UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Control de seguimiento de trayectorias para vehículos autoguiados en entornos con información limitada

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Lloris Rodríguez, Cayetana

Tutor/a: Cuenca Lacruz, Ángel Miguel

Director/a Experimental: Carbonell Lázaro, Rafael

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

CONTROL DE SEGUIMIENTO DE TRAYECTORIAS PARA VEHÍCULOS AUTOGUIADOS EN ENTORNOS CON INFORMACIÓN LIMITADA.

Autora: Cayetana Lloris Rodríguez

Tutor: Ángel Miguel Cuenca Lacruz

Cotutor: Rafael Carbonell Lázaro

Trabajo Fin de Máster presentado en el Departamento de Ingeniería de Sistemas y Automática de la Universitat Politècnica de València para la obtención del Título de Máster Universitario en Automática e Informática Industrial.

Curso 2023-24

Valencia, septiembre de 2024

Contenido

Capítulo 1. Introducción

Los más recientes avances en tecnología han creado el entorno ideal para la investigación de los coches autónomos. Hardware y software de última generación armonizan para que un vehículo pueda generar rutas, ser controlado y tomar decisiones en tiempo real.

Solo los vehículos que dominan todas estas habilidades tienen el nivel de autonomía 5, el más alto. Son aquellos que no requieren ninguna intervención humana. Hasta la fecha, no existen vehículos particulares que sean completamente autónomos. Sin embargo, waymo tiene un servicio de viajes autónomos (taxis), que funciona en San Francisco y Phoenix por ahora [1]. En este trabajo se presentan propuestas relacionadas hasta el nivel 4, automatización elevada [2].

El siguiente paso sería adoptar algoritmos de aprendizaje profundo y toma de decisiones. Eso supondría adoptar una ruta distinta para el trabajo.

El objetivo principal de este TFM (Trabajo Fin de Máster) es analizar y contrastar diversas alternativas de control para el seguimiento de trayectorias en vehículos autoguiados que operan en ambientes con información limitada.

Esta escasez de datos puede deberse a diferentes factores como el uso de redes inalámbricas, donde involuntariamente pueden perderse paquetes, o donde voluntariamente puede utilizarse la comunicación disparada por eventos (periodic event-triggered communication, PETC), con el fin de ahorrar recursos y otros beneficios mencionados en [3].

La transmisión de datos también puede estar limitada por el uso de sensores que operen a una frecuencia más baja de la que se requiere en el control [4-7].

1.1 Estado del arte

Consiste en aplicar técnicas de control avanzadas relativamente novedosas, y aplicar un nuevo enfoque para aprovechar al máximo las acciones de control calculadas por el controlador. Entonces, se evaluará la robustez ante situaciones de escasez de información.

Por un lado, se diseñarán controladores avanzados como el controlador predictivo basado en modelo (Model Predictive Control, MPC) para casos de distinta complejidad y se estudiarán las diferencias respecto a un tipo de control más básico, el basado en el modelo cinemático inverso de bicicleta (Inverse Kinematic Bicycle Model, IKIBI) de Rajamani [5,6,8].

El MPC, usará un modelo lineal de parámetros variables (Linear parameter-varying, LPV) [6,8,9], basado en el modelo cinemático de bicicleta de Rajamani. Las ecuaciones simplificadas que describen la dinámica del vehículo en base a un parámetro variable se basan en estructuras polinomiales con coeficientes que deben ser calculados. Para este proceso de identificación del sistema existen métodos basados en formular problemas de optimización a partir de las funciones de transferencia del sistema y resolverlos [10].

La universidad, por trabajos previos, contaba con los coeficientes para el vehículo que se va a simular en el TFM, el modelo 2017 Lincoln MKZ [8]. La ruta de referencia proviene de una prueba realizada en Richmond Field Station, de la Universidad de California en Berkeley [5,6,8,9].

Se añadirá ruido que se compensará con un filtro de Kalman (Kalman Filter, KF) . De esta forma la simulación será más realista. Además, el tiempo de muestreo del modelo debe ser lo suficiente pequeño para que represente correctamente el comportamiento del vehículo. En la vida real, los sensores podrían no ser capaces de captar los valores de los parámetros del vehículo tan rápido. También podría haber pérdidas o escasez de datos debido al uso de condiciones de disparo basadas en eventos. Estos escenarios pueden ser contemplados con un filtro de Kalman extendido bifrecuencia (Dual Rate Extended Kalman Filter, DREKF). Tendrá dos finalidades: primero, estimar (corregir) cuando sea necesario los estados no perdidos y segundo, predecir los estados en los momentos en que los sensores no han sido actualizados a tiempo [5-9].

El objetivo del vehículo es seguir una ruta. Como no hay requerimientos temporales en el seguimiento, se trata de *path-following* control. Si los hubiese, sería *path-tracking* control. Además, es un problema de control lateral, porque los modelos del vehículo tienen como entrada el ángulo de giro [2].

La transformación de posición al ángulo se realiza con el algoritmo de seguimiento de rutas Pure Pursuit de [5]. Es un algoritmo comúnmente usado para este tipo de aplicaciones relacionadas con el control de AGVs [4]. Además de funcionar bien con los controladores IKIBI y MPC, está demostrado que se acopla bien a otros tipos de controladores, como el árbol de exploración aleatoria rápida (Rapidly Exploring Random Tree, RRT). Se trata de un algoritmo de búsqueda que cumple la finalidad de un algoritmo de control al implementarse en un bucle cerrado, como se ilustra en [11].

Al final de la memoria, se realizarán comparaciones para poner a pruebas las técnicas mencionadas y se extraerán conclusiones.

1.2 Herramientas.

El entorno de desarrollo de este trabajo ha sido MATLAB (MATrix LABoratory, laboratorio de matrices). Es un programa avanzado de cómputo numérico.

Este apartado se incluye con el propósito de que todo el mundo, y no solo los que estén familiarizados con esta aplicación, puedan entender mejor esta memoria.

MATLAB cuenta con un lenguaje interpretado propio de alto nivel. El nivel aumenta a medida que es más fácil de entender por el programador de manera que el nombre de las funciones es autoexplicativo en muchas ocasiones.

El aspecto de la aplicación de escritorio es:

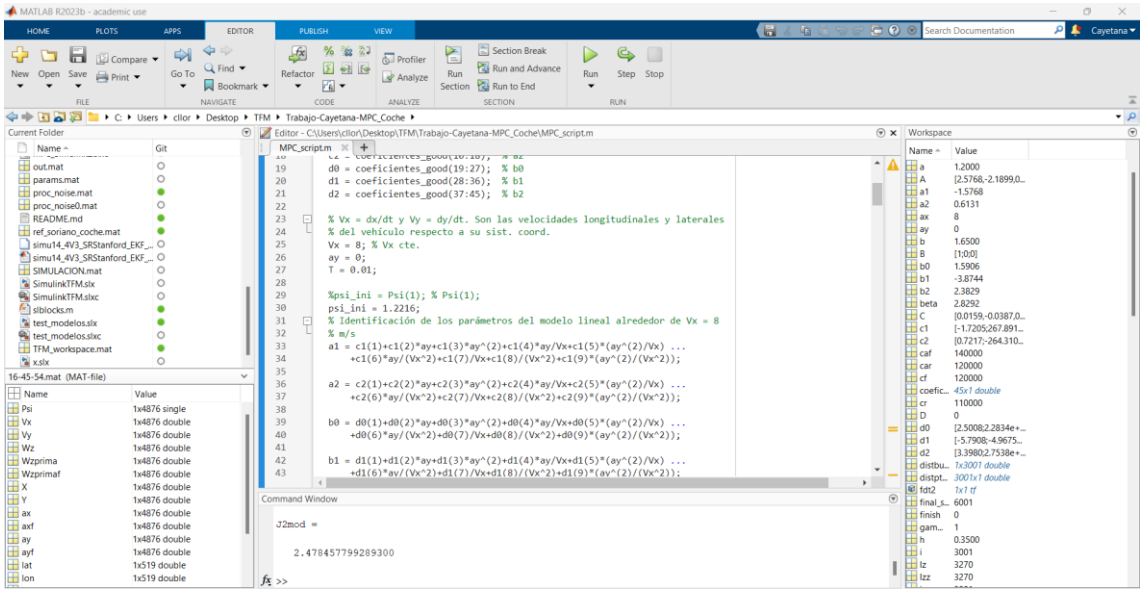

Figura 1. Aplicación de escritorio de MATLAB.

La sección *Current Folder* muestra el directorio en el que se ha abierto la aplicación. Solo se pueden abrir los archivos incluidos en la ruta de búsqueda o en la carpeta actual.

En el editor, *Editor*, se gestionan los scripts que se van a ejecutar. Todos los elementos que se van a explorar en este trabajo se pueden crear y manejar desde un script. Estos son archivos de texto con el lenguaje de MATLAB con terminación .m

Command Window acepta comandos del programador y los ejecuta.

Workspace es el entorno de trabajo. Guarda las variables de las ejecuciones realizadas. Está comunicado directamente con la *Command Window*.

Simulink es un entorno de diseño y simulación orientado a bloques. Es un método más intuitivo. En esta memoria se recalcarán las ventajas y desventajas entre la programación con ambos entornos. También se explicará cómo se conectan.

Los bloques se Simulink, según su función, se agrupan en bibliotecas, esto es importante porque se pueden crear bibliotecas personalizadas. Para este trabajo se ha hecho. Esto permite que terceros puedan manejar fácilmente los bloques, utilizarlos en casos distintos a los planteados y ver cómo se comportan. Toda la información disponible acerca de las bibliotecas personalizadas de MATLAB se encuentra en [12].

Otra cuestión práctica de MATLAB es que es compatible con Github, lo cual facilita su distribución y el trabajo colaborativo.

Capítulo 2. Estrategia de control

2.1 Planteamiento

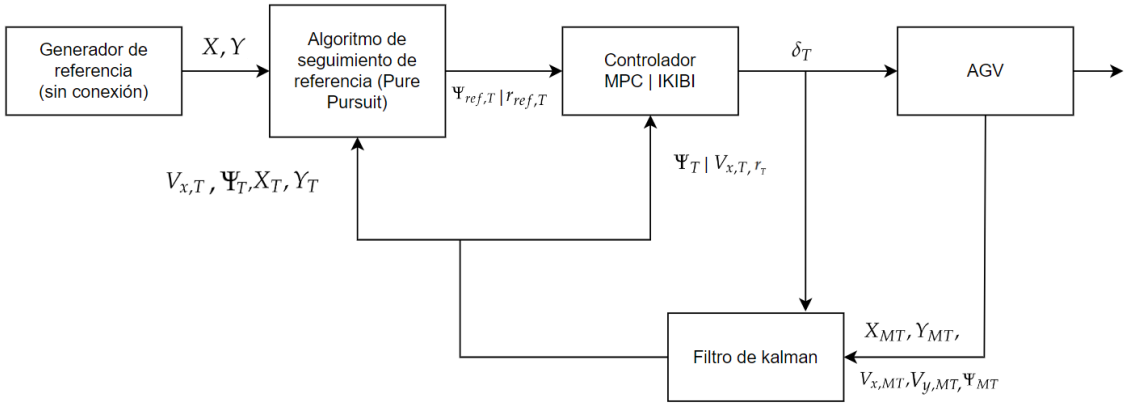

Figura 2. Control en bucle cerrado propuesto. KF, filtro de Kalman.

Primero se escoge la ruta a seguir (X, Y), entonces el generador de referencia Pure Pursuit asegura que haya un punto de referencia adecuado para el vehículo de guiado automático (Automatic Guided Vehicle, AGV) y para el momento de la simulación. El bucle cerrado es el propuesto en [6], aunque con un EKF en vez de un DREKF.

Este caso base se denominará situación nominal. MT, el periodo de sensorización, es 0.1 segundos. El periodo de control es 10 veces menor, de manera que T = 0.01 segundos.

La nomenclatura de la Figura 2 utiliza estos periodos como subíndices de las diferentes variables. Mientras no se implementa el filtro, en las pruebas iniciales, cuando se evalúan los modelos del AGV, no se considera el periodo de sensorización, es decir, se utiliza el mismo periodo T para sensorización y control. El caso completo sí contempla los dos periodos, tal como muestra la Figura 2.

Para este problema de control, las características de la planta indican que la entrada del sistema sea el ángulo de dirección (δ). Los bloques explicados en esta sección tienen como función adicional la de transformar los datos para que sirvan al bloque al que están conectados.

El controlador tomará el ángulo de guiñada y su referencia en el caso del MPC (Ψ, Ψ_{ref}) o su derivada, cuando se escoge el controlador IKIBI ($\dot{\Psi}$, o también r, r_{ref}), y devolverá el ángulo de dirección óptimo ($\delta_{óptimo}$).

El algoritmo Pure Pursuit toma la pose de referencia y la posición actual del robot para obtener el ángulo de guiñada de referencia. Su derivada, la calcula, además, con la velocidad horizontal del AGV (Vx), y con el ángulo de guiñada en ese momento.

El algoritmo Pure Pursuit, el controlador MPC y el KF están optimizados. Por lo tanto, los errores de seguimiento se pueden reducir a través de varias vías.

Finalmente, se reducen errores de la pose del robot $[X \quad Y \quad \Psi]$ con el KF y se actualiza el punto de referencia, $[X \quad Y \quad \Psi]_{ref}$.

2.2 Índices de coste.

Existen varios mecanismos para lograr las funciones descritas en la Figura 2. Por un lado, por ejemplo, dentro de los controladores, se contemplan varios tipos: IKIBI y MPC. Por otro lado, dentro del algoritmo Pure Pursuit, aun tratándose del mismo método, se han

contemplado dos vías distintas de implementarlo: programando las fórmulas directamente y partiendo de un bloque incluido en la librería de Simulink.

Con el propósito de cuantificar la eficiencia de los diferentes métodos se han calculado varios índices de coste [5], [6].

El índice de coste J_1 mide la exactitud con la que se ha seguido la trayectoria.

$$J_1 = \sum_{k=1}^{l} \min_{1 \le k' \le l} \sqrt{\left(X_k - X_{ref,k'}\right)^2 + \left(Y_k - Y_{ref,k'}\right)^2} \quad (1)$$

l: Número de iteraciones necesarias para que el vehículo llegue al final del camino.

$(X, Y)_k$: Posición actual del vehículo.

$(X_{ref}, Y_{ref})_{k'}$: Es la posición de la referencia más cerca del vehículo.

El mismo índice normalizado, J_{1norm}, también se ha considerado.

$$J_{1norm} = \frac{\sum_{k=1}^{l} \min_{1 \le k' \le l} \sqrt{\left(X_k - X_{ref,k'}\right)^2 + \left(Y_k - Y_{ref,k'}\right)^2}}{t_{sim}} \quad (2)$$

Donde t_{sim} es el tiempo total de la simulación.

El índice de coste J_2 mide la exactitud con el error máximo que se ha cometido al seguir la referencia.

$$J_2 = \max_{1 \le k \le l} \left\{ \min_{1 \le k' \le l} \sqrt{\left(X_k - X_{ref,k'}\right)^2 + \left(Y_k - Y_{ref,k'}\right)^2} \right\} \quad (3)$$

Para tratar aspectos de comunicación basada en eventos periódicos (PETC), se introducen los índices de coste J_{3c}, controlador, y J_{3s}, sensor, que miden el porcentaje de transmisiones según el criterio establecido en la comunicación PETC en comparación con las que se harían con un sistema de control basado en el tiempo (time-triggered control, TTC).

$$J_3 = \frac{NoT_{PETC}}{NoT_{TTC}} 100\% \quad (4)$$

NoT_{PETC}: Cantidad de envíos con PETC.

NoT_{TTC}: Cantidad de envíos con TTC.

Otro índice de coste, J_4, utilizado solo en [5], evalúa la comodidad que siente el pasajero al montarse en el vehículo. Aumenta junto a medida que crece la velocidad con la que varía el ángulo de dirección. Sirve sobre todo para la comparación de restricciones sobre la variable manipulada de control. Este tipo de restricciones son fácilmente aplicables en controladores MPCs en MATLAB.

$$J_4 = \frac{\sum_{k=1}^{l-1} \|\delta(k+1) - \delta(k)\|}{t_{sim}} \quad (5)$$

Aumentará si las variaciones en la acción de control son grandes, lo que se traduce en cambios bruscos en la dirección.

Capítulo 3. Modelos

3.1 Modelo lineal LPV

El controlador lateral del vehículo, es decir, de la dirección, puede ser un MPC con el siguiente modelo:

$$\dot{\psi}(k) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \delta(k) \quad (6)$$

$\dot{\psi}(k)$: Velocidad de giro del vehículo.

b_0, b_1, b_2, a_1, a_2: Coeficientes obtenidos experimentalmente en el laboratorio [8,9]. Cambian según la componente horizontal de la velocidad, Vx. También explicado en [6].

$\delta(k)$: Ángulo de orientación de la rueda delantera.

Estas definiciones tienen en cuenta el sistema de referencia que corresponde al modelo de bicicleta de Rajamani y que se muestra en la siguiente imagen:

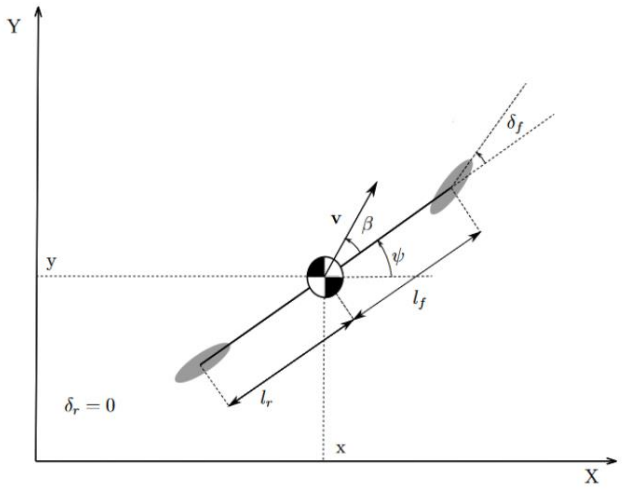

Figura 3. Modelo de la bicicleta.

Se asume que el ángulo de giro de la rueda trasera es nulo, por lo que ($\delta_f = \delta$).

l_r y l_f son las distancias desde el centro de gravedad a los ejes trasero y delantero respectivamente.

La letra subíndice r siempre hará referencia en este trabajo a la rueda trasera ('rear' en inglés). Igualmente, f está asociado con rueda delantera ('front' en inglés).

Es preferible tratar directamente con el ángulo de giro, no la velocidad, ya que la ruta está definida en términos de posición y orientación.

Al integrar la velocidad y sustituir en la ecuación anterior:

$$\psi(k) = \frac{Tb_0 z^{-1} + Tb_1 z^{-2} + Tb_2 z^{-3}}{1 + (a_1 - 1) z^{-1} + (a_2 - a_1) z^{-2} - a_2 z^{-3}} \delta(k) \quad (7)$$

T es el periodo de muestreo.

El modelo discreto en espacio de estados queda de la siguiente forma:

$$x(k+1) = Ax(k) + Bu(k) \quad (8)$$

$$y(k) = Cx(k) + Du(k) \quad (9)$$

$u(k-1)$ es la acción de control. $u(k-1) = (a_x(k-1), \delta(k-1))^T$. Se ha asumido $a_x(k) = 0\ \forall k$, por lo que $u(k-1) = \delta(k-1)$.

x es el vector de estados $[\Psi, X, Y]^T$.

y es el ángulo de guiñada Ψ. También la salida del modelo.

A es la matriz de transición de estados.

$$A = \begin{bmatrix} 1-a_1 & a_1-a_2 & a_2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (10)$$

B es la matriz de entrada.

$$B = [1 \quad 0 \quad 0]^T \quad (11)$$

C es la matriz de salida.

$$C = [Tb_0 \quad Tb_1 \quad Tb_2] \quad (12)$$

D es la matriz de alimentación del sistema.

$$D = 0 \quad (13)$$

3.2 Modelo no lineal de bicicleta de Rajamani

Según el estudio realizado en [6], este modelo está comprobado que es fiable y el error respecto al comportamiento real es muy pequeño.

$$F_{yf}(k) = -C_{\alpha f}\arctan\left(\frac{V_y(k-1) + r(k-1)l_f}{\max(V_x(k-1), V_{\min})} - \delta(k-1)\right) \quad (14)$$

$$F_{yr}(k) = -C_{\alpha r}\arctan\left(\frac{V_y(k-1) + r(k-1)l_r}{\max(V_x(k-1), V_{\min})}\right) \quad (15)$$

$$a_x(k) = a_x(k-1) \quad (16)$$

$$a_y(k) = -V_x(k-1)r(k-1) + \frac{F_{yf}(k-1) + F_{yr}(k-1)}{m} \quad (17)$$

$$\dot{r}(k) = \frac{l_f F_{yf}(k-1)\cos\big(\delta(k-1)\big) - l_r F_{yr}}{I_{zz}} \quad (18)$$

$$V_x(k) = V_x(k-1) + Ta_x(k-1) \quad (19)$$

$$V_y(k) = V_y(k-1) + Ta_y(k-1) \quad (20)$$

$$r(k) = r(k-1) + \dot{r}(k-1) \quad (21)$$

$$X(k) = X(k-1) + T[V_x(k-1)\cos\big(\psi(k-1)\big) - V_y(k-1)sin\big(\psi(k-1)\big)] \quad (22)$$

$$Y(k) = Y(k-1) + T[V_x(k-1)\cos\big(\psi(k-1)\big) + V_y(k-1)sin\big(\psi(k-1)\big)] \quad (23)$$

$$\psi(k) = \psi(k-1) + Tr(k-1) \quad (24)$$

m: Masa del vehículo.

$l_{f/r}$: Distancia de las ruedas delantera y trasera, respectivamente, desde el punto de proyección normal del centro de gravedad del vehículo sobre el plano del eje común.

I_{zz}: El momento de inercia del cuerpo del vehículo en torno al eje fijo Z del cuerpo.

$C_{\alpha,f/r}$: Rigidez en curva. Esta constante representa una aproximación lineal de la relación entre el ángulo de deslizamiento (α) y la fuerza lateral (F_y).

μ: Coeficiente de fricción del vehículo.

h: Altura del centro de gravedad del vehículo por encima del plano del eje.

$F_{yf/r}$: Fuerza lateral de las ruedas.

α: Ángulo de deslizamiento.

r: Velocidad de giro ($\dot{\psi}$).

V_{min}: Velocidad mínima

Los valores de estos parámetros son los del coche de Berkeley, 2017 Lincoln MKZ:

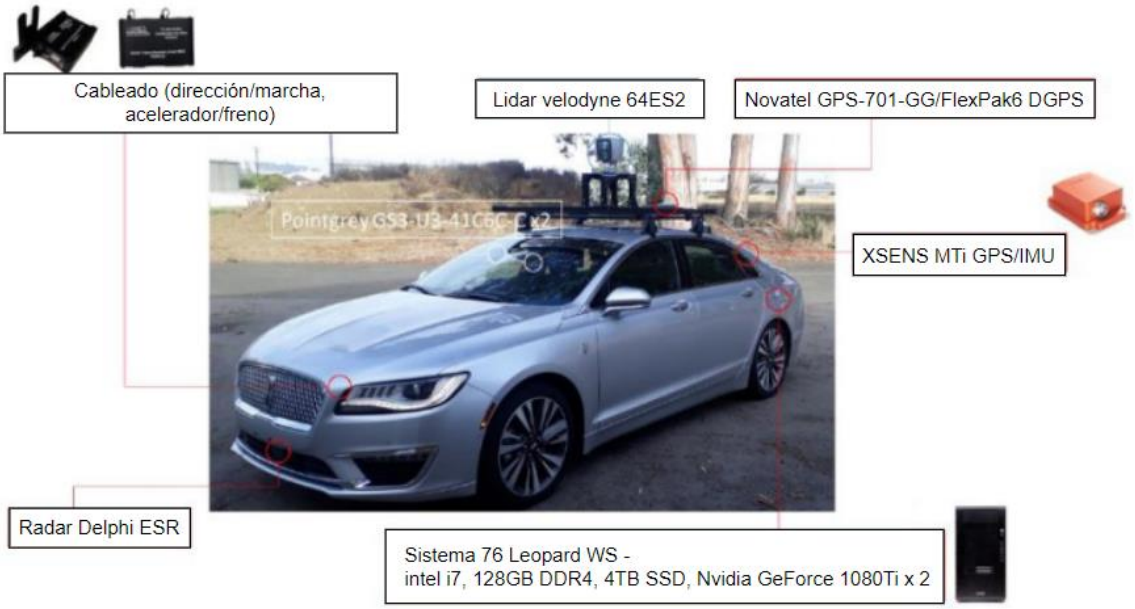

Figura 4. Coche usado en las pruebas.

Parámetros	Valor	Unidad
$\boldsymbol{m}$	1800	kg
$\boldsymbol{l_f}$	1.2	m
$\boldsymbol{l_r}$	1.65	m
$\boldsymbol{\mu_f}$	0.6	-
$\boldsymbol{\mu_r}$	0.6	-
$\boldsymbol{C_{\alpha f}}$	140	kN/rad
$\boldsymbol{C_{\alpha r}}$	120	kN/rad
$\boldsymbol{h}$	0.35	m
$\boldsymbol{I_{zz}}$	3270	Kg·m^2
$\boldsymbol{V_{min}}$	2.23	m/s

Tabla 1. Parámetros del modelo de la bicicleta de Rajamani.

Las ruedas del coche no pueden girar más de 0.32 radianes en ninguno de los dos sentidos y la velocidad angular de la guiñada debe estar comprendida entre [0.84, -0.84] rad/s. Estas restricciones no se reflejan directamente en las ecuaciones del modelo LPV. Sin embargo, hay otras formas de incorporarlas en el entorno Simulink. Los modelos no lineales saturan δ y Ψ para que cumplan las restricciones.

3.3 Modelo no lineal de Stanford.

Es otra propuesta para modelar un coche autónomo [5].

El modelo dinámico define las fuerzas laterales de las ruedas delantera y trasera de la siguiente forma:

$$F_{yf} = -C_{\alpha f} \arctan\left(\frac{V_y + r l_f}{\max\left(V_x, V_{min}\right)}\right) - \delta \quad (25)$$

$$F_{yr} = -C_{\alpha r} \arctan\left(\frac{V_y - r l_f}{\max\left(V_x, V_{min}\right)}\right) \quad (26)$$

$$V_x(k) = V_x(k-1) + T a_x(k-1) \quad (27)$$

$$\begin{aligned} V_y(k) = V_y(k-1) \\ + T\Big[\tan\left(\delta(k-1)\right)\left(a_x(k-1) - r(k-1)V_y(k-1)\right) \\ + \frac{F_{yf}}{m\cos\left(\delta(k-1)\right)} + \frac{F_{yr}}{m} - r(k-1)V_x(k-1)\Big] \end{aligned} \quad (28)$$

$$x(k) = x(k-1) + T\left[V_x(k-1)\cos\left(\psi(k-1)\right) - V_y(k-1)\sin\left(\psi(k-1)\right)\right] \quad (29)$$

$$y(k) = y(k-1) + T\left[V_x(k-1)\sin\left(\psi(k-1)\right) - V_y(k-1)\cos\left(\psi(k-1)\right)\right] \quad (30)$$

$$\psi(k) = \psi(k-1) + Tr(k-1) \quad (31)$$

El caso que se va a estudiar en este trabajo, para reducir la dificultad, es con aceleración horizontal nula (a_x = 0) y velocidad en esa misma orientación constante.

$$r(k) = r(k-1) + T\left[\frac{l_f F_{yf}\cos\left(\delta(k-1)\right) - l_r F_{yr}}{I_z}\right] \quad (32)$$

$$a_y(k) = -V_x(k-1)r(k-1) + \frac{F_{yf}\cos\left(\delta(k-1)\right) + F_{yr}}{m} \quad (33)$$

El valor de los parámetros se considerará igual para ambos modelos. En [5] los coeficientes de rigidez en curva y la distancia de las ruedas delantera y trasera varían un poco, suficiente como para desestimarlo ya que no es especialmente relevante para el objeto de estudio de este trabajo.

3.4 Comparación de modelos.

Se ha empleado el entorno de programación virtual, Simulink, junto a MATLAB para realizar las simulaciones.

La estructuración en la interfaz de Simulink es distinta a la de un script de MATLAB. Las ventajas y desventajas entre ambas formas de programación se discutirán en la memoria.

Un aspecto importante en control en tiempo real es el tiempo de muestreo. Los coeficientes del vehículo para el modelo lineal fueron tomados para un tiempo de muestreo de T = 0.01 segundos. El mismo controlador puede no ser lo suficiente robusto o eficaz para referencias distintas.

La ruta escogida para el Lincoln MKZ (2017) contiene dos giros rápidos de 90º y un giro de 180º. En la misma fuente en la que se encontraron estos datos venía la referencia también en función de psi. Sin embargo, tenía un periodo de muestreo de T = 0.02 segundos.

Era necesario otro método para evaluar la precisión de los modelos independientemente del generador de referencia, de manera que uno se pueda asegurar que el buen seguimiento de la ruta es por el modelo escogido y no por la acción del Pure Pursuit.

3.4.1 Generación de referencias.

Con el propósito de generar referencias para los distintos modelos, se ha empleado un archivo en Simulink, *generador_referencias.slx*.

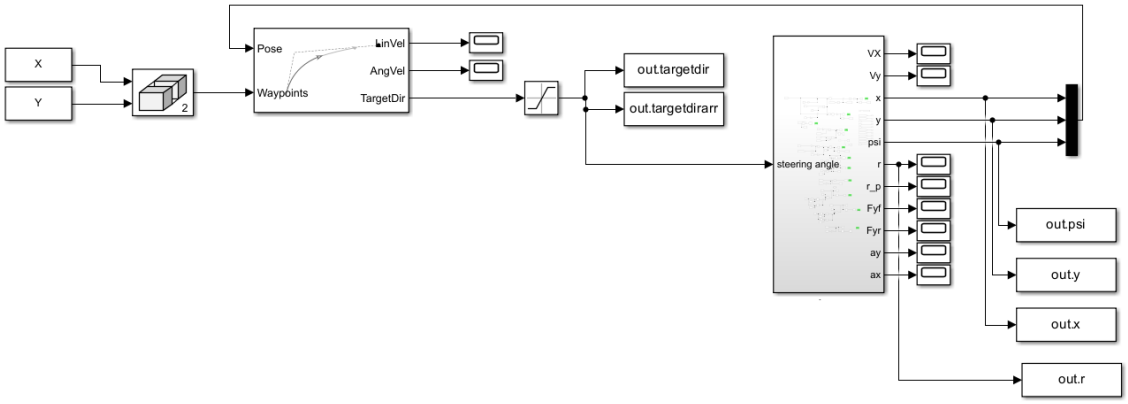

Figura 5. Generador de referencias con bloque Pure Pursuit y modelo no lineal de Rajamani.

Partiendo de una trayectoria [X, Y], con vectores de posición unidos con el bloque *Matrix concatenate*, se obtienen los puntos guía para generar la referencia. El bloque *Pure Pursuit* [13] usa el algoritmo con el mismo nombre para calcular comandos de velocidad lineal y angular para seguir una trayectoria utilizando un conjunto de puntos de referencia o guía y realimentando con la pose actual del vehículo de tracción diferencial (un vector con la posición y el ángulo de orientación [X, Y, ψ]). Al realimentarse se convierte en control en bucle cerrado.

El bloque *Pure Pursuit*, aunque se use en este escenario para generar una referencia de prueba, es un controlador. Genera la velocidad y aceleración horizontal para un vehículo con tracción diferencial y puede configurarse para que devuelva el ángulo de dirección.

La dirección en la que avanza del vehículo se considera cero radianes, con ángulos positivos medidos en sentido contrario a las agujas del reloj. Esta misma es la definición de la entrada de los modelos del vehículo.

Usar bloques facilita el trabajo de los ingenieros ya que no tienen que conocer en profundidad el funcionamiento del algoritmo. Posteriormente, en el Capítulo 6, se presentará en esta memoria otro generador de referencias, también Pure Pursuit, conformado por ecuaciones. Es un planteamiento diferente que permite configurar el algoritmo del Pure Pursuit, por si se quiere modificar.

La trayectoria al ser independiente del tiempo consta en el espacio de simulación como bloques del tipo *Constant*. Sin embargo, las salidas tanto del Pure Pursuit como del modelo están asociadas a instantes de tiempo específicos y serán del tipo *timeseries.*

Se necesita el bloque *To workspace* para poder llevarlas al espacio de trabajo de MATLAB desde Simulink.

Como Simulink está orientado al tiempo, está diseñado para representar el valor de las variables en relación con el tiempo de la simulación. Por el contrario, MATLAB es mejor para representarlas en función a otras variables.

Tampoco es útil trasladarlas en cualquier formato. Los tipos de formatos disponibles son:

- Timeseries: Objeto de que se utiliza para representar y manejar series temporales. Tiene métodos y propiedades específicos de su clase.
- Structure with time: Estructura que guarda por separado los datos y el tiempo en que se obtuvieron.
- Structure: Agrupación de datos heterogéneos.
- Array: Agrupación de datos homogéneos.

Una verificación realizada para comprobar que el modelo es correcto es comparar el comportamiento con el de un bloque de Simulink *Vehicle Body 3DOF Dual Track* que modela cuerpos rígidos con 3 grados de libertad con las ecuaciones de [14].

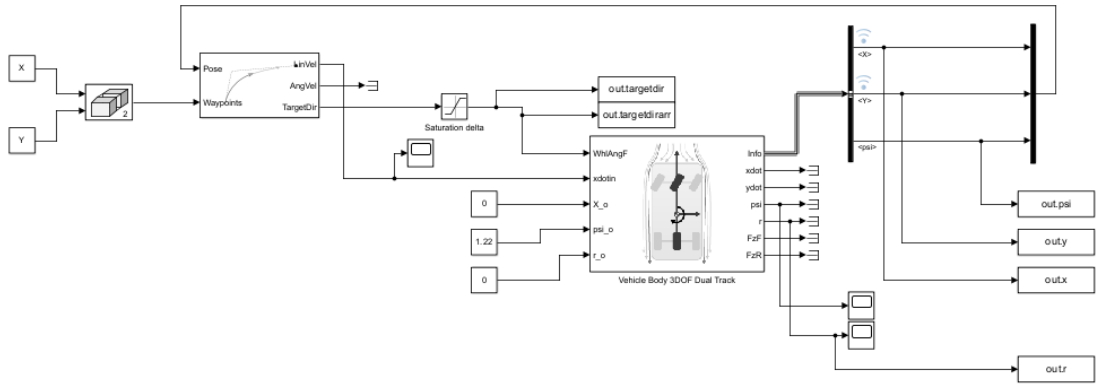

Figura 6. Generador de referencias con bloque Pure Pursuit y modelo con bloque Vehicle body 3DOF Dual Track.

A ambos se les ha asignado los parámetros de la Tabla 1 y deben de devolver un ángulo de giro parecido cuando completan la trayectoria.

Entonces comparamos los modelos propuestos. De la prueba anterior se extrae un ángulo de dirección controlado que permite al vehículo seguir la trayectoria sin desviarse para cada modelo. Si los modelos están correctamente implementados, todos los ángulos de dirección calculados deberían ser semejantes.

La saturación del ángulo de dirección, δ, se ha considerado. La del ángulo de guiñada, Ψ, no ha sido posible puesto que no existe la posibilidad de configurarlo en el bloque.

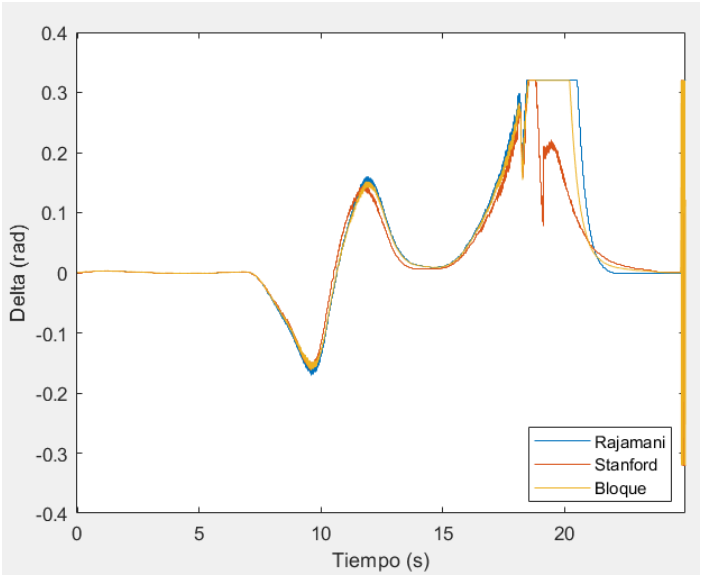

Figura 7. Ángulo de dirección controlado para los modelos no lineales.

El modelo del bloque da la opción de incluir características medioambientales como la velocidad del viento, fricción, momentos externos o incluso fuerzas de enganche si se acopla un remolque. Una forma de continuar con el trabajo de este TFM sería sustituir en el bucle de control final, para todos los casos estudiados, el modelo no lineal por modelos

más realista. El bloque *Vehicle Body 3DOF Dual Track es una opción*, que podría servir de paso intermedio antes de incluir un modelo mucho más detallado hecho con Simscape Multibody, una herramienta que sirve expresamente para modelar y simular sistemas mecánicos multicuerpo [4,7]. Es el tema del TFM de Abraham Planells Bolós, otro alumno de la UPV, que se titula *Modelado y simulación realistas para control de seguimiento de trayectorias en vehículo autónomo*.

Realmente, lo que comenzó siendo parte de una prueba para cerciorarse de que el modelo no lineal es adecuado, ha terminado siendo un control Pure Pursuit basado en un modelo de vehículo con tracción diferencial para dos modelos diferentes: el de Rajamani y el de la bicicleta que proporciona Simulink para el bloque con el mismo propósito.

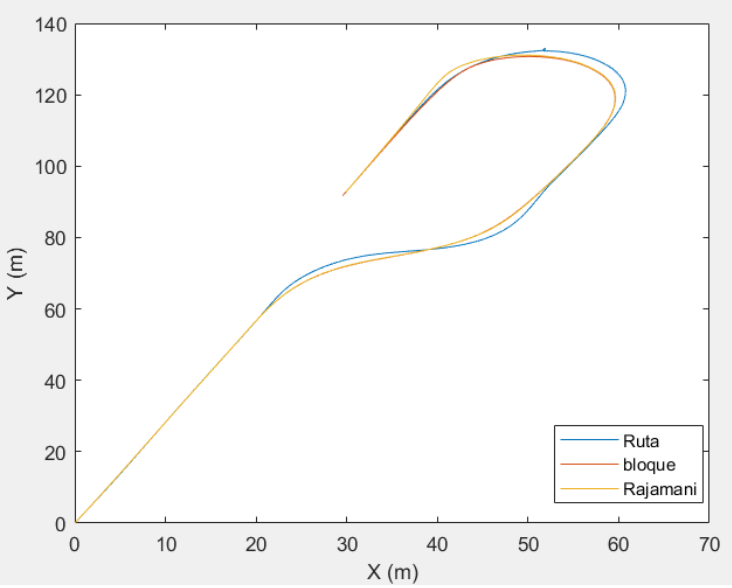

Figura 8. Modelo no lineal de Rajamani versus el del bloque de Simulink.

Índice	Bloque de Simulink	Rajamani
J1	1.385’9545	1.133’8035
J1 normalizado	0’2842	0’2325
J2	2’043	1’9242

Tabla 2. Índices de los modelos no lineales de Rajamani y el bloque de Simulink.

El bloque de Simulink es ligeramente peor en este caso. Se podría mejorar el resultado ajustando las características que no se encuentran reflejadas en el modelo de Rajamani. No se ha hecho porque no es el objetivo de este trabajo.

A continuación, una comparación entre el modelo de Rajamani (naranja) y el modelo lineal LPV (azul). Para ello, se emplean como referencia los ángulos de dirección de la Figura 7. Se realiza una combinación entre el control Pure Pursuit + IKIBI y los diferentes

modelos. La Tabla 2 muestra que los índices extraídos del modelo no lineal de Rajamani son menores. Así que se usó ese dato de referencia.

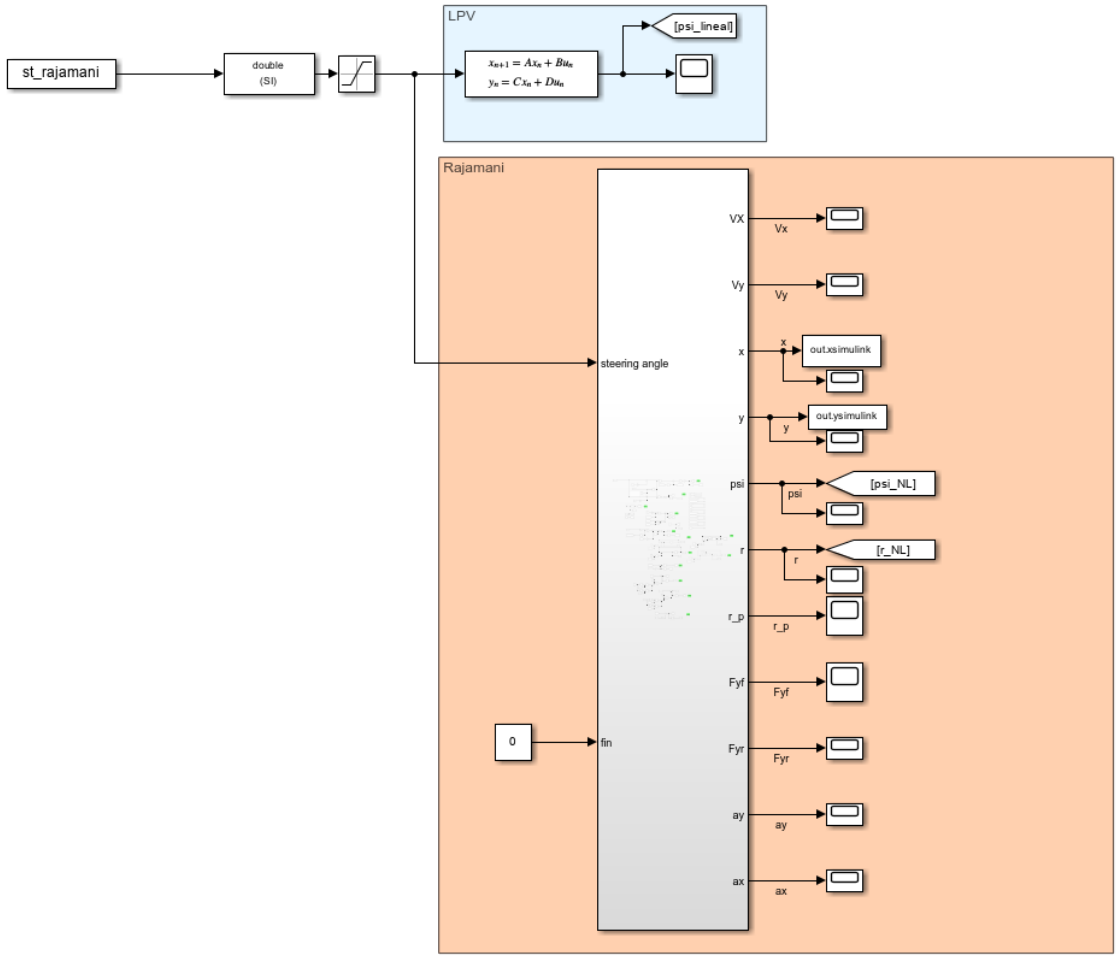

Figura 9. Comparación de modelos con control Pure Pursuit + IKIBI en Simulink.

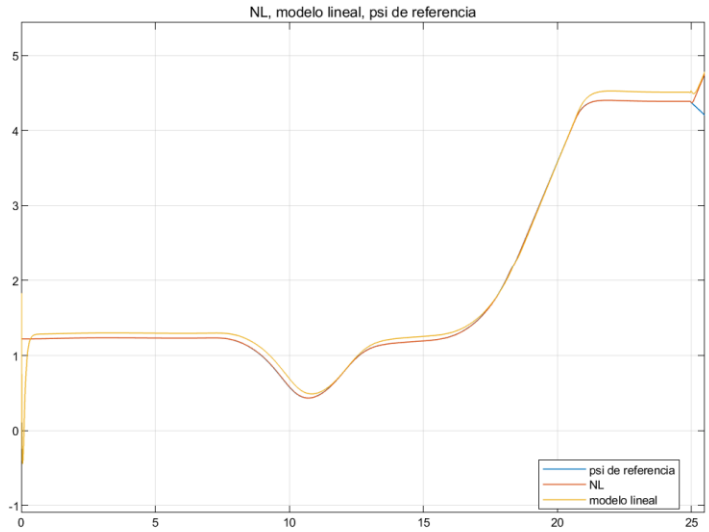

Figura 10. Comparación ángulo de guiñada del control Pure Pursuit + IKIBI con el modelo de Rajamani versus el LPV.

El modelo no lineal de Rajamani (NL en la leyenda) y la referencia son iguales. El modelo lineal no se inicializa correctamente. Durante el primer periodo de muestreo el ángulo tiene el valor 1.2216. Es preciso que el vehículo comience girado o no podrá seguir la trayectoria deseada, por las restricciones sobre el ángulo de giro sobre las ruedas y velocidad máximos.

El ángulo de inicialización del espacio de estados no es el mismo que el del modelo. Necesita ser transformado según la siguiente ecuación:

$$1.2216/(T * b0) = 76.799 \quad (34)$$

En un diseño posterior, al introducir el controlador MPC el desfase del principio se eliminará. No ocurre instantáneamente, por lo que esta parte (fase de inicialización) no entrará dentro del rango temporal elegido para calcular índices que midan la eficiencia de los controladores.

Es estudio en [8], compara diversos modelos dinámicos y llega a la conclusión de que el de Rajamani es el que tiene el menor error absoluto de media. El vehículo modelado es el mismo y de las rutas utilizadas también. Por eso, aunque en los archivos adjuntos de este trabajo también se encuentra el modelo Stanford, acabó siendo descartado.

Índice	LPV	Rajamani
J1	60’7822	7’5750
J1 normalizado	0’0238	0’0029
J2	0’8754	0’3556

Tabla 3. Índices de los modelos no lineales de Rajamani y LPV.

Se han modificado los índices para medir la exactitud con la que se sigue la referencia y el punto más alejado de esta respecto a Ψ, en vez de X e Y.

Al estar la referencia extraída expresamente para el modelo de Rajamani, los índices son más pequeños. Y no sigue la trayectoria perfectamente porque la saturación de δ está colocada fuera del controlador Pure Pursuit. De nuevo, al tratarse de un bloque incluido en Simulink, hay aspectos que no se pueden modificar.

Todas las gráficas representadas a partir de este capítulo de la memoria usarán el modelo de Rajamani.

Capítulo 4. Controladores

Volviendo al caso presentado en el Capítulo 2, concretamente en la Figura 2, este capítulo se centrará en la parte de los controladores.

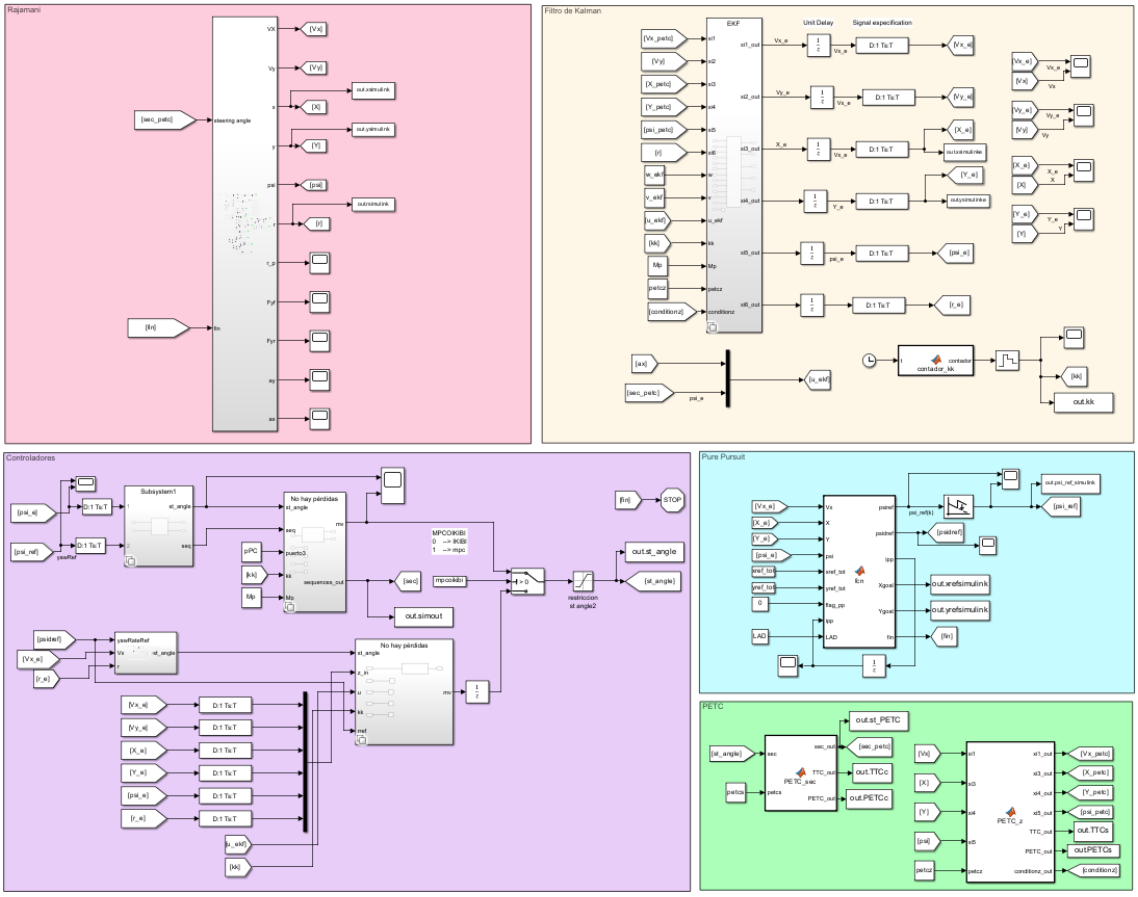

Figura 11. Implementación del bucle de control propuesto en la figura 2 en Simulink.

El controlador MPC se realimenta con el ángulo de guiñada y el controlador IKIBI con la velocidad del ángulo de guiñada.

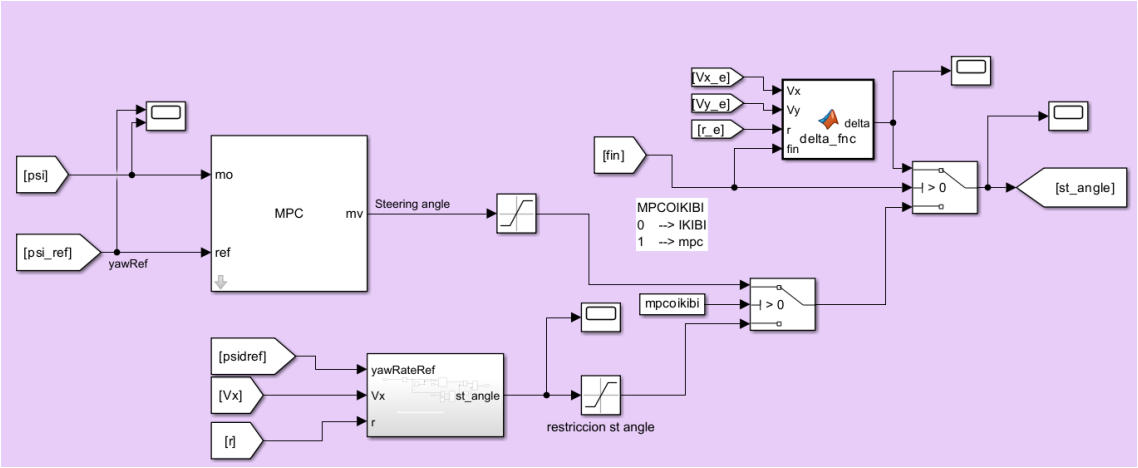

Figura 12. Ampliación del bucle de control. Parte de los controladores.

El ángulo de guiñada de referencia se actualiza con la información que viene del coche, tal y como se ha hecho en la sección anterior. En la parte izquierda de la Figura 12 están los controladores. La variable *mpcoikibi* está diseñada de manera que para usar el controlador IKIBI tiene que ser menor que 0, y al contrario para el MPC. Por comodidad, se han empleado los valores 0 y 1 según el criterio a continuación:

mpcoikibi	
0	IKIBI
1	MPC

Tabla 4. Criterio de selección entre controladores.

La parte derecha de la Figura 12 (bloque delta_fnc) es un método aparte implementado para frenar el vehículo al terminar el recorrido. Se explica en detalle en el Capítulo 6 , Pure Pursuit.

4.1 Controlador MPC

4.1.1 Teoría

El controlador MPC, como su nombre indica, se basa en modelos para calcular acciones de control futuras.

Existen por un lado los MPC lineales y los no lineales, estos últimos normalmente emplean modelos linealizados alrededor de un punto. Ese punto, en el caso del modelo LPV extraído a partir del modelo no lineal de Rajamani es la velocidad horizontal Vx = 8 m/s. A medida que la velocidad aumenta, el controlador rendirá menos.

El controlador MPC recibe en un instante inicial la posición del vehículo. Luego, prueba distintas acciones de control y calcula la nueva posición del vehículo. El horizonte de control es el número de acciones de control calculadas en el futuro. Cuanto mayor es, significa que el controlador predice más. De esas acciones calculadas, no todas se aplican siempre. Esto es porque las acciones se calculan con información únicamente del instante actual, que queda desactualizada a medida que pasa el tiempo, hasta perder toda su relevancia. No solo eso, sino que además con cada iteración dentro del bucle cerrado se van acumulando errores no previsibles.

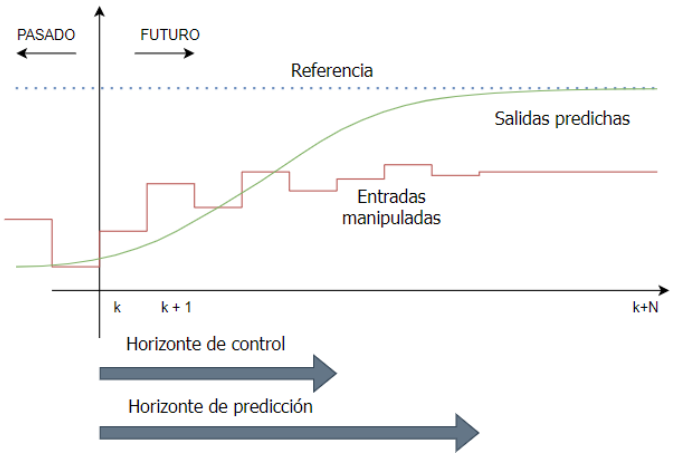

Figura 13. Esquema del funcionamiento de un controlador MPC.

La clave está en encontrar un horizonte de predicción y uno de control que permitan seguir la referencia eficazmente y que no implique un coste computacional demasiado elevado.

Para elegir entre todas las opciones cuáles son las acciones de control que se van a aplicar, se minimiza un índice de coste, que incluye las restricciones. Controladores más simples como el IKIBI no suelen admitir restricciones. Al formular el índice de coste, se pueden representar las restricciones de forma numérica con notaciones del tipo mayor que o menor que ($\geq, \leq$).

La función de coste cuadrática estándar tiene esta forma:

$$J(x(0),U) = \sum_{k=0}^{N-1} [(xref - x'(k))Q(xref - x(k)) + (uref - u'(k))R(uref - u(k))] + (xref - x'(N))P(xref - x(N)) \quad (35)$$

$J(x(0),U)$: Función de coste total que se quiere minimizar. Representa la suma ponderada de los errores en los estados y las entradas a lo largo del horizonte de predicción.

$xref$: Vector de referencia del estado de la planta. Este es el objetivo que se desea alcanzar para los estados del sistema.

$x(k)$: Vector del estado de la planta en el instante actual (k).

Q: Matriz de ponderación para los errores en los estados. Es una matriz diagonal que penaliza las desviaciones de los estados respecto a sus valores de referencia. Normalmente es simétrica y semidefinida positiva.

$uref$: Vector de referencia de las entradas de control. Este es el objetivo que se desea alcanzar para las entradas de control.

$u(k)$: Entrada de control del sistema en el presente.

R: Matriz de ponderación para los errores en las entradas de control. Es una matriz diagonal que penaliza las desviaciones de las entradas respecto a sus valores de referencia. Normalmente es simétrica y semidefinida positiva.

P: Matriz de ponderación final para los errores en los estados en el último paso de tiempo N. Es una matriz diagonal que penaliza las desviaciones de los estados respecto a sus valores de referencia al final del horizonte de predicción. Normalmente es simétrica y semidefinida positiva.

Imaginemos que tenemos un horizonte de predicción $N = 5$ y un horizonte de control $M = 3$. En los primeros tres instantes el AGV no se acerca mucho a la referencia. Sin embargo, el último de los movimientos considerados en N reduce mucho el error. El índice de coste se reducirá para este conjunto de 5 pasos.

De esos 5 pasos, solo se van a aplicar 3. Después, el proceso de evaluación de posibles acciones de control a tomar comenzará de nuevo. El último paso, que era el mejor de todos, no se ha llegado a aplicar.

Con el fin de dar prioridad a los primeros movimientos, se penalizan los últimos más con la matriz P. Incluso si ($N = M$), el error es acumulativo con $M > 1$.

Esto es debido a que se eligen M pasos con información del presente (instante k), y se aplican en distintas iteraciones hasta el instante $k + (M - 1)$. Ejemplo: Para $M = 3$, aplico las acciones de control en $(k, k + 1, k + 2)$.

Los parámetros elegidos han sido:

Parámetro	Valor
Horizonte de predicción (N)	20
Horizonte de control (M)	10
Q	1
R	0.001

Tabla 5. Parámetros del controlador MPC.

En el bloque de Simulink *MPC Controller* se admiten restricciones sobre la variable manipulada y se ha incluido que el ángulo de giro esté comprendido entre [-0.32,0.32] radianes.

La restricción relacionada con la velocidad de variación del ángulo de guiñada no se podía incluir directamente en el controlador, ya que el bloque no tiene esa opción. Se ha incluido con un bloque de saturación con el objetivo de que varíe entre [-0.84,0.84] radianes por segundo.

Sin embargo, sí deja la opción de imponer restricciones sobre la variable manipulada, el ángulo de giro. Las simulaciones finales incluyen la limitación para que la derivada del ángulo de giro se encuentre entre [-1,1]. Esto aumenta el confort del pasajero cuando se monta en el vehículo. Era más difícil implementarlo en Simulink en el controlador IKIBI, y no se ha incluido.

Hasta aquí queda explicado el funcionamiento de un MPC clásico. En el Capítulo 7 hay un estudio más profundo de este controlador, que explora nuevas posibilidades para afrontar las pérdidas de datos.

4.1.2 Métodos de resolución del problema de minimización de coste.

En el artículo [6], el controlador MPC estudiado resuelve el problema de optimización convexo óptimo con CVXGEN [15].

Existe otra herramienta, CVX, con la misma finalidad de automatizar la formulación y resolución de problemas de optimización convexa. Sin embargo, CVX está adaptado para MATLAB y CVXGEN genera códigos para usar en otras plataformas, como códigos en c.

Además, CVXGEN crea algoritmos más rápidos y orientados a aplicaciones en tiempo real, mientras que CVX es más lento, pero exacto.

Es una herramienta potente, capaz de resolver distintos tipos de problemas de optimización incluidos los de los controladores MPC, que se puede formular como problemas cuadráticos (quadratic problems, QPs).

Su principal inconveniente, comparado con los otros solvers de MATLAB, es que se trata de un sistema de modelado aparte. Aunque es compatible con MATLAB, hay que descargar el paquete aparte y tiene licencia aparte.

Por otra parte, la función de MATLAB que crea un controlador MPC (mpc()), viene con tres métodos integrados para la resolución de QP: *active-set*, *interior-point* y el *método de multiplicadores de dirección alterna* (alternated direction method of multipliers, ADMM) [16].

Active-set: Proporciona buenos resultados con problemas de optimización a pequeña y mediana escala. Usa un algoritmo KWIK de ramificación y poda.

Interior-point: Aporta mejores resultados a mayor escala. También lo hace al trabajar con restricciones y horizontes de predicción y control grandes. Incorpora un algoritmo primal – dual, que se enfoca en resolver el problema primal y dual en vez de solo uno de ellos. Todos los problemas de optimización se componen de dos problemas: el de minimización y el que marcan las restricciones. Este método utiliza información de ambos problemas, creando una zona factible que dé solución a ambos y optimizando dentro de ella.

Se han comparado los dos primeros métodos, y el segundo, *interior-point*, es ligeramente mejor. El caso empleado en la comparación ha sido el caso base completo de la Figura 2. No se ha introducido ruido ya que afectaría a los índices negativamente.

4.2 Controlador IKIBI

Está adaptado a la dinámica del vehículo. Es usado en el mismo contexto en [6].

Se expresa con la fórmula mostrada abajo:

$$\delta(k+1) = \left[atan_2\left(\frac{r_{ref}L_v}{v_x(k)}\right) + K_p(r_{ref}(k+1) - r(k))\right]\gamma \qquad (36)$$

K_p: Ganancia proporcional del controlador por prealimentación. Al ajustar el controlador, K_p = 0.55.

r_{ref}: Variación de velocidad de cambio del ángulo de giro deseada establecida por el algoritmo Pure Pursuit.

L_v: Longitud del vehículo. Sumatorio de l_r y l_f.

γ: Coeficiente del vehículo que traduce el ángulo del neumático en el ángulo de dirección. En este caso, $\gamma = 1$ ya que la señal de entrada se considera directamente como el ángulo del neumático.

Al igual que un controlador proporcional tradicional, la ganancia multiplica al error respecto a la referencia.

La acción de control resultante es el ángulo de dirección. En realidad, la acción de control es un vector compuesto por el ángulo de dirección y la aceleración en el eje horizontal, y $a_x = 0$.

El signo positivo o negativo del ángulo depende del cuadrante en el que se encuentre, por eso se incluye la función tangente inversa de cuatro cuadrantes, cuya notación es $atan_2$. Devuelve la tangente con el signo correspondiente al cuadrante en el que se encuentra para que tenga un rango comprendido entre [-π ,π].

Un controlador con prealimentación se anticipa y elimina efectos indeseados antes de que produzcan error en el seguimiento de la referencia.

Capítulo 5. Filtro de Kalman.

Para recabar datos de los componentes del vehículo necesarios para su control y monitorización, se usan sensores. Hay ocasiones en la que los sensores son lentos o hay datos que se pierden o perturbaciones externas (ruido) que alteran el valor real de la medida. También hay situaciones en las que es difícil acceder a ciertas medidas, como, en este caso, V_y y r.

Se han simulado estas situaciones y solucionado los problemas que conllevan usando un filtro de Kalman.

Es un algoritmo de estimación óptimo con el que se pretende conseguir estados estimados lo más parecidos posibles a los estados reales. El filtro tendrá dos funciones: reducir el ruido y estimar.

El aspecto del espacio de simulación de Simulink con el filtro incluido es el siguiente:

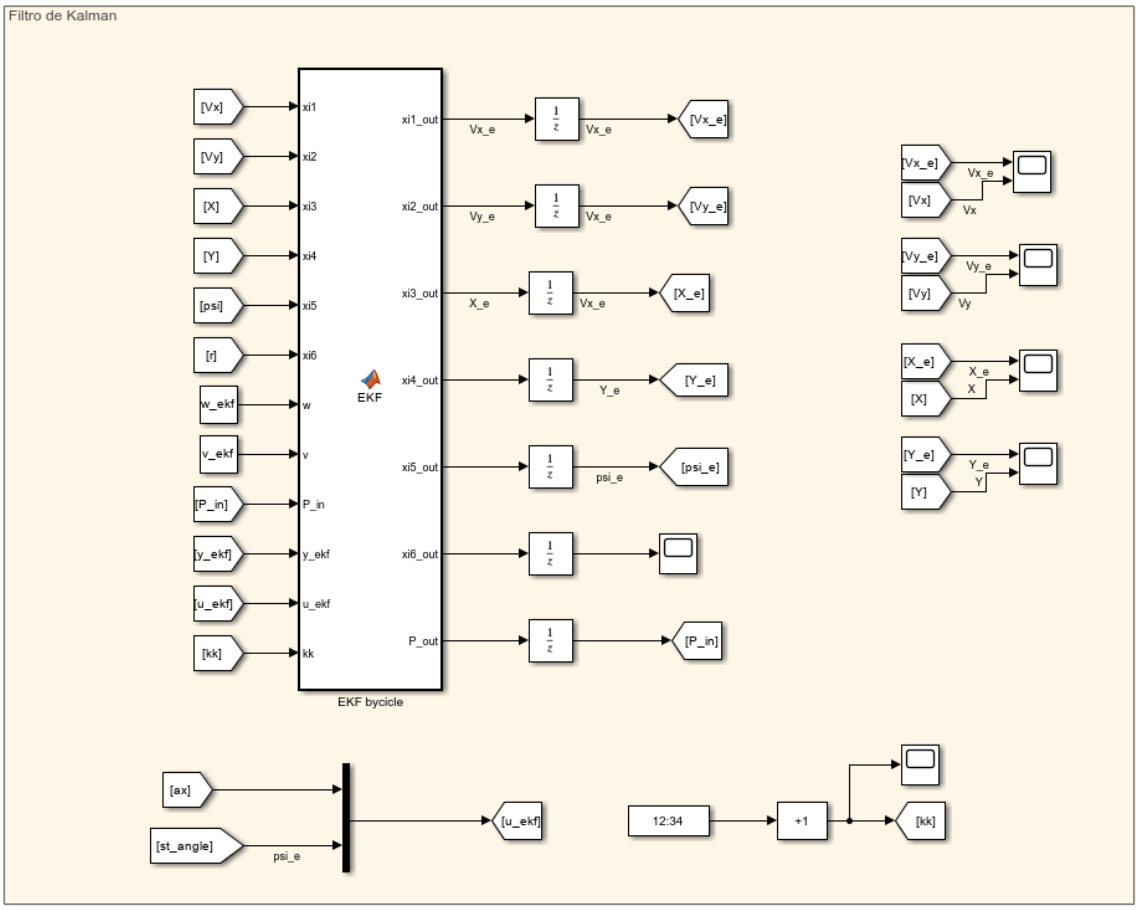

Figura 14. Filtro de Kalman en Simulink.

Se ha optado por emplear un bloque del tipo *MATLAB Function*. De esta forma, se combinan los entornos de Simulink y MATLAB, pudiendo expresar operaciones de forma textual, pero con las entradas y salidas de Simulink.

Si se hiciese directamente sobre un script de MATLAB, habría que crear un bucle con la misma finalidad. Además, en Simulink se recogen automáticamente todos los datos de la

simulación y se pueden representar fácilmente en función del tiempo. En MATLAB tienes que crear la variable donde guardarlas manualmente y luego asociarlas a un vector de tiempo creado también fuera del bucle y actualizado con cada iteración.

El modelo no lineal de Rajamani queda expresado en espacio de estados al igual que en [6] de la forma:

$$\begin{cases} \xi(k) = f\big(\xi(k-1), n_1(k-1), u(k-1)\big) \\ z(k) = h(\xi(k), n_2(k)) \end{cases} \quad (37)$$

Estado del vehículo:

$$\xi(k) = \big(V_x(k), V_y(k), X(k), Y(k), \psi(k), r(k)\big)^T \quad (38)$$

Acción de control:

$$u(k-1) = (a_x(k-1), \delta(k-1))^T \quad (39)$$

Salidas medidas:

$$z(k) = \big(V_x(k), X(k), Y(k), \psi(k)\big)^T \quad (40)$$

$n_1(k-1)$: ruidos del proceso. Multivariable de media 0 y varianza $\bar{Q} = 0.01$.

n_2: ruidos de las medidas. Multivariable de media 0 y varianza $\bar{R} = 0.01$.

El filtro de Kalman primero calcula la predicción con la acción de control y los estados actuales. Después calcula las matrices jacobianas para linealizar el modelo sobre el estado actual y el ruido del proceso.

Son necesarias para obtener la propagación de la covarianza.

$$\hat{\xi}(k|k-1) = f\big(\xi(k-1|k-1), n_1(k-1), u(k-1)\big) \quad (41)$$

$$P(k|k-1) = A(k)P(k-1|k-1)A(k)^T + L(k)\bar{Q}(k-1)L(k)^T \quad (42)$$

Siendo $\xi(0) = E[\xi(0)]$ y $P(0) = E[(\xi(0) - E[\xi(0)])(\xi(0) - E[\xi(0)])^T$. E [·] es la esperanza. Se considerará que la notación $\hat{\xi}(j|i)$ representa la estimación del estado en el instante jT realizada en el instante iT.

El filtro de Kalman es óptimo porque busca minimizar P, la covarianza del error de estado estacionario.

Las matrices jacobianas quedan de la siguiente forma:

$$\bar{A}(k) = \frac{\partial f}{\partial \xi}\Big|_{\hat{\xi}(k-1|k-1), n_1(k-1), u(k-1)} \quad (43)$$

$$\bar{L}(k) = \frac{\partial f}{\partial n_1}\Big|_{\hat{\xi}(k-1|k-1), n_1(k-1), u(k-1)} \quad (44)$$

Cuando las medidas no están disponibles para un instante determinado, se asume el valor anterior de $\hat{\xi}(k|k)$ y $P(k|k)$.

$$\hat{\xi}(k|k) = \hat{\xi}(k|k-1) \quad (45)$$

$$P(k|k) = P(k|k-1) \quad (46)$$

Otro escenario posible era el de que el tiempo de toma de medidas de los sensores sea inferior al tiempo de muestreo del modelo. Sería el caso multifrecuencia.

La predicción de la salida es:

$$\hat{z}(k) = h(\hat{\xi}(k|k-1), n_2(k)) \quad (47)$$

El filtro de Kalman cuenta con una ganancia K(k) que pondera la confianza en la estimación del estado predicho frente a la nueva medición observada.

$$K(k) = P(k|k-1)H(k)^T(H(k)P(k|k-1)H(k)^T + M(k)\bar{R}(k)M(k)^T)^{-1} \quad (48)$$

Ahora las jacobianas necesarias serán las de linealización del modelo de salida y del próximo estado, pero sobre el ruido de las medidas.

$$H(k) = \left.\frac{\partial f}{\partial \xi}\right|_{\hat{\xi}(k-1|k-1), n_2(k-1)} \quad (49)$$

$$M(k) = \left.\frac{\partial f}{\partial n_2}\right|_{\hat{\xi}(k-1|k-1), n_2(k-1)} \quad (50)$$

El nuevo estado y matriz de covarianza estimados para el instante actual en el presente quedan así.

$$\hat{\xi}(k|k) = \hat{\xi}(k|k-1) + K(k)(z(k) - \hat{z}(k)) \quad (51)$$

$$P(k|k) = K(k)\bar{R}(k)K(k)^T + \big(I - K(k)H(k)\big)P(k|k-1)(I - K(k)H(k))^T \quad (52)$$

En un EKF no se usan las fórmulas (45) y (46). Y siempre (para todo k), se realizan las correcciones del estado y la covarianza de las fórmulas (51) y (52).

Sin embargo, en un DREKF:

Para k = MT → (47) a (52).

Para k ≠ MT → (45) y (46).

5.1 Análisis de la presencia de ruidos.

Los ruidos tanto del proceso (w_{ekf}) como de las medidas (v_{ekf}) son generados atendiendo a las siguientes fórmulas.

$$w_{ekf} = W_\sigma \cdot N(0,1) + \mu \quad (53)$$

$$v_{ekf} = V_\sigma \cdot N(0,1) + \mu \quad (54)$$

Son gaussianos y de media 0 ($\mu = 0$).

Se ha observado el efecto del incremento del ruido con el fin de escoger valores para las comparaciones finales del trabajo y de estudiar la robustez de los controladores. Como el propósito del trabajo es someter el AGV a situaciones con pérdidas de información, el filtro de Kalman utilizado para la prueba ha sido el DREKF.

Después de probar ambos controladores con el mismo ruido, se aprecia que el controlador IKIBI se desvía y no puede seguir la ruta. Las gráficas a continuación son con el controlador MPC.

Con W_σ = 0.1 y V_σ = 0.1:

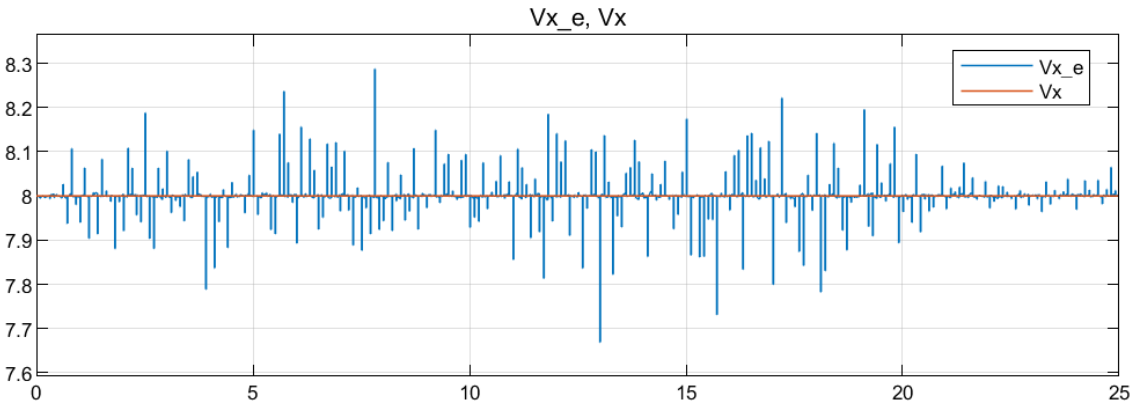

Figura 15. Vx estimada con W_σ y V_σ = 0.1

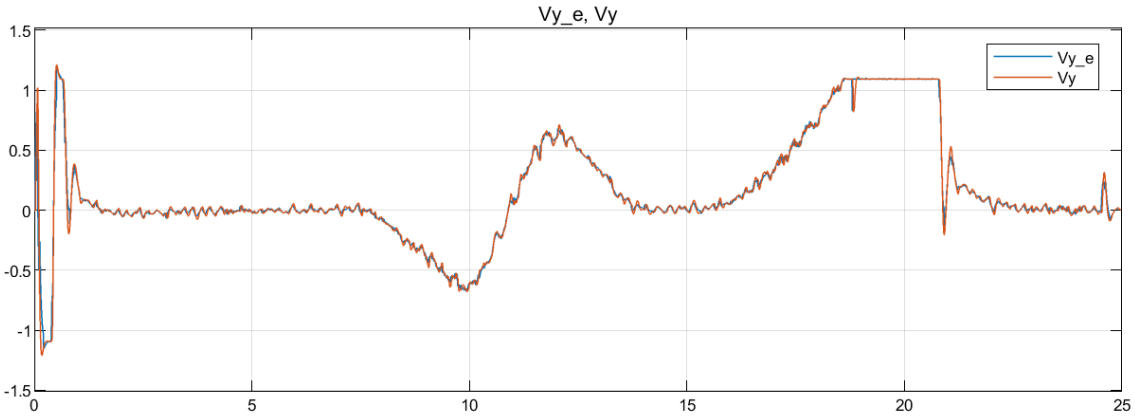

Figura 16. Vy estimada con W_σ y V_σ = 0.1.

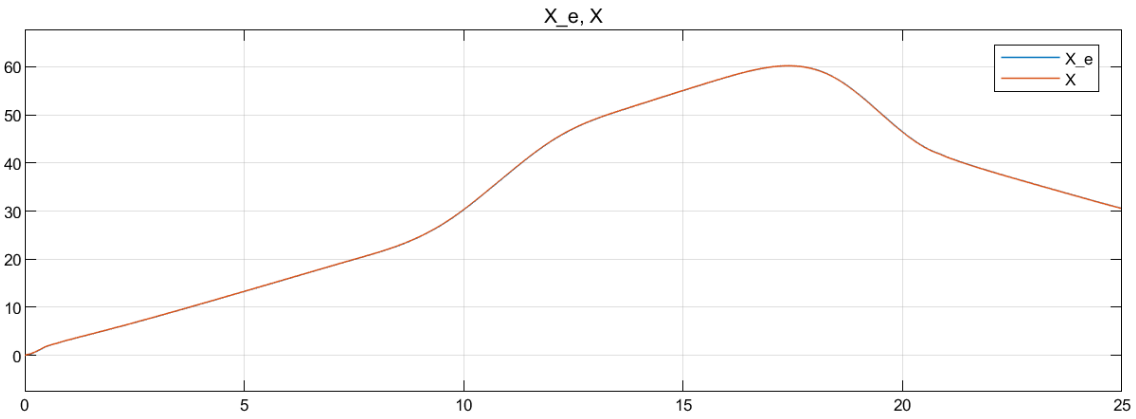

Figura 17. X estimada con W_σ y V_σ = 0.1.

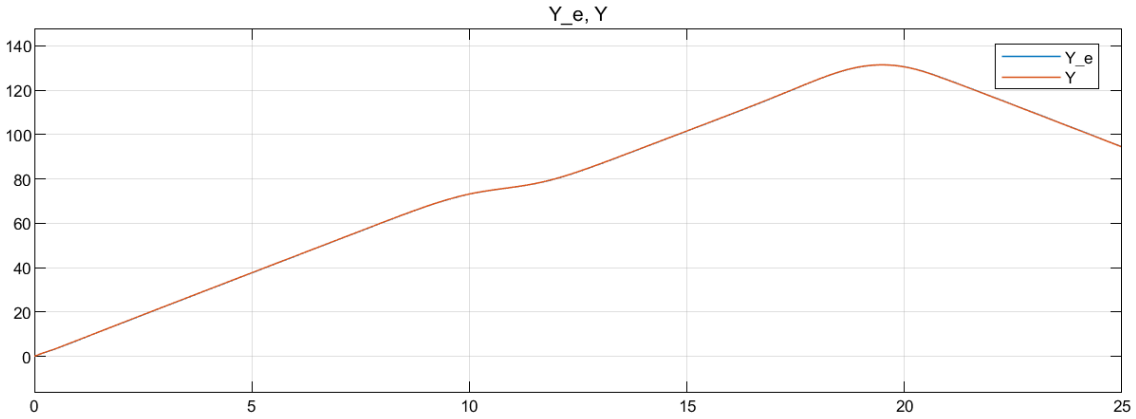

Figura 18. Y estimada con W_σ y V_σ = 0.1.

Con W_σ = 10 y V_σ = 10:

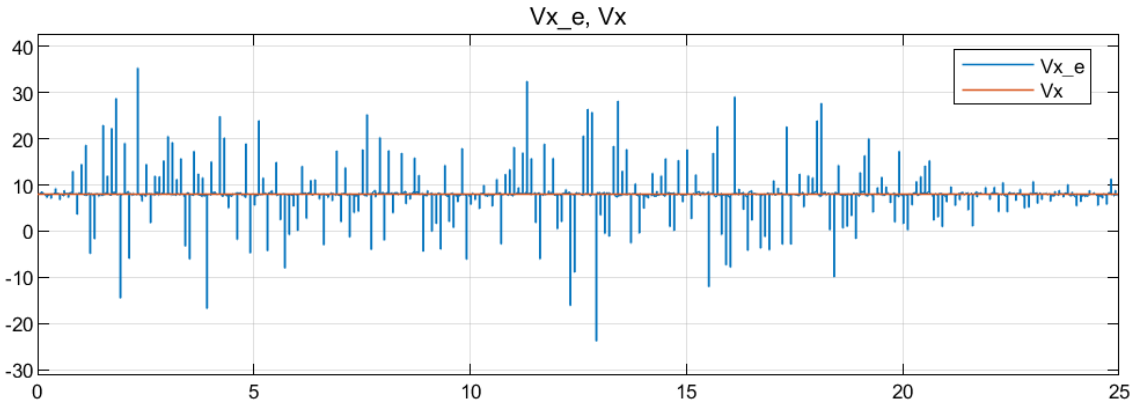

Figura 19. Vx del modelo no lineal y tras pasar por el filtro de Kalman con W_σ y V_σ = 10.

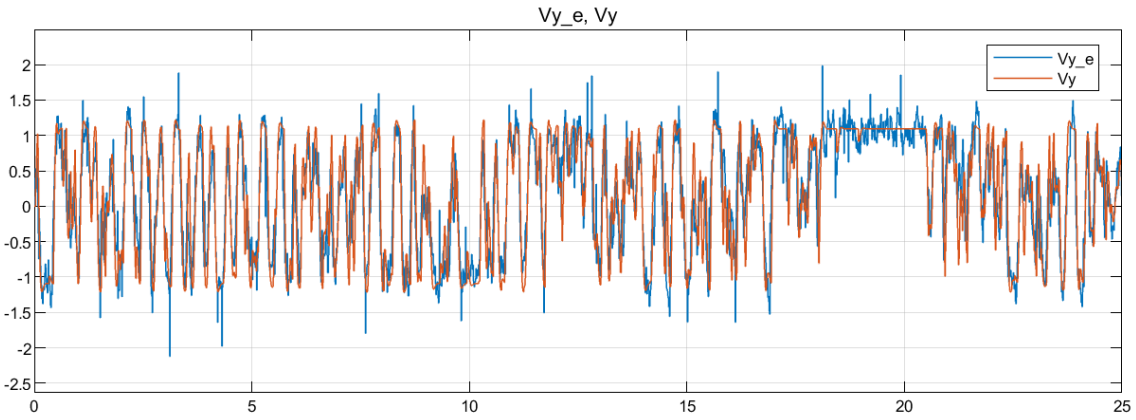

Figura 20. Vy del modelo no lineal y tras pasar por el filtro de Kalman con W_σ y V_σ = 10.

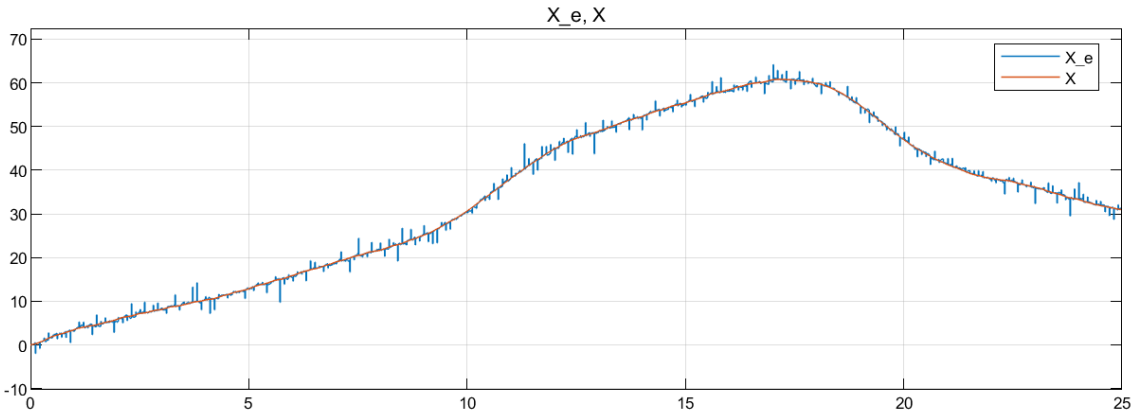

Figura 21. X estimada con W_σ y V_σ = 10.

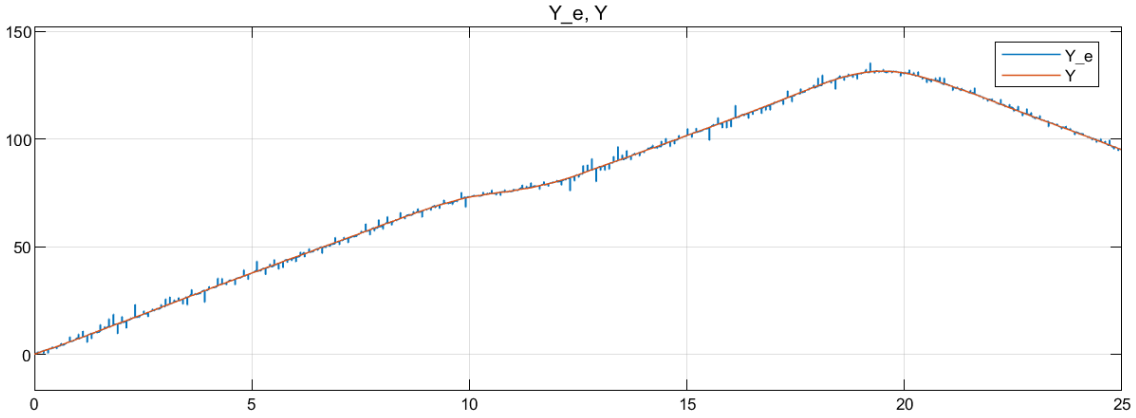

Figura 22. Y estimada con W_σ y V_σ = 10.

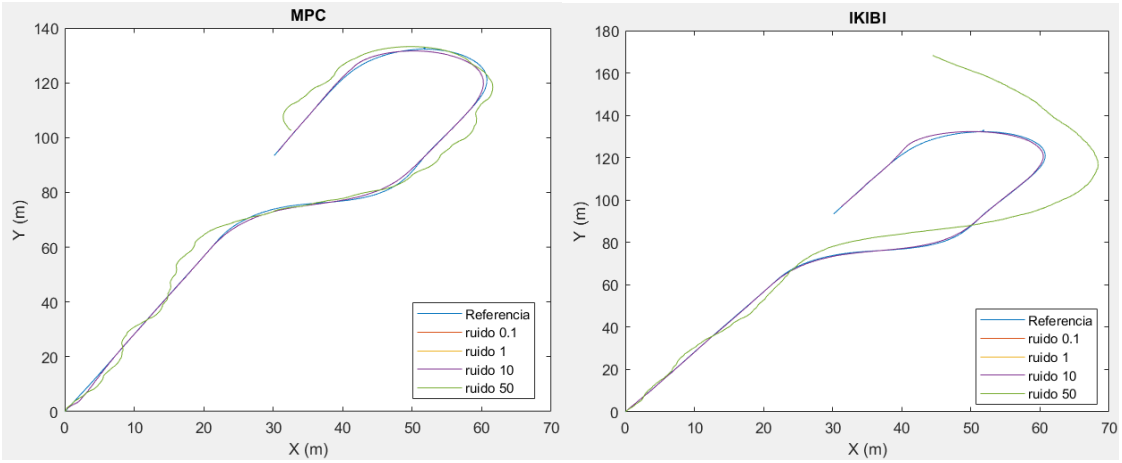

Figura 23. Comparación del movimiento del vehículo con distintos ruidos.

W_σ y V_σ	Índices					
	MPC			IKIBI		
	J1	J1 normalizado	J2	J1	J1 normalizado	J2
0.1	566’8942	0’2698	1’1798	472’2728	0’2248	1’2742
10	1.792’3539	0’8530	4’7460	2.199’9671	1’0471	3’7881

Tabla 6. Índices para el bucle de control con distintos ruidos.

Como es de esperar, al aumentar el ruido, aumentan los índices. El controlador MPC soporta mejor los ruidos. En casos particulares el controlador IKIBI tiene mejor rendimiento, pero el MPC es mejor adaptándose.

En conclusión, las decisiones que se han tomado para la implementación del bucle de control respecto a los modelos han sido que el modelo para simular la dinámica del vehículo va a ser el de Rajamani. Sin embargo, el modelo empleado para la estimación en el filtro de Kalman será el de Stanford, que está comprobado también que no da fallos.

Capítulo 6. Pure Pursuit.

El último elemento añadido en Simulink es el algoritmo de navegación Pure Pursuit, cuyo funcionamiento se explica en detalle en [17].

Calcula la velocidad que debe alcanzar el vehículo para dirigirse a un punto situado delante suya. El punto viene condicionado por la distancia *look-ahead*. El primer punto de la referencia que se encuentre a esa distancia del vehículo es el punto que debe seguir. Genera arcos entre ambos puntos, lo que provoca que el movimiento resultante sea suave [13].

Si la distancia es demasiado pequeña, el movimiento es oscilatorio y el AGV puede desviarse por completo. Si, por el contrario, es demasiado grande, el movimiento será menos preciso. Es importante distinguir, sobre todo en los capítulos finales de la memoria, si los errores en el seguimiento de la trayectoria se deben a la configuración del algoritmo o no.

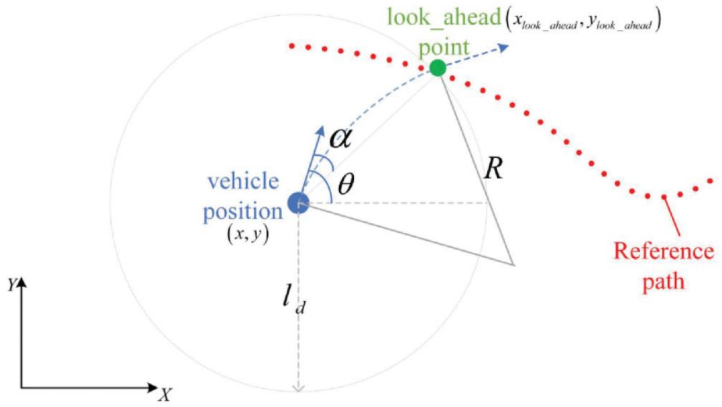

Figura 24. Explicación del algoritmo de navegación Pure Pursuit.

En Simulink tiene este aspecto:

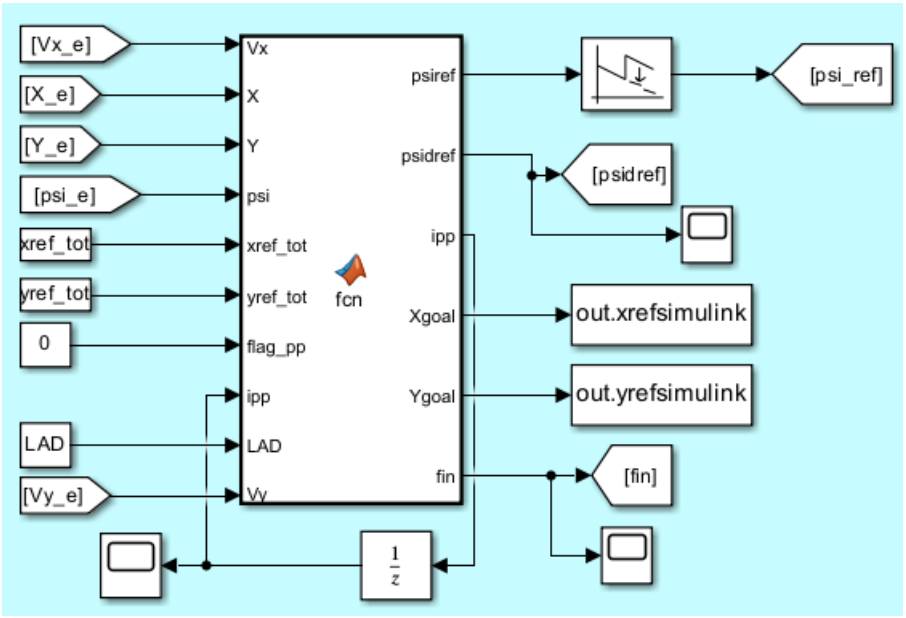

Figura 25. Implementación del algoritmo Pure Pursuit en Simulink.

V_{x_e}, V_{y_e}, $X_{\_e}$, $Y_{\_e}$, $psi_{\_e}$ son los parámetros estimados del filtro de Kalman. *[xref_tot , yref_tot]* es la ruta total de referencia. *Flag_pp* es un indicador que cambia cuando se ha elegido un nuevo punto de referencia, *ipp* es otro índice que recorre los puntos de la referencia y se incrementa con cada iteración, *psi_ref k* y *psid_ref* son la referencia y velocidad del ángulo de guiñada creado respectivamente y *fin* es la condición de parada del AGV (que el vehículo se encuentre a menos de 1 metro del final de la ruta).

Primero se calcula la distancia entre la posición actual del AGV y cada uno de los puntos de la trayectoria.

La distancia en el eje X:

$$dist_x = |X - xref\_tot(ipp)| \quad (55)$$

La distancia en el eje Y:

$$dist_y = |Y - yref\_tot(ipp)| \quad (56)$$

Y la distancia total:

$$dist_{tot} = \sqrt{{dist_x}^2 + {dist_y}^2} \quad (57)$$

Si la distancia es mayor que la del "look-ahead" (LAD):

$$X_{refn} = xref\_tot(ipp) \quad (58)$$

$$Y_{refn} = yref\_tot(ipp) \quad (59)$$

Además, aumenta el índice ipp.

Una vez elegido el punto, se calcula la distancia entre el objetivo y el ángulo. Con ella se obtienen el ángulo y derivada del ángulo de guiñada.

$$dd = \sqrt{(Y_{goal} - Y)^2 + (X_{goal} - X)^2} \quad (60)$$

$$\alpha = atan2\left(Y_{goal} - Y, X_{goal} - X\right) - \psi \quad (61)$$

$$\psi_{dref} = \frac{2V_x \sin(\alpha)}{dd} \quad (62)$$

$$\psi_{ref} = atan2\left(Y_{goal} - Y, X_{goal} - X\right) \quad (63)$$

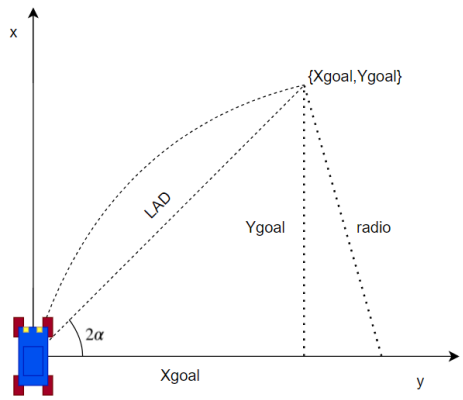

Figura 26. Esquema visual para las ecuaciones del Pure Pursuit.

Las fórmulas (55) - (63) describen el método formulado igual que en [5], en un contexto muy parecido. Sin embargo, como ya se ha mencionado con anterioridad, es eficaz en otros problemas de control distintos con AGVs involucrados [18].

Un último ajuste requerido en la trayectoria generada por el Pure Pursuit es arreglar la continuidad cuando el ángulo se sale del rango $[\pi, -\pi]$.

La solución es conectar el ángulo con el bloque *Unwrap*. Arregla desfases con ángulos.

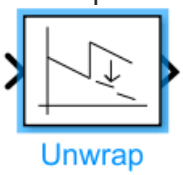

Figura 27. Bloque Unwrap.

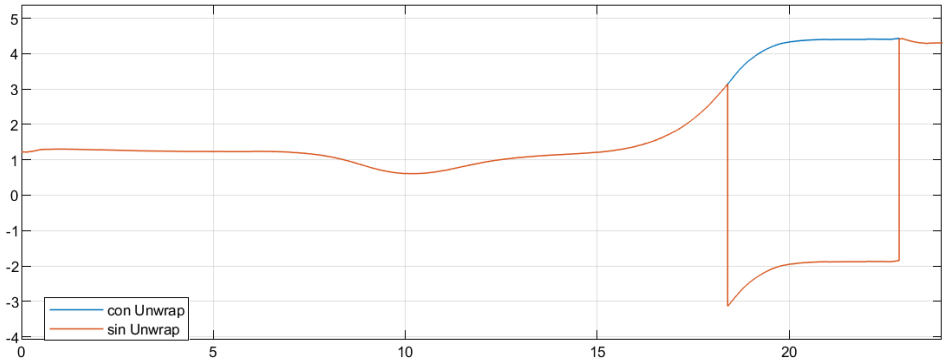

Figura 28. Ángulo de guiñada calculado con Pure Pursuit con y sin el bloque Unwrap.

Con el propósito de analizar el efecto de la distancia LAD, se ha utilizado el bucle de control mostrado en la Figura 11 y planteado en la Figura 2.

La comparación se ha realizado con el modelo Rajamani para el caso base, el controlador IKIBI y el filtro de Kalman con W_σ y V_σ = 1 para el ruido.

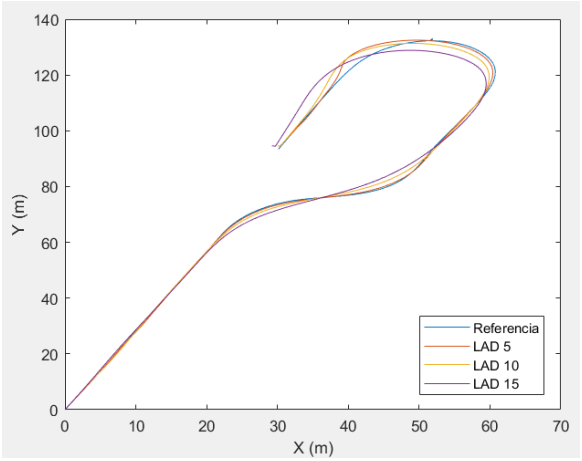

Figura 29. Bucle de control para LAD 5,10 y 15 m.

LAD (m)	IKIBI		
	J1	J1norm	J2
5	879’6443	0’2931	2’4865
10	1.724’5951	0’5746	4’9765
15	3.659’1351	1’2193	7’5192

Tabla 7. Índices para LAD 5, 10 y 15 m.

La referencia tiene un giro muy pronunciado aproximadamente en la posición [51.7 , 132.5] m. Un LAD muy pequeño proporciona un mejor seguimiento de la referencia. Normalmente es un aspecto positivo, aunque con referencias que incluyen cambios bruscos no es buena idea, ya que puede ser demasiado restrictivo y el vehículo puede descontrolarse.

Si el LAD es muy grande, las curvas pronunciadas no las sigue con precisión.

Aunque en este trabajo simplemente se ha probado con varios LADs y se ha escogido el que asegura el mejor seguimiento de la trayectoria, se trata de un parámetro que se puede optimizar, como se propone en [5,18].

El algoritmo no estabiliza el AGV al acabar la ruta de referencia. Una forma de evitarlo es imponiendo una condición de parada al llegar al último punto deseado.

La condición de parada deja un margen de 1 metro, por si el AGV no pasa por el punto exacto del final de la ruta. Cuando se cumple, se fuerza al coche que pare siguiendo el razonamiento descrito a continuación.

Las variables que pueden ser manipuladas para influir en el comportamiento del vehículo, son el ángulo de dirección y la aceleración . Recordemos que $u = [a_x, \delta]$.

Según las ecuaciones del modelo no lineal, el vehículo se detendrá si sus velocidades horizontal y vertical se anulan.

La aceleración a_x en el caso base es nula. V_x depende del valor que uno le asigne inicialmente y cambiará según a_x. La única forma de que V_x tome un valor determinado es estableciéndolo previamente.

El caso de la velocidad vertical es algo más complejo. No se modifica directamente y hay que recurrir al ángulo de dirección. Cuando se activa la condición de parada, el ángulo de dirección pasa de ser el calculado por el controlador y se ajusta para ser el que garantice que Vy sea 0.

El método es una solución ad hoc que se diseñó expresamente para este problema. Así, no hay que estar pendiente de establecer el tiempo final de simulación según si se establece la velocidad es 8 m/s, en cuyo caso la simulación debe durar aproximadamente 25 segundos para finalizar la ruta, y 12 m/s, cuando la simulación debe durar aproximadamente 17 segundos.

Posteriormente a la creación del método, se descubrió un bloque de Simulink que finaliza la simulación en cuanto recibe una señal. Se llama *Stop Simulation*. Para usarlo la condición de parada es la misma y el resto del procedimiento es automático.

Figura 30. Bloque Stop Simulation de Simulink.

El método manual se usó con las simulaciones de la Figura 30. Las del Capítulo 8 son con el método automático porque el primer método provoca pequeñas oscilaciones que dificultan la lectura de las gráficas.

Capítulo 7. Análisis de la escasez de información.

Una vez comparados los controladores como se describe en el Capítulo 4 y añadidos el filtro de Kalman (Capítulo 5) y el generador de referencias (Capítulo 6), queda completo el caso base y se puede incluir la falta de datos.

Las razones más comunes por las que puede haber poca información son las pérdidas, medidas lentas de los sensores, y PETC. Esto último, como ya se ha mencionado antes, consiste en evaluar una condición antes de activar la comunicación entre la parte de control y la de la planta. De esta manera, se dispone de menos información, pero al mismo tiempo se ahorran envíos (mejora el ancho de banda y disminuye el desgaste de las baterías de los dispositivos al usarse menos).

La localización de la falta de datos se va a suponer en la comunicación, por un lado, desde los sensores hacia el controlador, y se tomarán medidas con el DREKF, y por otro desde el controlador a los actuadores (pegados a la planta, llevan a cabo las instrucciones del controlador, convirtiendo las señales de control en movimientos o acciones físicas dentro del proceso industrial).

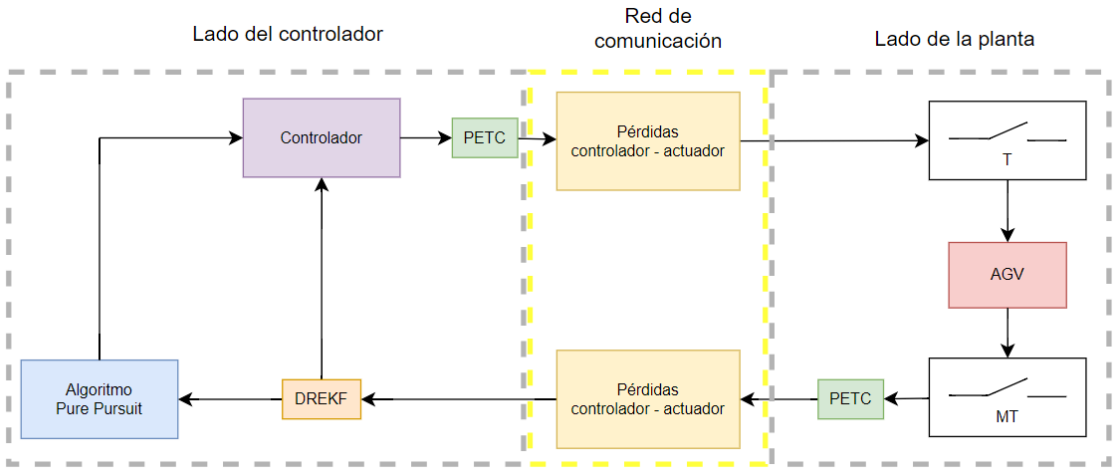

Figura 31. Esquema del caso base con pérdidas.

En la Figura 31 se presenta el mismo caso de [5] aunque solo con pérdidas, sin retrasos en el envío de la información.

En total se estudiarán 4 casos desde el punto de vista tanto del controlador y de la planta. La probabilidad de pérdida será la misma en ambos, ya que se supone un único canal.

7.1 Casos.

Caso 1. Escasez de datos con x probabilidad

Hay cierta probabilidad de pérdida.

Al tratarse de una red de comunicación, lo más realista es suponer que los datos se organizan en paquetes, de forma que se transmiten todos los datos del paquete o no lo hace ninguno porque este es una unidad que se pierde por completo.

Caso 2. Bifrecuencia. (T = 0.01s y MT = 0.1s; Mp = 10)

Se puede ver como un caso en el que el filtro recibe solo 1 muestra, y se pierden Mp-1 consecutivas entre medias.

Solo se producen envíos exitosos por parte del MPC cada Mp instantes.

Caso 3. Bifrecuencia no uniforme.

Algunas de las pocas muestras que llegan en el caso bifrecuencia también se pierden.

Caso 4. Escasez cada dato con x probabilidad

Hay cierta probabilidad de pérdida en cada uno de los parámetros de forma individual (Vx,Vy,X,Y,psi,r). Es un caso menos realista y más sofisticado.

Solo se contempla desde el punto de vista de la vía de comunicación sensor-controlador.

Este caso tiene sentido suponiendo que la pérdida no venga de la red, sino de alguno de los sensores en particular (por ejemplo, debido a averías).

Esto no ocurre con el controlador. El conjunto de acciones calculadas en el horizonte de predicción del MPC se enviará en un paquete, que llegará o no, pero no es viable que llegue la mitad del paquete. Entonces, no tiene sentido el caso 4 para el enlace controlador, actuador, por lo que solo se aplicaría al otro enlace.

7.1.1 Manejo de pérdidas controlador-actuador.

Caso 1.

Los MPC cuando predicen el futuro calculan muchas acciones de control óptimas que acaban descartadas. La relevancia de las acciones de control obtenidas se pierde conforme transcurre el tiempo.

Siendo $ui(k+j)$ la acción de control calculada en i para el momento j. N es el horizonte de predicción.

Instante de tiempo	Acciones de control calculadas
k	u(k), u(k+1), u(k+2), u(k+3), ..., u(k + N)
k+1	u_1(k+1), u_1(k+2), u_1(k+3), u_1(k+4)..., u_1(k + N+1)
k+2	u_2(k+2), u_2(k+3), u_2(k+4), u_2(k+5)..., u_2(k+N+2)
k+3	u_3(k+3), u_3(k+4), u_3(k+5), u_3(k+6)..., u_3(k+N+3)

Figura 32. Manejo de pérdidas controlador-actuador en el caso 1.

Supongamos que en k+2 el sensor falla y el MPC no dispone de suficiente información para hacer los cálculos. Las acciones u(k+2) y u1(k+2) serían las mejores opciones posibles, ya que fueron estimadas para el mismo instante de tiempo. Sin embargo, la

situación en k no es la misma que en k+1. El error se acumula, ya que estamos tratando con predicciones.

Decantarse por u1(k+2) es lo mejor.

El nuevo enfoque que se propone para el controlador MPC en la memoria consiste en aprovechar las predicciones no solo para la elección de la mejor acción de control, sino para solucionar problemas de falta de información.

Ahora bien, puede haber pérdidas consecutivas. Es decir, si en k+1 se pierde el paquete, entonces se aplica u(k+1). Si en k+2 vuelve a haber pérdida, todavía no hemos podido refrescar el paquete de k+1. Entonces u1(k+2) no está disponible, por lo que aplicaríamos u(k+2).

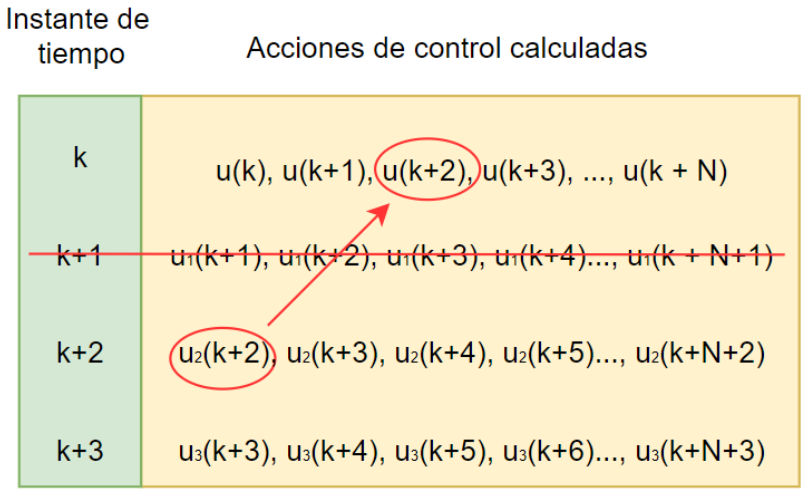

Figura 33. Manejo de pérdidas controlador-actuador en el caso 1(2)

Trabajar con matrices que cambian de tamaño puede dar problemas. En Simulink sobre todo porque la implementación de código requiere bloques que a su vez están diseñados para entradas y salidas de parámetros de tamaño conocido.

Es posible sortear este obstáculo, pero como de todas formas cuanta menos memoria se ocupe mejor, y como se ha dicho antes, las acciones de control dejan de ser eficaces después de varias iteraciones, se ha optado por un buffer circular de tamaño (N, N+1).

Concretamente, el código actúa de forma que, si no hay pérdida, se guardan todas las acciones de control calculadas en ese momento para todo el horizonte de predicción en una fila.

La pérdida se representa en la tabla con una fila de ceros. Se hace así para facilitar la visualización de la cantidad de pérdidas. En todos los casos estudiados como el horizonte de predicción es lo suficientemente grande (N = 20), no perjudica que queden espacios de la tabla con valores nulos.

Han sido detectadas varias excepciones al comportamiento normal del manejador que se han tenido que programar individualmente.

Excepción 1: En el instante inicial hay una pérdida. La tabla está vacía, por lo que no se aplica ninguna acción de control.

Excepción 2: Hay tantas pérdidas que la tabla no tiene ningún valor distinto de 0. La solución es la misma que con la excepción 1.

Excepción 3: La cantidad de pérdidas es mayor que N. La acción de control aplicada es la que se encuentra en la fila escogida según el procedimiento normal, para el último instante disponible, k + 20.

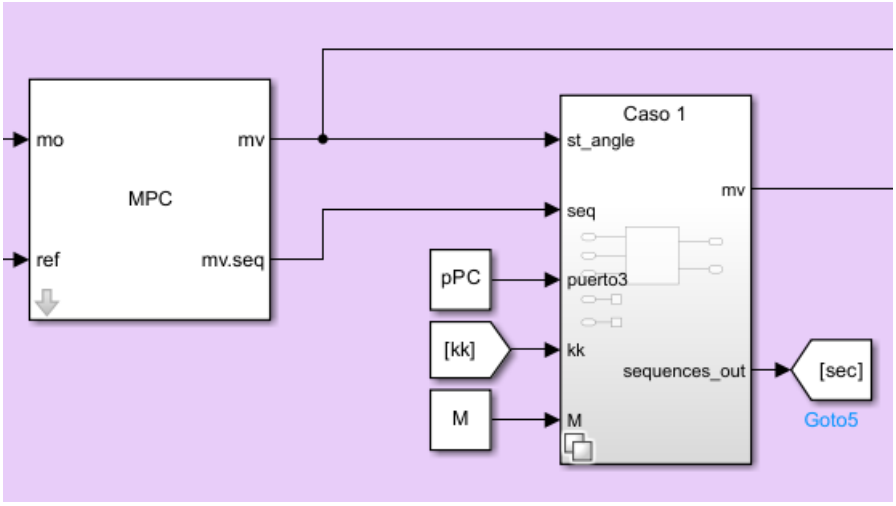

Figura 34. Manejo de pérdidas controlador-actuador para MPC en Simulink.

El controlador IKIBI guarda la primera acción de control que es enviada con éxito y la actualiza si no hay pérdida. Por tanto, la acción de control aplicada siempre es la almacenada, que puede ser actual o de algún instante anterior.

Al igual que con el controlador MPC, cuando se inicia la simulación, no se aplica ninguna acción de control hasta que la primera es enviada correctamente.

A esto lo denominaremos método 1. Otra opción, que es más sofisticada, incluye una parte de predicción. Consiste, cuando hay pérdida, en aplicar el ángulo de dirección calculado a partir de estados predichos para ese instante con las mismas fórmulas del filtro de Kalman pero sin ruido incluido. Se repite la operación tantas veces como haga falta hasta el siguiente instante Mp.

La predicción se realiza dentro del bloque manejador con los estados estimados del filtro de Kalman y la acción de control estimada.

Los dos métodos proporcionan resultados muy similares en el caso 1. La diferencia se puede apreciar mejor en el caso 2.

Caso 2.

El índice kk, que se incrementa cada T = 0.01 segundos, se usa en los casos 2,3 y 4. Mp indica un envío exitoso cada Mp transmisiones. Mp = 10, por ejemplo, implica que de cada 10 envíos, uno consigue que los datos lleguen correctamente y 9 veces seguidas no se producen.

En este caso, la idea para el controlador es la misma, con la diferencia que automáticamente se guardan todas las acciones de control calculadas en caso de que el envío de datos sea exitoso, y se van aplicando una a una en los siguientes instantes.

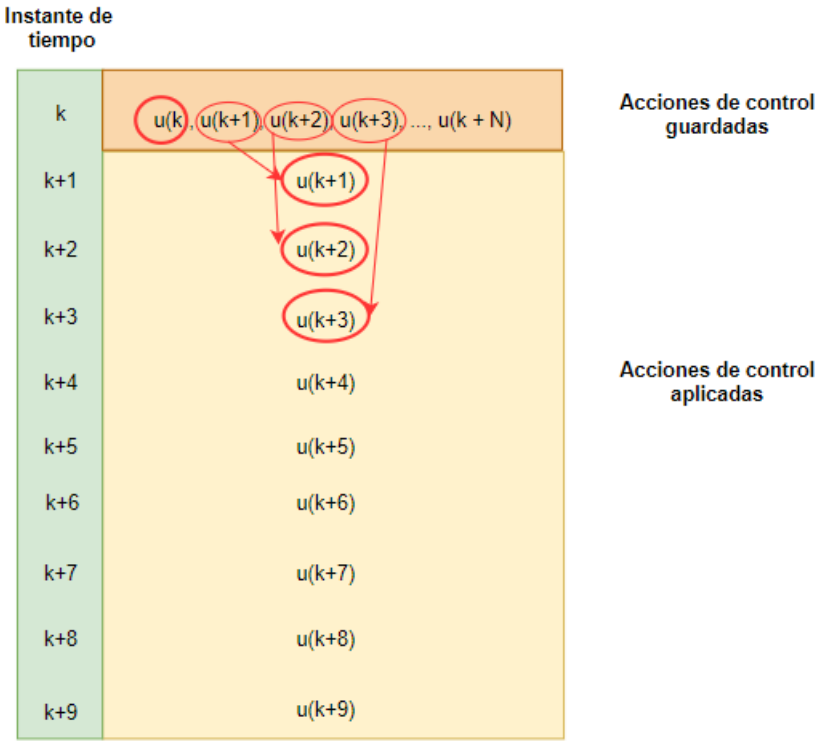

Figura 35. Manejo de pérdidas controlador – actuador en el caso 2.

El controlador IKIBI es más sencillo que el MPC, ya que está compuesto de una ecuación, en comparación con todos los pasos de predicción del MPC. La programación resulta más sencilla por el hecho de que las pérdidas son controladas y no aleatorias.

De forma análoga al caso 1, una idea para el controlador IKIBI es guardar la última acción de control que se transmitió correctamente y aplicarla Mp - 1 veces seguidas.

Para el método 2, cada Mp instantes se aplica directamente el ángulo de dirección calculado con los parámetros del AGV que vienen del filtro de Kalman. Después, se aplica la misma función que dentro del filtro para predecir el siguiente estado, y se calcula la acción de control a partir del estado predicho. Se repite la operación tantas veces como haga falta hasta el siguiente instante Mp.

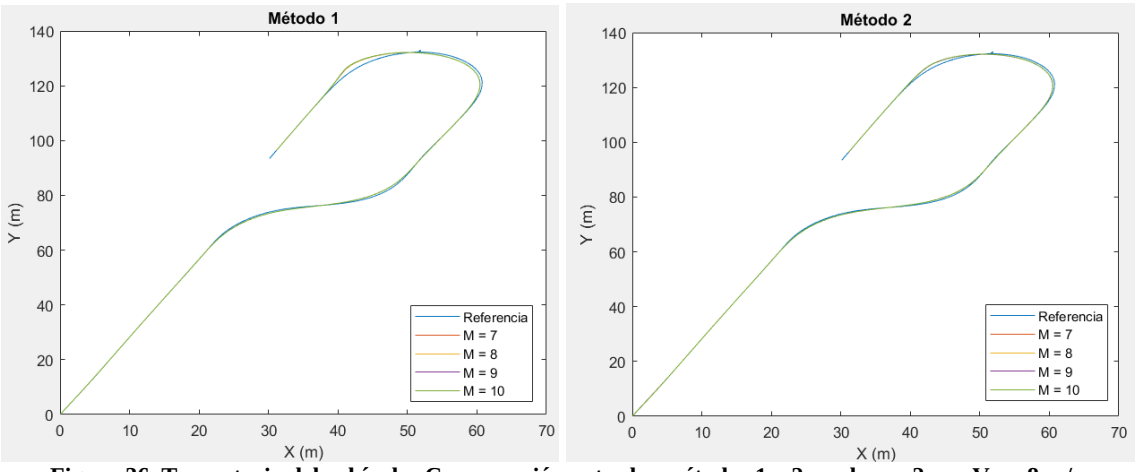

Figura 36. Trayectoria del vehículo: Comparación entre los métodos 1 y 2 en el caso 2 con Vx = 8 m/s.

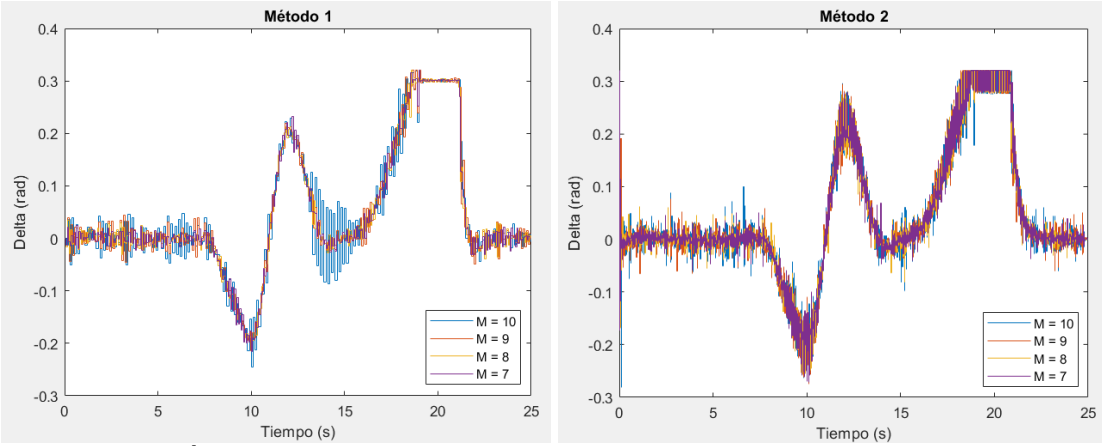

Figura 37. Ángulo de dirección: Comparación entre los métodos 1 y 2 en el caso 2 con Vx = 8 m/s.

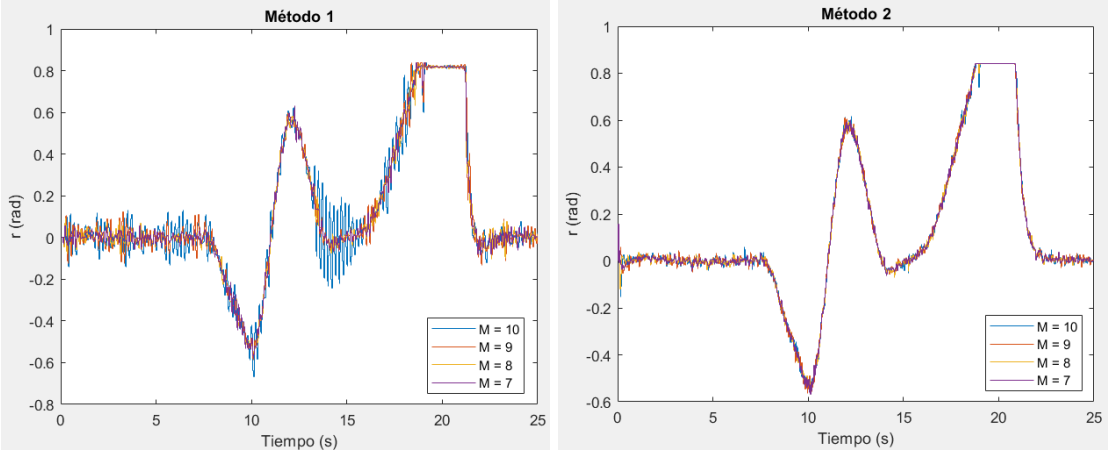

Figura 38. Velocidad de la guiñada: Comparación entre los métodos 1 y 2 en el caso 2 con Vx = 8 m/s.

El segundo método es mejor, aunque ambas opciones son buenas. La diferencia incrementaría con Mp más altos o con periodos de muestreo más grandes. Parece mucho peor que de 10 envíos solo 1 se produzca a decir que el tiempo de muestreo es 0.1 segundos. Hay que tener esto en cuenta para analizar correctamente los resultados finales.

El método 1 presenta ligeras oscilaciones al seguir la trayectoria. Son más notorias cuando la velocidad es de 12 m/s. De todas formas, pueden considerarse despreciables a gran escala.

Las comparaciones del final de la memoria se harán con el método 2. Si en la vida real se desea hacer el experimento y supone mucho más esfuerzo, el método 1 sigue siendo una opción viable.

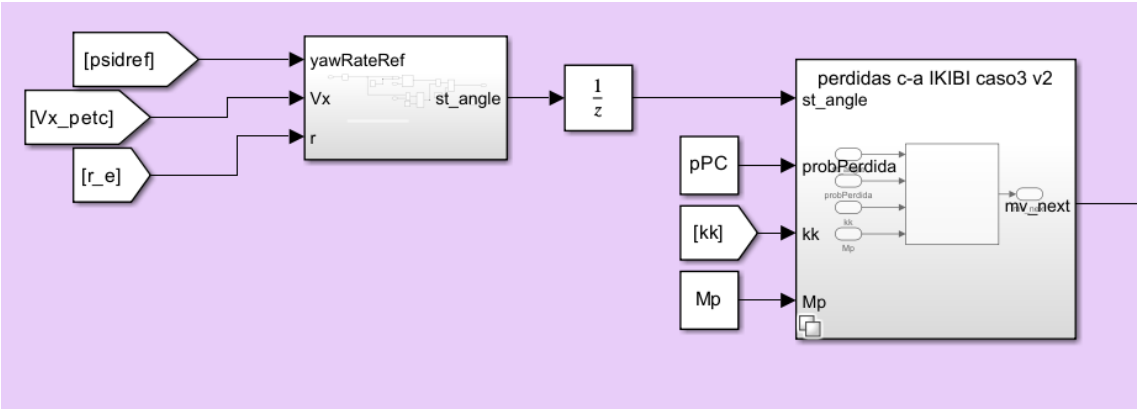

Figura 39. Manejo de pérdidas controlador-actuador para IKIBI en Simulink método 1.

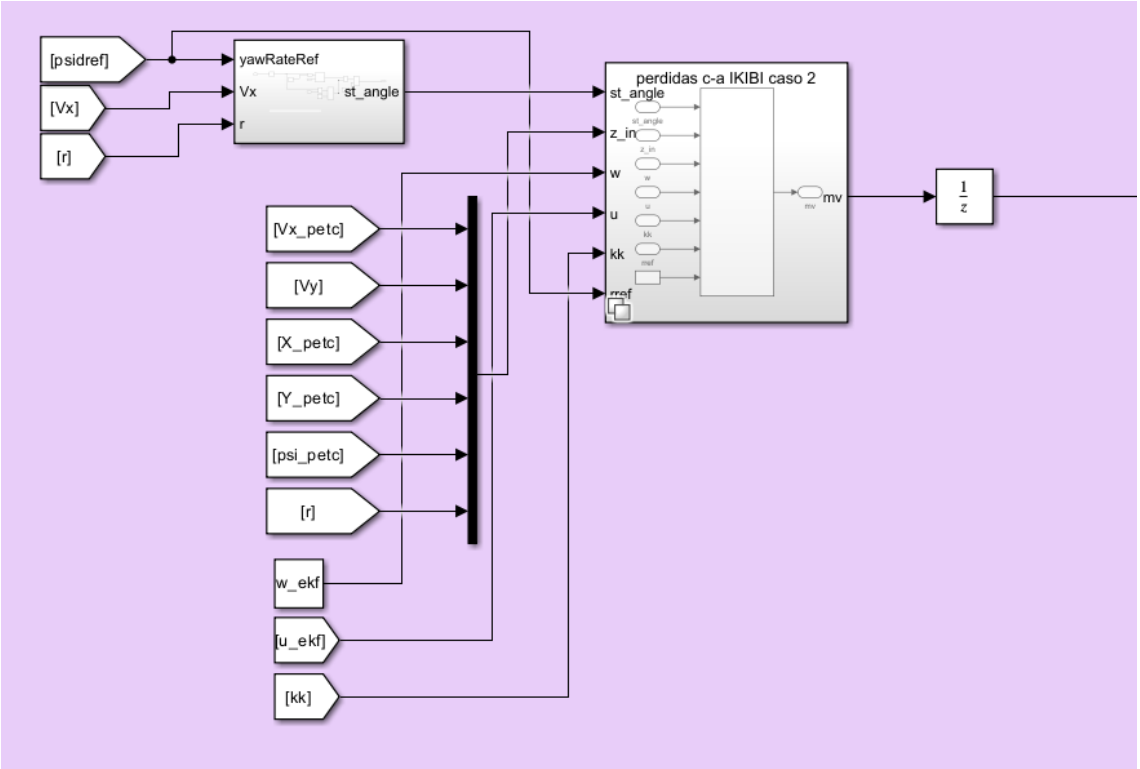

Figura 40. Manejo de pérdidas controlador-actuador para IKIBI en Simulink método 2.

Caso 3.

El caso 3 utiliza el mismo principio para asignar las acciones de control. En caso de pérdida, la secuencia no se guarda y se usa la anterior. Para hacer frente a este caso de mejor manera, esto es, sin repetir acciones calculadas con anterioridad, se debería tener un horizonte de predicción mucho más amplio. Esto supone un aumento del coste computacional. Además, las acciones estimadas con un horizonte tan amplio acumulan más error de predicción.

Instante de tiempo	Acciones de control calculadas
k - (k+9)	u(k), u(k+1), u(k+2), u(k+3), u(k+4), u(k+5), u(k+6), u(k+7), u(k + 8),u(k + 9), u(k + 10), u(k + 11),u(k + 12), u(k + 13), u(k + 14), u(k + 15), u(k + 16), u(k + 17), u(k + 18), u(k + 19), u(k + 20)
(k+10) - (k+19)	u_1(k+10), u_1(k+11), u_1(k+12), u_1(k+13), u_1(k+14), u_1(k+15), u_1(k+16), u_1(k+17), u_1(k + 18),u_1(k + 19), u_1(k + 20), u_1(k + 21),u_1(k + 22), u_1(k + 23), u_1(k + 24), u_1(k + 25), u_1(k + 26), u_1(k + 27), u_1(k + 28), u_1(k + 29), u_1(k + 30)
(k+20) - (k+29)	u_1(k+10), u_1(k+11), u_1(k+12), u_1(k+13), u_1(k+14), u_1(k+15), u_1(k+16), u_1(k+17), u_1(k + 18),u_1(k + 19), u_1(k + 20), u_1(k + 21),u_1(k + 22), u_1(k + 23), u_1(k + 24), u_1(k + 25), u_1(k + 26), u_1(k + 27), u_1(k + 28), u_1(k + 29), u_1(k + 30)
(k+30) - (k+39)	u_1(k+10), u_1(k+11), u_1(k+12), u_1(k+13), u_1(k+14), u_1(k+15), u_1(k+16), u_1(k+17), u_1(k + 18),u_1(k + 19), u_1(k + 20), u_1(k + 21),u_1(k + 22), u_1(k + 23), u_1(k + 24), u_1(k + 25), u_1(k + 26), u_1(k + 27), u_1(k + 28), u_1(k + 29), u_1(k + 30)

Figura 41. Manejo de pérdidas controlador - actuador en el caso 3.

En este ejemplo, al iniciar la simulación no se produce ninguna pérdida, y se guardan las acciones de control del horizonte de predicción. Se aplican al igual que en caso 2 hasta el momento k + Mp – 1. Supongamos que Mp = 10.

Siguiendo con el caso 3, ahora puede producirse una pérdida. No ocurre, y por lo tanto se guarda un nuevo paquete de acciones de control y se vuelven aplicar las 10 primeras.

Entonces, se produce una pérdida. Se aplican las 10 siguientes acciones del mismo paquete, puesto que no se dispone de uno nuevo.

El horizonte de predicción es 20, con lo cual aseguramos tener acciones con pérdidas no consecutivas. Si ocurren de forma consecutiva, se siguen aplicando las mismas que durante la primera pérdida hasta que llegue nueva información.

Lo ideal, como se ha mencionado antes, sería tener un MPC con un horizonte de control alto para hacer frente a todas las pérdidas consecutivas que puedan suceder. Es ineficiente aumentar mucho la cantidad de acciones de control calculadas, si en la mayoría de los casos ni se usarían para las probabilidades de pérdida y Mp establecidos. N = 20 es un buen compromiso para los escenarios estudiados.

El control IKIBI según el método 1, guarda la acción de control cuando llega y la aplica constantemente hasta que puede actualizarse.

El bloque manejador de pérdidas controlador-actuador con el método 2, actualiza y guarda el último paquete que le llega con el estado del vehículo. Entonces, predice los estados correspondientes al momento actual y calcula la acción de control con los estados predichos.

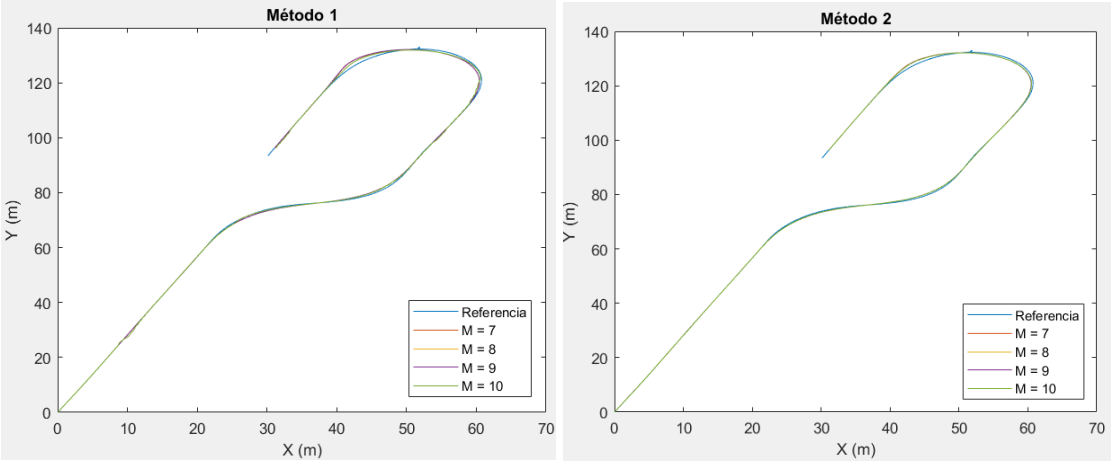

Figura 42. Trayectoria del vehículo: Comparación entre los métodos 1 y 2 en el caso 3 para IKIBI con Vx = 8 m/s.

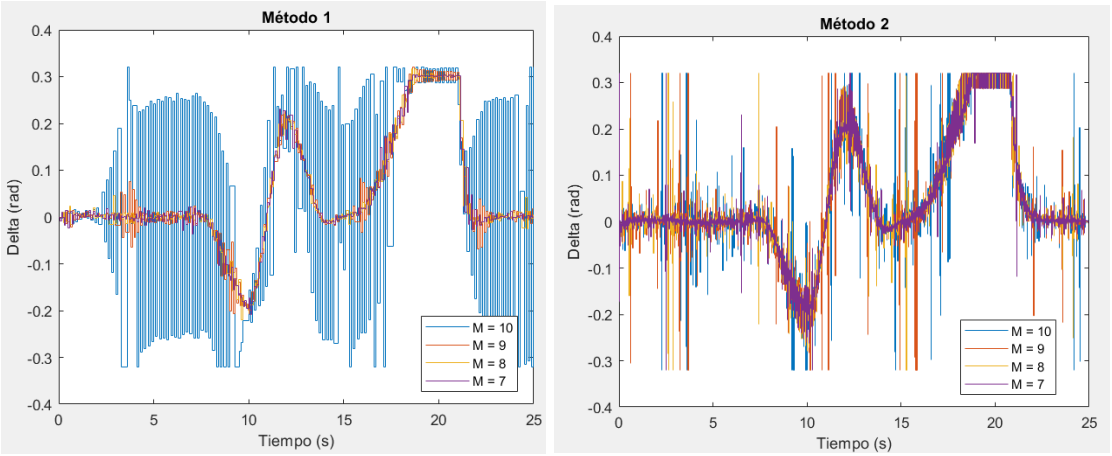

Figura 43. Ángulo de dirección: Comparación entre los métodos 1 y 2 en el caso 3 para IKIBI con Vx = 8 m/s.

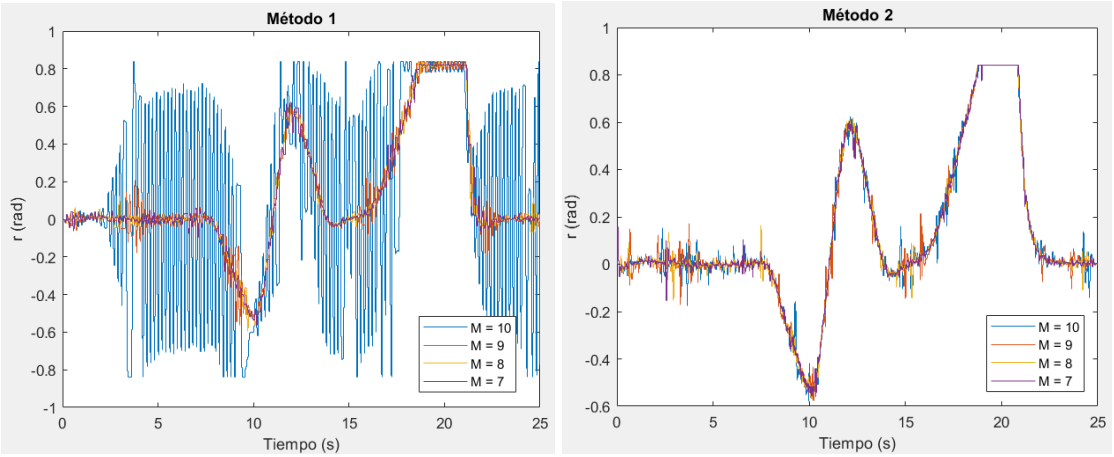

Figura 44. Velocidad angular de la guiñada: Comparación entre los métodos 1 y 2 en el caso 3 para IKIBI con Vx = 8 m/s.

7.1.2 Manejo de pérdidas sensor-controlador.

En los 3 primeros casos, si no hay dato el DREKF predice y si llegan, corrige.

El último caso es distinto. Se guarda un vector de estado en el que los parámetros se actualizan conforme llegan, de manera que parte del vector corresponde al momento actual y otra parte corresponde a instantes anteriores.

El filtro de Kalman solo predice si no llega ningún dato. En el resto de las situaciones corrige con solo parte de los datos actualizados, mientras que los demás son antiguos.

Es verdad que la corrección del filtro se realizará con información errónea, pero es mejor que la alternativa; dar todo el paquete por perdido. El error acumulado no llega a ser muy grande porque el modelo es bueno; no se han incluido muchas perturbaciones externas y el periodo de muestreo es bajo. Este tipo de sistema (AGV) necesita un control bastante rápido, aunque las medidas pueden ser tomadas a periodo más lento. Por eso la multifrecuencia.

7.2 PETC.

Hasta ahora las pérdidas de información han sido infortunios achacados a la red de comunicaciones. Sin embargo, ahora se va a presentar un caso en el que el envío de la información puede omitirse de forma voluntaria: cuando es despreciable el cambio de los parámetros [5].

Los estados y las acciones de control que se van a transmitir se comparan, y si la diferencia de los nuevos valores respecto a los ya obtenidos anteriormente apenas varían (comprobado con un pequeño margen de variación), no se envían los nuevos valores.

Esta condición se evalúa cada T = 0.01 segundos en los sensores y el controlador.

Cada instante en el que se evalúa la condición es $K \in \mathbb{N}_{\geq 1}$.

$$\sum_{i=1}^{s_z} \|\bar{z}_i(K-1) - z_i(K)\|^2 > \sum_{i=1}^{s_z} \sigma_{zi}\|z_i(K)\|^2 + \mu_{zi} \quad (64)$$

s_z = Tamaño del vector de estados del AGV.

$\bar{z}_i(K-1)$: Última medición del sensor transmitida y almacenada en el instante de evaluación anterior.

$z_i(K)$: Medición actual del sensor.

$$\sum_{i=1}^{s_u} \|\bar{u}_i(K-1) - u_i(K)\|^2 > \sum_{i=1}^{s_u} \sigma_{ui}\|u_i(K)\|^2 + \mu_{ui} \quad (65)$$

s_u = Tamaño del vector de acciones de control.

$\bar{u}_i(K-1)$: Última acción de control calculada y almacenada en el instante de evaluación anterior.

$u_i(K)$: Acción de control calculada.

σ_{zi} , $\sigma_{ui} \in [0, 1]$ y μ_{zi}, $\mu_{ui} \in R^+$ son parámetros que ponderan la importancia de cada de cada elemento del vector de mediciones en el criterio de transmisión y ajustan la sensibilidad del mecanismo. Se han tomado los valores óptimos de [5].

σ_u	0.05
μ_u	0.00001
σ_z	[0.01, 0.0015, 0.0015, 0.01]
μ_z	[0.1, 0.1, 0.1, 0.1]

Tabla 8. Parámetros de ponderación PETC.

Al cumplirse la condición, se impone que $\lambda_z(K) = 1$ y $\lambda_u(K) = 1$. Entonces, se activa la comunicación y se envía el paquete actual, si no, se utiliza el que está guardado.

$$\bar{z}(K) = \lambda_z(K)z(K) + (1 - \lambda_z)\bar{z}(K-1) \quad (66)$$

$$\bar{u}(K) = \lambda_u(K)u(K) + (1 - \lambda_u)\bar{u}(K-1) \quad (67)$$

Capítulo 8. Resultados.

En este apartado se extraen conclusiones partiendo del caso base de la Figura 2. El enfoque de este trabajo es la comparación entre los controladores diseñados en entornos con información limitada.

Factores modificables del bloque de Simulink *Vehicle Body 3DOF* relacionados con la aerodinámica, el ambiente [14] y la simulación se excluirán porque podrían estar teniendo efectos no deseados.

La referencia vendrá del algoritmo Pure Pursuit de las ecuaciones (55) - (63) . El modelo empleado será el de Rajamani, por lo explicado en el apartado 3.4. El ruido se ha establecido con W_σ y V_σ = 0.1. No se añade más para tener gráficas claras. Se pusieron a prueba los controladores con ruidos más altos en el apartado 5.1.

Es mejor, para el LAD, escoger un valor pequeño. LAD = 1 es demasiado restrictivo por los giros pronunciados de 90º y 180º de la ruta a seguir, así que se ha optado por LAD = 5 dentro de lo posible. Si el AGV se descontrola, se intenta con LAD = 10.

En cuanto al controlador IKIBI, solo se ha contemplado que tenga los parámetros γ = 1 y Kp = 0’55. Y el controlador MPC tiene horizonte de predicción N = 20 y de control M = 10 o M = 2 en los casos 2 y 3. Primero se diseñó con los mismos parámetros que aparecen en [6], aunque debido al fenómeno observado en la fase de inicialización, el AGV se descontrola.

A continuación, la comparación de los controladores MPC e IKIBI, tanto saturado como sin saturar.

8.1 Caso base sin ruido.

8.1.1 LAD = 5 m.

El mejor caso de todos, con el filtro EKF recibiendo información siempre y corrigiendo continuamente.

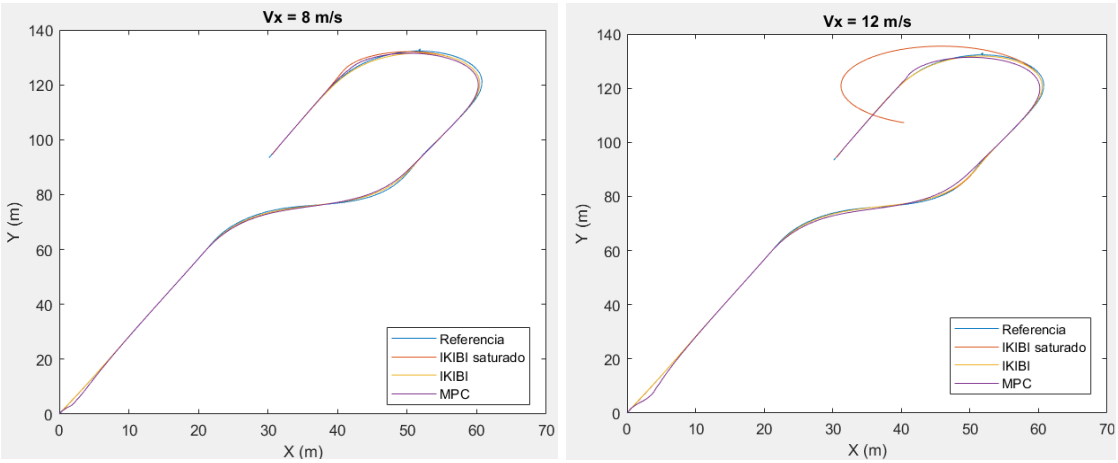

Figura 45. Trayectoria del vehículo: prueba de respuesta rápida del sensor sin ruido y con LAD = 5 m.

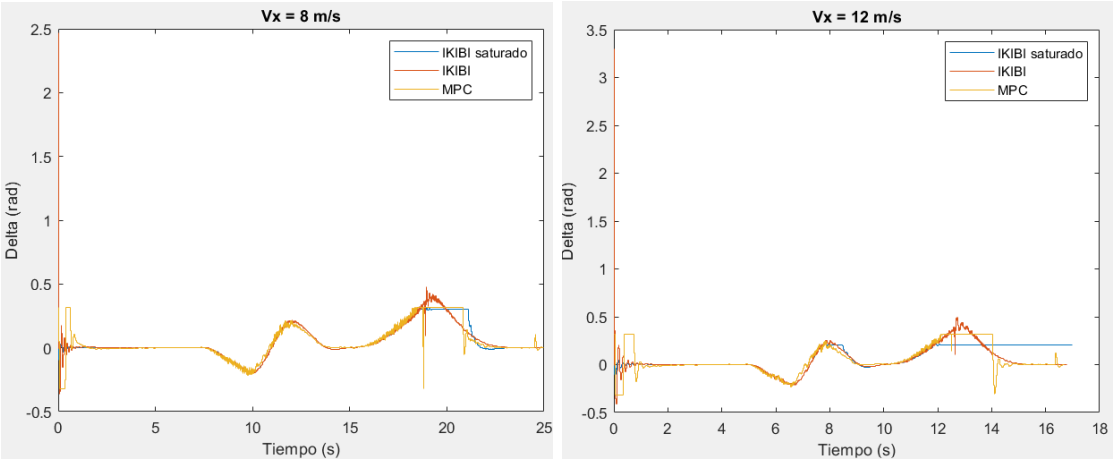

Figura 46. Ángulo de dirección: prueba de respuesta rápida del sensor sin ruido y con LAD = 5 m.

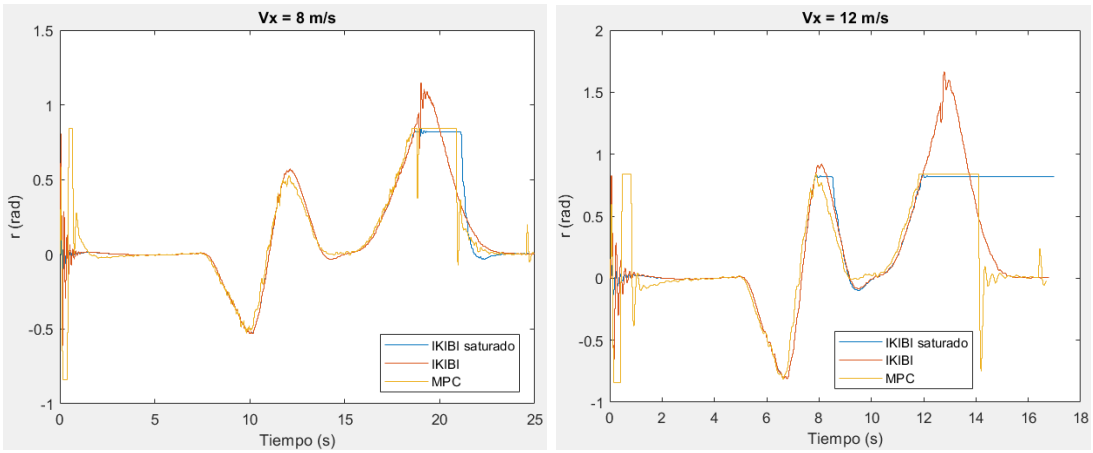

Figura 47. Velocidad angular de la guiñada: prueba de respuesta rápida del sensor sin ruido y con LAD = 5 m.

Cuando el vehículo inicia el recorrido girado, se ha comprobado que la inicialización no es del todo correcta con el controlador MPC. Se han estudiado todos los parámetros de los bloques de espacio de estados y del controlador MPC y no se ha hallado ninguno que compense completamente este efecto. La solución por la que se ha optado es generar los índices con parte de los datos, eliminando los iniciales.

Se han seleccionado los datos a partir de 401 (4 segundos) cuando Vx = 8 m/s y a partir de 201 (2 segundos) si Vx = 12 m/s.

Índices	Vx = 8 m/s			Vx = 12 m/s		
	IKIBI saturado	IKIBI	MPC	IKIBI saturado	IKIBI	MPC
J1	466’9255	380’8477	580’6560	5.363’8493	167’3330	450’4341
J1 normalizado	0’2222	0’1812	0’2764	3’5735	0’1130	0’3066
J2	1’2396	0’8122	1’1586	22’4841	0’5254	1’3260
J4	0’3556	0’7139	0’6536	0’4092	1’1883	0’7151

Tabla 9. Índices caso base sin ruido. Controladores IKIBI saturado, IKIBI y MPC.

El controlador IKIBI sin saturar permite visualizar una situación teórica, ideal. En la vida real, las limitaciones físicas del vehículo no permiten girar tanto las ruedas ni alcanzar una velocidad angular de giro tan elevada.

El controlador IKIBI saturado es una buena opción para la velocidad de 8 m/s, ya que es un tipo de controlador específico creado en base al modelo no lineal de Rajamani.

El índice J4 es menor en el IKIBI saturado ya que la acción de control suele modificarse menos al estar saturada. De hecho, para 12 m/s, aunque el resto de los índices difieren mucho de los valores iniciales, el J4 sigue siendo mejor que con el IKIBI no saturado, ya que no se impone ninguna restricción sobre el ángulo de guiñada.

El MPC es el que mejor se adapta a diferentes velocidades. Cuando la velocidad es 8 m/s, es preferible emplear el controlador IKIBI saturado, pero al salir de esa velocidad, se vuelve inestable y el AGV no consigue seguir la ruta.

8.1.2 LAD = 10 m.

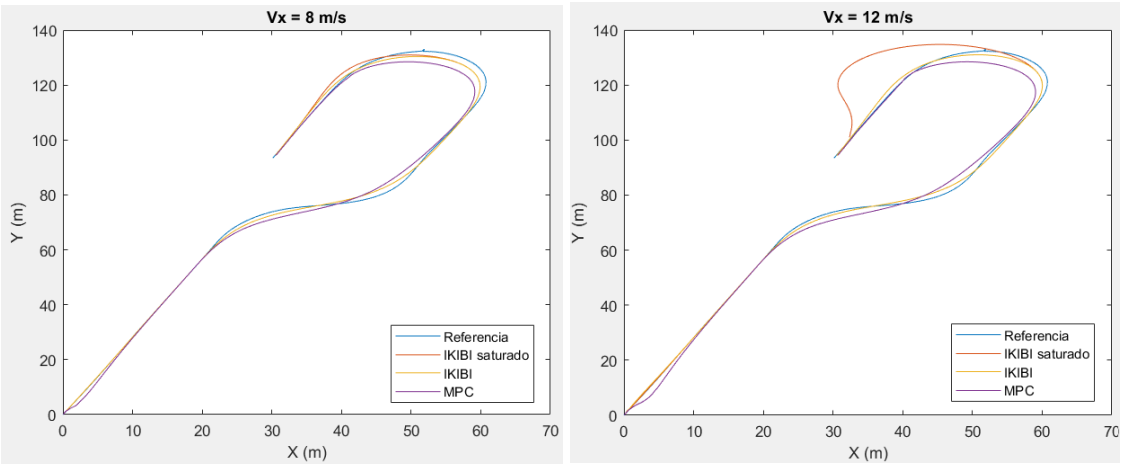

Figura 48. Trayectoria del vehículo: prueba de respuesta rápida del sensor sin ruido y con LAD = 10 m.

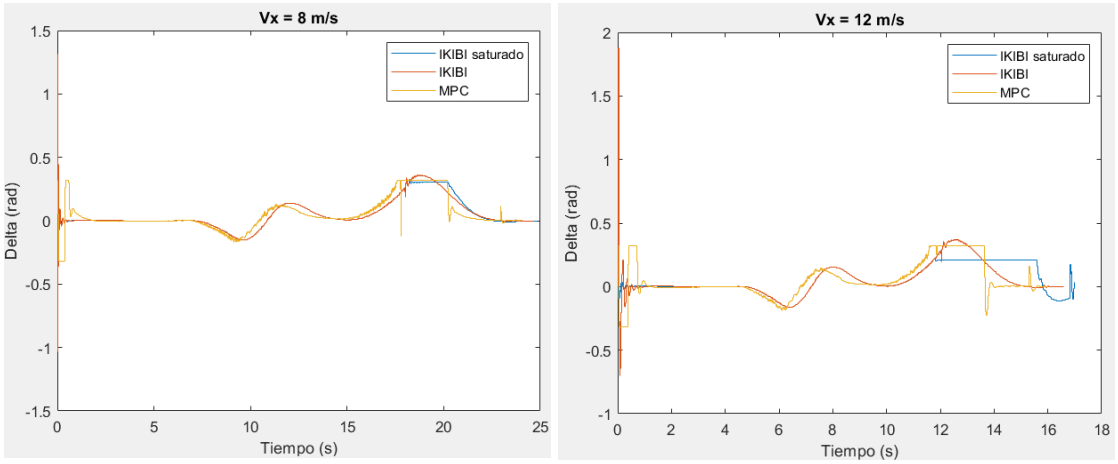

Figura 49. Ángulo de dirección: prueba de respuesta rápida del sensor sin ruido y con LAD = 10 m.

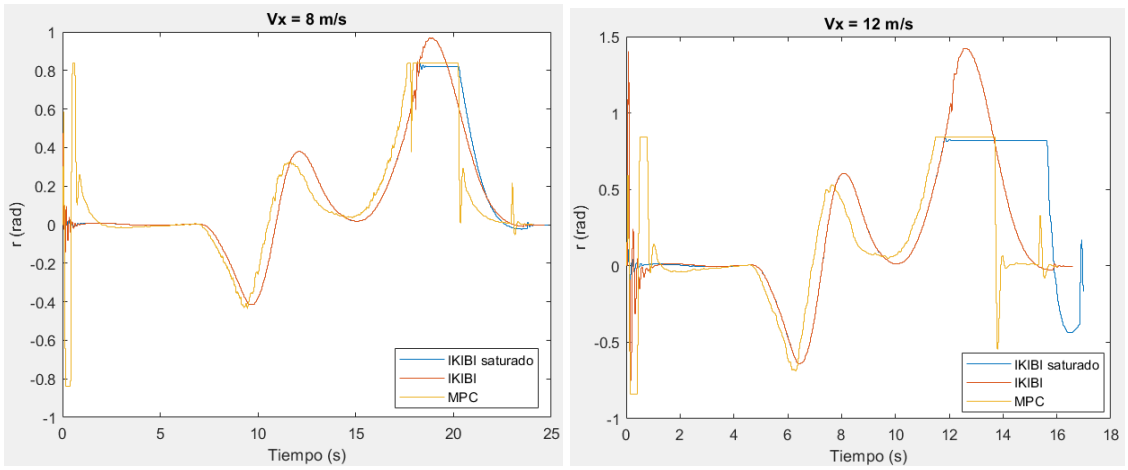

Figura 50. Velocidad angular de la guiñada: prueba de respuesta rápida del sensor sin ruido y con LAD = 10 m.

El controlador IKIBI saturado no se descontrola por completo, aunque se desvía bastante. Ya se ha comprobado en el Capítulo 6 que un LAD grande puede ser contraproducente. Eso es lo que está ocurriendo aquí. El MPC sigue siendo muy potente, pero la referencia que crea el algoritmo Pure Pursuit con LAD = 10 es la que provoca que el giro de 180º se realiza peor.

Por otro lado, aumentar el LAD también tiene un efecto positivo sobre el controlador IKIBI. Al aumentarlo, proporciona más margen de maniobra para que el AGV consiga seguir la referencia.

8.2 Caso base con ruido Wσ, Vσ = 0.1.

La prueba se repite añadiendo un ruido con W_σ, V_σ = 0.1. Ponemos a prueba la robustez de los controladores.

8.2.1 LAD = 5 m.

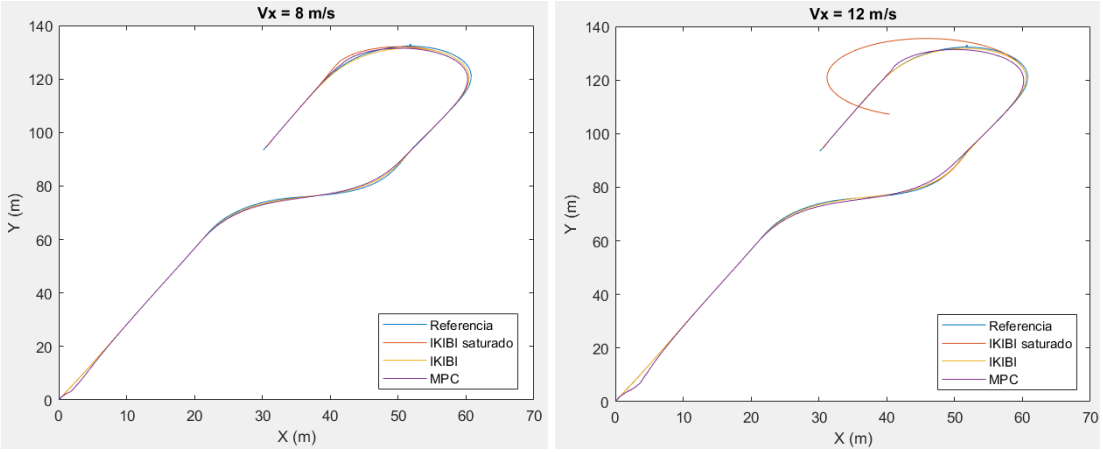

Figura 51. Trayectoria del vehículo: prueba de respuesta rápida del sensor con ruido y con LAD = 5 m.

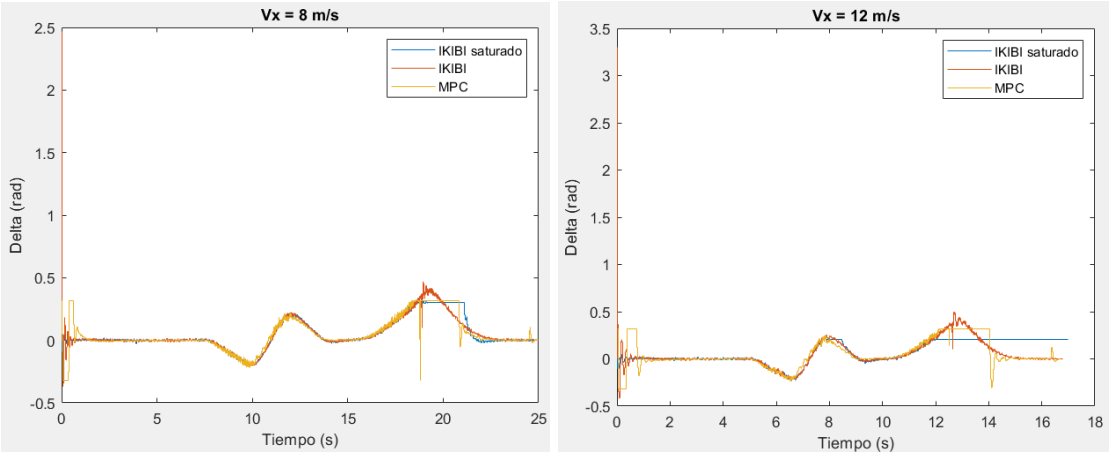

Figura 52. Ángulo de dirección: prueba de respuesta rápida del sensor con ruido y con LAD = 5 m.

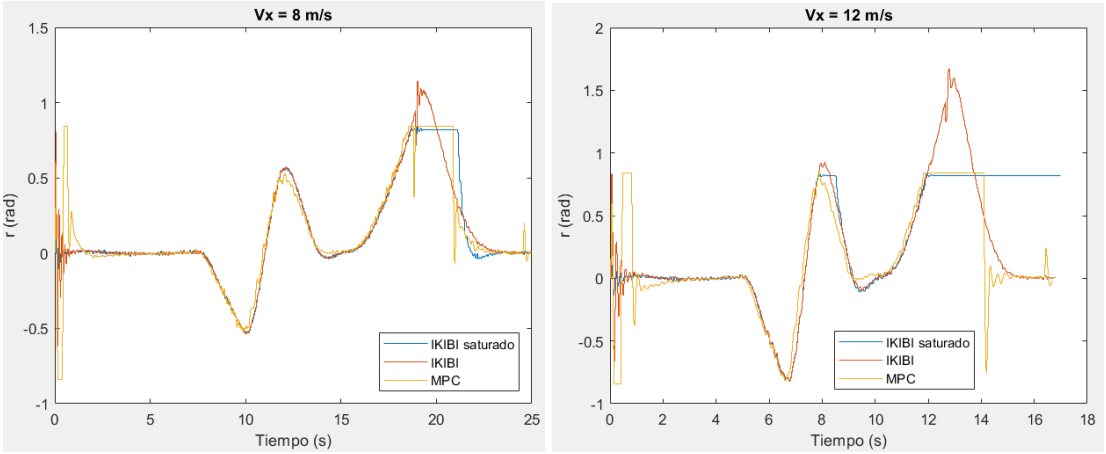

Figura 53. Velocidad angular de la guiñada: prueba de respuesta rápida del sensor con ruido y con LAD = 5 m.

Índices	Vx = 8 m/s			Vx = 12 m/s		
	IKIBI saturado	IKIBI	MPC	IKIBI saturado	IKIBI	MPC
J1	467’3471	467’2785	578’3519	5.364’6492	167’8486	451’6360
J1 norma lizado	0’2224	0’2224	0’2752	3’5740	0’1134	0’3076
J2	1’2406	1’2408	1’1801	22’4894	0’5292	1’3453
J4	0’6658	0’6683	0’5777	0’7063	1’5541	0’6467

Tabla 10. Índices para Vx = 8 y 12 m/s. Controladores IKIBI saturado, IKIBI y MPC con ruido y LAD = 5 m.

Las conclusiones son las mismas ya que los índices son muy parecidos. Con y sin ruido para 8 m/s ambos controladores son buenas opciones pero para 12 m/s el MPC es la mejor opción.

El filtro de Kalman está probado que es muy eficaz reduciendo el ruido. Añadir más provocaría que las gráficas fuesen más difíciles de analizar.

8.2.2 LAD = 10 m.

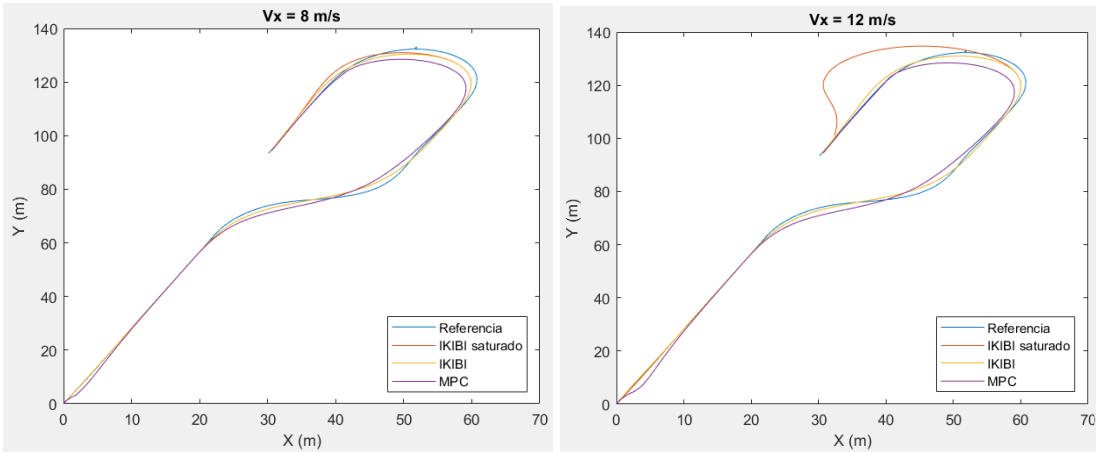

Figura 54. Trayectoria del vehículo: prueba de respuesta rápida del sensor con ruido y con LAD = 10 m.

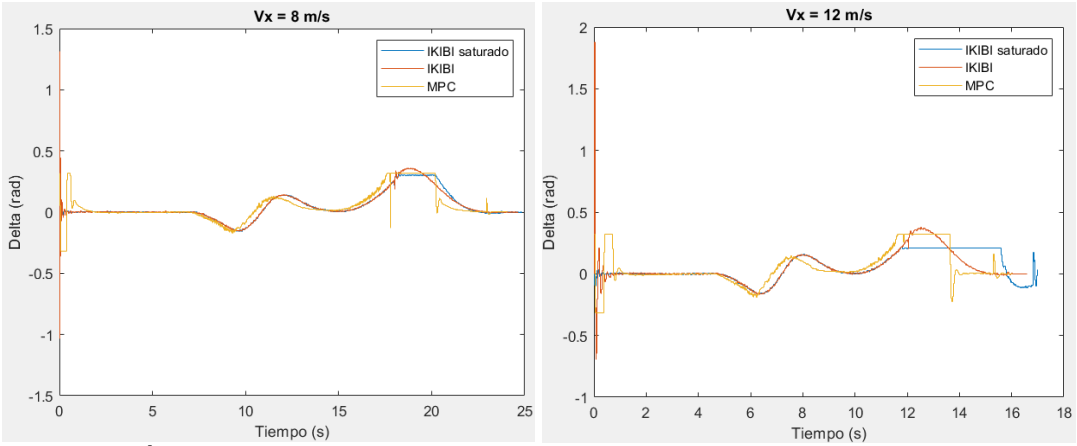

Figura 55. Ángulo de dirección: prueba de respuesta rápida del sensor con ruido y con LAD = 10 m.

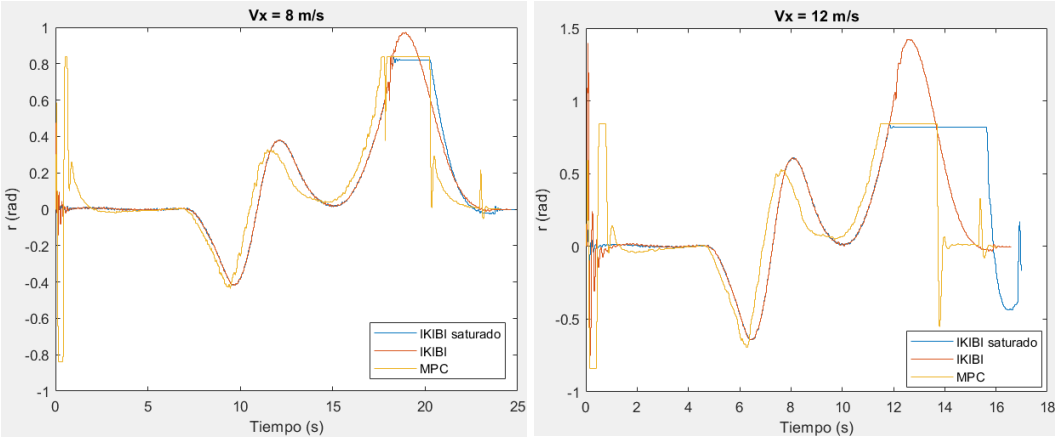

Figura 56. Velocidad angular de la guiñada: prueba de respuesta rápida del sensor con ruido y con LAD = 10 m.

Índices	Vx = 8 m/s			Vx = 12 m/s		
	IKIBI saturado	IKIBI	MPC	IKIBI saturado	IKIBI	MPC
J1	1.104’6040	1.023’2760	1.748’1072	2.673’2227	641’4698	1.263’4873
J1 normalizado	0’5300	0’4948	0’8719	1’7810	0’4397	0’8993
J2	1’8142	1’9789	4’2201	9’2252	1’4783	4’3919
J4	0’3242	0’5861	0’3655	0’4365	1’1314	0’4396

Tabla 11. Índices para Vx = 8 y 12 m/s. Controladores IKIBI saturado, IKIBI y MPC con ruido y LAD = 10 m.

Este caso se estudiará con y sin comunicación PETC, de ahí que se añada la gráfica en este y no en otros apartados similares con LAD = 10.

8.3 Caso 1.

8.3.1 LAD = 5 m y Vx = 8 m/s

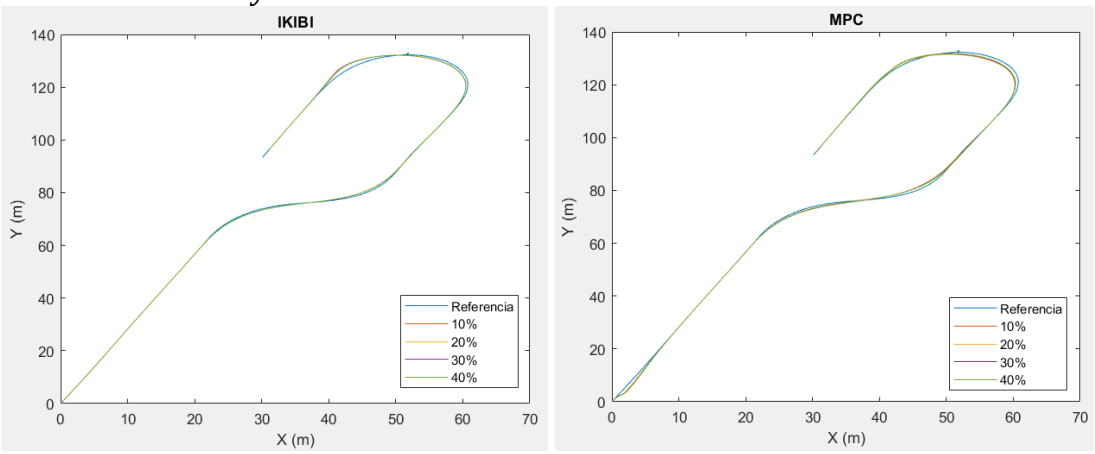

Figura 57. Trayectoria del vehículo: caso 1 con Vx = 8 m/s y LAD = 5 m.

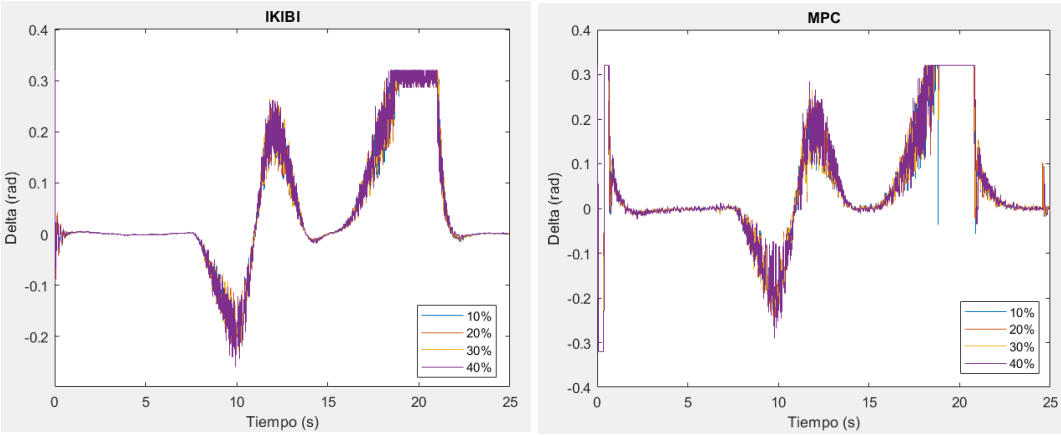

Figura 58. Ángulo de dirección: caso 1 con Vx = 8 m/s y LAD = 5 m.

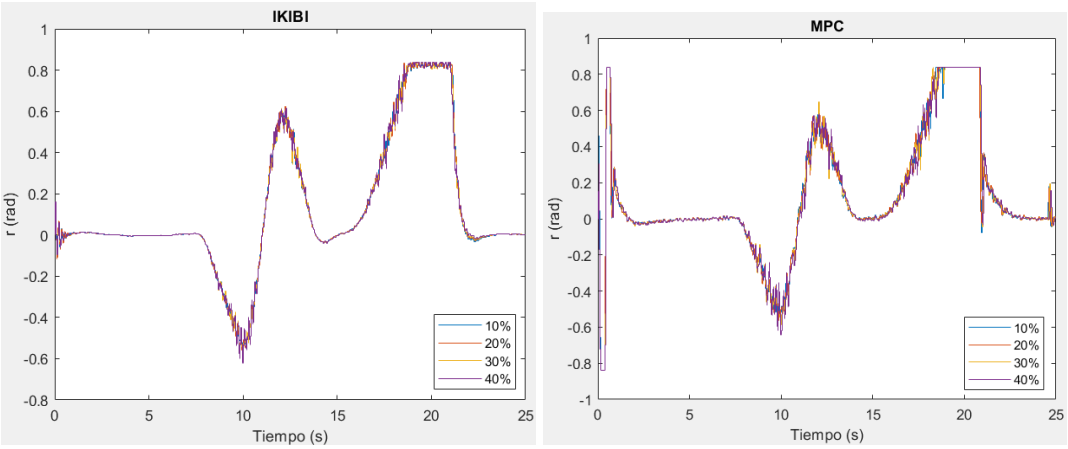

Figura 59. Velocidad angular de la guiñada: caso 1 con Vx = 8 m/s y LAD = 5 m.

Índices	IKIBI				MPC			
	10%	20%	30%	40%	10%	20%	30%	40%
J1	465’7504	457’4076	450’4098	435’0615	561’8980	558’8721	527’2369	507’5157
J1_norm	0’2216	0’2177	0’2143	0’2070	0’2674	0’2660	0’2509	0’2415
J2	1’2522	1’2181	1’1962	1’1303	1’1330	1’0512	0’9765	0’8052
J4	0’7122	0’9692	1’1436	1’2621	0’7152	0’8826	1’0789	1’3919

Tabla 12. Índices caso 1 con LAD = 5m y Vx = 8 m/s.

Aunque el controlador IKIBI sigue mejor el camino en general, el MPC tiene un índice J2 menor, es decir, la máxima desviación que presenta respecto a la trayectoria es menor que la de IKIBI. Aunque aumente el porcentaje de pérdidas, las respuestas apenas empeoran (los índices varían muy poco, menos el J4).

8.3.2 LAD = 10 m y Vx = 12 m/s.

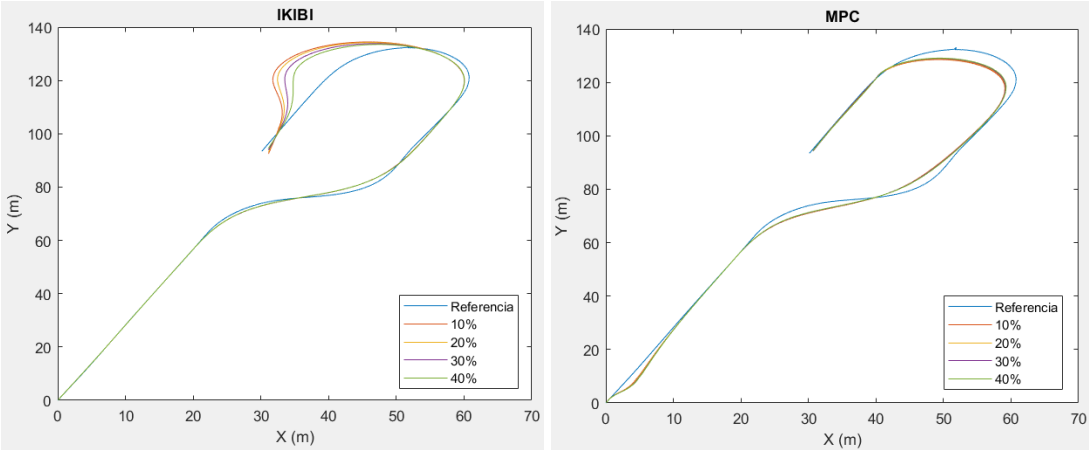

Figura 60. Trayectoria del vehículo: caso 1.

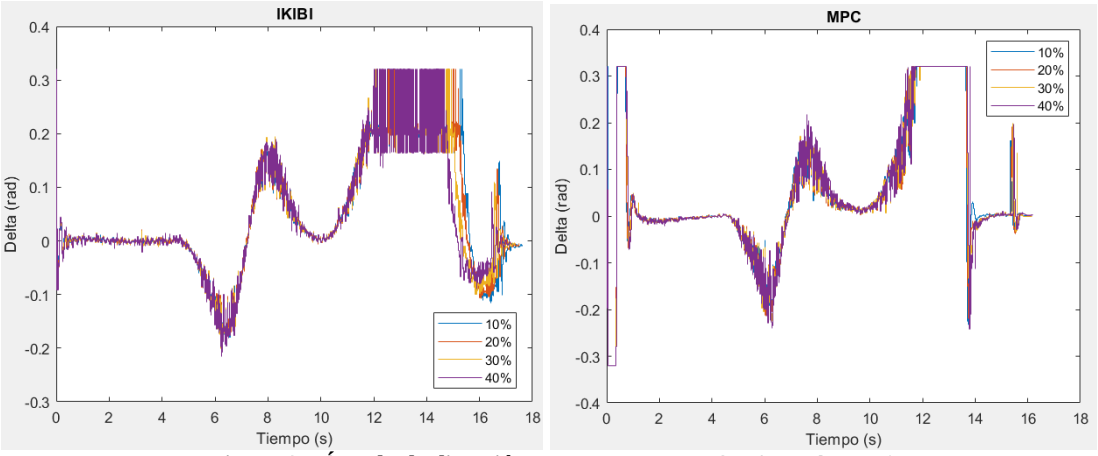

Figura 61. Ángulo de dirección: caso 1 con Vx = 12 m/s y LAD = 10 m.

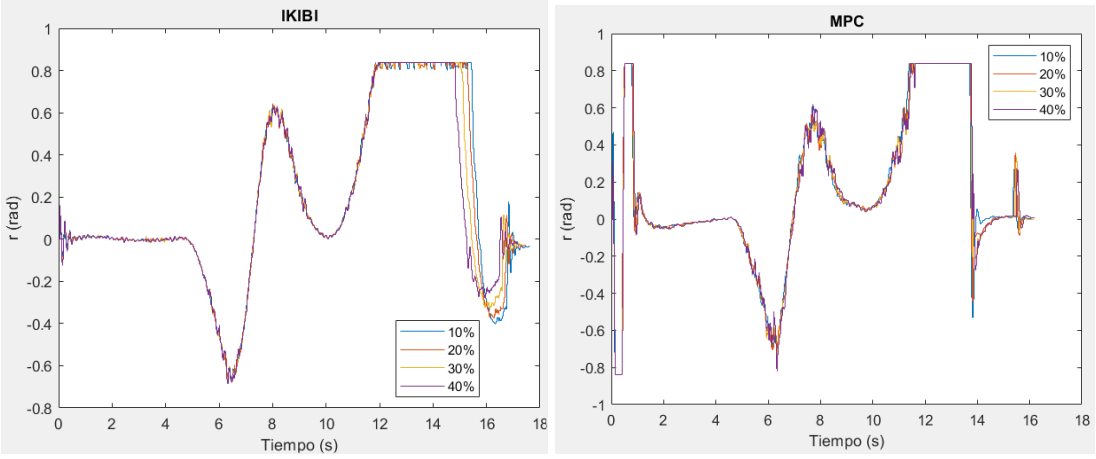

Figura 62. Velocidad angular de la guiñada: caso 1 con Vx = 12 m/s y LAD = 10 m.

No están las simulaciones con LAD = 5 m porque el controlador IKIBI se descontrola. En estas simulaciones se ve que, bajo estas condiciones, el MPC es claramente mejor.

Los resultados con el método 1 (guardando la acción de control cuando llegue y aplicando la misma hasta que se obtenga otra) se adjuntarán en los casos con diferencias más notorias, donde se reflejen mejor las conclusiones extraídas.

El ruido se introduce en el filtro de Kalman cuando corrige, por eso con el método 2, a mayor porcentaje de pérdidas, mejor resultado. En otras circunstancias, en las que el

modelo sea más realista y se tengan en cuenta fenómenos como la fricción, las estimaciones no serían tan buenas.

Es importante aclarar que, a diferencia de la idea propuesta y enunciada en [5], que consiste en fijar un horizonte de predicción muy grande, 50, para el filtro de Kalman, y utilizar esas mismas predicciones con el MPC; lo que se plantea con el método 2 es que el propio bloque manejador haga las predicciones, sin ruido a demanda.

El confort del pasajero siempre es mejor en este caso con el método 1, al mantener la misma acción de control durante más tiempo, el índice 4 se reduce. Si el seguimiento de la ruta es pésimo, se descarta por completo esta opción. Sin embargo, en la situación inicial de este caso, es una ventaja.

8.4 Caso 2.

8.4.1 LAD = 5 m y Vx = 8 m/s.

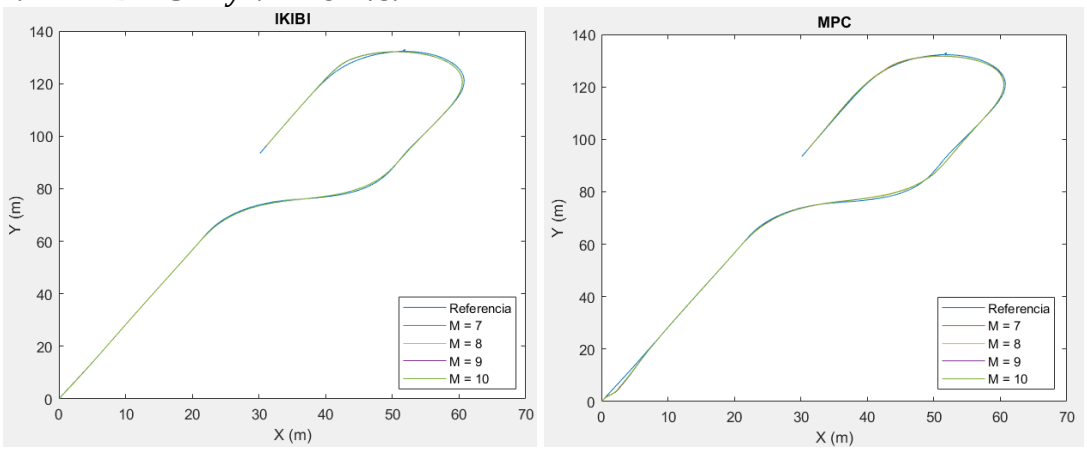

Figura 63. Trayectoria del vehículo: caso 2 con Vx = 8 m/s y LAD = 5.

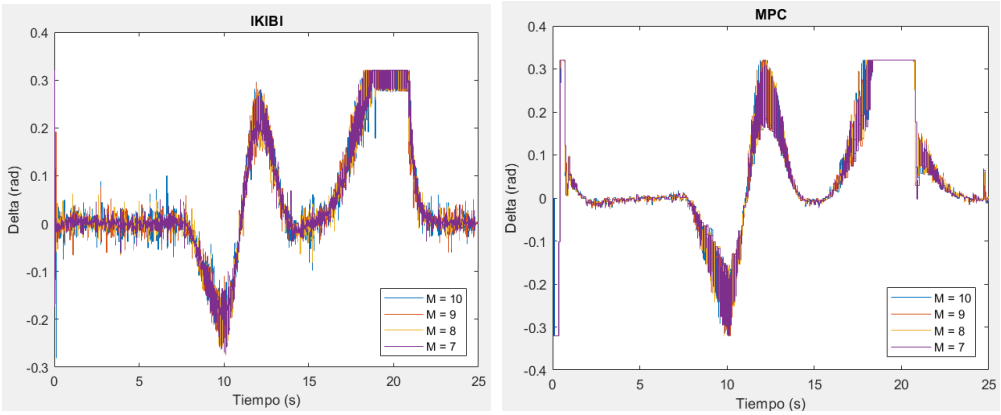

Figura 64. Ángulo de dirección: caso 2 con Vx = 8 m/s y LAD = 5

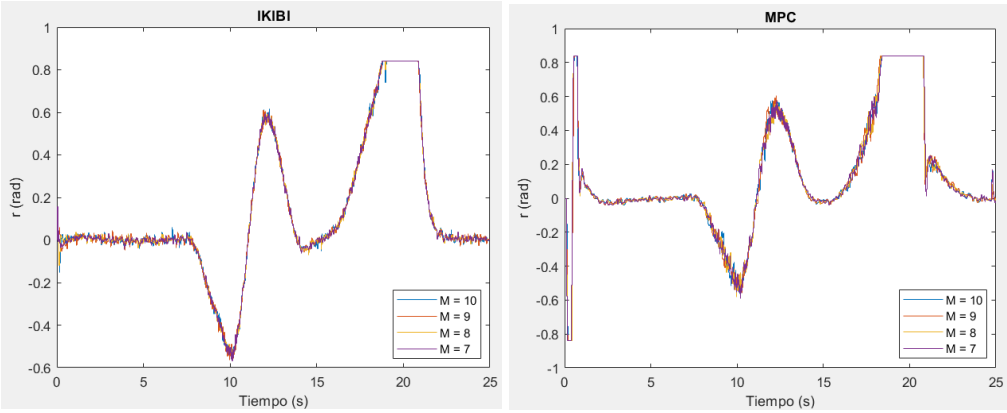

Figura 65. Velocidad angular de la guiñada: caso 2 con Vx = 8 m/s y LAD = 5 m.

Índices	IKIBI				MPC			
	Mp = 7	Mp = 8	Mp = 9	Mp = 10	Mp = 7	Mp = 8	Mp = 9	Mp = 10
J1	361’4746	357’7918	355’0078	356’0037	417’8303	416’2159	395’6608	399’8764
J1_norm	0’1720	0’1702	0’1689	0’1694	0’1988	0’1981	0’1883	0’1903
J2	0’8798	0’9111	0’8556	0’9221	0’7951	0’8132	0’6993	0’7166
J4	2’0114	2’2339	2’2658	2’7754	0’9815	0’8656	0’7846	0’7144

Tabla 13. Índices caso 2.

La diferencia entre distintos Mp es de centímetros y el controlador MPC provoca giros menos bruscos. Por otra parte, los índices son muy similares, y la principal diferencia con respecto al caso base es fundamentalmente debida a que el horizonte de control del MPC se ha tenido que reducir a 2.

8.4.2 LAD = 5 m y Vx = 12 m/s.

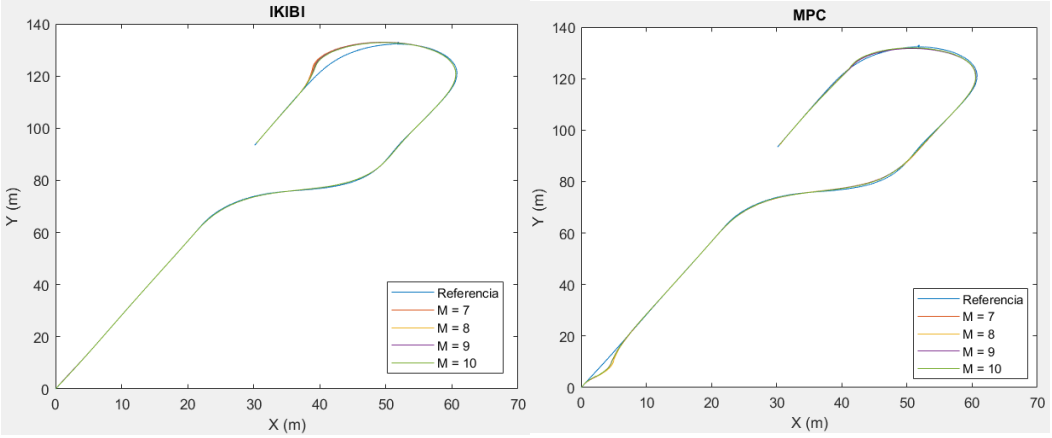

Figura 66. Trayectoria del vehículo: caso 2 con Vx = 12 m/s y LAD = 5 m.

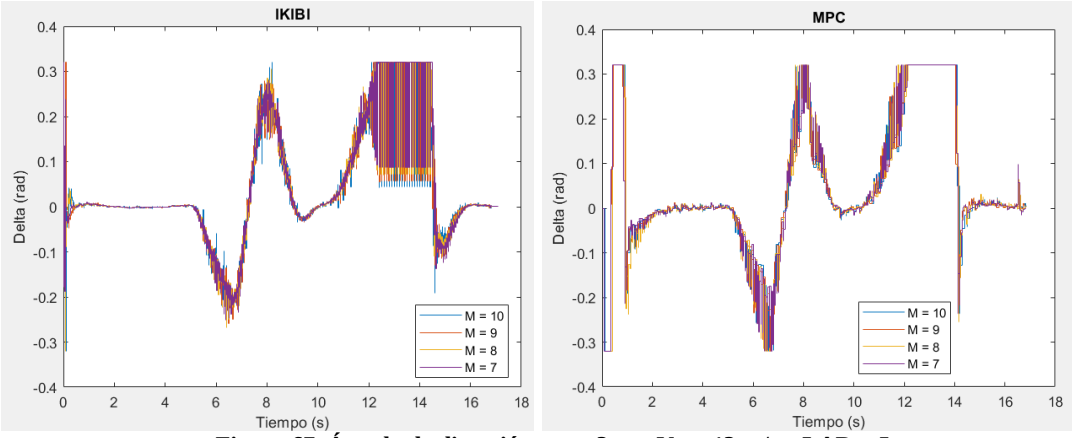

Figura 67. Ángulo de dirección: caso 2 con Vx = 12 m/s y LAD = 5 m.

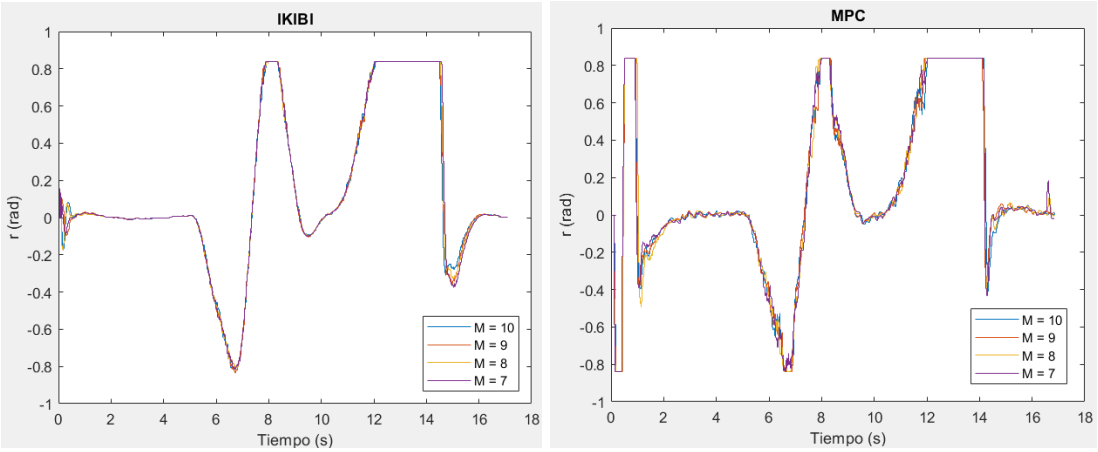

Figura 68. Velocidad angular de la guiñada: caso 2 con Vx = 12 m/s y LAD = 5 m.

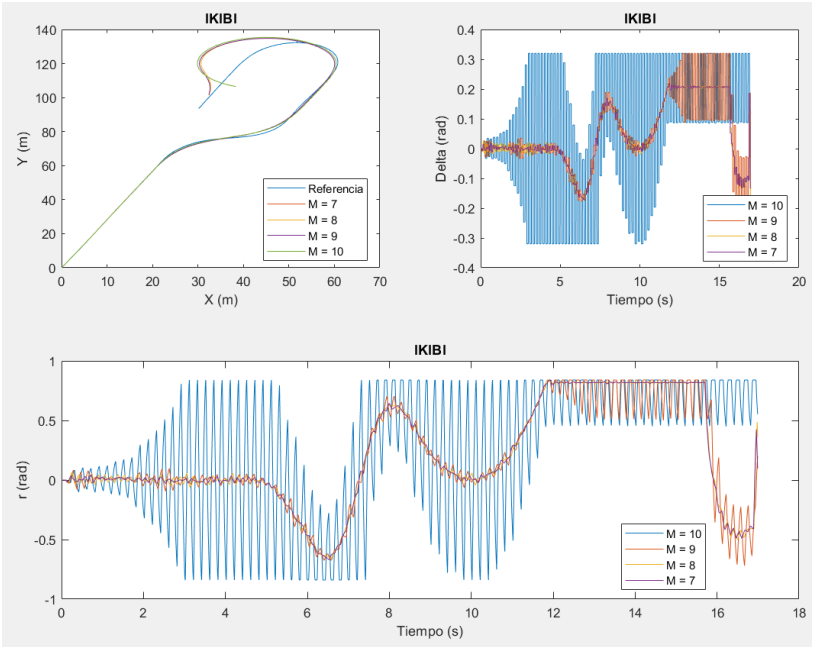

Figura 69. Caso 2 con controlador IKIBI y método 1 m.

En las gráficas recién mostradas se aprecia mejor la diferencia entre ambos métodos para el controlador IKIBI. El método 2 es mejor.

8.5 Caso 3.

8.5.1 LAD = 5m y Vx = 8 m/s.

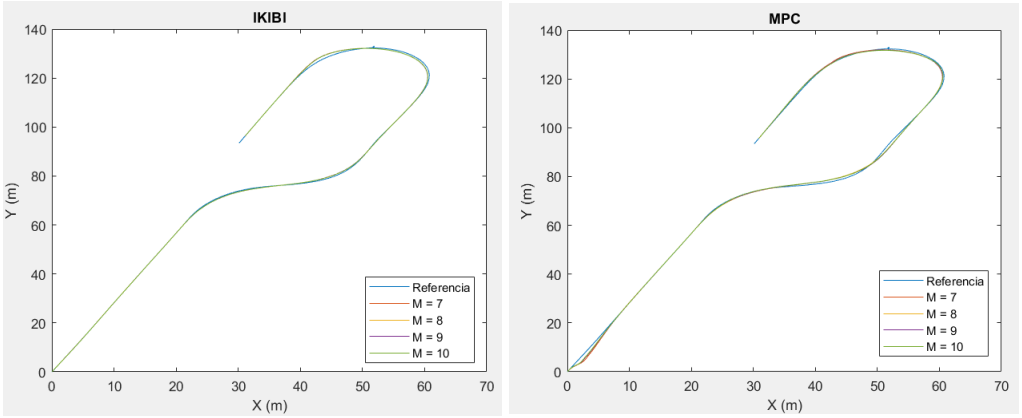

Figura 70. Trayectoria del vehículo: caso 3 con Vx = 8 m/s y LAD = 5 m.

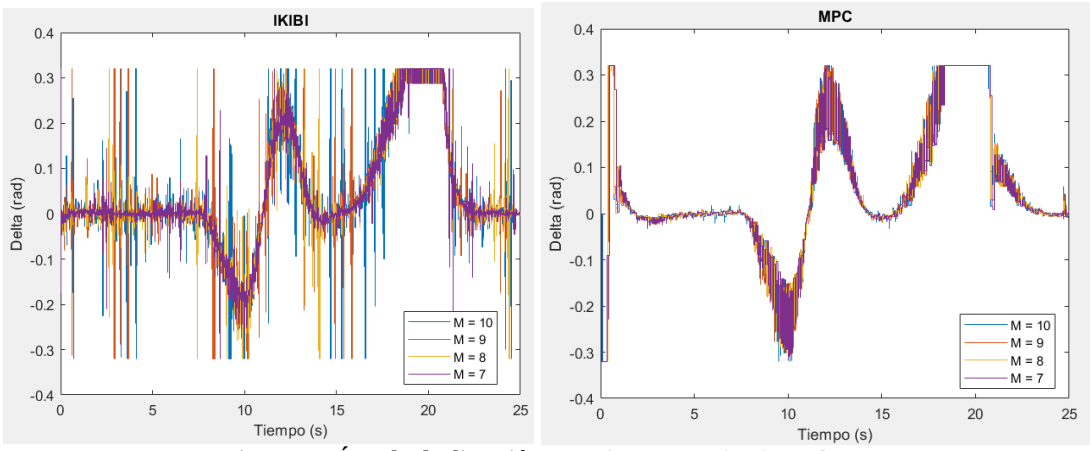

Figura 71. Ángulo de dirección: caso 3 con Vx = 8 m/s y LAD = 5 m.

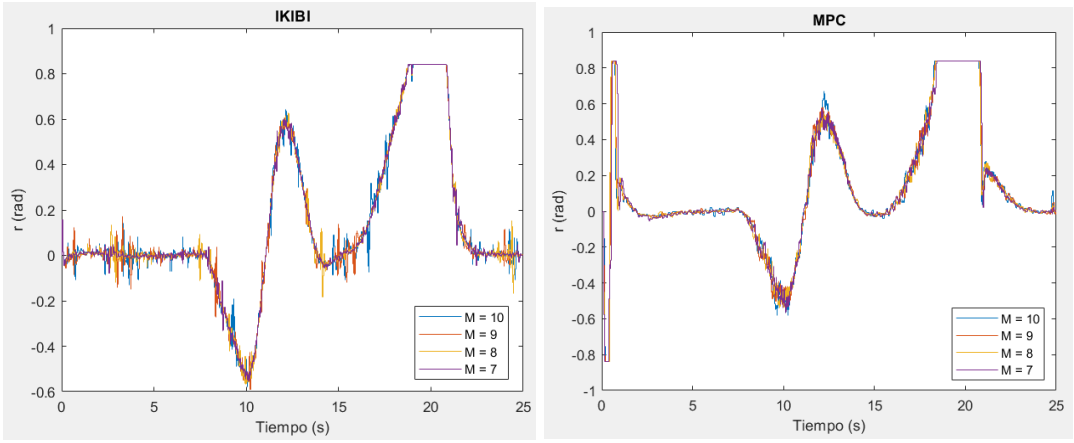

Figura 72. Velocidad angular de la guiñada: caso 3 con Vx = 8 m/s y LAD = 5 m.

Índices	IKIBI				MPC			
	Mp = 7	Mp = 8	Mp = 9	Mp = 10	Mp = 7	Mp = 8	Mp = 9	Mp = 10
J1	352’9263	344’4820	344’1874	326’9748	436’2036	406’7709	406’7709	406’4917
J1_norm	0’1679	0’1639	0’1638	0’1556	0’2076	0’1967	0’1936	0’1934
J2	0’8846	0’8443	0’8243	0’7845	0’8274	0’8123	0’7553	0’7932
J4	2’1940	2’5506	3’2923	3’8532	0’9816	0’8624	0’7861	0’7455

Los controladores (de forma mucho más evidente en el IKIBI que en el PMC) saturan a lo largo de la simulación, lo que es indicio de que se están tomando medidas drásticas dentro de los límites establecidos para hacer frente a las faltas puntuales de información, que ahora son más notorias. No obstante, el comportamiento es satisfactorio (parecido al del caso 2).

8.5.2 LAD = 5 m y Vx = 12 m/s.

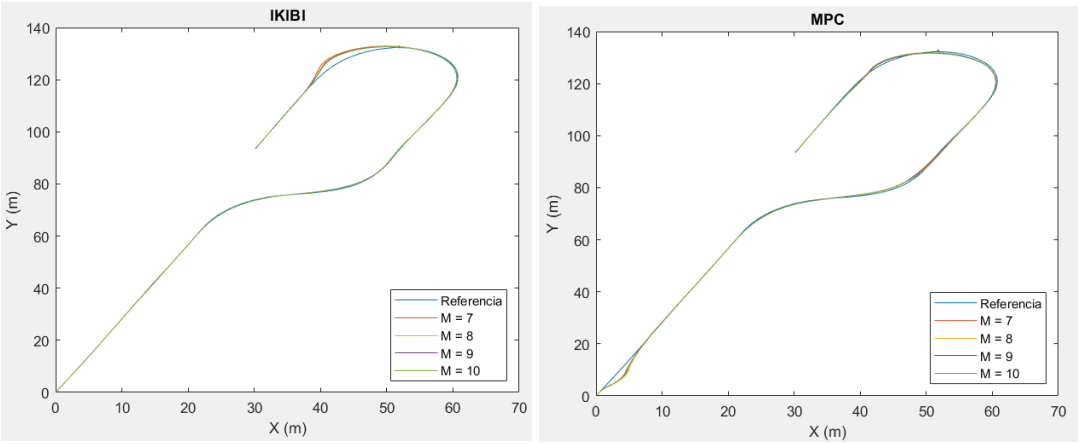

Figura 73. Trayectoria del vehículo: caso 3 con Vx = 12 m/s y LAD = 5 m.

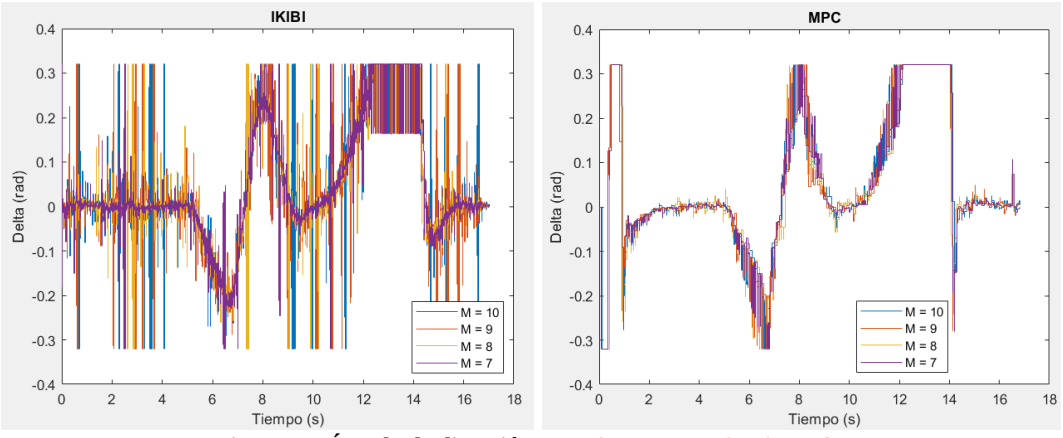

Figura 74. Ángulo de dirección: caso 3 con Vx = 12 m/s y LAD = 5 m.

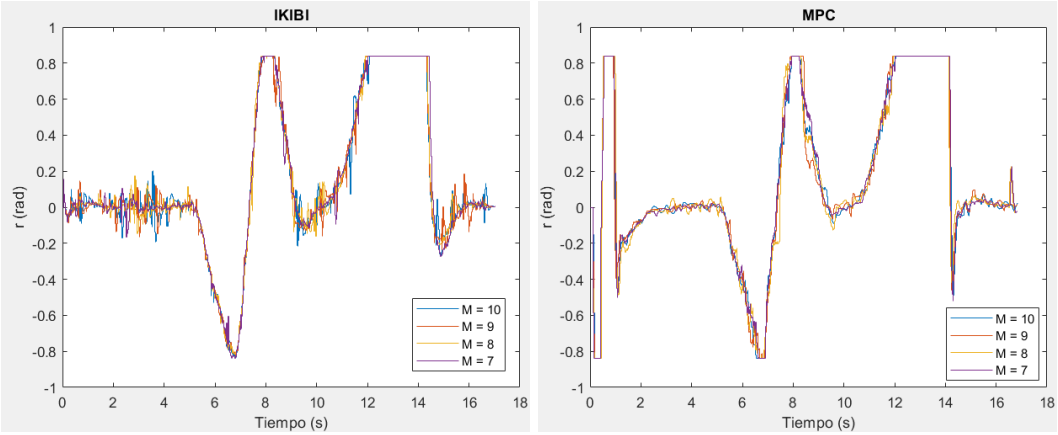

Figura 75. Velocidad angular de la guiñada: caso 3 con Vx = 12 m/s y LAD = 5 m.

De nuevo, como en el caso 2, el controlador IKIBI empeora al aumentar Vx.

Con el método 2 se obtienen los resultados recién mostrados. El método 1 no se adjunta porque no se consigue seguir la trayectoria.

8.6 Caso 4.

8.6.1 Con LAD = 5 m y Vx = 12 m/s.

Es una variante del caso 1. No se considera un único paquete con toda la información del estado del AGV. Es como si cada parámetro llegase de un canal distinto y cada uno fallase con una determinada probabilidad.

Lo que se propone en este trabajo es guardar los últimos estados que han sido enviados correctamente y hacer la corrección con datos que son en parte correctos y en parte no están actualizados.

Con probabilidades de pérdida pequeñas, se asume que la mayoría de los datos con los que se realiza la corrección están actualizados, es decir, han conseguido llegar y son del instante actual. El resto son los de la última vez que llegaron correctamente.

La alternativa es considerar que todo el paquete se ha perdido, y no realizar ninguna corrección. Sería como el caso 1 con una probabilidad de pérdida del 60% si cada dato tiene una probabilidad individual de perderse del 10%.

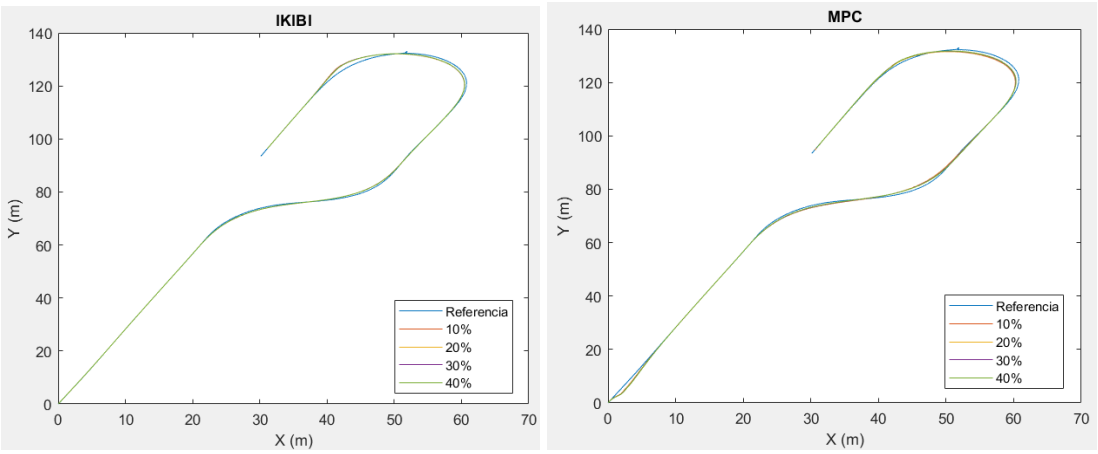

Figura 76. Trayectoria del vehículo: caso 4 con Vx = 8 m/s y LAD = 5 m.

Índices	IKIBI				MPC			
	10%	20%	30%	40%	10%	20%	30%	40%
J1	465,2124	458,1568	450,6085	439,0837	560,6583	529,1774	521,9925	478,1281
J1_norm	0,2214	0,2181	0,2145	0,209	0,2669	0,2519	0,2484	0,2276
J2	1,2421	1,2144	1,198	1,1441	1,0871	0,9935	0,8968	0,8383
J4	1,0278	1,2894	1,4828	1,5982	1,0198	1,2284	1,457	1,5536

Tabla 14. Índices caso 4.

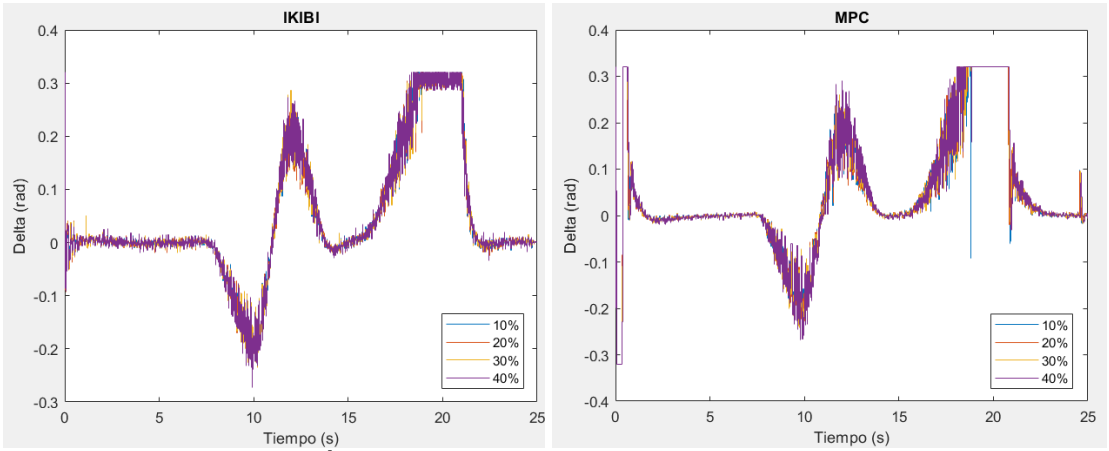

Figura 77. Ángulo de dirección: caso 4 con Vx = 8 m/s y LAD = 5 m.

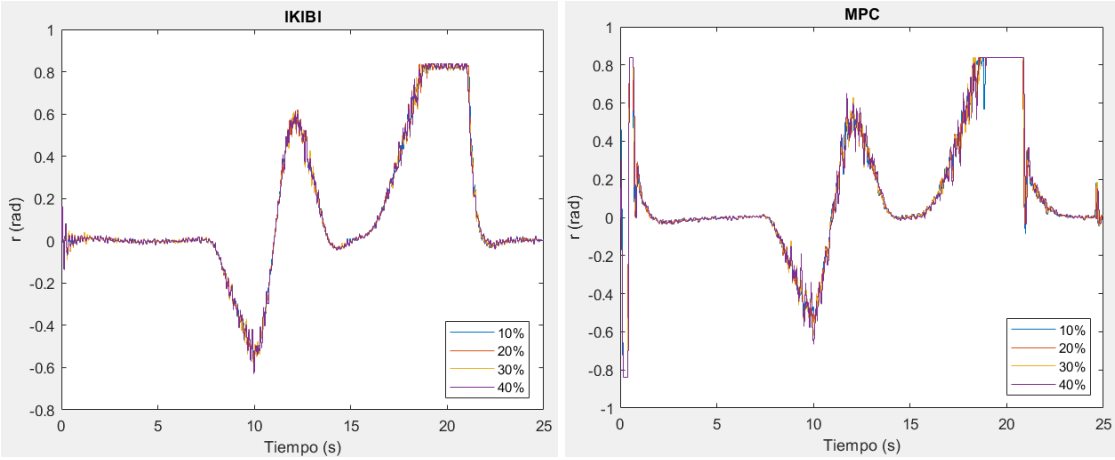

Figura 78. Velocidad angular de la guiñada: caso 4 con Vx = 8 m/s y LAD = 5 m.

El caso 4 tiene índices parecidos al caso 1. Se ha logrado el objetivo de conservar parte de los elementos desactualizados.

8.6.2 Con LAD = 10m y Vx = 12 m/s.

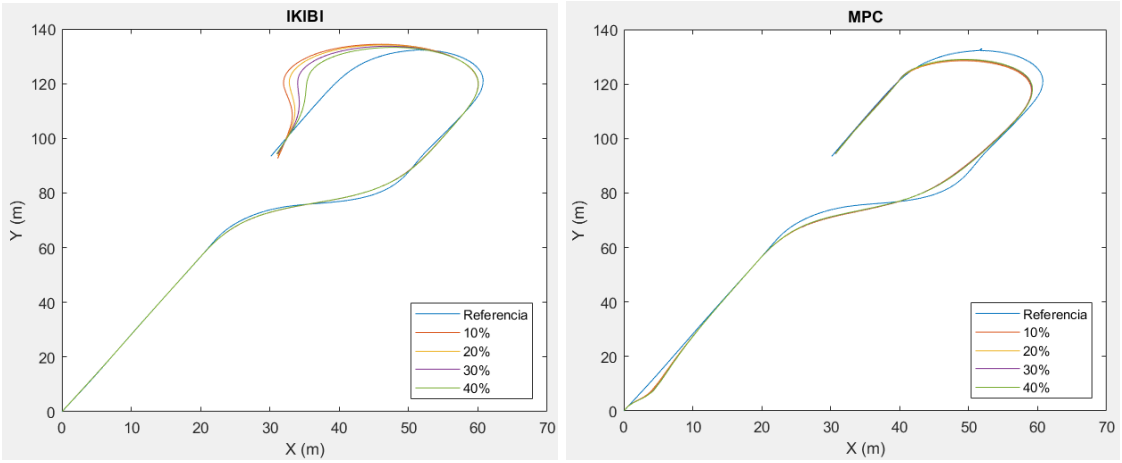

Figura 79. Trayectoria del vehículo: caso 4 con Vx = 12 m/s y LAD = 10 m.

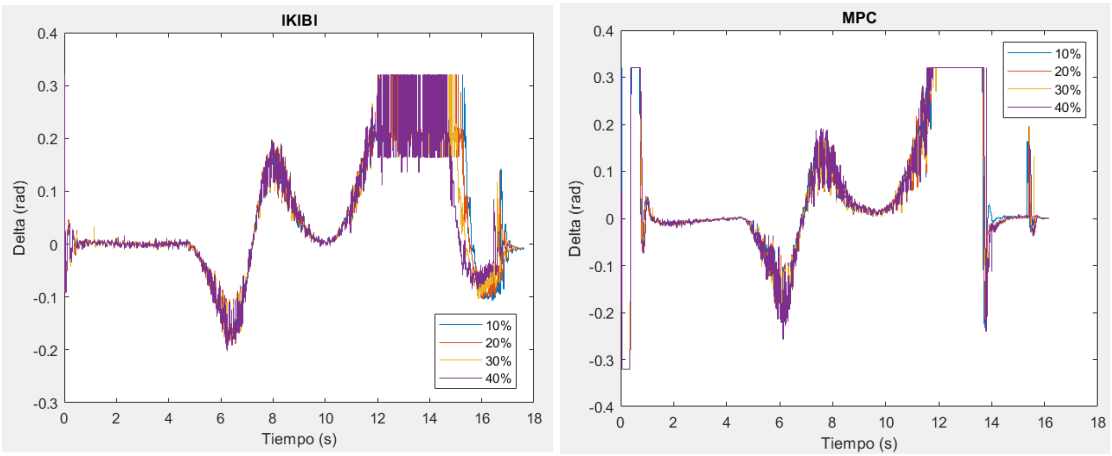

Figura 80. Ángulo de dirección: caso 4 con Vx = 12 m/s y LAD = 10 m.

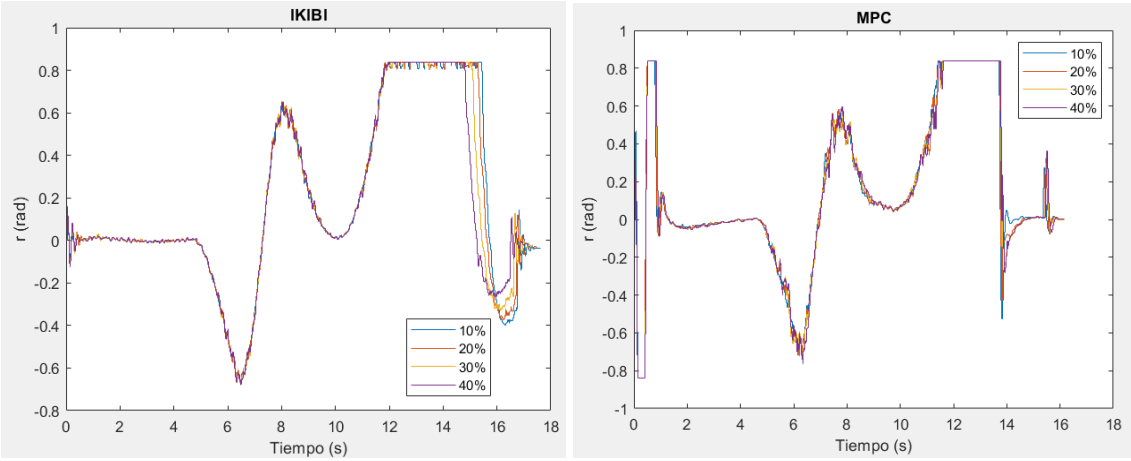

Figura 81. Velocidad angular de la guiñada: caso 4 con Vx = 12 m/s y LAD = 10 m.

8.7 PETC.

8.7.1 LAD = 5 m y Vx = 8 m/s.

Bajo las mismas condiciones del caso base. Con el EKF.

Si se cumple la condición para el vector de estados, se transmite la información y el filtro corrige, de no ser así, el filtro predice.

Si se cumple la condición impuesta sobre la acción de control, se envía la calculada para ese momento. En caso contrario, se envía la anterior (como en el método 1 del IKIBI cuando hay pérdidas).

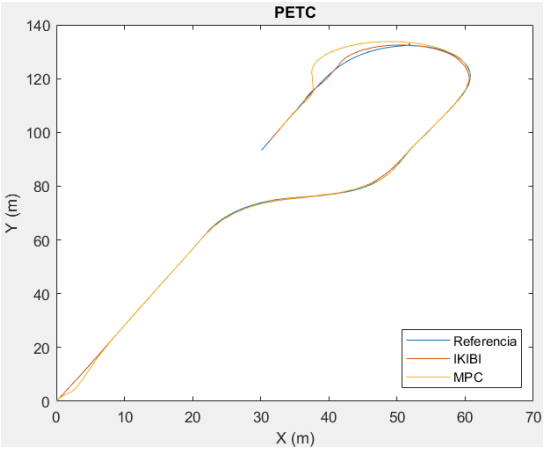

Figura 82. Trayectoria del vehículo: comunicación PETC con Vx = 8 m/s y LAD = 5 m.

Índices	IKIBI	MPC
J1	2.140’4976	2707’1224
J1_norm	1’0188	1’2885
J2	2’8679	4’3971
J3c (%)	42’0064	31’5348
J3s (%)	2’8389	2’8389
J4	1’5697	1’2252

Tabla 15. Índices caso con comunicación PETC con Vx = 8 m/s y LAD = 5 m.

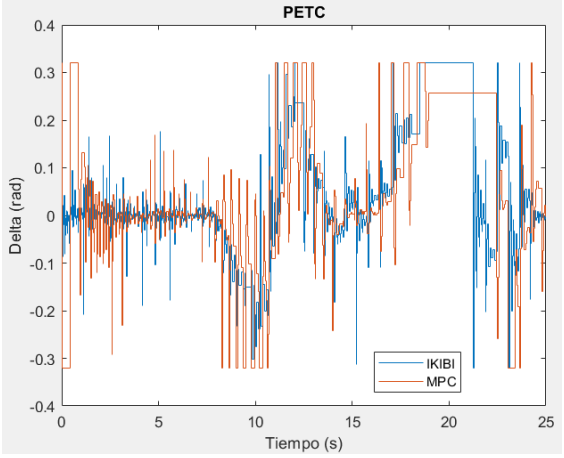

Figura 83. Ángulo de dirección: Comunicación PETC con Vx = 8 y LAD = 5 m.

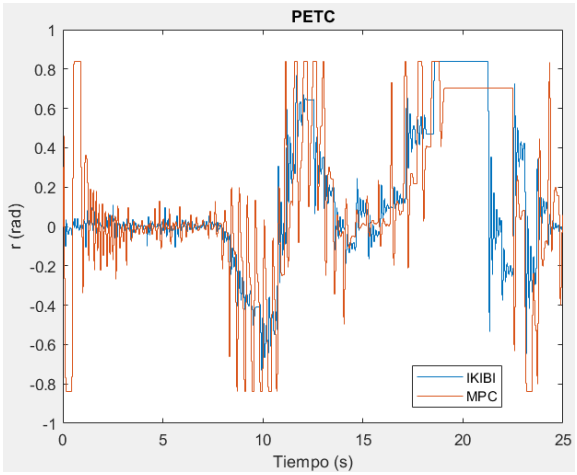

Figura 84.Velocidad angular de la guiñada: comunicación PETC con Vx = 8 m/s y LAD = 5 m.

Los índices muestran que el controlador IKIBI ahorra un 10’47% menos en los envíos controlador-actuador.

Es conveniente comparar con el caso base para averiguar en qué medida disminuye el rendimiento al usar la comunicación PETC.

Incremento de los índices (%)	IKIBI	MPC
J1 normalizado	+ 358,11	+ 368,05
J2	+ 130,91	+ 272,72
J4	+ 133,41	+ 112,35

Tabla 16. Comparación de índices con Vx = 8 m/s para el caso base con y sin PETC.

J1 no se va a incluir en las comparaciones ya que cada recorrido tiene un número de iteraciones diferentes, por lo que los resultados no son concluyentes.

El rendimiento es inversamente proporcional a los índices, puesto que el objetivo de los índices de coste en los problemas de control es reducirlos.

Por la condición de evaluación para decidir si se elimina o no el envío de información, se reducen más envíos con el controlador MPC. Sin embargo, el seguimiento se ve menos perjudicado con el IKIBI.

Este comportamiento es típico: a más envíos, mejor rendimiento en general, pero se ahorra menos.

8.7.2 LAD = 10 m y Vx = 12 m/s.

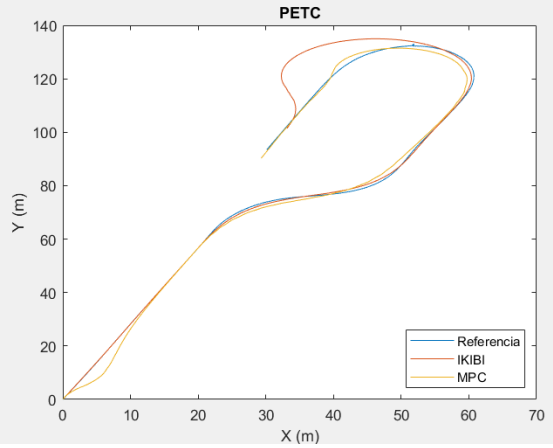

Figura 85. Trayectoria del vehículo: comunicación PETC con Vx = 12 m/s y LAD = 10 m.

Índices	IKIBI	MPC
J1	3.035’1152	1.905’5222
J1_norm	2’0221	1’2695
J2	8’1932	3’6598
J3c (%)	34’4888	40’7168
J3s (%)	3’7625	3’8213
J4	1’2623	1’6156

Tabla 17. Índices caso con comunicación PETC con Vx = 12 m/s y LAD = 10 m.

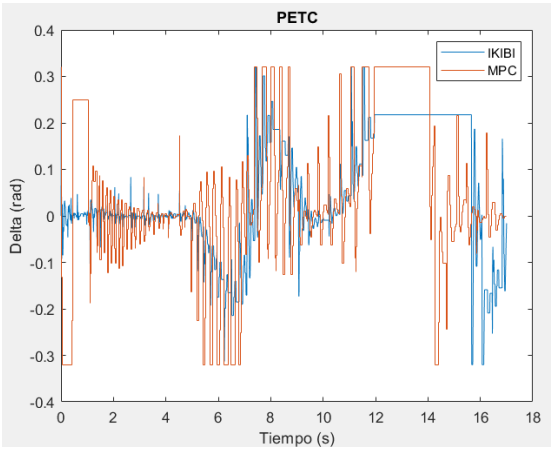

Figura 86. Ángulo de dirección: Comunicación PETC con Vx = 12 y LAD = 10 m.

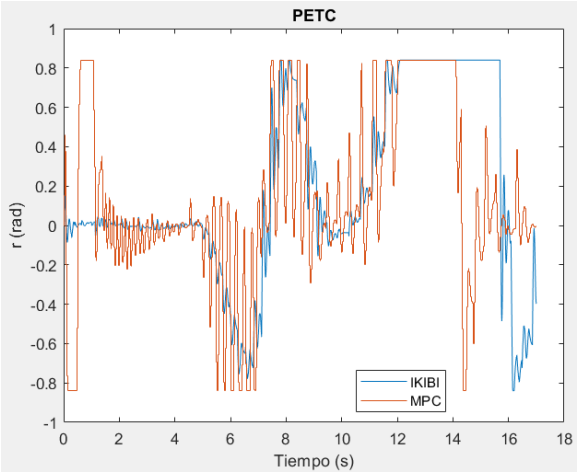

Figura 87.Velocidad angular de la guiñada: comunicación PETC con Vx = 12 m/s y LAD = 10 m.

Para Vx = 12 m/s el MPC es claramente superior en rendimiento y los resultados caben dentro de lo esperado.

Incremento de los índices (%)	IKIBI	MPC
J1 normalizado	+ 13,54	+ 41,17
J2	-11,19	-16,67
J4	+ 189,17	+ 267,51

Tabla 18. Comparación de índices con Vx = 12 m/s para el caso base con y sin PETC.

Por el contrario, el IKIBI en este caso funciona mejor, pero puede que sea porque el rendimiento con Vx = 12 m/s para empezar no sea tan bueno. El controlador IKIBI en esta situación, al dar el giro de 180º, calcula acciones de control con poca variación y se mantiene la misma durante varios segundos. En los instantes de tiempo en los que el controlador saturaba con acciones de control de 0.32 rad, ahora utiliza acciones de control de 0.21 rad.

El MPC ahora envía menos y se rendimiento empeora más. Sin embargo, es porque las nuevas condiciones (una velocidad distinta al punto alrededor del cual está linealizado el LPV) son más demandantes.

Los índices J3 del controlador IKIBI son más pequeños. Significa que se realizan menos envíos porque la información que proviene de los sensores y del controlador varía menos. Por otro lado, el controlador MPC satura más y presenta más variaciones en las señales de control, por lo que requiere más envíos para hacer frente a la situación.

En este apartado, el 8.7, los índices J3s son bastante más pequeños en comparación con los J3c. Según el criterio establecido para medir la variación de la información con el fin de determinar si es descartable o no, los datos del sensor varían menos. Se concluye que, normalmente, si los sensores van provistos de baterías, se consumirán menos estas.

8.8 Conclusiones.

En resumen, las diferencias son pequeñas porque el modelo y las estimaciones, por tanto, son muy buenas, y porque la simulación no incluye condiciones del entorno que empeoran el rendimiento del vehículo.

Las acciones de control estimadas son bastante similares a las reales.

El controlador MPC se adapta mejor a diferentes velocidades y admite restricciones.

Para trayectos sencillos en línea recta y giros de 90º el controlador IKIBI es mejor.

La idea de guardar las acciones de control del MPC y usarlas en distintos instantes ha dado resultados positivos y merecería la pena hacer más pruebas y explorar más a fondo este método.

El caso 1 y el caso 4 devuelven índices con valores parecidos, lo cual indica que el procedimiento empleado es aconsejable usarlo, al menos para la situación simulada.

La comunicación PETC funciona muy bien junto al MPC, ya que se ahorra más del 50% de las transmisiones y sigue ofreciendo un buen rendimiento, a pesar de lo mucho que empeora respecto al caso base.

Con el controlador IKIBI también ahorra más de la mitad de los envíos, pero el seguimiento de la ruta es peor. En algunos casos incluso resulta beneficioso usar la comunicación PETC acompañada de este tipo de controlador, pero son casos excepcionales que no constituyen una norma general.

8.9 Trabajo futuro.

Estas son algunas ideas de cómo podría continuar este trabajo.

- Hacer simulaciones realistas. Usar un modelo Simscape Multibody como el mencionado en el apartado 3.4.1. Como punto intermedio primero se puede configurar el bloque *Vehicle Body 3DOF*.
- Diseñar un controlador MPC no lineal. Mejorará los resultados y las ecuaciones ya se han usado para hacer las predicciones, así que sería sencillo añadirlo.
- Implementar todos los elementos del bucle de control en un vehículo físico.
- Estudiar más casos.

ANEXOS

Relación del trabajo con los Objetivos de Desarrollo Sostenible (ODSs) de la agenda 2020

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza**				X
ODS 2. **Hambre cero**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación y calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**		X		
ODS 12. **Producción y consumo responsables.**			X	
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Tabla 19. Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2020.

El tema de este trabajo está más relacionado con el ODS de Industria.

Abreviaturas

MPC – Model Predictive Control, control predictivo basado en modelos.

IKIBI – Inverse Kinematic Bicycle Model, modelo cinemático inverso de la bicicleta.

LPV – Linear parameter-varying, modelo lineal de parámetros variables.

TFM – Trabajo de fin de Máster.

KF – Kalman Filter, Filtro de Kalman.

DREKF – Dual Rate Extended Kalman Filter, filtro de Kalman extendido bifrecuencia.

RRT - Rapidly Exploring Random Tree, árbol de exploración aletoria rápida.

MATLAB – MATrix LABoratory, laboratorio de matrices.

AGV – Automatic guided vehicle, vehículo de guiado automático.

PETC - Periodic event-triggered communication, comunicación con disparos por eventos.

QP – Quadratic problems, problemas cuadráticos.

ADMM – Alternating Direction Method of Multipliers, método de los multiplicadores de dirección alterna.

ODS – Objetivo de Desarrollo Sostenible.

Referencias.

[1] N. author found, “Waymo - Self-Driving Cars - Autonomous Vehicles - Ride-Hail.” [Online]. Available: https://waymo.com/intl/es/

[2] X. Song, H. Gao, T. Ding, Y. Gu, J. Liu, and K. Tian, “A Review of the Motion Planning and Control Methods for Automated Vehicles”, *Sensors (Basel, Switzerland)*, vol. 23. Sensors (Basel, Switzerland), Jul. 01, 2023. [Online]. Available: https://doi.org/10.3390/s23136140

[3] M. Heemels *et al.*, «Periodic Event-Triggered Control», en *Event-Based Control and Signal Processing*, Marek Miskowicz, pp. 105-119. doi: 10.1201/b19013-8.

[4] R. Pizá, R. Carbonell, V. Casanova, Á. Cuenca, and J. S. S. Llobregat, “Nonuniform Dual-Rate Extended Kalman-Filter-Based Sensor Fusion for Path-Following Control of a Holonomic Mobile Robot with Four Mecanum Wheels”, *Applied Sciences*. Applied Sciences, Mar. 31, 2022. [Online]. Available: https://doi.org/10.3390/app12073560

[5] G. Alite, Á. Cuenca, J. Salt, and M. Tomizuka, “Resource-Efficient Path-Following Control for a Self-Driving Car in a Networked Control System”, *IEEE Access*, vol. 11. IEEE Access, pp. 108011–108023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3321269

[6] J. M. S. Ducaju, J. S. Llobregat, Á. Cuenca, and M. Tomizuka, “Autonomous Ground Vehicle Lane-Keeping LPV Model-Based Control: Dual-Rate State Estimation and Comparison of Different Real-Time Control Strategies”, *Sensors (Basel, Switzerland)*, vol. 21. Sensors (Basel, Switzerland), Feb. 01, 2021. [Online]. Available: https://doi.org/10.3390/s21041531

[7] R. Carbonell, Á. Cuenca, V. Casanova, R. Pizá, and J. S. Llobregat, “Dual-Rate Extended Kalman Filter Based Path-Following Motion Control for an Unmanned Ground Vehicle: Realistic Simulation”, *Sensors (Basel, Switzerland)*, vol. 21. Sensors (Basel, Switzerland), Nov. 01, 2021. [Online]. Available: https://doi.org/10.3390/s21227557

[8] J. M. S. Ducaju, C. Tang, M. Tomizuka, and C.- yao Chan, “Application Specific System Identification for Model-Based Control in Self-Driving Cars”, *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 384–390, Oct. 19, 2020. [Online]. Available: https://doi.org/10.1109/IV47402.2020.9304586.

[9] H. C. Sergio, M. Tomizuka, C. Tang, U. P. De València Departamento De Máquinas Y Motores Térmicos - Departament De Màquines I Motors Tèrmics, and U. P. De València Escuela Técnica Superior De Ingeniería Del Diseño - Escola Tècnica Superior D’Enginyeria Del Disseny, “Lateral dynamics modeling: A black-box modeling of the dynamics of a car,” May 12, 2021. https://riunet.upv.es/handle/10251/142201

[10] C. Xiang, X. Wang, Y. Ma, and B. Xu, “Practical Modeling and Comprehensive System Identification of a BLDC Motor”, *Mathematical Problems in Engineering*, vol. 2015. Mathematical Problems in Engineering, pp. 1–11, Apr. 07, 2015. [Online]. Available: https://doi.org/10.1155/2015/879581

[11] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, “Real-Time Motion Planning With Applications to Autonomous Urban Driving”, *IEEE Transactions on Control Systems Technology*, vol. 17. IEEE Transactions on Control Systems Technology, pp. 1105–1118, Jul. 28, 2009. [Online]. Available: https://doi.org/10.1109/TCST.2008.2012116

[12] N. author found, “Bibliotecas personalizadas - MATLAB & Simulink - MathWorks España.” [Online]. Available: https://es.mathworks.com/help/simulink/libraries.html

[13] «Pure Pursuit Controller», en *Navigation Toolbox User's Guide*, 2019, pp. 915-916. [Online]. Disponible en: https://es.mathworks.com/help/pdf_doc/nav/nav_ug.pdf

[14] N. author found, "Vehicle Body 3DOF - 3DOF rigid vehicle body to calculate longitudinal, lateral, and yaw motion - Simulink - MathWorks España." [Online]. Available: https://es.mathworks.com/help/vdynblks/ref/vehiclebody3dof.html

[15] J. Mattingley and S. P. Boyd, "CVXGEN: a code generator for embedded convex optimization", *Optimization and Engineering*, vol. 13. Optimization and Engineering, pp. 1–27, Nov. 06, 2011. [Online]. Available: https://doi.org/10.1007/s11081-011-9176-9

[16] N. author found, "QP Solvers - MATLAB & Simulink - MathWorks España." [Online]. Available: https://es.mathworks.com/help/mpc/ug/qp-solver.html

[17] C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm",1992. [Online]. Available: https://www.semanticscholar.org/paper/ee756e53b6a68cb2e7a2e5d537a3eff43d793d70

[18] R. Wang, Y. Li, J. Fan, T. Wang, and X. Chen, "A Novel Pure Pursuit Algorithm for Autonomous Vehicles Based on Salp Swarm Algorithm and Velocity Controller", *IEEE Access*, vol. 8. IEEE Access, pp. 166525–166540. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3023071

Índice de tablas.

Índice de figuras.

Manual de usuario.

TFM_script.m

El script *TFM_script.m* contiene la información para representar cualquier caso. Es una modificación del código empleado para representar los casos de [6].

Es lo primero que hay que ejecutar para que se inicialicen todas las variables.

```
clear; close all; clc; % Elimina todos las gráficas y limpia la ventana
% Workspace
%% Workspaces
load('datos.mat');
load('coeficientes.mat');
load ('meas_noise.mat');
load ('proc_noise.mat');

Psi = double(Psi);
c1 = coeficientes_good(1:9);    % a1
c2 = coeficientes_good(10:18);  % a2
d0 = coeficientes_good(19:27);  % b0
d1 = coeficientes_good(28:36);  % b1
d2 = coeficientes_good(37:45);  % b2

% Vx = dx/dt y Vy = dy/dt. Son las velocidades longitudinales y laterales
% del vehículo respecto a su sistema de coordenadas
Vx = 8; % Vx cte.
Vy = 0;
ay = 0;
T = 0.01;
MT = 0.1;

psi_ini = 1.2216; %psi_ini = Psi(1);
% Identificación de los parámetros del modelo lineal alrededor de Vx m/s
a1 = c1(1)+c1(2)*ay+c1(3)*ay^(2)+c1(4)*ay/Vx+c1(5)*(ay^(2)/Vx) ...
    +c1(6)*ay/(Vx^2)+c1(7)/Vx+c1(8)/(Vx^2)+c1(9)*(ay^(2)/(Vx^2));

a2 = c2(1)+c2(2)*ay+c2(3)*ay^(2)+c2(4)*ay/Vx+c2(5)*(ay^(2)/Vx) ...
    +c2(6)*ay/(Vx^2)+c2(7)/Vx+c2(8)/(Vx^2)+c2(9)*(ay^(2)/(Vx^2));

b0 = d0(1)+d0(2)*ay+d0(3)*ay^(2)+d0(4)*ay/Vx+d0(5)*(ay^(2)/Vx) ...
    +d0(6)*ay/(Vx^2)+d0(7)/Vx+d0(8)/(Vx^2)+d0(9)*(ay^(2)/(Vx^2));

b1 = d1(1)+d1(2)*ay+d1(3)*ay^(2)+d1(4)*ay/Vx+d1(5)*(ay^(2)/Vx) ...
    +d1(6)*ay/(Vx^2)+d1(7)/Vx+d1(8)/(Vx^2)+d1(9)*(ay^(2)/(Vx^2));

b2 = d2(1)+d2(2)*ay+d2(3)*ay^(2)+d2(4)*ay/Vx+d2(5)*(ay^(2)/Vx) ...
    +d2(6)*ay/(Vx^2)+d2(7)/Vx+d2(8)/(Vx^2)+d2(9)*(ay^(2)/(Vx^2));

%% Modelo lineal LPV
A = [(1-a1) (a1-a2) a2; 1 0 0; 0 1 0];
B = [1 0 0]';
C = [T*b0 T*b1 T*b2];
D = 0;

model = ss(A,B,C,D,T);
```

```
z=tf([1 0],1,0.01);
fdt2 = (b0*z^2 + b1*z + b2)/(z^2 + a1*z + a2);

%% Constantes para el modelo no lineal
% Los mismos valores para el modelo de Rajamani y el de Stanford.
m = 1800; % kg
a = 1.2; % m
b = 1.65; % m
muf = 0.6;
mur = 0.6;
cf = 140000; % N/rad
cr = 120000; % N/rad
h = 0.35; % m
Izz = 3270; % kg m2
Lv = a + b; % Sumar las distancias entre las ruedas delanteras y traseras
% con respecto al centro de gravedad (CG) del vehículo.
Vmin = 5*0.44704; % m/s

%% Controlador MPC
% MV = u = ángulo de dirección
% MO = y = ángulo de guiñada
P = 20; % Horizonte de predicción
M = 10; % Horizonte de control
mpc1 = mpc(model,T,P,M);
mpc1.Weights.ManipulatedVariables = 0.001; % R
mpc1.Weights.OutputVariables = 1; % Q
mpc1.ManipulatedVariables(1).Min = -0.32;
mpc1.ManipulatedVariables(1).Max = 0.32;
mpc1.ManipulatedVariables(1).RateMin = -1;
mpc1.ManipulatedVariables(1).RateMax = 1;
% Crear el estado MPC con el valor inicial de Psi
xmpc = mpcstate(mpc1,[0;0;1.2216/(T*b0)]);
state0 = [0;0;1.2216/(T*b0)];
xmpc.Plant = state0;

mpc2 = mpc(model,T,P,M);
mpc2.Weights.ManipulatedVariables = 0.001; % R
mpc2.Weights.OutputVariables = 1; % Q
mpc2.ManipulatedVariables(1).Min = -0.32;
mpc2.ManipulatedVariables(1).Max = 0.32;
mpc2.ManipulatedVariables(1).RateMin = -1;
mpc2.ManipulatedVariables(1).RateMax = 1;
mpc2.Optimizer.Algorithm = "interior-point";
xmpc1 = mpcstate(mpc2,[0;0;1.2216/(T*b0)]);

%% Para el EKF
final_simu = 6000; % Ruidos para simulaciones hasta 60 segundos.
% Inicialización de los ruidos.
w_ekf=zeros(6,final_simu); % Ruidos de las medidas
v_ekf=zeros(4,final_simu); % ruidos del proceso

%% Para crear nuevos ruidos
% Comentar si se quieren usar siempre los mismos ruidos.
 mu=0;
 Wsigma=0.1;
 for i=1:6
     w=Wsigma*randn(final_simu,1)+mu;
     w=w';
     w_ekf(i,:)=w;
```

```
 end
 Vsigma=0.1;
 for i=1:4
     v=Vsigma*randn(final_simu,1)+mu;
     v=v';
     v_ekf(i,:)=v;
 end

 save meas_noise w_ekf
 save proc_noise v_ekf

%% Para usar los mismos ruidos.
load meas_noise
load proc_noise

%% Escasez de información
% Probabilidades de pérdida
pPC = 0.4; % controlador - actuador
% sensor - controlador
pP1 = 0.4; % caso 1
pP3 = 0.1; % caso 3
pP4 = 0.1/6; % caso 4
% FK: caso 1 | caso 2 | caso 3 | DREKF | EKF | caso 4
Mp = 10;
FK = 1;

% PETC (Comunicación periódica activada por eventos)
% 0 desactivada | 1 activada
petcs = 0; % Controlador
petcz = 0; % Sensor

%% Algoritmo Pure Pursuit
%% REFERENCIAS
LAD = 5;
xref_tot = X;
yref_tot = Y;

%% Controlador IKIBI
gamma = 1; % La señal de entrada es considerada directamente el ángulo de la
rueda
Kp = 0.55; % Constante proporcional
%% índices
% mpcoikibi: 0 IKIBI | 1 MPC
mpcoikibi = 0;
open('TFM_Simulink.slx');
out = sim('TFM_Simulink.slx');

start = 401; % Para que el inicio del MPC no afecte a los índices
Xsim = out.xsimulink.signals.values(start:end);
Ysim = out.ysimulink.signals.values(start:end);

Xrefsim = out.xrefsimulink.signals.values;
Yrefsim = out.yrefsimulink.signals.values;
index = 1;
distmin = inf;
for i=1:length(out.xrefsimulink.signals.values)
    dist = sqrt((Xrefsim(i)-Xsim(1)).^2+(Yrefsim(i)-Ysim(1)).^2);
    if dist < distmin
        distmin = dist;
```

```
        index = i;
    end
end
Xrefsim = Xrefsim(index:index + length(Xsim) - 1);
Yrefsim = Yrefsim(index:index + length(Ysim) - 1);
stsim = out.st_angle.data;

distptopto = sqrt((Xrefsim-Xsim).^2+(Yrefsim-Ysim).^2);

% Índice J1
J1=0;
for i=1:length(Xrefsim)
    J1=J1+distptopto(i);
end

distbuena=zeros(1,length(distptopto));
for i=1:length(Xsim)
    distbuena(i)=100000;
    for j=1:length(Xrefsim)
        distbuena(i)=min(distbuena(i),(sqrt((Xrefsim(j)-
Xsim(i))^2+(Yrefsim(j)-Ysim(i))^2)));
    end
end

figure
plot(distptopto)
hold on
plot(distbuena)
legend('distptopto','distbuena')

% Índice J1 del paper
J1mod=0;
for i=1:length(Xrefsim)
    J1mod=J1mod+distbuena(i);
end

% Índice J1 normalizado
J1mod_norm = J1mod / length(Xrefsim);

% Índice J2
J2=distptopto(1);
for i=1:length(Xrefsim)
    J2=max(J2,distptopto(i));
end

% Índice J2 del paper
J2mod=distbuena(1);
for i=1:length(Xrefsim)
    J2mod=max(J2mod,distbuena(i));
end

% Índice J3 del paper
% Índice J3c controlador - actuador
NoTPETCc = max(out.PETCc);
NoTTTCc = max(out.TTCc);
J3c = (NoTPETCc/NoTTTCc)*100;

% Índice J3s sensor - controlador
NoTPETCs = max(out.PETCs);
```

```
NoTTTCs = max(out.TTCs);
J3s = (NoTPETCs/NoTTTCs)*100;

% J5 evalúa el confort del pasajero. Sirve para estudiar
% restricciones sobre delta (st angle).
J4 = 0;
for i=1:(length(stsim)-1)
    J4=J4+norm(stsim(i+1)-stsim(i))/max(out.tout);
end

% En la memoria J1 es J1mod y J2 es J2mod
J1;
J1mod
J1mod_norm
J2;
J3c
J3s
J2mod
J4
save("Indices.mat","J1mod","J1mod_norm","J2mod","J4");

%% GRÁFICAS
figure
plot(xref_tot,yref_tot);
hold on;
plot(out.xsimulink.signals.values,out.ysimulink.signals.values);
legend('Referencia','xySimulink');
xlabel("X (m)");
ylabel("Y (m)");
```

Después de ejecutar, se obtienen dos gráficas.

Una muestra en azul la distancia, punto a punto, entre la referencia calculada por el algoritmo Pure Pursuit y la posición del AGV y en rojo la distancia mínima entre la posición del AGV y cualquier punto de la referencia generada. Sirven para obtener los índices J1 y J2.

La segunda gráfica compara la ruta de referencia, no la del algoritmo, con el recorrido del vehículo.

TFM_Simulink.slx

El archivo en Simulink donde se realiza la simulación, y están todos los elementos conectados, se llama *TFM_Simulink.slx*.

El aspecto varía en ocasiones con lo mostrado en figuras previas en la memoria, debido a que se han encapsulado muchos bloques para analizar mejor los casos. Por eso, las otras figuras lo que enseñan son los bloques dentro de los subsistemas y los subsistemas variantes, que se activan según las condiciones de *TFM_script.m*.

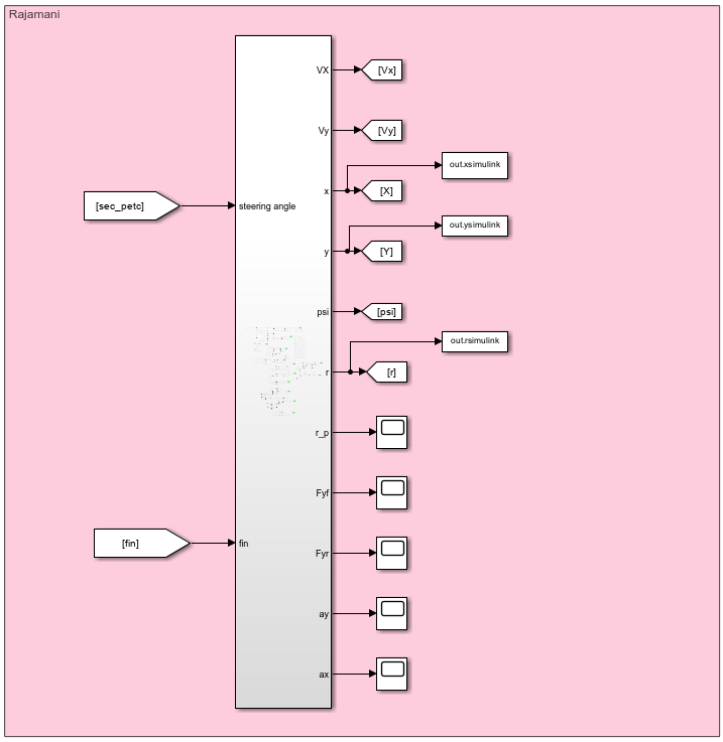

Figura 88. TFM_Simulink: Modelo no lineal de Rajamani

Intercambiable por el de Stanford. Las entradas están configuradas para que sean las de PETC, si está desactivado, no se produce ningún cambio y la salida del bloque es la misma que la entrada.

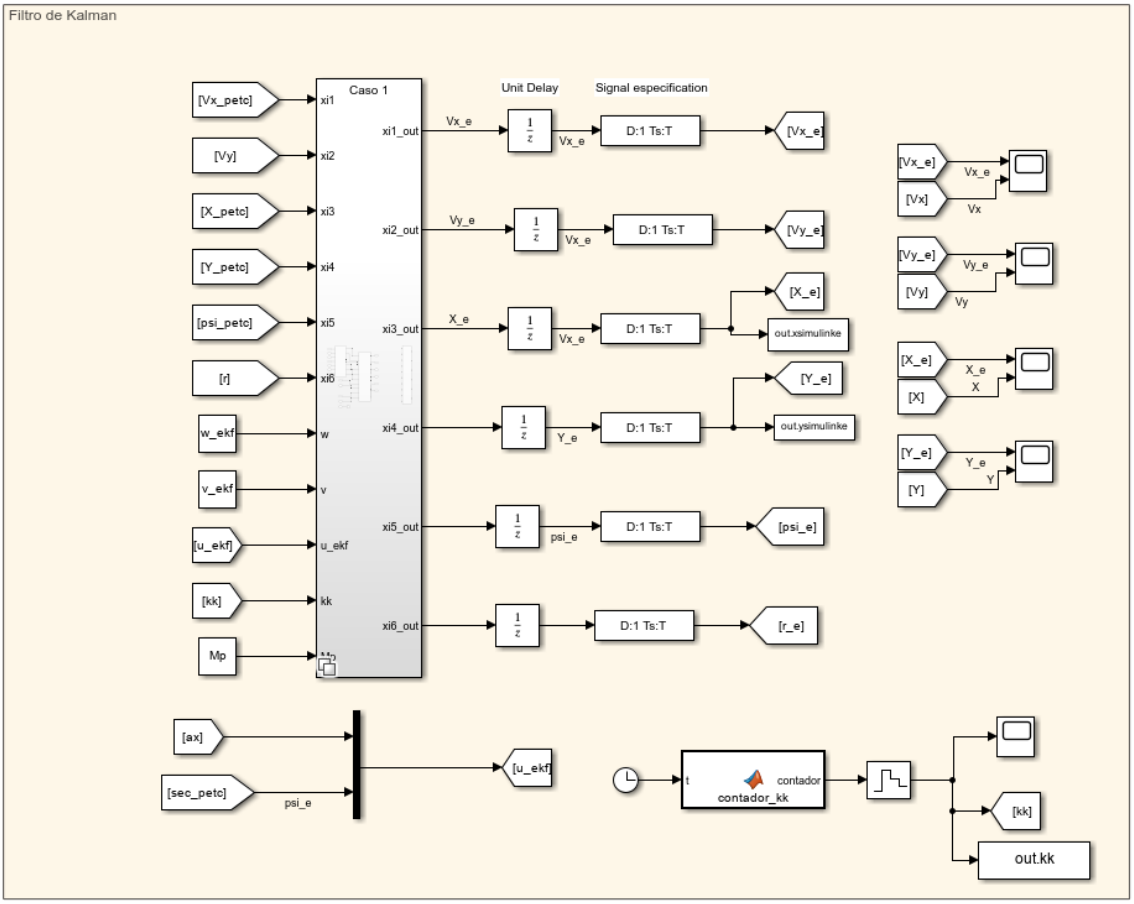

Figura 89. TFM_Simulink: Filtro de Kalman.

Hay riesgo cuando se trabaja con bloques en Simulink de generar bucles algebraicos, los cuales pueden derivar en discontinuidades, o uno puede acabar queriendo tomar como entrada datos que no existen. Se puede evitar usando bloques *Unit Delay*, que retrasan la señal.

Otro posible problema, que surgió en la elaboración de este trabajo, son las señales con dimensiones no especificadas. La solución, añadir bloques del tipo *Signal especification*.

El contador kk es un índice que aumenta una unidad cada periodo, T = 0.01 segundos.

El bloque llamado *caso 1* cambia según el valor de FK del script.

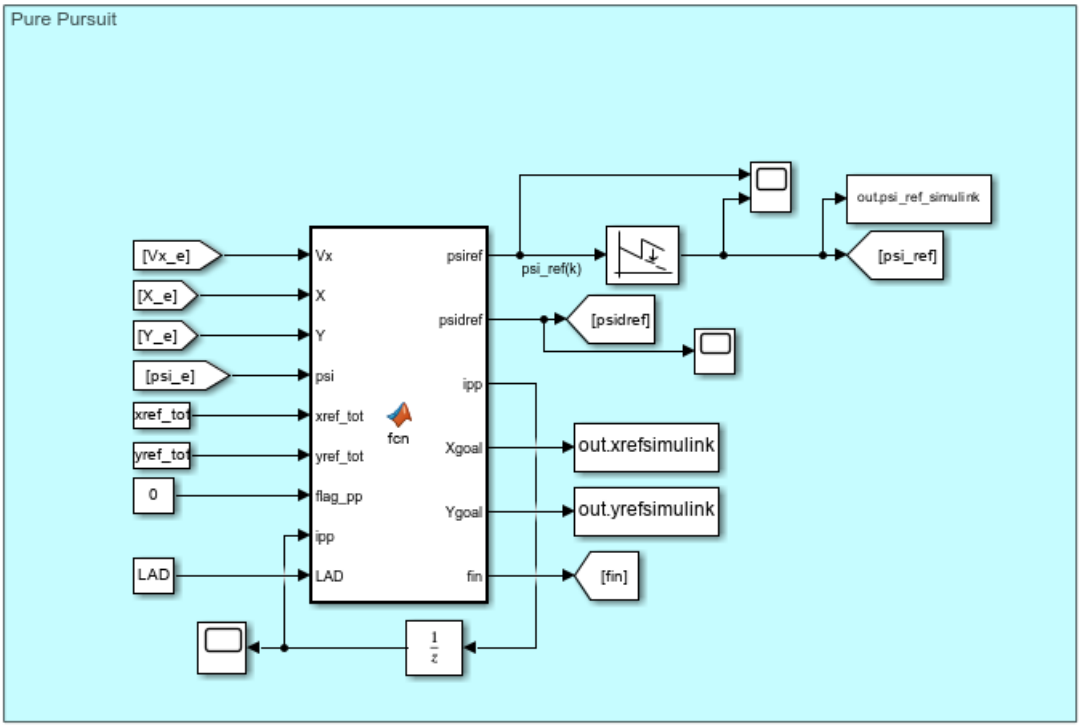

Figura 90. TFM_Simulink: Algoritmo Pure Pursuit.

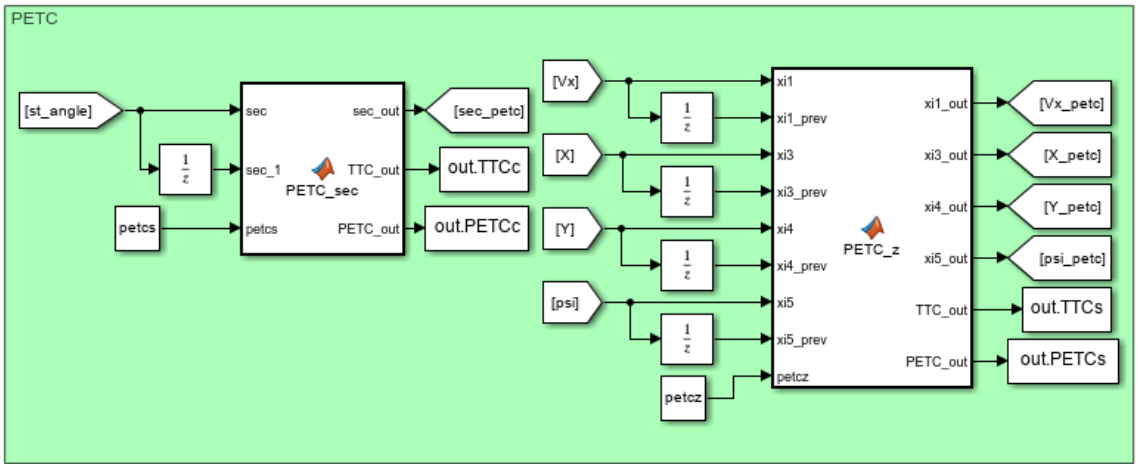

Figura 91. TFM_Simulink: Comunicación PETC.

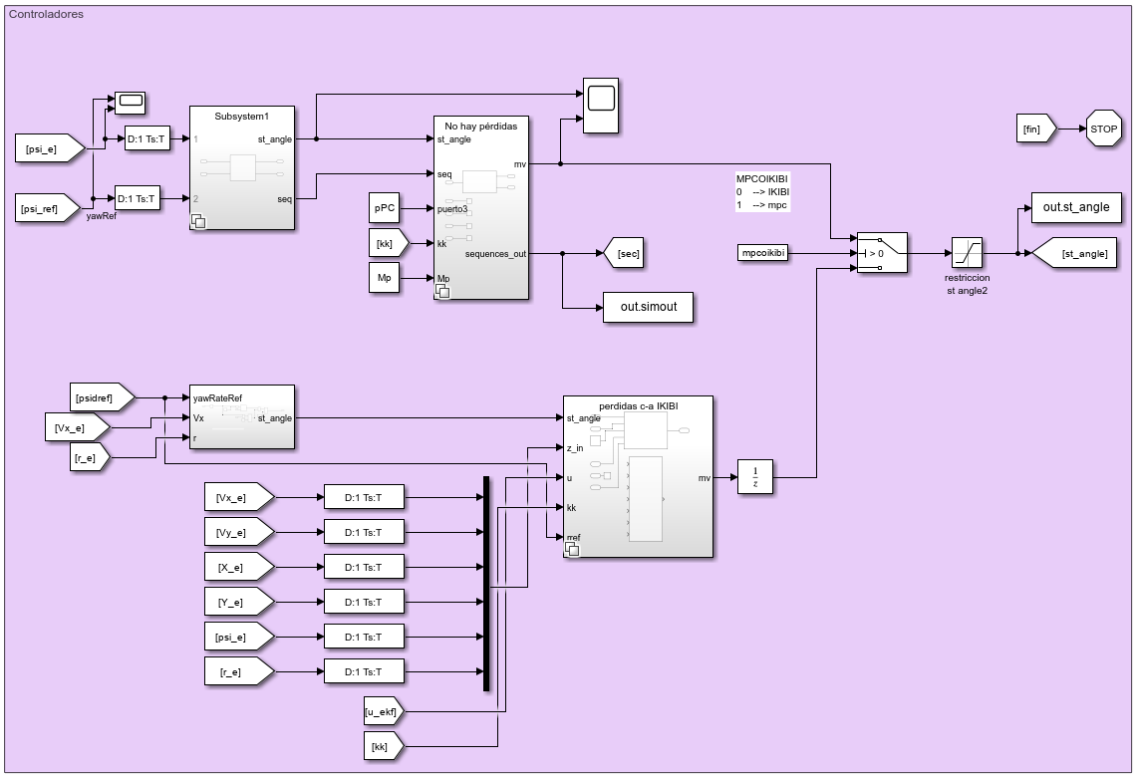

Figura 92. TFM_Simulink: Controladores.

Hay dos controladores MPC, cada uno con un algoritmo interno distinto para hacer las predicciones. Se tienen que cambiar desde Simulink.

Encima del subsistema, haciendo click con el botón derecho, hay que seleccionar *Block Parameters (Subsystem)* y escribir en la opción deseada, *true* y en la otra *false*.

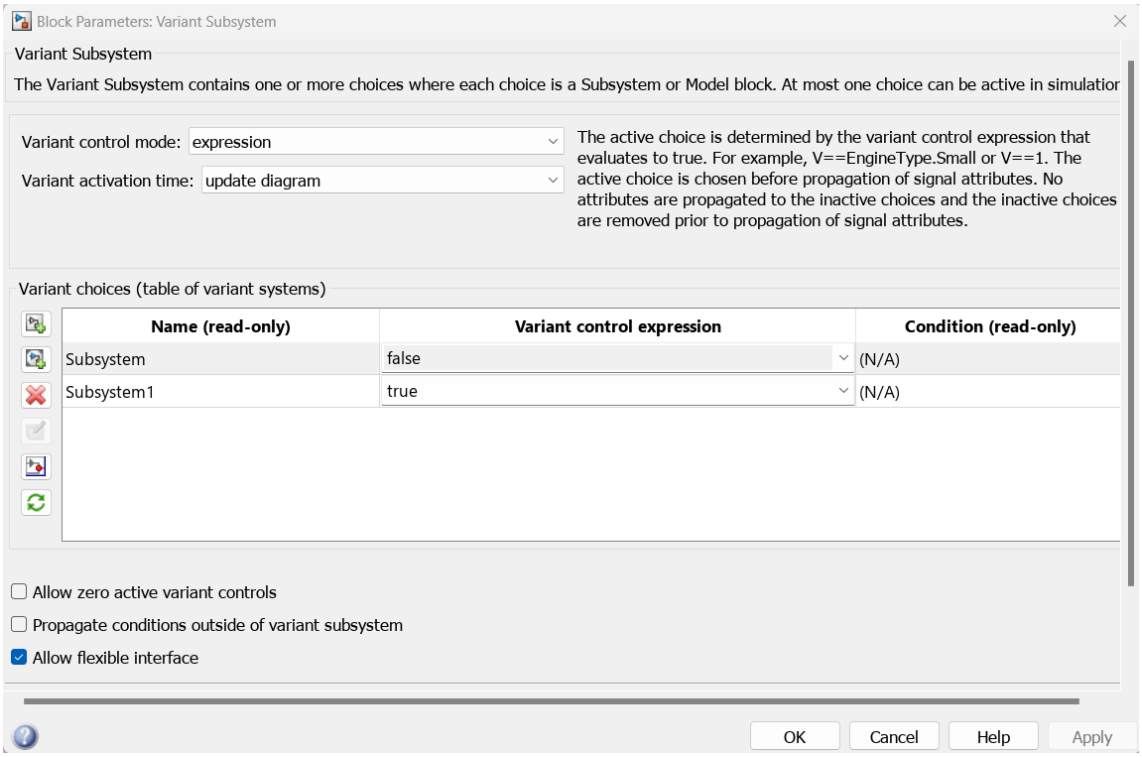

Figura 93. Parámetros de un subsistema variante.

Para el resto de los subsistemas no hace falta este paso, ya está programada la elección.

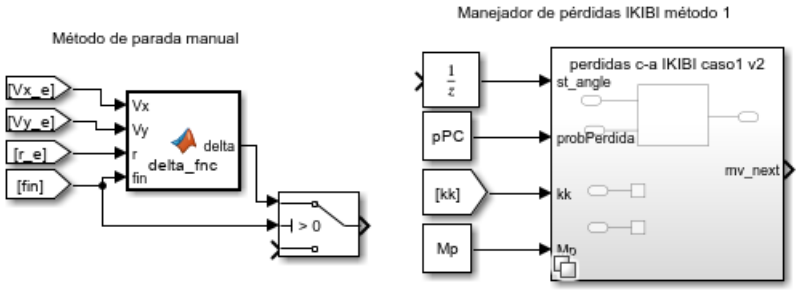

Figura 94. TFM_Simulink: Otras opciones.

biblioteca_TFM.slx

Casi todos los bloques, menos los más sencillos que se han creado están guardados en una biblioteca personalizada de MATLAB, con el nombre biblioteca_TFM.slx.

Los cambios a los bloques que pertenecen a la biblioteca se tienen que realizar desde su archivo Simulink después de desbloquearla.

Ejecutar *slblocks.m* para que la biblioteca se encuentre en el buscador de librerías *Library Browser* con el nombre *TFM*.

Archivos de simulación, generación de gráficas y tablas.

Habiendo ejecutado previamente *TFM_script.m*, cada caso se puede simular con su archivo correspondiente.

Tienen el mismo nombre que el caso seguido de *_sim* (Menos las que tienen el método 2 en el controlador IKIBI, que acaban con *_sim2*). Hay diferencias entre casos como el horizonte de control del MPC, la velocidad horizontal del AGV o el LAD que cambian a menudo, por lo que es recomendable ejecutar el archivo y esperar a que se realicen todas las simulaciones, en vez de iniciar solo una parte.

Lo único que hay que revisar es que el tiempo de simulación sea el adecuado antes de lanzar cualquier script. Con 25 segundos basta para todas las simulaciones incluidas en la memoria.

Los archivos están programados para hacer automáticamente las simulaciones con Vx = 8 y 12 m/s, con LAD = 5 y 10 m.

La información recopilada de las simulaciones se guarda en archivos .mat con la nomenclatura como la del siguiente ejemplo:

caso15IKIv12_2.mat

caso1: Nombre del caso.

5/10 : LAD.

IKI/MPC : Controlador.

v12 / : Se añade si la velocidad son 12 m/s.

_2/ : Se añade si el método del controlador IKIBI es 2.

Las gráficas se obtienen con los códigos que tienen el mismo nombre que los de las simulaciones, pero con la terminación *_graf*. El de las gráficas con el método 2 del controlador IKIBI acaban con *_graf2*.

Para cambiar entre los métodos 1 y 2, es el usuario quien debe desconectar el bloque correspondiente al antiguo método y conectar el nuevo, con ayuda de las Figura 39 y Figura 40, en las páginas 43 y 44 de la memoria.

En cuanto a las tablas de resultados, resulta tedioso llamar todos los índices desde la ventana de comandos, por lo que se ha hallado una alternativa. *tablas.m* crea las tablas en MATLAB en un formato compatible con Excel, y las envía a la aplicación. Cada una a un cuaderno de Excel *.xlsx* distinto.

Todos los archivos nuevos se crean en el mismo directorio en el que ubiquen los archivos principales. Desde Excel, es sencillo copiar los índices y pegarlos directamente en tablas de Word.