaciana que presenta la topología de la red.

También se ha analizado y resuelto el problema de consenso para sistemas multiagente del tipo doble integrador cuando existen retardos de comunicación tanto en las señales de posición como de velocidad, utilizadas en la solución del problema. Se considera la descripción del problema dada en [18] y [36].

3.1. Algoritmo de consenso para sistemas de un solo integrador con tiempo de retardo

Considérese la clase de sistemas lineales una entrada-una salida de la forma

$$\dot{x}_i(t) = u_i(t) \tag{3.1}$$

para $i = 1, 2, ..., n$ y donde $x_i \in \mathbb{R}^n$ es el estado correspondiente al i-ésimo agente y $u_i \in \mathbb{R}^n$ es la entrada de control correspondiente. Defínase además, para el grupo de n agentes, $x = [x_1, ..., x_n]^T \in \mathbb{R}^n$ y $u = [u_1, ..., u_n]^T \in \mathbb{R}^n$.

El protocolo de consenso en el caso de una topología fija ([40], [19], [33]) o variante ([19]), donde no existe retardo en la comunicación está dado por,

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}(t)(x_i(t) - x_j(t)). \tag{3.2}$$

Denótese ahora, τ_{ij} al tiempo de retardo en la comunicación de la información entre el vehículo j y el vehículo i. En este caso, (3.2) se modifica en la forma,

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}(t)(x_i(t - \tau_{ij}) - x_j(t - \tau_{ij})). \tag{3.3}$$

En el caso más simple se considera que $\tau_{ij} = \tau$, es decir, el retardo en la comunicación es el mismo para todos los agentes en la red.

El protocolo de consenso para el caso sin retardo dado en la ecuación (3.2), para una red fija, es escrito en forma matricial como,

$$\dot{x}(t) = -Lx(t) \tag{3.4}$$

donde L es el Laplaciano generado por la topología considerada y definido como,

$$l_{ij} = \begin{cases} \sum_{k=1, k\neq i}^{n} a_{ik} & j = i \\ -a_{ij} & j \neq i. \end{cases}$$

El problema de consenso es resuelto para un grupo de vehículos si, para todo $x_i(0)$ y todo $i, j = 1..., n$,

$$\mid x_i(t) - x_j(t) \mid \to 0$$

cuando $t \to \infty$. La propiedad de estabilidad del sistema (3.4) depende entonces de los valores valores propios de la matriz Laplaciana generada por la topología de la red.

Cuando existe un tiempo de retardo en la comunicación, el protocolo de consenso dado en (3.3) se expresa matricialmente como,

$$\dot{x}(t) = -Lx(t - \tau) \tag{3.5}$$

donde $x \in \mathbb{R}^n$ y $x_i(t - \tau)$ es la posición del i-ésimo agente retardado τ unidades de tiempo.

El principal resultado de consenso promedio en una red con tiempo de retardo en la comunicación y topología fijo en [41], es:

Teorema 1. *Considere una red de agentes integradores con el mismo tiempo de retardo $\tau > 0$ en todos los lazos de la comunicación. Asúmase que la topología de la red G es fija, no directa y conectada. Entonces, el algoritmo (3.3) con $\tau_{ij} = \tau$, globalmente asintótica resuelve el problema de consenso promedio si y solo si cualquiera de las dos siguientes condiciones equivalentes son satisfechas.*

i) $\tau \in (o, \tau^*)$ *con* $\tau^* = \pi/2\lambda_n$, $\lambda_n = \lambda_{max}(L)$

ii) La gráfica de Nyquist de $\Gamma(s) = e^{-\tau s}/s$ *tiene un cero al rededor de* $-1/\lambda_k$, $\forall k > 1$.

Por otra parte, para $\tau = \tau^*$ el sistema tiene una solución oscilatoria global asintóticamente estable con frecuencia $w = \lambda_n$.

Basado en la parte i) del Teorema 1, una conclusión de que el límite superior en el tiempo de retardo admisible en canal de la red, es inversamente a λ_n, esto es, el eigenvalor más grande del Laplaciano del flujo de información. Se sabe por el teorema de *Gersgorin* que $\lambda_n \leq 2d_{max}(G)$ donde $d_{max}(G)$ es el máximo grado de los vértices de G. Por lo tanto una condición suficiente para el algoritmo de consenso (3.3) es,

$$\tau \leq \frac{\pi}{4d_{max}(G)}. \tag{3.6}$$

Esto significa que las redes con Vértices que tienen un alto grado no pueden tolerar tiempo de retardo en la comunicación relativamente alto.

En la otra mano, sea $\tilde{A} = kA$ con $k > 0$ la matriz de adyacencia de $\tilde{G}$. Denote el laplaciano de $\tilde{G}$ por $\tilde{L}$ y note que $\lambda_n(\tilde{L}) = k\lambda_n(L)$. Así, para cualquier retardo arbitrario $\tau > 0$, existe una $k > 0$ suficientemente pequeña tal que,

$$\tau < \pi/(2k\lambda_n). \tag{3.7}$$

3.1.1. Validación del algoritmo

Las propiedades de estabilidad de la solución propuesta en [19] para el sistema (3.1) en lazo cerrado con (3.3) se muestran a continuación mediante un ejemplo descrito en el mismo trabajo.

Ejemplo 1

Considere un sistema de 3 agentes con la configuración de red dada en la Figura 3.1.

Se considera un conjunto de agentes retardados de primer orden de la forma,

$$\dot{x}_i(t) = u_i(t - \tau) \tag{3.8}$$

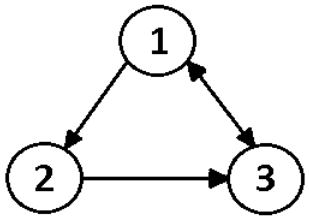


Figura 3.1: Topología de 3 agentes para ejemplo de algoritmo de consenso de un solo integrador.

para $i = 1, 2, 3$ y donde $u_i(t)$ esta formado por los valores de la matriz de adyacencia a_{ij} como se muestra en (3.3). A partir de la Figura 3.1, la matriz Laplaciana resulta,

$$L = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ -1 & -1 & 2 \end{bmatrix}$$

donde además se deduce que $\lambda_{max}(L) = 2$.

Siguiendo el análisis dado en [19], el tiempo de retardo máximo resulta,

$$\tau_{max} < \frac{\pi}{2\sigma_{max}(L)} = 0.785$$

esto es, el sistema se estabilizará para todo tiempo de retardo $\tau < 0.785\ s$.

La evolución de los estados para el sistema en lazo cerrado es mostrada en la Figura 3.2 para diferentes valores de tiempos de retardo. Se observa que el consenso es alcanzado siempre que $\tau < \tau_{max}$.

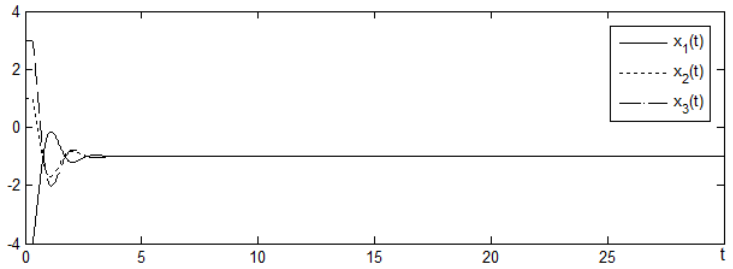
3.2. Algoritmo de consenso para sistemas doble integrador con tiempo de retardo

Considérese la clase de sistemas lineales con retardo a la entrada de la forma,

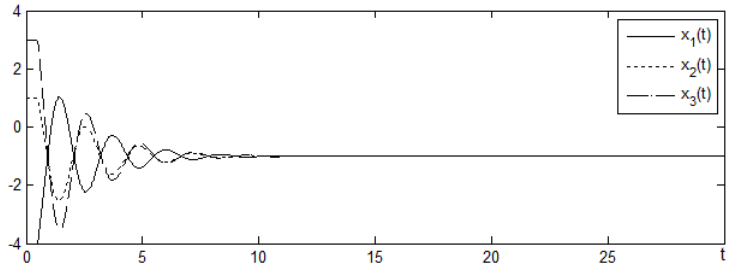
$$\begin{aligned} \dot{x}_{i_1} &= x_{i_2} \\ \dot{x}_{i_2} &= u_i(t-\tau) \end{aligned} \tag{3.9}$$

para $i = 1, 2, ..., n$ y donde x_{i_1}, $x_{i_2} \in \mathbb{R}$ son los estados del i-ésimo agente y $u_i \in \mathbb{R}$ es la entrada de control correspondiente. Además, $\tau > 0$, es el retardo de tiempo existente en la i-ésima entrada.

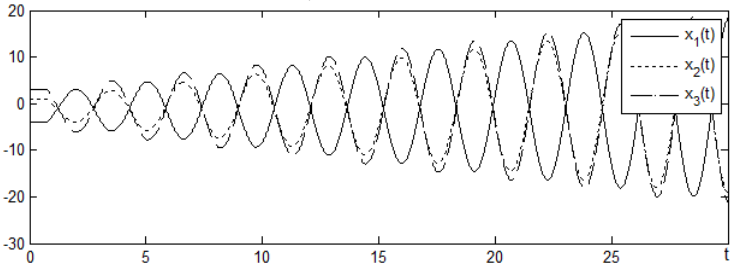
El consenso del sistema (3.9) es alcanzado si para todo $x_{i_1}(0)$ y $x_{i_2}(0)$, $x_{i_1}(t) \to x_{j_1}(t)$ y $x_{i_2}(t) \to x_{j_2}(t)$ asintóticamente cuando $t \to \infty$.



a) $\tau = 0.33$ *seg.*



b) $\tau = 0.5$ *seg.*



c) $\tau = 0.785$ *seg.*

Figura 3.2: Evolución de los estados x_i.

El sistema (3.9) puede escribirse de manera equivalente como

$$\begin{aligned} \dot{x}_1 &= x_2(t) \\ \dot{x}_2 &= u(t-\tau) \end{aligned} \tag{3.10}$$

donde

$$\begin{aligned} x_1 &= \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \end{bmatrix}^T \\ x_2 &= \begin{bmatrix} x_{12} & x_{22} & \cdots & x_{n2} \end{bmatrix}^T \\ u &= \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}^T. \end{aligned}$$

En el caso libre de retardo, $\tau = 0$, el problema de consenso es resuelto mediante la retroalimentación [18], [46],

$$u_i = -\sum_{i=0}^{n} a_{ij} \left[(x_{i_1} - x_{j_1}) + \gamma (x_{i_2} - x_{j_2}) \right] \tag{3.11}$$

cuando el sistema en lazo cerrado (3.10)–(3.11) tiene exactamente dos polos cero y todos los restantes tienen parte real negativa.

Basado en lo anterior, en el caso con retardo donde $\tau \neq 0$, una retroalimentación de la forma,

$$u_i(t) = -\sum_{j=0}^{n} a_{ij}[(x_{i_1}(t+\tau) - x_{j_1}(t+\tau)) + \gamma(x_{i_2}(t+\tau) - x_{j_2}(t+\tau))] \quad (3.12)$$

en lazo cerrado con (3.9) teóricamente resuelve el problema, sin embargo, la necesidad de tiempos futuros del estado limita su implementación.

En [36] se presenta una solución al problema considerando una variación de la retroalimentación causal (3.11) de la forma,

$$u_i(t) = \sum_{j=1}^{n} a_{ij}\left[k_1(x_{i1} - x_{j1}) + k_2(x_{i2} - x_{j2})\right]. \quad (3.13)$$

Considerando tiempos de retardo en la comunicación, la retroalimentación (3.13) se reescribe como,

$$u_i(t) = \sum_{j=1}^{n} a_{ij}\left[k_1(x_{i1}(t-\tau) - x_{j1}(t-\tau)) + k_2(x_{i2}(t-\tau) - x_{j2}(t-\tau))\right] \quad (3.14)$$

con k_1 y k_2 constantes positivas y donde la solución del problema esta determinada por un tiempo de retardo máximo τ^* tal que $\tau < \tau^*$.

Teorema 2. *[36]. Para una red directa de agentes con topología fija de comunicación de G que tiene un árbol de expansión, el algoritmo (3.13), resuelve el problema de consenso de forma globalmente asitótica si y solo si $k_2 > 0$ y $0 < k_1 < k_2^2 k_0$, donde,*

$$k_0 = min_{\|\lambda_i\| \neq 0}\left\{\frac{\|\lambda_i\|^2 \, real(\lambda_i)}{imag(\lambda_i)^2}\right\}$$

donde λ_i es el eigenvalor del Laplaciano del grafo; $real(\lambda_i)$ e $imag(\lambda_i)$ son la parte real y la imaginaria de λ_i respectivamente.

Teorema 3. *[36]. Asúmase que la comunicación de la topología G es no directa y conectada con $k_1 > 0$, $k_2 > 0$. Entonces el protocolo (3.14) resuelve el problema de consenso de forma globalmente asintótica si y solo si $\tau < \tau^*$ con τ^* dado por,*

$$\tau^* = \frac{arccos\dfrac{1}{\dfrac{k_2^2\lambda_{max}}{2k_1} + \sqrt{\left(\dfrac{k_2^2\lambda_{max}}{2k_1}\right)^2 + 1}}}{\sqrt{\dfrac{k_2^2\lambda_{max}^2}{2} + \sqrt{\left(\dfrac{k_2^2\lambda_{max}^2}{2}\right)^2 + k_1^2\lambda_{max}^2}}} \quad (3.15)$$

ó

$$\tau^* = \frac{k_2 arctan\left(\sqrt{\zeta}\right)(1-\zeta)}{k_1\sqrt{\zeta}}$$

donde λ_{max} *es el eigenvalor más grande de* L *y* ζ *es una raíz positiva de la siguiente ecuación,*

$$\zeta^3 + \left(1 + \frac{4k_1}{k_2^2\lambda_{max}}\right)\zeta^2 + \left(\frac{4k_1}{k_2^2\lambda_{max}} - 1\right)\zeta - 1 = 0. \tag{3.16}$$

Además, (3.16) tiene una y solo una raíz positiva.

Una alternativa a (3.11) y (3.14), consiste en la retroalimentación basada en estados estimados de la forma,

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}[(\hat{x}_{i1}(t+\tau) - \hat{x}_{j1}(t+\tau)) + \gamma(\hat{x}_{i2}(t+\tau) - \hat{x}_{j2}(t+\tau))] \tag{3.17}$$

donde, $\hat{x}_{ij}(t+\tau)$ representa el valor futuro estimado de $x_{ij}(t)$.

3.2.1. Validación del algoritmo

A continuación se presentan una simulación numérica para ilustrar los resultados teóricos obtenidos en [36]. La simulación esta realizada con seis agentes, con la configuración que se muestra en la Figura 3.3.

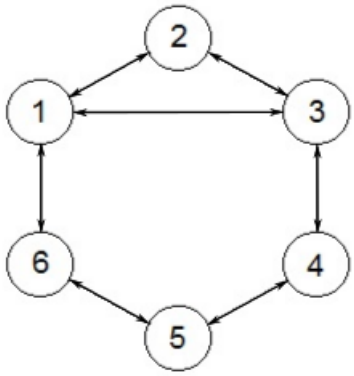
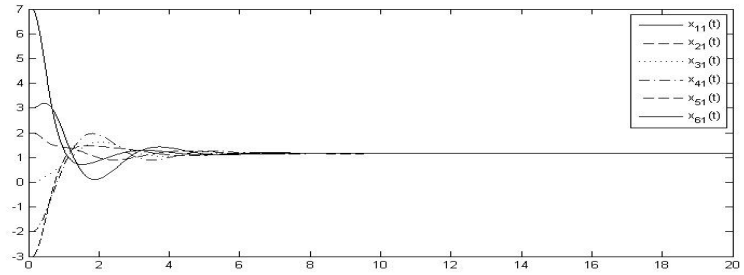
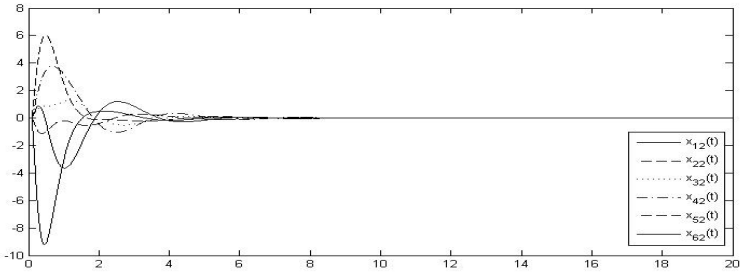


Figura 3.3: Topología de 6 agentes para ejemplo de algoritmo de consenso de doble integrador.

La simulación se llevo acabo considerando las condiciones iniciales $x_{11}(0) = 7$, $x_{21}(0) = -3$, $x_{31}(0) = 0$, $x_{41}(0) = -2$, $x_{51}(0) = 2$, $x_{61}(0) = 3$ y $x_{i2}(0) = 0$ para $i = 1, ..., 6$. Los parámetros del algoritmo que se tomaron son $k_1 = 2$ y $k_2 = 1$. Por (3.15), se obtiene que $\tau^* = 0.2456$. La Figura 3.4 y la Figura 3.5 muestran las trayectorias de los estados para $\tau < \tau^*$ y $\tau = \tau^*$, respectivamente. Ciertamente, el consenso puede ser alcanzado cuando $\tau < \tau^*$ mientras que oscilaciones síncronas son ilustradas cuando $\tau = \tau^*$.

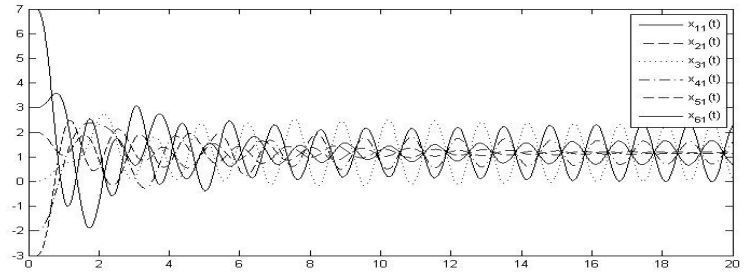


a) Información de los estados de posición x

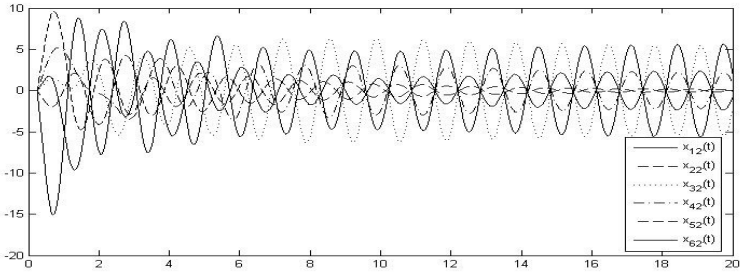


b) Información de los estados de velocidad $\dot{x}$

Figura 3.4: Respuesta de posición y velocidad con retardo $\tau = 0.1$ seg.



a) Información de los estados de posición x



b) Información de los estados de velocidad $\dot{x}$

Figura 3.5: Respuesta de posición y velocidad con retardo $\tau = \tau^* = 0.2456$ seg.

Capítulo 4

Propuesta de solución: Esquema de control Predictor-Observador

En este trabajo se ha estudiado el problema de consenso y se ha revisado algoritmos implementados por diferentes autores para redes de sistemas multiagentes con topología fija y tiempo de retardo en la comunicación. El retardo en este tipo de sistemas ha sido de gran interés para muchos investigadores debido al problema que implica, esto es, no poder cumplir la tarea de consenso cuando existe de por medio un tiempo de retardo relativamente grande en la comunicación. Hasta ahora, en la mayor parte de la literatura en la que se ha estudiado este tema en particular, se presentan criterios específicos de estabilidad para garantizar la convergencia del sistema.

Particularmente, para sistemas de un solo integrador, el algoritmo utilizado en [19] garantiza el consenso promedio aun con tiempo de retardo en la transmisión de información, teniendo como limitante un retardo máximo, cuyo valor depende del eigenvalor más grande de la matriz Laplaciana que presenta la topología de la red. De manera muy similar, para sistemas de doble integrador, en [36] se presenta un criterio para obtener el valor máximo de retardo, el cual esta ligado directamente con el eigenvalor más grande de la Laplaciana de la red.

Dado que la solución al problema de consenso para sistemas multiagentes con tiempos de retardo está ligado con la topología del sistema y no depende del diseño de una señal de retroalimentación, en este trabajo se plantea ampliar el valor del tiempo de retardo soportado por el sistema antes de inestabilizarse.

Como solución al problema se plantea un control basado en un esquema *predictor-observador* para proveer los estados futuros del sistema, adelantados τ unidades de tiempo. El esquema tiene la ventaja de que las condiciones de estabilización dependen de los parámetros de diseño y no de una topología en particular.

4.1. Solución al problema de consenso con tiempo de retardo para sistemas de un solo integrador

Para plantear la solución al problema de consenso por medio de un esquema de predicción–observación primero se mostrará cómo la dinámica de los agentes considerados, descritos en la ecuación,

$$\dot{x}_i(t) = u_i(t-\tau) \tag{4.1}$$

puede expresarse en términos de estados adelantados. Para tal efecto, considere las variables,

$$\begin{aligned} w_i(t) &= x_i(t+\tau) \\ w_i(t-\tau) &= x_i(t) \end{aligned} \tag{4.2}$$

que para el grupo de agentes producen el vector de estados, $w = [w_1, w_2, ..., w_n]^T$. Al derivar con respecto al tiempo se obtiene,

$$\dot{w}_i(t-\tau) = \dot{x}_i(t) = u_i(t-\tau)$$

donde considerando además un adelantando de tiempo τ, se produce,

$$\dot{w}_i(t) = \dot{x}_i(t+\tau) = u_i(t)$$

esto es,

$$\dot{w}(t) = u(t) \tag{4.3}$$

lo que produce un sistema en adelanto (sistema virtual), con respecto al sistema original (4.1), libre de retardo.

El sistema (4.3) se utilizará para proponer el esquema de predicción-observación que resuelva el problema de consenso por medio de la utilización de los estados estimados futuros del sistema (4.1). El esquema predictor-observador que se propone es de la forma,

$$\dot{\hat{w}}(t) = u(t) + \lambda e_w(t-\tau) \tag{4.4}$$

donde $\lambda \in \mathbb{R}^+$ es una constante de diseño del observador, $\hat{w} \in \mathbb{R}^n$ es el estado estimado, $u(t)$ es la entrada del sistema y $e_w(t-\tau)$ es el error de seguimiento, retrasado τ unidades de tiempo, definido como,

$$e_w(t-\tau) = w(t-\tau) - \hat{w}(t-\tau). \tag{4.5}$$

Considerando (4.2), el error también puede ser escrito como,

$$e_w(t-\tau) = x(t) - \hat{w}(t-\tau). \tag{4.6}$$

Nótese que en el predictor-observador (4.4), no es posible utilizar la señal $e_w(t)$ dado que,

$$e_w(t) = x(t+\tau) - \hat{x}(t+\tau)$$

lo cual requiere valores del vector de estados $x(t+\tau)$ del sistema original (4.1) en tiempos futuros que no se encuentran disponibles para su utilización.

En lugar de considerar la solución dada en [19],

$$u(t) = -Lx(t-\tau),$$

en este trabajo se considera la solución basado en estados estimados futuros de la forma,

$$u(t) = -L\hat{w}(t) = -L\hat{x}(t+\tau) \tag{4.7}$$
$$u(t-\tau) = -L\hat{w}(t-\tau) = -L\hat{x}(t)$$

que produce el sistema en lazo cerrado,

$$\dot{x}(t) = -L\hat{x}(t)$$

$$\dot{\hat{w}}(t) = -L\hat{w}(t) + \lambda e_w(t-\tau). \tag{4.8}$$

En la siguiente sección se mostrará cómo el predictor-observador dado en (4.4) produce un error de estimación de estados futuros asintóticamente convergentes y que el uso de la retroalimentación basada en estados estimados futuros (4.7), provee una solución al problema de consenso planteada en este trabajo que mejora sustancialmente al resultado obtenido en [19].

4.1.1. Análisis de estabilidad del sistema en lazo cerrado

Considérese el error de observación $e_w(t) = w(t) - \hat{w}(t)$, derivando con respecto al tiempo,

$$\dot{e}_w(t) = \dot{w}(t) - \dot{\hat{w}}(t)$$

sustituyendo (4.3) y (4.8),

$$\dot{e}(t) = u(t) - \left[L\hat{w}(t) + \lambda e_w(t-\tau)\right].$$

Se sabe por (4.7) que $u(t) = -L\hat{w}(t)$, por lo que,

$$\dot{e}(t) = -L\hat{w}(t) + L\hat{w}(t) - \lambda e_w(t-\tau).$$

Finalmente se obtiene,

$$\dot{e}(t) = -\lambda e_w(t-\tau). \tag{4.9}$$

Ahora bien, de (4.6) se tienen que $\hat{w}(t-\tau) = x(t) - e_w(t-\tau)$ y al sustituir en,

$$\dot{x}(t) = -L\hat{w}(t-\tau)$$

se tiene que,

$$\begin{aligned} \dot{x}(t) &= -L[x(t) - e_w(t-\tau)] \\ &= -Lx(t) + Le_w(t-\tau)] \end{aligned} \tag{4.10}$$

con lo cual, el sistema (4.1) en lazo cerrado con la retroalimentación (4.7) esta dado por (4.10) y (4.9) en la forma,

$$\begin{aligned} \dot{x}(t) &= -Lx(t) + Le_w(t-\tau) \\ \dot{e}_w(t) &= -\lambda e_w(t-\tau) \end{aligned} \tag{4.11}$$

donde $x = [x_1, ..., x_n]^T$, $e_w = [e_{w1}, ..., e_{wn}]^T$, $\lambda \in R^+$ y L es el Laplaciano generado por el grafo del sistema.

Nótese que un cambio de variable en la forma,

$$x = Pz, \quad z = P^{-1}x \tag{4.12}$$

produce,

$$\begin{aligned} P\dot{z} &= -LPz + Le_w(t-\tau) \\ \dot{z} &= -P^{-1}LPz + P^{-1}Le_w(t-\tau). \end{aligned}$$

Considerando a P como la matriz formada por los vectores propios generalizados de L se obtiene,

$$J = P^{-1}LP \tag{4.13}$$

donde J tiene la forma canónica de Jordan. Bajo estas condiciones el sistema (4.11) toma la forma,

$$\dot{z} = -Jz + P^{-1}Le_w(t-\tau) \tag{4.14}$$

$$\dot{e}_w(t) = -\lambda e_w(t-\tau). \tag{4.15}$$

La estabilidad de (4.15) puede establecerse fácilmente al considerar los resultados presentados en [47], donde se muestra que el sistema será estable para todo $\tau < \tau^*$.

Aplicando transformada de Laplace a (4.15) se realiza el desarrollo siguiente,

$$sE_w(s) = -\lambda e^{-s\tau}E_w(s)$$

$$\left(s + \lambda e^{-s\tau}\right) E_w(s) = 0$$

$$s + \lambda e^{-s\tau} = 0$$

considerando $s = jw$,

$$jw + \lambda e^{-jw\tau} = 0$$

$$jw + \lambda\left(Cos(w\tau) - jSen(w\tau)\right) = 0$$

$$jw + \lambda Cos(w\tau) - j\lambda Sen(w\tau) = 0$$

separando la parte real y la parte imaginaria se tiene,

$$\lambda Cos(w\tau) = 0 \tag{4.16}$$

$$w - \lambda Sen(w\tau) = 0. \tag{4.17}$$

Se eleva al cuadrado ambas expresiones y se realiza una suma entre ellas, lo cual nos da como resultado que $w = \lambda$. Ahora bien, despejamos τ de (4.17) obtenemos

$$\tau = \frac{1}{\lambda} arcSen\left(\frac{w}{\lambda}\right)$$

con $w = \lambda$ tenemos que $arcSen(\lambda/\lambda) = arcSen(1) = \pi/2$, por lo que τ esta dada por

$$\tau = \frac{\pi}{2\lambda}.$$

Se sabe que el sistema será estable siempre y cuando sus polos permanezcan del lado izquierdo del plano complejo, por lo que al considerar los polos que se encuentran exactamente sobre el eje imaginario, nos dará la condición del valor máximo del tiempo de retardo antes de inestabilizar al sistema, esto es,

$$\tau^* = \frac{\pi}{2\lambda} \tag{4.18}$$

donde λ es un parámetro de diseño que puede ajustarse a voluntad. Es posible ver que la clase del sistema (4.15) no sólo es asintóticamente estable si no que también es exponencialmente estable.

A continuación se mostrará la obtención explícita de cotas exponenciales para la clase considerada. Este hecho se utilizará posteriormente para mostrar la solución al problema de consenso planteado. Para tal efecto, considere la transformación,

$$z(t) = e^{\alpha t} e_w(t) \tag{4.19}$$

por lo que,

$$\begin{aligned} \dot{z}(t) &= \alpha e^{\alpha t} e_w(t) + e^{\alpha t} \dot{e}_w(t) \\ &= \alpha z(t) + e^{\alpha t}(-\lambda e_w(t-\tau)) \\ &= \alpha z(t) - \lambda e^{\alpha t} e^{-\alpha(t-\tau)} z(t-\tau). \end{aligned}$$

Obteniéndose entonces,

$$\dot{z}(t) = \alpha z(t) - \lambda e^{\alpha\tau} z(t-\tau). \tag{4.20}$$

Analizando la ecuación característica de (4.20), es posible determinar que este sistema será estable para todo $\tau < \tau^*$ tal que,

$$\tau^* = \frac{\arccos(\frac{\alpha}{\lambda e^{\alpha t}})}{\sqrt{(\lambda e^{\alpha\tau})^2 - \alpha^2}}.$$

En particular, considerando que la tasa de decaimiento α en (4.19) tiende a cero, se obtiene,

$$\lim_{\alpha\to 0}\tau^* = \lim_{\alpha\to 0}\frac{\arccos(\frac{\alpha}{\lambda e^{\alpha t}})}{\sqrt{(\lambda e^{\alpha\tau})^2-\alpha^2}} = \lim_{\alpha\to 0}\frac{\arccos(\frac{\alpha}{\lambda})}{\sqrt{\lambda^2-\alpha^2}}$$
$$= \frac{\arccos(0)}{\lambda} = \frac{\pi}{2\lambda}.$$

Por lo tanto, la solución de (4.15) es exponencialmente estable.

Observación 1. *Nótese que la estabilidad exponencial de (4.15) puede también establecerse a partir del Corolario 5.10, pp. 240, en [47]. Es posible entonces demostrar que el sistema será exponencialmente estable para tiempos de retardo $\bar{\tau}$, tales que,*

$$\bar{\tau} < \frac{1}{\lambda}$$

con tasa de decaimiento $\frac{\sigma}{\bar{\tau}}$, $\sigma > 0$ que satisface la ecuación trascendente,

$$-\lambda + \frac{\sigma}{\tau} + \bar{\tau}\lambda^2 e^{\frac{2\sigma}{1-\alpha}} = 0.$$

En este caso, la cota obtenida es más conservadora.

Considerando que (4.15) es exponencialmente estable, las soluciones de este sistema satisfacen ([48]),

$$\|e_w(t)\| \leq \beta e^{-\alpha t}$$

para algún $\beta \geq 1$, $\alpha \in R^+$ y para todo $t \geq 0$.

4.1.2. Valor de convergencia de posición

Considere ahora la ecuación (4.14) cuya solución puede escribirse en la forma,

$$z(t) = e^{-Jt}z(0) + \int_0^t e^{-J(t-s)}P^{-1}Le_w(s-\tau)ds. \tag{4.21}$$

Teorema 4. *(Forma canónica de Jordan) Sea A una matriz con valores propios reales λ_j, $j = 1, ..., k$ y valores propios complejos $\lambda_j = a_j + ib_j$, $\bar{\lambda}_j = a_j - ib_j$, $j = k+1, ..., n$. Entonces, existe una base $\{v_1, ..., v_k, u_{k+1}, ..., u_n, v_{n+1}, ..., v_{2n-k}\} \in R^{2n-k}$ donde v_j, $j = 1, ..., k$ y w_j,v_j $j = k+1, ..., n$ son vectores generalizados de A, $u_j = \mathrm{Re}(w_j)$ y $v_j = \mathrm{Im}(w_j)$ para $j = k+1, ..., n$, tal que la matriz $P = [v_1, ..., v_k, u_{k+1}, ..., u_n, v_{n+1}, ..., v_{2n-k}]$ es invertible y*

$$P^{-1}AP = diag\{B_i\}$$

con $i = 1, ..., r$ y donde los bloques elementales de Jordan B_j, $j = 1, ..., r$ son de la forma,

$$B = \begin{bmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix}$$

para cualquier valor propio λ real de A, o de la forma,

$$B = \begin{bmatrix} D & I_2 & & \\ & D & \ddots & \\ & & \ddots & I_2 \\ & & & D \end{bmatrix}$$

con

$$D = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}, \; I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

para cualquier valor propio complejo $\lambda = a + ib$.

Como en el caso particular de (4.13), se tiene que $v_1 = 1_n$, entonces,

$$J = P^{-1}LP = \begin{bmatrix} 0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_r \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \bar{J} \end{bmatrix}.$$

Para mostrar la solución al problema de consenso, nótese que,

$$\int_0^t e^{-J(t-s)} P^{-1} L e_w(s-\tau) ds =$$

$$\begin{aligned} &= \int_0^t e^{-J(t-s)} J P^{-1} e_w(s-\tau) ds \\ &\leq \int_0^t e^{-J(t-s)} J P^{-1} 1_n \beta e^{-\alpha(s-\tau)} ds \\ &\leq \int_0^t \begin{bmatrix} 0 \\ e^{-\bar{J}(t-s)} 1_{(n-1)} k\beta e^{-\alpha(s-\tau)} \end{bmatrix} ds \\ &= \begin{bmatrix} 0 \\ k\beta e^{\alpha\tau} e^{-\bar{J}t} \int_0^t e^{-\alpha s} e^{\bar{J}s} 1_{(n-1)} ds \end{bmatrix}. \end{aligned}$$

Nótese que como $|e_{wi}(s-\tau)| \leq \beta e^{-\alpha(s-\tau)}$ entonces siempre se satisface $e_{w_i}(s-\tau) \leq \beta e^{-\alpha(s-\tau)}$ y por lo tanto, elemento por elemento, $e_w(s-\tau) \leq 1_n \beta e^{-\alpha(s-\tau)}$. Además se ha considerado el hecho que $JP^{-1} = P^{-1}L$ y que $\|JP^{-1}\| \leq k$.

Tomando en cuenta que $x(t) = Pz(t)$ se obtiene,

$$\begin{aligned} P^{-1}x(t) &= e^{-Jt}z(0) + \int_0^t e^{-J(t-s)}P^{-1}Le_w(s-\tau)ds \\ x(t) &= Pe^{-Jt}z(0) + P\int_0^t e^{-J(t-s)}P^{-1}Le_w(s-\tau)ds \\ &= P\begin{bmatrix} 1 & 0 \\ 0 & e^{-\bar{J}t} \end{bmatrix} z(0) + P\int_0^t e^{-J(t-s)}P^{-1}Le_w(s-\tau)ds. \end{aligned}$$

Considerando ahora que,

$$\lim_{t\to\infty} \int_0^t e^{-J(t-s)}P^{-1}Le_w(s-\tau)ds = 0$$

es claro que,

$$\lim_{t\to\infty} x(t) = P\begin{bmatrix} 1 & 0 \\ 0 & 0_{(n-1)\times(n-1)} \end{bmatrix} z(0) = P\begin{bmatrix} z_1(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

y como la primera columna de la matriz P esta dada por 1_n, se obtiene,

$$\lim_{t\to\infty} x(t) = \begin{bmatrix} z_1(0) \\ \vdots \\ z_1(0) \end{bmatrix} = \begin{bmatrix} P_{1i}^{-1}x(0) \\ \vdots \\ P_{1i}^{-1}x(0) \end{bmatrix} \tag{4.22}$$

donde P_{1i}^{-1} corresponde a la primera fila de la matriz P^{-1}.

A partir de la ecuación (4.22), se muestra que los estados de los agentes $x_i(t)$ para $i = 1, ..., n$ convergen a un mismo valor dado por,

$$x_i(\infty) = z_i(0) = P_{1i}^{-1}x(0)$$

que depende de las condiciones iniciales de los agentes y representa una solución al problema de consenso.

4.1.3. Simulación y resultados

En está sección se muestran la simulación realizada que valida el funcionamiento del esquema de control predictor-observador propuesto, al ser implementado en el algoritmo de consenso en sistemas multiagentes de un solo integrador con tiempo de retardo en la comunicación.

Considérese nuevamente el sistema con la topología mostrada en la Figura 3.1, del cual (siguiendo el análisis desarrollado en [19]) se ha mostrado que el tiempo de retardo máximo soportado antes de inestabilizar al sistema, y sin la utilización del observador, está dado por $\tau_{max} = 0.785\ s$.

La simulación se realizó con la ayuda de la herramienta de computo MatLab, específicamente por medio de un sistema a bloques en Simulink (Figura 4.1). El sistema simulado es el presentado en (4.8).

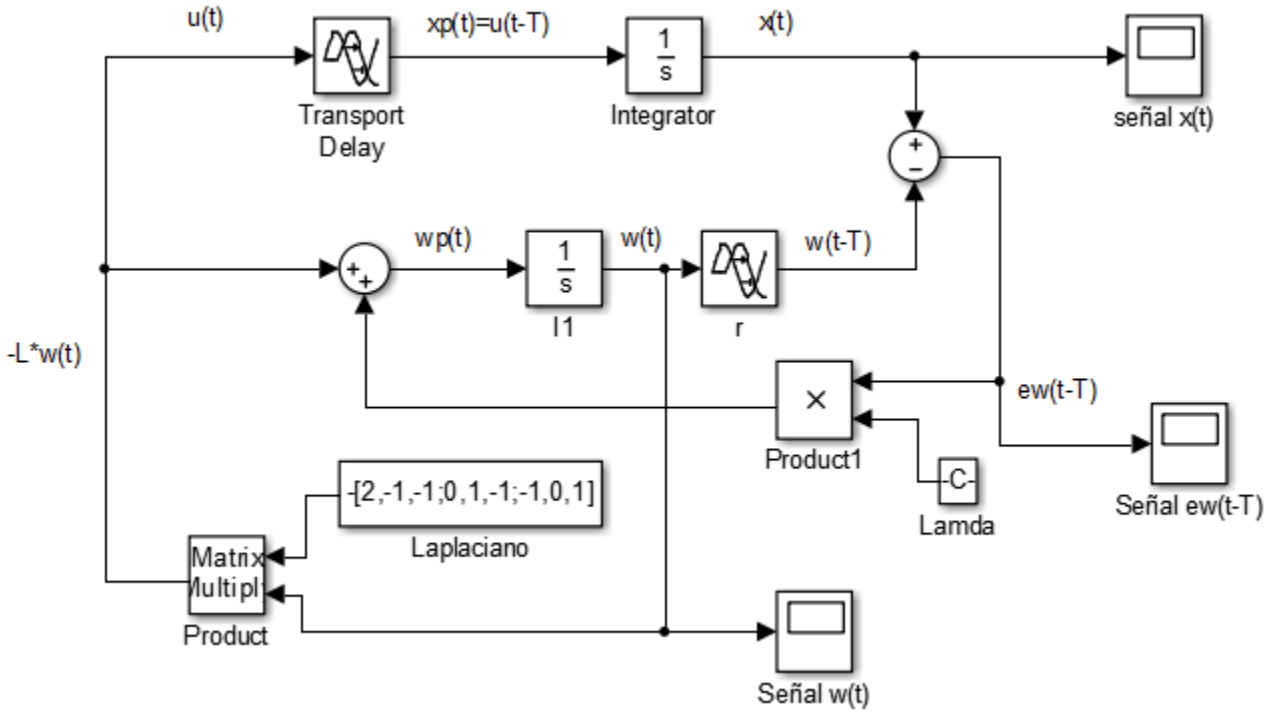


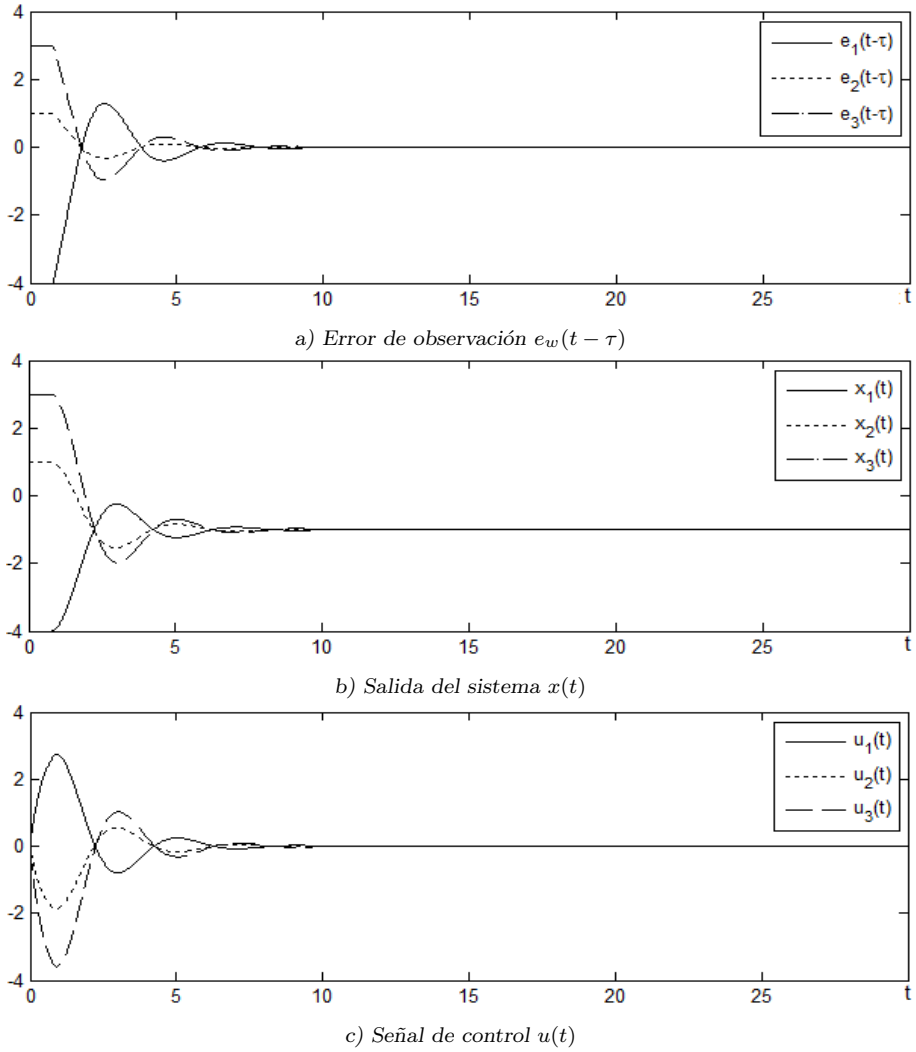
Figura 4.1: Sistema a bloques en Simulink. Algoritmo de consenso con esquema predictor-observador para sistemas de un solo integrador.

Como también se ha mostrado, la consideración de un predictor–observador como el propuesto en este trabajo permite la elección de un tiempo de retardo máximo τ_{max} que depende del parámetro λ, que puede ser ajustado a voluntad dado que es un parámetro de diseño en la ecuación (4.4).

Es fácil ver que con $\lambda = 1.0472$ se obtiene un tiempo de retardo máximo $\tau_{max} = 1.5\ s$, lo cual supera el resultado obtenido en [19]. A partir de este hecho, se desarrolla una simulación numérica para validar la estrategia propuesta, obteniéndose los resultados que se muestran enseguida.

En la Figura 4.2a se observa que el error de observación $e_w(t - \tau)$ converge a cero y que en la Figura 4.2b, los estados del sistema alcanzan el consenso, resolviendo

el problema planteado a pesar del aumento en el tiempo de retardo al límite de la solución dada en [19]. También se puede observar, en la Figura 4.2c, las señales de control, obtenidas a partir de estados estimados futuros.



a) Error de observación $e_w(t-\tau)$

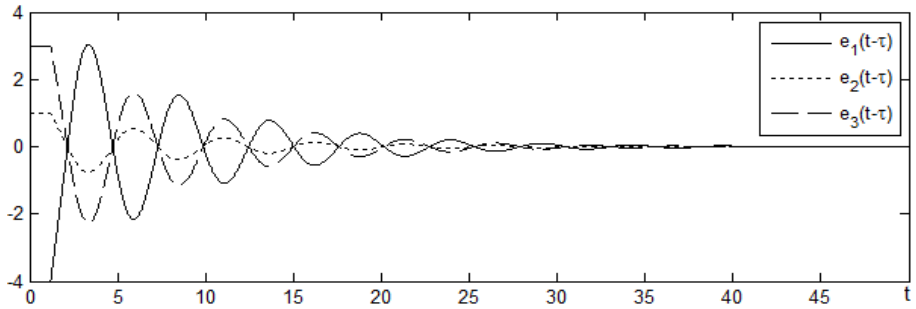
b) Salida del sistema $x(t)$

c) Señal de control $u(t)$

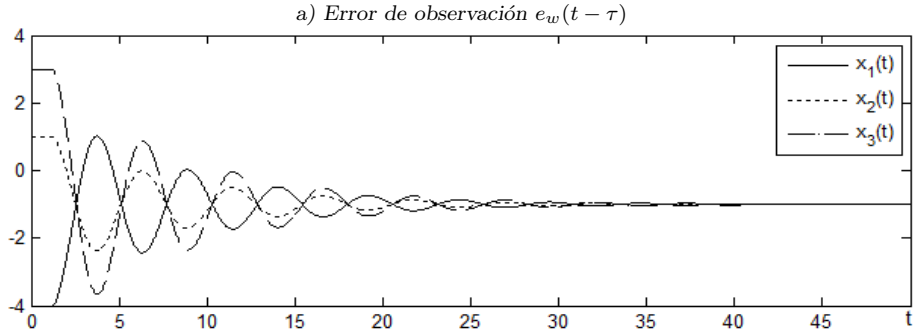
Figura 4.2: Evolución en el tiempo para el sistema de la figura 3.1 con $\tau = 0.785\ s$.

Por otra parte, en la Figura 4.3 se observa la evolución del sistema cuando el tiempo de retardo es $\tau = 1.2\ s$. Nótese que el sistema logra el consenso pero la tasa de convergencia resultante es baja, con respecto a los resultados mostrados en la Figura 4.2, dado que el retardo considerado está cerca del límite máximo permitido por los parámetros elegidos.

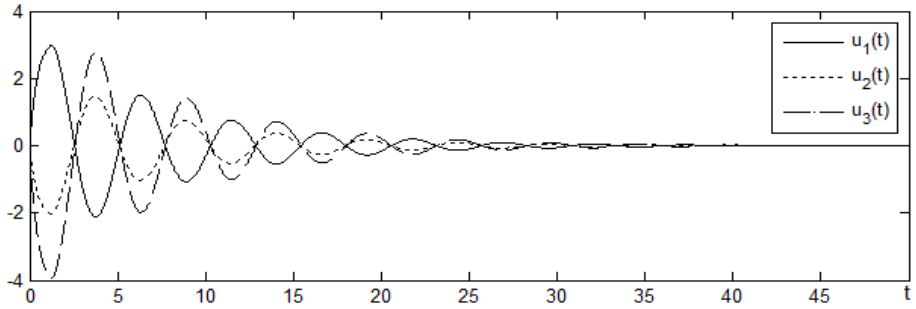
Para determinar el valor al cual convergen los estados de los agentes, es posible ver



a) Error de observación $e_w(t-\tau)$



b) Salida del sistema $x(t)$



c) Señal de control $u(t)$

Figura 4.3: Evolución en el tiempo para el sistema de la figura 3.1 con $\tau = 1.2\ s$.

que las matrices P y P^{-1} toman la forma

$$P = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -2 \\ 1 & 1 & 0 \end{bmatrix} \qquad P^{-1} = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ -0.5 & -0.25 & 0.75 \\ 0 & -0.5 & 0.5 \end{bmatrix}$$

por lo tanto, considerando la ecuación (4.22), con las condiciones iniciales $x(0) = [-4, 1, 3]^T$ y $P_{1i}^{-1} = [0.5, 0.25, 0.25]$ se tiene que,

$$P_{1i}^{-1}x(0) = \begin{bmatrix} 0.5 & 0.25 & 0.25 \end{bmatrix} \begin{bmatrix} -4 \\ 1 \\ 3 \end{bmatrix} = -1$$

lo cual concuerda con los resultados obtenidos.

4.2. Solución al problema de consenso con tiempo de retardo para sistemas doble integrador

Para el caso del doble integrador primero se mostrará cómo la dinámica del sistema

$$\begin{array}{rcl} \dot{x}_{i_1} & = & x_{i_2} \\ \dot{x}_{i_2} & = & u_i\left(t-\tau\right) \end{array}, \tag{4.23}$$

puede expresarse en términos de estados adelantados. Para tal efecto, considere las variables,

$$w_{i1}\left(t\right) = x_{i1}(t+\tau) \text{ y } w_{i2}\left(t\right) = x_{i2}\left(t+\tau\right) \tag{4.24}$$

para $i = 1, ..., n$. Bajo estas condiciones se tiene que,

$$\begin{aligned} \dot{w}_{i1}\left(t\right) &= \dot{x}_{i1}\left(t+\tau\right) = x_{i2}\left(t+\tau\right) = w_{i2}\left(t\right) \\ \dot{w}_{i2}\left(t\right) &= \dot{x}_{i2}\left(t+\tau\right) = u_i\left(t\right) \end{aligned}$$

esto es

$$\begin{array}{rcl} w_{i1}\left(t\right) & = & w_{i2}\left(t\right) \\ w_{i2}\left(t\right) & = & u_i\left(t\right). \end{array} \tag{4.25}$$

Definiendo las señales error de predicción–observación como,

$$e_{w_{i1}}\left(t\right) = w_{i1}\left(t\right) - \hat{w}_{i1}\left(t\right), \; e_{w_{i2}}\left(t\right) = w_{i2}\left(t\right) - \hat{w}_{i2}\left(t\right) \tag{4.26}$$

para $i = 1, ..., n$, se propone entonces un observador para el sistema en adelanto (4.25) de la forma,

$$\begin{aligned} \dot{\hat{w}}_{i1}\left(t\right) &= \hat{w}_{i2}\left(t\right) + \lambda_{i1} e_{w_{i1}}\left(t-\tau\right) \\ \dot{\hat{w}}_{i2}\left(t\right) &= u_i\left(t\right) + \lambda_{i0} e_{w_{i1}}\left(t-\tau\right). \end{aligned} \tag{4.27}$$

Para mostrar la convergencia de los errores de predicción, nótese que la dinámica de las señales de error toma la forma,

$$\begin{aligned} \dot{e}_{w_{i1}}\left(t\right) &= \dot{w}_{i1}\left(t\right) - \dot{\hat{w}}_{i1}\left(t\right) \\ &= w_{i2}\left(t\right) - \hat{w}_{i2}\left(t\right) - \lambda_{i1} e_{w_{i1}}\left(t-\tau\right) \\ &= e_{w_{i2}}\left(t\right) - \lambda_{i1} e_{w_{i1}}\left(t-\tau\right) \end{aligned}$$

y

$$\begin{aligned} \dot{e}_{w_{i2}}\left(t\right) &= \dot{w}_{i2}\left(t\right) - \dot{\hat{w}}_{i2}\left(t\right) \\ &= u_i\left(t\right) - u_i\left(t\right) - \lambda_{i0} e_{w_i}\left(t-\tau\right) \\ &= -\lambda_{i0} e_{w_i}\left(t-\tau\right). \end{aligned}$$

Por lo tanto,

$$\begin{aligned} \dot{e}_{w_{i1}}(t) &= e_{w_{i2}}(t) - \lambda_{i1} e_{w_{i1}}(t-\tau) \\ \dot{e}_{w_{i2}}(t) &= -\lambda_{i0} e_{w_{i1}}(t-\tau) \end{aligned} \tag{4.28}$$

para $i = 1, ..., n$, que puede reescribirse como

$$\begin{bmatrix} \dot{e}_{w_{i1}}(t) \\ \dot{e}_{w_{i2}}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_{w_{i1}}(t) \\ e_{w_{i2}}(t) \end{bmatrix} + \begin{bmatrix} -\lambda_{i1} & 0 \\ -\lambda_{i0} & 0 \end{bmatrix} \begin{bmatrix} e_{w_{i1}}(t-\tau) \\ e_{w_{i2}}(t-\tau) \end{bmatrix}. \tag{4.29}$$

Nótese que,

$$\begin{aligned} \ddot{e}_{w_{i1}}(t) &= \dot{e}_{w_{i2}} - \lambda_{i1} \dot{e}_{w_{i1}}(t-\tau) \\ &= -\lambda_{i0} e_{w_{i1}}(t-\tau) - \lambda_{i1} \dot{e}_{w_{i1}}(t-\tau) \end{aligned}$$

esto es

$$\ddot{e}_{w_{i1}}(t) + \lambda_{i1} \dot{e}_{w_{i1}}(t-\tau) + \lambda_{i0} e_{w_{i1}}(t-\tau) = 0. \tag{4.30}$$

4.2.1. Análisis de estabilidad del sistema

La estabilidad del sistema (4.30) puede establecerse por medio de un análisis directo de su ecuación característica [47],

$$s^2 + (\lambda_{i1} s + \lambda_{i0}) e^{-\tau s} = 0. \tag{4.31}$$

Para simplificar la notación de los desarrollos subsecuentes, se considerará sin pérdida de generalidad que $\lambda_{i1} = \lambda_1$, $\lambda_{i0} = \lambda_0$ para todo $i = 1, ..., n$.

Tomando en cuenta que el cruce de las raíces de (4.31) con el eje imaginario se produce para $s = jw$, se tiene que,

$$-w^2 + (\lambda_0 + j\lambda_1 w)(\cos(w\tau) - j\sin(w\tau)) = 0$$

esto es,

$$-w^2 + \lambda_0 \cos(w\tau) - j\lambda_0 \sin(w\tau) + j\lambda_1 w \cos(w\tau) + \lambda_1 w \sin(w\tau) = 0. \tag{4.32}$$

Por lo tanto,

$$-w^2 + \lambda_0 \cos(w\tau) + \lambda_1 w \sin(w\tau) = 0 \tag{4.33a}$$

$$-\lambda_0 \sin(w\tau) + \lambda_1 w \cos(w\tau) = 0. \tag{4.33b}$$

Nótese ahora que a partir de la ecuación (4.33b),

$$\tau = \frac{1}{w} \arctan\left(\frac{\lambda_1 w}{\lambda_0}\right). \tag{4.34}$$

Sustituyendo (4.34) en (4.33a) se obtiene,

$$\lambda_0 \cos\left(\arctan\left(\frac{\lambda_1 w}{\lambda_0}\right)\right) + \lambda_1 w \sin\left(\arctan\left(\frac{\lambda_1 w}{\lambda_0}\right)\right) = w^2 \tag{4.35}$$

con lo cual,

$$\lambda_0 \frac{1}{\sqrt{1+\left(\frac{\lambda_1 w}{\lambda_0}\right)^2}} + \lambda_1 w \frac{\frac{\lambda_1 w}{\lambda_0}}{\sqrt{1+\left(\frac{\lambda_1 w}{\lambda_0}\right)^2}} = w^2 \tag{4.36}$$

que a su vez produce,

$$\frac{\lambda_0^2 + \lambda_1^2 w^2}{\lambda_0 w^2} = \sqrt{1+\left(\frac{\lambda_1 w}{\lambda_0}\right)^2}.$$

Elevando al cuadrado ambos lados de la ecuación y realizando las simplificaciones correspondientes se obtiene,

$$w^6 + \left(\frac{\lambda_0^2 - \lambda_1^4}{\lambda_1^2}\right) w^4 - 2\lambda_0^2 w^2 - \frac{\lambda_0^4}{\lambda_1^2} = 0. \tag{4.37}$$

Considerando ahora el nuevo cambio de variable,

$$z = w^2$$

la ecuación 4.37 se reescribe en la forma,

$$z^3 + \left(\frac{\lambda_0^2 - \lambda_1^4}{\lambda_1^2}\right) z^2 - 2\lambda_0^2 z - \frac{\lambda_0^4}{\lambda_1^2} = 0 \tag{4.38}$$

de donde es posible obtener,

$$\left(z + \frac{\lambda_0^2}{\lambda_1^2}\right)\left(z^2 - \lambda_1^2 z - \lambda_0^2\right) = 0. \tag{4.39}$$

La ecuación anterior tiene las soluciones,

$$\begin{aligned} z_1 &= -\frac{\lambda_0^2}{\lambda_1^2} \\ z_2 &= \frac{1}{2}\lambda_1^2 + \frac{1}{2}\sqrt{4\lambda_0^2 + \lambda_1^4} \\ z_3 &= \frac{1}{2}\lambda_1^2 - \frac{1}{2}\sqrt{4\lambda_0^2 + \lambda_1^4} \end{aligned}$$

de donde es posible ver que z_2 es la única solución positiva.

A partir de los resultados anteriores, el sistema (4.28), que representa la dinámica del error de predicción, será asintóticamente estable para todo $\tau < \tau^*$ tal que,

$$\tau^* = \frac{1}{w} \arctan\left(\frac{\lambda_1 w}{\lambda_0}\right) \tag{4.40}$$

donde w esta dado por

$$w = \sqrt{\frac{1}{2}\lambda_1^2 + \frac{1}{2}\sqrt{4\lambda_0^2 + \lambda_1^4}}. \tag{4.41}$$

Estabilidad exponencial del error de predicción

Es importante resaltar que la estabilidad del sistema (4.28), no sólo es asintótica, sino exponencial. Para tal efecto considere la representación alterna dada en (4.29), y considere el cambio de variable

$$\xi_i(t) = e^{\alpha t} e_{w_i}(t), \ \alpha \in R^t. \tag{4.42}$$

Entonces, es posible escribir,

$$e_{w_i}(t) = e^{-\alpha t}\xi_i(t) \ \text{ y } \ e_{w_i}(t-\tau) = e^{-\alpha(t-\tau)}\xi_i(t-\tau) \tag{4.43}$$

y por lo tanto,

$$\begin{aligned}
\dot{\xi}_i(t) &= \alpha e^{\alpha t} e_{w_i}(t) + e^{\alpha t}\dot{e}_{w_i}(t) = \alpha e^{\alpha t} e_{w_i}(t) + e^{\alpha t}\left(A e_{w_i}(t)\right) + A_d e_{w_i}(t-\tau) \\
&= [\alpha I + A]\, e^{\alpha t} e_{w_i}(t) + e^{\alpha t} e^{\alpha(t-\tau)} e^{-\alpha(t-\tau)} A_d e_{w_i}(t-\tau) \\
&= [\alpha I - A]\, e^{\alpha t} e_{w_i}(t) + e^{\alpha\tau} A_d e^{\alpha(t-\tau)} e_{w_i}(t-\tau)
\end{aligned}$$

obteniéndose entonces,

$$\dot{\xi}_i(t) = [\alpha I - A]\,\xi_i(t) + e^{\alpha\tau} A_d \xi_i(t-\tau). \tag{4.44}$$

Nótese que si $\xi_i(t)$ es estable, entonces $e_{w_i}(t)$ es exponencialmente estable con tasa de decaimiento α.

Es evidente que cuando $\alpha \to 0$, las propiedades de estabilidad de los sistemas (4.29) y (4.44) son equivalentes. Por lo tanto, el sistema (4.29), para algún α suficientemente pequeño siempre es exponencialmente estable.

4.2.2. Solución al problema de consenso mediante estados predecidos

A partir de la definición de las variables de adelanto (4.24) y del predictor–observador definido en (4.27), es posible ver que la estimación futura de los estados de los agentes $\hat{x}_{ij}(t+\tau)$ esta definida como,

$$\hat{x}_{ij}(t+\tau) = \hat{w}_{ij}(t) \tag{4.45}$$

entonces la retroalimentación (3.12) puede escribirse como,

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}\left[\left(\hat{w}_{i1}(t) - \hat{w}_{j1}(t)\right) + \gamma\left(\hat{w}_{i2}(t) - \hat{w}_{j2}(t)\right)\right] \tag{4.46}$$

que en el caso matricial toma la forma,

$$u(t) = -L\hat{w}_1(t) - \gamma L\hat{w}_2(t) \tag{4.47}$$

donde $\hat{w}_j = [\hat{w}_{1j}, ..., \hat{w}_{nj}]$, $j = 1, 2$ y donde $L \in R^{n\times n}$ es el Laplaciano generado por la topología del grupo de agentes y está definido de la forma,

$$l_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} a_{ik} & j = i \\ -a_{ij} & j \neq i. \end{cases}$$

La retroalimentación (4.47), en lazo cerrado con el sistema (3.10), produce,

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -L\hat{w}_1(t-\tau) - \gamma L\hat{w}_2(t-\tau). \end{aligned} \tag{4.48}$$

Considerando que $\hat{w}_i(t-\tau) = \hat{x}_i(t)$, a partir de (4.26) se tiene que $e_{wj}(t-\tau) = x_j(t) - \hat{x}_j(t)$, para $j = 1, 2$, obteniéndose,

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -Lx_1(t) - \gamma Lx_2(t) + Le_{w_1}(t-\tau) - \gamma Le_{w_2}(t-\tau) \ . \end{aligned} \tag{4.49}$$

Entonces la dinámica del sistema en lazo cerrado (3.10)–(4.47) esta formada las ecuaciones (4.49) y (4.28), esto es,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -Lx_1 - \gamma Lx_2 + Le_{w_1}(t-\tau) - \gamma Le_{w_2}(t-\tau) \\ \dot{e}_{w_1} &= e_{w_2} - \lambda_1 e_{w_1}(t-\tau) \\ \dot{e}_{w_2} &= -\lambda_0 e_{w_1}(t-\tau). \end{aligned} \tag{4.50}$$

Considerando de nuevo (4.24), el sistema (4.50) puede representarse también en términos de las variables de adelanto, produciéndose en este caso, la dinámica en lazo cerrado,

$$\begin{aligned} \dot{w}_1 &= w_2 \\ \dot{w}_2 &= -Lw_1 - \gamma Lw_2 + Le_{w_1} + \gamma Le_{w_2} \end{aligned} \tag{4.51a}$$

$$\begin{aligned} \dot{e}_{w_1} &= e_{w_2} - \lambda_1 e_{w_1}(t-\tau) \\ \dot{e}_{w_2} &= -\lambda_0 e_{w_1}(t-\tau) \end{aligned} \tag{4.51b}$$

donde $w_j = [w_{1j}, ..., w_{nj}]$, $j = 1, 2$.

Tomando en cuenta que $e_w(t)$ es exponencialmente estable, se tiene que,

$$(Lw_1, w_2) = (0, 0) \tag{4.52}$$

es un punto de equilibrio para la dinámica de w.

Nótese que,

$$Lw_1 = 0$$

se satisface siempre que $w_{11} = w_{21} = ... = w_{n1}$, dado que la suma de los elementos de cada fila de L es nula.

Entonces w_{i1} converge a un valor común y w_{i2} tiende a cero resolviéndose el problema de consenso. Nótese que la convergencia de $w_{i1}(t) = x_{i1}(t+\tau)$ y $w_{i2}(t) = x_{i2}(t+\tau)$ implica la convergencia de los estados en tiempo presente $x_{i1}(t)$ y $x_{i2}(t)$.

A partir de la estabilidad exponencial del error de predicción e_w es claro también que las propiedades de estabilidad del lazo cerrado (4.51a) dependen de la dinámica del subsistema w_i y por lo tanto, puede establecerse la existencia de un principio de separación entre el diseño del predictor–observador (4.27) y el diseño de una retroalimentación estabilizante basada en estados predichos (4.46) lo que permite el diseño independiente del observador (4.27) y la retroalimentación (4.46).

Valor de convergencia de posiciones y velocidades

La dinámica del subsistema (4.51a) puede escribirse de manera alterna al considerar la transformación,

$$\begin{aligned} w_1 &= Pz_1,\ z_1 = P^{-1}w_1 \\ w_2 &= Pz_2,\ z_2 = P^{-1}w_2 \end{aligned} \tag{4.53}$$

produciéndose,

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= -P^{-1}Lz_1 - \gamma P^{-1}LPz_2 + P^{-1}Le_{w_1} + \gamma P^{-1}Le_{w_2}. \end{aligned} \tag{4.54}$$

Considerando una matriz $P \in R^{n\times n}$ formada por los vectores generalizados de L se tiene que,

$$P^{-1}LP = J_L \tag{4.55}$$

donde J_L es la forma de Jordan de L. El sistema (4.54) puede escribirse de manera equivalente en la forma,

$$\dot{z} = \Gamma_L z + B_L e_w \tag{4.56}$$

donde $z = [z_1, z_2]^T$ y

$$\Gamma_L = \begin{bmatrix} 0 & I \\ -J_L & -\gamma J_L \end{bmatrix}, \; B_L = \begin{bmatrix} 0 & 0 \\ P^{-1} & \gamma P^{-1} L \end{bmatrix}.$$

Considerando entonces que e_w tiende exponencialmente a cero, la estabilidad de (4.56) depende del sistema autónomo,

$$\dot{z} = \Gamma_L z. \tag{4.57}$$

Los valores propios de (4.57) corresponden a los valores propios del sistema,

$$\dot{w} = \Gamma w, \; \Gamma = \begin{bmatrix} 0 & I \\ -L & -\gamma L \end{bmatrix} \tag{4.58}$$

dado que el polinomio característico de (4.57) resulta,

$$\begin{aligned} P_z(s) &= \det\{sI - \Gamma_L\} \\ &= \det\{s^2 I + \gamma J_L s + J_L\} \\ &= \det\{Is^2 + (1+\gamma s) J_L\} \\ &= \det\{P^{-1}(Is^2 + (1+\gamma s) L) P\}. \end{aligned} \tag{4.59}$$

Considerando la forma de Jordan compleja de J_L con valores propios μ_i, $i = 1, \ldots, n$ se tiene que,

$$P_i(s) = \det\{sI - \Gamma_L\} = \Pi_{i=1}^{n} \left(s^2 + (1+\gamma s)\mu_i\right)$$

con lo cual cada valor propio de L produce dos valores propios de Γ_L.

Considerando entonces que el sistema (4.58) establece las propiedades de estabilidad del grupo de agentes, la solución al problema de consenso puede establecerse al considerar el resultado dado en [18] donde se trata el problema de consenso para un grupo de agente doble integradores libres de retardo, modificándolo adecuadamente para el caso con retardos tratado en este trabajo.

Lema 1. *El protocolo de consenso (4.46) basado en estados predecidos resuelve asintóticamente el problema de consenso asociado al sistema con retardo a la entrada (4.23), si y solo si, la matriz Γ tiene exactamente dos valores propios cero y todos los demás valores propios tienen parte real negativa. En particular, $w_1 \to 1_n p^T w_1(0) + t 1_n p^T w_2(0)$ y $w_2 \to 1_n p^T w_2(0)$ para t suficientemente grande, donde $p \in R^n$ es un vector propio izquierdo (no negativo) de $-L$ asociado con el valor propio 0 y $p^T 1_n = 1$.*

Como se mencionó anteriormente, la convergencia de $w_j(t)$ implica la convergencia de $x_j(t)$ debido a la relación descrita en (4.24).

4.2.3. Resultados en simulación numérica

Se considera un grupo de agentes como el mostrado en la Figura 3.3. El experimento se llevó a cabo considerando las condiciones iniciales $x_{11}(0) = 7$, $x_{21}(0) = -3$, $x_{31}(0) = 0$, $x_{41}(0) = -2$, $x_{51}(0) = 2$, $x_{61}(0) = 3$ y $x_{i2}(0) = 0$ para $i = 1, ..., 6$.

En el desarrollo del observador se consideró $\lambda_0 = 0.5$ y $\lambda_1 = 1$ mientras que para la implementación de la ley de control se fijó $\gamma = 1$. El sistema a bloques en Simulink es el mostrado en la Figura 4.4 y hace referencia al sistema presentado por (3.9) y (3.17).

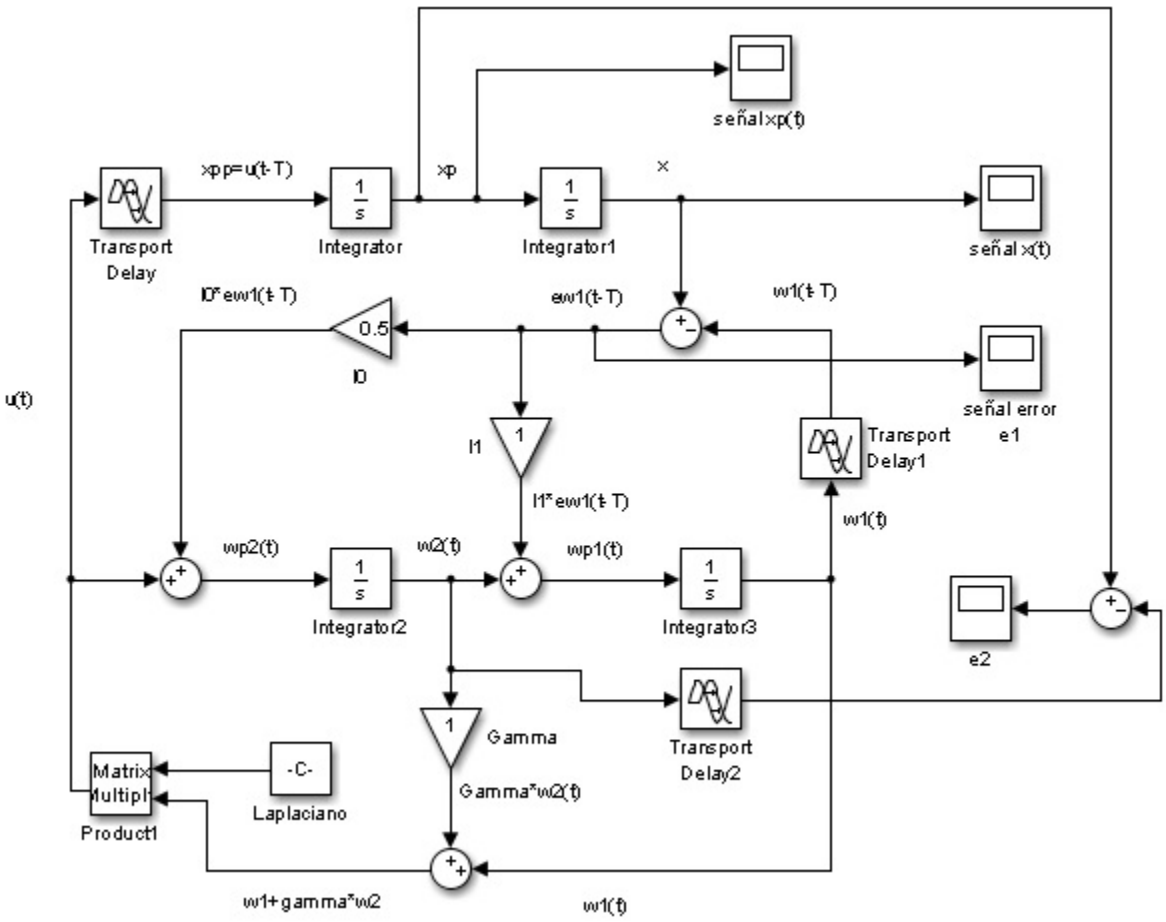
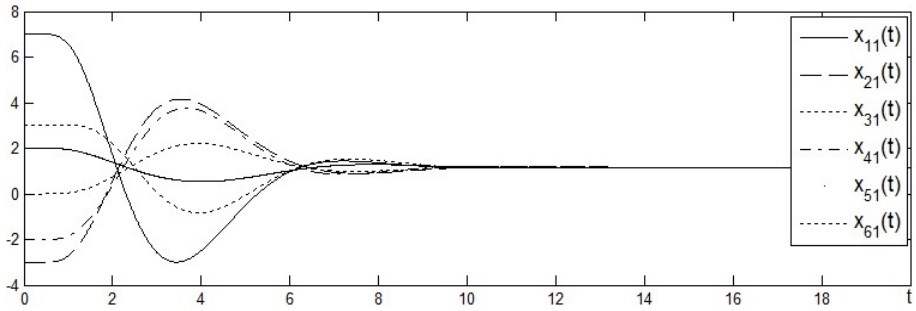


Figura 4.4: Sistema a bloques en Simulink. Algoritmo de consenso con esquema predictor-observador para sistemas doble integrador.

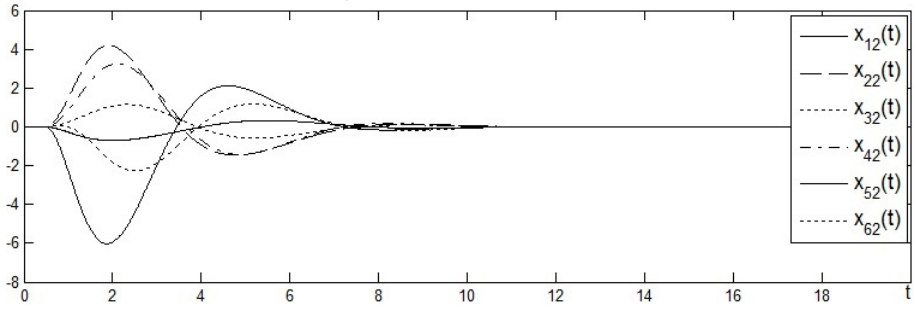
Tomando en cuenta los desarrollos presentados en [36] es posible ver que el retardo

máximo soportado por este esquema resulta $\tau^* = 0.2456$, lo que produce para este valor, una respuesta en lazo cerrado en posición x_{i1} y velocidad x_{i2} como las mostradas respectivamente en las Figuras 3.5a y 3.5b que muestran como el problema de consenso no puede ser resuelto bajo este esquema de control.

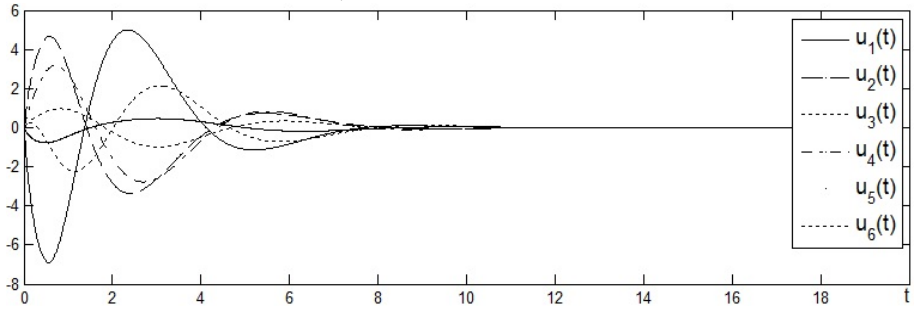
Para mostrar la superioridad del esquema de control propuesto en este trabajo, se ha considerado un retardo de tiempo $\tau = 0.5$, que no puede ser tratado con el resultado dado en [36]. Considerando el esquema basado en predicción, propuesto en este trabajo, la Figura 4.5a muestra la evolución temporal de la posición x_{i1} y la Figura 4.5b muestra la velocidad x_{i2} del conjunto de agentes. Es claro que se obtiene una solución al problema de consenso planteado. Los valores de las señales de control son mostradas en la Figura 4.5c.



a) Respuesta en posición.



b) Respuesta en velocidad.



c) Respuesta de las señales de control.

Figura 4.5: Resultados de simulación con la implementación del esquema de control predictor-observador cuando $\tau = 0.5\ s$.

Capítulo 5

Resultados Experimentales

5.1. Plataforma experimental

Esta sección esta dedicada a la descripción de los experimentos que se realizaron para poder validar el desarrollo teórico que se describió en los capítulos anteriores. Se discuten algunas consideraciones que se tomaron en cuenta para la implementación correcta de la ley de control basada en el esquema predictor-observador, en robots móviles de tipo 2.0, cuya configuración presenta restricciones no holónomas las cuales deben ser consideradas.

La plataforma experimental consta de tres sistemas principales:

- Sistema de visión.
- Robots Garcia tipo 2.0.
- Sistema de computo (software).

En general la plataforma consiste en un sistema de visión que detecta desde uno hasta varios objetos rígidos, de los cuales se puede obtener su ubicación, así como su orientación con respecto a un sistema coordenado; la información obtenida de los cuerpos rígidos es enviada a un sistema de computo donde se procesa y se realizan los cálculos matemáticos correspondientes a la ley de control; los resultados son interpretados y reflejados en un comportamiento físico de los cuerpos rígidos y finalmente son resensados por el sistema de visión para el siguiente ciclo de trabajo.

Un bosquejo de la plataforma experimental que ha sido usada para llevar a cabo la parte experimental y la validación de la metodología de control propuesta es mostrado en la Figura 5.1. A continuación se describirá brevemente cada uno de los elementos que la componen.

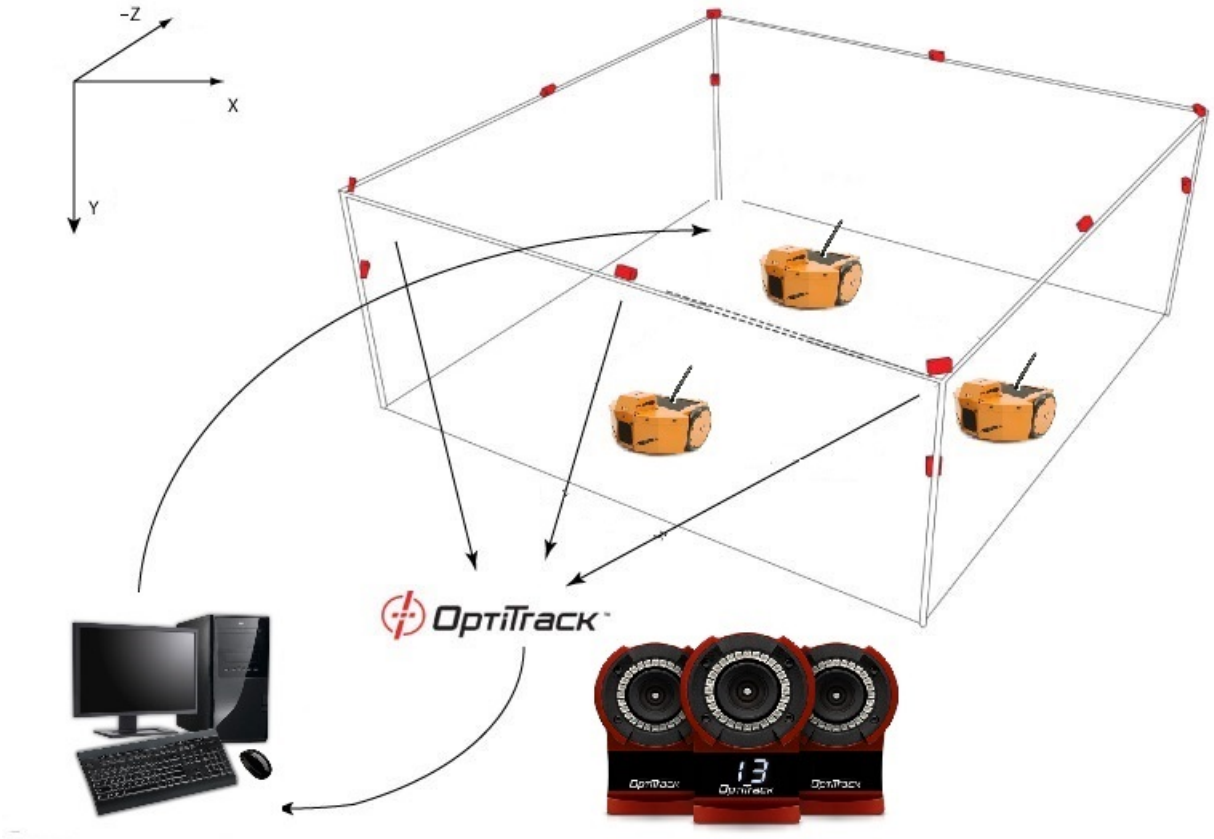


Figura 5.1: Plataforma experimental compuesta por sistema de visión (Optitrack), sistema de computo y robots Garcia.

5.1.1. Sistema de visión

La visión artificial se define como el procedimiento de adquisición de imágenes, sin contacto y mediante sistemas ópticos, donde se desea tomar una decisión con base en el análisis automático de las mismas, a fin de controlar un proceso o actividad. En los últimos años, los sistemas de visión han tenido un gran desarrollo industrial con aplicaciones en la industria automotriz, química y de alimentos, en el sector farmacéutico, así como en los campos de la electrónica, la robótica, etc. Entre las aplicaciones típicas de visión encontramos: la detección y ausencia de componentes, metrología, control de procesos, clasificación, control de calidad, monitorización, etc.

La detección de la posición y orientación de los robots son parte fundamental para el desarrollo de los experimentos y cálculos matemáticos, por tal motivo deben de ser obtenidos de la manera más precisa posible y así lograr una adecuada comparación entre la parte teórica y experimental. Para lograr este objetivo se cuenta con el sistema de visión y sensado especializado en la obtención de captura de movimiento vía señales

infrarrojas auxiliares OptiTrack, sistema desarrollado por la compañía NaturalPoint [49] que determina la posición y orientación de objetos en un espacio tridimensional limitado por el rango de trabajo de la constelación de cámaras.

El sistema de visión Optitrack implementado esta conformado por:

- 12 cámaras Flex 13, las cuales capturan el movimiento en tiempo real con gran resolución, trabajan a un máximo de 120 cuadros por segundo, cuentan con un microprocesador que puede realizar no solo el procesamiento de imagines, sino que realiza cálculos de visión artificial de los datos de posición de un conjunto de marcadores reflejantes filtrados.

- La conexión y recopilación de datos de las 12 cámaras Flex 13 esta a cargo de 2 concentradores repetidores de paquetes de datos, los cuales son conocidos como "*Hubs*". El sistema cuenta con múltiples configuraciones de conexión en red y amplificación de la señal de los datos hasta por un rango de 5 metros bajo la restricción de tener un adecuado cableado.

- La plataforma experimental utiliza una serie de marcadores reflejantes de señales infrarrojas configurados en patrones geométricos irregulares con la finalidad de distinguir a los objetos rígidos a sensar. Cada patrón (formado por esferas de plástico de material reflejante de señales infrarrojas) debe ser único y se debe de tener la característica de que si un marcador de un patrón dado es perdido por las cámaras, los restantes marcadores no generen un patrón similar al de otra configuración, ya que el sistema no distinguiría entre cuerpos rígidos al tener el mimos patrón geométrico. Cada configuración es sensada calculando el centroide la figura, para después trasladar este punto hacia el punto medio del eje de las llantas del robot (punto a controlar). De este punto trasladado se obtienen los datos necesarios para el controlador: posición y su orientación. Los marcadores tiene una medida de 7/16" lo que implica que pueden ser detectados a una distancia máxima de 28 pies con una cámara Flex 13.

- Motive es un programa creado por la compañía NaturalPoint que sirve como interfaz de control para el sistema OptiTrack. La interfaz esta diseñada para aplicar una gran gama de configuraciones del sistema, las cuales se extienden principalmente al sensado de cuerpos rígidos, sensado facial y sensado de cuerpos variables. Para los experimentos sólo se utilizarán las herramientas incluidas en Motive: Tracker para el sensado de cuerpos rígidos móviles, identificados por configuraciones geométricas irregulares de marcadores reflejantes de tamaño 7/16".

- Red Periférica de Realidad Virtual (Virtual Reality Peripheral Network - VRPN) es un conjunto de clases en una librería y un conjunto de servidores que son diseñados para implementar una interfaz transparente en red entre programas de aplicaciones y el conjunto físico de dispositivos (sensores, botones, cámaras,

etc.) usados en el sistema de realidad virtual. El sistema en sí, funciona teniendo una PC o cualquier otro host en cada estación de realidad virtual que controle los periféricos (rastreador (tracker), botones, dispositivos hápticos, entradas analógicas, sonido, etc.).

5.1.2. Sistema de computo

Se cuenta con una computadora en la cual se llevan a cabo todos los cálculos necesarios para la generación de la ley de control propuesta; en esta se reciben los valores de las posiciones y orientaciones de los cuerpos rígidos sensados por el Optitrack, los cuales se toman y procesan, para después enviar los datos a los robot Garcia, para que puedan alcanzar consenso.

Se uso el lenguaje de programación C++ para realizar los cálculos matemáticos y procesamiento de señales. La representación del diagrama a bloques del programa realizado se muestra en la Figura 5.2 y los programas realizados se muestran en el Apéndice A.

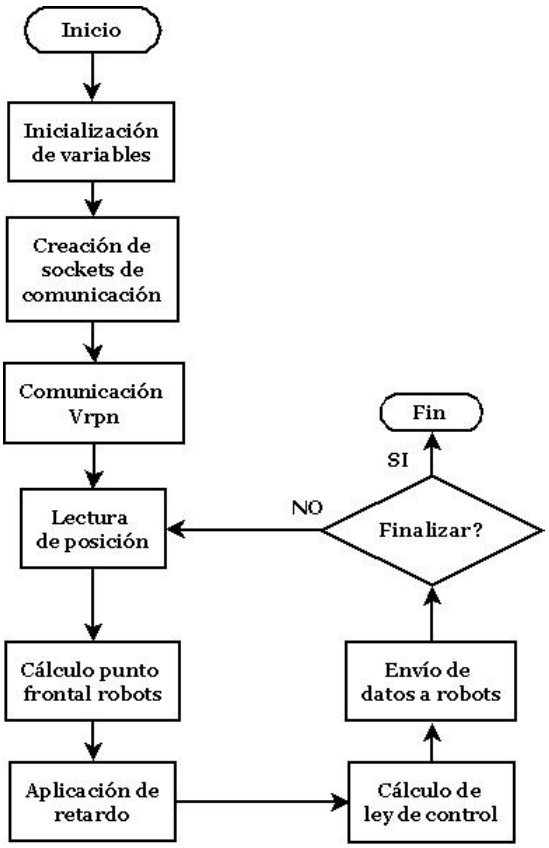


Figura 5.2: Diagrama a bloques del programa realizado para el control.

La secuencia de programación es la siguiente:

- Se inicializan las variables a utilizar.
- Se crean los sockets de comunicación de los robots Garcia.
- Se establece la conexión computadora-vrpn-optitrack y se declaran los nombres de los agentes, los cuales tiene que coincidir con las etiquetas puestas en el software *Motive*.
- Se realiza una sección para la creación de un archivo donde se guardarán los datos obtenidos en el programa para poder graficar los resultados.
- Se inicializan los parámetros de los robots Garcia, para que estos inicien sin movimiento.
- Se leen los valores de posición y orientación de los robots Garcia, obtenidos en tiempo real del Optitrack.
- Debido al control cinemático de los robots Garcia, se calcula un punto frontal de este.
- Se crea un retardo virtual (programado) y se aplica al sistema por medio de la entrada de control.
- Se realiza el cálculo de la ley de control donde están inmersos el algoritmo de consenso y el esquema predictor-observador.
- Se envían los datos calculados de las nuevas posiciones a las que deben de moverse los robots.
- Nuevamente se leen las posiciones de los robots y se vuelve a realizar el reproceso del cálculo de la ley de control.
- El programa finaliza hasta que se presione *"Esc"*.
- Finalmente se cierran los sockets de conexión de los robots Garcia.

5.1.3. Robot tipo 2.0 (Robot Garcia)

Los robots tipo uniciclo o (2,0) utilizados como agentes para la validación y experimentación de los desarrollos teóricos anteriores está a cargo de Robots García (Figura 5.3) fabricados por la compañía *Acroname Inc.*

Este robot es una plataforma de tracción diferencial canónica que se centra en la simplicidad, maniobrabilidad y control. Puede avanzar fácilmente sobre pisos de baldosas, alfombras de pelo corto, suelos de madera y pueden ser manejados fácilmente

Figura 5.3: Robots Garcias utilizados como agentes.

sobre azulejos. Superficies duras y planas son los más adecuados para su funcionamiento fiable.

Los robots Garcia cuentan con una gran cantidad de particularidades que permiten utilizarlos en una gran gama de aplicaciones y ambientes controlados. Los principales componentes que conforman el cuerpo del robot son: Cuenta con dos procesadores independientes de 40Mhz para las funciones del robot; la estructura del robot esta manufacturada en aluminio rígido, todas las tuercas son de seguridad con hilo de nylon para evitar aflojamientos y pérdidas; es alimentado por una batería estándar de 6 celdas $7.2V$ $5200mAh$ NiMH; utiliza dos motores Maxon con un encoder (codificador) óptico de cuadratura con una resolución de 16 cpr (cuentas por revolución) y una caja de engranes planetarios con una relación $19:1$; cuenta con 6 sensores infrarrojos con un rango de 4 a 8 pulgadas. Además cuenta con dos sensores debajo el robot que sensan la proximidad con el piso, para evitar caídas por salientes, escalones o finales del plano; cuentan con comunicación USB hacia otros dispositivos y conexión a redes inalámbrica; y están diseñados para manejar una carga útil moderada de máximo $2\ Kg$, la adición de la carga útil se compensa durante la operación ya que el robot utiliza un control PID para los motores. Las dimensiones de los Robots Garcia se muestran en la Figura 5.4.

Configuración de los Robots Garcia

Se cuenta con dos robots que se comunican vía WiFi, y uno que se comunica por Bluetooth. Para el caso de la comunicación bluetooth, solo se establece la tasa de baudios (BaudRate=38400)a la cual debe de enviarse la comunicación y vincular el robot con la computadora, la cual asigna un puerto de comunicación serial virtual al robot. Los agentes de comunicación WiFi generan una red local denominada *"Garcia"* la cual tiene que ser configurada de tal forma que los agentes tengan una misma dirección IP con la computadora de control. Los robots con WiFi se comunican por medio de los puertos 8000 y 8001 y en la dirección IP 192.168.2.2 y 192.168.2.3 respectivamente. La dirección IP para la computadora es 192.168.2.1.

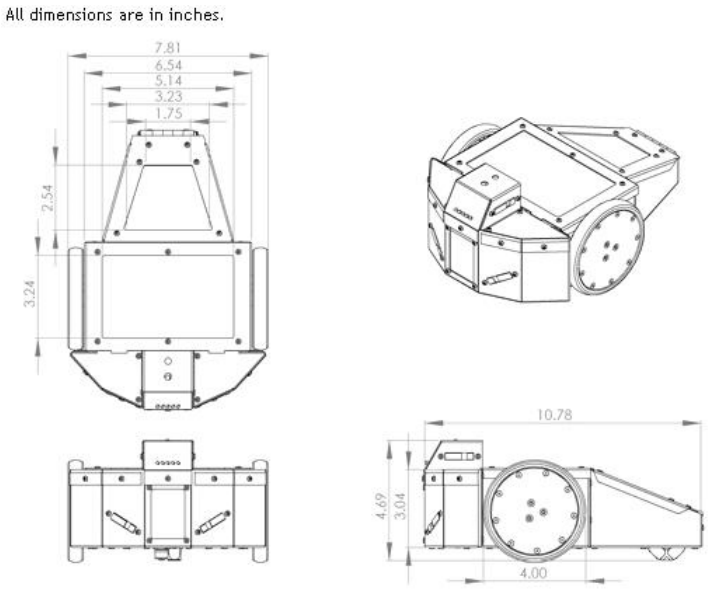


Figura 5.4: Dimensiones de un Robot Garcia.

El tipo de control sobre estos robots es con respecto a las velocidades de cada una de las llantas, por lo que la empresa *Acroname* determinó que la forma de control es por medio de un buffer de comunicación de 12 bytes, el cual facilita, para el usuario, la forma de enviar la información al robot. La descripción de cada byte se muestra a continuación:

- Para cada llanta se le es ha asignado 6 bytes, esto es, los primeros 6 son para una llanta y los restantes para la otra.
- Los 3 primeros bytes son reservados para el control PID de los motores, es decir, son para la ganancias del control.
- El siguiente byte es para determinar que motor controlar, izquierdo o derecho. Los valores correspondientes a este byte es 0 y 1, 0 para motor el izquierdo y 1 para el motor derecho.
- El quito byte determina el sentido de giro, horario y antihorario. El valor 0 corresponde al sentido horario y con 255 el giro es antihorario.
- El sexto determina la velocidad de la llanta.

Los dos últimos puntos son muy importantes al momento de controlar al robot Garcia y se les tiene que prestar atención, debido a que presentan un particular comportamiento, ya que se le deben enviar los valores correctos para el funcionamiento adecuado del robot. Primero consideremos que el sentido horario de las llantas realizará el movimiento de retroceso del robot, mientras que el sentido antihorario corresponderá al movimiento hacia adelante. Los valores que pueden ser escritos en el byte de velocidad van de

0–255, los valores de 0 a 127 corresponden al sentido de giro horario, siendo 0 la velocidad mas baja y 127 la velocidad máxima. De 128 a 255 corresponden al sentido antihorario, el valor 128 es la velocidad máxima y 255 es la velocidad mínima en ese sentido. Existe una zona de seguridad creada por los diseñadores del robot Garcia, por lo que los valores de velocidad aproximada máxima son 100 para el sentido horario y 170 para el sentido antihorario.

Estos valores se obtuvieron al realizar pruebas de velocidad con los robots, se enviaban valores de velocidad a las llantas y se leía el valor de velocidad real (Tabla 5.1 y Figura 5.1.3), lo cual nos pudo dar como resultado que el incremento de velocidad se comportaba de manera lineal, se obtuvieron las ecuaciones de la rectas que caracterizan la relación entre los valores de velocidad en el byte y la velocidad real a la que va el robot.

Se sabe que la ecuación de la recta es $y = xm + b$, x representa la valores de velocidad que se envía al Garcia, y las velocidades en m/s a la que avanza el robot, m corresponde a la pendiente de la recta y b es el valor por el cual cruza la linea sobre eje de las ordenadas. Para la ecuación de la línea que caracteriza la velocidad del sentido horario $m = 0.01$ y $b = 0.02$, mientras que para la ecuación de la línea para el sentido antihorario $m = -0.01$ y $b = 2.614$.

Sentido Antihorario		Sentido Horario	
Valor en el Byte	Velocidad Real (m/s)	Valor en el Byte	Velocidad Real (m/s)
10	0.1085	245	0.1185
20	0.2137	235	0.2254
30	0.3194	225	0.3315
40	0.4352	215	0.4356
50	0.5237	205	0.5404
60	0.6243	195	0.6413
70	0.7151	185	0.7359
80	0.8138	175	0.8325
90	0.908	165	0.9328

Cuadro 5.1: Valores experimetales para la caracterización de la velocidad de los robots Garcia

Como resultado de estas pruebas, se pudo concluir que la velocidad máxima del robot Garcia es de 1 m/s aproximadamente.

Modelo Cinemático del robot tipo 2.0 (Uniciclo)

La cinemática, se centra en el estudio del movimiento del robot en función de su geometría. Entre las aplicaciones inmediatas se encuentran la posibilidad de utilizarlo

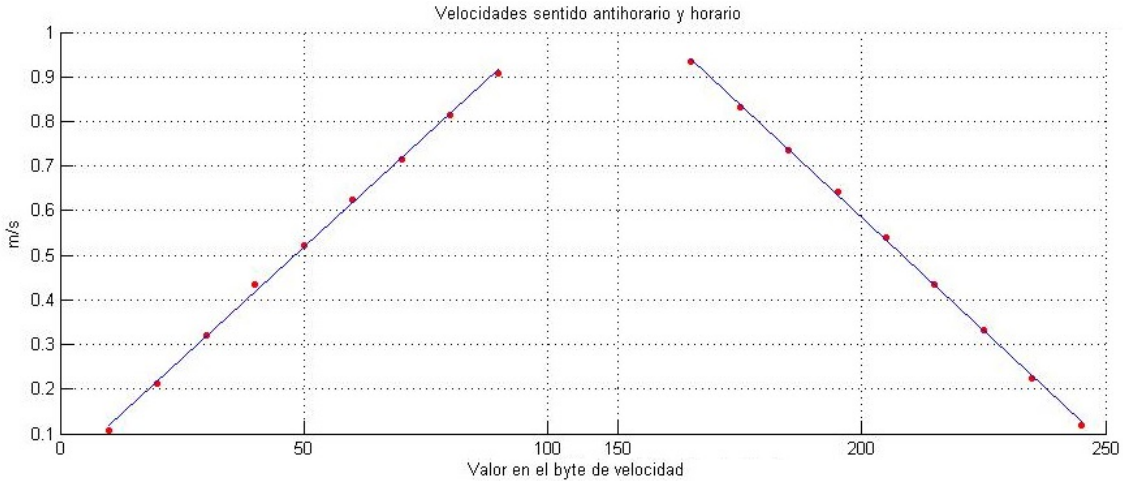


Figura 5.5: Gráfica de caracterización de las velocidades con respecto al los valores puestos en el byte de velocidad del robot Garcia.

como modelo matemático de partida para el diseño del controlador. Los robots móviles del tipo uniciclo se construyen utilizando dos ruedas fijas sobre el mismo eje con motores independientes. No utilizan ruedas orientables pero cuentan con una o más ruedas locas que proveen estabilidad. Su movimiento se logra al aplicar una velocidad diferente en cada una de las ruedas. Su grado de movilidad es dos y su direccionabilidad es cero.

Normalmente, se consideran las siguientes suposiciones para la construcción del modelo cinemático del robot tipo uniciclo:

- El robot se mueve sobre una superficie plana.
- No existen elementos flexibles en la estructura del robot (incluidas las ruedas).

Bajo estas suposiciones considere a $N = \{R_1, ..., R_n\}$ como un conjunto de robots móviles tipo uniciclo que se mueven en el plano con coordenadas $z_i(t) = [x_i(t), y_i(t)]^T$, $i = 1, ..., n$. El modelo cinemático de cada robot de acuerdo a la Figura 5.1.3 esta dado por,

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} v_i cos(\theta_i) \\ v_i sin(\theta_i) \\ w_i \end{bmatrix}. \tag{5.1}$$

La cual puede ser representada de forma vectorial de la siguiente forma,

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} cos(\theta_i) & 0 \\ sin(\theta_i) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{5.2}$$

donde v_i y w_i son las velocidades lineal y angular, respectivamente, del punto medio del eje de las ruedas del i-ésimo robot. El sistemas (5.2) es no lineal entonces no puede ser estabilizado por leyes de control lineal continua e invariante en el tiempo, como es el caso del esquema de control propuesto en este trabajo. Por esta razón, se estudia la cinemática de un punto $\alpha_i = (p_i, q_i)^T$ fuera del eje de las ruedas del robot. Las coordenadas del punto α_i están dadas por,

$$\alpha_i = \begin{bmatrix} p_i \\ q_i \end{bmatrix} = \begin{bmatrix} x_i + lcos(\theta_i) \\ y_i + lsin(\theta_i) \end{bmatrix} \tag{5.3}$$

donde l es la distancia del centro del eje de las ruedas al punto frontal que se desea controlar y la cinemática de las nuevas coordenadas queda definida de la siguiente forma,

$$\dot{\alpha}_i = \begin{bmatrix} \dot{x}_i - \dot{\theta}_i lsin(\theta)_i \\ \dot{y}_i + \dot{\theta}_i lcos(\theta)_i \end{bmatrix}. \tag{5.4}$$

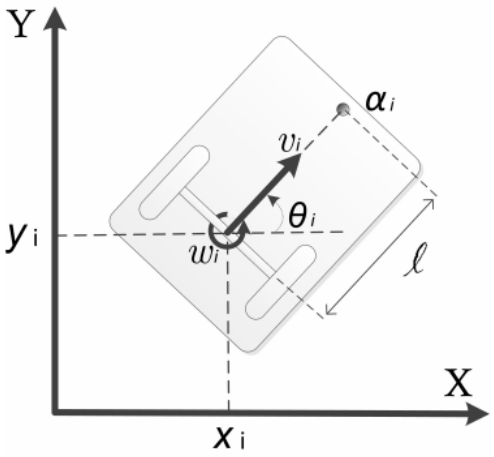


Figura 5.6: Modelo cinemático del robot tipo uniciclo.

Sustituyendo la ecuación (5.1) en (5.3) obtenemos,

$$\dot{\alpha}_i = A_i(\theta_i) \begin{bmatrix} v_i \\ w_i \end{bmatrix} \tag{5.5}$$

donde,

$$A_i(\theta_i) = \begin{bmatrix} cos(\theta_i) & -lsin(\theta_i) \\ sin(\theta_i) & lcos(\theta_i) \end{bmatrix} \tag{5.6}$$

A_i es llamada matriz de desacoplamiento, la cual es no singular dado que $det(A - i(\theta_i)) = l \neq 0$, con su inversa dado por,

$$A_i^{-1}(\theta_i) = \begin{bmatrix} cos(\theta_i) & sin(\theta_i) \\ -\dfrac{sin(\theta_i)}{l} & \dfrac{cos(\theta_i)}{l} \end{bmatrix}. \tag{5.7}$$

Esto permite el control de posición del punto α_i para robots tipo uniciclo. Definiendo una prealimentación inicial con una nueva señal de control $u_i = [u_{i1}, u_{i2}]^T$, tenemos,

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = A_i^{-1}(\theta_i) \begin{bmatrix} u_{i1} \\ u_{i2} \end{bmatrix}. \tag{5.8}$$

El sistema en lazo cerrado (5.5)–(5.9) produce entonces,

$$\begin{bmatrix} \dot{\alpha}_{i1} \\ \dot{\alpha}_{i2} \end{bmatrix} = \begin{bmatrix} u_{i1} \\ u_{i2} \end{bmatrix} \tag{5.9}$$

esto es,

$$\dot{\alpha}_i = u_i. \tag{5.10}$$

5.2. Algoritmo de consenso y estrategia de formación

El esquema de control propuesto en este trabajo se presenta considerando que se implementa sobre sistemas puntuales, y el objetivo de consenso consiste en que todos los vehículos de la red lleguen a un punto en común, en la parte experimental no se puede considerar que los robots lleguen a un mismos punto ya que estos tienen una dimensión definida, y se sabe por las leyes de la física que dos cuerpos no pueden ocupar el mismo espacio, y el intentar realizar dicha acción provocaría una colisión entre los robots. Una solución a este problema es considerar una estrategia de formación de agentes en el algoritmo de consenso, la cual se presenta en esta apartado.

Considérese el protocolo de consenso para el caso puntual mostrado en (3.2), la estrategia de formación debe considerar un conjunto de vectores que determinan la distancia a la cual deben de estar los agentes. Se trata del conjunto de distanciamiento $C = \{c_{ij} = R_i - R_j\}$ con $R_i, R_j \in E,\ c_{ji} \in \mathbb{R}^2,\ j \in V_i$, donde V_i es el conjunto de agentes que tienen un lazo de comunicación con el i-ésimo agente, se le llama *Conjunto adyacente*.

Considere la topología mostrada en la Figura 5.2.

- El conjunto de vehículos es $N = \{R_1, R_2, R_3\}$.
- El conjunto de aristas $E = \{(R_2R_1), (R_3R_2), (R_1R_3)(R_2R_3)\}$.
- EL conjunto adyacente de cada robot es:

$$\begin{aligned} V_1 &= \{R_2\} \\ V_2 &= \{R_3\} \\ V_3 &= \{R_1, R_2\}. \end{aligned}$$

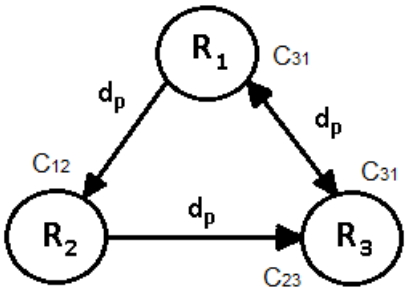


Figura 5.7: Modelo cinemático del robot tipo uniciclo.

- Finalmente los vectores de posición deseados son los siguientes,

$$\begin{aligned} c_{21} &= [-d_p cos(\theta), d_p sin(\theta)] \\ c_{32} &= [d_p, 0] \\ c_{13} &= [-d_p cos(\theta), -d_p sin(\theta)] \\ c_{32} &= [-d_p, 0]\,. \end{aligned}$$

El vector,

$$c_{ij} = \left[\frac{h_{ij}}{v_{ij}}\right]$$

denota la posición relativa deseada del agente i con respecto al agente j. h_{ij} es la componente horizontal del vector y v_{ij} es la componente vertical del vector.

Al agregar el vector de posición a (3.2) se obtiene el protocolo de consenso con estrategia de formación, como sigue,

$$u_i(t) = -\sum_{j=1}^{n} \left[a_{ij}(t)(x_i(t-\tau) - x_j(t-\tau) - c_{ij})\right]. \tag{5.11}$$

5.2.1. Implementación del esquema predictor-observador en el algoritmo de consenso y estrategia de formación, con retardo en la comunicación, para el control de los robots Garcia

Como se vio anteriormente, el control de los robots uniciclos por el punto medio del punto frontal, representa un sistema no lineal, el cual se puede linealizar por la retroalimentación (5.8) como se ve en (5.9). Con este resultado, ahora considere un retardo fijo en la entrada de control $u(t-\tau)$, lo que hace que el sistema (5.10) se escriba de la forma,

$$\dot{\alpha}_i = u_i(t-\tau) \tag{5.12}$$

por lo tanto, el algoritmo de consenso resulta

$$u_i(t-\tau) = -\sum_{j=1}^{n} \left[a_{ij}(t)(\alpha_i(t-\tau_{ij}) - \alpha_j(t-\tau_{ij}) - c_{ij}) \right]. \tag{5.13}$$

Escribiendo el sistema (5.12) en términos de estados adelantados y adelantando un tiempo τ, es fácil ver,

$$\begin{aligned} \dot{w}_{i1}(t) &= u_{i1}(t) \\ \dot{w}_{i2}(t) &= u_{i2}(t). \end{aligned} \tag{5.14}$$

Predictor–Observador en lazo abierto

Considerando el nuevo punto de control α, el esquema predictor–observador se escribe basado en este punto como sigue,

$$\dot{\hat{w}}_i = u(t) + \lambda e_w(t-\tau) \tag{5.15}$$

el error de estimación es,

$$e_w(t-\tau) = \alpha_i - w_i(t-\tau)$$

y finalmente reescribiendo el esquema de predicción,

$$\dot{\hat{w}}_i = -\sum_{j=1}^{n} \left[a_{ij}(t)(\alpha_i(t) - \alpha_j(t) - c_{ij}) \right] + \lambda(\alpha_i(t) - w_i(t-\tau)).$$

Predictor–Observador en lazo cerrado con el sistema

Considere la solución basada en estados estimados futuros, es decir, la entrada de control depende de estos estados estimados $u_{\hat{w}}$

$$u_{\hat{w}}(t) = -\sum_{j=1}^{n} \left[a_{ij}(t)(\hat{w}_i(t) - \hat{w}_j(t) - c_{ij}) \right]. \tag{5.16}$$

por lo tanto el observador es,

$$\dot{\hat{w}}_i = u_{\hat{w}}(t) + \lambda e_{\hat{w}}(t-\tau) \tag{5.17}$$

y el esquema de predicción es,

$$\dot{\hat{w}}_i = -\sum_{j=1}^{n} \left[a_{ij}(t)(\hat{w}_i(t) - \hat{w}_j(t) - c_{ij}) \right] + \lambda(\alpha_i(t) - \hat{w}(t-\tau)). \tag{5.18}$$

5.3. Experimentos realizados

Se utilizaron 3 robots Garcia para la parte experimental. Los experimentos que se han realizado para validar la solución al problema de consenso con esquema predictor-observador son los siguientes:

- Algoritmo de consenso con retardo sin esquema de predicción. Se implementó el algoritmo de consenso con retardo en la comunicación desarrollado en la literatura ([19]).
- Algoritmo de consenso con retardo con esquema de predicción. Finalmente se comprueba de manera experimental el esquema predictor-observador que da solución al problema de consenso para sistemas multiagentes con tiempo de retardo en la comunicación.

5.3.1. Experimentos del Algoritmo de consenso con retardo sin esquema de predicción

Se tiene el algoritmo de consenso con formación y retardo, descrito en (5.11), el cual es aplicado en este experimento para el sistema (5.12).

Tomemos (5.8), la cual nos representa las velocidades lineales y angulares a enviar a los robots Garcia, y cuyos valores dependerán de la solución del algoritmo de consenso, por lo tanto se tiene

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = A_i^{-1}(\theta_i)u_i(t-\tau).$$

esto es

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = A_i^{-1}(\theta_i)\left(-\sum_{j=1}^{n}\left[a_{ij}(\alpha_i(t-\tau)-\alpha_j(t-\tau)-c_{ij})\right]\right).$$

Dada la configuración de transmisión de información de la topología ejemplificada, se procede a escribir las ecuaciones de las velocidades lineales y angulares de cada agente. Llamemos a los robots Garcia con comunicación WiFi, *robot 2* y *robot 3*, y al robot con bluetooth, *robot b*, por lo cual para distinguir el modelo y ecuaciones de cada robot, se colocará un subíndice 2, 3 y b, que represente al robot 2, al robot 3 y al robot b, respectivamente.

Para el robot b se tiene que

$$\begin{bmatrix} v_b \\ \omega_b \end{bmatrix} = A_b^{-1}(\theta_b)\left\{-k\left(\alpha_1(t-\tau)-\alpha_3(t-\tau)-c_{31}\right)\right\},$$

para el robot 2

$$\begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix} = A_2^{-1}(\theta_2)\left\{-k\left(\alpha_2(t-\tau)-\alpha_1(t-\tau)-c_{12}\right)\right\},$$

y finalmente para el robot 3

$$\begin{bmatrix} v_3 \\ \omega_3 \end{bmatrix} = A_3^{-1}(\theta_3)\left\{-k\left(2\alpha_3(t-\tau)-\alpha_1(t-\tau)-\alpha_2(t-\tau)-c_{13}-c_{23}\right)\right\}.$$

Obsérvese la introducción de una constante k, la cual es una ganancia que se utiliza para el correcto funcionamiento y convergencia del sistema. La ganancia no solo afecta la convergencia sino que, como se observa con (3.7), modifica el valor de retardo máximo que soporta el sistema, k debe de considerarse pequeña.

Con las consideraciones anteriores, ahora se muestran las resultados obtenidos al realizar el algoritmo de consenso con los tres robots Garcia. Se consideró utilizar el sistema de agentes con la topología de comunicación mostrada en la Figura 3.1, la cual también se utilizó como ejemplo en validación teórica del algoritmo de consenso descrita en la literatura existente.

Se sabe que el valor de retardo máximo de esta topología en particular es de $\tau = 0.785\ s$, considerando la ganancia k a la cual se le designa un valor constante $k = 0.3$, se recalcula el valor de retardo máximo. Con la expresión (3.7) se tiene que $\tau_{max} = 2.61799\ s$, por lo tanto, los resultados están basados en mostrar la estabilidad del sistema antes de este valor de retardo. Las condiciones iniciales de las posiciones de los robots entre cada prueba se procuró que fueran aproximadamente iguales, pero esto dependía del lugar de inicio de cada robot.

La primera prueba que se realizó fue considerando un retardo $\tau = 0.4\ s$. En la Figura 5.8 se muestra que el sistema alcanza el consenso y a una formación tipo triangular, tal como se muestra en la Figura 3.1. Los estados X y Y de cada robot convergen a un valor constante como era de esperarse dada la estabilidad del sistema.

La segunda prueba se realizó con el retardo $\tau = 1.5\ s$. Los resultados mostrados (Figura 5.9) revelan que el consenso sigue siendo alcanzado, pero se puede observar que el tiempo en el cual alcanza la convergencia, es un poco mayor que en el caso anterior. También se observa la convergencia de los estados de cada agente.

Finalmente se realizó una última prueba donde $\tau = 2\ s$ (Figura 5.10). Con los resultados se deduce que el tiempo de convergencia es mayor que en los casos anteriores, se observa que el consenso puede ser alcanzado si se tomará un tiempo de ejecución mayor. El problema que se presentó en esta prueba fue que en espacio físico existió colisión entre los robots, debido a que tardan más tiempo en alcanzar consenso y oscilan mucho sobre la posición a la cual deberían de llegar. Esta limitante impidió que se pudiera realizar otros experimentos con tiempos de retardo mayor.

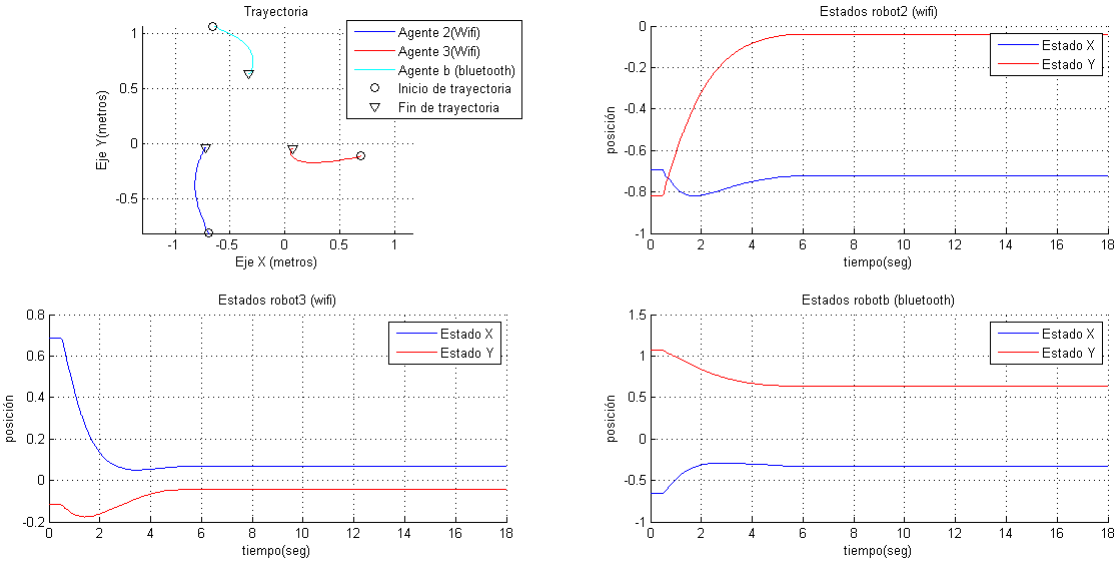


Figura 5.8: Experimento con $\tau = 0.4\ s$ sin observador.

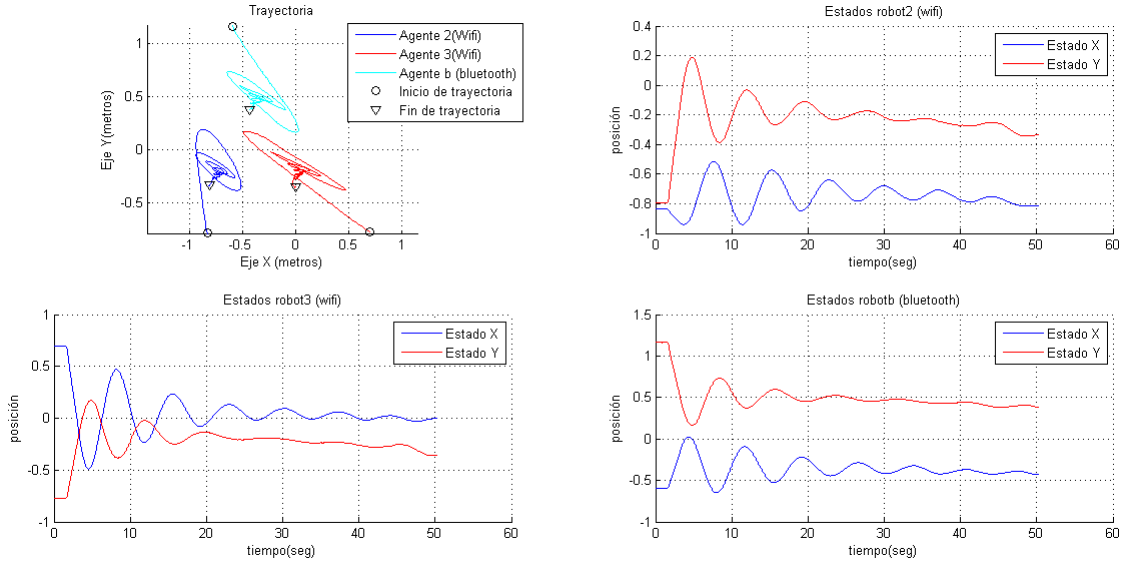


Figura 5.9: Experimento con $\tau = 1.5\ s$ sin observador.

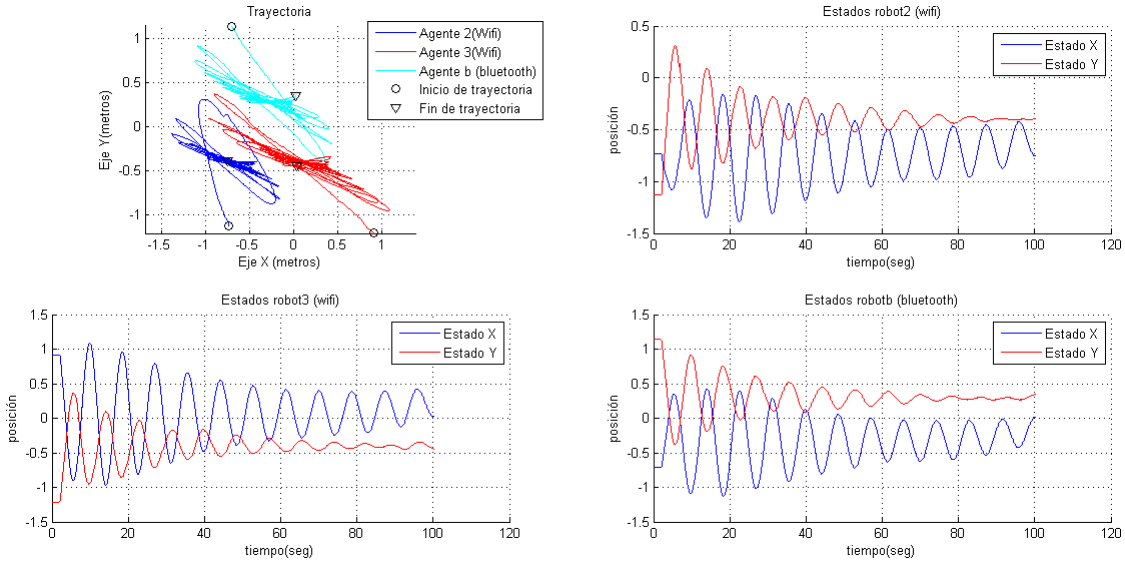


Figura 5.10: Experimento con $\tau = 2\ s$ sin observador.

5.3.2. Experimentos del Algoritmo de consenso con retardo con esquema de predicción

El sistema en lazo cerrado con el esquema de control de predicción de estados futuros, es el siguiente

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = A_i^{-1}(\theta_i) u_{\hat{w}}(t).$$

esto es

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = A_i^{-1}(\theta_i) \left(-\sum_{j=1}^{n} \left[a_{ij}(\hat{w}_i(t) - \hat{w}_j(t) - c_{ij}) \right] \right).$$

El esquema de control predictor–observador esta dado por (5.17). Por lo tanto, las ecuaciones de las velocidades lineales y angulares de los robots se escriben de la siguiente forma, par el robot b se tiene

$$\begin{bmatrix} v_b \\ \omega_b \end{bmatrix} = A_b^{-1}(\theta_b) \left\{ -k \left(\hat{w}_1(t) - \hat{w}_3(t) - c_{31} \right) \right\},$$

para el robot 2

$$\begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix} = A_2^{-1}(\theta_2) \left\{ -k \left(\hat{w}_2(t) - \hat{w}_1(t) - c_{12} \right) \right\},$$

y finalmente para el robot 3

$$\begin{bmatrix} v_3 \\ \omega_3 \end{bmatrix} = A_3^{-1}(\theta_3) \left\{ -k \left(2\hat{w}_3(t) - \hat{w}_1(t) - \hat{w}_2(t) - c_{13} - c_{23} \right) \right\}.$$

Considérese la misma topología de información que se ha utilizado hasta el momento (Figura 3.1). Sabemos que el retardo soportado máximo para este sistema de robots Garcia es de $\tau = 2.61799\ s$, por lo tanto bastaría con demostrar que el consenso se logra con valores de retardo mayores aplicando el esquema de control de predicción propuesto. Para este caso se tomará la variable de diseño del esquema predictor-observador con valor de $\lambda = 0.314159$ la cual nos da un tiempo máximo de retardo de $\tau^* = 5\ s$.

El primer ejemplo que se tiene es cuando $\tau = 1.5\ s$. Los resultados se observan en la Figura 5.11, el problema de consenso es resulto a pesar del retardo existente, se muestran las trayectorias de cada robot robot, así como trayectoria del predictor-observador para cada robot. Se observa que la tasa de convergencia es menor en comparación a los resultados mostrados en los experimentos sin el esquema de predicción. Los estados de todos los agentes convergen a un valor constante (Figuras 5.12, 5.13 y 5.14). Los errores de predicción tienden a cero (Figuras 5.15, 5.16 y 5.17).

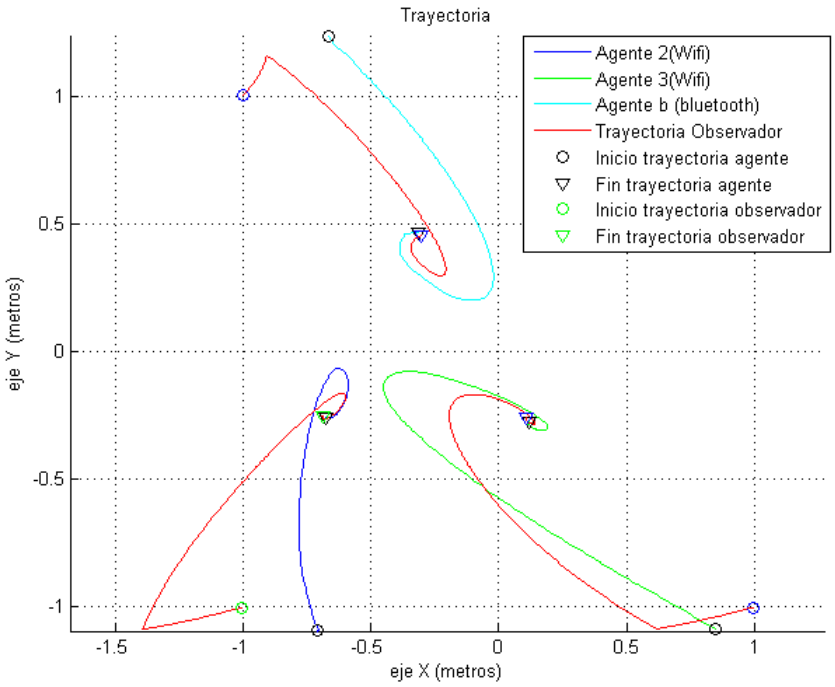


Figura 5.11: Consenso con $\tau = 1.5\ s$ con predictor-observador.

El segundo experimento que se realizó fue cuando $\tau = 2s$. En el ejemplo que se realizó sin esquema de predicción, se observó que el consenso tenia un tiempo de convergencia muy grande, ademas de las oscilaciones sobre el punto de encuentro el problema era la colisión entre agentes, impidiendo de esta manera finalizar el experimento adecuadamente. En este caso se muestra que la mejora de convergencia

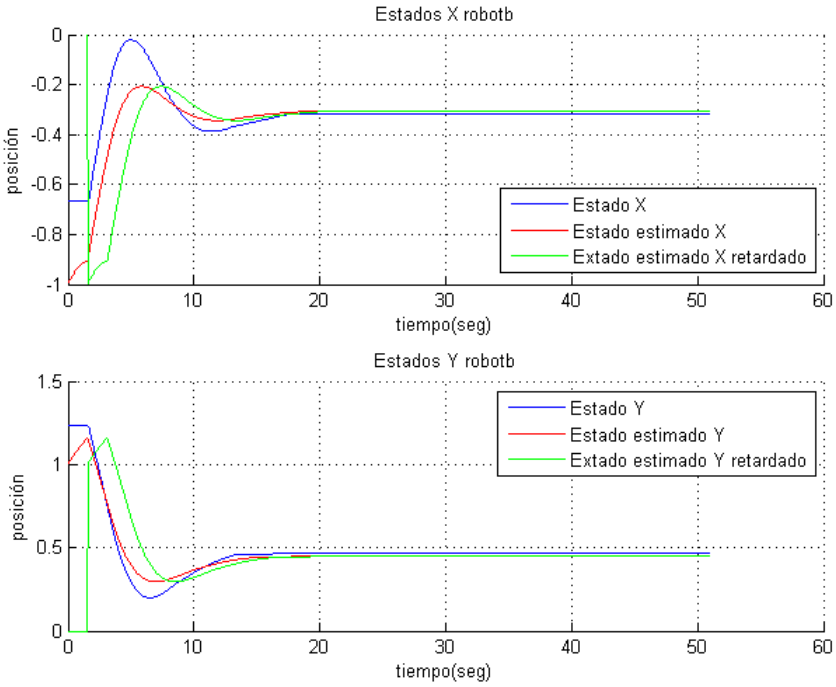


Figura 5.12: Convergencia de estados robot b con $\tau = 1.5\ s$.

para el sistema con $\tau = 2$ como retardo es evidente, ya que el sistema se estabiliza, lo que significa que alcanza el consenso, mostrando que el esquema predictor-observador es una buena herramienta para sistemas multiagentes. Las Figuras (5.18), (5.19), (5.20), (5.21), (5.22), (5.23) y (5.24) muestran los resultados de este experimento.

Finalmente el último experimento realizado fue con $\tau = 3\ s$, esto para determinar la eficacia total de esquema predictor-observador. En la Figura 5.25 se muestran los resultados. Se presenta el problema de colisión entre agentes lo que provoca que el consenso no se logre del todo bien, esto se observa en las gráficas de los estados de los robots (Figuras 5.26, 5.27 y 5.28) donde van aumentando o disminuyendo según la posición en el espacio de trabajo. El esquema predictor-observador trabaja adecuadamente, esto se ve en las gráficas de los errores de estimación mostrados en las Figuras 5.29 y **??**, los errores tienden a ser pequeños, pero debido a que no se alcanza el consenso, estos oscilan sobre cero.

En conclusión, la consideración de las dimensiones de los robots, aumento de tamaño del espacio de trabajo y quizás el incremento en el tiempo de ejecución del experimento, proporcionarían unos resultados mucho mejores y probablemente harían mas evidente la convergencia del sistema. A pesar de los inconvenientes, se alcanzó a mostrar que estos resultados también superan los obtenidos en la sección anterior, permitiéndonos tener una buena validación de funcionabilidad del es esquema predictor-observador

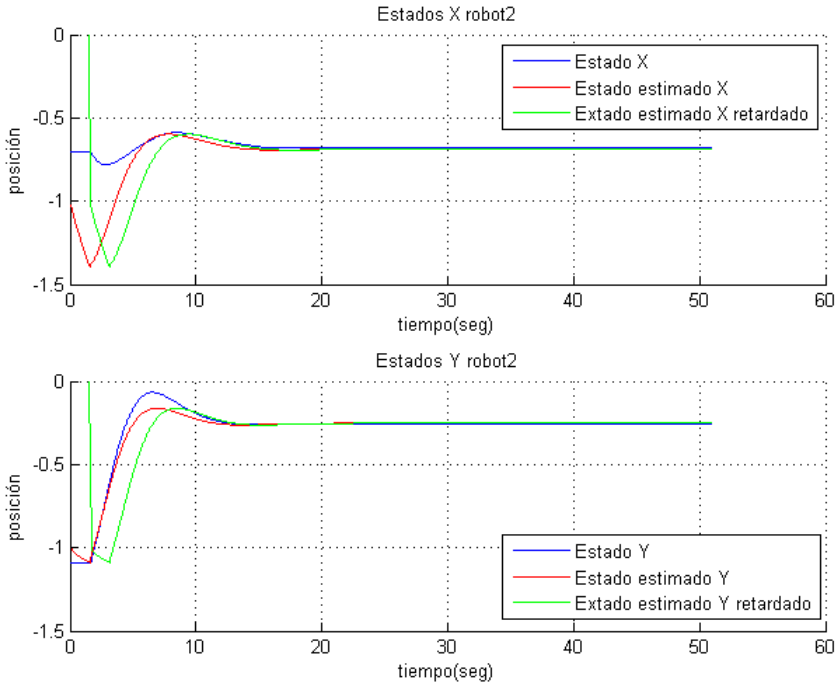


Figura 5.13: Convergencia de estados robot 2 con $\tau = 1.5\ s$.

propuesto.

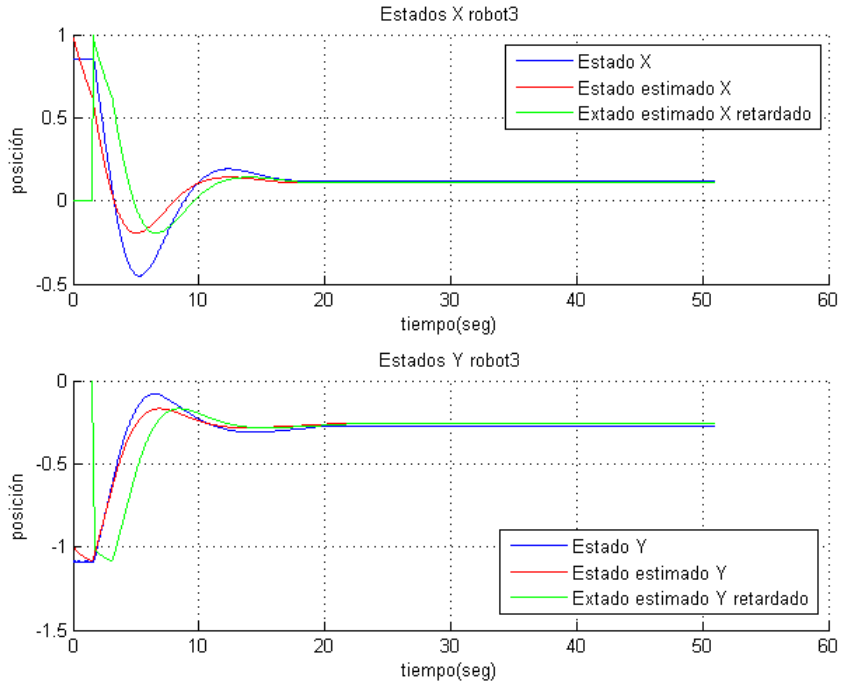


Figura 5.14: Convergencia de estados robot 3 con $\tau = 1.5\ s$.

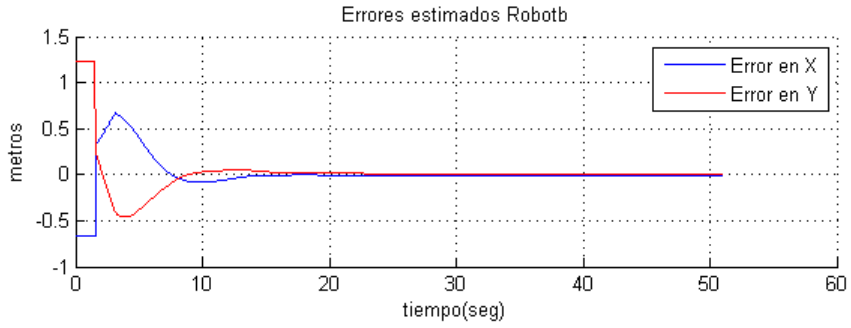


Figura 5.15: Errores de predicción robot b con $\tau = 1.5\ s$.

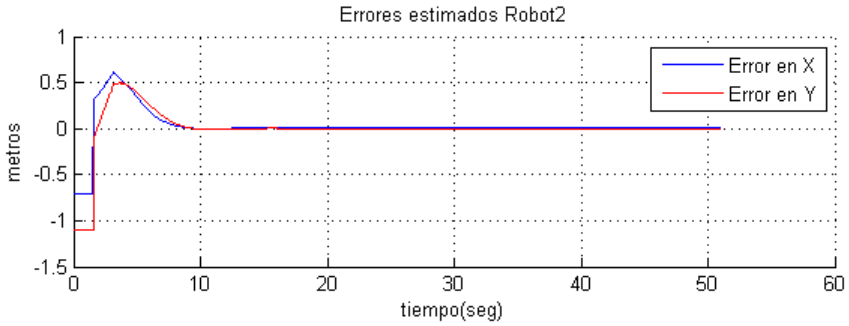


Figura 5.16: Errores de predicción robot 2 con $\tau = 1.5\ s$.

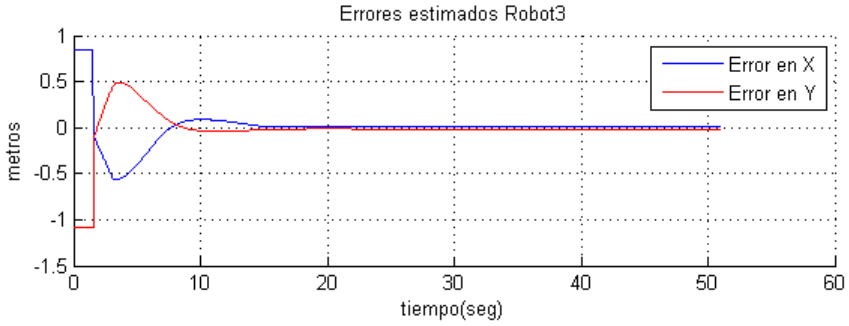


Figura 5.17: Errores de predicción robot 3 con $\tau = 1.5\ s$.

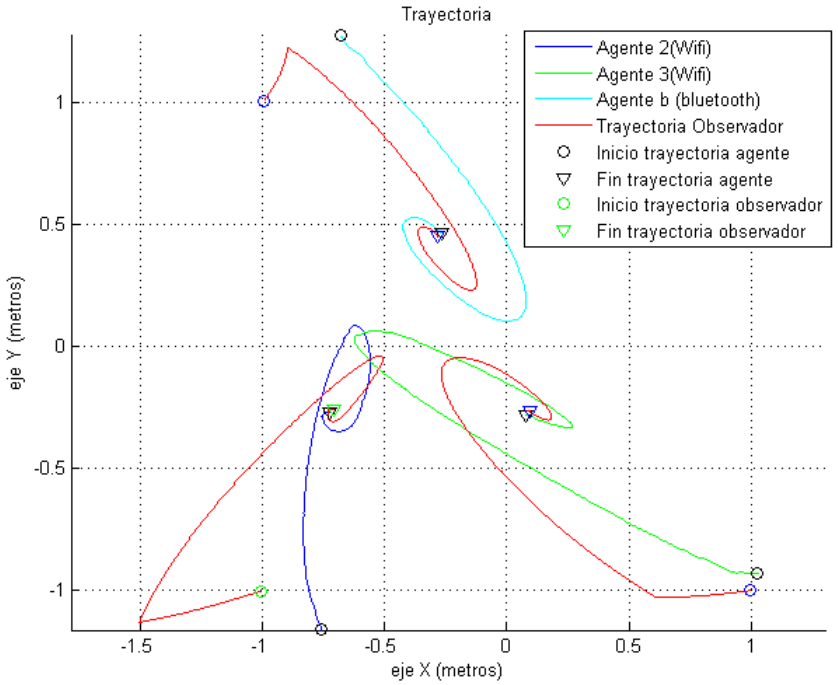


Figura 5.18: Consenso con $\tau = 2\ s$ con predictor-observador.

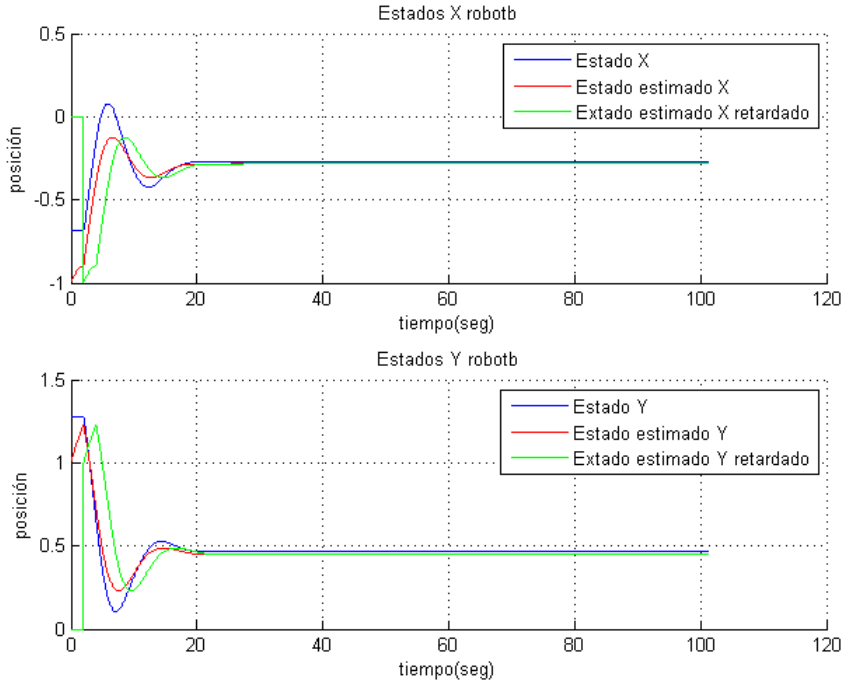


Figura 5.19: Convergencia de estados robot b con $\tau = 2\ s$.

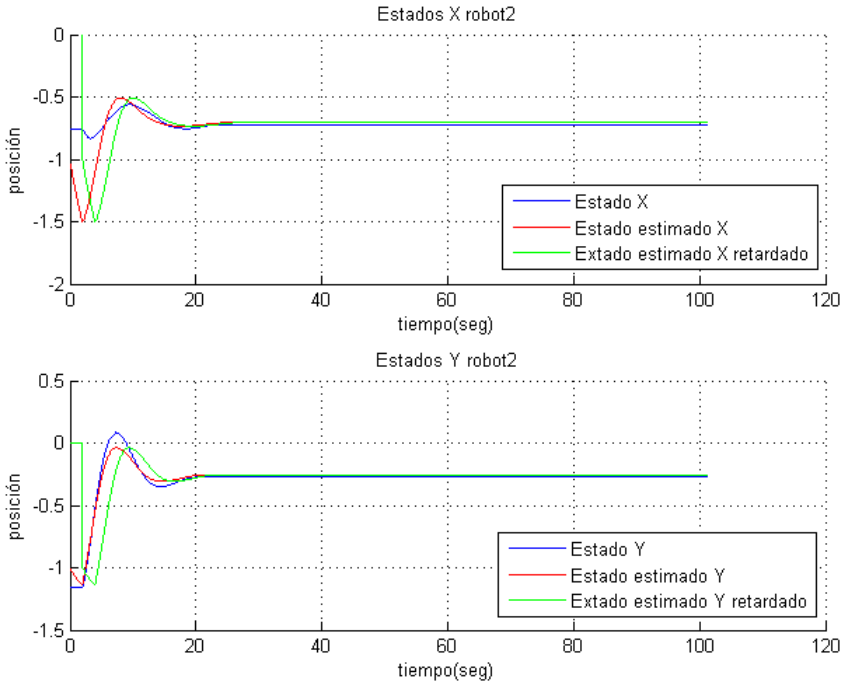


Figura 5.20: Convergencia de estados robot 2 con $\tau = 2\ s$.

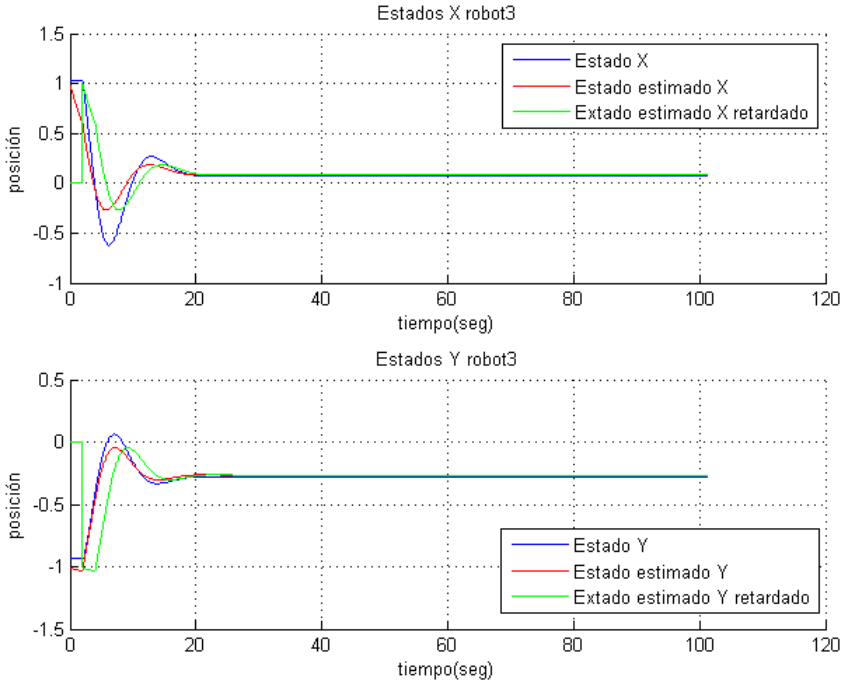


Figura 5.21: Convergencia de estados robot 3 con $\tau = 2\ s$.

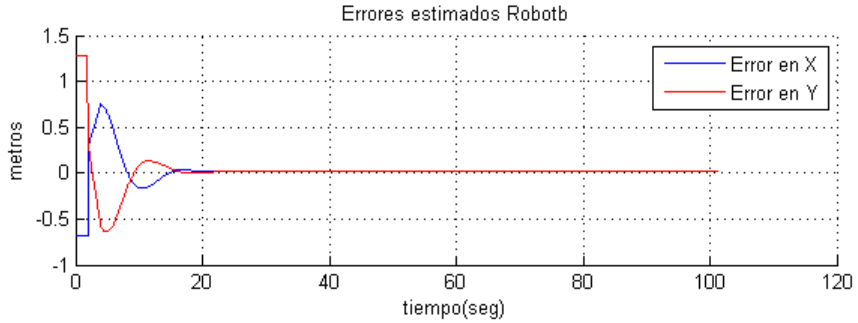


Figura 5.22: Errores de predicción robot b con $\tau = 2\ s$.

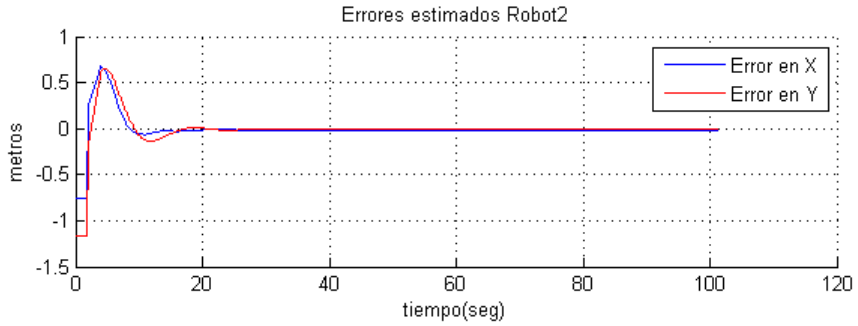


Figura 5.23: Errores de predicción robot 2 con $\tau = 2\ s$.

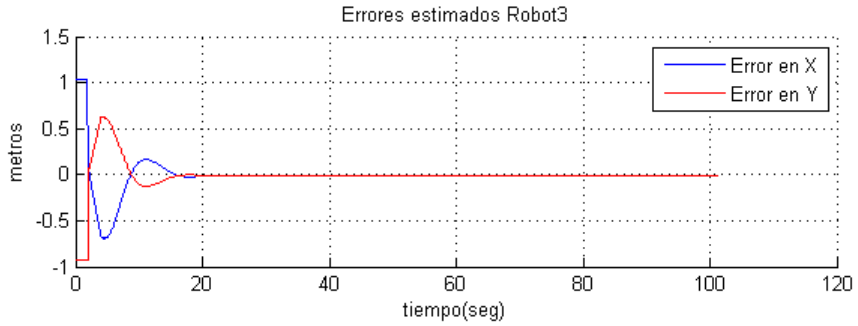


Figura 5.24: Errores de predicción robot 3 con $\tau = 2\ s$.

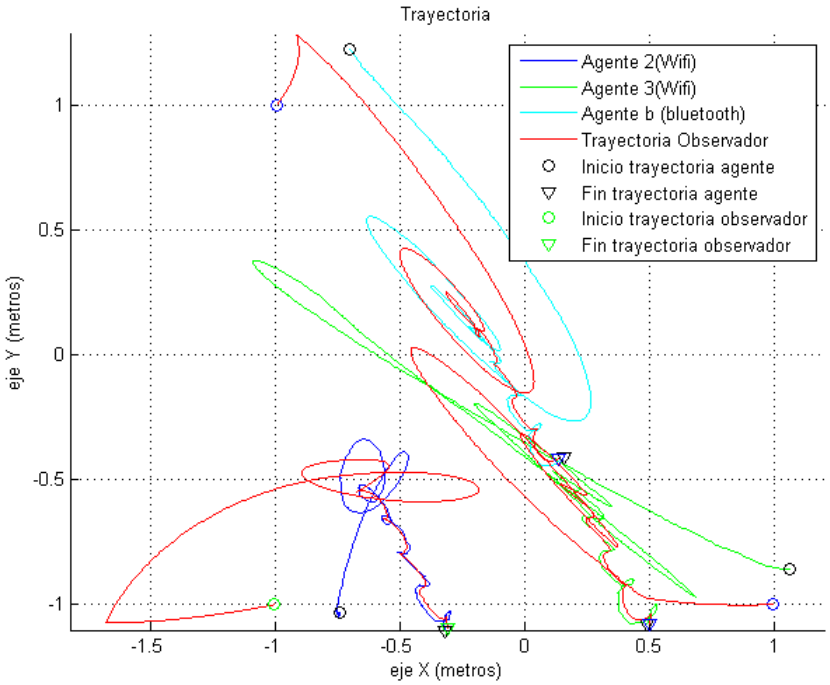


Figura 5.25: Consenso con $\tau = 3\ s$ con predictor-observador.

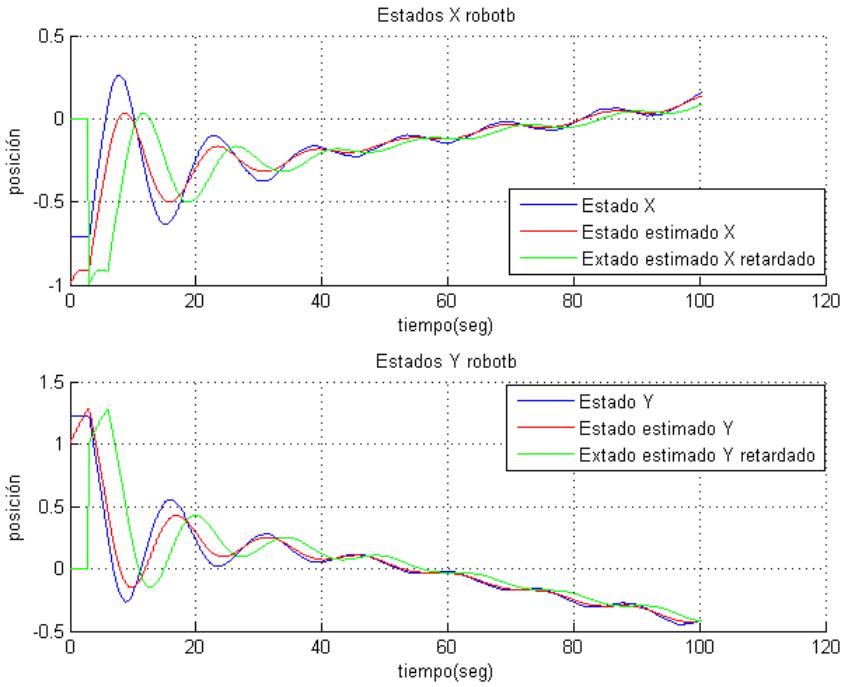


Figura 5.26: Convergencia de estados robot b con $\tau = 3\ s$.

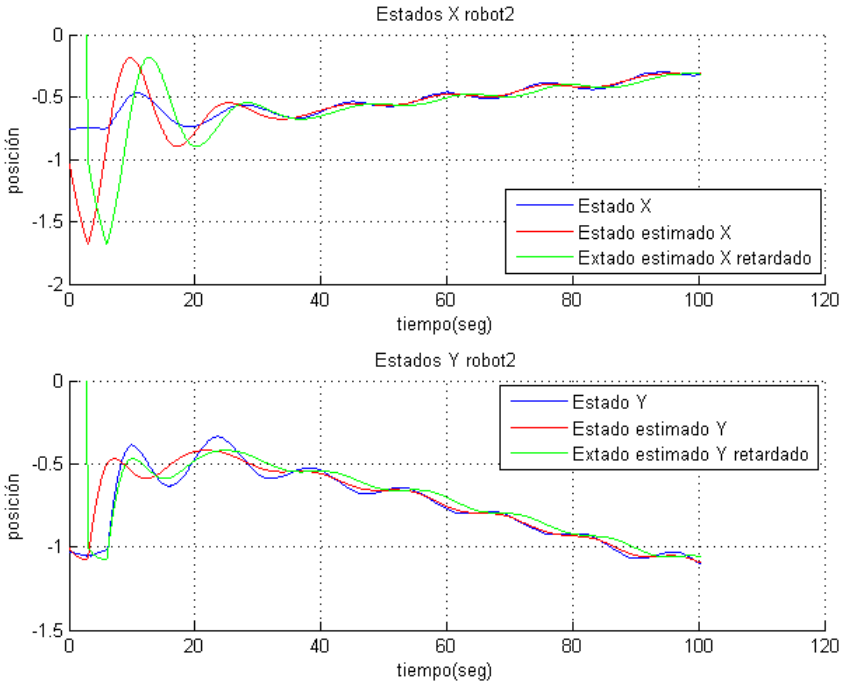


Figura 5.27: Convergencia de estados robot 2 con $\tau = 3\ s$.

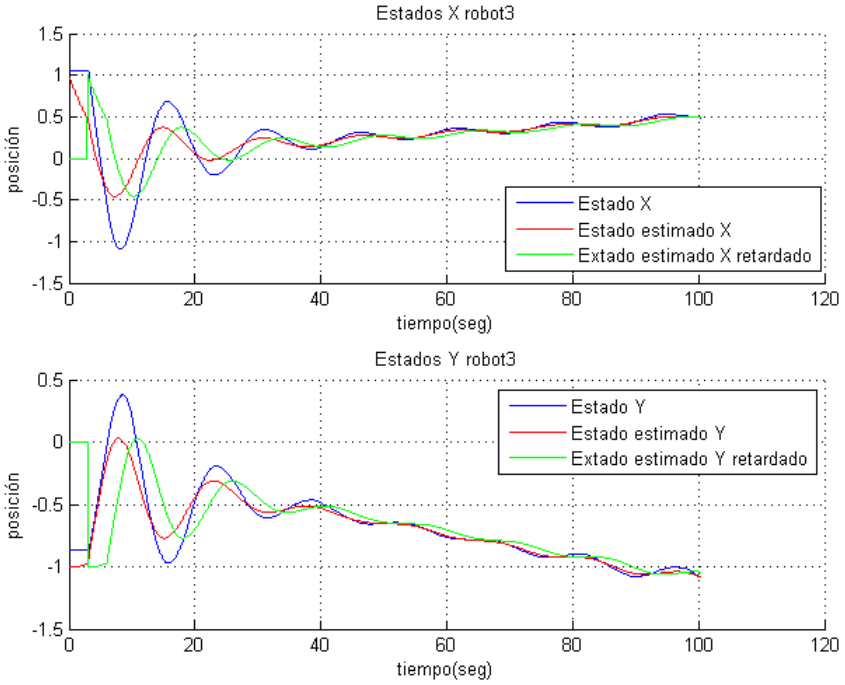


Figura 5.28: Convergencia de estados robot 3 con $\tau = 3\ s$.

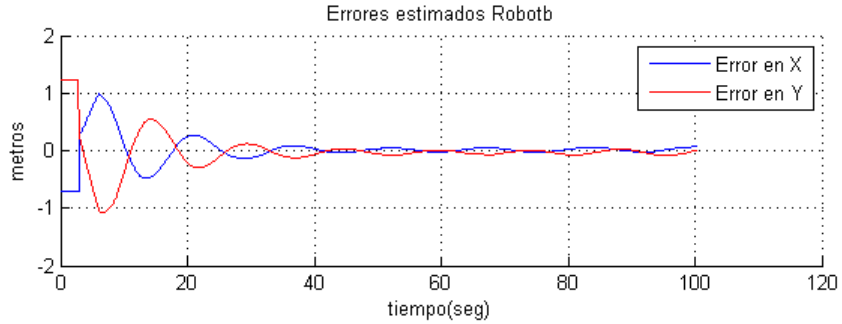


Figura 5.29: Errores de predicción robot b con $\tau = 3\ s$.

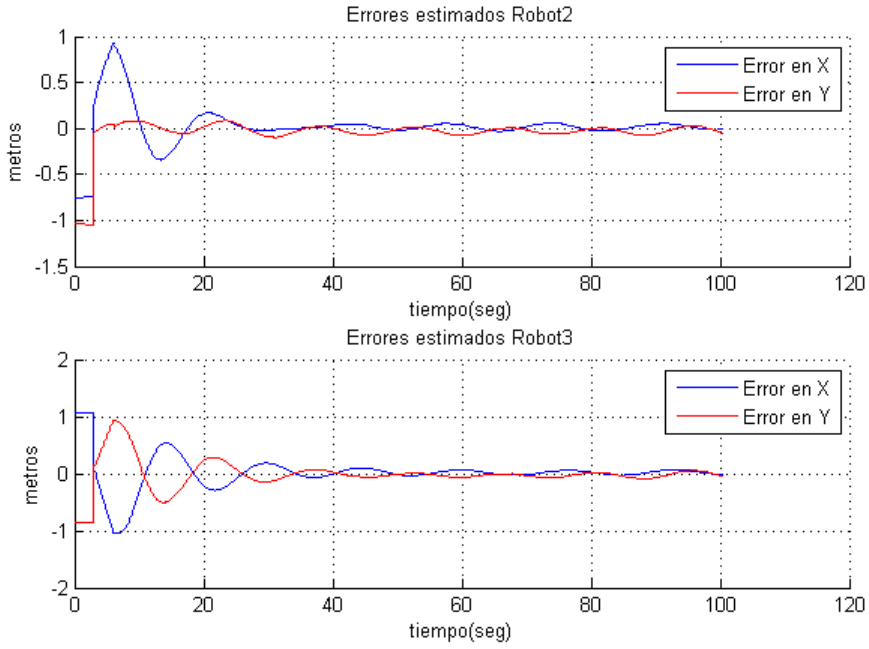


Figura 5.30: Errores de predicción robot 2 con $\tau = 3\ s$.

Capítulo 6

Conclusiones y perspectivas

6.1. Conclusiones

Hoy en día, ha habido un rápido progreso de nuevas teorías que crean una fusión entre la teoría de grafos y la teoría de sistemas de control para el problema de control cooperativo de sistemas de redes distribuidas. Ejemplo de uno de estos trabajos de investigación son los sistemas multiagentes, donde cada agente trabaja autónomamente al usar información de otros agentes en la red, por lo que es posible aplicar el problema de consenso o formación.

La información de consenso garantiza que los agentes que comparten información en una red con cualquier topología tengan una información consistente, la cual es vital para la tarea de coordinación. Por necesidad, los algoritmos de consenso son diseñados para ser distribuidos, asumiendo únicamente la interacción entre agentes de vecino a vecino. Los agentes actualizan el valor de información del estado con respecto a su vecino.

En este trabajo se analiza el problema de consenso para un grupo de agentes sujetos a tiempos de retardo fijo en la comunicación entre ellos. Se aborda el análisis sobre sistemas de un solo y doble integrador. Se considera que las entradas de los agentes son las afectadas por los retardos lo que conduce a proponer una retroalimentación en un esquema Predictor–Observador, el cual provee una solución al problema de consenso que mejora sustancialmente el resultado presentado en [19] para el caso de sistemas de un solo integrador, y el resultado presentado en [18] y [36] que corresponde al caso de sistemas doble integrador. El esquema propuesto recobra la convergencia del problema de retardo y tiene la ventaja de que las condiciones de estabilidad dependen de los parámetros de diseño y no de una topología en particular. El esquema de control propuesto es evaluado mediante simulaciones numéricas y experimentos en tiempo real mostrando un adecuado funcionamiento.

6.2. Perspectivas

- Se considera la posibilidad de aplicar es esquema de control predictor-observador en redes de agentes con topología variante y retardo variante en el tiempo, tanto para sistemas de un solo integrador así como de doble integrador.
- Implementación del esquema de control propuesto de forma experimental, del sistema de doble integración.
- Modificación al esquema de control en conjunto de la ley de consenso, de tal forma que se evite la colisión entre agentes al momento de realizar consenso.
- Se puede hablar de la implementación del esquema de predicción en estrategias de formación o seguimiento de trayectoria para sistemas multiagente, cuando existe un retardo en la comunicación.
- Aplicar el esquema de control en sistemas multiagentes con robots que no sean de configuración 2.0, por ejemplo, robots omnidireccionales, drones, entre otros; además de la implementación en sistemas multiagentes heterogéneos.
- Analizar y realizar un estudio para determinar la posibilidad de aumentar robustez al control de tal forma que resuelva el problema de consenso no solo considerando como problema los retardos de tiempo, sino que el esquema de control también sea capaz de corregir cuando existan perturbaciones internas o externas al sistema.

6.3. Artículos publicados

- J. Heras-Godínez, M. Velasco-Villa y J. A. Vásquez, “Problema de consenso en redes de agentes de primer orden con retardo en la comunicación”. Sometido al XVI Congreso Latinoamericano de Control Automático, Cancún, Quintana Roo, México, Octubre 2014.
- J. Heras-Godínez, M. Velasco-Villa y J. A. Vásquez, “Problema de consenso en agentes doble integrador y tiempos de retardo”. Sometido al XVI Congreso Mexicano de Robótica, Mazatlán, Sinaloa, México, Noviembre 2014.

Apéndice A

Programa en C++. Algoritmo de consenso con tiempo de retardo en la comunicación sin esquema Predictor-observador

En este apartado se presenta el código de programación en lenguaje $C++$ donde se implementa el algoritmo de consenso con tiempo de retardo desarrollado en [19] en un sistema multiagente de 3 robots Garcia, esto con el objetivo de verificar el funcionamiento practico de dicha teoría.

Programa A.1: Código principal del algoritmo de consenso con retardo en un sistema con 3 agentes

```
1  %=========================================================================
2
3  #include <fstream>
4  #include <cstdlib>                //libreria de tiempo
5  #include <time.h>                 //libreria de tiempo
6  #include <sys/timeb.h>            //libreria de tiempo
7  #include "socket.h"
8  #include "sistema.h"
9  #using <System.dll>                              // agregada por
     Bluetooth
10 #include <boost/lexical_cast.hpp>
11 #include <boost/numeric/odeint.hpp>                    //libreria
     para poder integrar
12 #include <boost/chrono/chrono.hpp>                    //libreria
     para usar boost::chronos::clock
13 #include "delay.h"
14 #include "derivada.h"
```

```
15 #include "vrpn.h"
16
17 using namespace std;
18 using namespace System;
19 using namespace System::IO::Ports;
20 using namespace Eigen;
21 using namespace boost::numeric::odeint;
22
23 char* dir2 = "192.168.2.2";
24 char* puerto2 = "8000";
25
26 char* dir3 = "192.168.2.3";
27 char* puerto3 = "8001";
28
29 #define dt  0.02   // Tiempo de muestreo
30
31 double t=0;
32 char op=0; // Variable usada para controlar el inicio del programa
     en el while
33 int contador=1;
34 double alpx2=0, alpz2=0, alpx3=0, alpz3=0;
35 double alpxb=0, alpzb=0;
36 double vel2=0, vel3=0;
37 double velb=0;
38
39 int main(){
40
41 #pragma region Parametros
42     sockett robot2,robot3;
43     Sistema retro;
44     Delay ax2r, az2r, ax3r, az3r; //Instancias para los controles
        retardados
45     Delay axbr, azbr;
46     Derivada Deri2, Deri3;
47     Derivada Derib;
48     fstream iofile;                  //Para crear archivo
49     ofstream graficar;               //Para guardas valores en
        graficar
50     char num[30];
51     char num2[30];
52     Vect12d Velo;
53
54 #pragma endregion
55
56 //-----------------------------------------------------------
57 #pragma region Robot2Socket
58     if(robot2.ConectarSocket(dir2,puerto2) == 0)
59     {
60         puts("\n No se pudo establecer conexion\n");
61         robot2.conectado_red = false;
62         puts("\nPresiona la tecla 'Enter' para seguir.\n");
```

```
63          getchar();
64          //goto stop;
65      }
66      else
67          robot2.conectado_red = true;
68  #pragma endregion
69
70  //-------------------------------------------------------------
71  #pragma region Robot3Socket
72      if(robot3.ConectarSocket(dir3,puerto3) == 0)
73      {
74          puts("\n No se pudo establecer conexion\n");
75          robot3.conectado_red = false;
76          puts("\nPresiona la tecla 'Enter' para seguir.\n");
77          getchar();
78          //goto stop;
79      }
80      else
81          robot3.conectado_red = true;
82  #pragma endregion
83
84  #pragma region BluetoothConexion
85  cli::array<unsigned char>^ buffb = gcnew cli::array<unsigned char
      >(12);
86
87      buffb[0]=4;                 //Buffer para el robot bluetooth
88      buffb[1]=4;
89      buffb[2]=62;
90      buffb[3]=0;                   //Llanta Izquierda
91      buffb[4]=0;
92      buffb[5]=0;
93      buffb[6]=4;
94      buffb[7]=4;
95      buffb[8]=62;
96      buffb[9]=1;                   //Llanta Derecha
97      buffb[10]=0;
98      buffb[11]=0;
99
100 SerialPort Bluetooth;
101
102 Bluetooth.BaudRate = 38400;
103 Bluetooth.DtrEnable = true;
104 Bluetooth.PortName = "COM6";
105 Bluetooth.RtsEnable = true;
106
107 Bluetooth.Open();
108
109 if(Bluetooth.IsOpen){
110     cout<<"Conectado con robot Bluetooth"<<endl;
111 }
112 #pragma endregion
```

```
113
114 #pragma region ConexionVRPN
115
116     sprintf(conexionName,"localhost:%d",DEFAULT_VRPN_PORT);
117     conexion = vrpn_get_connection_by_name(conexionName);
118
119     vrpn_Tracker_Remote *trackerb = new vrpn_Tracker_Remote("
          Trackerb", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Trackerb' debe ser el mismo en
          Optitrack
120     vrpn_Tracker_Remote *tracker2 = new vrpn_Tracker_Remote("
          Tracker2", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Tracker2' debe ser el mismo en
          Optitrack
121     vrpn_Tracker_Remote *tracker3 = new vrpn_Tracker_Remote("
          Tracker3", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Tracker2' debe ser el mismo en
          Optitrack
122
123     trackerb->register_change_handler (NULL, handle_posb);
124     tracker2->register_change_handler (NULL, handle_pos2);
125     tracker3->register_change_handler (NULL, handle_pos3);
126
127     puts("\nComunicacion VRPN iniciada.\n");    // Despues de
          establecer la comunicacion con el servidor VRPN se tiene que
          esperar un tiempo para estabilizar la comunicacion
128     Sleep(2000);
129
130     #pragma endregion
131
132 //
    ----------------------------------------------------------------------

133 //Creacion de Archivo donde se guardaran los datos a graficar
134 #pragma region Creacion de Archivos para Graficas
135     iofile.open("No_experimento.txt");
136     if(!iofile.is_open()){
137         cout << "No se pudo abrir archivo No_experimento" << endl;
138     }
139     iofile.getline(num,30);
140     int k=boost::lexical_cast<int>(num);
141     k++;
142     sprintf_s(num2,30,"%d",k);
143     sprintf_s(num,30,"graficas%d.txt",k);
144     graficar.open(num);
145     iofile.clear();
146     iofile.seekp(0,ios::beg);
147     iofile.write(num2,sizeof(num2));
148
149     if(graficar.fail())
```

```
150         cout << "El archivo para graficar no se abrio
              correctamente" << endl;
151 #pragma endregion
152
153 //-------------------------------------------------------------
154 #pragma region InicializandoRobot
155     robot2.EnviarCadena(0,0,0,0);
156     robot3.EnviarCadena(0,0,0,0);
157     Bluetooth.Write(buffb,0,12);
158 #pragma endregion
159
160     cout << "COMIENZO" << endl;
161     while(op!=27){
162             if (!kbhit()){
163                 boost::chrono::steady_clock::time_point start=
                      boost::chrono::high_resolution_clock::now(); //
                      Obtener el tiempo de inicio de procesos para
                      calculo del tiempo total de ejecucion
164 //-------------------------------------------------------------
165                 trackerb->mainloop();
166                 tracker2->mainloop();
167                 tracker3->mainloop();
168                 conexion->mainloop();
169
170 #pragma region Calculo del punto frontal
171
172                 alpx2=x2+ls*cos(yaw2);
173                 alpz2=z2+ls*sin(yaw2);
174
175                 alpx3=x3+ls*cos(yaw3);
176                 alpz3=z3+ls*sin(yaw3);
177
178                 alpxb=xb+ls*cos(yawb);
179                 alpzb=zb+ls*sin(yawb);
180 #pragma endregion
181 //------Aplicando retardo-------------------------
182 #pragma region Retardo
183                 if (contador <= MAX_RETARDO){
184                     ax2r.add(alpx2);
185                     az2r.add(alpz2);
186                     ax3r.add(alpx3);
187                     az3r.add(alpz3);
188                     axbr.add(alpxb);
189                     azbr.add(alpzb);
190
191                     contador = contador + 1;
192                     //cout << t << endl;
193                     goto alto;
194                 }
195                 else{
196                     ax2r.remove();
```

```
197                     ax2r.add(alpx2);
198                     az2r.remove();
199                     az2r.add(alpz2);
200
201                     ax3r.remove();
202                     ax3r.add(alpx3);
203                     az3r.remove();
204                     az3r.add(alpz3);
205
206                     axbr.remove();
207                     axbr.add(alpxb);
208                     azbr.remove();
209                     azbr.add(alpzb);
210                     }
211 #pragma endregion
212 //-------------------------------------------------------------
213                 Velo=retro.sistema(ax2r.Get(),az2r.Get(),yaw2,ax3r
                      .Get(),az3r.Get(),yaw3, axbr.Get(),azbr.Get(),
                      yawb);
214                 robot2.EnviarCadena(Velo[0],Velo[1],Velo[2],Velo
                      [3]);
215                 robot3.EnviarCadena(Velo[4],Velo[5],Velo[6],Velo
                      [7]);
216
217                 buffb[4]=Velo[8];
218                 buffb[5]=Velo[9];
219                 buffb[10]=Velo[10];
220                 buffb[11]=Velo[11];
221                 Bluetooth.Write(buffb,0,12);
222                 alto:
223 //-------------------------------------------------------------
224                 Deri2.derivar(x2,z2,yaw2);
225                 Deri3.derivar(x3,z3,yaw3);
226                 Derib.derivar(xb,zb,yawb);
227                 vel2=sqrt((Deri2.GetXp()*Deri2.GetXp())+(Deri2.
                      GetZp()*Deri2.GetZp())); //Calculo de la
                      velocidad lineal del garcia v=sqrt(xp^2+zp^2)
228                 vel3=sqrt((Deri3.GetXp()*Deri3.GetXp())+(Deri3.
                      GetZp()*Deri3.GetZp()));
229                 velb=sqrt((Derib.GetXp()*Derib.GetXp())+(Derib.
                      GetZp()*Derib.GetZp()));
230 //-------------------------------------------------------------
231 graficar << t << "\t" << alpx2 << "\t" << alpz2 << "\t" << yaw2 <<
      "\t" << alpx3 << "\t" << alpz3 << "\t" << yaw3 << "\t" << retro.
    GetV2() << "\t" << retro.GetW2() << "\t" << retro.GetV3() << "\t"
     << retro.GetW3() << "\t" << vel2 << "\t" << vel3 << "\t" << ax2r
    .Get() << "\t" << az2r.Get() << "\t" << ax3r.Get() << "\t" <<
    az3r.Get() << "\t" << alpxb << "\t" << alpzb << "\t" << yawb << "
    \t" << retro.GetVb() << "\t" << retro.GetWb() << "\t" << velb
232         << "\t" << axbr.Get() << "\t" << azbr.Get() << endl;
233 //-------------------------------------------------------------
```

```
234                 Deri2.Iteracion();
235                 Deri3.Iteracion();
236                 Derib.Iteracion();
237 //------Retardo de 10 ms-----------------------------------------
238                 cout << t << endl;
239                 t=t+0.02;
240                 boost::chrono::duration<double> sec = boost::
                      chrono::steady_clock::now()-start;
241                 double faltante=sec.count()*1000000000;
242                 faltante=dt*1000000000-faltante;
243                 if(faltante){
244                     auto go=boost::chrono::steady_clock::now()+
                          boost::chrono::nanoseconds((long)faltante);
245                     while (boost::chrono::steady_clock::now()<go);
246                     }
247                 boost::chrono::duration<double> second = boost::
                      chrono::steady_clock::now()-start;
248                 //cout << "Time: " << second.count() << endl;
249             }
250             else{
251                 op=_getch();
252                 cout << "ALTO" << endl;
253             }
254     }
255     buffb[4]=0;
256     buffb[5]=0;
257     buffb[10]=0;
258     buffb[11]=0;
259     Bluetooth.Write(buffb,0,12);
260     robot2.EnviarCadena(0,0,0,0);
261     robot3.EnviarCadena(0,0,0,0);
262     Sleep(3000);
263     robot2.CloseSocket();
264     robot3.CloseSocket();
265     Bluetooth.Close();
266     if(!Bluetooth.IsOpen){
267         cout<<"Desconexion bluetooth exitosa"<<endl;
268       }
269     else{
270         cout<<"Aun esta conectado"<<endl;
271     }
272     Sleep(2000);
273     return 0;
274 }
```

Apéndice B

Programa en C++. Algoritmo de consenso con tiempo de retardo en la comunicación con esquema Predictor-observador

En este apartado se presenta el código de programación en lenguaje $C++$ donde se implementa el esquema Predictor-Observador propuesto que resuelve es problema de consenso con tiempo de retardo en la comunicación para un sistema multiagente de 3 robots Garcia.

Programa B.1: Código principal del algoritmo de consenso con retardo y en lazo cerrado con el esquema de control predictor-observador en un sistema con 3 agentes

```
1
2  #include <fstream>
3  #include <cstdlib>              //libreria de tiempo
4  #include <time.h>               //libreria de tiempo
5  #include <sys/timeb.h>          //libreria de tiempo
6  #include "socket.h"
7  #include "sistema.h"
8  #using <System.dll>                          // agregada por
     Bluetooth
9  #include <boost/lexical_cast.hpp>
10 #include <boost/numeric/odeint.hpp>                  //libreria
     para poder integrar
11 #include <boost/chrono/chrono.hpp>                  //libreria
     para usar boost::chronos::clock
12 #include "delay.h"
13 #include "derivada.h"
14 #include "vrpn.h"
15
```

```
16 using namespace std;
17 using namespace System;
18 using namespace System::IO::Ports;
19 using namespace Eigen;
20 using namespace boost::numeric::odeint;
21
22 char* dir2 = "192.168.2.2";
23 char* puerto2 = "8000";
24
25 char* dir3 = "192.168.2.3";
26 char* puerto3 = "8001";
27
28 #define dt  0.02   // Tiempo de muestreo
29
30 double lambda=0.314159; //-----Para un retardo hasta 5seg
31 double t=0;
32 char op=0; // Variable usada para controlar el inicio del programa
     en el while
33 int contador=1;
34 double alpx2=0, alpz2=0, alpx3=0, alpz3=0;
35 double alpxb=0, alpzb=0;
36 double vel2=0, vel3=0;
37 double velb=0;
38
39 double ex2=0;
40 double ez2=0;
41 double ex3=0;
42 double ez3=0;
43 double exb=0;
44 double ezb=0;
45 double wx2r=0, wz2r=0;
46 double wx3r=0, wz3r=0;
47 double wxbr=0, wzbr=0;
48
49 typedef Eigen::Matrix<double,6,1> state_type;
50                 //Es igual escribir "typedef Eigen::Matrix<double
                      ,6,1> state_type;"
51
52 //--------Funcion Observador ----------------------------------
53 // Funcion que integra
54 void integral(const state_type &w_vector, state_type &dwdt, double
     ){
55     dwdt[0]=-K*(w_vector[0]-w_vector[4]-h31)+lambda*exb;    //dwdt
        [0] estado estimado x punto, donde w_vector[0] sera estado
        estimado wx del robot b
56     dwdt[1]=-K*(w_vector[1]-w_vector[5]-v31)+lambda*ezb;    //dwdt
        [1] estado estimado y punto, donde w_vector[1] sera estado
        estimado wy del robot b
57     dwdt[2]=-K*(w_vector[2]-w_vector[0]-h12)+lambda*ex2;     //dwdt
        [2] estado estimado x punto, donde w_vector[2] sera estado
        estimado wx del robot 2
```

```
58      dwdt[3]=-K*(w_vector[3]-w_vector[1]-v12)+lambda*ez2;     //dwdt
          [3] estado estimado y punto, donde w_vector[3] sera estado
          estimado wy del robot 2
59      dwdt[4]=-K*(2*w_vector[4]-w_vector[0]-w_vector[2]-h13-h23)+
          lambda*ex3;  //dwdt[4] estado estimado x punto, donde
          w_vector[4] sera estado estimado wx del robot 3
60      dwdt[5]=-K*(2*w_vector[5]-w_vector[1]-w_vector[3]-v13-v23)+
          lambda*ez3;  //dwdt[5] estado estimado y punto, donde
          w_vector[5] sera estado estimado wy del robot 3
61  }
62
63  int main(){
64  #pragma region Parametros
65      sockett robot2,robot3;
66      Sistema retro;
67      Delay ax2r, az2r, ax3r, az3r; //Instancias para los controles
          retardados
68      Delay axbr, azbr;
69      Delay Wx2, Wz2;              //Objeto para obtener los estados
          estimados retardados
70      Delay Wx3, Wz3;
71      Delay Wxb, Wzb;
72      Derivada Deri2, Deri3;
73      Derivada Derib;
74      fstream iofile;                  //Para crear archivo
75      ofstream graficar;               //Para guardas valores en
          graficar
76      char num[30];
77      char num2[30];
78      //Vect8d Velo; //Vector redundante por la weba de no hacer
          getters
79      Vect12d Velo;
80
81      state_type w_vector(6);      //Vector utilizado en la funcion
          de integracion, representa los estados estimado
82      w_vector << -1, 1, -1, -1, 1, -1;    //inicializando vector a
          integrar
83      runge_kutta4<state_type,double,state_type,double,
          vector_space_algebra> rk4;//Genera tipo de integracion
84
85  #pragma endregion
86
87  //-------------------------------------------------------------
88  #pragma region Robot2Socket
89      if(robot2.ConectarSocket(dir2,puerto2) == 0)
90      {
91          puts("\n No se pudo establecer conexion\n");
92          robot2.conectado_red = false;
93          puts("\nPresiona la tecla 'Enter' para seguir.\n");
94          getchar();
95          //goto stop;
```

```
96      }
97      else
98          robot2.conectado_red = true;
99  #pragma endregion
100
101 //-------------------------------------------------------------
102 #pragma region Robot3Socket
103     if(robot3.ConectarSocket(dir3,puerto3) == 0)
104     {
105         puts("\n No se pudo establecer conexion\n");
106         robot3.conectado_red = false;
107         puts("\nPresiona la tecla 'Enter' para seguir.\n");
108         getchar();
109         //goto stop;
110     }
111     else
112         robot3.conectado_red = true;
113 #pragma endregion
114
115 #pragma region BluetoothConexion
116 cli::array<unsigned char>^ buffb = gcnew cli::array<unsigned char
      >(12);
117
118     buffb[0]=4;                 //Buffer para el robot bluetooth
119     buffb[1]=4;
120     buffb[2]=62;
121     buffb[3]=0;                   //Llanta Izquierda
122     buffb[4]=0;
123     buffb[5]=0;
124     buffb[6]=4;
125     buffb[7]=4;
126     buffb[8]=62;
127     buffb[9]=1;                   //Llanta Derecha
128     buffb[10]=0;
129     buffb[11]=0;
130
131 SerialPort Bluetooth;
132
133 Bluetooth.BaudRate = 38400;
134 Bluetooth.DtrEnable = true;
135 Bluetooth.PortName = "COM6";
136 Bluetooth.RtsEnable = true;
137
138 Bluetooth.Open();
139
140 if(Bluetooth.IsOpen){
141     cout<<"Conectado con robot Bluetooth"<<endl;
142 }
143 #pragma endregion
144
145 #pragma region ConexionVRPN
```

```
146
147     sprintf(conexionName,"localhost:%d",DEFAULT_VRPN_PORT);
148     conexion = vrpn_get_connection_by_name(conexionName);
149
150     vrpn_Tracker_Remote *trackerb = new vrpn_Tracker_Remote("
          Trackerb", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Trackerb' debe ser el mismo en
          Optitrack
151     vrpn_Tracker_Remote *tracker2 = new vrpn_Tracker_Remote("
          Tracker2", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Tracker2' debe ser el mismo en
          Optitrack
152     vrpn_Tracker_Remote *tracker3 = new vrpn_Tracker_Remote("
          Tracker3", conexion); //Hace la conexion con Optitrack, el
          nombre del objeto a seguir 'Tracker2' debe ser el mismo en
          Optitrack
153
154     trackerb->register_change_handler (NULL, handle_posb);
155     tracker2->register_change_handler (NULL, handle_pos2);
156     tracker3->register_change_handler (NULL, handle_pos3);
157
158     puts("\nComunicacion VRPN iniciada.\n");    // Despues de
          establecer la comunicacion con el servidor VRPN se tiene que
          esperar un tiempo para estabilizar la comunicacion
159     Sleep(2000);
160
161     #pragma endregion
162
163 //-------------------------------------------------------------
164 //Creacion de Archivo donde se guardaran los datos a graficar
165 #pragma region Creacion de Archivos para Graficas
166     iofile.open("No_experimento.txt");
167     if(!iofile.is_open()){
168         cout << "No se pudo abrir archivo No_experimento" << endl;
169     }
170     iofile.getline(num,30);
171     int k=boost::lexical_cast<int>(num);
172     k++;
173     sprintf_s(num2,30,"%d",k);
174     sprintf_s(num,30,"graficas%d.txt",k);
175     graficar.open(num);
176     iofile.clear();
177     iofile.seekp(0,ios::beg);
178     iofile.write(num2,sizeof(num2));
179
180     if(graficar.fail())
181         cout << "El archivo para graficar no se abrio
              correctamente" << endl;
182 #pragma endregion
183
184 //-------------------------------------------------------------
```

```
185 #pragma region InicializandoRobot
186     robot2.EnviarCadena(0,0,0,0);
187     robot3.EnviarCadena(0,0,0,0);
188     Bluetooth.Write(buffb,0,12);
189 #pragma endregion
190 
191     cout << "COMIENZO" << endl;
192     while(op!=27){
193             if (!kbhit()){
194                 boost::chrono::steady_clock::time_point start=
                      boost::chrono::high_resolution_clock::now(); //
                      Obtener el tiempo de inicio de procesos para
                      calculo del tiempo total de ejecucion
195 //-------------------------------------------------------------
196                 trackerb->mainloop();
197                 tracker2->mainloop();
198                 tracker3->mainloop();
199                 conexion->mainloop();
200 
201 #pragma region Calculo del punto frontal y error de estimacion
202 
203                 alpxb=xb+ls*cos(yawb);
204                 alpzb=zb+ls*sin(yawb);
205                 exb=alpxb-wxbr;
206                 ezb=alpzb-wzbr;
207 
208                 alpx2=x2+ls*cos(yaw2);
209                 alpz2=z2+ls*sin(yaw2);
210                 ex2=alpx2-wx2r;
211                 ez2=alpz2-wz2r;
212 
213                 alpx3=x3+ls*cos(yaw3);
214                 alpz3=z3+ls*sin(yaw3);
215                 ex3=alpx3-wx3r;
216                 ez3=alpz3-wz3r;
217 
218 #pragma endregion
219 
220                 rk4.do_step(integral,w_vector,t,dt);          //(
                      funcion, estado a integrar, tiempo, paso de
                      integracion)
221 
222 //----Aplicando retardo-------------------------------------------
223 #pragma region Retardo
224                 if (contador <= MAX_RETARDO){
225                     axbr.add(w_vector[0]);
226                     azbr.add(w_vector[1]);
227                     ax2r.add(w_vector[2]);
228                     az2r.add(w_vector[3]);
229                     ax3r.add(w_vector[4]);
230                     az3r.add(w_vector[5]);
```

```
231
232                     Wxb.add(w_vector[0]);
233                     Wzb.add(w_vector[1]);
234                     Wx2.add(w_vector[2]);
235                     Wz2.add(w_vector[3]);
236                     Wx3.add(w_vector[4]);
237                     Wz3.add(w_vector[5]);
238
239                     contador = contador + 1;
240                     //cout << t << endl;
241                     goto alto;
242                 }
243                 else{
244                     axbr.remove();
245                     axbr.add(w_vector[0]);
246                     azbr.remove();
247                     azbr.add(w_vector[1]);
248
249                     ax2r.remove();
250                     ax2r.add(w_vector[2]);
251                     az2r.remove();
252                     az2r.add(w_vector[3]);
253
254                     ax3r.remove();
255                     ax3r.add(w_vector[4]);
256                     az3r.remove();
257                     az3r.add(w_vector[5]);
258
259                     wxbr=Wxb.remove();
260                          Wxb.add(w_vector[0]);
261                     wzbr=Wzb.remove();
262                          Wzb.add(w_vector[1]);
263
264                     wx2r=Wx2.remove();
265                          Wx2.add(w_vector[2]);
266                     wz2r=Wz2.remove();
267                          Wz2.add(w_vector[3]);
268
269                     wx3r=Wx3.remove();
270                          Wx3.add(w_vector[4]);
271                     wz3r=Wz3.remove();
272                          Wz3.add(w_vector[5]);
273                     }
274 #pragma endregion
275
276 //-------------------------------------------------------------
277                 Velo=retro.sistema(ax2r.Get(),az2r.Get(),yaw2,ax3r
                      .Get(),az3r.Get(),yaw3, axbr.Get(),azbr.Get(),
                      yawb);
278                 robot2.EnviarCadena(Velo[0],Velo[1],Velo[2],Velo
                      [3]);
```

```
279                 robot3.EnviarCadena(Velo[4],Velo[5],Velo[6],Velo
                      [7]);
280
281                 buffb[4]=Velo[8];
282                 buffb[5]=Velo[9];
283                 buffb[10]=Velo[10];
284                 buffb[11]=Velo[11];
285                 Bluetooth.Write(buffb,0,12);
286
287                 alto:
288 //-------------------------------------------------------------
289                 Deri2.derivar(x2,z2,yaw2);
290                 Deri3.derivar(x3,z3,yaw3);
291                 Derib.derivar(xb,zb,yawb);
292                 vel2=sqrt((Deri2.GetXp()*Deri2.GetXp())+(Deri2.
                      GetZp()*Deri2.GetZp())); //Calculo de la
                      velocidad lineal del garcia v=sqrt(xp^2+zp^2)
293                 vel3=sqrt((Deri3.GetXp()*Deri3.GetXp())+(Deri3.
                      GetZp()*Deri3.GetZp()));
294                 velb=sqrt((Derib.GetXp()*Derib.GetXp())+(Derib.
                      GetZp()*Derib.GetZp()));
295 //-------------------------------------------------------------
296 #pragma region Datos a Graficar
297
298 graficar << t << "\t" << alpx2 << "\t" << alpz2 << "\t" << yaw2 <<
      "\t" << alpx3 << "\t" << alpz3 << "\t" << yaw3 << "\t" << retro.
    GetV2() << "\t" << retro.GetW2() << "\t" << retro.GetV3() << "\t"
     << retro.GetW3() << "\t" << vel2 << "\t" << vel3 << "\t" << ax2r
    .Get() << "\t" << az2r.Get() << "\t" << ax3r.Get() << "\t" <<
    az3r.Get() << "\t" << alpxb << "\t" << alpzb << "\t" << yawb << "
    \t" << retro.GetVb() << "\t" << retro.GetWb() << "\t" << velb <<
    "\t" << axbr.Get() << "\t" << azbr.Get() << "\t" << w_vector[0]
    << "\t" << w_vector[1] << "\t" << w_vector[2] << "\t" << w_vector
    [3] << "\t" << w_vector[4] << "\t" << w_vector[5] << "\t" << wxbr
     << "\t" << wzbr << "\t" << wx2r << "\t" << wz2r << "\t" << wx3r
    << "\t" << wz3r << "\t" << exb << "\t" << ezb << "\t" << ex2 << "
    \t" << ez2 << "\t" << ex3 << "\t" << ez3 << endl;
299 #pragma endregion
300 //-------------------------------------------------------------
301                 Deri2.Iteracion();
302                 Deri3.Iteracion();
303                 Derib.Iteracion();
304 //--------Retardo de 10 ms-----------------------------------
305                 cout << t << endl;
306                 t=t+0.02;
307                 boost::chrono::duration<double> sec = boost::
                      chrono::steady_clock::now()-start;
308                 double faltante=sec.count()*1000000000;
309                 faltante=dt*1000000000-faltante;
310                 if(faltante){
```

```
311                         auto go=boost::chrono::steady_clock::now()+
                              boost::chrono::nanoseconds((long)faltante);
312                         while (boost::chrono::steady_clock::now()<go);
313                         }
314                     boost::chrono::duration<double> second = boost::
                          chrono::steady_clock::now()-start;
315                     //cout << "Time: " << second.count() << endl;
316                 }
317                 else{
318                     op=_getch();
319                     cout << "STOP" << endl;
320                 }
321     }
322     buffb[4]=0;
323     buffb[5]=0;
324     buffb[10]=0;
325     buffb[11]=0;
326     Bluetooth.Write(buffb,0,12);
327     robot2.EnviarCadena(0,0,0,0);
328     robot3.EnviarCadena(0,0,0,0);
329     Sleep(3000);
330
331     robot2.CloseSocket();
332     robot3.CloseSocket();
333     Bluetooth.Close();
334     if(!Bluetooth.IsOpen){
335         cout<<"Desconexion bluetooth exitosa"<<endl;
336       }
337     else{
338         cout<<"Aun esta conectado"<<endl;
339     }
340     Sleep(2000);
341     return 0;
342 }
```

Programa B.2: **Libreria vrpn.h**

```
1
2  #include "vrpn_Connection.h"
3  #include "vrpn_Tracker.h"
4  #include "vrpn_Button.h"
5  #include "vrpn_Analog.h"
6
7  vrpn_Connection *conexion;
8  char conexionName[128];
9  BOOLEAN conectado_vrpn = FALSE;
10
11 double x2,y2,z2,q02,q12,q22,q32;
12 double roll2,pitch2,yaw2;
13
14 double x3,y3,z3,q03,q13,q23,q33;
```

```
15 double roll3,pitch3,yaw3;
16
17 double xb,yb,zb,q0b,q1b,q2b,q3b;
18 double rollb,pitchb,yawb;
19
20
21 #define DEFAULT_VRPN_PORT   3883
22
23 void VRPN_CALLBACK handle_pos2 (void *, const vrpn_TRACKERCB t)
24 {
25     x2 = t.pos[0];
26     y2 = t.pos[1];
27     z2 = -t.pos[2];
28     q02 = t.quat[3];
29     q12 = t.quat[0];
30     q22 = t.quat[2];
31     q32 = t.quat[1];
32     roll2 = atan(((2*q22*q32)+(2*q02*q12))/((2*q02*q02)+(2*q32*q32
         )-1));
33     pitch2 = (-asin((2*q12*q32)-(2*q02*q22)));
34     yaw2 = atan2(((2*q12*q22)+(2*q02*q32)),((2*q02*q02)+(2*q12*q12
         )-1));
35 }
36
37 void VRPN_CALLBACK handle_pos3 (void *, const vrpn_TRACKERCB t)
38 {
39     x3 = t.pos[0];
40     y3 = t.pos[1];
41     z3 = -t.pos[2];
42     q03 = t.quat[3];
43     q13 = t.quat[0];
44     q23 = t.quat[2];
45     q33 = t.quat[1];
46     roll3 = atan(((2*q23*q33)+(2*q03*q13))/((2*q03*q03)+(2*q33*q33
         )-1));
47     pitch3 = (-asin((2*q13*q33)-(2*q03*q23)));
48     yaw3 = atan2(((2*q13*q23)+(2*q03*q33)),((2*q03*q03)+(2*q13*q13
         )-1));
49 }
50
51 void VRPN_CALLBACK handle_posb (void *, const vrpn_TRACKERCB t)
52 {
53     xb = t.pos[0];
54     yb = t.pos[1];
55     zb = -t.pos[2];
56     q0b = t.quat[3];
57     q1b = t.quat[0];
58     q2b = t.quat[2];
59     q3b = t.quat[1];
60     rollb = atan(((2*q2b*q3b)+(2*q0b*q1b))/((2*q0b*q0b)+(2*q3b*q3b
         )-1));
```

```
61      pitchb = (-asin((2*q1b*q3b)-(2*q0b*q2b)));
62      yawb = atan2(((2*q1b*q2b)+(2*q0b*q3b)),((2*q0b*q0b)+(2*q1b*q1b
          )-1));
63  }
```

Programa B.3: **Libreria delay.h**

```
1
2   #include <iostream>
3
4   using namespace std;
5
6   #define MAX_RETARDO 150
7
8   class Delay{
9   private:
10      //double MAX_RETARDO;
11      double fila[MAX_RETARDO];
12      double estado;
13      int array_return;
14  public:
15      Delay() {array_return=0; estado=0;}  //Constructor(Construir
          instancias e inicializar los atributos)
16      void add(double valor);
17      double remove();
18      double Get(){return estado;}
19  };
```

Programa B.4: **Libreria sistema.h**

```
1
2   #ifndef EIGEN_DONT_ALIGN_STATICALLY
3       #define EIGEN_DONT_ALIGN_STATICALLY
4   #endif
5
6   #ifndef _USE_MATH_DEFINES
7       #define _USE_MATH_DEFINES
8   #endif
9
10  #include <iostream>
11  #include <string>
12  #include <Eigen/Dense>
13  #include <math.h>
14
15  using namespace std;
16  using namespace Eigen;
17
18  #define ls 0.167    //distancia al punto alpha
19
20  #define K 0.2
21
22  #define h31 -0.4
```

```
23 #define v31   0.69282
24
25 #define h12 -0.4
26 #define v12 -0.69282
27
28 #define h13   0.4
29 #define v13 -0.69282
30
31 #define h23   0.8
32 #define v23   0
33
34 typedef Eigen::Matrix<double,12,1> Vect12d;
35
36 class Sistema{
37 private:
38     Matrix2d Atheta2Inv;
39     Matrix2d Atheta3Inv;
40     Matrix2d AthetabInv;
41     Matrix2d Conver;
42     Vector2d Control;
43     Vector2d Control3;
44     Vector2d Controlb;
45     Vector2d Vel;
46     Vector2d Vel3;
47     Vector2d Velb;
48     Vector2d VelLineal;
49     Vector2d VelLineal3;
50     Vector2d VelLinealb;
51     Vector2d Venviar;
52     Vector2d Venviar3;
53     Vector2d Venviarb;
54     double L, r, m, m1, b, b1, u1, u2;
55     double velocidadD, sentidoD, velocidadI, sentidoI;
56     double velocidadD3, sentidoD3, velocidadI3, sentidoI3;
57     double velocidadDb, sentidoDb, velocidadIb, sentidoIb;
58     Vect12d VelocidadesID;
59 public:
60     Sistema(){L=0.198374; r=0.0508; m=0.01; m1=-0.01; b=0.02; b1
         =2.614; u1= 0.5; u2= 0.5;}
61     Vect12d sistema(double ax2r, double az2r, double yaw2, double
         ax3r, double az3r, double yaw3, double axbr, double azbr,
         double yawb);
62     double GetV2(){return Vel[0];}
63     double GetW2(){return Vel[1];}
64     double GetV3(){return Vel3[0];}
65     double GetW3(){return Vel3[1];}
66     double GetVb(){return Velb[0];}
67     double GetWb(){return Velb[1];}
68 };
```

Programa B.5: **Libreria socket.h**

```
1
2  #include <iostream>
3  #include <conio.h>
4  #include <string>
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <math.h>
8  #include <sys/types.h>
9
10 #undef UNICODE
11 #define WIN32_LEAN_AND_MEAN
12
13 #include <Windows.h>
14 #include <winsock2.h>
15 #include <ws2tcpip.h>
16 #include <fcntl.h>
17 #include <errno.h>     //librerias para crear y utilizar el socket
18
19 #pragma comment(lib, "Ws2_32.lib")
20
21 using namespace std;
22
23 class sockett{
24     private:
25         WSADATA WsaData;
26         SOCKET socket1;
27         struct sockaddr_in server;
28         char *message1;
29         unsigned char buff[12];
30     public:
31         bool conectado_red;
32         sockett(){conectado_red = false;}
33         int ConectarSocket(char* IP_ADDRESS_SERVER, char*
            DEFAULT_PORT);
34         void CloseSocket();
35         void EnviarCadena(int sentiI, int velI, int sentiD, int
            velD);
36 };
```

Programa B.6: **Libreria derivada.h**

```
1
2  #ifndef EIGEN_DONT_ALIGN_STATICALLY
3      #define EIGEN_DONT_ALIGN_STATICALLY
4  #endif
5
6  #ifndef _USE_MATH_DEFINES
7      #define _USE_MATH_DEFINES
8  #endif
9
```

```
10 #include <iostream>
11 #include <string>
12 #include <boost/numeric/odeint.hpp>
13 #include <boost/math/special_functions/sign.hpp>
14 #include <Eigen/Dense>
15 #include <boost/chrono/chrono.hpp>
16 #include <math.h>
17
18 using namespace std;
19 using namespace boost::numeric::odeint;
20 using namespace boost::math;
21 using namespace Eigen;
22
23 class Derivada{
24 private:
25     int j,i;
26     double x11,x12,x13,x21,x22,x23;
27     double x11_aux[3],x12_aux[3],x13_aux[3];
28     double x21_aux[4],x22_aux[4],x23_aux[4];
29
30 public:
31     Derivada(){i=1;j=0;x11=0;x12=0;x13=0;x21=0;x22=0;x23=0;}
32     void derivar(double x, double z, double yaw);
33     double GetX(){return x11;}
34     double GetY(){return x12;}
35     double GetYaw(){return x13;}
36     double GetXp(){return x21;}
37     double GetZp(){return x22;}
38     double GetYawp(){return x23;}
39     void Iteracion(){i++;}
40 };
```

Programa B.7: **Subprograma delay.cpp**

```
1
2 #include "delay.h"
3
4 void Delay::add(double valor){
5
6     fila[array_return] = valor;
7
8     if(array_return == MAX_RETARDO-1)
9         array_return = array_return-(MAX_RETARDO-1);
10    else
11        array_return = array_return+1;
12    //cout << "Agregando  " << array_return << endl;
13 }
14
15 double Delay::remove()
16 {
17     estado = fila[array_return];
```

```
18      return estado;
19      //cout << "Leyendo  " << array_return << endl;
20  }
```

Programa B.8: **Subprograma sistema.cpp**

```
1
2   #include "sistema.h"
3
4   Vect12d Sistema::sistema(double ax2r, double az2r, double yaw2,
      double ax3r, double az3r, double yaw3, double axbr, double azbr,
      double yawb){
5   //-------------------------------------------------------
6       Atheta2Inv << cos(yaw2), sin(yaw2),
7                     -sin(yaw2)/ls, cos(yaw2)/ls;
8
9       Control << -K*(ax2r-axbr-h12),-K*(az2r-azbr-v12);
10
11      Vel = Atheta2Inv * Control;
12
13      Conver << 1,  L/2,
14                1, -L/2;
15
16      VelLineal = Conver * Vel;       //Velocidades Lineales
17
18      Venviar = (1/r) * VelLineal;    //Velocidades Angulares que
          deben ser enviadas al Robot
19
20      if (Venviar[0]<0){                                          //
          Bytes a enviar al Buff
21          velocidadD=(unsigned char)(-(100*Venviar[0]/20.4));
                // Sentido Horario/Atras -> 0,  velocidad 0-110,
              siendo 0 min y 110 max
22          sentidoD=(unsigned char)0;}
                //Como las velocidades hacia atras en el calculo
              suelen ser valores
23      if (Venviar[1]<0){
            //negativos por lo que se multiplica por (-) y se manda
          tal cual al Garcia
24          velocidadI=(unsigned char)(-(100*Venviar[1]/20.4));
25          sentidoI=(unsigned char)0;}
26      if (Venviar[0]>=0){
27          velocidadD=(unsigned char)(255-(100*Venviar[0]/20.4));
                //Sentido Antihorario/Adelante -> 255, velocidad
              160-255, siendo 160 max y 255 min
28          sentidoD=(unsigned char)255;}
                //Las velocidades hacia delante son valores positivos
               que van desde 0 hacia adelante
29      if (Venviar[1]>=0){
            //pero los valores a enviar al robot tiene que se de
          160-255, pero 160 es la velocidad max
```

```
30          velocidadI=(unsigned char)(255-(100*Venviar[1]/20.4));
                  //por ello se resta 255-Venviar, ejemplo, si Venviar
              =50, entonces 255-50=205, y seria la
31          sentidoI=(unsigned char)255;}
32
33  //-------------------------------------------------------------
34      Atheta3Inv << cos(yaw3), sin(yaw3),
35                    -sin(yaw3)/ls, cos(yaw3)/ls;
36
37      Control3 << -K*(2*ax3r-axbr-ax2r-h13-h23), -K*(2*az3r-azbr-
          az2r-v13-v23);
38
39      Vel3 = Atheta3Inv * Control3;
40
41      VelLineal3 = Conver * Vel3;
42
43      Venviar3 = (1/r) * VelLineal3;
44
45      if (Venviar3[0]<0){                                          //
          Bytes a enviar al Buff
46          velocidadD3=(unsigned char)(-(100*Venviar3[0]/20.4));
                        // Sentido Horario/Atras -> 0,  velocidad
              0-110, siendo 0 min y 110 max
47          sentidoD3=(unsigned char)0;}
                     //Como las velocidades hacia atras en el calculo
              suelen ser valores
48      if (Venviar3[1]<0){
            //negativos por lo que se multiplica por (-) y se manda
          tal cual al Garcia
49          velocidadI3=(unsigned char)(-(100*Venviar3[1]/20.4));
50          sentidoI3=(unsigned char)0;}
51      if (Venviar3[0]>=0){
52          velocidadD3=(unsigned char)(255-(100*Venviar3[0]/20.4));
                        //Sentido Antihorario/Adelante -> 255,
              velocidad 160-255, siendo 160 max y 255 min
53          sentidoD3=(unsigned char)255;}
                  //Las velocidades hacia delante son valores positivos
               que van desde 0 hacia adelante
54      if (Venviar3[1]>=0){
            //pero los valores a enviar al robot tiene que se de
          160-255, pero 160 es la velocidad max
55          velocidadI3=(unsigned char)(255-(100*Venviar3[1]/20.4));
                           //por ello se resta 255-Venviar, ejemplo,
              si Venviar=50, entonces 255-50=205, y seria la
56          sentidoI3=(unsigned char)255;}
57
58  //-------------------------------------------------------------
59      AthetabInv << cos(yawb), sin(yawb),
60                    -sin(yawb)/ls, cos(yawb)/ls;
61
62      Controlb << -K*(axbr-ax3r-h31), -K*(azbr-az3r-v31);
```

```
63
64      Velb = AthetabInv * Controlb;
65
66      Conver << 1,  L/2,
67                1, -L/2;
68
69      VelLinealb = Conver * Velb;       //Velocidades Lineales
70
71      Venviarb = (1/r) * VelLinealb;    //Velocidades Angulares que
          deben ser enviadas al Robot
72
73
74      if (Venviarb[0]<0){                                          //
          Bytes a enviar al Buff
75          velocidadDb=(unsigned char)(-(100*Venviarb[0]/20.4));
                        // Sentido Horario/Atras -> 0,  velocidad
            0-110, siendo 0 min y 110 max
76          sentidoDb=(unsigned char)0;}
                    //Como las velocidades hacia atras en el calculo
            suelen ser valores
77      if (Venviarb[1]<0){
            //negativos por lo que se multiplica por (-) y se manda
          tal cual al Garcia
78          velocidadIb=(unsigned char)(-(100*Venviarb[1]/20.4));
79          sentidoIb=(unsigned char)0;}
80      if (Venviarb[0]>=0){
81          velocidadDb=(unsigned char)(255-(100*Venviarb[0]/20.4));
                        //Sentido Antihorario/Adelante -> 255,
            velocidad 160-255, siendo 160 max y 255 min
82          sentidoDb=(unsigned char)255;}
                //Las velocidades hacia delante son valores positivos
             que van desde 0 hacia adelante
83      if (Venviarb[1]>=0){
            //pero los valores a enviar al robot tiene que se de
          160-255, pero 160 es la velocidad max
84          velocidadIb=(unsigned char)(255-(100*Venviarb[1]/20.4));
                          //por ello se resta 255-Venviar, ejemplo,
             si Venviar=50, entonces 255-50=205, y seria la
85          sentidoIb=(unsigned char)255;}
86
87      VelocidadesID << sentidoI, velocidadI, sentidoD, velocidadD,
          sentidoI3, velocidadI3, sentidoD3, velocidadD3, sentidoIb,
          velocidadIb, sentidoDb, velocidadDb;
88
89      return VelocidadesID;
90  }
```

Programa B.9: **Subprograma socket.cpp**

```
1
2  #include "socket.h"
3
4  void sockett::EnviarCadena(int _sentiI, int _velI, int _sentiD,
     int _velD){
5
6      buff[0] = 4;
7      buff[1] = 4;
8      buff[2] = 62;
9      buff[3] = 0;
10     buff[4] = _sentiI;
11     buff[5] = _velI;
12     buff[6] = 4;
13     buff[7] = 4;
14     buff[8] = 62;
15     buff[9] = 1;
16     buff[10] = _sentiD;
17     buff[11] = _velD;
18
19     send(socket1, (const char*) (&buff), sizeof(unsigned char) *
         12,0);
20 }
21
22 int sockett::ConectarSocket(char* IP_ADDRESS_SERVER, char*
     DEFAULT_PORT){
23
24         if (WSAStartup(MAKEWORD(2,2),&WsaData) != 0)
25     {
26         cout << "Inicializacion fallo. Error: "<<WSAGetLastError()
             ;
27         return 0;
28     }
29 //------------------------------------------------
30     //puts("\nCreando el socket del agente...");
31
32     if((socket1 = socket (AF_INET , SOCK_STREAM , 0)) ==
         INVALID_SOCKET)
33     {
34         cout<< "No se pudo crear el socket del agente. Error: " <<
             WSAGetLastError();
35         WSACleanup();
36         return 0;
37     }
38
39     //puts("Socket creado satisfactoriamente.\n");
40
41     server.sin_addr.s_addr = inet_addr(IP_ADDRESS_SERVER);
42     server.sin_family = AF_INET;
43     server.sin_port = htons(atoi(DEFAULT_PORT));
44
```

```
45      cout<< "\nEstableciendo conexion con el servidor remoto en..."
           <<IP_ADDRESS_SERVER;
46
47      if(connect(socket1 , (struct sockaddr *)&server , sizeof(
          server)) < 0)
48      {
49          puts("Error de conexion.");
50          WSACleanup();
51          return 0;
52      }
53      puts("\nConectado\n");
54      Sleep(3000);
55      return 1;
56  }
57
58  void sockett::CloseSocket(){
59      closesocket(socket1);
60      WSACleanup();
61      puts("\n...La comunicacion ha finalizado con exito\n");
62  }
```

Programa B.10: **Subprograma derivada.cpp**

```
1
2   #include "derivada.h"
3
4   void Derivada::derivar(double x, double z, double yaw){
5       x11 = x;                          // Estados
6       x12 = z;
7       x13 = yaw;
8       if( i > 2 )                       // Derivada sucia
9       {
10          x11_aux[2] = x11;
11          x21_aux[3] = (x11 - x11_aux[0])/(2*0.02);
12          x11_aux[0] = x11_aux[1];
13          x11_aux[1] = x11_aux[2];
14
15          x12_aux[2] = x12;
16          x22_aux[3] = (x12 - x12_aux[0])/(2*0.02);
17          x12_aux[0] = x12_aux[1];
18          x12_aux[1] = x12_aux[2];
19
20          x13_aux[2] = x13;
21          x23_aux[3] = (x13 - x13_aux[0])/(2*0.02);
22          x13_aux[0] = x13_aux[1];
23          x13_aux[1] = x13_aux[2];
24
25          if( i > 8)
26          {
27              x21 = (x21_aux[0] + x21_aux[1] + x21_aux[2] + x21_aux
                  [3])/(4);
```

```
28              x22 = (x22_aux[0] + x22_aux[1] + x22_aux[2] + x22_aux
                  [3])/(4);
29              x23 = (x23_aux[0] + x23_aux[1] + x23_aux[2] + x23_aux
                  [3])/(4);
30          }
31          else
32          {
33              x21 = x21_aux[3];
34              x22 = x22_aux[3];
35              x23 = x23_aux[3];
36          }
37          x21_aux[0] = x21_aux[1];
38          x21_aux[1] = x21_aux[2];
39          x21_aux[2] = x21_aux[3];
40
41          x22_aux[0] = x22_aux[1];
42          x22_aux[1] = x22_aux[2];
43          x22_aux[2] = x22_aux[3];
44
45          x23_aux[0] = x23_aux[1];
46          x23_aux[1] = x23_aux[2];
47          x23_aux[2] = x23_aux[3];
48      }
49      else
50      {
51          x21 = 0;
52          x22 = 0;
53          x23 = 0;
54          x11_aux[i-1] = x11;
55          x12_aux[i-1] = x12;
56          x13_aux[i-1] = x13;
57
58          x21_aux[0] = 0;
59          x21_aux[1] = 0;
60          x21_aux[2] = 0;
61          x21_aux[3] = 0;
62
63          x22_aux[0] = 0;
64          x22_aux[1] = 0;
65          x22_aux[2] = 0;
66          x22_aux[3] = 0;
67
68          x23_aux[0] = 0;
69          x23_aux[1] = 0;
70          x23_aux[2] = 0;
71          x23_aux[3] = 0;
72      }
73  }
```

Bibliografía

[1] C. Canudas, B. Siciliano, G. Bastin, B. Brogliato, G. Campion, B. D'Andrea-Novel, A. De Luca, W. Khalil, R. Lozano, R. Ortega, C. Samson, and P. Tomei. *Theory of Robot Control*. Springer-Verlag, London, 1996.

[2] R. Siegwart and I. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, Massachusetts, USA and London, England, 2004.

[3] M. Groover. *Industrial Robotics*. McGraw Hill, 1986.

[4] R. Dupuis. Search, identify and destroy: a robotic solution to urban warfare. *Land Forces Technical Staff Programme V. Royal Military College, Kingston*, 2000.

[5] Gh. Lazea and E. Lupu. Aspects on path planning for mobile robots. *Technical University of Cluj-Napoca, Automation department*, 2001.

[6] U. Nehmzow and I. R. Nourbakhsh. *Mobile robotics: A practical introduction*. Springer, 2000.

[7] R. Siegwart and I. R. Nourbakhsh. *Introduction to autonomous mobile robots*. Bradford, 2004.

[8] A. Elfes. Sonar based real world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3), 1987.

[9] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *IEEE Journal of Robotics and Automation*, 2:90–98, 1986.

[10] T. Lozano-Perez. Spatial planning: a configuration approach. *Computers, IEEE Trans.*, 32:108–120, 1983.

[11] W. Ren and R. W. Beard. Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. *Control Systems Technology, IEEE Transactions on*, 12(5):706–716, 2004.

[12] T. Eren, P. N. Belhumeur, and A. S. Morse. Closing ranks in vehicle formations based on rigidity. *IEEE Conf. Decision and Control*, 2002.

[13] R. Vidal, O. Shakernia, and S. Sastry. Formation control of nonholonomic mobile robots omnidirectional visual servoing and motion segmentation. *IEEE International Conference on Robotics and Automation*, 1:584–589, 2003.

[14] C. W. Reynolds. Flocks, herds, and schools: a distributed behavioral model. *ACM SIGGRAPH '87 Conf*, 21:25–34, 1987.

[15] J. Cortes and F. Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM J. Control Optim.*, 2003.

[16] A. S. Fukunaga, Y. U. Cao, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robotics*, 4(1):34–46, 1997.

[17] H. Yamaguchi. A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations. *Robotics and Autonomous Systems*, 43(1):257–282, 2003.

[18] W. Ren and R. W. Beard. *Distributed consensus in multi-vehicle cooperative control: theory and applications*. Springer, 2007.

[19] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[20] P. Lin and Y. Jia. Avarage consensus in networks of multiagents with both switching topology and couplin time-delay. *Physica A: Statistical Mechanics and its Applications*, 387:303–313, 2008.

[21] M. Mesbahi and F. Y. Hadegh. Formation flying of multiple spacecraft via graphs, matrix inequalities, and switching. *AIAA J. Guid., Control, Dyna.*, 24(2):369–377, 2000.

[22] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference. Proceedings of the 2005*, pages 1859–1864. IEEE, 2005.

[23] A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[24] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer-Verlag, 2001.

[25] P. Bliman and G. Ferrari-Trecate. Average consensus problems in networks of agents with delayed communications. *IEEE Conference Decision Control*, pages 7066–7071, 2005.

[26] T. Namerikawa and C. Yoshioka. Consensus control of observer-based multi-agent system with communicaton delay. *SICE Annual Conference*, pages 2414–2419, 2008.

[27] H. Yamaguchi. A distributed control scheme for multiple robotic vehicles to make group formations. *Robotics and Autonomous Systems*, 36:125–147, 2001.

[28] J. P. Desai, J. P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Trans. Robot. Automat.*, 17:905–908, 2001.

[29] S. Skogestad. Simple analytic rules for model reduction and pid controller tuning. *Journal of Process Control*, 13(4):291–309, 2003.

[30] O. J. M. Smith. Closer control of loops with deadtime. *Chem. Eng. Prog.*, 53(5):217–219, 1957.

[31] Z. J. Palmor. Time-delay compensation smith predictor and its modifications. *The control handbook*, pages 224–237, 1996.

[32] R. Olfati-Saber and R. M. Murray. Consensus problems for undirected networks of dynamics agents with communication time-delay. *Control and Dynamical Systems California Institute of Technology, Pasadena, CA*, 2003.

[33] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems*, 27(2):71–82, 2007.

[34] H. G. Tanner and D. K. Christodoulakis. State synchronization in local-interaction networks is robust with respect to time delays. *44th IEEE Conference on decision and control*, pages 4945–4950, 2005.

[35] Y. P. Huang and K. Zhou. Robust stability of uncertain time-delay systems. *IEEE Trans. on Automatic Control*, 45(11).

[36] P. Lin, Y. Jia, D. Junping, et al. Distributed consensus control for second-order agents with fixed topology and time-delay. In *Control Conference, 2007. CCC 2007. Chinese*, pages 577–581. IEEE, 2007.

[37] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[38] R. Merris. Laplacian matrices of graphs: A survey. *Linear Algebra and its Applications*, 197–198:143–176, 1994.

[39] Y. Kim and M. Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. *IEEE Trans. Automat. Contr.*, 51(1):116–120, 2006.

[40] R. Olfati-Saber and R. M. Murray. Distibuted cooperative control of multiple vehicle formations using structural potential functions. *15th IFAC Triennial World Congress*, 2002.

[41] R. Murray and R. Olfati-Saber. Consensus protocols for networks of dynamic agents. In *Proceedings of the 2003 American Controls Conference*, 2003.

[42] Y. Liu and G. Liu. Mobile manipulation using tracks of a tracked mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 948–953, St. Louis, MO, 2009.

[43] W. Ren and R. W. Beard. Consensus algorithms for double-integrator dynamics. *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*, pages 77–104, 2008.

[44] P. Lin and Y. Jia. Consensus of second-order discrete-time multi-agent systems with nonuniform time-delays and dynamically changing topologies. *Automatica*, 45(9):2154–2158, 2009.

[45] L. Moreau. Stability of continuous-time distributed consensus algorithms. *43rd IEEE Conference on Decision and Control*, 4:3998–4003, 2004.

[46] W. Ren. On consensus algorithms for double-integrator dynamics. *Automatic Control, IEEE Transactions on*, pages 1503–1509, 2008.

[47] S. I. Niculescu. *Delay effects on Stability: A Robust Control Approach*. Springer-Verlag, 2001.

[48] S. I. Niculescu, C. E. de Souza, L. Dugard, and J. M. Dion. Robust exponential stability of uncertain systems with time-varying delays. *IEEE Transactions on Automatic Control*, 43(5):743–748, 1998.

[49] NaturalPoint. Opti track. In *In http://www.naturalpoint.com/optitrack/*, 2015.