UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Control en red de vehículo autónomo.

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Alite Cerezuela, Guillermo Joaquín

Tutor/a: Cuenca Lacruz, Ángel Miguel

CURSO ACADÉMICO: 2021/2022


Universidad Politécnica de Valencia

Máster universitario en automática e informática industrial

Control en red de un vehículo autónomo

Proyecto Fin de Máster

Autor:

Guillermo Joaquín Alite Cerezuela

Tutor:

Ángel Miguel Cuenca Lacruz

Curso 2021-2022

Índice

Índice de figuras

Índice de tablas

1. Resúmenes

Resumen

Este trabajo nace como una continuación de otras investigaciones realizadas en el grupo CO3 (Comunicación y Control por Computador) del Instituto ai2 (Automática e Informática Industrial). El objetivo es integrar en este trabajo algunas de las estrategias planteadas en investigaciones previas [1], [2] y [3]. En concreto, se pretende controlar un vehículo terrestre autónomo (AGV), mediante un control en red. El vehículo modelado es 2017 Lincoln MKZ, el cual dispone de sensores para medir distintas variables, así como dispositivos de comunicación en red.

Se van a estudiar los problemas que aparecen debido a la inclusión de la red: Pérdidas de información y retardos. En primer lugar, se estudiará en qué grado afectan estos problemas al seguimiento de trayectorias. En segundo lugar, se propondrá una estrategia de control que permita solventar estos problemas. Finalmente, se compararán los resultados obtenidos antes y después de implementar la solución. El sistema de control se divide en tres partes: Parte local, parte remota y red de comunicación. La parte local hace referencia al propio vehículo, y es en el lugar en el que están ubicados los sensores y actuadores. La parte remota hace referencia al sistema de control, y la red es el canal de comunicación entre ambos.

La estrategia de control establecida se basa en: Una ley de control determinada a partir del IKIBI (Inverse Kinematic Bycicle Model), un observador del estado (Filtro de Kalman Extendido bifrecuencia), y un algoritmo de estimación h pasos hacia adelante. Con estos tres elementos se calcula una batería de acciones de control para aplicar en el futuro. De manera que si los actuadores no reciben información nueva, pueden aplicar la siguiente acción de control estimada.

Palabras clave: Control en red, vehículo terrestre autónomo (AGV), filtro de Kalman, algoritmo de predicción, IKIBI.

Resum

Aquest treball neix com una continuació d'altres investigacions realitzades en el grup CO3 (Comunicació i Control per Computador) de l'Institut ai2 (Automàtica i Informàtica Industrial). L'objectiu és integrar en aquest treball algunes de les estratègies plantejades en investigacions prèvies [1], [2] i [3]. En concret, es pretén controlar un vehicle terrestre autònom (AGV), mitjançant un control en xarxa. El vehicle modelat és 2017 Lincoln MKZ, el qual disposa de sensors per mesurar diferents variables, així com dispositius de comunicació en xarxa.

S'estudiaran els problemes que apareixen a causa de la inclusió de la xarxa: Pèrdues d'informació i retards. En primer lloc, s'estudiarà en quin grau afecten aquests problemes al seguiment de trajectòries. En segon lloc, es proposarà una estratègia de control que permeti resoldre aquests problemes. Finalment, es compararan els resultats obtinguts abans i després d'implementar la solució. El sistema de control es divideix en tres parts: Part local, part remota i xarxa de comunicació. La part local fa referència al propi vehicle, i és en el lloc on estan ubicats sensors i actuadors. La part remota fa referència al sistema de control, i la xarxa és el canal de comunicació entre ambdós.

L'estratègia de control establerta es basa en: Una llei de control determinada a partir de l'IKIBI (Inverse Kinematic Bycicle Model), un observador de l'estat (Filtre de Kalman Estès bifreqüència), i un algoritme d'estimació h passos cap endavant. Amb aquests tres elements es calcula una bateria d'accions de control per aplicar en el futur. De manera que si els actuadors no reben informació nova, poden aplicar la següent acció de control estimada.

Paraules clau: Control en xarxa, vehicle terrestre autònom (AGV), filtre de Kalman, algoritme de predicció, IKIBI.

Abstract

This work was born as a continuation of other research carried out in the CO3 group (Communication and Computer Control) of the ai2 Institute (Automation and Industrial Informatics). The objective is to integrate in this work some of the strategies proposed in previous research [1], [2] and [3]. Specifically, it is intended to control an autonomous ground vehicle (AGV), through a networked control. The modeled vehicle is 2017 Lincoln MKZ, which has sensors to measure different variables, as well as network communication devices.

The problems that appear due to the inclusion of the network are: Loss of information and delays. First, it will be studied how these problems affect the monitoring of trajectories. Secondly, a control strategy will be proposed to solve these problems. Finally, the results obtained before and after implementing the solution will be compared. The control system is divided into three parts: Local part, remote part and communication network. The local part refers to the vehicle itself, and it is in the place where sensors and actuators are located. The remote part refers to the control system, and the network is the communication channel between both.

The established control strategy is based on: A control law determined from the IKIBI (Inverse Kinematic Bicycle Model), a state observer (Dual-rate Extended Kalman Filter), and an h-step ahead estimation algorithm. With these three elements, a package of control actions is calculated to be applied in the future. So if the actuators do not receive new information, they can apply the following estimated control action.

Keywords: Networked control, autonomous ground vehicle (AGV), Kalman filter, prediction algorithm, IKIBI.

2. Introducción

La motivación de este trabajo es dar solución a un problema que hoy en día está cada vez más presente, esto es, que un vehículo autoguiado (AGV) sea capaz de seguir de manera remota una determinada trayectoria. Es un problema complejo en el que se ven implicadas muchas variables de distinta índole, que están relacionadas con toma de decisiones, seguimiento de trayectorias, control de movimiento, etc.... A lo largo de este documento se va a desarrollar la solución obtenida para la parte de control de movimiento. Es decir, se trabaja en el nivel de control, el nivel más bajo. A partir de una trayectoria dada, se ha de conseguir el mejor seguimiento posible de esta.

Este trabajo de fin de máster propone una estrategia de control en red para controlar la trayectoria de un vehículo autónomo. Para este proyecto no se ha utilizado un modelo real, sino que se ha utilizado un modelo matemático del vehículo 2017 Lincoln MKZ [2] [4] . Al utilizar un control en red se introducen ventajas como la utilización de recursos compartidos, reducción de cableado y peso de la instalación, mayor facilidad en el mantenimiento del sistema etc, pero aparecen ciertos problemas que en un control convencional no se tienen en cuenta, como retardos o pérdidas de paquetes de información [5]. Por tanto, la estructura de control que se va a formular necesita poder lidiar con estos problemas. A lo largo de este trabajo se va a hablar de las soluciones adoptadas.

Para la programación se ha utilizado el software MatLab, con su complemento Simulink, pero además se ha añadido una herramienta sobre el núcleo de MatLab, conocida como True Time, desarrollada por la universidad de Lund, Suecia [6]. Esta herramienta permite lanzar un programa de Simulink en el que se ejecutan distintas tareas a tiempo real, además permite la comunicación en red, que es justo lo que se busca para este proyecto.

La aplicación desarrollada se puede dividir en tres partes: actuadores, sensores y controlador. Los actuadores y sensores simplemente simularán los procesos

reales que pueden darse en el modelo real, mientras que en el bloque del controlador están programadas todas las funcionalidades desarrolladas. Entre estas tenemos el cálculo de las acciones de control, la estimación de estado basada en Filtro de Kalman y una etapa de predicción adicional de h pasos hacia adelante mediante modelo de referencia.

2.1. Trabajos previos del CO3

En la realización de este proyecto se ha utilizado información de la línea de investigación de Sistemas de Control en Red dentro del grupo CO3 (Comunicación y Control por Computador) del Instituto ai2 (Automática e Informática Industrial) [7] de la Universidad Politécnica de Valencia, en el que se encuentra el tutor de este trabajo Ángel Miguel Cuenca Lacruz. A partir de las publicaciones [1], [2], [3] se ha obtenido el modelo dinámico del vehículo. También se ha extraído la información necesaria para comprender el sistema de control propuesto, ya que en estos artículos se explica la forma de aplicar controladores multifrecuencia y el filtro de Kalman para experimentos muy similares.

Figura 1: Robot Lego [1]

Una de las primeras diferencias que podemos observar entre este proyecto y la investigación que se ha hecho con anterioridad dentro del grupo CO3 está en

el modelo utilizado. En trabajos previos,[1] y [3] se ha utilizado el modelo de un robot Lego (figura 1) para las simulaciones, puesto que se podía probar más tarde con el propio sistema real y validar los resultados experimentalmente, tal y como se hizo en [8]. Por su parte, en [2], ya se utilizó el modelo del vehículo Lincoln MKZ, pero todavía sin incluir el control en red.

En el caso de este proyecto se va a retomar el modelo del 2017 Lincoln MKZ para aplicarle un control remoto, en el que se tendrán que afrontar problemas típicos de red como son retardos variables y pérdida de paquetes. Este vehículo (ver figura 2) se encuentra en la Universidad de California, Berkeley, y está completamente sensorizado para conducción autónoma. En nuestro caso disponemos del modelo matemático que lo representa de manera aproximada. Con los resultados que se obtengan al finalizar este trabajo, si son satisfactorios, podrían empezar a hacerse pruebas con el vehículo real para comprobar la validez de las simulaciones, en colaboración con el MSC (Mechanical System Control) Lab liderado por el Prof. Tomizuka en la Universidad de California, Berkeley.

Figura 2: 2017 Lincoln MKZ [2]

Los resultados obtenidos para el robot Lego fueron bastante satisfactorios, es por esto que en este trabajo se va a dar el siguiente paso. Sustituir el modelo lineal del robot Lego, utilizado en [8], o el no lineal utilizado en [1] y [2], por un

modelo no lineal más complejo como es el del 2017 Lincoln MKZ. Este cambio requiere muchas modificaciones en el sistema de control, ya que a pesar de que el concepto es similar, se requieren de muchas más variables a calcular y además de mayor complejidad.

Como se ha mencionado anteriormente, este proyecto se va a focalizar en el control de movimiento para seguimiento de trayectorias en un vehículo autónomo. En lo relativo al algoritmo de planificación de trayectorias se va a utilizar el conocido algoritmo Pure Pursuit utilizado en [2]. Su misión es calcular el siguiente punto de la trayectoria a seguir por el vehículo. De este mismo artículo [2] se ha extraído la información necesaria para comprender y utilizar el filtro de Kalman. Este filtro es ampliamente utilizado en la actualidad, y nos permite obtener la predicción de la evolución del proceso, para poder anticiparnos a la respuesta del sistema.

Por último, destacar que este control es monofrecuencia, mientras que la estimación hecha por el filtro de Kalman es multifrecuencia. Esto se debe a que no en todos los instantes de control se dispone de información de los sensores, por lo que la estrategia de control debe tener en cuenta esta característica. En los artículos [1] [2] y [3] se trata ampliamente esta variación del filtro de Kalman. Con toda esta información se ha procedido a diseñar el programa en MatLab que permita realizar todas estas funcionalidades de manera satisfactoria.

3. Escenario del problema

Uno de los problemas más usuales al que nos enfrentamos en el control en red es la pérdida de información, ya sea debido a retardos en el cálculo y/o envío de las acciones de control, o por la pérdida de paquetes a la hora de enviar y recibir datos. Por ello, la estrategia de control encargada de seguir las referencias debe ser capaz de lidiar con las pérdidas de información. Para este proyecto se ha considerado que pueden existir pérdidas de paquetes y retardos en:

- Envío desde el sensor al controlador.
- Envío desde el controlador al actuador.

Se considera el caso nominal como aquella situación en la que no hay retardos ni pérdidas, es decir, como si no hubiera red. A partir del caso nominal se incluye la red con sus posibles pérdidas y/o retardos, y se analiza la pérdida de prestaciones de control que suponen. Posteriormente, se diseñan estructuras de control (etapa de h pasos hacia adelante, funciones en actuadores...) para retomar las prestaciones de control pese a retardos y/o pérdidas.

En las simulaciones se estudiarán condiciones más y menos adversas, variando la duración de los retardos y la cantidad de paquetes no enviados. Esto se puede conseguir variando el % de fallos de comunicación, o variando el valor y la distribución de los retardos. Para el caso de estudio la distribución de los retardos es exponencial negativa, como se puede apreciar en la figura 18. La distribución con la que se aproximan los retardos en las redes Ethernet, es una distribución exponencial generalizada, cuya probabilidad viene definida por la ecuación (2).

En (1) se define retardo $\tau(k)$ como la suma de los retardos considerados en el problema 1, siendo $\tau^{lr}(k)$ el retardo desde el lado local al remoto provocado por la red, $\tau^{rl}(k)$ el retardo desde el lado remoto al local y $\tau^{c}(k)$ el retardo de cómputo inducido por el controlador.

$$\tau(k) = \tau^{lr}(k) + \tau^{rl}(k) + \tau^{c}(k) \tag{1}$$

$$P[\tau(k)] = \begin{cases} \frac{1}{\phi} e^{\frac{-(\tau(k)-\eta)}{\phi}} & , \tau(k) \geq \eta \\ 0 & , \tau(k) < \eta \end{cases} \tag{2}$$

Se puede definir el valor esperado del retardo como $E[\tau(k)] = \phi + \eta$, siendo la varianza del retardo $V[\tau(k)] = \phi^2$. Para la definición de η se utilizará la mediana del retardo. A partir de η y de valores experimentales de $E[\tau(k)]$, se puede obtener

el valor de ϕ [9]. De igual manera en este tipo de comunicación mediante red pueden aparecer pérdidas de paquetes, que usualmente se modelan mediante la distribución de Bernoulli. En este caso concreto se definen las variables $d^{lr}(k)$ y $d^{rl}(k)$ como las posibles pérdidas de información desde el lado local al remoto y a la inversa, para el instante k. Por lo que la probabilidad de pérdida de paquetes se define como:

$$\begin{aligned} p^{lr} &= Pr[d^{lr}(k) = 0] \in [0,1) \\ p^{rl} &= Pr[d^{rl}(k) = 0] \in [0,1) \end{aligned} \tag{3}$$

En una estrategia clásica de control, solo se calcula una acción para cada instante de muestreo, por lo que al intentar aplicar esta metodología dentro del área de control en red, pueden existir instantes en los que no sea posible inyectar la acción de control debido a la pérdida de paquete o retraso. Para que esto no suceda, se ha planteado una estrategia de predicción en la que en cada instante de muestreo se calculan h acciones de control futuras, este paquete de acciones de control se envía al actuador. De este modo, en el caso de que en una iteración se pierda información, siempre dispondremos de una acción de control que aplicar al modelo.

Se define k como el instante actual de muestreo a periodo T, y h como el horizonte de predicción para el que se van a calcular las h acciones de control. Las predicciones hacia adelante para cualquier variable, digamos $*$, estarán denotadas de la siguiente manera: $[\hat{*}(k+1), ..., \hat{*}(k+h)]$. En la sección 4, se explicarán con detalle las diferentes variables utilizadas en el modelo no lineal del vehículo.

En la figura 3 se muestra la estrategia de control diseñada, en la que aparecen los distintos elementos del proyecto así como las variables que intercambian unos con otros. La estrategia de control se puede separar en tres partes:

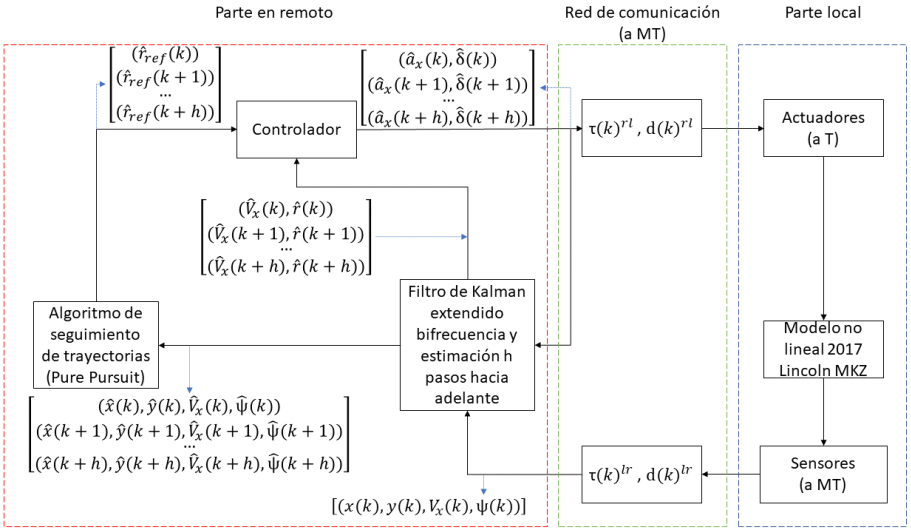

Figura 3: Diagrama de control

1. **Parte en remoto**, elementos no incluidos dentro del vehículo. Es donde se implementa la solución de control.

2. **Red de comunicación (Ethernet)**, que puede introducir retardos y pérdidas de paquetes.

3. **Parte local**, elementos que actúan sobre el vehículo o que obtienen información de él.

En la parte remota tenemos el algoritmo para calcular referencias de la trayectoria (Pure Pursuit), el filtro de Kalman extendido bifrecuencia con estimación de h pasos hacia adelante y el controlador. Esta parte es la encargada de lograr un seguimiento satisfactorio de la trayectoria, a pesar de las posibles pérdidas de información a través de la red de comunicación. El protocolo con el que se ha programado el sistema de control es Ethernet, pero se podría haber utilizado cualquier otro de los disponibles en la herramienta True Time.

En última instancia, tenemos la parte local, es decir, los elementos que están situados en el vehículo, como los actuadores y sensores. Los actuadores se encargan de aplicar las acciones de control calculadas, y de comprobar si existe algún tipo de pérdida de información, son los encargados de aplicar las acciones esti-

madas. Finalmente, los sensores son los encargados de suministrar información al sistema de control, es importante destacar que los sensores no funcionan todos a la misma velocidad. Por motivos de eficiencia de uso de la red se ha determinado fusionar la información proveniente de la sensorización y enviarla cada MT instantes de tiempo a través de la red. Por ello el filtro de Kalman solo puede corregir la información que le llega cada M periodos de muestreo. Por el mismo motivo de eficiencia, el paquete de acciones de control estimadas se enviará cada MT instantes de tiempo al actuador, es decir, se sigue la técnica de control basado en paquetes.

Otra razón importante para que no se envíe información desde los sensores a la parte en remoto es la duración de las baterías, por ejemplo, si se están usando balizas para localizar la posición del vehículo, y se envía información cada T instantes de tiempo, en vez de cada MT, la batería de estos sensores se agotaría más rápido, y además se libera la red de tráfico. Con el mismo enfoque se puede pensar en el ancho de banda de la red, en el caso de estar enviando datos cada T instantes, la red estaría más saturada, haciendo que la probabilidad de que se pierda un paquete de datos sea mayor.

Esta solución de control planteada tiene cierta versatilidad, ya que podría funcionar en otras estructuras de red, dado que el problema al que nos enfrentamos se puede enfocar de maneras muy distintas. Por ejemplo, en [1] se incluyó el controlador dentro de la parte local, dejando la parte remota para aplicaciones de niveles más altos de la estructura jerárquica del vehículo, como puede ser la toma de decisiones o la replanificación de trayectorias.

4. Modelado cinemático y dinámico

En este apartado se desarrollarán las ecuaciones del modelo cinemático y dinámico. Partiendo del modelo cinemático de la bicicleta, y utilizando ecuaciones que relacionen este modelo con las fuerzas que sufre el vehículo, así como

la variación de su ángulo de giro, se llegará a las ecuaciones que se utilizan para simular el modelo no lineal del vehículo, definido en las ecuaciones (11)-(16).

4.1. Modelo cinemático

El modelo cinemático propuesto en (4)-(7) está definido por las coordenadas (x e y) del centro de masas, y el ángulo (ψ) que forma el vehículo con respecto a los ejes de coordenadas, véase figura 4.

$$\dot{x} = V\cos(\psi + \beta) \tag{4}$$

$$\dot{y} = V\sin(\psi + \beta) \tag{5}$$

$$\dot{\psi} = \frac{V\cos(\beta)}{l_f + l_r}(\tan(\delta_f) - \tan(\delta_r)) \tag{6}$$

$$\beta = \arctan(\frac{l_f\tan(\delta_r) + l_r\tan(\delta_f)}{l_f + l_r}) \tag{7}$$

Se puede apreciar en la figura 4 la representación gráfica de las variables definidas en las ecuaciones (4)-(7), para una mejor comprensión de la relación entre ángulos y velocidades del vehículo. Como no suele haber dirección en las ruedas traseras, es habitual considerar $\delta_r = 0$. Entonces, de ahora en adelante, $\delta_f = \delta$.

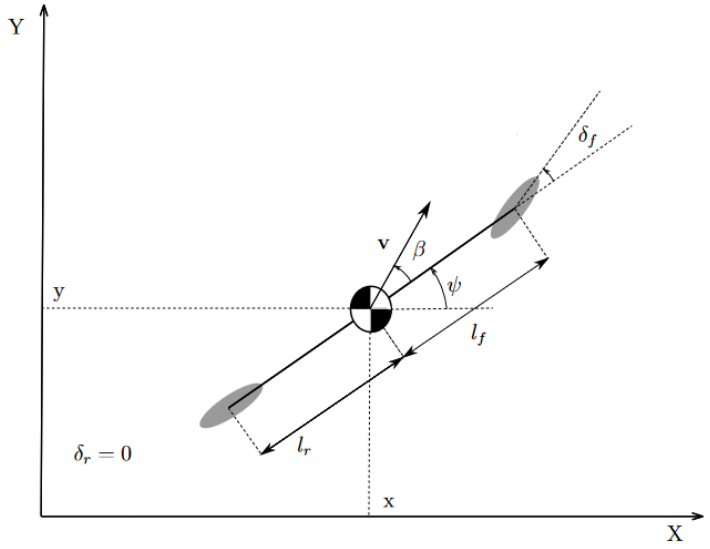

Figura 4: Modelo de bicicleta [2]

4.2. Modelo dinámico

Se necesita una ecuación (8) que relacione la variación del giro sobre el vehículo $\dot{r}$ y las fuerzas que sufre este en la rueda delantera F_{yf} y trasera F_{yr}. Para ello se utilizará la ecuación propuesta por el equipo de Stanford [10].

$$\dot{r} = \frac{ml_f \tan\delta}{I_z}(a_x - rV_y) + \frac{l_f F_{yf}}{I_z \cos\delta} - \frac{l_r F_{yr}}{I_z} \tag{8}$$

$$F_{yf} = -C_{af} \arctan(\frac{V_y + rl_f}{\text{máx}(V_x, V_{min})} - \delta) \tag{9}$$

$$F_{yr} = -C_{ar} \arctan(\frac{V_y - rl_f}{\text{máx}(V_x, V_{min})}) \tag{10}$$

Las variables V_x y V_y son las componentes en X e Y de la velocidad y se definirán respectivamente en (11)-(12). En la ecuación (8) I_z representa la inercia y m la masa del vehículo, l_f y l_r son las distancias desde el centro de gravedad al extremo frontal y trasero respectivamente. Por otro lado, C_{af} y C_{ar} son las rigideces de las ruedas en los giros. En las ecuaciones (9)-(10) aparece la velocidad mínima del vehículo V_{min}, en el modelo se define como 5 $\frac{millas}{h}$, por lo que para pasarla a $\frac{m}{s}$ se tiene que multiplicar por 0.44704. Finalmente aparecen a_x y δ que son la aceleración y el ángulo de giro de la rueda delantera. Estos dos últimos parámetros serán los utilizados como acciones de control. Una vez planteadas todas estas ecuaciones se puede extender el modelo cinemático y obtener el modelo dinámico del vehículo.

El modelo que se utiliza tanto para simular el modelo no lineal del coche, como para calcular las predicciones mediante el filtro de Kalman y la estimación de h pasos hacia adelante, es el modelo de Stanford discretizado a periodo T. A continuación se exponen sus ecuaciones y el significado de las variables. Se ha utilizado el modelo de Stanford porque es el más aceptado. Alternativamente se pueden encontrar otros modelos como el modelo Rajamani o el modelo alternativo que usó el equipo de Stanford en el DARPA Urban Challenge. Todos estos modelos se encuentran en [11] y [12].

$$V_x(k) = V_x(k-1) + T \cdot a_x(k-1) \tag{11}$$

$$V_y(k) = V_y(k-1) + T[\tan(\delta(k-1))(a_x(k-1) - r(k-1)V_y(k-1)) + \\ + \frac{F_{yf}}{m\cos(\delta(k-1))} + \frac{F_{yr}}{m} - r(k-1)V_x(k-1)] \tag{12}$$

$$x(k) = x(k-1) + T[V_x(k-1)\cos(\psi(k-1)) - V_y(k-1)\sin(\psi(k-1))] \tag{13}$$

$$y(k) = y(k-1) + T[V_x(k-1)\sin(\psi(k-1)) - V_y(k-1)\cos(\psi(k-1))] \tag{14}$$

$$\psi(k) = \psi(k-1) + T \cdot r(k-1) \tag{15}$$

$$r(k) = r(k-1) + T\left[\frac{ml_f\tan(\delta(k-1))}{I_z}(a_x(k-1) - r(k-1)V_y(k-1)) + \\ + \frac{l_f F_{yf}}{I_z\cos(\delta(k-1))} - \frac{l_r F_{yr}}{I_z}\right] \tag{16}$$

A la hora de hacer simulaciones se han programado las ecuaciones que representan el modelo dinámico del vehículo en un subsistema de Simulink, el cual dispone 2 entradas y 6 salidas. Las entradas serán las acciones de control, esto es, aceleración en el eje X a_x y ángulo de giro de la rueda delantera δ. Las acciones de control se pueden expresar como $u(k-1) = (a_x(k-1), \delta(k-1))^\top$. Las seis salidas serán el estado completo del vehículo y se definen como $\xi(k) = (V_x(k), V_y(k), x(k), y(k), \psi(k), r(k))^\top$

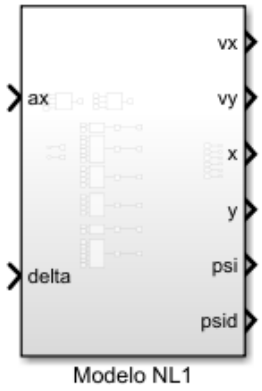

Figura 5: Subsistema del modelo no lineal del 2017 Lincoln MKZ

5. Estrategias de control utilizadas

En esta sección se va a presentar la estrategia de control utilizada. En la literatura existen diversas leyes de control para el seguimiento de trayectorias, por ejemplo, basadas en IKIBI, en MPC, etc.... El IKIBI (Inverse Kinematic Bicycle Model) es el que se va a utilizar para el control del vehículo en este caso. El objetivo es tener una ley de control que calcule δ para que sea posible seguir la referencia r_{ref}. Esto se debe a que la trayectoria con la que se va a simular es conocida, por lo que las referencias $(x, y, \psi)_{ref}$ están planeadas, y el algoritmo Pure Pursuit tiene que calcular el siguiente r_{ref}, que a su vez viene definido por una distancia mínima a recorrer (conocida como Look Ahead Distance o LAD).

Otro aspecto importante en la estrategia de control es el filtro de Kalman. Como se ha comentado, en este caso se ha utilizado una variante conocida como Filtro de Kalman extendido bifrecuencia. Se utiliza el filtro extendido debido a la naturaleza no lineal del modelo, y la variante bifrecuencia para estimar los estados del vehículo a periodo rápido con información a periodo lento. Junto con el filtro de Kalman, se explicará la función que calcula las h predicciones hacia adelante.

También se hablará de los programas que están ubicados en la parte local, en los actuadores y sensores, y de cómo gestionan la información que les llega y envían por la red. Es importante que a pesar de que no llegue información por la red en algún periodo de tiempo, el actuador sepa qué acciones de control debe ejecutar, al menos durante el horizonte h.

5.1. IKIBI (Inverse Kinematic Bicycle Model)

La ley de control utilizada es la conocida como IKIBI, la cual, además de considerar el modelo cinemático de la bicicleta, considera un control por prealimentación proporcional con ganancia K_p.

$$\delta(k+1) = (\arctan_2(\frac{r_{ref}L}{V_x(k)}) + K_p(r_{ref}(k+1) - r(k)))\gamma \tag{17}$$

También aparece en la ecuación la constante L, que es la distancia longitudinal de extremo a extremo del vehículo. Otro de los elementos que tenemos en (17) es r_{ref}, esta referencia se recalcula en cada iteración por el algoritmo de *path planning*. Se muestra también el coeficiente γ, que convierte el ángulo de la rueda en ángulo de dirección. Finalmente, destacar que el cálculo se realiza mediante la función $\arctan_2$, que representa la arcotangente para el cuarto cuadrante.

La otra acción de control considerada es la aceleración en el eje X del vehículo, y se considera constante a lo largo de toda la simulación, es decir, $a_x(k+1) = a_x(k)$, $\forall k$.

5.2. Algoritmo de seguimiento de trayectorias (Pure Pursuit)

Esta parte del control es la encargada de fijar las referencias para r, para ello este algoritmo hace uso de la referencia programada. Es decir se conocen las coordenadas en x e y de todo la trayectoria que debe seguir el vehículo. El algoritmo funciona de la siguiente manera:

1. Se parte de un objetivo inicial.

$$[x_{ref}(1), y_{ref}(1)]$$

2. Se define la distancia mínima al objetivo LAD (Look Ahead Distance).

3. Se calcula la distancia total desde la posición actual al objetivo.

$$dist_x(k) = |x(k) - x_{ref}(i)| \tag{18}$$

$$dist_y(k) = |y(k) - y_{ref}(i)| \tag{19}$$

$$dist_{tot}(k) = \sqrt{dist_x^2 + dist_y^2} \tag{20}$$

La trayectoria programada está dividida en 1200 puntos, por lo que se necesita un índice para recorrer el vector de trayectorias de referencia. Este índice se define como i, que aparece en (18)-(19).

4. Se comprueba si esta distancia es superior a la distancia mínima con el objetivo (LAD).

   a) Si la distancia total es mayor que la distancia mínima, se mantiene la referencia de x e y.

   b) Si la distancia total es menor que la distancia mínima, se pasa a la siguiente referencia, es decir se incrementa en 1 el valor de i.

5. A partir de la referencia de x e y definida en el algoritmo, se pasa a calcular la referencia de r.

$$\alpha(k) = \arctan_2\left(\frac{y_{ref}(i) - y(k)}{x_{ref}(i) - x(k)}\right) - \psi(k) \tag{21}$$

$$r_{ref}(k+1) = \frac{2V_x(k)\sin(\alpha(k))}{dist_{tot}(k)} \tag{22}$$

Una vez completado todo el proceso, el controlador dispone de toda la información necesaria para calcular la siguiente acción de control a aplicar, utilizando la ecuación 17.

5.3. Filtro de Kalman extendido bifrecuencia

El filtro de Kalman, o KF [13], se emplea como observador para obtener una predicción de los estados, si el sistema, además, es no lineal se utiliza la versión extendida del filtro, conocida como EKF. Como ya se ha comentado con anterioridad, estamos trabajando con control en red, por lo que tenemos los problemas relacionados con:

- Pérdidas de información.
- Ancho de banda restrictivo de la red.

Por tanto, en este trabajo se propone el uso de un filtro de Kalman extendido bifrecuencia (DREKF [2], esto es, Dual-rate EKF) debido a que:

- Se dispone de medidas a diferentes frecuencias, que además, podrían perderse.
- Se pretende enviar poca información por la red para no ocupar excesivo ancho de banda de la misma, y para reducir el uso de las baterías de los dispositivos conectados a ellas.

La característica principal de esta variación del filtro de Kalman se encuentra en que solo se corrigen los estados y la ganancia del filtro cuando llega información, es decir cada MT periodos de muestreo y si no hay pérdida sensor-controlador $d^{lr}(k) = 1$. Se define M como la multiplicidad del DREKF. En el caso concreto de este proyecto se ha tomado habitualmente $M = 10$, aunque también se ha hecho una prueba con $M = 5$ para comparar prestaciones (sección 7.5). En los siguientes párrafos se expone el funcionamiento más en detalle del filtro de Kalman.

En el apartado 4.2, se hace referencia a ξ, este vector se define como el vector de estados del filtro de Kalman, expresado en espacio de estados. Por otro lado también se define el vector u, el cual representa las entradas del observador (esto es, las acciones de control). Las salidas que se han especificado son $z(k) = (V_x(k), x(k), y(k), \psi(k))^\top$, dado que son las únicas que podemos obtener de la sensorización. También se considera la existencia de ruidos en el proceso y la medida, serán definidos como n_1 y n_2 respectivamente. Los ruidos se consideran Gaussianos y de media cero, se definen $Q(k)$ y $R(k)$ como la covarianza de los ruidos respectivamente.

La notación que se va a utilizar es $\hat{\xi}(j|i)$, es decir, los estados estimados para el instante j, calculados en el instante i. A continuación, se explica el funcionamiento bifrecuencia del filtro de Kalman. Dado el modelo dinámico sintetizado en (23) a partir del introducido en la sección 4.2, el filtro se formula en los siguientes pasos.

$$\begin{cases} \xi(k) = f\left(\xi(k-1), n_1(k-1), u(k-1)\right) \\ z(k) = h\left(\xi(k), n_2(k)\right) \end{cases} \tag{23}$$

1. Cálculos a periodo T para instantes $k \neq MT$, o $k = MT$ si hay pérdida sensor-controlador ($d^{lr}(k) = 0$):
   En este apartado se va a calcular la predicción del siguiente estado $\hat{\xi}(k|k-1)$ y la propagación de la covarianza $P(k|k-1)$

$$\begin{aligned} \hat{\xi}(k|k-1) &= f\left(\hat{\xi}(k-1|k-1), n_1(k-1), u(k-1)\right) \\ P(k|k-1) &= A(k)P(k-1|k-1)A(k)^\top + L(k)Q(k-1)L(k)^\top \end{aligned} \tag{24}$$

   Los estados y su covarianza se inicializan de la siguiente manera: $\hat{\xi}(0) = E[\xi(0)]$, y $P(0) = E\left[\left(\xi(0) - E[\xi(0)]\right)\left(\xi(0) - E[\xi(0)]\right)^\top\right]$, siendo $E[\cdot\,]$ la esperanza. Por otro lado están $A(k)$ y $L(k)$, que son las matrices Jacobianas calculadas para la linealización del modelo, a partir del estado actual y el ruido del proceso.

$$\begin{aligned} A(k) &= \left.\frac{\partial f}{\partial \xi}\right|_{\hat{\xi}(k-1|k-1), n_1(k-1), u(k-1)} \\ L(k) &= \left.\frac{\partial f}{\partial n_1}\right|_{\hat{\xi}(k-1|k-1), n_1(k-1), u(k-1)} \end{aligned} \tag{25}$$

   Finalmente el vector de estados y la covarianza se desplazan para realizar los cálculos en la siguiente iteración. Como se ha mencionado anteriormente, estos cálculos se realizan para los periodos en los que no se dispone de información, es decir para $k \neq MT$, o $k = MT$ si $d^{lr}(k) = 0$:

$$\begin{aligned} \hat{\xi}(k|k) &= \hat{\xi}(k|k-1) \\ P(k|k) &= P(k|k-1) \end{aligned} \tag{26}$$

2. Cálculos a periodo MT si no hay pérdida sensor-controlador, $d^{lr}(k) = 1$:
   En este caso se hace una predicción de la salida y se recalcula la ganancia del filtro a partir de las matrices anteriormente mencionadas.

$$\hat{z}(k) = h\left(\hat{\xi}(k|k-1), n_2(k)\right) \tag{27}$$

$$K(k) = P(k|k-1)H(k)^\top \left(H(k)P(k|k-1)H(k)^\top + M(k)R(k)M(k)^\top\right)^{-1} \tag{28}$$

   En este caso se utilizan $H(k)$ y $M(k)$ que representan las matrices Jacobianas calculadas para la linealización de la salida del modelo, calculada a partir del vector de estados y del ruido de medida.

$$\begin{aligned} H(k) &= \left.\frac{\partial h}{\partial \xi}\right|_{\hat{\xi}(k|k-1), n_2(k)} \\ M(k) &= \left.\frac{\partial h}{\partial n_2}\right|_{\hat{\xi}(k|k-1), n_2(k)} \end{aligned} \tag{29}$$

   En último lugar, se realiza la corrección del vector de estados $\hat{\xi}(k|k)$, y la corrección de la covarianza $P(k|k)$.

$$\begin{aligned} \hat{\xi}(k|k) &= \hat{\xi}(k|k-1) + K(k)(z(k) - \hat{z}(k)) \\ P(k|k) &= K(k)R(k)K(k)^\top + (I - K(k)H(k))P(k|k-1)(I - K(k)H(k))^\top \end{aligned} \tag{30}$$

5.4. Predicción *h* pasos hacia adelante

Como se ha expuesto en apartados anteriores, el control en red tiene pérdidas de información, por lo que se necesita un mecanismo que pueda solventar este problema. En este apartado se va a detallar la función que permite al sistema de control calcular h acciones de control hacia adelante. Es decir, en cada periodo de medida no solo se va a calcular la acción de control deseada para la siguiente

iteración, sino que también se van a calcular h ($h > M$) acciones futuras estimadas. Estas acciones de control serán utilizadas en el caso de que el actuador no reciba información en alguno de los periodos de control.

El flujo que se sigue dentro del sistema de control es el siguiente:

1. El filtro de Kalman estima el siguiente estado del proceso. En el caso de que estemos en un instante kMT, también se realiza la corrección del estado y la covarianza (30).

2. Se calcula la acción de control óptima para la siguiente iteración, a partir de (17).

3. Con la predicción de los estados y con las acciones de control se entra en la función de estimación de h pasos hacia adelante.

   a) Se toman los datos calculados anteriormente $\hat{\xi}(k)$, como el primer elemento de la siguiente iteración $\hat{\xi}(k+1)$.

   $$\hat{\xi}(k+1) = \hat{\xi}(k) \tag{31}$$

   b) Se entra en un bucle que se repite $h-1$ veces.

   b.1) En el bucle se calculan, mediante modelo de referencia (11)-(16), los estados estimados a partir de las acciones de control calculadas en la iteración previa.

   b.2) Se simula un segundo algoritmo de seguimiento de trayectorias para calcular las referencias necesarias, con las ecuaciones (18)-(22).

   b.3) Se calculan las acciones de control según (17), a partir de las referencias simuladas en el algoritmo de seguimiento de trayectorias y el estado previo. La acumulación de estados y acciones estimadas genera las matrices $\hat{\Xi}$ y $\hat{U}$, de dimensiones hx6 y hx2 respectivamente (32)-(33).

$$\hat{\Xi} = \begin{bmatrix} \hat{\xi}(k+1)^\top \\ \hat{\xi}(k+2)^\top \\ \dots \\ \hat{\xi}(k+h)^\top \end{bmatrix} \tag{32}$$

$$\hat{U} = \begin{bmatrix} \hat{u}(k+1)^\top \\ \hat{u}(k+2)^\top \\ \dots \\ \hat{u}(k+h)^\top \end{bmatrix} \tag{33}$$

Todo este proceso se expresa de manera gráfica en la figura 3, en la parte remota del sistema de control. Hay que entender este diagrama como un bucle que se repite hasta tener las h acciones de control. Una vez terminado se lanza por la red cada MT instantes de tiempo el paquete de datos con la matriz $\hat{U}$.

5.5. Sensores y actuadores

La última parte de la estrategia de control de la que se va a hablar son los sensores y actuadores. Son los encargados de aplicar las acciones de control (Actuadores) y de medir las variables deseadas del modelo (Sensores). Para las simulaciones se han programado ciertos comportamientos para la gestión de los paquetes de información que se envían por la red.

Estos *scripts* están programados en una tarea cíclica de dentro del sistema True Time y son conocidos como *kernel*. Mediante las funciones de recepción y envío de mensajes, se puede saber si se ha perdido un paquete, o se ha retrasado su envío. Por tanto, podemos identificar si el mensaje está vacío o contiene información, y actuar en consecuencia. El esquema final que se ha programado en Simulink se puede ver en la figura 6. En esta imagen se pueden apreciar las etiquetas de cada *kernel*, donde los sensores son el elemento 1 de la red, el controlador el elemento 2, y los actuadores el elemento 3. Se comunican de manera secuencial en el orden

1, 2, 3 a través de mensajes por la red Ethernet.

La herramienta True Time nos permite monitorizar la red y comprobar en qué momentos se ha realizado el envío de un paquete de datos. Esta funcionalidad está disponible desde el pin de salida de los *kernels* denominado *Schedule*, o desde el mismo pin *Schedule* del elemento que define la red Ethernet, en cuyo caso podremos ver todas las comunicaciones que ha habido por la red.

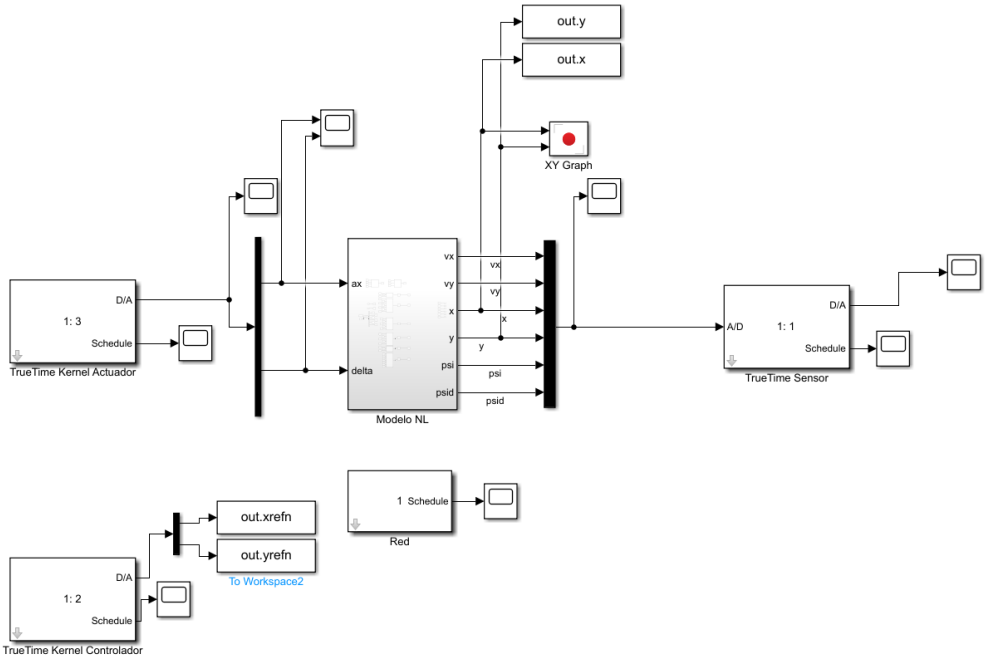

Figura 6: Programa en Simulink para simulación

5.5.1. Sensores

Los sensores son el primer elemento en la etapa de control, se encargan de registrar las señales que proporciona el modelo no lineal del 2017 Lincoln MKZ. Una vez adquiridas las señales mediante funciones de True Time, se empaquetan en un mensaje para ser enviadas.

Mediante la función *ttAnalogIn* de True Time, podemos medir las señales que se muestran en la figura 3. Con esta función será posible reproducir el muestreo de las señales y guardar en variables locales los valores. Como se ha mencionado

con anterioridad, en el modelo real los sensores pueden funcionar a distintas velocidades. Por lo que el muestreo de cada uno de los sensores es diferente. Para unificar todos estos tiempos de muestreo, el envío de los datos por la red de comunicación se realiza cada MT instantes de tiempo.

Debido a que se trabaja con una simulación, se ha modificado la característica que se menciona en el párrafo anterior. Es decir, en la simulación se muestrean todas las señales a la vez, esto se hace para simplificar el modelo de Simulink.

Finalmente, se ha simulado en el comportamiento de los sensores la pérdida de datos por la red. Se dispone de un vector de pérdidas, que ha sido generado siguiendo una distribución de Bernoulli con determinada probabilidad de pérdida. Este vector contiene una combinación de 1 y 0. En el momento de enviar los datos, se hace una comprobación de este vector, si se especifica un 1, el envío se realiza correctamente, mientras que si aparece un 0, el paquete de datos no se envía, y simula la pérdida de información.

5.5.2. Actuadores

En último lugar tenemos los actuadores. Este *kernel* recibe la batería de acciones de control calculadas por el controlador a través de la red de comunicación. Estas señales se aplican en el modelo de Simulink mediante la función *ttAnalogOut* de True Time. En este paquete están contenidas h acciones de control futuras a aplicar por los actuadores sobre las variables de control, en nuestro caso $\hat{U}$ (33).

Siguiendo la estrategia planteada, los actuadores aplican las señales de control que tienen almacenadas. Si no hay pérdida de información o retraso, los actuadores reciben cada MT instantes de tiempo un nuevo paquete de acciones de control. En el caso de no poder leer el paquete de información nueva, el actuador no se ve afectado, ya que tiene almacenadas acciones de control adicionales. En esta situación los actuadores disponen de un contador interno que determina qué acción de control es la que hay que aplicar en ese instante. Esta funcionalidad evi-

ta que si hay un retraso en el envío de datos desde el controlador, los actuadores inyecten la acción de control incorrecta.

Debido a la naturaleza del modelo y las pruebas realizadas con diferentes probabilidades de pérdidas, se ha determinado que el horizonte de predicción no es constante, por lo que en cada caso particular se tiene que realizar un estudio para determinar qué valor de h es el necesario para obtener un control satisfactorio. Se debe garantizar que $h \geq M$ para proporcionar al actuador como mínimo las M acciones a periodo T que corresponden a cada vector de medida a periodo MT.

6. Índices de coste

Con el fin de evaluar los resultados de las simulaciones, se han programado dos índices de coste para comparar el control de la trayectoria real del vehículo, con respecto a la trayectoria de referencia.

- J_1, este índice utiliza la norma-l_2 para indicar cómo se ha comportado el vehículo a la hora de seguir la trayectoria.

$$J_1 = \sum_{k=1}^{l} \min_{1 \leq k' \leq l} \sqrt{(x(k) - x_{ref}(k')))^2 + (y(k) - y_{ref}(k'))^2} \tag{34}$$

  En 34 aparece el parámetro l, que hace referencia a la cantidad de iteraciones necesarias para alcanzar el final de la trayectoria. $x(k)$ e $y(k)$, hacen referencia a la posición actual del vehículo, $x_{ref}(k')$ e $y_{ref}(k')$, es la referencia dinámica fijada por el algoritmo de seguimiento de trayectoria para la posición actual del vehículo.

- J_2 Este índice está basado en la norma-l_∞, y muestra la máxima diferencia entre el camino recorrido por el vehículo y la trayectoria deseada.

$$J_2 = \max_{1 \leq k \leq l} \left\{ \min_{1 \leq k' \leq l} \sqrt{(x(k) - x_{ref}(k'))^2 + (y(k) - y_{ref}(k'))^2} \right\} \tag{35}$$

7. Simulaciones

Esta sección está dedicada al estudio de los resultados obtenidos en las simulaciones. Se van a estudiar tres casos distintos:

- **Caso nominal:** En este caso se tiene la estructura de control inicial, es decir, sin red y sin etapa de predicción. Este es el comportamiento ideal y el deseado a la hora de incluir la red a la estructura.
- **Caso 2:** Para este estudio se añadirá la red, pero no la etapa de predicción. De manera que el tamaño de los paquetes está limitado a M acciones de control. Este caso presenta los diferentes problemas relacionados con el control en red, pérdidas de información y retardos.
  - **Caso 2.1:** Solo se añaden pérdidas de información.
  - **Caso 2.2:** Solo se añaden retardos.
  - **Caso 2.3:** Se añaden pérdidas de información y retardos.
- **Caso 3:** Ensayo en el que se tiene el control por red, y se añade la etapa de predicción de h pasos hacia adelante. El objetivo de este caso es comparar el resultado obtenido en el caso nominal y el caso 2, mediante los índices de costes definidos en (34)-(35).

7.1. Valores iniciales y trayectoria de referencia

A continuación, en la tabla 1 se muestran los valores iniciales que toman las distintas variables que intervienen en el proceso. Todas estas variables han sido definidas con anterioridad en los apartados 4 y 5.

El hecho de trabajar con la herramienta True Time nos fuerza a trabajar con las variables globales de MatLab. En este caso se ha definido una variable del tipo estructura, denominada *INICIOKERNELS*. En esta estructura están inicializadas todas las variables necesarias para los distintos *kernels* de la simulación.

Para la ejecución de las simulaciones, primero se debe crear la estructura global *INICIOKERNELS*, posteriormente se inicializan las variables en la estructura, y por último, se lanzará el modelo de Simulink que se muestra en la figura 6.

Variable	Valor inicial
M	10
K_p	0,55
γ	1
x	$0m$
y	$79m$
V_x	$5m/s$
V_y	$0m/s$
ψ	$4{,}764748rad$
r	$0rad/s$
a_x	$0{,}05m/s^2$
a_y	$-0{,}001m/s^2$
δ	$0rad$
l_f	$1{,}2m$
l_r	$1{,}65m$
C_{af}	140000
C_{af}	120000
I_z	$3270kgm^2$
V_{min}	$2{,}2352m/s$
T	$0{,}01s$
LAD	$5m$
r_{ref}	$0{,}1rad/s$

Tabla 1: Valores iniciales

En las simulaciones se ha utilizado una trayectoria con forma cuadrada. Esta trayectoria es muy exigente, ya que pretende que el vehículo realice giros de 90^{o} en las esquinas. El recorrido está dividido en 1200 puntos, esto es debido a la naturaleza dinámica del algoritmo de seguimiento de trayectorias, que en conjunto representan la figura 7.

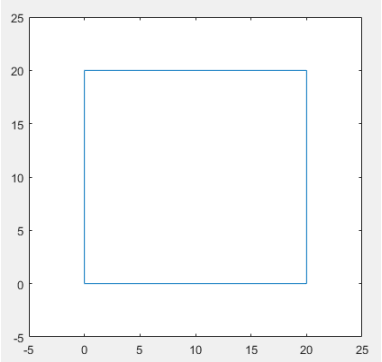

Figura 7: Trayectoria de referencia en metros

7.2. Caso nominal

El caso nominal se define como el caso ideal, en el que no tenemos los problemas relacionados con el control en red. Estos son, retardos de cómputo y/o envío en las acciones de control, o pérdidas de información entre los elementos de la red. Este experimento servirá para comparar los distintos escenarios que se proponen en los casos 2 y 3. Al no existir la red, la velocidad de comunicación con el actuador se convierte a periodo T, de esta manera siempre va a tener una acción de control disponible.

Por otro lado, al no haber pérdidas de información ni retardos, tampoco está implementada la función de estimación h pasos hacia adelante. Esto implica que tan solo se trabaja con una acción de control para cada variable de entrada al sistema. En las figuras 8 y 9 se muestran los resultados de la simulación para el caso nominal.

En la figura 8, el vehículo simulado sigue bastante bien la referencia; es en las esquinas donde se desvía en mayor medida de la referencia fijada. Este comportamiento también se aprecia en la figura 9, los datos en naranja representan la diferencia entre la posición real del vehículo y la trayectoria establecida. Se observa que hay cuatro picos que se identifican con las cuatro esquinas de la trayectoria cuadrada. Por otro lado, los datos en azul de la figura 9 representan la distancia con la referencia dinámica establecida por el algoritmo de seguimiento

de trayectorias. Su valor oscila sobre 5, ya que se ha establecido que la distancia mínima al objetivo sea $LAD = 5m$.

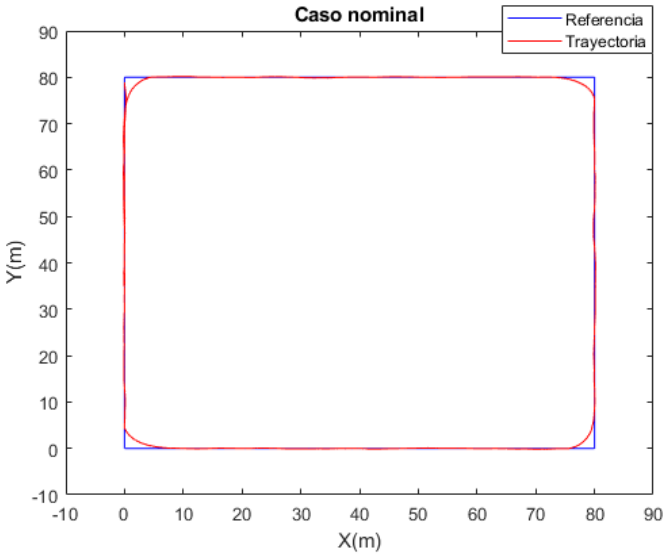

Figura 8: Trayectoria realizada en el caso nominal

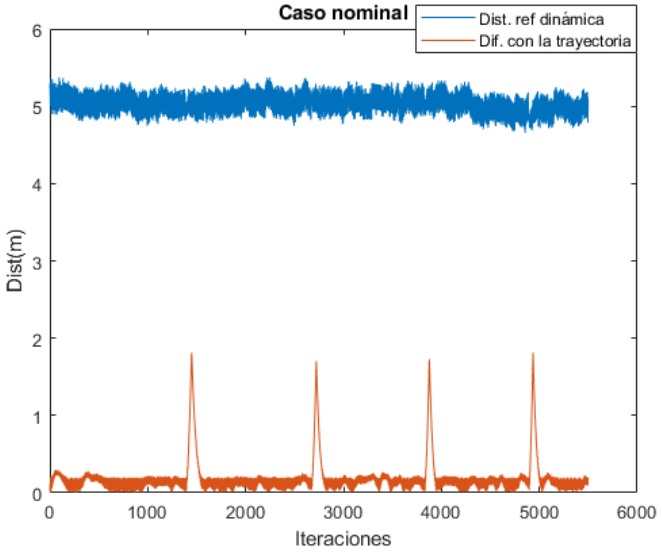

Figura 9: Distancia a referencia dinámica y desviación de la trayectoria

El valor obtenido para los índices de costes expuestos en (34) y (35) se muestra en la tabla 2. En esta tabla también se ha incluido la desviación típica de la señal denominada *Dif. con referencia dinámica*. En el caso nominal, su variación es pequeña ($Desv.típica = 0{,}1314$). En caso de tener retardos o pérdidas, este indi-

cador puede mostrar cómo de bien se mantiene la referencia dinámica establecida por el algoritmo Pure Pursuit, con valor de $LAD = 5m$.

J_1	J_2	Desv. típica
1017.7	1.8123	0.1314

Tabla 2: Índices de coste para el caso nominal

7.3. Caso 2

En este segundo caso se irán añadiendo a la simulación los problemas relacionados con el control en red. Primeramente, se evaluará el control al añadir pérdidas de información, posteriormente se cambiarán las pérdidas por los retardos, y finalmente se experimentará con ambos problemas implementados en la simulación.

Debido a la naturaleza de la red, se utiliza una distribución exponencial generalizada para los retardos. Se crea un vector de retardos con media de 0,009s, las ecuaciones de la distribución están definidas en (2).

7.3.1. Caso 2.1 (Pérdida de información)

El primer experimento consiste en añadir a la simulación las pérdidas de información. Se van a realizar distintas simulaciones, variando el % de datos que se pierden a lo largo de la simulación. Se realizará un estudio variando estos %, tanto en el envío desde los sensores al controlador, como desde el controlador a los actuadores. Para este estudio se van a utilizar pérdidas de información de aproximadamente el 15 %, 25 %, 50 % y 75 %.

Remarcar que las pérdidas de paquetes son aleatorias, por tanto, no están repartidas homogéneamente. Esto puede provocar algunas diferencias entre las simulaciones. Estos posibles fallos de comunicación pueden ocurrir cada MT periodos de control, ya que es el periodo establecido para el envío de datos.

Pérdidas del 15 %:

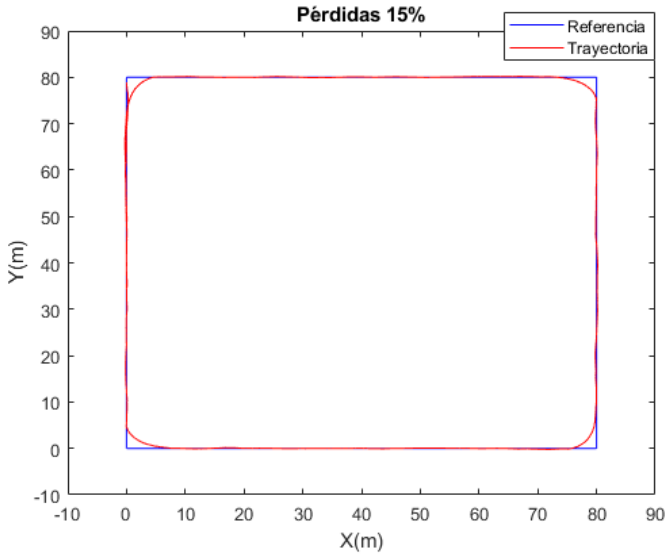

Figura 10: Trayectoria realizada con 15 % de pérdidas de información

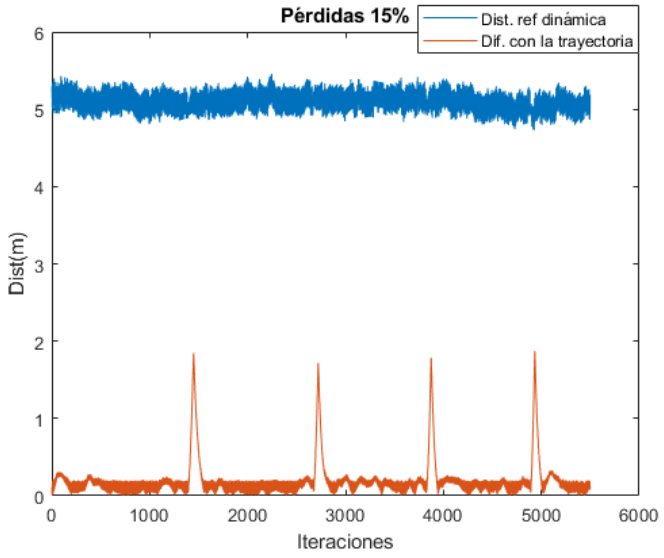

Figura 11: Distancia a referencia dinámica y desviación de la trayectoria para 15 % de pérdidas

J_1	J_2	Desv. típica
1066.9	1.873	0.128

Tabla 3: Índices de coste para el caso con 15 % de pérdidas

Pérdidas del 25 %:

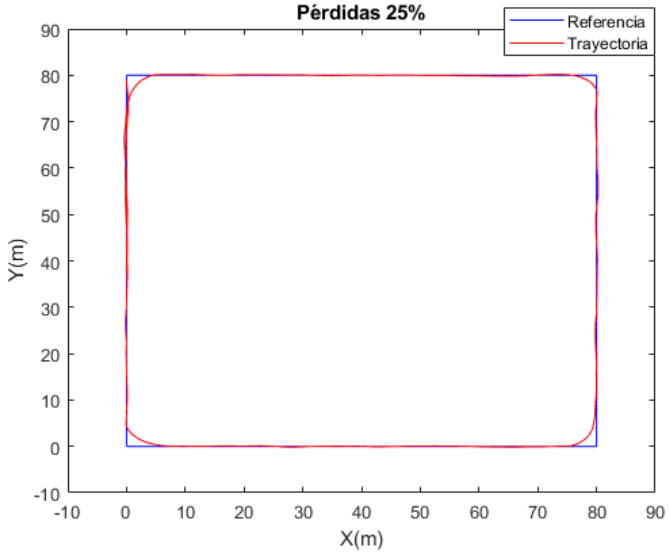

Figura 12: Trayectoria realizada con 25 % de pérdidas de información

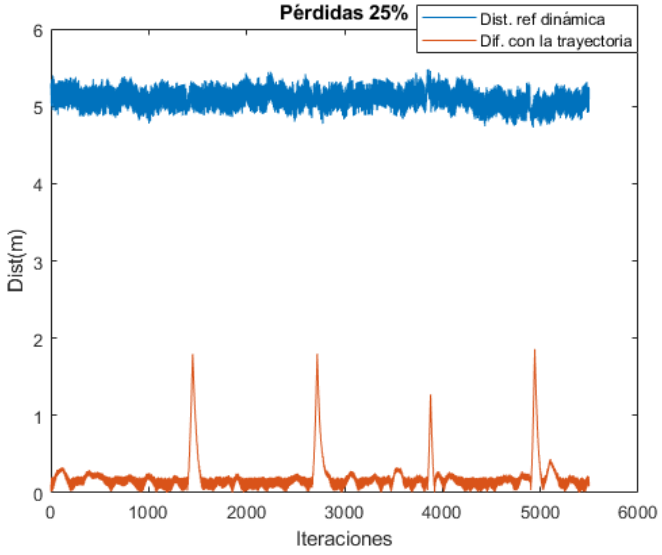

Figura 13: Distancia a referencia dinámica y desviación de la trayectoria para 25 % de pérdidas

J_1	J_2	Desv. típica
1085.3	1.8595	0.1322

Tabla 4: Índices de coste para el caso con 25 % de pérdidas

Pérdidas del 50 %:

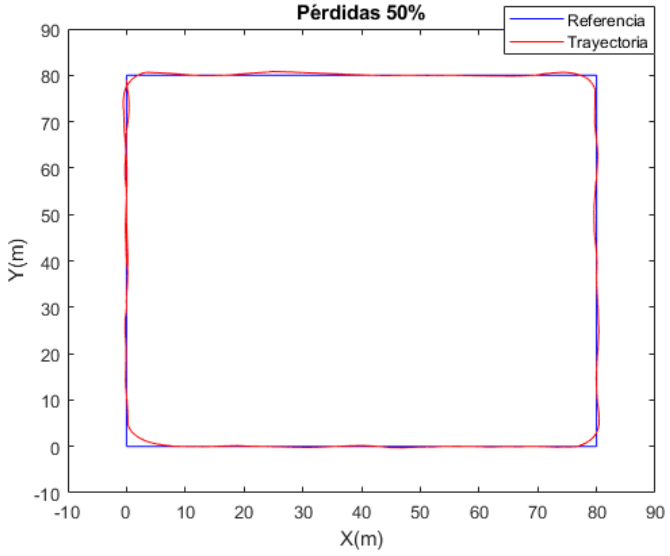

Figura 14: Trayectoria realizada con 50 % de pérdidas de información

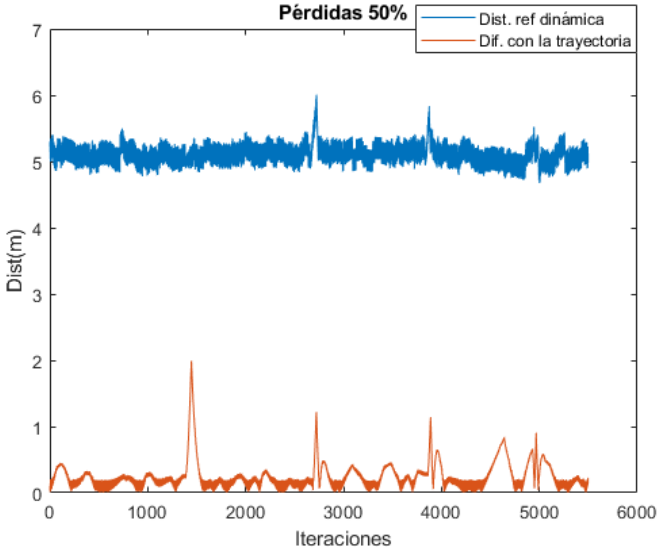

Figura 15: Distancia a referencia dinámica y desviación de la trayectoria para 50 % de pérdidas

J_1	J_2	Desv. típica
1337.9	1.9929	0.1334

Tabla 5: Índices de coste para el caso con 50 % de pérdidas

Pérdidas del 75 %:

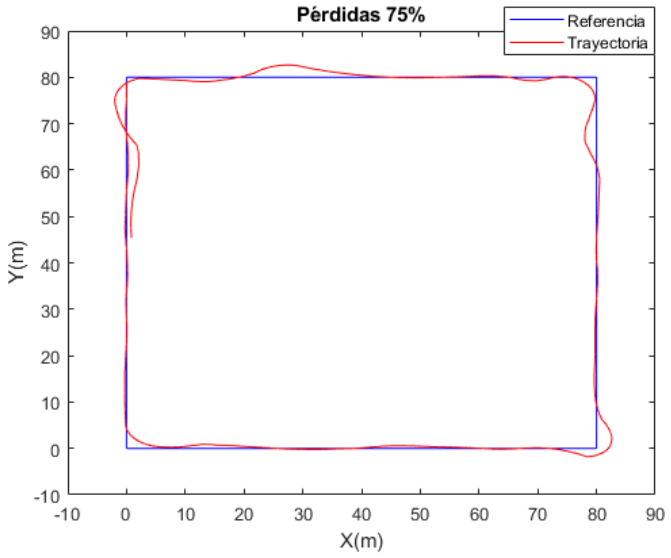

Figura 16: Trayectoria realizada con 75 % de pérdidas de información

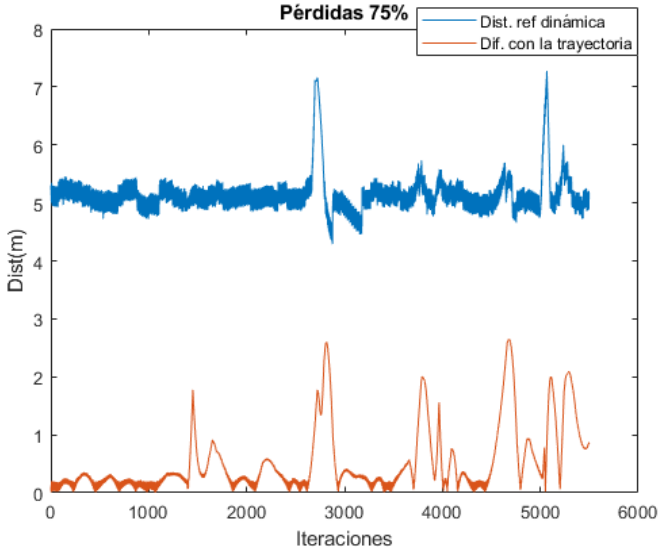

Figura 17: Distancia a referencia dinámica y desviación de la trayectoria para 75 % de pérdidas

J_1	J_2	Desv. típica
2930	2.65	0.3278

Tabla 6: Índices de coste para el caso con 75 % de pérdidas

Comparando los cuatro casos anteriores, se aprecia como al ir incrementando el % de pérdidas en el envío de datos, la trayectoria se desvía cada vez más de la referencia. En cuanto al primer índice de coste, está directamente relacionado con el % de pérdidas de datos. Este mide la diferencia entre la trayectoria que ha realizado el vehículo, y la trayectoria de referencia (figura 7). Al tener poca información, se desvía más de la referencia, y esta diferencia hace que el índice J_1 aumente. Si se aumenta hasta un 75 % de pérdidas, se puede apreciar en la figura 16 cómo el vehículo se desvía bastante de la trayectoria.

Por otro lado, el índice de coste J_2, en este caso se analiza la máxima diferencia entre la trayectoria realizada por el AGV, y la referencia. Este índice también se ve afectado por el % de pérdidas, incrementándose hasta un valor de $J_2 = 2{,}65$ en el caso del 75 % de pérdidas. Otro dato que indica un peor seguimiento de referencias dinámicas es la desviación típica de la señal azul de la figura 17,que ha aumentado desde *desv. típica* = 0,1314 hasta un valor de *desv. típica* = 0,3278, aproximadamente un 150 %, lo que muestra que la variabilidad de la distancia con la referencia dinámica ha empeorado. Es decir, el vehículo, con las acciones de control que recibe no es capaz de seguir con la misma exactitud las referencias dinámicas que marca el algoritmo Pure Pursuit.

Se puede afirmar que la pérdida de información debido a la red provoca un control con peores prestaciones. Esto puede observarse tanto en el incremento del índice de coste J_1, como en las gráficas 11-15. J_1 se ha incrementado aproximadamente un 190 % pasando de $J_1 = 1017{,}7$ a $J_1 = 2930$ (para el caso de 75 % de pérdidas). J_2 también se ha incrementado aproximadamente un 45 %, variando desde $J_2 = 1{,}8123$ hasta $J_2 = 2{,}65$ (para el caso de 75 % de pérdidas).

7.3.2. Caso 2.2 (Retardos)

En el siguiente experimento se analizará solamente el efecto de los retardos producidos por la red. Como se ha mencionado, se va a utilizar una distribución exponencial generalizada como se puede ver en la figura 18. De modo que los

retardos están concentrados en su mayoría en la media ($\mu = 0{,}009$) o por debajo de ella. El retardo más grande tiene un valor de $0{,}07s$, menor que $MT = 0{,}1s$, por lo tanto no puede existir desorden en el envío de paquetes. Este tema podría abordarse en futuras ampliaciones del proyecto.

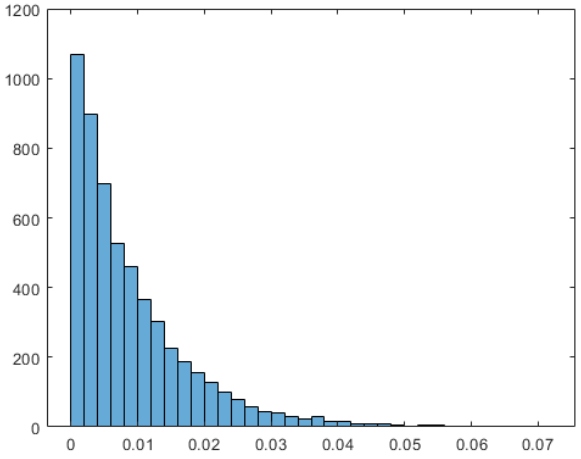

Figura 18: Histograma de los retardos implementados

En las figuras 19 y 20 se pueden ver los resultados de las simulaciones. En ellos se aprecia un peor seguimiento de la referencia comparado con el caso nominal, figuras 8 y 9. En concreto, en la figura 20 se aprecia cómo las señales tienen mayor variabilidad que el caso nominal.

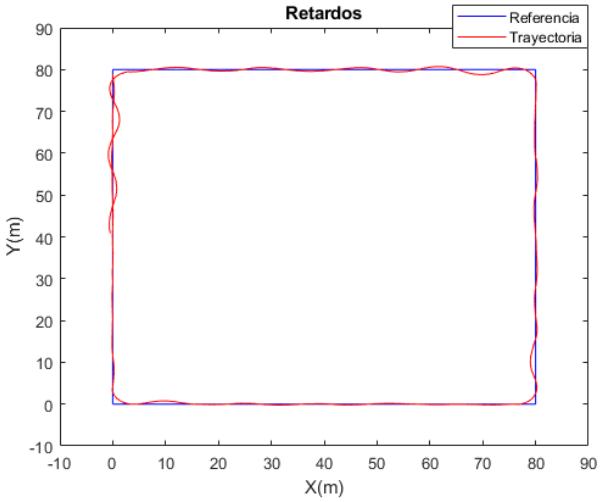

Figura 19: Trayectoria realizada con retardos incluidos

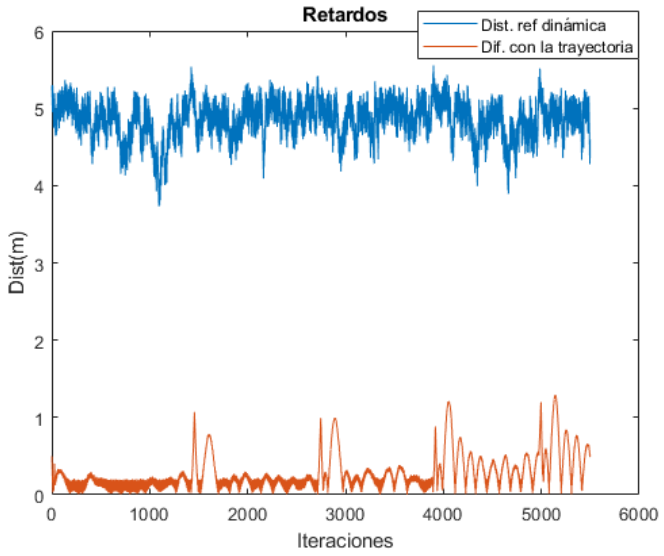

Figura 20: Distancia a referencia dinámica y desviación de la trayectoria con retardos incluidos

J_1	J_2	Desv. típica
1531.9	1.2896	0.249

Tabla 7: Índices de coste para el caso con retardos incluidos

En este experimento el índice J_1 se ha incrementado aproximadamente un 50 %, yendo de $J_1 = 1017{,}7$ a $J_1 = 1531{,}9$, esto indica que las prestaciones del control se han reducido, y como se ha mencionado antes, la trayectoria que sigue el vehículo es menos fiel a la referencia.

Por otro lado, el índice J_2 se ha reducido aproximadamente un 30 %. No obstante, si se analiza la trayectoria que traza el vehículo, se observa que debido a los retardos, tarda más en realizar el giro de la esquina y su trayectoria no se aleja tanto de la referencia. También es debido a que en este experimento, la información de los sensores siempre llega al actuador, por lo que este es capaz de corregir sus estimaciones. Sin embargo, este fenómeno causa oscilaciones en la trayectoria tras salir de la curva.

Un dato a remarcar es el incremento en la desviación típica de la señal azul en la figura 20 (Dif. con referencia dinámica), aproximadamente un 90 %. Este

incremento señala que debido a los retardos, el controlador tarda más en aplicar acciones de control, y el vehículo se acerca más que en el caso nominal a la referencia dinámica. Es decir, al incluir retardos en el control, el controlador no puede seguir tan bien las referencias dinámicas establecidas por el Pure Pursuit. Este efecto se ve acentuado en la sección 7.3.3, donde se combinan retardos y pérdidas de información.

7.3.3. Caso 2.3 (Pérdida de información y retardos)

El último experimento de esta sección 7.3, consiste en unificar los dos casos anteriores, es decir, añadir retardos y pérdidas de información. En la simulación se utilizarán los mismos retardos que en la sección 7.3.2, y de igual manera que en la sección 7.3.1 el % de pérdidas de información se irá incrementando.

En los distintos experimentos del caso 2 el controlador calcula M acciones de control, que son enviadas a los actuadores. Cuando aparece un retardo o una pérdida de información, los actuadores al haber terminado el paquete de M acciones de control, mantienen la última acción de control de la que disponen. En el siguiente caso (7.4), se estudiarán los resultados que se obtienen al incrementar el tamaño del paquete enviado desde el controlador.

Al combinar los dos problemas del control en red, se espera que los resultados tengan índices de coste peores que en los experimentos anteriores (7.3.1 y 7.3.2). Este sería el problema real al que se enfrenta el controlador, así que en la siguiente sección, se compararán los resultados obtenidos mediante la solución adoptada, con los resultados obtenidos en esta sección.

Pérdidas del 15 %

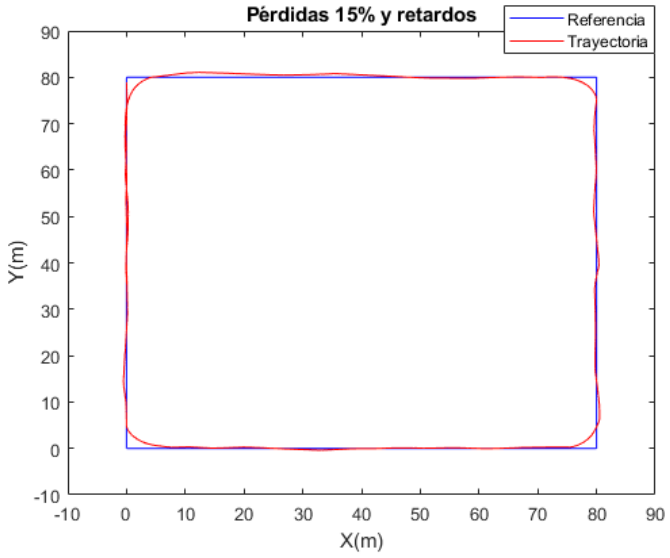

Figura 21: Trayectoria realizada con retardos incluidos y 15 % de pérdidas de información

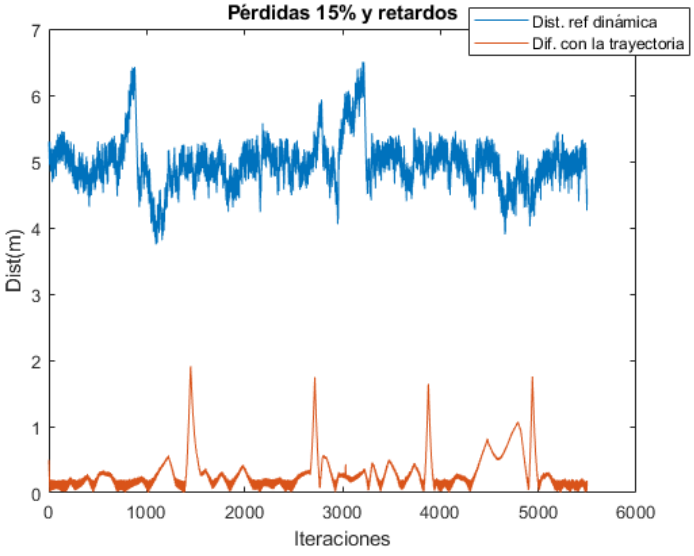

Figura 22: Distancia a referencia dinámica y desviación de la trayectoria con retardos incluidos y 15 % de pérdidas de información

J_1	J_2	Desv. típica
1613.2	1.9442	0.3686

Tabla 8: Índices de coste para el caso con retardos incluidos y 15 % de pérdidas de información

Pérdidas del 25 %

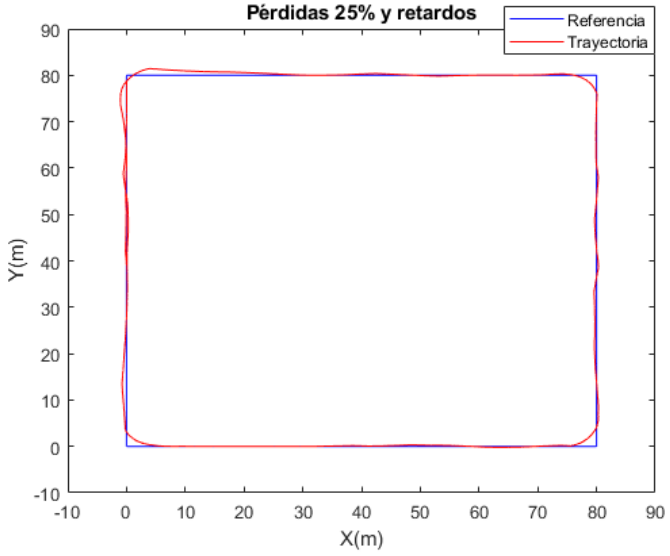

Figura 23: Trayectoria realizada con retardos incluidos y 25 % de pérdidas de información

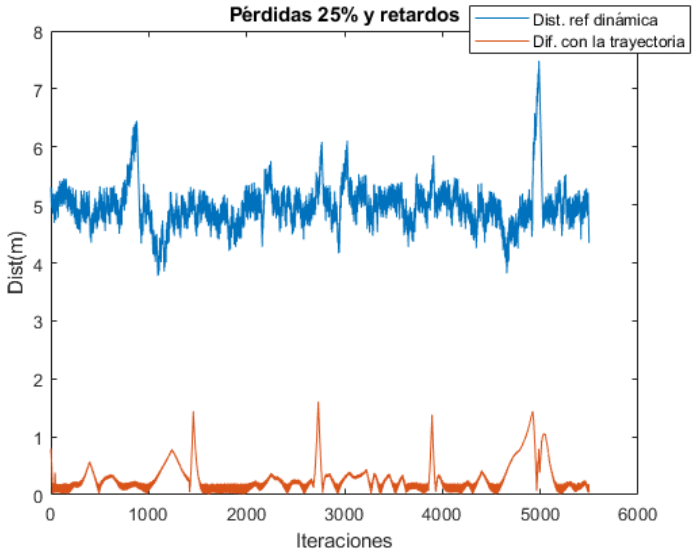

Figura 24: Distancia a referencia dinámica y desviación de la trayectoria con retardos incluidos y 25 % de pérdidas de información

J_1	J_2	Desv. típica
1892.3	1.7827	0.3397

Tabla 9: Índices de coste para el caso con retardos incluidos y 25 % de pérdidas de información

Pérdidas del 50 %

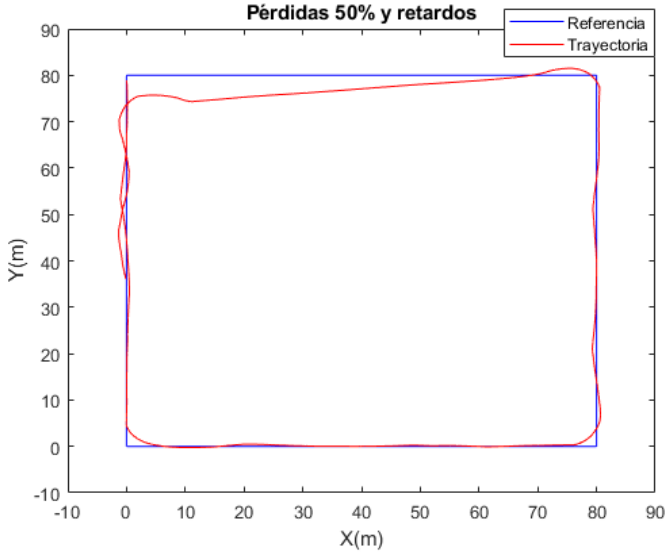

Figura 25: Trayectoria realizada con retardos incluidos y 50 % de pérdidas de información

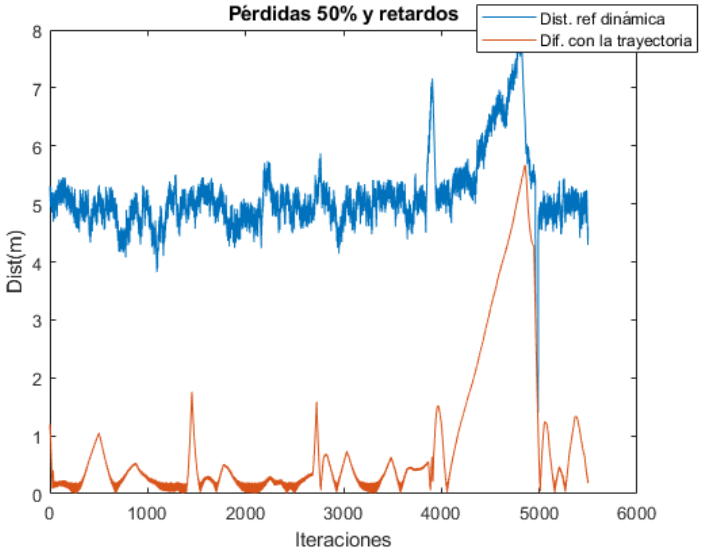

Figura 26: Distancia a referencia dinámica y desviación de la trayectoria con retardos incluidos y 50 % de pérdidas de información

J_1	J_2	Desv. típica
2948.1	2.6363	0.3624

Tabla 10: Índices de coste para el caso con retardos incluidos y 50 % de pérdidas de información

Pérdidas del 75 %

Este experimento es inestable y no se puede completar la trayectoria.

J_1	J_2	Desv. típica
∞	∞	∞

Tabla 11: Índices de coste para el caso con retardos incluidos y 75 % de pérdidas de información

Si analizamos los resultados obtenidos en las figuras 21-24 y las tablas 8-11 se puede comprobar cómo las suposiciones eran correctas, en este experimento los índices de coste aumentan con respecto a los de los casos anteriores. Además, el caso con pérdidas del 75 % se ha vuelto inestable, y el vehículo no ha sido capaz de trazar la trayectoria correctamente.

	J_1	J_2	Desviación típica	Incremento de J_1 en %
Caso nominal %	1017.7	1.9453	0.1314	-
Pérdidas 15 %	1066.9	1.873	0.128	4.8
Pérdidas 25 %	1085.3	1.8595	0.1334	6.6
Pérdidas 50 %	1337.9	1.9929	0.1312	31.4
Pérdidas 75 %	2930	2.65	0.3278	187.9
Retardos	1531.9	1.2896	0.249	50.5
Pérdidas 15 % y retardos	1613.2	1.9442	0.3686	58.5
Pérdidas 25 % y retardos	1892.3	1.7827	0.3397	85.9
Pérdidas 50 % y retardos	2948.1	2.6363	0.3624	189.6
Pérdidas 75 % y retardos	∞	∞	∞	-

Tabla 12: Índices de coste para cada experimento de Caso 2

Estudiando la tabla 12 se puede ver cómo al ir empeorando el escenario de control, los índices de coste se incrementan. Esto se traduce en un peor seguimiento de referencias, llegando incluso a inestabilizar el sistema, haciendo imposible completar la trayectoria. Se puede ver también en las figuras que representan la trayectoria realizada por el AGV, a simple vista se aprecia cómo el vehículo se separa mucho más de la referencia que en el caso nominal.

7.4. Caso 3 (Estimación h pasos hacia adelante)

Finalmente se aplicará la solución propuesta, un algoritmo de estimación h ($h > M$) pasos hacia adelante para reducir el peso de los problemas del control en red (pérdidas de información y retrasos). Anteriormente, por la red se enviaban paquetes de tamaño M en los instantes kMT, por lo que si existía una pérdida de información o retraso los actuadores repetían la última acción de control que hay en el paquete.

En lugar de enviar una batería de M acciones de control, se envía un paquete con h acciones de control, los actuadores son capaces de ir aplicando las acciones de control estimadas, a periodo T, hasta que reciban un nuevo paquete. En este caso el paquete recibido por el actuador es más grande, y nos ayuda a resolver parte de los problemas del control en red. Y es que, si el controlador sufre un retardo o una pérdida de información, el actuador puede seguir controlando el vehículo utilizando las estimaciones hechas por el controlador.

No existe una fórmula para obtener el valor de h óptimo por varios motivos. El primero es que puede darse el caso de tener experimento en el que se tiene un 50 % de pérdidas y que estén repartidas de manera homogénea, o podría darse el caso de tener las pérdidas concentradas en un intervalo de tiempo dentro del experimento. El segundo motivo es que h depende del valor de los retardos y el % de pérdidas de información. Sí que se puede establecer una relación directa, es decir, cuanto mayor sean los retardos y las pérdidas, mayor será el valor de h.

Es por ello, que se han realizado varios experimentos, variando el valor de h en el controlador para identificar qué h es el óptimo para cada caso. El criterio de selección se basa en utilizar el h que tenga menor índice J_1 en primer lugar, en el caso de igualdad en este índice, se escogerá el que menor índice J_2 tenga. A continuación se muestran los resultados obtenidos aplicando la estimación de h pasos en el futuro.

Pérdidas del 15 %

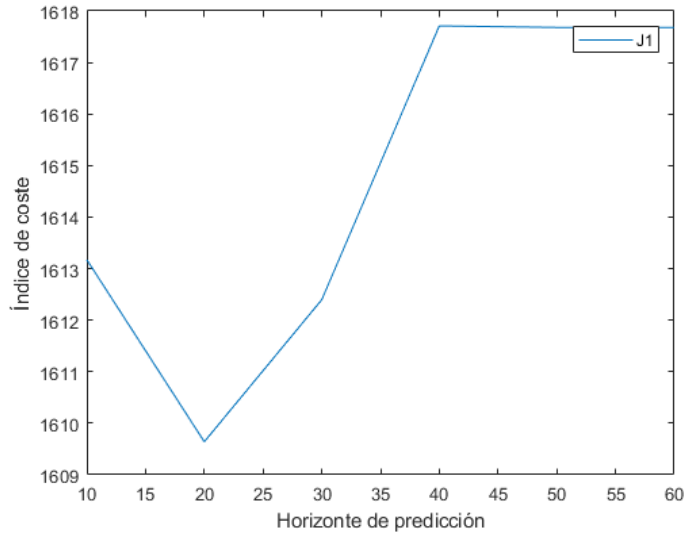

Figura 27: Variación del índice de coste J_1 en función de h

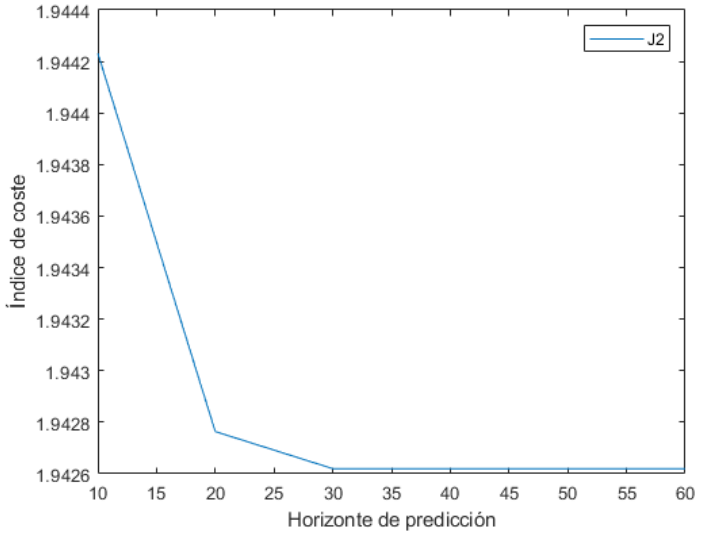

Figura 28: Variación del índice de coste J_2 en función de h

Se ha variado el valor de h desde 10 hasta 60, se puede ver en la figura 27 que el índice de coste J_1 se hace mínimo para $h = 20$, a pesar de que la variación del índice de coste es muy pequeña.

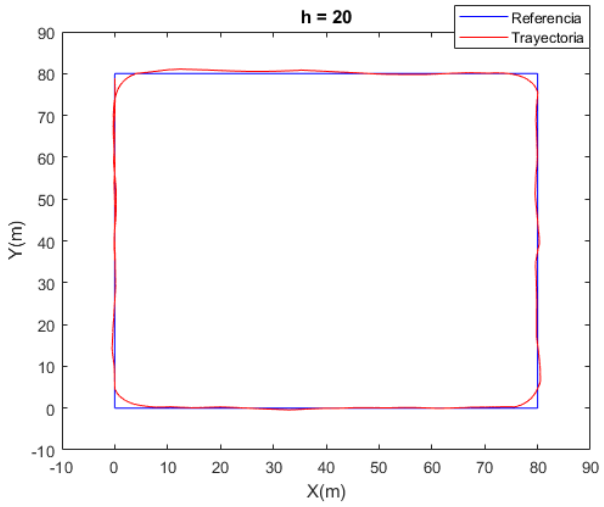

Figura 29: Trayectoria realizada con estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 15 %

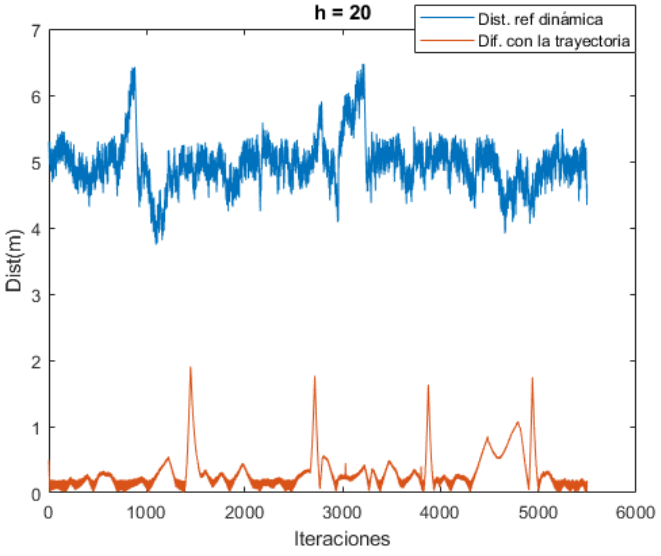

Figura 30: Distancia a referencia dinámica y desviación de la trayectoria con estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 15 %

J_1	J_2	Desv. típica	h
1609.6	1.9428	0.3703	20

Tabla 13: Índices de coste para el caso estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 15 %

Pérdidas del 25 %

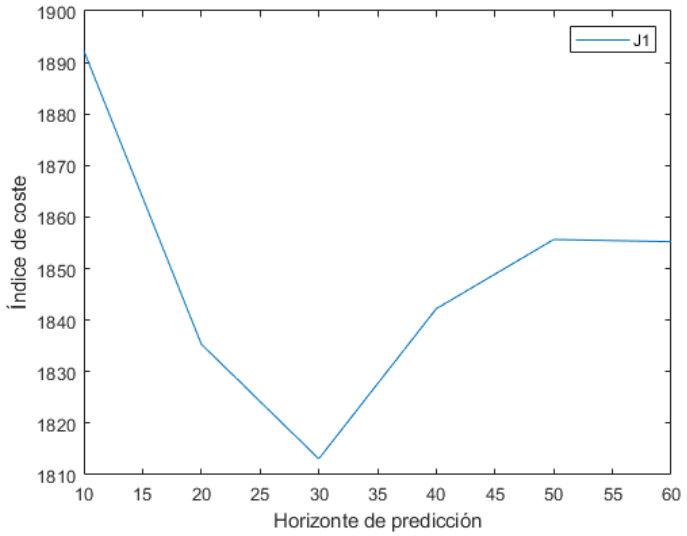

Figura 31: Variación del índice de coste J_1 en función de h

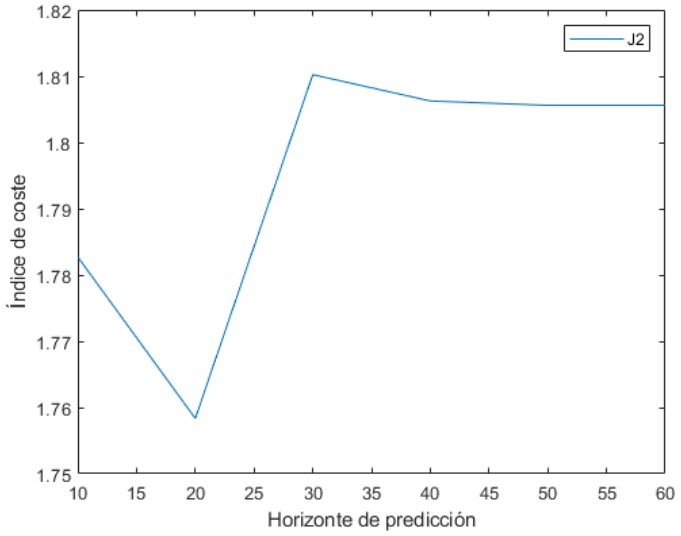

Figura 32: Variación del índice de coste J_2 en función de h

En este caso se obtiene el valor mínimo en $h = 30$, y se puede apreciar en la figura 31 que la variación del índice de coste en función de h tiene un rango mayor.

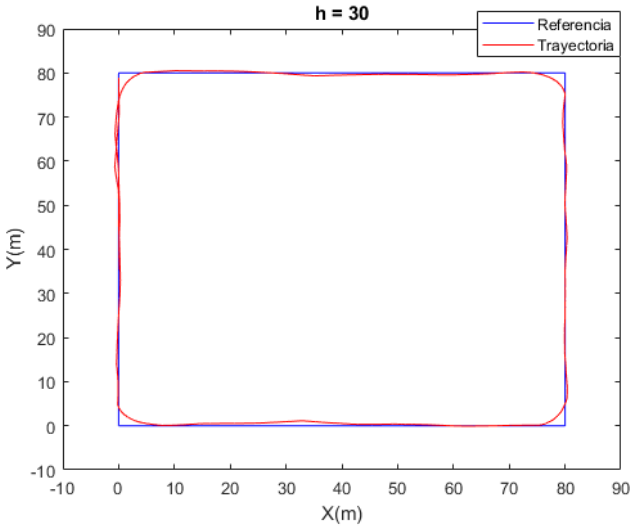

Figura 33: Trayectoria realizada con estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 25 %

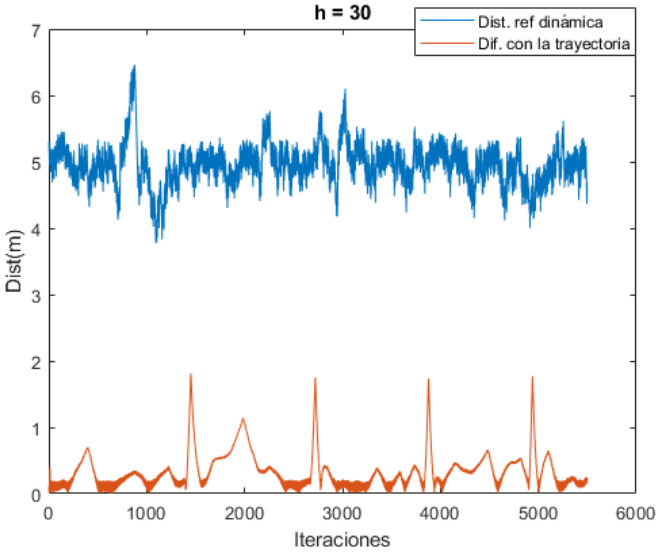

Figura 34: Distancia a referencia dinámica y desviación de la trayectoria con estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 25 %

J_1	J_2	Desv. típica	h
1813.1	1.8102	0.3111	30

Tabla 14: Índices de coste para el caso estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 25 %

Pérdidas del 50 %

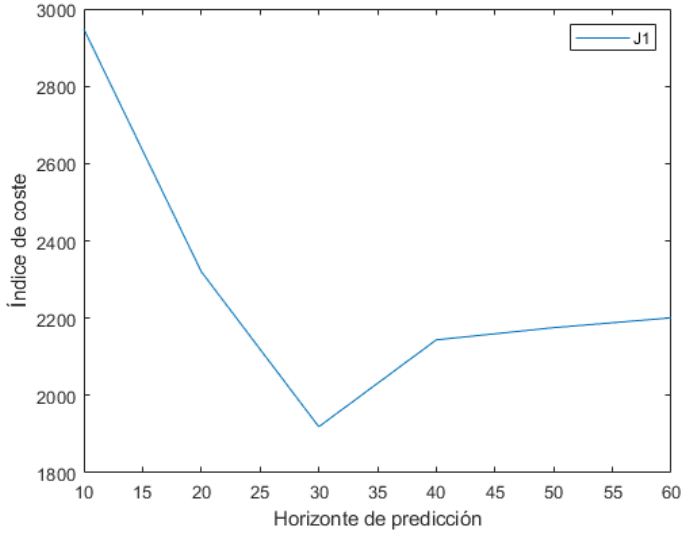

Figura 35: Variación del índice de coste J_1 en función de h

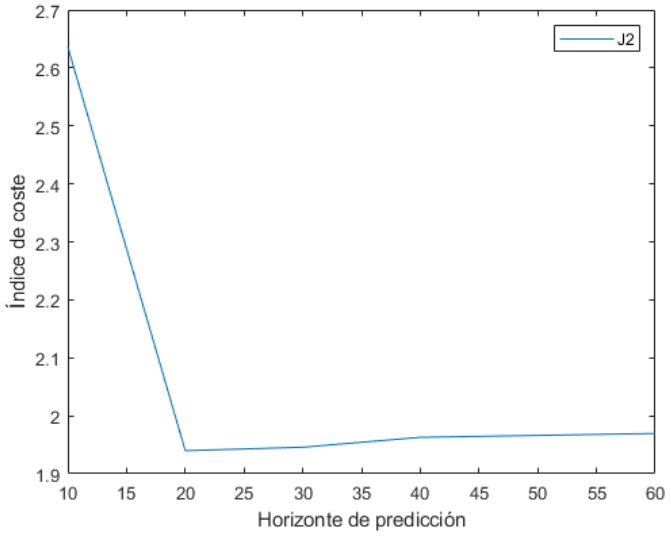

Figura 36: Variación del índice de coste J_2 en función de h

De nuevo el estudio se ha realizado hasta $h = 60$. En las figuras 35 y 36 se aprecia que el valor óptimo de nuevo es para $h = 30$, por lo que se ha escogido este valor de h para realizar el experimento.

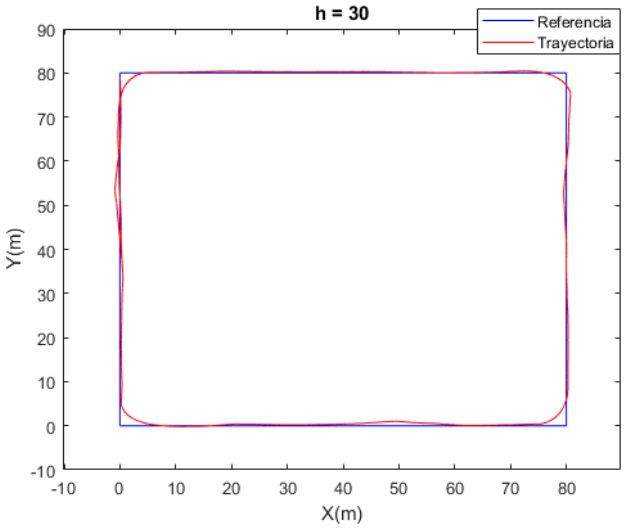

Figura 37: Trayectoria realizada con estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 50 %

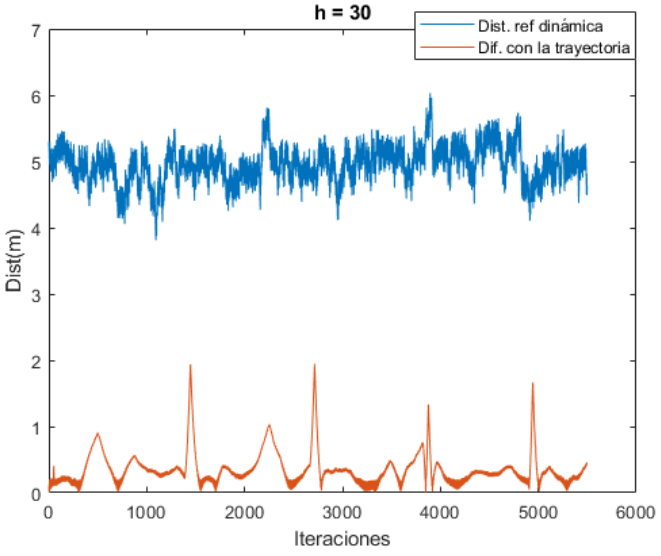

Figura 38: Distancia a referencia dinámica y desviación de la trayectoria con estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 50 %

J_1	J_2	Desv. típica	h
1919.1	1.9458	0.2811	30

Tabla 15: Índices de coste para el caso estimación $h = 30$ pasos hacia adelante, retardos y pérdidas del 50 %

Pérdidas del 75 %

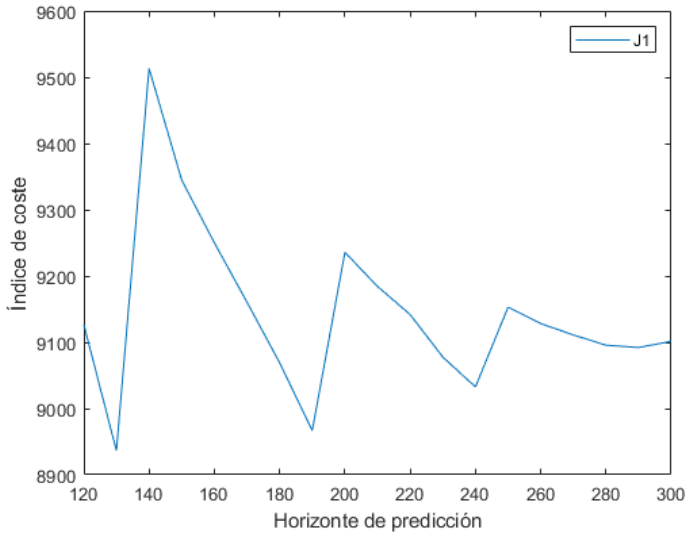

Figura 39: Variación del índice de coste J_1 en función de h

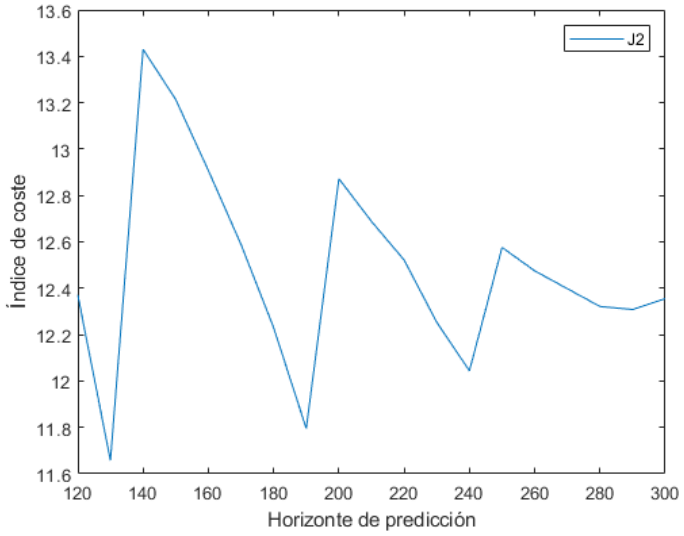

Figura 40: Variación del índice de coste J_2 en función de h

Para el caso más desfavorable, se ha determinado que para que el proceso no fuese inestable, se necesita como mínimo un valor de $h = 120$, por lo que se ha comenzado el estudio en este valor, y se ha ido incrementando hasta un valor de $h = 300$. Observando la figura 39, se determina que el valor óptimo es $h = 130$.

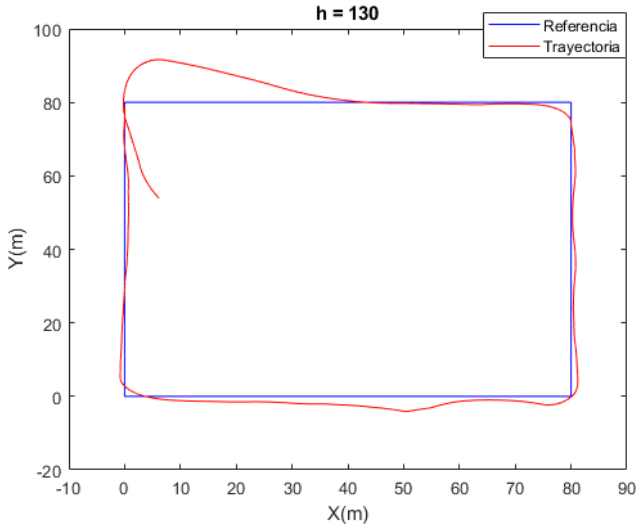

Figura 41: Trayectoria realizada con estimación $h = 130$ pasos hacia adelante, retardos y pérdidas del 75 %

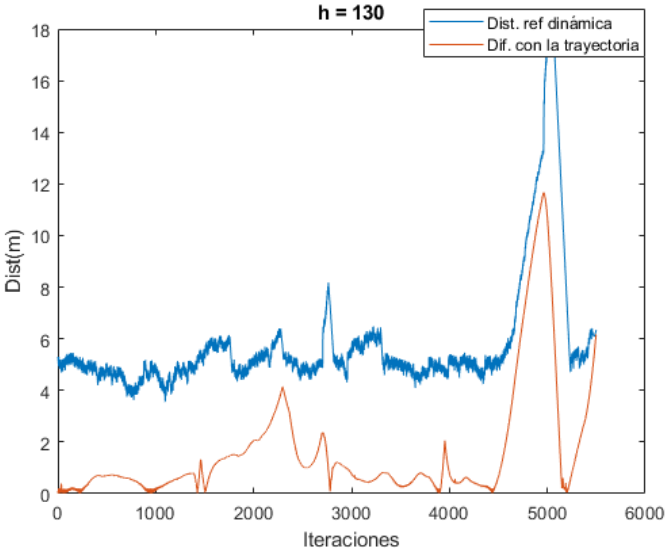

Figura 42: Distancia a referencia dinámica y desviación de la trayectoria con estimación $h = 130$ pasos hacia adelante, retardos y pérdidas del 75 %

J_1	J_2	Desv. típica	h
8936.4	11.65	2.3943	130

Tabla 16: Índices de coste para el caso estimación $h = 130$ pasos hacia adelante, retardos y pérdidas del 75 %

Analizando los resultados obtenidos, se puede afirmar que se ha mejorado el comportamiento del vehículo con respecto al obtenido en la sección previa. El índice J_1 se ha reducido bastante, indicando que el vehículo sigue de manera más fiel la trayectoria de referencia. A pesar de que el experimento con 15 % de pérdidas no ha sido representativo ya que su mejora en tanto por ciento es tan solo del 0.22 %. Por otro lado, el experimento con 25 % de pérdidas es un poco más representativo, pero el margen de mejora sigue siendo bajo 4.1 %. Es en los experimentos del 50 % y del 75 % donde podemos ver resultados interesantes.

Mediante el algoritmo de estimación se ha conseguido que experimentos de la sección previa con índices de coste muy altos, pueden llegar a tener ahora buenas prestaciones, este es el caso del experimento con 50 % de pérdidas y retardos que tiene un una mejora de su índice J_1 del 35.1 %.

Otro experimento a remarcar, es el caso con 75 % de pérdidas y retardos. En la sección anterior resultó ser inestable, pero al aplicar el algoritmo de estimación h pasos hacia adelante se vuelve estable. Sin embargo, el resultado no es del todo bueno, ya que al final se desvía bastante de la trayectoria (esquina superior izquierda de la figura 41) obteniendo un índice de coste J_1 demasiado alto.

Con todos estos datos se puede afirmar que la estrategia de control es satisfactoria y permite solventar los problemas del control en red expuestos a lo largo de este documento, las pérdidas de información y los retardos. Por último, se muestra la tabla 17, en la que se hace un resumen de todos los resultados obtenidos en esta sección.

	J_1	J_2	Desviación típica	h	% mejora J_1
Pérdidas 15 % y retardos	1609.6	1.9428	0.3703	20	0.22 %
Pérdidas 25 % y retardos	1846	1.7966	0.3109	30	4.1 %
Pérdidas 50 % y retardos	1919.1	1.9458	0.2811	30	35.1 %
Pérdidas 75 % y retardos	8936.4	11.65	2.3943	130	-

Tabla 17: Índices de coste para cada experimento

7.5. Caso de estudio con $M = 5$

En este apartado se modifica la multiplicidad del filtro a $M = 5$, esta variación hace que el controlador disponga de más información para calcular las acciones de control. Mientras que los actuadores inyectan acciones de control a periodo T.

Pérdidas del 50 % con $M = 5$

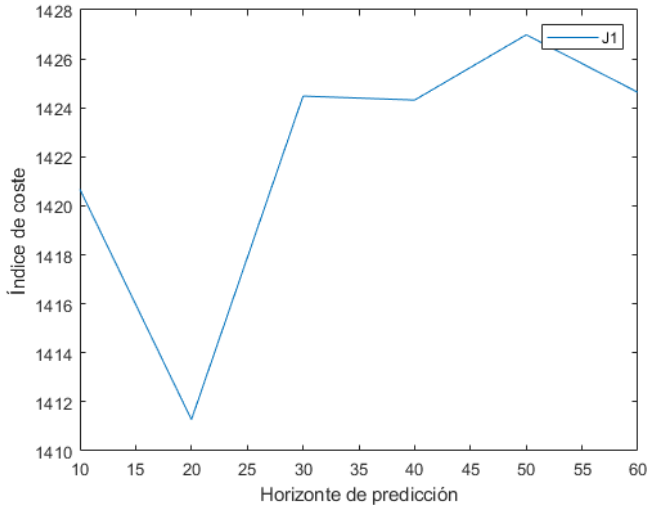

Figura 43: Variación del índice de coste J_1 en función de h para el caso con $M = 5$

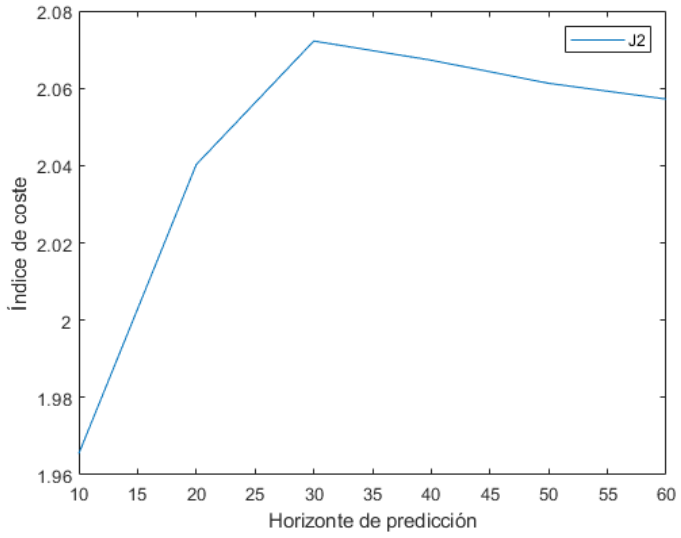

Figura 44: Variación del índice de coste J_2 en función de h para el caso con $M = 5$

Al reducir el valor de M, se puede ver en la figura 43 cómo el valor de h óptimo se ha reducido también, en este caso para las simulaciones se utilizará $h = 20$.

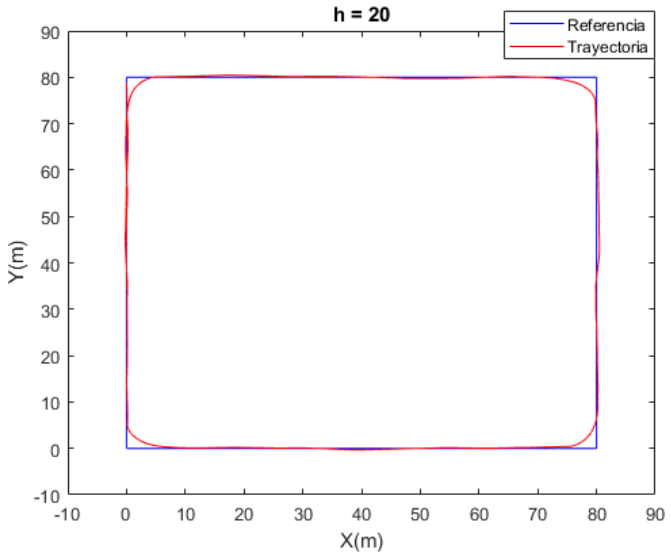

Figura 45: Trayectoria realizada con estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 50 % para el caso con $M = 5$

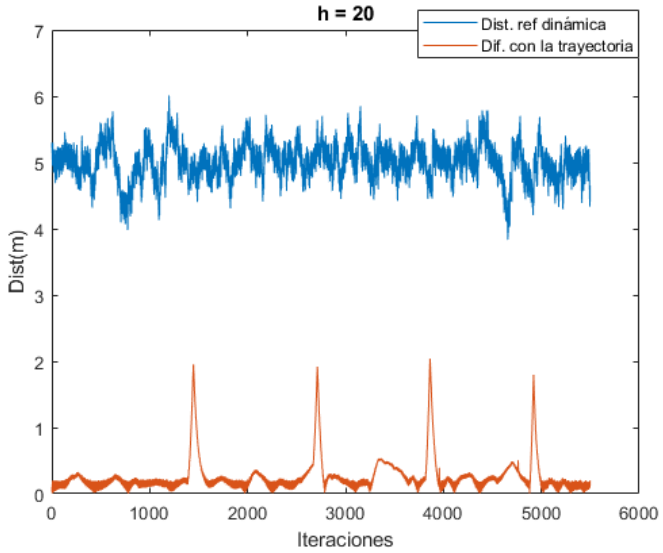

Figura 46: Distancia a referencia dinámica y desviación de la trayectoria con estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 50 % para el caso con $M = 5$

J_1	J_2	Desv. típica	h
1411.3	2.0403	0.2782	20

Tabla 18: Índices de coste para el caso estimación $h = 20$ pasos hacia adelante, retardos y pérdidas del 50 %

Si se compara este caso con su equivalente con $M = 10$ se observa que el índice de coste J_1 se ha reducido hasta un valor de $J_1 = 1411{,}3$, aproximadamente un 15 %. Además, en la figura 45 se aprecia cómo la trayectoria que sigue el vehículo es más fiel a la referencia, y no se desvía tanto en las esquinas, que es el punto crítico de la trayectoria. Otro aspecto importante es el hecho de que el valor óptimo de h se ha reducido, haciendo que el tiempo de cómputo baje, ya que no hay que calcular tantas acciones de control estimadas.

Se puede concluir que este experimento es satisfactorio y se obtienen mejores resultados que con $M = 10$. Pero no hay que olvidar los inconvenientes que esto podría traer, ya que al bajar M se incrementa el envío de datos a través de la red, en concreto, en este caso se ha multiplicado por cuatro la cantidad de mensajes a través de la red; debido a que tanto el controlador como el sensor envían el doble de mensajes cada uno. Esto provocaría que los problemas relacionados con las baterías de los sensores, la sobrecarga de la red y el ancho de banda, mencionados en el apartado 3, aparecieran con mayor frecuencia.

8. Conclusiones

En este proyecto se ha trabajado con los problemas que supone el control en red, utilizando el modelo del 2017 Lincoln MKZ, y se ha propuesto como solución la implementación de un controlador que utiliza:

- IKIBI (Inverse Kinematic Bicycle Model) como modelo cinemático y dinámico del vehículo.

- DREKF (Filtro de Kalman extendido bifrecuencia)
- Función de estimación h pasos hacia adelante

La combinación de estos tres elementos permite sensorizar las variables de salida del sistema a periodo lento MT, que el controlador calcule h acciones de control que se envían a los actuadores a periodo MT, y que finalmente los actuadores apliquen estas acciones de control a periodo T. Con esta solución se puede ver a lo largo de la sección 7, cómo se recuperan las prestaciones del control, no obstante, no se llega exactamente hasta los valores del caso nominal.

A pesar de no obtener los índices de coste del caso nominal, se aprecia una gran mejoría en las trayectorias realizadas después de implementar la función de estimación. Por otro lado, con esta solución se ha conseguido que trayectorias que antes eran irrealizables, ahora sí que se pueden completar.

Un aspecto a destacar del algoritmo de estimación h pasos hacia adelante, es que para cada caso concreto se necesita un h distinto. Es decir, para el diseño de esta estrategia de control se deben realizar múltiples simulaciones para distintos escenarios. Cuanto peor sea la comunicación, mayores retardos y/o pérdidas, por tanto, se necesitará un paquete de h acciones de control más grande. Este incremento implica un mayor tiempo de cómputo tanto en simulaciones como a la hora de implementar el control en el mundo real.

Como colofón, se puede decir que se ha solventado el problema propuesto de pérdidas de información y retardos, y se ha conseguido un control aceptable, a pesar de que no se ha llegado al nivel del caso nominal. Además, el alumno ha adquirido y reforzado conocimientos que se han ido estudiando a lo largo del máster como: Filtro de Kalman extendido bifrecuencia, funciones de estimación, algoritmos de seguimiento de trayectorias, modelos no lineales y el software True Time Lund.

9. Presupuesto

Este apartado contiene las tablas utilizadas para calcular el coste final del proyecto. Los costes del proyecto están asociados a: Las horas de trabajo del graduado, a los programas de diseño utilizados y al ordenador de torre en el que se ha trabajado. En las siguientes secciones se muestran los distintos cuadros de precios desglosados, y el precio final del proyecto. Para las amortizaciones de los materiales se ha considerado que un año equivale a 249 días laborables con una jornada de 8h, por lo que al año se dispone de 1992h laborables.

9.1. Cuadro de precios Nº1: Mano de obra

Código	Descripción	Precio (€/h)
MO.1	Graduado en Ingeniería en Tecnologías industriales	35,00

Figura 47: Cuadro de precios Nº1: Mano de obra

9.2. Cuadro de precios Nº2: Materiales y amortizaciones

Código	Descripción	Precio (€)	Precio (€/año)	Precio (€/h)
Software				
MAT.S1	MatLab	x	840,00	0,42
MAT.S2	True Time Lund	x	x	x
MAT.S3	Overleaf (Editor de LaTeX)	x	x	x
Hardware				
MAT.H1	Ordenador de torre	1400,00	350,00	0,18

Figura 48: Cuadro de precios Nº2: Materiales y amortizaciones

Para el software MatLab se ha considerado una amortización de un año, en el caso del ordenador de torre se ha establecido una vida útil de 4 años.

9.3. Cuadro de precios Nº3: Precios parciales

Código	Unidades	Descripción	Medición	Precio	Importe
Capítulo 1. Estudio de algoritmos y diseño de la estrategia de control					
UO1	Ud	UO1: Comprensión de los disitntos algoritmos a utilizar en la estrategia de control, y el posterior diseño de esta.	1,00	1076,38	1076,38
		Total Capítulo 1			**1.076,38 €**
Capítulo 2. Programación en MatLab y True Time Lund					
UO2.1	Ud	UO2.1: Programación del modelo no lineal del AGV.	1,00	1089,28	1089,28
UO2.2	Ud	UO2.2: Programación mediante True Time Lund del controlador, actuadores y sensores.	1,00	5083,31	5083,31
		Total Capítulo 2			**6.172,59 €**
Capítulo 3. Simulaciones					
UO3	Ud	UO3: Corrección de errores y extracción de datos mediante simulaciones en el entorno de MatLab y True Time Lund	1,00	2152,75	2152,75
		Total Capítulo 3			**2.152,75 €**
Capítulo 4. Redacción de la memoria del proyecto					
UO4	Ud	UO4: Recopilación y redacción de datos obtenidos en las simulaciones.	1,00	1445,20	1445,20
		Total Capítulo 4			**1.445,20 €**

Figura 49: Cuadro de precios Nº3: Precios parciales

9.4. Cuadro de precios Nº4: Precios descompuestos

Capítulo 1. Estudio de algoritmos y diseño de la estrategia de control					
Código	**Unidades**	**Descripción**	**Rendimiento**	**Precio**	**Importe**
MO.1	h	Graduado en Ingeniería en Tecnologías industriales	30,00	35,00	1050,00
MAT.H1	h	Ordenador de torre	30,00	0,18	5,27
	%	Costes directos complementarios	2,00		21,11
		Coste total UO1			**1076,38**

Figura 50: Precios descompuestos capítulo 1

Capítulo 2. Programación en MatLab y True Time Lund					
UO2.1: Programación del modelo no lineal del AGV.					
MO.1	h	Graduado en Ingeniería en Tecnologías industriales	30,00	35,00	1050,00
MAT.S1	h	MatLab	30,00	0,42	12,65
MAT.H1	h	Ordenador de torre	30,00	0,18	5,27
	%	Costes directos complementarios	2,00		21,36
		Coste total UO2.1			**1.089,28 €**
UO2.2: Programación mediante True Time Lund del controlador, actuadores y sensores.					
MO.1	h	Graduado en Ingeniería en Tecnologías industriales	140,00	35,00	4900,00
MAT.S1	h	MatLab	140,00	0,42	59,04
MAT.S2	h	True Time Lund	140,00	0,00	0,00
MAT.H1	h	Ordenador de torre	140,00	0,18	24,60
	%	Costes directos complementarios	2,00		99,67
		Coste total UO2.2			**5.083,31 €**
		Coste total UO2			**6.172,59 €**

Figura 51: Precios descompuestos capítulo 2

Capítulo 3. Simulaciones					
MO.1	h	Graduado en Ingeniería en Tecnologías industriales	60,00	35,00	2100,00
MAT.S1	h	MatLab	60,00	0,00	0,00
MAT.S2	h	True Time Lund	60,00	0,00	0,00
MAT.H1	h	Ordenador de torre	60,00	0,18	10,54
	%	Costes directos complementarios	2,00		42,21
		Coste total UO3			**2.152,75 €**

Figura 52: Precios descompuestos capítulo 3

Capítulo 4. Redacción de la memoria del proyecto					
MO.1	h	Graduado en Ingeniería en Tecnologías industriales	40,00	35,00	1400,00
MAT.S3	h	Overleaf (Editor de LaTeX)	40,00	0,00	0,00
MAT.H1	h	Ordenador de torre	40,00	0,42	16,87
	%	Costes directos complementarios	2,00		28,34
		Coste total UO4			**1.445,20 €**

Figura 53: Precios descompuestos capítulo 4

9.5. Presupuesto final

Presupuesto final	
	Importe
Capítulo 1	1.076,38 €
Capítulo 2	6.172,59 €
Capítulo 3	2.152,75 €
Capítulo 4	1.445,20 €
Presupuesto total de ejecución material	**10.846,92 €**
Gastos generales (12%)	1.301,63 €
Beneficio industrial (6%)	650,82 €
Presupuesto total de inversión	**12.799,37 €**
IVA (21%)	2.687,87 €
Presupuesto base de licitación	**15.487,23 €**

Figura 54: Presupuesto final

Finalmente, el proyecto alcanza un coste de **quince mil cuatrocientos ochenta y siete euros con veintitrés céntimos**.

10. Anexos

Actuadores

```
function act_init(INICIOKERNELS)

% Initialize TrueTime kernel
ttInitKernel('prioFP'); % nbrOfInputs, nbrOfOutputs, fixed priority

% Create task data (local memory)

data.period = INICIOKERNELS.T;
data.Nbif=INICIOKERNELS.Nbif;
data.iteract = 0;
data.itercont=0;
data.total_iter = INICIOKERNELS.ITERACIONES;
data.flag=1;
data.nodata = 0;
data.count=1;
data.horizonte = INICIOKERNELS.horizonte;

%Variables TFM
data.ax=[INICIOKERNELS.ax INICIOKERNELS.ax INICIOKERNELS.ax INICIOKERNELS.ax
    INICIOKERNELS.ax];
data.delta=[INICIOKERNELS.delta INICIOKERNELS.delta INICIOKERNELS.delta ...
INICIOKERNELS.delta INICIOKERNELS.delta];


offset = 0;
ttCreatePeriodicTask('act_task', offset, data.period, 'act_code',data);

function [exectime, data] = act_code(seg, data)

switch seg

        case 1
            if data.iteraux == 10
                    data.iteraux = 0;
            end

        data.iteract = data.iteract + 1;
        data.iteraux = data.iteraux + 1;
        exectime = 0;
        case 2
            msgrcv = ttGetMsg;
```

```
            if isempty(msgrcv)
                disp(["No msg en actuador" string(ttCurrentTime)]);
                if (data.flag==1 && data.iteract==1)
                    data.flag=0;
                else
                    data.nodata = 1;
                    data.count = data.count + 1;
            end
        else
            disp(["Si hay msg en actuador" string(ttCurrentTime)])
            data.count = 1;
            data.ax = msgrcv.ax;
            data.delta = msgrcv.delta;
            data.itersens=msgrcv.itersens;
            data.itercont=msgrcv.itercont;
        end
        exectime = 0;

    case 6
        if(data.nodata == 0)
            ttAnalogOut(1, data.ax(data.iteraux));
            ttAnalogOut(2, data.delta(data.iteraux));
            data.count = data.iteraux;
            disp(['Aplico DREKF' string(ttCurrentTime)]);
        elseif(data.count<=data.horizonte)
            data.nodata=0;
            disp(['Aplico predicci n' string(data.count) string(ttCurrentTime)
    ])
            ttAnalogOut(1, data.ax(data.count));
            ttAnalogOut(2, data.delta(data.count));
        else
            disp(['Aplico predicci n' string(data.horizonte) string(
    ttCurrentTime)])
            data.nodata=0;
            ttAnalogOut(1, data.ax(end));
            ttAnalogOut(2, data.delta(end));
        end
        exectime = 0;

    case 10
        exectime = -1;

    otherwise
        exectime = 0;
end
```

Sensores

```
function sens_init(INICIOKERNELS)

ttInitKernel('prioFP');

period = INICIOKERNELS.T;                          % sampling period
data.itersens = 0;                                  % iteration in sensor device
data.total_iter = INICIOKERNELS.ITERACIONES;        % number of iterations
offset = 0;

data.lossc = INICIOKERNELS.lossc;

ttCreatePeriodicTask('sens_task', offset, period, 'sens_code',data);

function [exectime, data] = sens_code(seg, data)
switch seg
        case 1
            data.itersens = data.itersens+1;
            exectime = 0;
        case 2
            data.vx_limpia = ttAnalogIn(1);
            data.vy_limpia = ttAnalogIn(2);
            data.x_limpia = ttAnalogIn(3);
            data.y_limpia = ttAnalogIn(4);
            data.psi_limpia = ttAnalogIn(5);
            data.psid_limpia = ttAnalogIn(6);
            exectime = 0;
        case 6
            ttAnalogOut(1, data.vx_limpia)
            ttAnalogOut(2, data.vy_limpia)
            ttAnalogOut(3, data.x_limpia)
            ttAnalogOut(4, data.y_limpia)
            ttAnalogOut(5, data.psi_limpia)
            ttAnalogOut(6, data.psid_limpia)
            exectime = 0;
        case 7
            if(data.lossc(data.itersens) == 1)
                disp(['mensaje_enviado_desde_sensor' string(ttCurrentTime)])
                msgenv.itersens = data.itersens;
                msgenv.vx = data.vx_limpia;
                msgenv.vy = data.vy_limpia;
                msgenv.x = data.x_limpia;
                msgenv.y = data.y_limpia;
                msgenv.psi = data.psi_limpia;
                msgenv.psid = data.psid_limpia;
```

```
                ttSendMsg(2, msgenv, 8);
            else
                disp(['mensaje no enviado desde sensor' string(ttCurrentTime)])
        end
        exectime = 0;
    case 10
        exectime = -1;
    otherwise
        exectime = 0;
end
```

Controlador

```
function slow_ctrl_init(INICIOKERNELS)


ttInitKernel('prioFP');

data.period = INICIOKERNELS.T;
data.Nbif = INICIOKERNELS.Nbif;
data.nodata = 0;
data.itercont = 0;
data.total_iter = INICIOKERNELS.ITERACIONES;
data.flag=1;
data.horizonte = INICIOKERNELS.horizonte;

data.retardoCA = INICIOKERNELS.retardoCA;

data.losca = INICIOKERNELS.losca;

data.z_EKF = [INICIOKERNELS.vx; 0; INICIOKERNELS.x; INICIOKERNELS.y;
    INICIOKERNELS.psi; INICIOKERNELS.psid];

s=tf('s');

%Acciones de control iniciales
data.ax = INICIOKERNELS.ax;
data.delta = INICIOKERNELS.delta;
                    % measurement noises
data.w_ekf=INICIOKERNELS.w_ekf;
data.v_ekf=INICIOKERNELS.v_ekf;
                        % EKF matrices
data.P=INICIOKERNELS.P;
data.Q=INICIOKERNELS.Q;
data.R=INICIOKERNELS.R2;
data.vhMdl=INICIOKERNELS.vhMdl;
data.trMdl=INICIOKERNELS.trMdl;
data.dt=INICIOKERNELS.dt;
data.T=INICIOKERNELS.T;
data.M=INICIOKERNELS.M;
data.psidrefn=INICIOKERNELS.psidrefn;
data.L=INICIOKERNELS.L;
data.Kp=INICIOKERNELS.Kp;
data.gamma=INICIOKERNELS.gamma;
data.ipp=INICIOKERNELS.ipp;
data.xref_tot=INICIOKERNELS.xref_tot;
data.yref_tot=INICIOKERNELS.yref_tot;
```

```
data.xrefnac=[];
data.yrefnac=[];
data.ddac=[];
data.psidrefnac=[];
data.xrefn = INICIOKERNELS.xref_tot(1);
data.yrefn = INICIOKERNELS.yref_tot(1);


offset = (data.period)/1.01;
ttCreatePeriodicTask('slow_ctrl', offset, data.period, 'slow_ctrl_code',data);

function [exectime, data] = slow_ctrl_code(seg, data)

switch seg

    case 1
        if data.flag==1
            data.flag=0;
            ttSetNextSegment ( 10 )
        else
            data.itercont = data.itercont + 1;
        end
        exectime=0;

        case 2
            msgrcv = ttGetMsg;
        if isempty(msgrcv)
            disp (['no_hay_msg_en_Controlador' string(ttCurrentTime)])
            data.nodata = 1;
            data.y_ekf(:,data.itercont) = [data.z_EKF(1,data.itercont,1);...
            data.z_EKF(3,data.itercont,1); data.z_EKF(4,data.itercont,1);...
            data.z_EKF(5,data.itercont,1)];
        else
            data.nodata = 0;
            data.itersens = msgrcv.itersens;
            data.vx = msgrcv.vx;
            data.vy = msgrcv.vy;
            data.x = msgrcv.x;
            data.y = msgrcv.y;
            data.psi = msgrcv.psi;
            data.psid = msgrcv.psid;
            data.y_ekf(:,data.itercont) = [data.vx; data.x; data.y; data.psi];
            disp (['si_hay_msg_en_Controlador' string(ttCurrentTime)])
        end
        exectime = 0;
```

```
    case 3

    data.u_ekf(:,data.itercont) = [data.ax(data.itercont,1), data.delta(data
.itercont,1)];
    % FILTRO DE KALMAN
    [data.z_EKF(:,data.itercont,2), data.P] = EKF(data.z_EKF(:,data.itercont
,1),...
    data.w_ekf, data.v_ekf, data.P, data.y_ekf(:,data.itercont),...
    data.Q, data.R, data.u_ekf(:,data.itercont),data.vhMdl,...
    data.trMdl, data.dt, data.itercont, data.M,data.nodata);

    data.vxe=data.z_EKF(1,data.itercont,2);     % para calcular delta(kk+1)
    data.vxef=data.vxe;          % para calcular psidrefn en path planning
    %vye(kk+1)=z_EKF(2);
    data.xe=data.z_EKF(3,data.itercont,2);
    data.xecf=data.xe;            % para calcular dist_x en path planning
    data.ye=data.z_EKF(4,data.itercont,2);
    data.yecf=data.ye;           % para calcular dist_y en path planning
    data.psie=data.z_EKF(5,data.itercont,2);
    data.psief=data.psie;         % para calcular alfa y psirefn en plath
planning
    data.pside=data.z_EKF(6,data.itercont,2);    % para calcular delta(kk+1)


    %C lculo acciones de control
    data.ax(data.itercont+1,1)=data.ax(data.itercont,1);
    data.delta(data.itercont+1,1) = atan2(data.psidrefn*data.L,data.vxe)+...
    (data.psidrefn-data.pside)*data.Kp*data.gamma;

    %hStepEstimation

    %Utilizar esta linea de c digo en el caso de desactivar hstep
    %data.z_EKF(:,data.itercont+1,1) = data.z_EKF(:,data.itercont,2);

    [data.z_EKF,data.ax,data.delta] = hStepEstimation(data.z_EKF,data.ax,
data.delta,...
    data.vhMdl, data.trMdl,data.horizonte,data.itercont,...
    data.ipp,data.xref_tot,data.yref_tot,data.dt,...
    data.w_ekf,data.xrefn,data.yrefn);

    %% Path Planning.
    data.flag_pp = true;

    data.look_ahead = 5;
    while(data.flag_pp && data.ipp<1200)
```

```
        data.ipp = data.ipp+1;
        data.dist_x = abs(data.xecf - data.xref_tot(data.ipp));
        data.dist_y = abs(data.yecf - data.yref_tot(data.ipp));
        data.dist_total = sqrt(data.dist_x*data.dist_x + data.dist_y*data.
dist_y);
        if(data.dist_total > data.look_ahead)
            data.xrefn = data.xref_tot(data.ipp);
            data.yrefn = data.yref_tot(data.ipp);
            data.flag_pp = false;
            data.ipp = data.ipp-1;
        end
    end



    ttAnalogOut(1,data.xrefn);
    ttAnalogOut(2,data.yrefn);
    data.xrefnac=[data.xrefnac; data.xrefn];
    data.yrefnac=[data.yrefnac; data.yrefn];

    data.dd=sqrt((data.yrefn-data.yecf)^2+(data.xrefn-data.xecf)^2);
    data.ddac=[data.ddac data.dd];
    data.alfa=atan2((data.yrefn-data.yecf),(data.xrefn-data.xecf))-data.
psief;

    data.psidrefn=(2*data.vxef*sin(data.alfa))/data.dd;

    data.psidrefnac=[data.psidrefnac; data.psidrefn];
    data.psirefn=data.psief+data.T*data.psidrefn;

    exectime = data.retardoCA(data.itercont);
    %exectime=0;

    case 7
    if(data.losca(data.itercont)==1)
        disp(['mensaje enviado desde el controlador' string(ttCurrentTime)])
        msgenv.ax = data.ax(data.itercont+1,:);
                msgenv.delta = data.delta(data.itercont+1,:);
                msgenv.itersens=data.itersens;
                msgenv.itercont=data.itercont;
                ttSendMsg(3, msgenv, 8);
    else
        disp('mensaje no enviado desde el controlador')
    end
            exectime = 0;
```

```
        case 10
        exectime = -1;

        otherwise
                exectime = 0;
end
end

%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [mx_km1, P_km1] = EKF(mx_k, mw_k, mv_k, P_k, y_kp1, Q, R, u_ekf, vhMdl
    ,...
trMdl, dt, kk, M, nodata)

  % EKF calculations
  xDim = size(mx_k,1);
  mx_kp1= f_BicycleModel(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk); % Prediction
  mx_kp1;
  A = numerical_jac_x(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk);
  L = numerical_jac_w(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk);
  P_kp1 = A*P_k*A' + L*Q*L';
  if (mod(kk+1,M)==0 && nodata ==0)
    disp(['Correcci n_DREKF!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!' string(ttCurrentTime)
    ])
    my_kp1 = h_BicycleModel(mx_kp1, mv_k, kk);
    H = numerical_jac_x_2(mx_kp1, mv_k, kk);
    M = numerical_jac_w_2(mx_kp1, mv_k, kk);
    P12 = P_kp1*H';
    K = P12*inv((H*P12)+(M*R*M'));
    mx_km1 = mx_kp1 + K*(y_kp1 - my_kp1);  % Correction
    yy=(y_kp1 - my_kp1);
    P_km1 = K*R*K' + (eye(xDim)-K*H)*P_kp1*(eye(xDim)-K*H)';
    mx_km1;
  else
    disp('<>0')
    mx_km1 = mx_kp1;  % Shift
```

```
    P_km1 = P_kp1;
  end
end

function [z_EKF,u1,u2] = hStepEstimation(z_EKF,u1,u2,vhMdl,trMdl,...
horizonte,itercont,ipp,xref_tot,yref_tot,dt,w,xrefn,yrefn)

    %Estimaci n h pasos en el futuro
    z_EKF(:,itercont+1,1) = z_EKF(:,itercont,2);
    kk=itercont;
    Kp = 1;
    gamma=0.55;
    %Modelo del proceso
    for i=2:horizonte

        % get state and control actions
        Vx=z_EKF(1,itercont+1,i-1);
        Vy=z_EKF(2,itercont+1,i-1);
        X=z_EKF(3,itercont+1,i-1);
        Y=z_EKF(4,itercont+1,i-1);
        psi=z_EKF(5,itercont+1,i-1);
        wz=z_EKF(6,itercont+1,i-1);

        ax=u1(itercont+1,i-1);
        delta=u2(itercont+1,i-1);

        % extract parameters
        a=vhMdl(1);
        b=vhMdl(2);
        m=vhMdl(3);
        I=vhMdl(4);
        L = a +b;

        Caf=trMdl(1);
        Car=trMdl(2);

        % compute lateral tire forces
        Fyf = -Caf * atan(( (Vy + wz*a)/max(Vx, 5*0.44704) ) - delta);
        Fyr = -Car * atan( (Vy - wz*b)/max(Vx, 5*0.44704) );

        % compute next state Hacer prueba con ruido y sin ruido
        Vx_next = Vx + dt*(ax + w(1,kk));
        Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m -
    wz*Vx +w(2,kk));
        X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi) + w(3,kk));
```

```
        Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi) + w(4,kk));
        psi_next = psi + dt*(wz + w(5,kk));
        wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta)) -
    b*Fyr/I + w(6,kk));

%Sin ruido
%         Vx_next = Vx + dt*(ax);
%         Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m
    - wz*Vx);
%         X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi));
%         Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi));
%         psi_next = psi + dt*(wz);
%         wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta))
    - b*Fyr/I);

        z_EKF(:,itercont+1,i) = [Vx_next; Vy_next; X_next; Y_next; psi_next;
    wz_next];

        vxe=z_EKF(1,itercont+1,i);
        vxef=vxe;
        xe=z_EKF(3,itercont+1,i);
        xecf=xe;
        ye=z_EKF(4,itercont+1,i);
        yecf=ye;
        psie=z_EKF(5,itercont+1,i);
        psief=psie;
        pside=z_EKF(6,itercont+1,i);

        %Path planing
        flag_pp = true;

        look_ahead = 5;
        while(flag_pp && ipp<1200)
            ipp = ipp+1;
            dist_x = abs(xecf - xref_tot(ipp));
            dist_y = abs(yecf - yref_tot(ipp));
            dist_total = sqrt(dist_x*dist_x + dist_y*dist_y);

            if(dist_total > look_ahead)
                xrefn = xref_tot(ipp);
                yrefn = yref_tot(ipp);
                flag_pp = false;
                ipp = ipp-1;
            end
        end
```

```
        if ipp == 1200
            ipp = 1;
        end

        dd=sqrt((yrefn-yecf)^2+(xrefn-xecf)^2);
        alfa=atan2((yrefn-yecf),(xrefn-xecf))-psief;
        psidrefn=(2*vxef*sin(alfa))/dd;
        psirefn=psief+dt*psidrefn;

        %Calcular acci n de control futura
        u1(itercont+1,i)=u1(itercont+1,i-1); %ax
        u2(itercont+1,i) = atan2(psidrefn*L,vxe)+ (psidrefn-pside)*Kp*gamma; %
    delta
    end


end

function [z_next] = f_BicycleModel(z, w, u, vhMdl, trMdl, dt, kk)

  Vx=z(1,1);
  Vy=z(2,1);
  X=z(3,1);
  Y=z(4,1);
  psi=z(5,1);
  wz=z(6,1);

  ax=u(1);
  delta=u(2);

  a=vhMdl(1);
  b=vhMdl(2);
  m=vhMdl(3);
  I=vhMdl(4);

  Caf=trMdl(1);
  Car=trMdl(2);

  Fyf = -Caf * atan(( (Vy + wz*a)/max(Vx, 5*0.44704) ) - delta);
  Fyr = -Car * atan( (Vy - wz*b)/max(Vx, 5*0.44704) );

  Vx_next = Vx + dt*(ax + w(1,kk));
  Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m - wz*Vx
    +w(2,kk));
```

```
  X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi) + w(3,kk));
  Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi) + w(4,kk));
  psi_next = psi + dt*(wz + w(5,kk));
  wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta)) - b*Fyr/
    I + w(6,kk));


  z_next = [Vx_next; Vy_next; X_next; Y_next; psi_next; wz_next];
end


function [y_next] = h_BicycleModel(z, v, kk)

  % compute next output
  y_next = [z(1,1)+v(1,kk); z(3,1)+v(2,kk); z(4,1)+v(3,kk); z(5,1)+v(4,kk)];
end


% Functions to compute Jacobian matrices
function [jac]=numerical_jac_x(x, w, u_ekf, vhMdl, trMdl, dt, kk)

  y = f_BicycleModel(x, w, u_ekf, vhMdl, trMdl, dt, kk);
  jac=zeros(size(y,1),size(x,1));
  eps=1e-5;
  xp=x;
  for i=1:size(x,1)
    xp(i,1) = x(i,1) + eps/2.0;
    yhi = f_BicycleModel(xp, w, u_ekf, vhMdl, trMdl, dt, kk);
    xp(i,1) = x(i,1) - eps/2.0;
    ylo = f_BicycleModel(xp, w, u_ekf, vhMdl, trMdl, dt, kk);
    xp(i,1) = x(i,1);
    jac(:,i) = (yhi-ylo)/eps;
  end
end

 function [jac]=numerical_jac_w(x, w, u_ekf, vhMdl, trMdl, dt, kk)

  y = f_BicycleModel(x, w, u_ekf, vhMdl, trMdl, dt, kk);
  jac=zeros(size(y,1),size(w,1));
  eps=1e-5;
  wp=w;
  for i=1:size(w,1)
    wp(i,kk) = w(i,kk) + eps/2.0;
    yhi = f_BicycleModel(x, wp, u_ekf, vhMdl, trMdl, dt, kk);
    wp(i,kk) = w(i,kk) - eps/2.0;
```

```
    ylo = f_BicycleModel(x, wp, u_ekf, vhMdl, trMdl, dt, kk);
    wp(i,1) = w(i,kk);
    jac(:,i) = (yhi-ylo)/eps;
  end
 end

function [jac]=numerical_jac_x_2(x, v, kk)

  y = h_BicycleModel(x, v, kk);
  jac=zeros(size(y,1),size(x,1));
  eps=1e-5;
  xp=x;
  for i=1:size(x,1)
    xp(i,1) = x(i,1) + eps/2.0;
    yhi = h_BicycleModel(xp, v, kk);
    xp(i,1) = x(i,1) - eps/2.0;
    ylo = h_BicycleModel(xp, v, kk);
    xp(i,1) = x(i,1);
    jac(:,i) = (yhi-ylo)/eps;
  end
end

function [jac]=numerical_jac_w_2(x, v, kk)

  y = h_BicycleModel(x, v, kk);
  jac=zeros(size(y,1),size(v,1));
  eps=1e-5;
  vp=v;
  for i=1:size(v,1)
    %wp(i,1) = v(i,1) + eps/2.0;
    vp(i,kk) = v(i,kk) + eps/2.0;
    yhi = h_BicycleModel(x, vp, kk);
    %wp(i,1) = v(i,1) - eps/2.0;
    vp(i,kk) = v(i,kk) - eps/2.0;
    ylo = h_BicycleModel(x, vp, kk);
    %wp(i,1) = v(i,1);
    vp(i,kk) = v(i,kk);
    jac(:,i) = (yhi-ylo)/eps;
  end
end
```

Programa principal

```
%%% EXECUTE THIS CODE THE FIRST TIME TO GENERATE THE GLOBAL
VARIABLE "INICIOKERNELS" (FOR INITIALIZATIONS) %%%
clc;
clear all;

tic
global INICIOKERNELS;


for simu=20:10:20
%Variables TFM
titulo = "h = " + int2str(simu);
%titulo = "P rdidas 25% y retardos"
INICIOKERNELS.horizonte=simu;
INICIOKERNELS.flag_ini=1;
INICIOKERNELS.M=10; %Multiplicidad en el filtro
INICIOKERNELS.Kp=1.0;
INICIOKERNELS.gamma=0.55;
INICIOKERNELS.x(1)=0.0;
INICIOKERNELS.y(1)=79;
INICIOKERNELS.vx(1)=5;
INICIOKERNELS.vy(1)=0;
INICIOKERNELS.psi(1)=273*pi/180;
INICIOKERNELS.psid(1)=0;
INICIOKERNELS.ax(1)=0.05;
INICIOKERNELS.delta=0;
INICIOKERNELS.ay(1)=-0.001;
INICIOKERNELS.vxe(1)=INICIOKERNELS.vx(1);
INICIOKERNELS.pside(1)=INICIOKERNELS.psid(1);
INICIOKERNELS.psidd(1)=0;
INICIOKERNELS.vxf=INICIOKERNELS.vx(1);
INICIOKERNELS.vyf=INICIOKERNELS.vy(1);
INICIOKERNELS.ayf=INICIOKERNELS.ay(1);
INICIOKERNELS.psif=INICIOKERNELS.psi(1);
INICIOKERNELS.xcf = INICIOKERNELS.x(1);
INICIOKERNELS.ycf = INICIOKERNELS.y(1);
INICIOKERNELS.psidf=0.1;
INICIOKERNELS.axf=INICIOKERNELS.ax(1);
%Par metros rajmani
INICIOKERNELS.lf=1.2;
INICIOKERNELS.lr=1.65;
INICIOKERNELS.m=1800;
INICIOKERNELS.Caf=140000;
INICIOKERNELS.Car=120000;
```

```
INICIOKERNELS.Iz=3270;
INICIOKERNELS.a=1.20;
INICIOKERNELS.b=1.65;
INICIOKERNELS.L=INICIOKERNELS.a+INICIOKERNELS.b;
INICIOKERNELS.ftire_stiffness=140000.0;
INICIOKERNELS.rtire_stiffness=120000.0;
INICIOKERNELS.mass=1800.0;
INICIOKERNELS.iz=3270.0;
%Para el EKF
INICIOKERNELS.vhMdl=[INICIOKERNELS.a,INICIOKERNELS.b,INICIOKERNELS.mass,
    INICIOKERNELS.iz];
INICIOKERNELS.trMdl=[INICIOKERNELS.Caf,INICIOKERNELS.Car];
load meas_noise
load proc_noise
INICIOKERNELS.v_ekf=v_ekf;
INICIOKERNELS.w_ekf=w_ekf;
var_gps=1.0e-6;
var_v=1.0e-4;
var_psi=1.0e-6;
var_ax=1.0e-4;
var_delta=1.0e-4;
var_noise=1.0e-4;
INICIOKERNELS.var1 = [var_ax, var_delta, var_noise, var_noise, var_noise,
    var_noise];
INICIOKERNELS.var2 = [var_v, var_gps, var_gps, var_psi];
INICIOKERNELS.P=eye(6);
INICIOKERNELS.Q=diag(INICIOKERNELS.var1);
INICIOKERNELS.R2=diag(INICIOKERNELS.var2);

INICIOKERNELS.ITERACIONES = 5500;     % number of iterations
INICIOKERNELS.T = 0.01;   % actuation period
INICIOKERNELS.NT = INICIOKERNELS.T*INICIOKERNELS.M;% sampling period
INICIOKERNELS.Nbif = INICIOKERNELS.M;     % multiplicity
% Sampling time
INICIOKERNELS.dt=INICIOKERNELS.T;

load ref_soriano_coche %Carga xref_tot yref_tot cada 0.1 metros en cuadrado 20 x
    20 metros
factor_escala = 4; %escalamos las referencias
xref_tot = xref_tot*factor_escala;
yref_tot = yref_tot*factor_escala;
psiref_tot = [zeros(400,1);(pi/2)*ones(400,1);pi*ones(400,1);(3*pi/2)*ones
    (400,1)];
%
INICIOKERNELS.xref=xref_tot;
```

```
INICIOKERNELS.yref=yref_tot;
INICIOKERNELS.xref_tot=[INICIOKERNELS.xref(600:800);INICIOKERNELS.xref(1:800)
    ;...
INICIOKERNELS.xref(1:200)];
INICIOKERNELS.yref_tot=[INICIOKERNELS.yref(600:800);INICIOKERNELS.yref(1:800)
    ;...
INICIOKERNELS.yref(1:200)];

for hh=1:length(INICIOKERNELS.xref_tot)-1
    INICIOKERNELS.dref(hh)=sqrt((INICIOKERNELS.xref_tot(hh+1)-...
    INICIOKERNELS.xref_tot(hh))^2+(INICIOKERNELS.yref_tot(hh+1)-...
    INICIOKERNELS.yref_tot(hh))^2);
end



INICIOKERNELS.psirefn=270*pi/180;
INICIOKERNELS.psidrefn=0.1;
INICIOKERNELS.km = 1;
INICIOKERNELS.ipp=1;

% delays
load retardos_009.mat

INICIOKERNELS.retardoCA=retardo;

% dropouts
load perdidas_15_doblecero.mat

INICIOKERNELS.lossc=lossc;

INICIOKERNELS.losca=losca;

load w_ekf_sim.mat

sim ('Control_en_red.slx');

ind = 1;

x = ans.x';
x=x(3:length(x)-1);
y = ans.y';
y=y(3:length(y)-1);
xrefnac = ans.xrefn';
xrefnac = xrefnac(3:length(xrefnac)-1);
yrefnac = ans.yrefn';
```

```
yrefnac = yrefnac(3:length(yrefnac)-1);

distp2p=(sqrt((xrefnac-x).^2+(yrefnac-y).^2));

distbuena=zeros(1,length(distp2p));
for i=1:length(x)
    distbuena(i)=100000;
    for j=1:length(xrefnac)
        distbuena(i)=min(distbuena(i),(sqrt((xrefnac(j)-x(i))^2+(yrefnac(j)-y(i)
    )^2)));
    end
end

figure
plot(distp2p)
hold on
plot(distbuena)
xlabel("Iteraciones");
ylabel("Dist(m)");
legend("Dist. ref din mica","Dif. con la trayectoria");
title(titulo)

% ndice J1
J1mod=0;
for i=ind:length(xrefnac)
    J1mod=J1mod+distbuena(i);
end

% ndice J2
J2mod=distbuena(ind);
for i=ind:length(xrefnac)
    J2mod=max(J2mod,distbuena(i));
end

J1mod
J2mod
desv = std2(distp2p)

% Ploteos;

figure
plot(xref_tot,yref_tot,'b');
xlabel("X(m)");
ylabel("Y(m)");
hold on
```

```
plot(x,y,'-r','MarkerSize',0.5);
legend("Referencia","Trayectoria");
title(titulo);

filename = strcat("Simulacion_15_",int2str(simu));
save(filename,"J1mod","J2mod","desv")
end
toc
```

Esquema Simulink

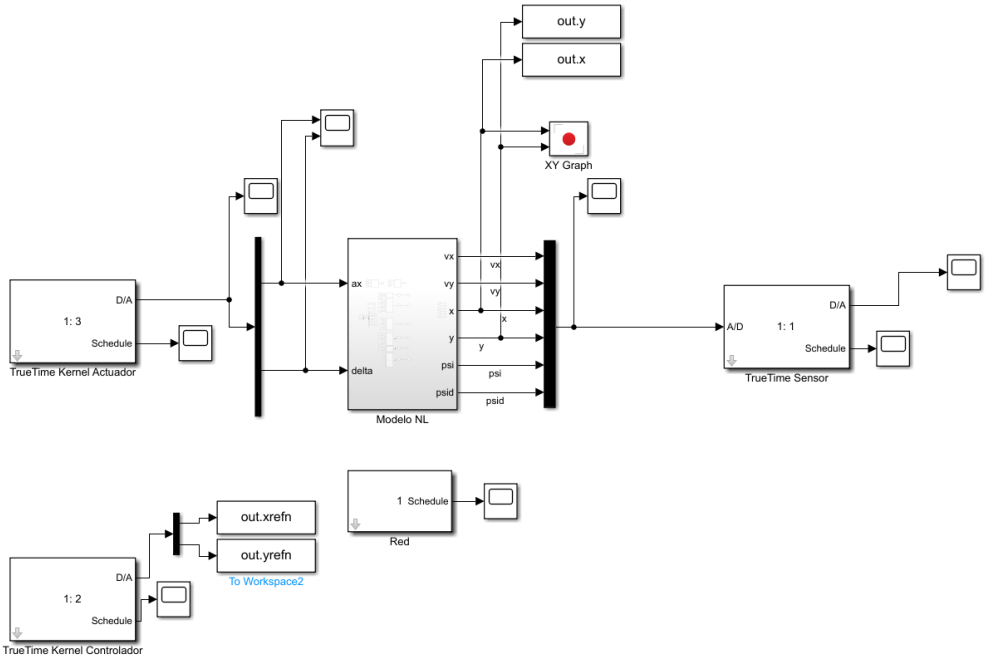

Figura 55: Esquema Simulink

Referencias

[1] Á. Cuenca, W. Zhan, J. Salt, J. Alcaina, C. Tang, M. Tomizuka, A remote control strategy for an autonomous vehicle with slow sensor using Kalman filtering and dual-rate control, Sensors 19 (13) (2019). doi:10.3390/s19132983.
URL https://www.mdpi.com/1424-8220/19/13/2983

[2] J. Salt Ducajú, J. Salt Llobregat, Á. Cuenca, M. Tomizuka, Autonomous ground vehicle lane-keeping LPV model-based control: Dual-rate state estimation and comparison of different real-time control strategies, Sensors 21 (4) (2021). doi:10.3390/s21041531.
URL https://www.mdpi.com/1424-8220/21/4/1531

[3] R. Carbonell, Á. Cuenca, V. Casanova, R. Pizá, J. Salt, Dual-rate extended Kalman filter based path-following motion control for an unmanned ground vehicle. Realistic simulation, Sensors 21 (22) (2021).
URL https://www.mdpi.com/1424-8220/21/22/7557

[4] Ford Motor Company, Ficha técnica del vehículo 2017 lincoln mkz (January 2017).
URL https://es.lincoln.com/img/lincoln/ES_140329.pdf

[5] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, C. Peng, Networked control systems: A survey of trends and techniques, IEEE/CAA Journal of Automatica Sinica 7 (1) (2020) 1–17. doi:10.1109/JAS.2019.1911651.

[6] A. Cervin, D. Henriksson, M. Ohlin, Truetime 2.0 reference manual (April 2016).
URL http://archive.control.lth.se/media/Research/Tools/TrueTime/report_2016-02-10.pdf

[7] ai2, Áreas de investigación ai2 (January 2015).
URL https://www.ai2.upv.es/area-de-control-de-procesos/

[8] A. González, Á. Cuenca, J. Salt, J. Jacobs, Robust stability analysis of an energy-efficient control in a networked control system with application to unmanned ground vehicles, Information Sciences 578 (2021) 64–84. doi:https://doi.org/10.1016/j.ins.2021.07.016.
URL https://www.sciencedirect.com/science/article/pii/S0020025521007064

[9] J. Alcaina, A. Cuenca, J. Salt, V.Casanova, R. Pizá, Delay-independent dual-rate pid controller for a packet-based networked control system, Information Sciences 484 (2019) 27–43. doi:https://doi.org/10.1016/j.ins.2019.01.059.
URL https://www.sciencedirect.com/science/article/pii/S0020025519300726

[10] R. Rajamani, Vehicle Dynamics and Control, Springer (2012). doi:10.1007/978-1-4614-1433-9.

[11] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Oren-stein, J. Paefgen, I. Penn, S. Thrun Junior, The Stanford entry in the Urban Challenge, Journal of Field Robotics 25 (9) (2008) 569–597. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20258, doi:https://doi.org/10.1002/rob.20258.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20258

[12] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, How, Real-time motion planning with applications to autonomous urban driving, IEEE Transactions on Control Systems Technology 17 (5) (2009) 1105–1118. doi:10.1109/TCST.2008.2012116.

[13] A. Sala, Predicción en sistemas dinámicos lineales: observador óptimo (filtro de Kalman) (2018).

URL http://personales.upv.es/asala/DocenciaOnline/material/FiltroKalman.pdf