Trabajo Fin de Máster

Máster Universitario en Ingeniería en Electrónica, Robótica y Automática

Control Mediante Aprendizaje Iterativo

Autor: Antón Casas Román

Tutor: José Ángel Acosta Rodríguez

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla. 2020

Trabajo Fin de Máster
Máster Universitario en Ingeniería en Electrónica, Robótica y Automática

Control Mediante Aprendizaje Iterativo

Autor:
Antón Casas Román

Tutor:
José Ángel Acosta Rodríguez
Profesor titular

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Resumen: En este trabajo se presentan dos métodos de control en bucle abierto por aprendizaje iterativo centrados en procesos de optimización. Dada una trayectoria de referencia, el control mejora en cada ejecución el desempeño del sistema mediante el uso de los datos recogidos en ejecuciones anteriores. Los sistemas de control utilizan métodos óptimos tanto en la estimación del error en las trayectorias como en el cálculo de las señales de entrada, e incluyen restricciones en las entradas y en las salidas para este segundo proceso. Se han implementado dos procesos de obtención de modelos lineales a partir de sistemas reales, uno de ellos deducido en su totalidad de las ecuaciones de equilibrio dinámico del sistema, y el otro basado en técnicas de identificación de sistemas. Se han probado ambos algoritmos de control sobre el modelo simulable de un péndulo sobre un carro móvil, que es un sistema altamente no lineal y subactuado. En ambos casos el sistema de control aprende satisfactoriamente y consigue realizar el seguimiento de trayectoria en bucle abierto. Por último, se ha realizado una comparación de los dos métodos de control.

Palabras clave: Control mediante aprendizaje iterativo, seguimiento de trayectoria, optimización convexa, péndulo invertido, filtro de Kalman, linealización de sistemas, identificación de sistemas.

Abstract: In this work we present two methods of open-loop optimization-based iterative learning control. Given a reference trajectory, every execution the control algorithm improves the performance of the system by using the data collected in previous executions. The control systems use optimal methods both in the estimation of error in the trajectories and in the calculation of input signals, and include restrictions on inputs and outputs for this later process. Two processes have been implemented to obtain linear models from real systems, one of them deduced entirely from the dynamic equilibrium equations of the system, and the other based on system identification techniques. Both control algorithms have been tested on the simulated model of a pendulum on a moving car, which is a highly nonlinear and underactuated system. In both cases the control system learns successfully and manages to perform open-loop trajectory tracking. Finally, a comparison of the two control methods is made.

Keywords: Iterative learning control, trajectory tracking, convex optimization, inverted pendulum, Kalman filtering, system linearization, system identification.

Agradecimientos

A Cristina, Carlos, Sergi, Samuel y toda la peñita que hizo que la vida en Sevilla fuera maravilla.

A Álvaro Galán, por interesarse tanto por el trabajo y ayudarme con sus correcciones.

A Alexandra Elbakyan, por su labor hacia el mundo y la investigación.

A la comunidad de desarrollo de software libre.

Al movimiento antiespecista.

A la gente buena.

A Belén Oliva, por darme la idea de la siguiente sección.

Desagradecimientos

A Belén Oliva, por supuesto.

A la gente que presiona con los estudios y el trabajo como si fuera lo único importante.

A los que echan la culpa de todo a los jóvenes™.

Al coronavirus, supongo.

A todos los caseros.

Al capitalismo.

Índice

Índice de figuras

Índice de tablas

1 Introducción

Los sistemas automáticos están presentes en un rango de aplicaciones muy grande: desde control de temperatura y sistemas de ascensores en edificios, hasta centrales de generación de energía, pasando, por supuesto, por toda la industria química, las líneas de ensamblaje, y vehículos de todo tipo. Todos los sistemas automáticos requieren algún tipo de control que les permita funcionar adecuadamente.

A lo largo de la historia, conforme avanzaba la tecnología, y especialmente ligado al fenómeno de la industrialización, más y más procesos se han ido automatizando, librando al ser humano de tareas tediosas o repetitivas convirtiéndolas en trabajos de supervisión. Durante las últimas décadas en particular, los sistemas automáticos, y, por ende, los sistemas de control, han progresado mucho, adquiriendo complejidad, con el fin de realizan tareas cada vez más complicadas. Este avance presenta retos, que deben superarse mediante el desarrollo de estrategias de control más sofisticadas e inteligentes.

Los métodos de control más tradicionales toman lecturas de las salidas de un sistema e intentan controlarlas ajustándolas a una referencia. El hecho de tomar medidas permite al controlador reaccionar ante perturbaciones e imprevistos. Sin embargo, la señal de control no suele ser óptima en términos de rapidez, robustez, estabilidad, etc.

El control predictivo, del cual hay muchas variantes, realiza un cálculo en cada momento para encontrar la entrada óptima para el sistema, que haga que este se comporte de la mejor manera posible. La exactitud del resultado dependerá de la precisión con la que conozcamos tanto el estado como el comportamiento dinámico del sistema. Por otra parte, estos cálculos, que entran dentro del área de la optimización matemática, no son triviales, y pueden requerir una gran cantidad de potencia computacional. Debido a ello, comenzaron aplicándose en procesos muy lentos, como los de la industria química.

En la actualidad existen microprocesadores y procesadores digitales de señales muy rápidos y pequeños, que han permitido implementar este tipo de control en tiempo real en situaciones de dinámica rápida, como en sistemas de control de coches. Aún así, hay sistemas demasiado rápidos para controlarse en tiempo real con este tipo de métodos, como drones multirrotor.

Para controlar de forma óptima este tipo de sistemas, se ha desarrollado una forma de control *offline* (que no se ejecuta en tiempo real) que permite encontrar la entrada idónea para que un sistema siga una trayectoria determinada.

1.1 Antecedentes y estado actual de la tecnología

Podríamos considerar que el control por realimentación tiene una historia prácticamente tan antigua como la propia vida, ya que la conducta de la mayoría de los organismos vivos se basa en la respuesta a estímulos externos. Sin embargo, la capacidad de proporcionar esta autonomía a artificios y mecanismos es mucho más reciente.

1.1.1 Inicios de la ingeniería de control

Las primeras apariciones de sistemas de control suelen atribuirse a la antigua Grecia, y datan del siglo IV a. e. c. [1] Los sistemas eran puramente mecánicos, y sus principios de funcionamiento relativamente simples, siendo este el agua en muchas ocasiones. Un ejemplo es un despertador diseñado por Platón para evitar que los alumnos de su academia tuvieran problemas para levantarse puntualmente por la mañana. Este consistía en un depósito al que entraba agua a lo largo de la noche. Al llenarse, volcaba una pieza flotante que se encontraba en la parte superior del depósito y contenía unas bolas que rodaban y caían fuera, sobre el suelo o una bandeja metálica, provocando un estruendo. Mientras que este dispositivo funciona en bucle abierto, es el antiguo reloj de agua de Ctesibios en Alejandría, alrededor del siglo III a. e. c. el primer dispositivo de control mediante realimentación del que se tiene constancia. Medía el tiempo manteniendo constante el nivel de agua en un recipiente y, por lo tanto, el flujo de agua de ese recipiente.

Hacia el año 1200, H. U. Lansperg diseñó uno de los primeros reguladores de la historia [1]. Consistía en un molino que al girar golpeaba una rampa con pendiente muy pequeña que servía de alimentador para introducir el grano. Si el viento era fuerte, la rampa era golpeada con más frecuencia, haciendo caer más grano sobre la piedra solera.

En la segunda mitad del siglo XVIII, durante la primera revolución industrial, fue cuando los sistemas de control comenzaron a ser desarrollados como partes necesarias para los sistemas que controlaban, dejando de ser meros accesorios. La máquina de vapor desarrollada por Watt, por ejemplo, utilizaba un regulador centrífugo para controlar la velocidad de operación [2]. En 1868, en su artículo *On Governors*, James Clerk Maxwell explicó las inestabilidades presentadas por el regulador centrífugo usando ecuaciones diferenciales para describir el sistema de control. Este trabajo puede considerarse el origen de la teoría de control. Sin embargo, no tuvo mucha influencia sobre el futuro de la disciplina, que dependió en mayor medida de matemáticos como Laplace, Cauchy, Fourier y Liapunov, que desarrollaron la teoría necesaria [2].

1.1.2 Las teorías del control

Los sistemas de control fueron principalmente mecánicos hasta principios del siglo XX, cuando fueron superados por los sistemas hidráulicos [2]. Posteriormente, durante la primera mitad del mismo siglo, surgieron los controladores electrónicos.

La teoría clásica del control se consolidó durante la primera mitad del siglo XX. Minorsky, en 1922, en su trabajo *Directional Stability of Automatic Steered Bodies*, reconoce la no linealidad de algunos sistemas y aplica linealización. Nyquist, por su parte, presenta en 1932, en *Regeneration Theory*, su criterio de estabilidad, trabajando con amplificadores realimentados. Hazen, en 1934, en su trabajo *Theory of Servomechanism*, analiza el funcionamiento de algunos sistemas mediante las entradas escalón y rampa. Bode, en 1940, en *Relations Between Attenuation and phase in Feedback Amplifier Design* definió los conceptos de amplitud y fase y presentó sus diagramas logarítmicos. Por último, Ziegler y Nichols, en 1942, presentaron en *Optimum Settings for Automatic Controllers* su método empírico para sintonizar PIDs. En los años siguientes se realizó una enorme cantidad de avances, principalmente para ser aplicados a la ingeniería militar, durante la segunda guerra mundial [1].

Quedó entonces desarrollada la que llamamos la teoría clásica del control, que a día de hoy se sigue utilizando en aplicaciones simples, mediante el análisis en frecuencia y los controladores PID.

En los años 50, en la Unión Soviética, con el objetivo de mejorar el control para aplicaciones aeroespaciales, se desarrolla la representación en espacio de estados (o representación interna). Estos métodos, a través de una representación matemática de los sistemas más compleja, permitieron trabajar en el dominio del tiempo (frente al de la frecuencia), mejorar la rapidez y la estabilidad del control. Kalman definió los criterios de controlabilidad y observabilidad de los sistemas.

La teoría del control desarrollada en base al control en espacio de estados se llamó, entonces, teoría moderna del control. La representación en espacio de estados se sigue utilizando a día de hoy junto a muchos métodos de control, siendo, posiblemente, la más común.

1.1.3 El control por computador

Si hasta la década de los cincuenta los controladores electrónicos eran puramente analógicos, el desarrollo de la computación permite el surgimiento del control digital.

> "El desarrollo de la tecnología del computador aplicada al control de procesos industriales recibió a finales de los años cincuenta un gran impulso, debido a que existían industrias como las refinerías petrolíferas donde los procesos a controlar son complicados." [1]

La computación tiene una historia muy extensa, pero podemos considerar que los primeros computadores funcionales se construyeron durante los años cuarenta. Los computadores digitales para el control industrial se implementaron a finales de los años cincuenta. El primer caso es el de una refinería de Texas, que comenzó a funcionar controlada en bucle cerrado por un computador en 1959 [1].

El desarrollo de los microprocesadores en los años setenta y ochenta permitió la construcción de computadores más baratos que requerían mucha menos energía para funcionar. Esto, a su vez, posibilitó la expansión de los controladores digitales para una cantidad mucho mayor de procesos. Si bien los controladores analógicos se han seguido usando hasta la actualidad, estos se ven limitados por las posibilidades de la electrónica analógica, y desde los años ochenta, la mayoría de las nuevas técnicas de control se diseñan para implementarse sobre controladores digitales.

1.1.4 El control predictivo

El control predictivo surge en base al concepto de optimalidad, es decir, la existencia de un conjunto factible idóneo de estímulos de control para provocar en el sistema un comportamiento deseado. El principio de optimalidad fue formalizado por Bellman en el contexto de la programación dinámica, y dice así:

> "Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [3]

Esto es, simplificando y aplicando al control, que en cada instante de tiempo se plantea un problema de decisión en el controlador: ¿Qué valores de las señales de entrada (variables manipulables) son las que generan una respuesta óptima en la salida teniendo en cuenta el comportamiento del sistema y las perturbaciones medidas? Bellman plantea la posibilidad de descomponer el problema en distintos subproblemas más sencillos que, resueltos todos de forma óptima, en su conjunto plantean la solución óptima al problema [4].

Dentro del contexto del control, fue Norbert Wiener, en 1942, quien introdujo el concepto utilizar el modelo del sistema para realizar predicciones de su comportamiento [5]. Sin embargo, no llegó a desarrollarse hasta mucho más tarde.

J. Richalet et al. presentaron en 1978 en *Model predictive heuristic control* la primera aplicación de control predictivo [5]. Iba contenida en un software para el control de una unidad de destilación de petróleo. El método de control se conoce tanto por sus siglas MPHC como MAC (*Model Algorithmic Control*).

El algoritmo DMC (*Dynamic Matrix Control*) fue desarrollado durante los años setenta por Cuttler y Ramaker trabajando para Shell [5]. Fue ampliamente aceptado en la industria petroquímica y se usa a día de hoy. Además de ser un algoritmo de optimización capaz de tratar procesos multivariable, DMC cubre la identificación del modelo de la planta [6]. Los algoritmos del DMC y el MPHC son muy similares [6].

En 1986, García y Morshedi presentan en *Quadratic programming solution of dynamic matrix control (QDMC)* una formulación del MPC utilizando programación cuadrática, con lo que se consigue introducir restricciones en las entradas y en las salidas, simplificar el problema de optimización, y reducir el coste computacional [5].

En el ámbito académico, el GPC (*Generalized Predictive Control*) propuesto por Clarke et al. se ha convertido en uno de los métodos de control predictivo más populares [7]. Entre las ventajas frente a otros métodos se encuentran: el hecho de que proporciona soluciones analíticas en caso de estar libre de restricciones, y que puede controlar plantas inestables y de fase no mínima [8].

Todos los métodos citados de control predictivo se incluyen dentro del término MPC (*Model Predictive Control*) junto a muchos otros que no se han nombrado, tanto desarrollados entre medio como variantes posteriores.

> "Los algoritmos MPC tienen como característica común usar un modelo dinámico de la planta para predecir las futuras acciones de las variables manipuladas sobre las salidas, y difieren entre sí en el modelo usado para representar el proceso y el ruido, así como en los elementos de su función objetivo a ser minimizado." [5]

De forma genérica, un MPC plantea, para cada instante, un problema de optimización con un horizonte de predicción finito. Se resuelve el problema, utilizando el estado actual y la referencia a seguir, y obteniendo como resultado una secuencia de entradas óptima. De esta secuencia óptima el MPC aplica el primer valor, y espera al siguiente instante de tiempo para recibir la próxima medida del estado del sistema y realizar de nuevo el proceso entero.

En la figura 1 podemos ver un cronograma que describe el principio de funcionamiento de un MPC de forma genérica. A lo largo de todo la imagen podemos ver la referencia, que es la trayectoria que quiere que el sistema siga. En la parte izquierda del cronograma, en el pasado, observamos la evolución de la salida y las entradas introducidas al sistema. En la parte derecha del cronograma, en el futuro, observamos la señal entrada óptima calculada por el MPC en el momento actual y la salida que se espera conseguir con la susodicha señal de entrada. El MPC aplicará el primer valor de la señal calculada, y desechará el resto. En el siguiente instante de tiempo (k+1) volverá a realizar las medidas y el cálculo completo.

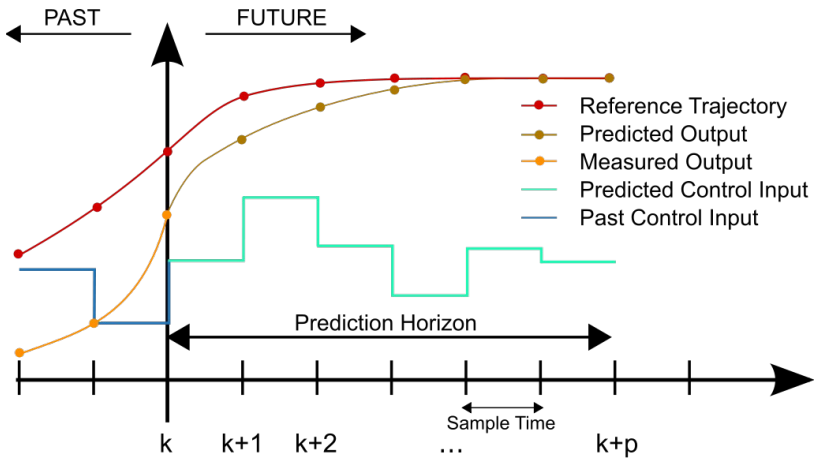

Figura 1: Principio de funcionamiento genérico de un MPC.
Imagen de Martin Behrendt [9]. CC BY-SA 3.0.

1.1.5 El control por aprendizaje iterativo

En 1984, Arimoto et al. presentaron en *Bettering Operation of Robots by Learning* la primera estrategia de control mediante aprendizaje iterativo (ILC, de *Iterative Learning Control*). El objetivo consistía en mejorar la ejecución de un trabajo realizado por un motor de corriente continua. Para ello, cada vez que se ejecutaba el experimento, se recogía la salida, y a través de un procesamiento matemático se calculaba una entrada de control más adecuada para que el motor se ajustara a la referencia pedida.

Desde entonces, el ILC se ha desarrollado en el ámbito teórico y se ha ido aplicando a sistemas más y más complejos. En [10], publicado en 2009, se implementa un ILC con programación convexa. En el mismo trabajo se consigue posicionar verticalmente un péndulo controlándolo únicamente en bucle abierto.

En [11], el mismo grupo de investigación publica un método para identificar un modelo dinámico al que se pueda aplicar el ILC en base a un modelo numérico de un sistema. En el trabajo citado realizan este proceso con un dron quadrotor.

En [12], una tercera publicación del mismo grupo, los investigadores aplican la estrategia con éxito a un dron quadrotor real.

Al igual que sus predecesores, los sistemas de control predictivo clásico (MPC) suelen regular el comportamiento de los sistemas dinámicos reaccionando al ruido y a las perturbaciones en la salida del sistema medido. Al basarse en un modelo matemático de la dinámica del sistema, el rendimiento está limitado por la precisión de este modelo y la causalidad de la acción de control, que sólo compensa las perturbaciones a medida que se producen. Los efectos desfavorables de

estas limitaciones se observan especialmente en los regímenes en los que la retroalimentación no es capaz de reaccionar a tiempo [12]. El ILC resuelve este problema tomando datos de experiencias anteriores para adelantarse a perturbaciones repetitivas y evitar su efecto.

1.2 Objetivo del trabajo

El objetivo general de este proyecto fin de máster es la aplicación de un esquema de control mediante aprendizaje iterativo.

Se desglosa en los siguientes objetivos parciales:

1. En primer lugar, la implementación en una simulación el método de control iterativo descrito en [10], [11] y [12]. Comprobar, simultáneamente, que la formulación descrita en los trabajos es capaz de funcionar, y conseguir que lo haga.
2. En segundo lugar, aplicar la estrategia de control al modelo simulable de un péndulo invertido, incluyendo la identificación del modelo dinámico.

1.3 Estructura del documento

El documento se divide en seis capítulos:

1. En este primer capítulo se hace una introducción al problema de ingeniería que se plantea, se expone una breve historia del control y de las técnicas relacionadas con la estrategia desarrollada, se define el estado actual de la técnica explorada y se hace un estudio de los trabajos preexistentes, se define el objetivo de este trabajo de fin de máster, y se describe, en este mismo apartado, la estructura del presente documento.
2. En el segundo capítulo se desarrolla el sistema de control, aplicado sobre un sistema arbitrario, utilizado como prueba para la efectividad y el funcionamiento de la implementación. El capítulo se divide en tres apartados: El primero trata la representación del sistema sobre el que se simula y al que se aplica la estrategia de control; el segundo explora la optimización cuadrática, utilizada para encontrar la solución a los problemas matemáticos planteados, así como el bucle sobre el que se obtienen las mejoras del esquema iterativo, que constituye la parte esencial del aprendizaje; y el tercero analiza las técnicas aplicadas para tratar con el ruido, necesarias en cualquier entorno real.
3. En el tercer capítulo se estudia la aplicación del sistema de control al péndulo sobre un carro móvil. Incluye la descripción e implementación del sistema a controlar, y la obtención de un modelo lineal con la finalidad de poder aplicar el sistema de control. Se analizan y aplican dos procesos diferentes de obtención de modelos lineales.

4. En el cuarto capítulo se comentan los resultados obtenidos al aplicar el sistema de control al dispositivo real descrito en el capítulo tercero. Se realizan dos experimentos, con dos métodos de control con la misma base, pero con algunas diferencias de concepto. Se comparan los resultados de ambos experimentos.
5. En el quinto capítulo se dan las conclusiones en base al trabajo realizado y a los resultados obtenidos en el capítulo cuarto.
6. En el sexto y último capítulo se presentan las referencias utilizadas para el desarrollo y la documentación del trabajo.

Junto a este documento, se presenta un anexo que incluye el código de las versiones finales de los programas desarrollados.

2 Sistema de control

Se ha implementado un controlador mediante aprendizaje iterativo basado en los trabajos de Angela Schöllig, Raffaello D'andrea, y Fabian L. Mueller. El sistema de control aparece descrito por primera vez en [10], y se utiliza y desarrolla también en [11] y [12].

Frente a los métodos de control habituales, basados en la realimentación de medidas en tiempo real, se ha desarrollado un sistema de control que actúa en bucle abierto en cada ejecución. Cuando se aplica una señal de entrada en bucle abierto, esta no se modifica frente al plan inicial durante la ejecución. Por tanto, para ejecutar el control en bucle abierto, debe utilizarse una señal de entrada, concebida *a priori*, con la que se espere conseguir una determinada salida deseada. Debido a que existen ruido y errores, causados por numerosas fuentes, a menudo imposibles o difíciles de tener en cuenta, se produce un error entre la señal de referencia y la de salida.

El sistema de control completo se puede dividir en un total de tres bloques o subsistemas que realizan tareas distintas. En la figura 2 se incluye un diagrama que muestra los tres bloques y sus interconexiones.

El bloque del sistema está constituido por el sistema real a controlar. En el ámbito de este trabajo se sustituye por un modelo que permite ser simulado. Este modelo puede ser de cualquier tipo, mientras se pueda simular y proporcione una salida satisfactoria para el uso que se le esté dando (e.g. modelo numérico).

Además del sistema real (o, como ya se ha dicho, modelo simulable), es necesaria una representación del mismo en forma de matrices de estados, variantes o invariantes en el tiempo, para poder aplicar la fase de optimización. Esta representación, debido a que difícilmente será perfecta, tendrá diferencias con el sistema real que generarán cierto error a la salida.

El bloque del estimador tiene la función de estimar el error repetitivo entre iteraciones, provocado por las incertidumbres en el modelo, ignorando el ruido aleatorio. Toma la salida medida directamente del sistema, que incluye ruido aleatorio, y trata de eliminarlo utilizando un modelo del sistema y la información de la señal de entrada. Como salida proporciona una estimación de la diferencia entre la salida y la referencia, habiendo aliviado el efecto del ruido. Este ruido eliminado por el estimador es el producido por errores en la medida, las limitaciones de los sensores, y otras fuentes que habitualmente generan ruido aleatorio. Es diferente del error repetitivo, nombrado en el párrafo anterior, que provocarán, principalmente las diferencias entre el sistema real y el modelo.

El bloque de control, mediante una referencia dada, y el modelo del sistema, formula y resuelve un problema de optimización con el que se obtiene una señal de entrada, que al aplicarse al sistema, debería hacer que este se comportara siguiendo la referencia proporcionada. En [10], [11] y [12], la referencia que se introduce en el problema de optimización es el error que se ha producido en la última iteración. De esta forma se obtiene una entrada que, restada a la utilizada en la iteración anterior, serviría para neutralizar los errores. En este trabajo se ha desarrollado adicionalmente otro enfoque, en el que al problema de optimización se le introduce la referencia absoluta que deben seguir las salidas, modificada mediante el error, para obtener la señal de entrada al sistema de forma absoluta, y no relativa a la usada previamente.

La señal de entrada obtenida se aplica al sistema en bucle abierto. Se mide la respuesta del sistema, pero esta no se usa para controlar el sistema en tiempo real a través de control por realimentación, sino que se almacena para procesarse en los bloques de estimación y control.

El proceso se repite de forma iterativa: Se aplica una señal de entrada, se mide el error, se actualiza el problema de optimización, y se repiten los cálculos para mejorar la señal de entrada. Así, es posible eliminar el error provocado por todas las fuentes de ruido que no cambian entre ejecuciones del experimento. Los errores totalmente aleatorios, sin embargo, no se eliminan con esta estrategia de control, ya que son distintos en cada ejecución y es imposible predecirlos. Para minimizarlos, se podrían utilizar controladores de bucle cerrado dentro del propio sistema, ya que son compatibles con el control iterativo descrito [12].

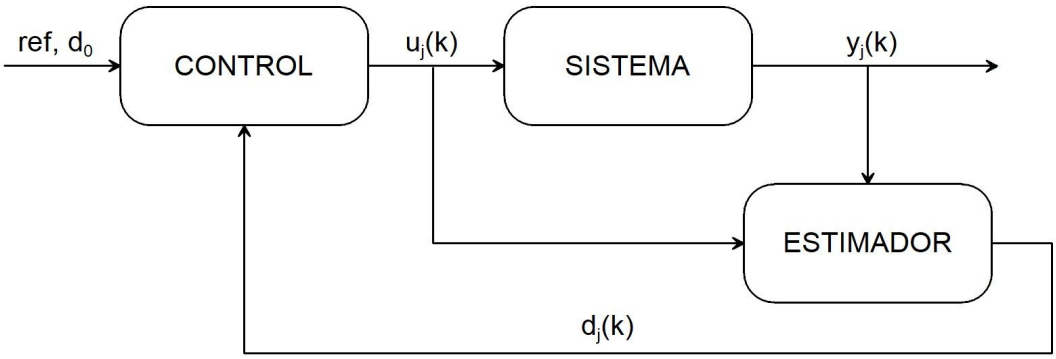

Figura 2: Diagrama de bloques del sistema de control. k hace referencia al número de muestra a lo largo de la simulación. j hace referencia a la iteración del bucle general.

En los apartados siguientes de este capítulo se describe la implementación del sistema de control, la cual se ha realizado íntegramente en Matlab. Por tanto, de manera ilustrativa se incluyen fragmentos de código en este lenguaje.

2.1 Definición del sistema controlado

El sistema real no tiene ningún requerimiento específico para poder ser controlado mediante este esquema ILC. A grandes rasgos, cualquier sistema sobre el que pudiera aplicarse un control predictivo tipo MPC, podría controlarse también con esta estrategia; añadiendo al conjunto los sistemas demasiado rápidos como para utilizar en ellos realimentación en tiempo real.

2.1.1 Representación del modelo dinámico

Es muy probable que a partir del sistema se pueda deducir un modelo que capture la esencia del comportamiento dinámico del sistema con la forma dada por las ecuaciones diferenciales, no lineales, variantes en el tiempo:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) \\ y(t) &= g(x(t), t) \end{aligned} \quad (1)$$

Donde x(t) denota los estados del sistema, u(t) las entradas, e y(t) las salidas.

$$\begin{aligned} x(t) &\in \mathbb{R}^{n_x} \\ u(t) &\in \mathbb{R}^{n_u} \\ y(t) &\in \mathbb{R}^{n_y} \end{aligned} \quad (2)$$

Dado que se pretende controlar el sistema a lo largo de una trayectoria específica; a partir de las ecuaciones 1, el comportamiento del sistema se puede aproximar en torno a la susodicha trayectoria mediante series de Taylor de primer orden, dando como resultado un sistema lineal variante en el tiempo [10]:

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t) \end{aligned} \quad (3)$$

Donde las matrices A, B, C y D son las matrices jacobianas variantes en el tiempo que definen el sistema.

Dado que tanto el sistema de control como el estimador funcionan en tiempo discreto, se debe discretizar también el modelo, para tener concordancia y que todo funcione adecuadamente. Así, las ecuaciones 3 se transforman en un sistema lineal discreto variante en el tiempo:

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= C(k)x(k) + D(k)u(k) \end{aligned} \quad (4)$$

Mientras que la deducción según las ecuaciones 1→3→4 es opcional, sí es absolutamente indispensable disponer de un sistema según la forma de las ecuaciones 4 para la aplicación del sistema de control.

2.1.2 Implementación

Con la finalidad de implementar y probar el método de control descrito previamente, se ha definido, arbitrariamente, un sistema de dos estados, dos entradas, y dos salidas.

Utilizamos la representación discreta en forma de matrices de estados A, B, C, y D. En primera instancia, las definimos invariantes en el tiempo.

$$\begin{aligned} x(k+1) &= Ax(k)+Bu(k) \\ y(k) &= Cx(k)+Du(k) \end{aligned} \quad (5)$$

Consideramos nula la matriz D, de forma que las entradas no tengan influencia directa sobre las salidas; y utilizamos como la matriz C la identidad, de forma que las salidas sean directamente los estados. Los valores dados son:

$$A=\begin{bmatrix} 0{,}9 & 0{,}01 \\ 0{,}02 & 0{,}85 \end{bmatrix}, \quad B=\begin{bmatrix} 1 & 0{,}1 \\ 0 & 1 \end{bmatrix}, \quad C=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6)$$

Se definen restricciones para los valores límite de las entradas, y a los valores límite de las salidas:

$$\begin{aligned} -0{,}15 \le u_1(k) \le 0{,}15 \\ -0{,}2 \le u_2(k) \le 0{,}2 \\ -0{,}05 \le y_1(k) \le 0{,}9 \\ -0{,}05 \le y_2(k) \le 1{,}05 \end{aligned} \quad (7)$$

Que se introducen en Matlab mediante:

```
A = [0.9, 0.01; 0.02, 0.85];
B = [1, 0.1; 0, 1];
C = eye(2);

U_max = [0.15; 0.2];
U_min = -1*[0.15; 0.2];

y_max = [0.9; 1.05];
y_min = -1*[0.05; 0.05];
```

Las referencias para las dos salidas se han definido también de manera arbitraria, y se pueden ver en la figura 3.

2.2 Optimización

El proceso de optimización se compone de una serie de definiciones iniciales; y un bucle donde todas las iteraciones realizan exactamente el mismo proceso, modificándose únicamente los valores de algunas de las variables. Para facilitar la explicación, se ha dividido este apartado en dos secciones: La primera referente a las definiciones, y la segunda referente al proceso iterativo

de optimización. A lo largo de ambos apartados se explican las diferencias de implementación entre aplicar el control sobre las referencias absolutas o sobre el vector de error únicamente.

2.2.1 Definiciones

En base a las matrices de descripción del sistema se calculan dos elementos esenciales para el proceso de optimización: La matriz F y el vector d. La primera describe la respuesta forzada del sistema, mientras que la segunda describe la respuesta libre del sistema (ante entradas nulas). La respuesta total del sistema se describirá mediante la suma de la respuesta libre y de la respuesta forzada.

$$x = Fu + d \quad (8)$$

Dado que nos encontramos en el instante inicial (al comienzo del experimento), y el vector d se irá actualizando conforme se realicen iteraciones, nombramos el segundo término de la ecuación 8 con el superíndice 0. Si además asumimos que se desean controlar todos los estados, podemos, opcionalmente, asumir que los estados son directamente las salidas, obteniendo la ecuación 9.

$$y = Fu + d^0 \quad (9)$$

La formulación simplificada de la ecuación 9 es la que encontramos en [12]. En [10] la formulación descrita es similar, pero consiste en dos ecuaciones:

$$\begin{aligned} x &= Fu + d^0 \\ y &= Gx + Hu \end{aligned} \quad (10)$$

Donde la matriz G define la relación entre los estados y las salidas, y la matriz H la relación directa entre las entradas y las salidas. La matriz G es análoga a la matriz C de la ecuación 4, y la matriz H es análoga a la matriz D de la misma ecuación.

En [11] también se formula en dos ecuaciones, pero con un grado de simplificación intermedio, ya que se omite el segundo término de la segunda ecuación de 10:

$$\begin{aligned} x &= Fu + d^0 \\ y &= Gx \end{aligned} \quad (11)$$

En última instancia, todas las formas de estas ecuaciones aluden a la misma estructura de representación, y se deben utilizar unas u otras dependiendo de la complejidad del sistema, que hará necesarios todos los términos, o permitirá omitir algunos.

1. El vector d describe la respuesta libre del sistema. Tiene tantas columnas como estados tiene la matriz A, y a lo largo de las filas se va desarrollando la evolución libre predicha. Su valor inicial se define de la siguiente manera:

$$d^0=[\tilde{x}_0, A_D(0)\tilde{x}_0, A_D(1)A_D(0)\tilde{x}_0, \ldots, \prod_{q=0}^{N-1} A_D(q)\tilde{x}_0]^T \quad (12)$$

Donde N es el número de muestras para el cual se quiere realizar el cálculo. Al multiplicarse sucesivamente el vector de estados inicial por la matriz A, se obtiene, en cada fila, el vector de estados predicho para cada instante de tiempo q. Esta forma es la dada por [10] para el mismo vector d^0.

El vector d se actualizará cada vez que se ejecute una iteración del experimento, y se encargará de retener la información que permite al sistema de control mejorar la eficacia de las señales (desarrollado en los apartados posteriores 2.2.2.2 y 2.3.2).

2. La matriz F describe la respuesta forzada del sistema. Es una matriz triangular inferior y se define, como aparece en [10], mediante:

$$F_{(l,m)}=\begin{Bmatrix} A_D(l-1)\ldots A_D(m+1)B_D(m) & si\ m<l-1 \\ B_D(m) & si\ m=l-1 \\ 0 & si\ m>l-1 \end{Bmatrix} \quad (13)$$

3. La matriz G describe la relación entre los estados y las salidas del sistema. Es una matriz diagonal por bloques y se define, como aparece en [10], mediante:

$$G_{(l,m)}=\begin{Bmatrix} C_D(l) & si\ \ l=m \\ 0 & en\,cualquier\,otro\,caso \end{Bmatrix} \quad (14)$$

4. La matriz H describe la relación directa entre las entradas y las salidas del sistema. Es una matriz diagonal por bloques y se define, como aparece en [10], mediante:

$$H_{(l,m)}=\begin{Bmatrix} D_D(l) & si\ \ l=m \\ 0 & en\,cualquier\,otro\,caso \end{Bmatrix} \quad (15)$$

Con estos elementos definidos, podemos definir el problema de optimización de la siguiente manera:

$$\min_{u_{j+1}} \|Fu_{j+1}+d_j-ref\|_l \quad (16)$$

Donde j indica el índice de iteración, l (el subíndice al final de la ecuación) indica el orden de la norma, y ref representa el vector de referencias. Si se prefiere, es posible calcular únicamente la variación en la entrada para contrarrestar el error, y dejar aparte el seguimiento de las referencias, desembocando en la forma dada por [10]:

$$\min_{u_{j+1}} \|Fu_{j+1}+d_j\|_l \quad (17)$$

La norma escogida afecta a la convergencia y al resultado [10], siendo las más comunes las normas 1, 2 e ∞. Elegimos la norma 2 o norma Eúclidea debido a que es la más natural, y a que permite formular la ecuación 16 (o 17) como un problema de optimización convexo. La norma Euclídea se minimiza con la siguiente ecuación [10]:

$$\min_{u_{j+1}} \ (Fu_{j+1}+d_j)^T(Fu_{j+1}+d_j) \quad (18)$$

2.2.2 Resolución del problema de optimización

Para resolver el problema de optimización se ha utilizado el comando quadprog de Matlab, que utiliza la sintaxis descrita a continuación:

```
u = quadprog(Fqp,dqp,Rc,b)
```

Que minimiza $\frac{1}{2}u^T F_{qp} u + d_{qp}^T u$ sujeto a las restricciones $R_c \cdot u \leq b$.

Por tanto, se deben transformar la ecuación 18 y las restricciones definidas en el apartado 2.1 para adaptarse a la forma requerida. Para ello se realizan las siguientes transformaciones de la ecuación 18:

$$\begin{gathered}\min_{u_{j+1}} \ (Fu_{j+1}+d_j)^T(Fu_{j+1}+d_j) \\ \min_{u_{j+1}} \ (u_{j+1}^T F^T + d_j^T)(Fu_{j+1}+d_j) \\ \min_{u_{j+1}} \ u_{j+1}^T F^T F u_{j+1} + u_{j+1}^T F^T d_j + d_j^T F u_{j+1} + d_j^T d_j\end{gathered} \quad (19)$$

De donde deducimos, separando los términos según el exponente del vector u que:

$$\begin{gathered}F_{qp} = 2F^T F \\ d_{qp} = 2d_j F\end{gathered} \quad (20)$$

La transformación de las restricciones es mucho más directa ya que las ecuaciones 21 implementan los límites superior e inferior para el vector de entradas; y las ecuaciones 22 implementan los límites superior e inferior para las salidas. Para aplicar todas al mismo tiempo basta con concatenar verticalmente las matrices y los vectores.

$$\begin{gathered}I \cdot u \leq u_{max} \\ -I \cdot u \leq -u_{min}\end{gathered} \quad (21)$$

$$\begin{gathered}F \cdot u \leq y_{max} - d_j \\ -F \cdot u \leq -y_{min} + d_j\end{gathered} \quad (22)$$

Donde u_{max} es un vector columna con los valores máximos permitidos para cada entrada e instante temporal, y donde el resto de constantes funcionan análogamente. En el caso de calcular

solamente la variación en la entrada para contrarrestar el error, debe modificarse la formulación de las ecuaciones 21 y 22, resultando en las ecuaciones 23 y 24. Así, las restricciones siguen refiriéndose a los valores absolutos de las señales, y no a la variación.

$$\begin{aligned} I \cdot u &\leq u_{max} - u_{j-1} \\ -I \cdot u &\leq -u_{min} + u_{j-1} \end{aligned} \quad (23)$$

$$\begin{aligned} F \cdot u &\leq y_{max} - y_{j-1} \\ -F \cdot u &\leq -y_{min} + y_{j-1} \end{aligned} \quad (24)$$

Se incluyen también dos matrices de pesos R y Q. Ambas son matrices diagonales cuyo tamaño depende respectivamente del número de entradas y el número de salidas, y ambas de la longitud del experimento.

- La matriz R se utiliza para dar más prioridad a algunas partes de la trayectoria, o a algunas salidas sobre otras. Un número mayor indica mayor importancia sobre los demás, mientras que un cero indica que no se tiene en cuenta.
- La matriz Q sirve para penalizar los valores altos en las entradas: Números mayores en Q resultarán en señales con valores más pequeños, mientras que una matriz Q de ceros no ponderaría las entradas.

A continuación se muestra el código de Matlab para el proceso explicado.

```
r  = [1; 1]; % Cuánto importa cada salida respecto de la otra
R_ = repmat(r, long, 1);
R_ = diag(R_);

q  = [1; 1]; % Penalización a las entradas
Q_ = repmat(q, long, 1);
Q_ = diag(Q_);

% transformar para quadprog
Fqp = 2*((F'*R_*F)+Q_);
Fqp = (Fqp+Fqp')/2; % forzar simetría
```

Y se reformulan las restricciones para darles la forma requerida por quadprog.

```
Rc = [];
b = [];

if uflag == 1
    Rc = [Rc; eye(nu*long); -1*eye(nu*long)];
    b = [b; repmat(u_max, long, 1); -1*repmat(u_min, long, 1)];
end

if yflag == 1
    Rc = [Rc; G*F; -1*G*F];
    b = [b; repmat(y_max, long, 1) - G*d(:,iteration); -1*repmat(y_min,
long, 1) + G*d(:,iteration)];
```

```
end
```

Mediante el comando quadprog se realiza el cálculo de la secuencia de entradas óptima para que la salida del sistema sea igual a la referencia. En el caso de querer calcular solamente la variación de la entrada para contrarrestar el error almacenado en el vector d, tan solo es necesario eliminar el vector de referencias w de la ecuación.

```
dqp = (2*(d(:,iteration) - w)'*R_*F)'; % w = vector de referencias

% Calcular señal de control:
temp = quadprog(Fqp, dqp, Rc, b);
```

2.2.2.1 Primera etapa

En caso de calcular solamente la variación de la entrada para contrarrestar el error, la primera señal de entrada que se aplica al sistema debe calcularse o estimarse *a priori*. Por tanto, en ese caso la primera ejecución se utiliza para medir las salidas y estimar el error con esa entrada. No se ejecuta la optimización hasta la segunda iteración.

En caso de calcular la entrada de forma absoluta para que el sistema siga las referencias, no es necesario estimar ninguna señal de entrada previa y la primera etapa optimización se realiza antes de la primera ejecución del experimento.

Se simula el modelo, sin ruido, y siendo exactamente el mismo sistema el simulado y el que se utiliza para el cálculo. Las dos salidas del sistema y sus respectivas referencias se muestran en la figura 3. En la figura 4 se muestran las dos entradas calculadas y utilizadas. Debido a que se trata de un caso ideal donde no hay ningún tipo de error ni de imprecisión: la trayectoria simulada se ajusta a la perfección a la referencia. El error cuadrático medio es muy pequeño en ambas salidas (del orden de 10e−6 y 10e−4 respectivamente).

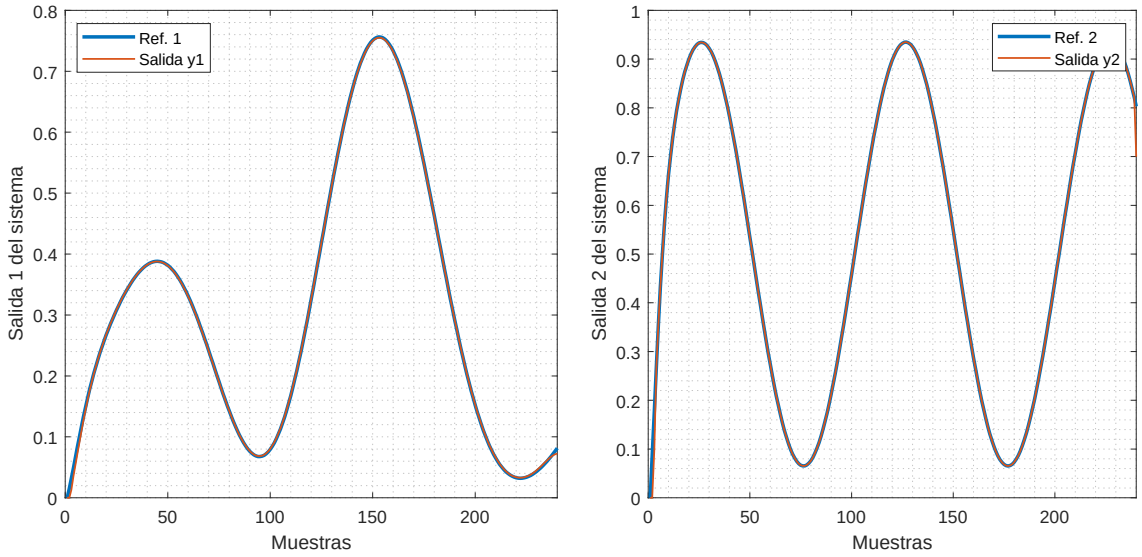

Figura 3: Salidas del sistema en el caso ideal.

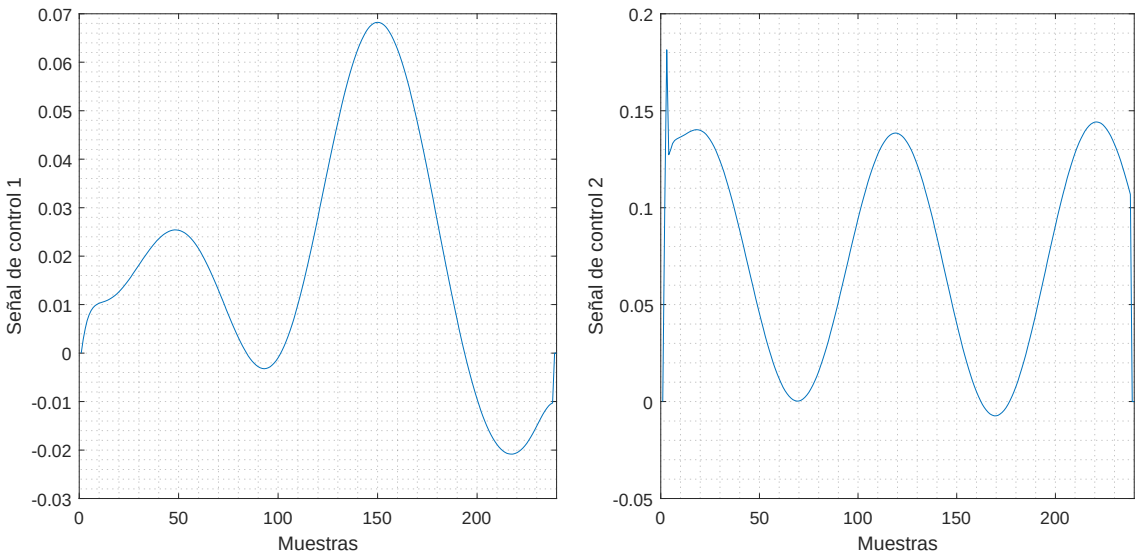

Figura 4: Entradas utilizadas en el caso ideal.

2.2.2.2 Etapas sucesivas

El caso ideal es muy conveniente a nivel teórico para demostrar el funcionamiento del cálculo de la entrada, pero, evidentemente, no es realista, ni aplicable a un caso real. Para acercar la simulación al asunto, modificamos la representación en matrices de espacio de estados para crear una disonancia entre el sistema sobre el que se calcula la entrada, y el sistema sobre el que se simula. Esta diferencia entre el sistema real y el modelado, provocada principalmente por dinámicas menores no modeladas, son la causa principal del error en las aplicaciones reales.

```
% Sistema estimado
A = [0.9, 0.01; 0.02, 0.85];
B = [1, 0.1; 0, 1];
C = eye(2);

% Sistema real
Ar = A+rand(2)*0.05;
Br = B+rand(2)*0.05;
Cr = C;
```

Ejecutamos el código para crear un sistema real nuevo, y comprobamos el funcionamiento. Los sistemas resultantes son los siguientes:

$$A=\begin{bmatrix}0{,}9 & 0{,}01\\ 0{,}02 & 0{,}85\end{bmatrix} \qquad B=\begin{bmatrix}1 & 0{,}1\\ 0 & 1\end{bmatrix}$$
$$A_r=\begin{bmatrix}0{,}9208 & 0{,}0516\\ 0{,}0621 & 0{,}8628\end{bmatrix} \qquad B_r=\begin{bmatrix}1{,}0307 & 0{,}127\\ 0{,}0291 & 1{,}0435\end{bmatrix} \tag{25}$$

En la figura 5 podemos observar como con esta diferencia entre los sistemas, las salidas se alejan considerablemente de la referencia.

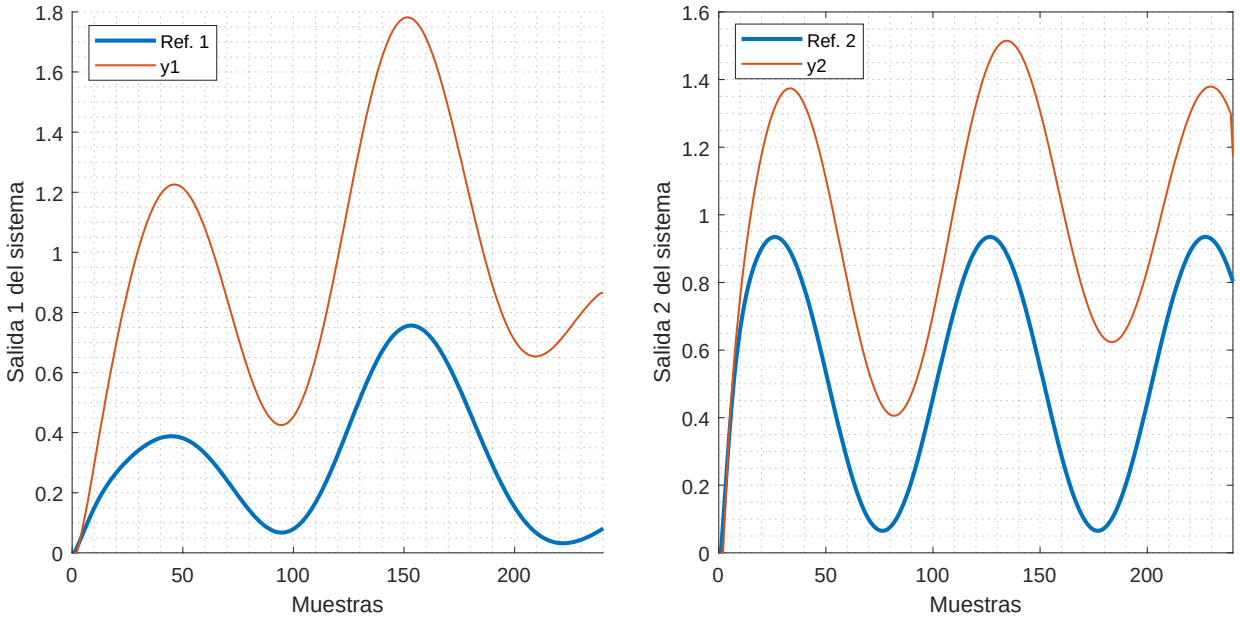

Figura 5: Salidas del sistema con un modelo inexacto.

Para solucionarlo, aplicamos la estrategia de control actualizando el vector d⁰ mediante el error obtenido en la primera ejecución. Para guardar los datos de todas las ejecuciones se ha decidido utilizar matrices tridimensionales, en las que, a lo largo de la tercera dimensión, se almacenan los datos de las iteraciones sucesivas.

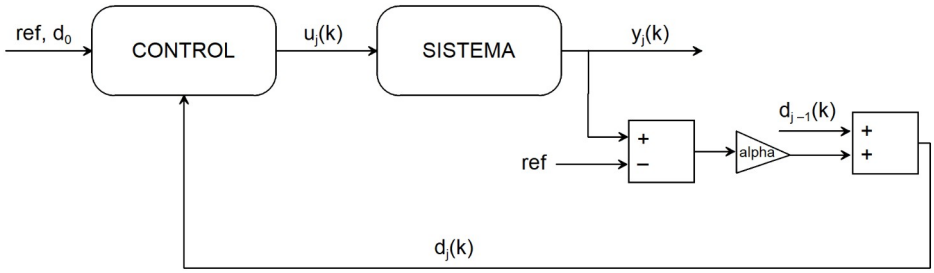

Figura 6: Diagrama de bloques del sistema sin estimador. Se actualiza el vector d de manera muy simple.

La actualización del vector d se realiza a través de un cálculo muy simple mostrado en el diagrama de la figura 6. Podríamos considerar este cálculo una versión primitiva del estimador que aparece en la figura 2, el cual se implementa propiamente en el apartado posterior 2.3.2.

Si trabajásemos calculando las variaciones sobre el vector de entrada para eliminar el error, la actualización del vector d se realizaría exactamente de la misma manera. El único cambio sobre la figura 6 sería que el bloque de control no utilizaría la señal de referencia, y que la señal u_j se sumaría a la entrada original u_1.

El error se calcula restando la salida y la referencia. El vector d se actualiza sumándole el error multiplicado por un factor *alpha*: Este factor puede ser una constante, como en el código mostrado a continuación, o un vector con el que se realizaría una multiplicación elemento a elemento. Además de actualizar del vector d, se calcula el error cuadrático medio para tener una medida numérica del error y poder comparar entre iteraciones.

```
% Medición de errores
for i = 1:ny
    % Cálculo del error en las trayectorias
    e(i,:) = (y(i,:,iteration) - ref(i,1:t_final));
    % Cálculo del error cuadrático medio
    ecm(iteration,i) = e(i,:)*e(i,:)'/t_final;
end

% Actualización del vector d
d(:, iteration + 1) = d(:, iteration) + alpha*e(3:end-2)';
```

En la figura 7 podemos ver la salida del mismo sistema de la figura 5 en las dos primeras iteraciones con un factor *alpha = 0,3*. Se aprecia a simple vista como la salida en la segunda iteración se aproxima mucho más a la referencia que la primera.

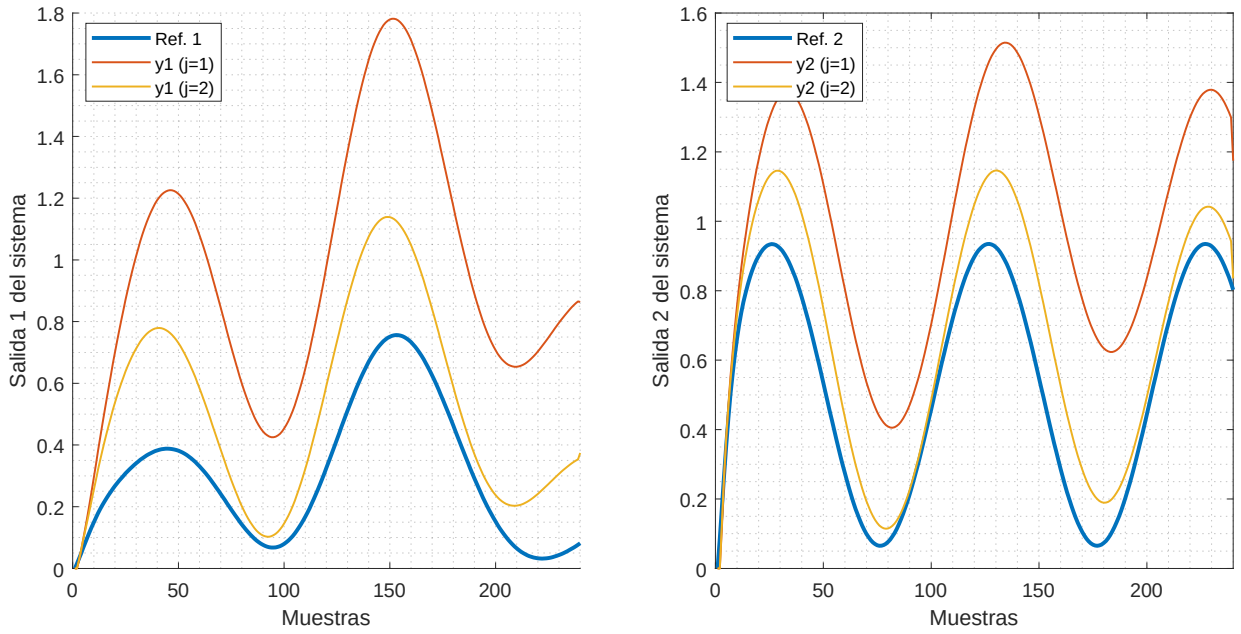

Figura 7: Dos iteraciones de la salida del sistema con un modelo inexacto.

Si repetimos el proceso un número mayor de iteraciones, la salida se acerca cada vez más a la referencia. En la figura 8 podemos ver las salidas para cinco iteraciones sobre el mismo sistema. Se observa como la salida se acerca progresivamente a la referencia.

También podemos comprobar la mejora a través del error cuadrático medio, mostrado en la tabla 1, en el que observamos un descenso paulatino de ambos errores.

Tabla 1: Error cuadrático medio de las dos salidas a lo largo de cinco iteraciones aplicando sucesivamente la optimización.

Iteración	Salida 1	Salida 2
1	0,4734	0,2327
2	0,0618	0,0244
3	0,0118	0,0032
4	0,0034	0,0007
5	0,0012	0,0003

En la figura 9 podemos ver la evolución de las entradas. Observamos que los cambios más grandes se producen en las primeras actualizaciones.

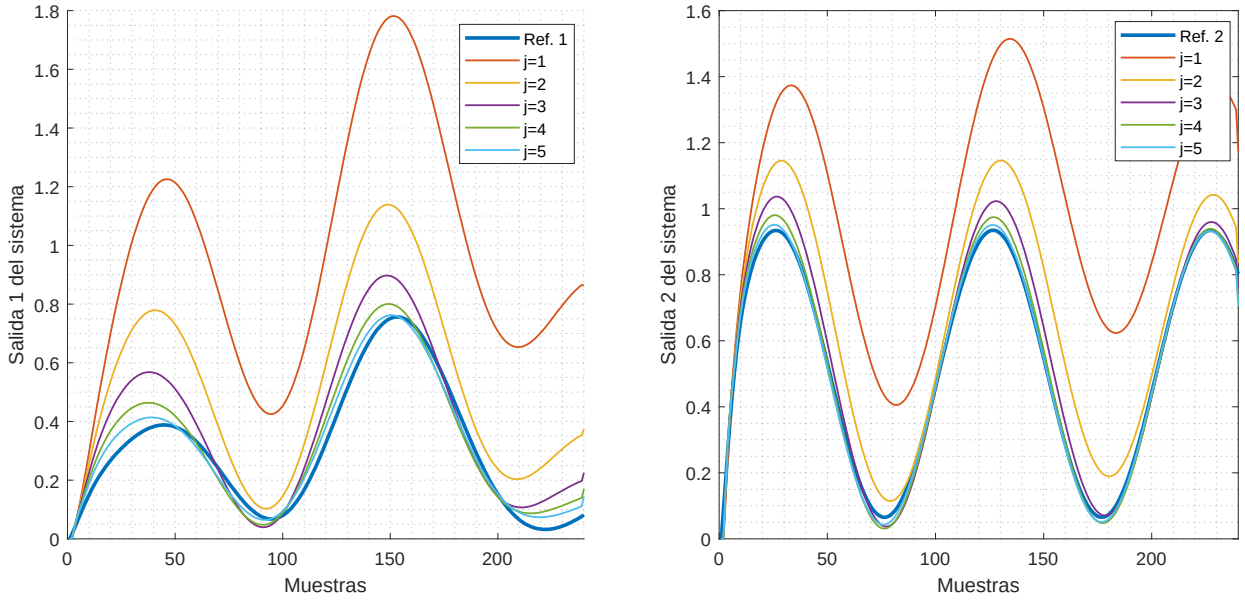

Figura 8: Salida del sistema con un modelo inexacto y 5 iteraciones.

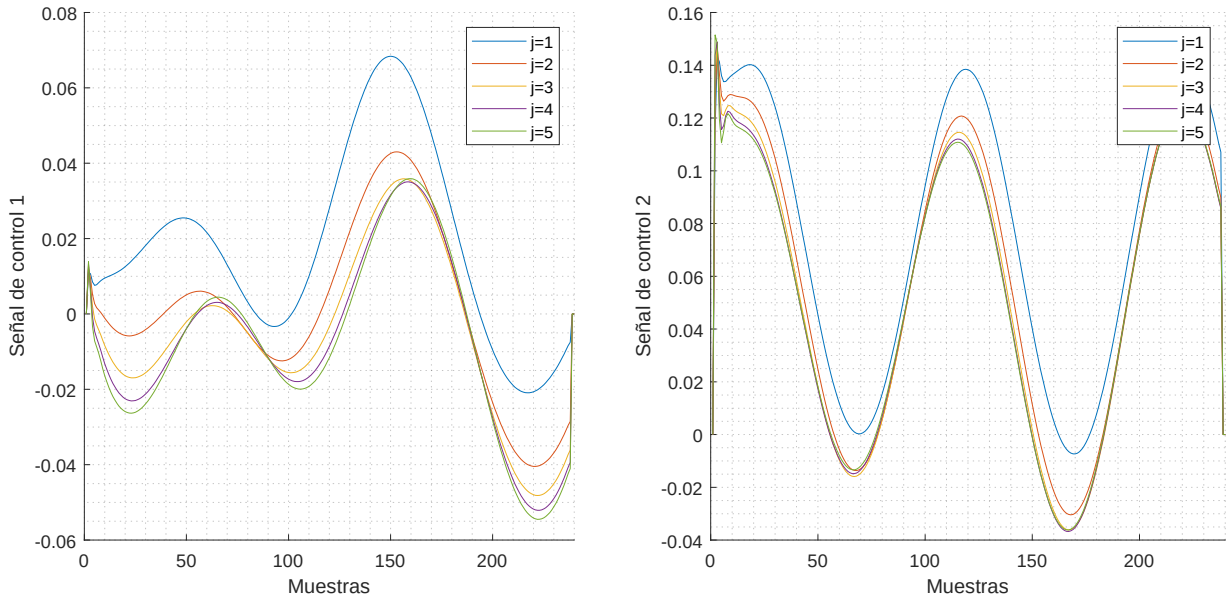

Figura 9: Entradas del sistema a lo largo de 5 iteraciones.

2.3 Filtro de ruido

Para acercar un poco más la simulación al comportamiento de un experimento real, añadiremos ruido al sistema junto a un filtro para intentar mitigar sus efectos.

En electrónica, el ruido es una perturbación no deseada en una señal eléctrica [13], es generado por muy diversas razones e innumerables fuentes, y suele estar siempre presente. Podríamos decir que un caso real en ingeniería de control no está completo si no tenemos en cuenta el ruido.

Para minimizarlo, utilizamos un filtro de Kalman, capaz de proporcionar una salida óptima e integrarse junto al proceso de optimización [10][11][12].

> "El filtrado de Kalman es un proceso de estimación de estados óptimo aplicado a un sistema dinámico que incluye perturbaciones aleatorias. Más precisamente, el filtro de Kalman proporciona un algoritmo recursivo lineal, no sesgado y con mínima variación de error para estimar óptimamente el estado desconocido de un sistema dinámico a partir de datos ruidosos tomados en tiempo real discreto." [14]

Para hacer el sistema más realista frente al apartado 2.2.2.2, aplicaremos ruido aditivo sobre la señal de entada al sistema y sobre la señal de salida. Ambas señales de ruido se ajustarán a una distribución normal y tendrán media cero, la desviación estándar se explorará probando varios valores para observar la variación en el comportamiento y los limites de funcionamiento. La asunción de que el ruido es Gaussiano (se ajusta a una distribución normal) es habitual, funciona bien en la mayoría de los casos, y, de hecho, es necesaria para que las ecuaciones principales del filtro de Kalman se puedan deducir (para el funcionamiento óptimo) [15].

Se han elaborado y puesto a prueba dos enfoques para el filtro de Kalman: Uno clásico, basado en muestras temporales, y estimaciones y predicciones realizadas para cada instante; y uno adaptado a la estructura del sistema de control, basado en iteraciones, que realiza *a posteriori* estimaciones para todos los instantes temporales por los que ha pasado cada variable.

2.3.1 Filtro de Kalman basado en el tiempo

El enfoque clásico del filtro de Kalman supone que el sistema se ejecuta en tiempo real, y trata de obtener una estimación del estado del sistema, en un momento preciso, en base a los datos obtenidos hasta ese momento. Esto proporciona una solución muy adecuada para la mayoría de situaciones, ya que controlar sistemas en tiempo real a través de un bucle cerrado de realimentación es lo más habitual en ingeniería de control. Sin embargo, esta estrategia no se ajusta a la estructura del problema presente.

Por una parte, al ejecutar el experimento completo y recoger todas las medidas antes de aplicar el filtrado, se podrían utilizar todas las muestras (incluidas las de muestras posteriores) para realizar la predicción de todos los instantes; algo que un filtro diseñado para funcionar en tiempo real

no contempla ya que estaría entrando en el dominio de la no causalidad. En consecuencia, no estaríamos aprovechando las posibilidades de trabajar con las medidas de todo el experimento.

Por otra parte, partimos de que existen diferencias considerables entre el modelo del sistema real y estimado. Si estas diferencias son grandes, el filtro de Kalman clásico no es capaz de dar buenos resultados, ya que el ajuste de la ganancia no puede cubrir adecuadamente el compromiso entre la fiabilidad del modelo y la de las medidas. Para demostrar esto, se ha implementado un filtro de Kalman variante en el tiempo con actualización automática de la ganancia sobre un sistema similar al descrito en el apartado 2.2.2.2. Se definen en el listado a continuación las ecuaciones que definen el filtro [16]:

1. Consideramos una planta discreta con ruido Gaussiano aditivo en la entrada *w[n]* y en la salida *v[n]*.

$$\begin{aligned} x[x+1] &= Ax[n]+Bu[n]+Gw[n] \\ y[n] &= Cx[n]+v[n] \end{aligned} \quad (26)$$

2. El filtro de Kalman constará de las ecuaciones de medida y actualización:

$$\begin{aligned} \hat{x}[n|n] &= \hat{x}[n|n-1]+M[n]\left(y[n]-C\hat{x}[n|n-1]\right) \\ M[n] &= P[n|n-1]C^T\left(R[n]+CP[n|n-1]C^T\right)^{-1} \\ P[n|n] &= \left(I-M[n]C\right)P[n|n-1] \end{aligned} \quad (27)$$

$$\begin{aligned} \hat{x}[n+1|n] &= A\hat{x}[n|n]+Bu[n] \\ P[n+1|n] &= AP[n|n]A^T+GQ[n]G^T \end{aligned} \quad (28)$$

   Donde R y Q son las matrices de covarianza de ruido y se definen:

$$\begin{aligned} Q[n] &= E\left(w[n]w[n]^T\right) \\ R[n] &= E\left(v[n]v[n]^T\right) \end{aligned} \quad (29)$$

   Siendo E la función que indica el valor esperado.

A continuación se muestra el código de Matlab que implementa las ecuaciones 27, 28 y 29 para un experimento ya ejecutado del que se tienen los vectores completos de la salida medida y de la entrada aplicada:

```
P = B*Q*B';       % Initial error covariance
xe = x(:,1);      % Estados iniciales
ye = zeros(t_final,ny); % Salida estimada

for i = 1:t_final
    % Actualización de las medidas
    Mn = P*C'/(C*P*C'+R);
    xe = xe + Mn*(yv(i,:)'-C*xe);  % x[n|n]
    P = (eye(nx)-Mn*C)*P;          % P[n|n]
```

```
        ye(i,:) = C*xe; % Salida estimada

        % Actualización de tiempo
        xe = A*xe + B*u(i,:,1)';       % x[n+1|n]
        P = A*P*A' + B*Q*B';      % P[n+1|n]
    end
```

Si para simplificar se elimina el error en la entrada y se utiliza el filtro de Kalman para reducir únicamente el error en la salida, podemos comprobar en las ecuaciones 27, 28 y 29 que el filtro confía íntegramente en el modelo e ignora las medidas. Esto no es muestra de un mal funcionamiento por parte del filtro, ya que para su utilización se presupone que se tiene un modelo relativamente bueno de la planta. Sin embargo, en nuestro sistema, donde se busca robustez frente a modelos "malos", nos encontramos con una limitación.

Se ha puesto a prueba la eficacia del filtro para una planta y un modelo ligeramente diferentes (similar a los modelos descritos en el apartado 2.2.2.2), ruido Gaussiano únicamente en la salida, y matrices de covarianza Q y R no nulas. De esta forma el filtrado en torno al ruido de entrada colaborará a reducir el error debido a la diferencia de modelos. En la figura 10 podemos observar como para valores pequeños de Q la salida se aleja mucho tanto de la real como de la medida, mientras que para valores grandes de Q la salida se acerca mucho a la señal ruidosa, sin eliminar apenas el ruido. En la tabla 2 podemos ver como el error cuadrático medio de las estimaciones tiene su mínimo con valores pequeños de Q, donde coincide con el de la medida. Concluimos por tanto que este tipo de filtro no funciona adecuadamente para este problema.

Tabla 2: Error cuadrático medio de la salida medida y las estimaciones frente a la salida real utilizando un filtro de Kalman convencional.

	Salida 1	Salida 2
Salida medida	0,0009	0,0010
Estimación Q = 0,0001	0,0260	0,0125
Estimación Q = 0,001	0,0020	0,0024
Estimación Q = 0,01	0,0009	0,0010

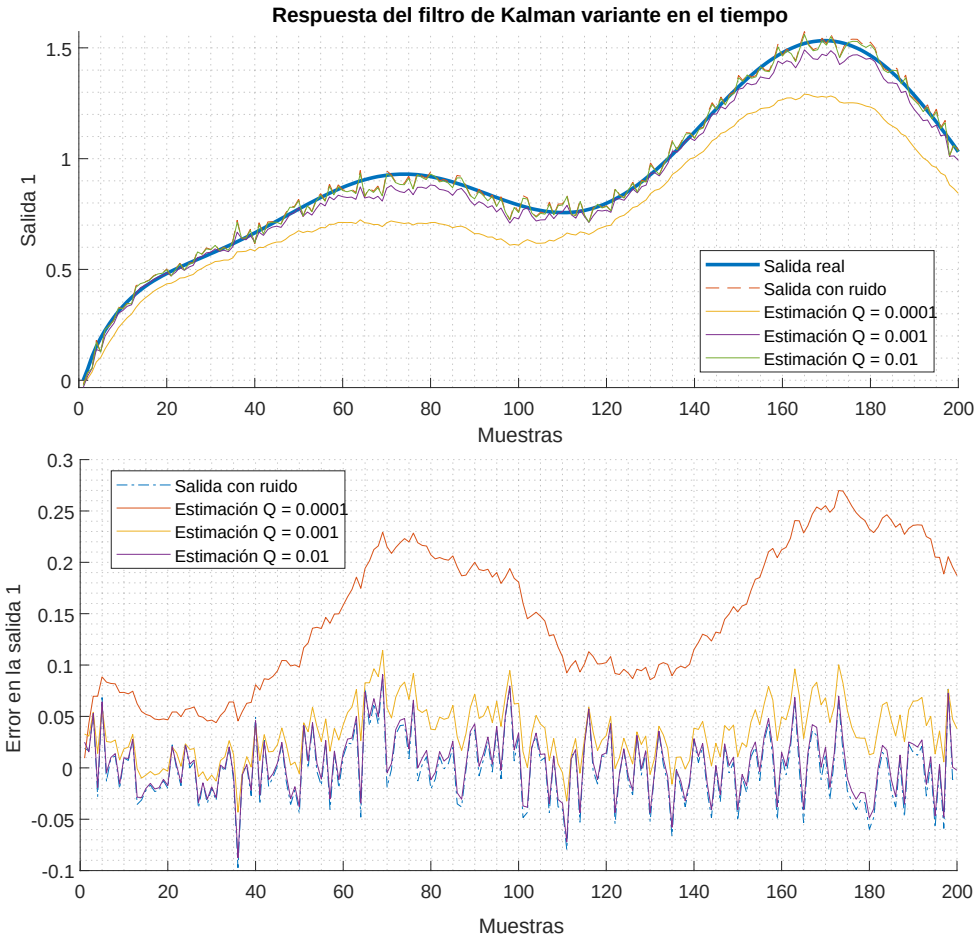

Figura 10: Estimaciones de la salida del sistema mediante un filtro de Kalman variante en el tiempo.

2.3.2 Filtro de Kalman basado en iteraciones

El filtro de Kalman propuesto para este proceso de control está construido sobre el dominio de las iteraciones, y retiene toda la información disponible de las ejecuciones previas del experimento [10][11][12]. La estructura, a grandes rasgos, es la misma que la del filtro clásico, pero los datos que trata son más amplios, las variables que utiliza son también mayores y, además, se ajustan a un origen del ruido diferente al del enfoque clásico.

El objetivo del filtro de Kalman en el dominio de las iteraciones es estimar el vector $\hat{d}_j$ definido en el apartado 2.2.1. Este vector cambia de una iteración a otra, y estos cambios se deben, principalmente, a las diferencias entre el modelo y el sistema real. Es la actualización de este vector, y la consiguiente actualización de la entrada $u_j(t)$, la que permite compensar por esos errores. Para un caso en el que exista una solución factible, y la entrada $u_j(t)$ converja, la secuencia d_j conver-

gerá también [10]. Durante la estimación, el filtro tendrá en cuenta, como es lógico, las fuentes de error en el sistema.

Se desarrolla el filtro a través de la siguiente serie de ecuaciones y pasos [10][12]:

1. La dinámica de la variable d_j a través de las iteraciones se define:

$$d_j = d_{j-1} + \omega_{j-1} \quad (30)$$

Donde podemos asumir que ω_j es una secuencia de ruido Gaussiano no correlacionada entre iteraciones.

$$\omega_j \sim N(0, \Omega_j) \quad (31)$$

Donde Ω_j es un parámetro de diseño que indica cuanto se espera que cambie d_j entre una iteración y la siguiente. Una posible definición es $\Omega_j = \epsilon_j I$, donde $\epsilon_j < \epsilon_{j-1}$, ya que queremos que el sistema converja.

Cada valor de la matriz Ω indica cuánto se quiere tener en cuenta el error en un instante de tiempo de cara al proceso de aprendizaje. Un cero indica que el error en ese instante no se tiene en cuenta, como si no hubiera ocurrido. Por el contrario, un error grande provoca una gran respuesta al error ocurrido en ese momento.

Los valores de la diagonal más cercanos a la esquina superior izquierda de la matriz hacen referencia a los instantes de tiempo más cercanos al inicio, mientras que los valores cercanos a la esquina inferior derecha hacen referencia a los instante de tiempo más cercanos al final. Los valores fuera de la diagonal hacen referencia al error correlacionado entre distintas salidas o instantes de tiempo.

2. Uniendo esto a las ecuaciones 10 de definición del sistema, obtenemos las ecuaciones del filtro:

$$\begin{aligned} d_j &= d_{j-1} + \omega_{j-1} \\ y_j &= G d_j + (GF + H) u_j + \mu_j \end{aligned} \quad (32)$$

Donde μ_j es una secuencia de ruido Gaussiano, totalmente independiente de ω_j, caracterizada por:

$$\mu_j \sim N(0, \mathbf{M}_j) \quad (33)$$

Al igual que Ω_j, M_j es un parámetro de diseño que debería coincidir con la covarianza de los errores del ruido del sistema y de los sensores utilizados. Sin embargo, no necesita ser muy preciso, y $\mathbf{M}_j = \eta_j I$ suele ser suficiente [12].

El ratio entre Ω_j y M_j será lo que especifique la relación entre la confianza el modelo del sistema y la confianza en las medidas. Debido a su definición, M_j no cambia mucho a lo largo de las iteraciones, por lo que puede permanecer constante. Al comienzo, es probable que partamos de un modelo malo, y conviene confiar más en las medidas. Conforme se ejecutan iteraciones y se actualiza el vector d, podemos aumentar la confianza en el modelo renovado, y reducir la matriz de covarianza Ω_j.

3. Definiendo también la varianza del error:

$$P_{j|j}=E[(d_j-\hat{d}_{j|j})(d_j-\hat{d}_{j|j})^T] \quad (34)$$

Para obtener el valor inicial de la matriz P utilizamos el cálculo de d_0 con la ecuación 12, pero necesitamos dar una estimación inicial del vector $\hat{d}_{j|j}$. Una estimación inicial razonable suele ser $\hat{d}_{j|j}=0$.

4. Obtenemos las ecuaciones del filtro de Kalman que calculan la ganancia óptima para nuestro planteamiento:

$$\begin{aligned} P_{j|j-1}&=P_{j-1|j-1}+\Omega_{j-1}\\ \Theta_j&=GP_{j|j-1}G^T+M_j\\ K_j&=P_{j|j-1}G^T\Theta_j^{-1}\\ P_{j|j}&=(I-K_{jG})P_{j|j-1} \end{aligned} \quad (35)$$

5. Una vez calculada la ganancia K_j, el vector $d_{j|j}$ se calcula mediante:

$$\hat{d}_{j|j}=\hat{d}_{j-1|j-1}+K_j(y_j-G\hat{d}_{j-1|j-1}-(GF+H)u_j) \quad (36)$$

A continuación se incluye el código de Matlab que implementa las ecuaciones 34, 35 y 36. Nótese que la actualización del vector d es diferente de la empleada en el apartado 2.2.2.2, aunque las dos tienen la misma finalidad.

```
Omega = diag(epsilon*ones(396,1));
...
P = d(:,1)*d(:,1)'; % Primera estimación de la matriz P (filtro de ruido)
...
% Actualización de la ganancia K
P = P + Omega;
Theta = G*P*G' + M_;
K = P*G'/Theta; % P*G'*inv(Theta)
P = ( eye(nx*(long)) - K*G )*P; % (I-KG)*P
Omega = Omega/10;

% Actualización del vector d
y_aux = y(:,:,iteration);
e_ = (y_aux(1:long*ny)' - G*F*temp);
d(:, iteration + 1) = d(:, iteration) + K*(e_ -G*d(:, iteration)); % d = d
+ K(y -G*d -(GF+H)*u
```

2.3.2.1 Aplicación del filtro

Se ha puesto a prueba el filtro para un sistema similar al utilizado en el apartado anterior (2.3.1), con ruido Gaussiano aplicado únicamente en la salida. Definimos inicialmente Ω como una matriz diagonal $\Omega_0=\epsilon_0 I$ con $\epsilon_0=0{,}02$, y en las iteraciones posteriores actualizamos de acuerdo con $\epsilon_j=\epsilon_{j-1}/10$. La matriz M_j se ha definido con la varianza del ruido Gaussiano aplicado y no se ha modificado. Se ha ejecutado el experimento con diez iteraciones para observar el aprendizaje.

En la figura 12 podemos ver cómo el sistema converge rápidamente, y a partir de la cuarta iteración los cambios en las salidas son casi imperceptibles. En la figura 13 podemos ver cómo las entradas también convergen adecuadamente. Presentan bastante rizado debido al ruido Gaussiano aditivo en la salida. Dependiendo de los actuadores y de las condiciones reales de la implementación, esto podría suponer un inconveniente. Para evitarlo, podría filtrarse la salida antes de aplicarla. En la figura 11 comprobamos, a través del error cuadrático medio, que el sistema de control converge completamente alrededor de la sexta iteración, y que a partir de allí los cambios son muy pequeños, debiéndose probablemente al ruido aleatorio. Por tanto, probamos que este filtro funciona muy bien con esta estructura de control, y que además se complementa perfectamente con la resolución del problema de optimización, convergiendo tan rápidamente como en el caso sin ruido (apartado 2.2.2.2).

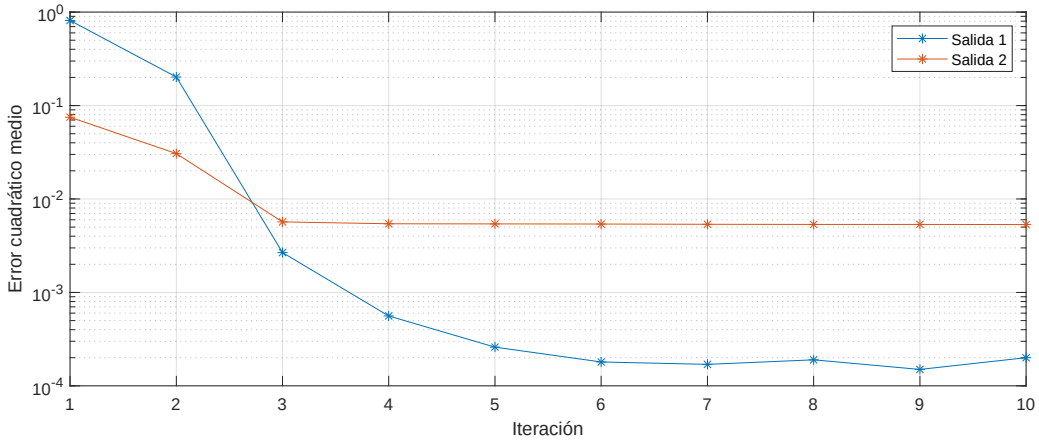

Figura 11: Evolución del error cuadrático medio de las salidas para cada iteración utilizando un filtro de Kalman basado en iteraciones.

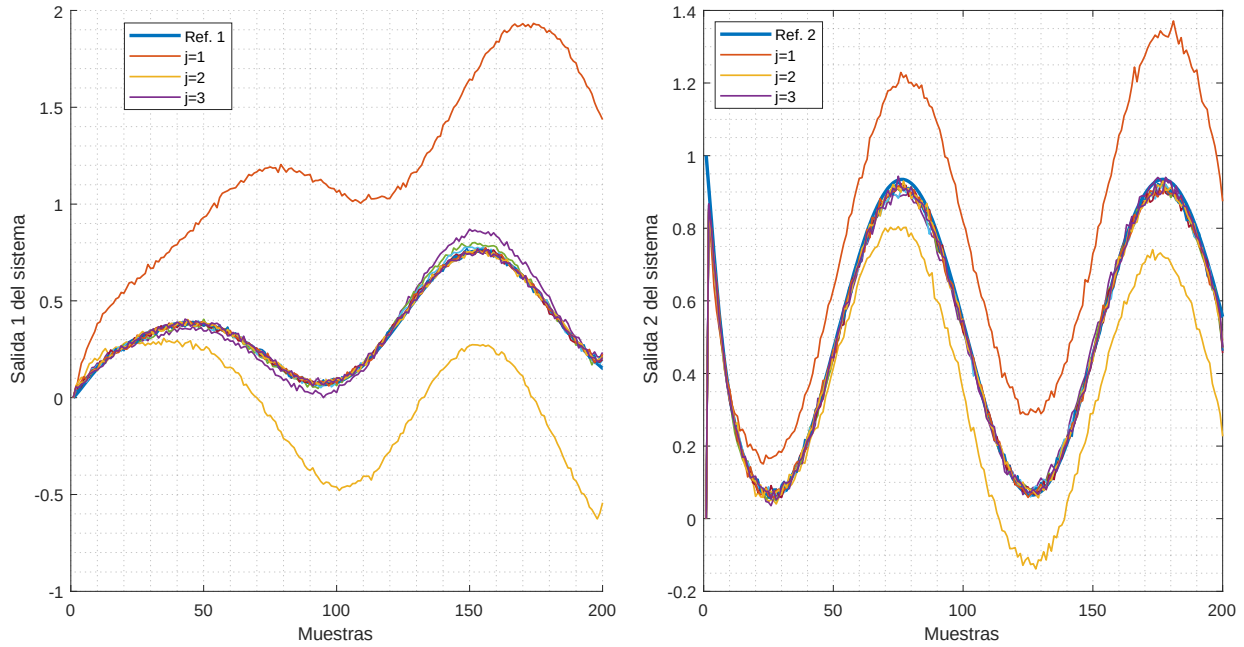

Figura 12: Salidas del sistema a lo largo de diez iteraciones con ruido y filtro de Kalman basado en iteraciones. A partir de la cuarta iteración las trayectorias se solapan, por lo que no se incluyen en la leyenda.

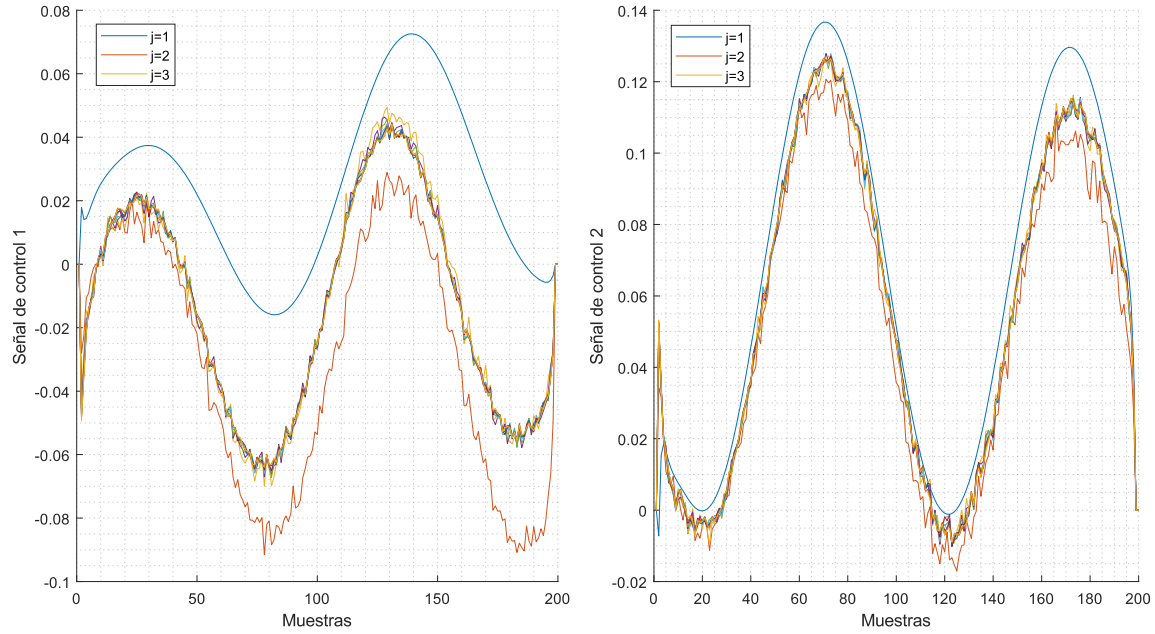

Figura 13: Entradas del sistema a lo largo de diez iteraciones, con ruido y filtro de Kalman basado en iteraciones. A partir de la cuarta iteración las trayectorias se solapan, por lo que no se incluyen en la leyenda.

2.3.2.2 Adición de todas las fuentes de ruido

Para completar el escenario añadimos ruido aditivo a la entrada (además de a la salida), y estimamos de forma inexacta la covarianza del ruido aditivo.

Simulamos un sistema similar al de los casos anteriores y observamos en la figura 15 que a pesar de la gran cantidad de ruido el sistema converge rápidamente. Comprobamos además en la figura 14, a través del error cuadrático medio, que converge en apenas tres iteraciones, y, a partir de entonces, el error oscila debido a la gran cantidad de ruido.

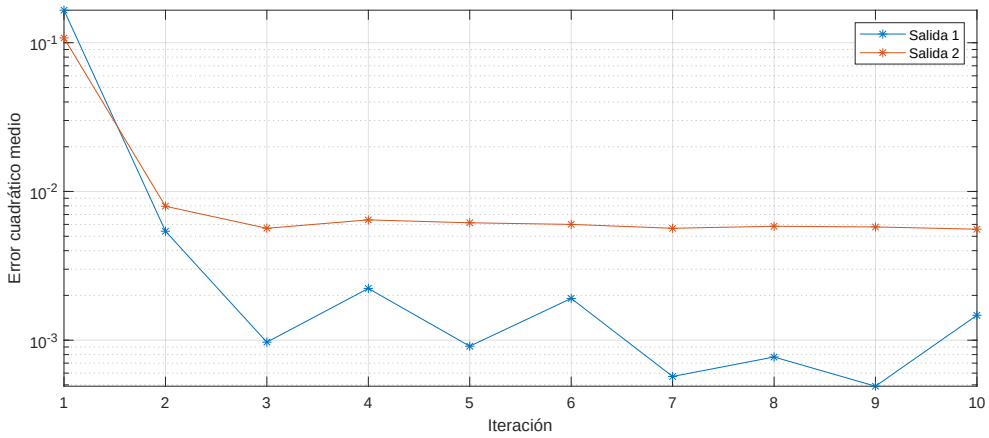

Figura 14: Evolución del error cuadrático medio de las salidas a lo largo de diez iteraciones con todas las fuentes de ruido y un filtro de Kalman basado en iteraciones.

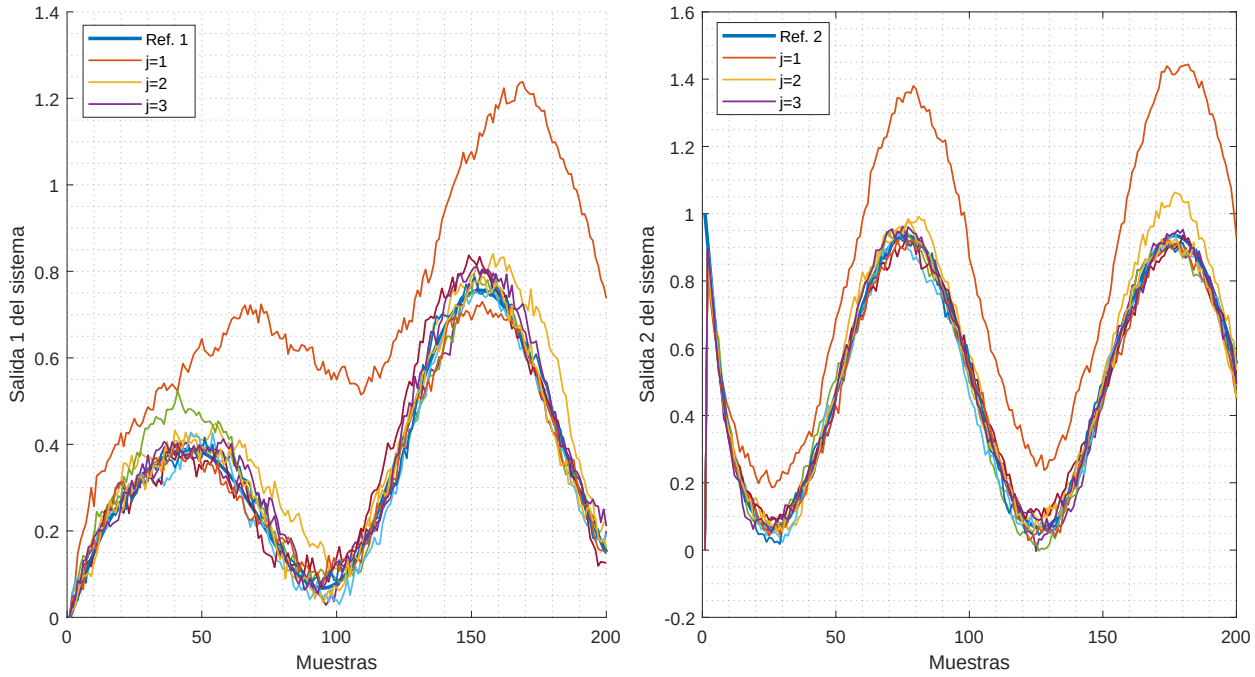

Figura 15: Salidas del sistema a lo largo de diez iteraciones con todas las fuentes de ruido y el filtro de Kalman basado en iteraciones. Las entradas solapadas no se muestran en la leyenda.

2.3.3 Filtrado de la señal de entrada

En situaciones muy ruidosas, como la simulada en el apartado anterior (2.3.2.2), puede ser conveniente hacer un filtrado de la señal de entrada para eliminar los cambios muy rápidos. Estos cambios muy rápidos, que podríamos llamar tirones o sacudidas, son artificios de alta frecuencia provocados por el ruido Gaussiano que se transmite de las medidas a las entradas a través del proceso de optimización. Si bien en un entorno de simulación estos tirones son totalmente inocuos, pueden provocar daños al aplicarse a dispositivos reales como motores eléctricos.

El objetivo de este filtrado es el suavizado de la señal de entrada. Para ello existen numerosas técnicas, como la media móvil [17], la media de las envolventes [18], los filtros paso bajo o los filtros Savitzky-Golay [18]. Para este proyecto se ha implementado un filtro de paso bajo tipo *Butterworth*, ya que es un filtro muy elemental, fácil de aplicar, y suficientemente bueno.

Un requerimiento importante a tener en cuenta es que el filtro debe no generar desfase en ninguna frecuencia. Para ello se ha realizado un filtrado digital de fase cero mediante la función *filtfilt* de Matlab. Esta función, después de filtrar los datos en la dirección natural, invierte la secuencia filtrada y la vuelve a pasar por el filtro. El resultado tiene distorsión de fase cero, y un orden de filtrado que es el doble del orden del filtro especificado [19]. En la figura 16, donde se superponen

una señal sin filtrar y la misma señal filtrada, podemos ver cómo el filtro respeta perfectamente la fase.

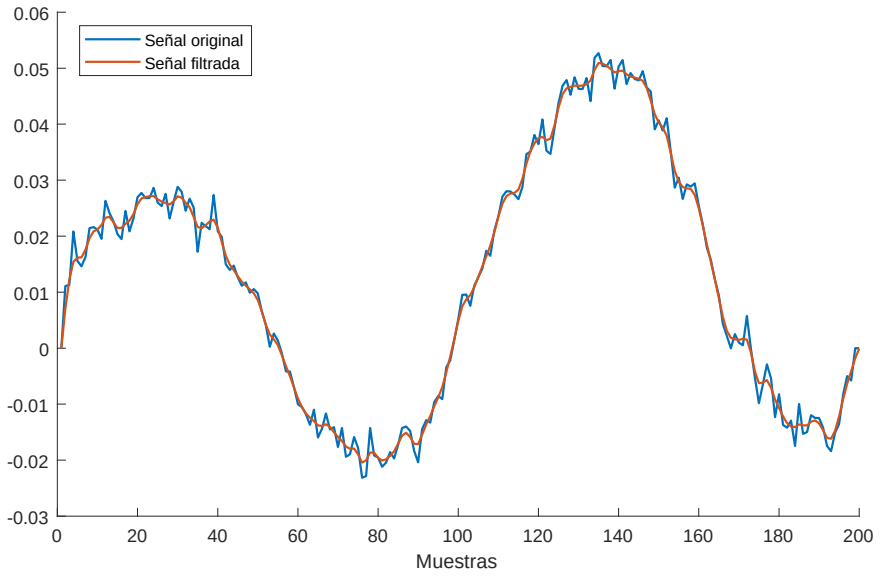

Figura 16: Señal sin filtrar y filtrada, sin distorsión de fase, por el filtro Butterworth paso bajo.

El filtro implementado es de orden uno (filtrará como si fuera de orden dos), y su banda de paso llega hasta la frecuencia normalizada 0,3. A continuación se incluye el código de Matlab que implementa el filtrado de las entradas.

```
[num,den] = butter(1,0.3); % Filtro butterworth de paso bajo de orden 1
for i = 1:nu
    u(i,:,iteration) = filtfilt(num,den,u(i,:,iteration)); % Filtrado de
fase 0 (orden 2*1)
end
```

Con la adición de este filtro el sistema converge muy similarmente a como lo hace sin él. Por otra parte: suaviza las señales de entada, como se puede ver en la figura 17, donde se representan estas, filtradas y sin filtrar; y reduce ligeramente el ruido en las salidas.

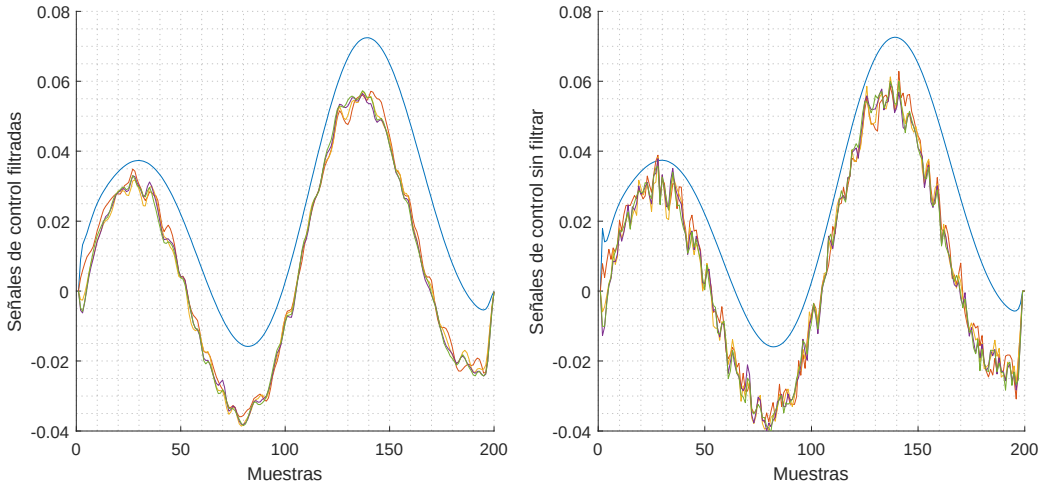

Figura 17: Entradas del sistema a lo largo de cinco iteraciones. A la izquierda las señales se han filtrado, a la derecha no.

2.4 Horizonte extensible

De manera adicional a los tres elementos principales del sistema de control, se puede añadir, opcionalmente, una estrategia para hacer más seguras las ejecuciones y mejorar la convergencia [12].

> "Durante la ejecución, la trayectoria real puede comenzar a desviarse de la trayectoria de referencia; si la desviación excede un determinado límite (especificado a través de una condición de terminación), la ejecución se detiene. En este caso el algoritmo de aprendizaje considera solo la parte de la ejecución que se ha realizado y devuelve una entrada actualizada para esa primera parte de la trayectoria. [...] Se ejecuta una nueva iteración. En general, la siguiente iteración funciona mejor durante el segmento inicial de la trayectoria y termina en un instante posterior. Es decir, el horizonte temporal del algoritmo de aprendizaje se amplía gradualmente." [12]

La condición de terminación se define mediante una función dependiente de las trayectorias medidas y de las referencias:

$$h(y(k), ref(k)) \geq \gamma(k) \quad (37)$$

Donde la función *h* realiza una transformación al dominio de las variables de terminación γ. La ejecución de una iteración se detiene en el instante N_j en el que se cumple la inecuación 37.

La variable γ puede variar a lo largo de la trayectoria, la elección de sus valores es crucial para la convergencia del proceso de aprendizaje [12]. Valores de γ muy pequeños pueden hacer que las iteraciones siempre se detengan prematuramente y nunca alcancen la duración nominal, mien-

tras que valores muy grandes pueden hacer que la optimización trabaje en zonas donde el comportamiento del modelo lineal se aleja mucho del del sistema real [12].

De cara al proceso de optimización, el vector de error d_j debe tener la estructura siguiente para una iteración detenida por la condición de terminación:

$$d_j(m)=\begin{Bmatrix} d_j(m) & para\, m\in\{1,\dots,N_j\} \\ d_0(m) & para\, m\in\{N_j,N_0\} \end{Bmatrix} \quad (38)$$

Donde N_0 es la duración nominal de las iteraciones. Que resultaría en una entrada con la estructura siguiente:

$$u_{j+1}(m)=\begin{Bmatrix} u_{j+1}(m) & para\, m\in\{1,\dots,N_j\} \\ u_1(m) & para\, m\in\{N_j,N_0\} \end{Bmatrix} \quad (39)$$

Donde u_1 es la entrada utilizada en la primera ejecución del experimento. Para conseguirlo, ajustamos los valores de la matriz Ω, de forma que la optimización solo actúe sobre los instantes temporales que nos interesan:

$$\Omega_j(l,m)=\begin{Bmatrix} \Omega_j(l,m) & para\ \ l,m\in\{1,\dots,N_j\} \\ 0 & en\, cualquier\, otro\, caso \end{Bmatrix} \quad (40)$$

Esto se ha implementado en Matlab con el siguiente código:

```
% Extensión de horizonte
llimit = find( abs(e(1,:)') > 1 ,1); % Momento en que el error se hace
demasiado grande
if isempty(llimit), llimit = long;
elseif llimit > long, llimit = long; end
% Re-creación de la matriz Omega
Omega = diag( Omega(1)*[ones(nx*llimit,1); zeros(nx*(long-llimit),1)] );
```

Donde en la primera línea se encuentra el instante N_j, y en la última se actualiza la matriz Ω.

Es importante destacar que en el ámbito de este trabajo se ha implementado la extensión de horizonte con el único fin de mejorar la efectividad del aprendizaje, dado que, al trabajar en simulación, no nos preocupa la seguridad del dispositivo en la ejecución del experimento.

3 Aplicación a un sistema real

Con la finalidad de evaluar el sistema desarrollado en un ambiente más realista, se ha implementado un péndulo sobre un carro que se puede mover horizontalmente. El sistema, que puede verse en las figuras 18 y 19, tiene una sola entrada de control, que ejerce una fuerza sobre el carro en la dirección en la que este puede moverse.

El objetivo del experimento es conseguir colocar el péndulo hacia arriba, partiendo de una posición de reposo en la que el péndulo se encuentra abajo. La principal complicación que presenta este sistema es que es altamente no lineal, las aproximaciones lineales funcionan con precisión en rangos muy pequeños de movimiento, y la trayectoria deseada abarca el rango completo de la no linealidad.

> "El frecuentemente citado problema de balancear un péndulo hasta su posición vertical representa una referencia que implica el mantenimiento del equilibrio. [...] El gran número de publicaciones sobre el péndulo invertido muestra la importancia y el amplio interés que suscita este sistema altamente no lineal y poco controlable." [10]

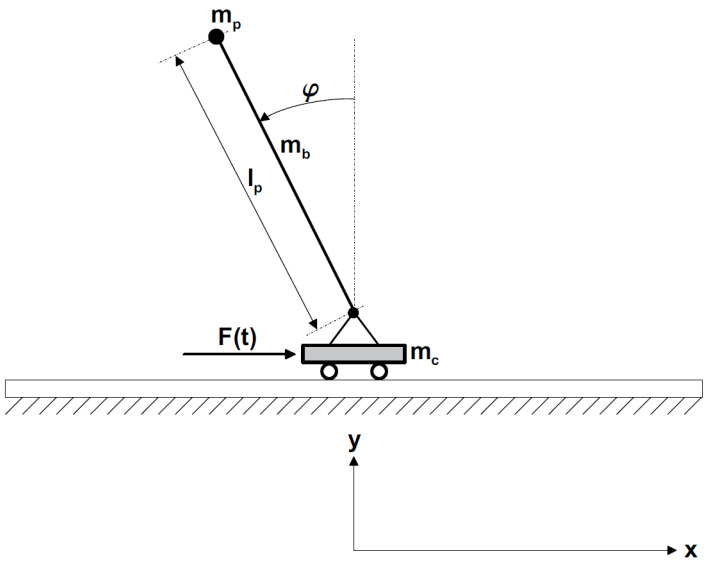

Figura 18: Esquemático del péndulo móvil.

3.1 Implementación experimental

El sistema mecánico se ha implementado en Simulink con bloques de la biblioteca de Simscape Multibody, que proporciona un entorno de simulación para sistemas mecánicos tridimensionales. Simscape formula y resuelve automáticamente las ecuaciones de movimiento para el sistema

mecánico completo, por lo que se elimina el riesgo de cometer errores, no en el sistema de control, sino en el modelo simulado. Además de esto, Simscape Multibody genera animaciones tridimensionales que muestran el comportamiento del sistema, lo cual facilita el análisis de los resultados. En la figura 19 se puede ver el modelo tridimensional generado con Simscape del péndulo móvil.

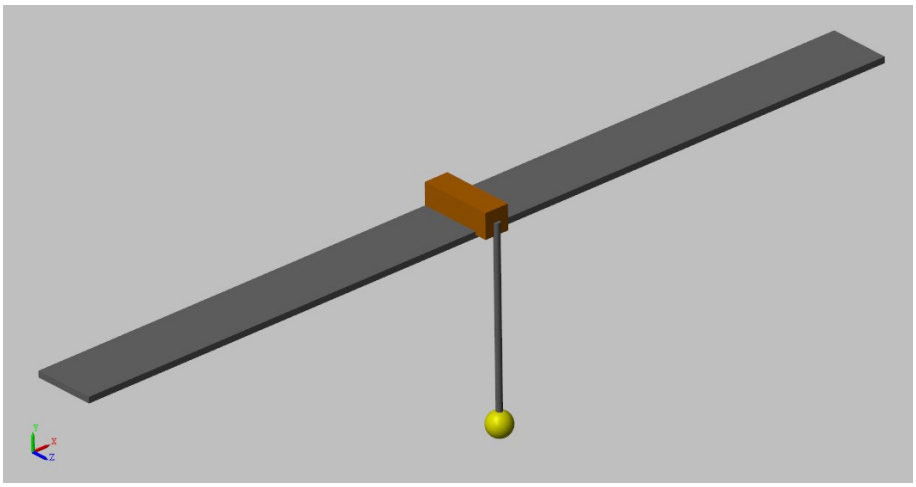

Figura 19: Vista tridimensional del péndulo móvil en estado de reposo.

El modelo de Simulink consta de tres bloques principales que se pueden ver en la figura 20: Un bloque central (color cian) que incluye el modelo del sistema físico como tal; un bloque posterior (color magenta) que añade ruido a las medidas, simulando los sensores reales; y un bloque al comienzo (color verde), que acondiciona la señal de entrada, y le añade ruido.

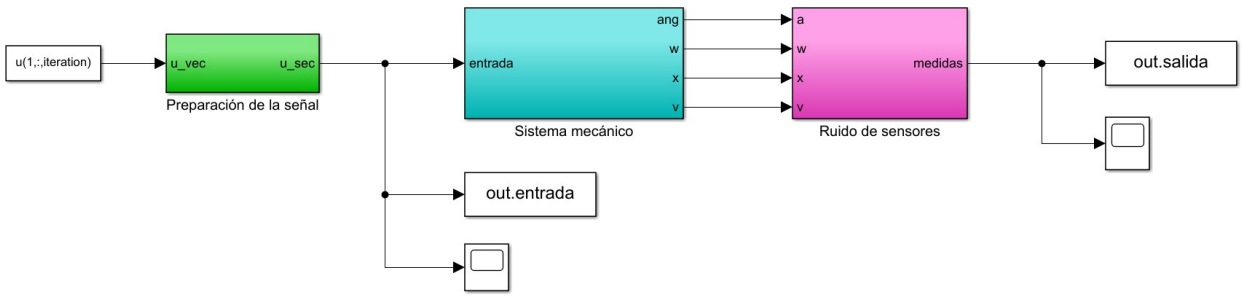

Figura 20: Vista general del modelo de Simulink que implementa la simulación del péndulo móvil.

3.1.1 Bloque del sistema mecánico

Este bloque esta formado íntegramente por los componentes del sistema mecánico. Estos son:

1. Una base, que constituye la pieza plana y larga sobre la que se apoya el carro. Esta pieza es fija y limita en sus extremos el movimiento lateral del carro.

2. Una articulación prismática que permite al carro moverse a lo largo de la base. Esta articulación incluye amortiguamiento. Además proporciona medidas tanto de posición como de velocidad lineal.
3. El carro, constituido por un prisma rectangular, que tiene, por supuesto, masa. La única fuerza que actúa sobre el sistema lo hace sobre este carro, en la dirección en la que tiene libertad.
4. Una articulación de revolución que une el carro con el péndulo, y que al igual que la otra articulación, tiene amortiguamiento y proporciona medidas de posición y velocidad.
5. El brazo del péndulo, el cual es rígido y tiene masa.
6. El peso del péndulo, unido al final del brazo, que, por supuesto, también tiene masa.

En adición a estos seis componentes, el modelo incluye varios bloques más que no se traducirían en la realidad por ningún elemento físico, sino que se utilizan para proporcionar toda la información necesaria al software. Se trata de: transformadas rígidas que definen las traslaciones y rotaciones entre los distintos elementos, un marco que permite establecer qué elementos están fijos, la definición de la gravedad, y los parámetros del solucionador de la simulación. En la figura 21 pueden verse todos los bloques empleados y sus conexiones.

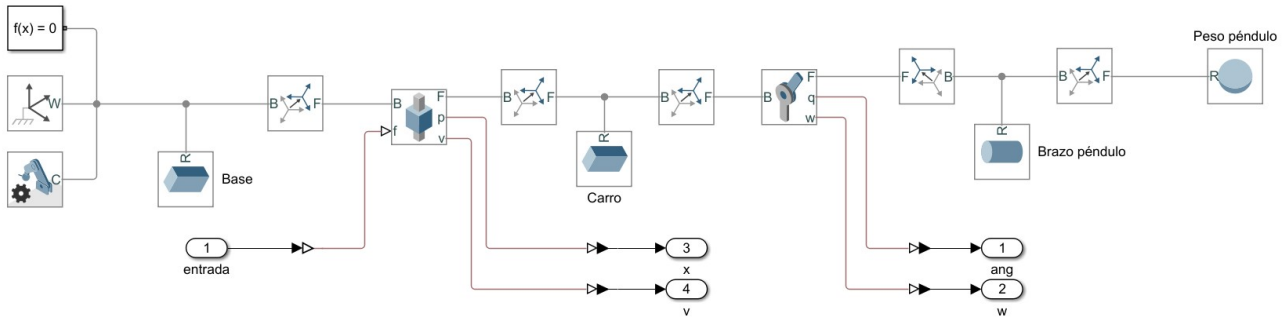

Figura 21: Modelo de Simulink que implementa el péndulo móvil con bloques de Simscape.

3.1.2 Ruido de sensores

En el bloque del sistema mecánico se realizan medidas de la posición y la velocidad de las articulaciones. Estas medidas no simulan las de sensores, sino que son perfectas. En el bloque de ruido de sensores se tratan las señales para adecuarlas al experimento.

En primer lugar, se aplica una compensación a la medida del ángulo para que el criterio coincida con el de la figura 18: Se mide el ángulo comenzando en la vertical hacia arriba y en el sentido positivo del eje z. Dado que la medida original comienza con el péndulo abajo, restamos media vuelta para corregir el desajuste.

En segundo lugar, las medidas se multiplexan en un solo vector de medidas, para manejarlas más cómodamente.

En tercer lugar, se aplica ruido Gaussiano aditivo a las cuatro medidas. Todas las señales de ruido tienen media cero e igual varianza, y se implementan con el bloque de Simulink *Random Number*. Dado que este bloque utiliza semillas para poder obtener repetibilidad, y se busca que el ruido no esté correlacionado entre señales ni entre iteraciones, utilizamos como semillas números aleatorios entre 1 y 10^9. En la figura 22 se pueden ver los bloques empleados y las conexiones entre ellos.

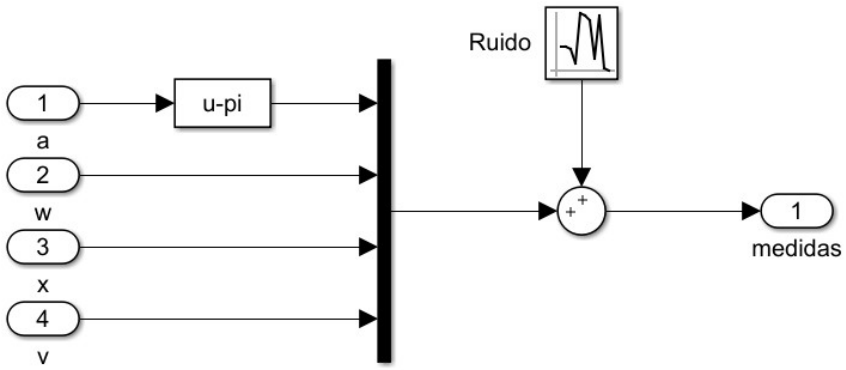

Figura 22: Bloques del modelo de Simulink que implementan la lectura de los sensores.

3.1.3 Manejo de la señal de entrada

La señal de entrada se calcula completamente en bucle abierto, por lo que se introduce un vector completo al modelo de Simulink. Lógicamente, al sistema mecánico hay que introducirle los datos de entrada uno a uno, por lo que es necesario secuenciar el vector de entrada. Para ello se ha utilizado un contador discreto y una función de código de Matlab. El contador tiene el mismo tiempo de muestreo que la señal de entrada, y la función utiliza el número dado por el contador para acceder al dato del vector con el mismo índice. El código de la función se resume por tanto en una sola línea, que se incluye a continuación.

```
u_s = u(t2+1);
```

Posteriormente, se suma a la señal un ruido Gaussiano de media cero, y varianza inferior a la que se añade a las señales de salida, utilizando el mismo método que con estas. Por último, se utiliza un *hold* de primer orden, ya que una señal discreta se va a introducir en un modelo continuo. En la figura 23 se pueden ver todos los bloques empleados y las conexiones entre ellos.

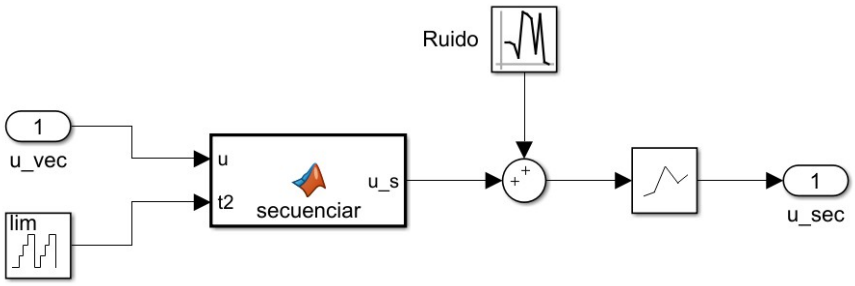

Figura 23: Bloques del modelo de Simulink que tratan la señal de entrada.

3.2 Modelo del sistema

Para aplicar el sistema de control es necesario un modelo del sistema lineal, definido en matrices de espacio de estados, que permita la construcción de las matrices vistas en el capítulo 2. Debido a que el péndulo es un sistema no lineal, es necesario linealizarlo.

Para ello se proponen dos opciones: La linealización de las ecuaciones dinámicas del sistema en torno a la trayectoria, y la identificación de sistemas.

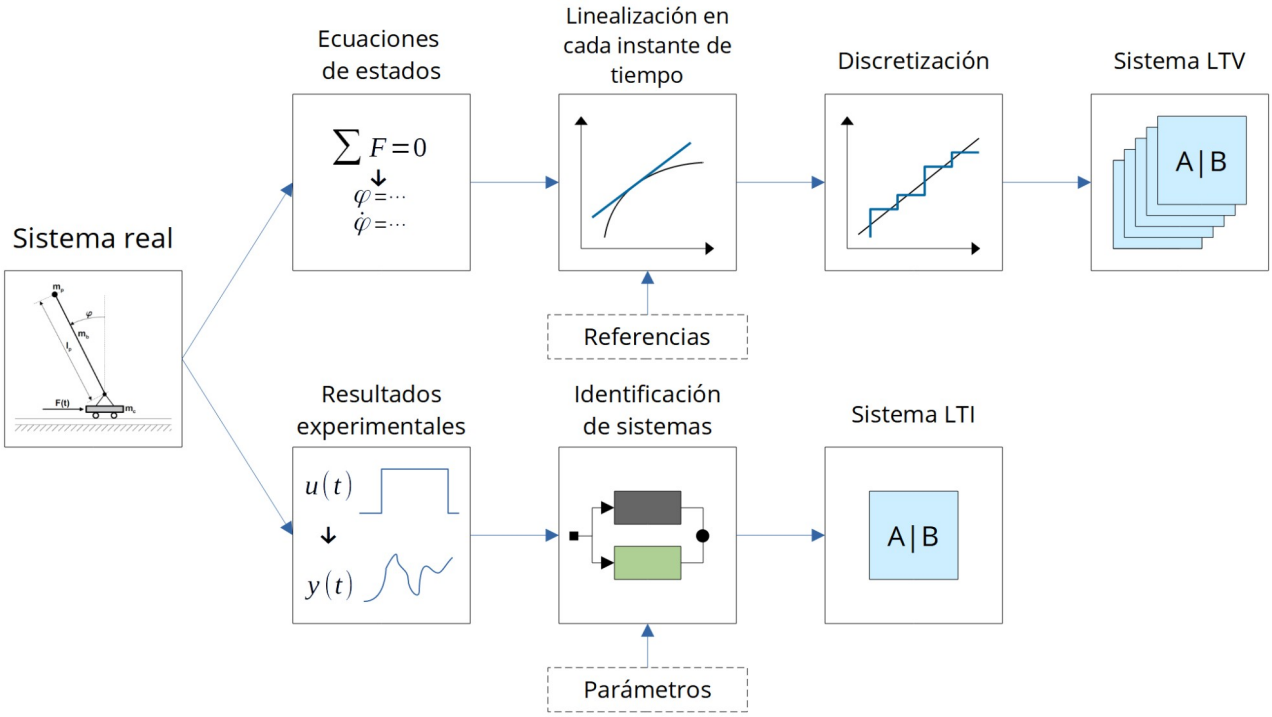

Figura 24: Diagrama con los pasos principales de cada método para obtener modelos lineales.

En la figura 24 se muestran los pasos principales de los dos procesos propuestos de obtención de modelos lineales.

El proceso de la parte superior del diagrama consiste en la deducción de las ecuaciones de estados no lineales a partir de las ecuaciones de equilibrio dinámico, la linealización de dichas ecuaciones de estados a lo largo de las trayectorias de referencia, y la discretización de los sistemas lineales obteniendo un sistema lineal variante en el tiempo (LTV). Se describe en detalle en el apartado 3.2.1.

El proceso de la parte inferior del diagrama consiste en la ejecución de un experimento con una entrada preconcebida que lleve al sistema medianamente cerca de las trayectorias deseadas. Con las medidas de ese experimento se realiza una identificación del sistema a través de técnicas de alto nivel que tienen como resultado, en este caso, un sistema lineal invariante en el tiempo (LTI). Se explica en detalle en el apartado 3.2.2.

3.2.1 Linealización de las ecuaciones dinámicas

En base a un esquema como el de la figura 18 se pueden deducir las ecuaciones de equilibrio dinámico del sistema. En las ecuaciones 41 se ignora el peso del brazo del péndulo m_b, ya que es pequeño en relación al del péndulo m_p, simplifica las ecuaciones, y contamos con que el aprendizaje iterativo corregirá este tipo de errores.

$$\left\{\begin{array}{c}(m_c+m_p)\ddot{x}+m_p l_p \ddot{\varphi}\cos(\varphi)-m_p l_p \dot{\varphi}\sin^2(\varphi)=F \\ m_p l_p \ddot{x}\cos(\varphi)+m_p {l_p}^2 \ddot{\varphi}-m_p g l_p \sin(\varphi)=0\end{array}\right\} \quad (41)$$

Desde el punto de vista matemático, la no linealidad se debe principalmente a las funciones trigonométricas que pueden observarse en las ecuaciones 41. La trayectoria que queremos que desarrolle el péndulo abarca desde la parte baja donde $\varphi=\pi\pm 2\pi n$, hasta la parte alta donde $\varphi=0\pm 2\pi n$, es decir, la mitad del rango de las funciones trigonométricas como mínimo. Por tanto, la linealización de este sistema no es trivial. Para compensarlo, es posible utilizar, en caso de necesidad, un modelo variante en el tiempo, en el que cada instante temporal corresponda con la linealización del modelo en el punto de la trayectoria en el que se espera que esté.

En base a las ecuaciones 41, despejando $\ddot{\varphi}$ y $\ddot{x}$ obtenemos las ecuaciones 42.

$$\left\{\begin{array}{c}\ddot{\varphi}=\dfrac{(\dfrac{m_c+m_p}{m_p}g-l_p\dot{\varphi}^2\cos(\varphi))\sin(\varphi)}{l_p(\dfrac{m_c}{m_p}+\sin^2(\varphi))}-\dfrac{\cos(\varphi)F}{m_p l_p(\dfrac{m_c}{m_p}+\sin^2(\varphi))} \\ \ddot{x}=\dfrac{-g\sin(\varphi)\cos(\varphi)+l_p\dot{\varphi}^2\sin(\varphi)}{\dfrac{m_c}{m_p}+\sin^2(\varphi)}+\dfrac{F}{m_c+m_p\sin^2(\varphi)}\end{array}\right\} \quad (42)$$

Definiendo el vector de estados $[\varphi, \dot{\varphi}, x, \dot{x}]^t$, se puede escribir la ecuación de las derivadas de los estados 43.

$$\begin{bmatrix} \dot{\varphi} \\ \ddot{\varphi} \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \dot{\varphi} \\ \dfrac{(\frac{m_c+m_p}{m_p} g - l_p \dot{\varphi}^2 \cos(\varphi))\sin(\varphi)}{l_p(\frac{m_c}{m_p}+\sin^2(\varphi))} \\ \dot{x} \\ \dfrac{-g\sin(\varphi)\cos(\varphi)+l_p\dot{\varphi}^2\sin(\varphi)}{\frac{m_c}{m_p}+\sin^2(\varphi)} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{-\cos(\varphi)}{m_p l_p(\frac{m_c}{m_p}+\sin^2(\varphi))} \\ 0 \\ \dfrac{1}{m_c+m_p\sin^2(\varphi)} \end{bmatrix} F \quad (43)$$

Derivando parcialmente respecto a los cuatro estados obtenemos las matrices jacobianas A y B.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{\dot{\varphi}^2\sin^2\varphi}{\sin^2\varphi+\frac{m_c}{m_p}} - \dfrac{\cos\varphi\,(l_p\cos\varphi\,\dot{\varphi}^2 - \frac{g(m_c+m_p)}{m_p})}{l_p(\sin^2\varphi+\frac{m_c}{m_p})} + \dfrac{2\cos\varphi\sin^2\varphi\,(l_p\cos\varphi\,\dot{\varphi}^2 - \frac{g(m_c+m_p)}{m_p})}{l_p(\sin^2\varphi+\frac{m_c}{m_p})^2} & \dfrac{-2\dot{\varphi}\cos\varphi\sin\varphi}{\sin^2\varphi+\frac{mc}{mp}} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{l_p\dot{\varphi}^2\cos\varphi - g\cos^2\varphi + g\sin^2\varphi}{\sin^2\varphi + m_c/m_p} + \dfrac{2\cos\varphi\sin\varphi\,(-l_p\sin\varphi\,\dot{\varphi}^2 + g\cos\varphi\sin\varphi)}{(\sin^2\varphi+\frac{m_c}{m_p})^2} & \dfrac{2l_p\dot{\varphi}\sin\varphi}{\sin^2\varphi+\frac{m_c}{m_p}} & 0 & 0 \end{bmatrix} \quad (44)$$

$$B = \begin{bmatrix} 0 \\ \dfrac{-\cos\varphi}{l_p m_p(\sin^2\varphi+\frac{m_c}{m_p})} \\ 0 \\ \dfrac{1}{m_p(\sin^2\varphi+\frac{m_c}{m_p})} \end{bmatrix} \quad (45)$$

Sustituyendo en las matrices jacobianas (44 y 45) las constantes por sus valores, y las variables por el valor central del rango en el que se quiere linealizar, se obtiene un sistema lineal continuo definido en espacio de estados por las matrices A y B. Estos sistemas linealizados están centrados siempre en cero, para todos los estados. Por tanto, si se toma un punto en el que los estados no son cero, el valor de los estados debería sumarse a la salida del sistema para que coincidan el valor de los sistemas real y linealizado. Por ejemplo, en la parte baja del péndulo, y para cierto valor de las constantes, se sustituyen los siguientes valores en las matrices:

```
% Constantes
lp = 0.30;
mp = 0.150;
mc = 1.5;
g = -9.81;

% Parte baja del péndulo
phi0 = -pi;
phi1 = 0;
x0 = 0;
x1 = 0;
```

Con lo que se obtienen las matrices A_0 y B_0:

$$A_0=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 35{,}97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0{,}981 & 0 & 0 & 0 \end{bmatrix},\quad B_0=\begin{bmatrix} 0 \\ 2{,}222 \\ 0 \\ 0{,}667 \end{bmatrix} \quad (46)$$

Para que el modelo emule correctamente el comportamiento del péndulo móvil, tanto el ángulo φ como la velocidad angular deben encontrarse cerca de −π y 0 respectivamente. Sin embargo, en el sistema definido por A_0 y B_0 todos los estados se mueven alrededor del 0, por lo que, para hacer coincidir los criterios, se sumaría posteriormente −π al ángulo φ.

Si se desea modelar el comportamiento del péndulo, desde la parte baja ($\varphi=-\pi$) hasta la parte alta ($\varphi=0$), para poder cumplir la hipótesis de ángulos pequeños, es necesario crear un modelo variante en el tiempo dividiendo la trayectoria entre sistemas linealizados que expliquen el comportamiento en cada zona. En la figura 25 se esboza una posible división de la trayectoria del péndulo en sectores, cada uno representado por un sistema linealizado distinto, con sendas matrices A y B.

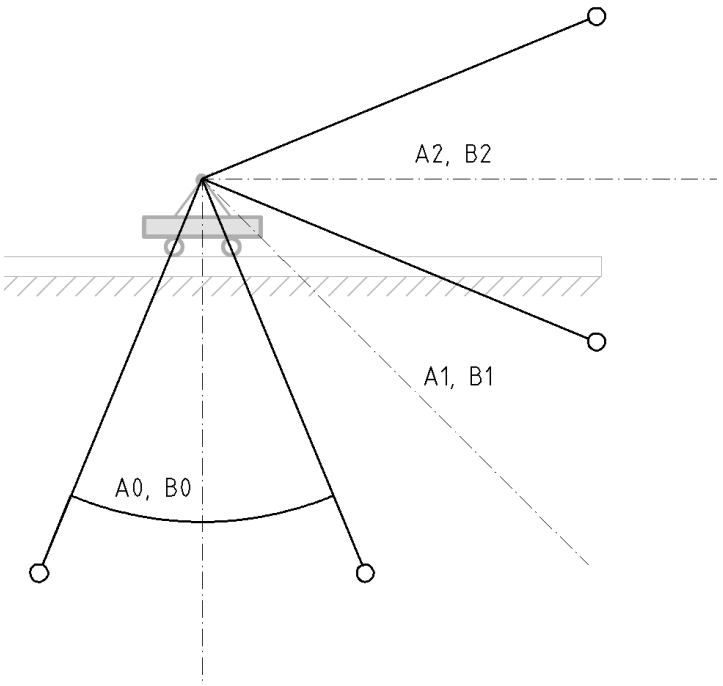

Figura 25: División en sectores de la trayectoria del péndulo. La línea de raya y punto indica la posición en la que se realiza la linealización. Las posiciones representadas del péndulo indican el límite de cada sector.

Esta estrategia presenta un problema: Al cambiar de una zona de linealización a otra, el valor de los estados debería cambiar consecuentemente para centrar el cero en el punto de equilibrio. La formación del vector d (12) y la matriz F (13) requiere las matrices A(t) y B(t) en todo el rango temporal, y no permite el ajuste del valor de los estados en instantes intermedios.

Para solucionarlo se cambia el sistema de referencia sobre el que trabaja el sistema de control: En el caso de querer calcular solamente la variación de la entrada para contrarrestar el error almacenado en el vector d, la referencia que se introduce en el problema de optimización es nula para todas las trayectorias. Esto se debe a que una señal de entrada previa se encarga de guiar al sistema en una trayectoria mínimamente cercana a la referencia, lo que permite que la tarea del control sea calcular un incremento en la entrada que llevará el error a una referencia cero. Así, en vez de introducir las referencias en el problema de optimización y pedirle una entrada que se ajuste a ellas, se introduce solamente el error medido previamente en el problema de optimización, y se pide una entrada que anule el error. Si consideramos que los errores en la trayectoria son relativamente pequeños, trabajaremos siempre con desviaciones pequeñas y cerca del cero, con lo que cumplimos las condiciones de utilización de los sistemas linealizados.

Cuando el sistema se aleja de los valores centrales de la linealización, la precisión de esta disminuye. Para mejorar la fidelidad del modelo lineal, es posible reducir el tamaño de los sectores linealizados. En la figura 25, cuanto menos arco cubra cada sistema de matrices lineales, más fiel al sistema real será la aproximación. Dado que el sistema resultante es lineal variante en el tiempo, el límite de tamaño para los sectores será una muestra temporal. Este tamaño es el escogido para la construcción del modelo linealizado.

El proceso completo consta de los siguientes pasos:

1. Definición de las referencias a lo largo del tiempo.
2. Linealización del sistema en las posiciones definidas para cada instante.
3. Discretización de todos los sistemas linealizados, con el mismo método y tiempo de muestreo.
4. Concatenación de las distintas matrices A, B y C en la dimensión temporal.

A continuación se incluye el código de Matlab mediante el que se linealiza el sistema en torno a todos los puntos de las referencias.

```
for i = 1:t_final
    % Sustitución de valores en cada instante de la trayectoria de
referencia
    % Valor de los estados en el instante temporal i
    phi0 = ref(1, i);
    phi1 = ref(2, i);
    x0 = ref(3, i);
    x1 = ref(4, i);

    % Sustitución en las matrices simbólicas A y B
    A0 = double(vpa(subs(A)));
    B0 = double(vpa(subs(B)));
    C0 = C; % La matriz C es invariante en el tiempo

    [A_tv(:,:,i), B_tv(:,:,i), C_tv(:,:,i), ~, ~] =
ssdata(c2d( ss(A0,B0,C0,[]), 0.01)); % Discretización
end
```

3.2.2 Identificación de sistemas

Con el fin de encontrar un único sistema lineal que explique el comportamiento del péndulo móvil a lo largo de la trayectoria se ha utilizado una técnica de identificación de sistemas: el comando de Matlab ssest, que crea un modelo lineal en espacio de estados con una respuesta similar a la de los datos introducidos.

> "ssest inicializa las estimaciones de los parámetros utilizando un enfoque subespacial no iterativo o un enfoque iterativo de estimación de funciones racionales. Luego

refina los valores de los parámetros utilizando el enfoque de minimización del error de predicción." [20]

Se utilizan los datos de entrada y salida de un experimento medianamente cercano a la trayectoria deseada. Se utilizan las medidas de todas las entradas y todas las salidas, que, en este caso, son todos los estados. Se muestran las señales de entrada y salida en la figura 26.

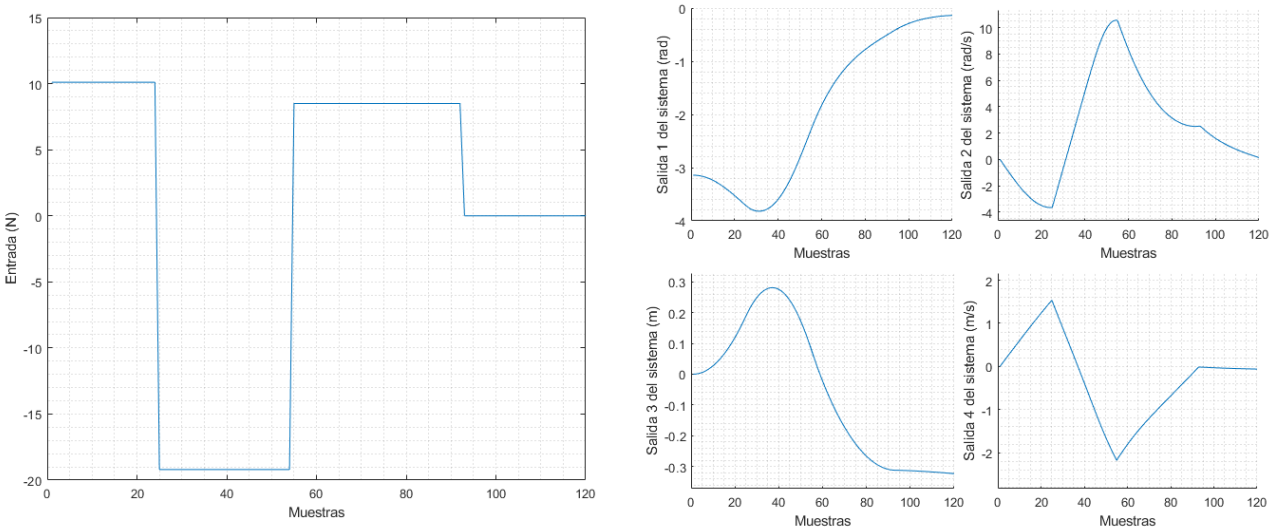

Figura 26: Entrada aplicada y salidas del sistema utilizadas para la identificación de sistemas.

Tras probar numerosas configuraciones, el mejor resultado lo proporciona la siguiente combinación:

- Modelo discreto, con tiempo de muestreo 0,01 segundos, que es el que se utilizará posteriormente con el control.
- Forma canónica observable. Es decir, que la matriz C es una matriz identidad. Esto sirve para que los estados tengan el mismo significado físico que en el modelo real, y es conveniente por si se quisiese combinar el modelo resultante con otro para crear un modelo variante en el tiempo.
- Método de estimación: *Prediction Error Minimization*, junto a esquema de pesos CVA (*Canonical Variate Algorithm*), enfoque en simulación (frente a predicción), y estados iniciales estimados mediante *backcast* (utilizando el mejor ajuste de mínimos cuadrados).

Lo cual se ejecuta con el siguiente código:

```
% State space model estimation
Options = ssestOptions;
Options.Display = 'on';
Options.EstimateCovariance = false;
```

```
Options.Focus = 'simulation';
Options.EnforceStability = 1;
Options.InitialState = 'backcast';

ss = ssest(4_pend_s, 4, 'Form', 'canonical', 'DisturbanceModel', 'none',
'Ts', 0.01, Options)
```

Y proporciona el siguiente resultado:

$$A=\begin{bmatrix} 1{,}007 & 0{,}0283 & 0{,}1745 & 0{,}103 \\ -0{,}2371 & 0{,}5613 & -2{,}5834 & -2{,}138 \\ -0{,}0038 & -0{,}0078 & 0{,}9457 & -0{,}0296 \\ 0{,}03 & 0{,}005 & 0{,}025 & 1{,}0203 \end{bmatrix},\quad B=\begin{bmatrix} 0{,}0044 \\ -0{,}0776 \\ -0{,}0013 \\ 0{,}0065 \end{bmatrix} \tag{47}$$

En la figura 27 se representa la evolución de los estados del modelo lineal obtenido junto a las medidas del sistema real ante una misma entrada. Observamos que la respuesta no es muy precisa, y en algunas partes de la trayectoria la aproximación es bastante mediocre. Sin embargo, sí se puede apreciar cómo capta el funcionamiento a grandes rasgos del sistema.

La obtención de un sistema lineal con este método es mucho más simple, rápido y fácil que con el presentado en el apartado 3.2.1; el sistema resultante también es más simple, y, además, no requiere la modificación de las referencias. Si este modelo es suficiente para aplicar el sistema de control a un sistema tan no lineal como el péndulo móvil, es razonable considerar que este método de linealización será el más adecuado para la mayoría de casos y sistemas.

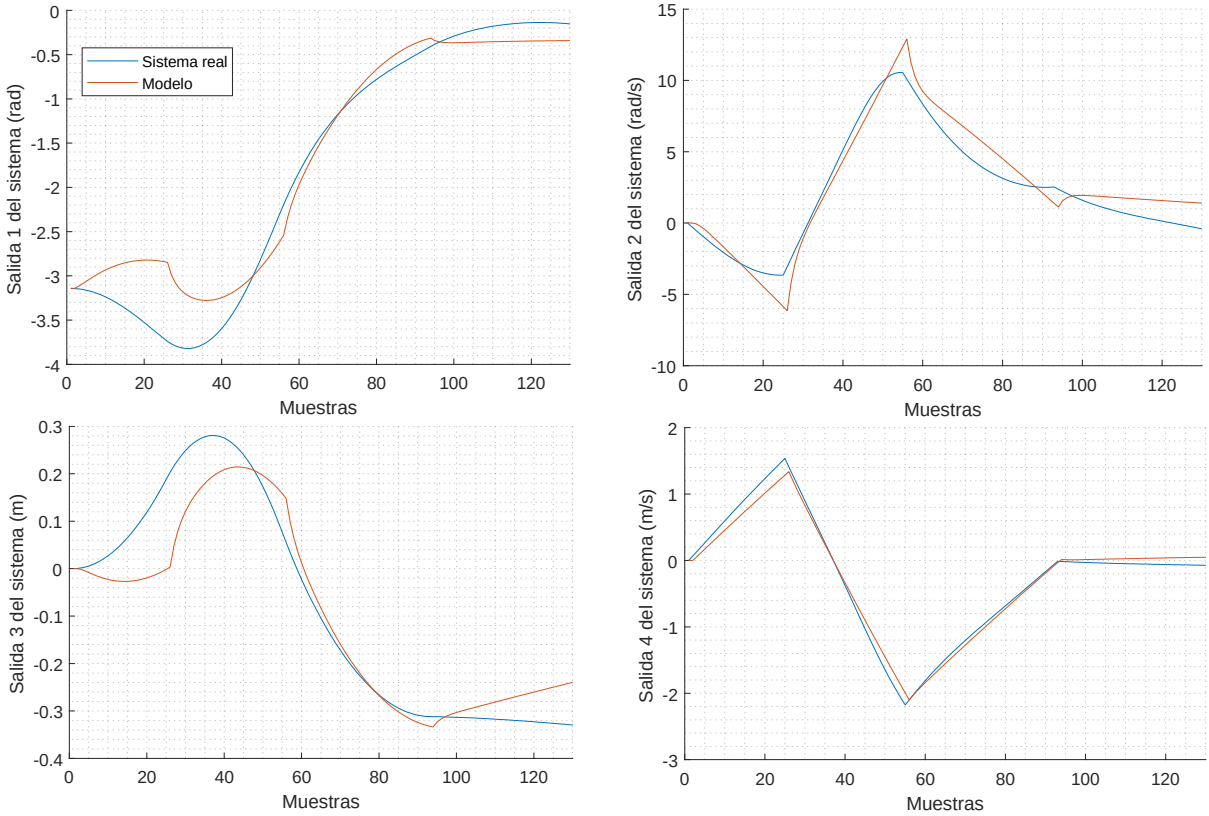

Figura 27: Respuesta del sistema real y el modelo linealizado para una misma entrada.

4 Resultados

Se han ejecutado dos modalidades distintas del experimento: Una primera que utiliza referencias absolutas y un modelo lineal obtenido mediante identificación de sistemas (según se describe en el apartado 3.2.2); y una segunda que utiliza referencias cero, cálculo del incremento de la señal de entrada, y un modelo lineal obtenido mediante sustitución en las matrices jacobianas en torno a las trayectorias de referencia (descrito en el apartado 3.2.1). Se han ejecutado ambos experimentos a lo largo de diez iteraciones.

4.1 Control con referencias absolutas

Se aplican los parámetros de la tabla 3 a los bloques de Simscape. Estos parámetros son los mismos que se utilizaron para obtener el modelo lineal en el apartado 3.2.2.

Se aplican los parámetros de la tabla 4 al sistema de control. Nótese que los límites aplicados a la posición horizontal del carro son más pequeños que la mitad del rango máximo de movimiento x_{range}. Se dejan estos márgenes debido a que, a causa de las imprecisiones del modelo lineal, los límites no se respetan al milímetro. El resto de límites aplicados a las salidas son bastante amplios, ya que no hay ninguna frontera clara para ellos, y en principio es suficiente con que no sean extremadamente grandes. El límite aplicado a la entrada es bastante laxo también, pero como se puede ver en las gráficas de resultados, el sistema nunca da valores demasiado grandes a las entradas: En general se limita a lo necesario.

Tabla 3: Parámetros aplicados al modelo simulable de Simscape.

Descripción	Nombre	Valor
Masa del extremo del péndulo	m_p	0,09 kg
Masa del brazo del péndulo	m_b	0,06 kg
Masa del carro	m_c	1,50 kg
Longitud del brazo	l_p	0,3 m
Longitud de la base	x_{range}	1,4 m
Aceleración gravitatoria	g	–9,80665 m/s^2
Amortiguamiento prismático (movimiento del carro)	α_l	8e–6 Ns/m
Amortiguamiento rotacional (giro del péndulo)	α_r	8e–6 Ns/m
Desviación típica del ruido en la entrada	σ_u	0,0075
Desviación típica del ruido en las salidas	σ_y	0,03 (todas)
Estados iniciales	x(0)	[– π, 0, 0, 0]

Tabla 4: Parámetros aplicados al sistema de control para el primer experimento del péndulo.

Descripción	Nombre	Valor
Número de iteraciones	–	10
Duración de cada iteración	t_{final}	2 s
Confianza inicial en las medidas frente al modelo	ε	1000
Restricciones a la entrada	u_{lim}	-100 N < u(k) < 100 N
Restricciones a las salidas	y_{lim}	-8π rad < $y_1(k)$ < 4π rad -80 rad/s < $y_2(k)$ < 80 rad/s $-0{,}5$ m < $y_3(k)$ < 0,5 m -20 m/s < $y_4(k)$ < 20 m/s
Desviación típica estimada del ruido	σ_{est}	0,054
Pesos de las trayectorias	r	[1, 1, 1, 100]
Estados iniciales	x(0)	[$-\pi$, 0, 0, 0]

En la figura 28 se muestra el ángulo del péndulo a lo largo del tiempo en las diez iteraciones. Se puede apreciar cómo las iteraciones se ciñen a la referencia durante el primer tercio de la trayectoria. A partir del segundo 0,8, se observa cómo las primeras iteraciones se alejan más rápidamente de la referencia, y conforme avanza el número de iteración se van ajustando más tiempo a ella. A partir de la cuarta iteración podemos considerar que se cumple el objetivo de poner el péndulo hacia arriba. La décima iteración permanece cerca de un segundo en la posición vertical.

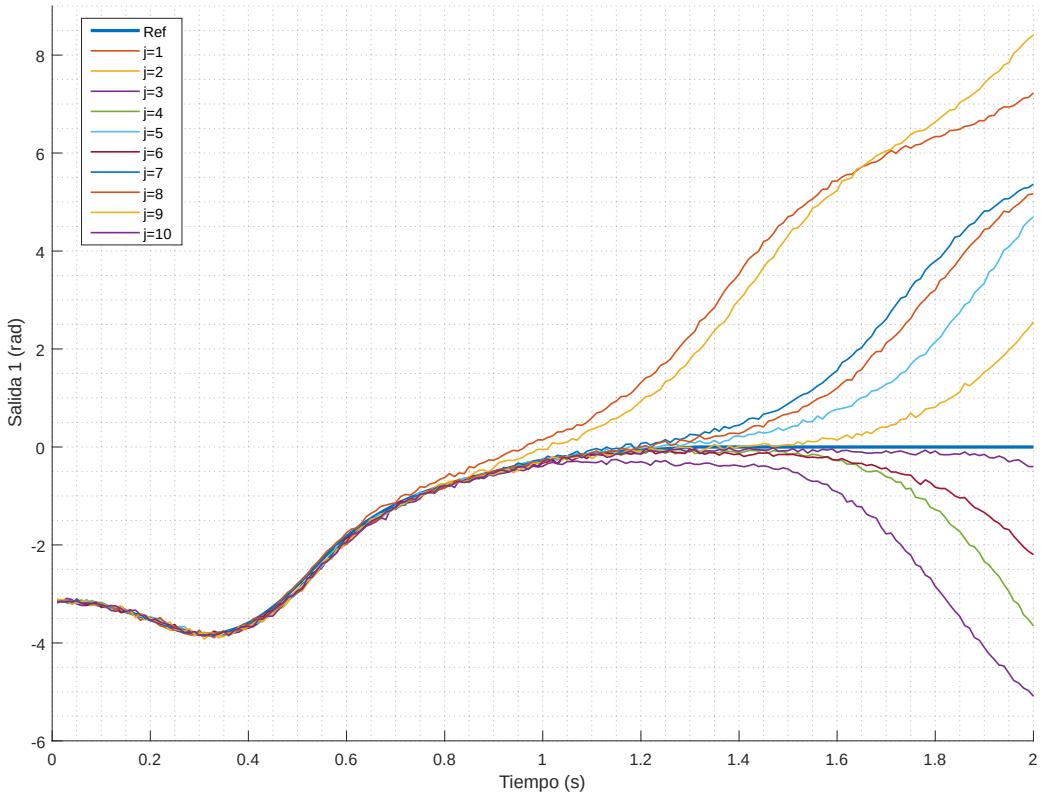

Figura 28: Evolución del ángulo del péndulo a lo largo del tiempo en diez iteraciones (exp. 1).

En la figura 29 se representan las entradas utilizadas que resultan en la evolución del ángulo de la figura 28. Observamos que durante la primera mitad de la trayectoria apenas se producen cambios a lo largo de las diez iteraciones. Durante la segunda mitad de la trayectoria las entradas evolucionan mucho más a lo largo de las iteraciones, provocando el progreso de las salidas de la figura 28. En cada iteración, la entrada se aplica estrictamente en bucle abierto, como se describe en el capítulo 2.

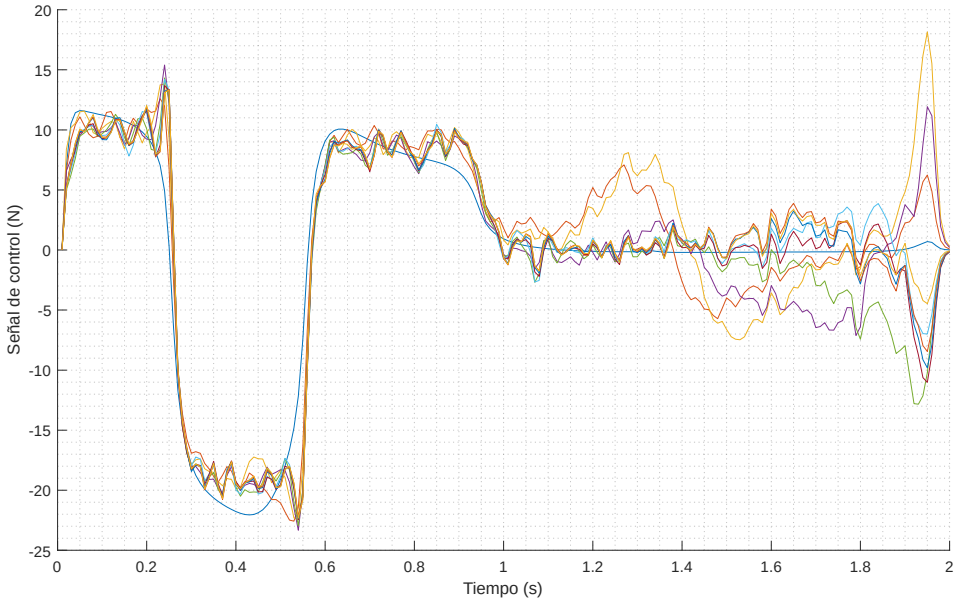

Figura 29: Evolución de las entradas a lo largo de las diez iteraciones (exp. 1).

En la figura 30 se muestra la evolución del error cuadrático medio entre las referencias y las salidas a lo largo de las iteraciones. La escala del eje de ordenadas es logarítmica. Se observa una tendencia general hacia abajo conforme avanzan las iteraciones. Esta tendencia no es monótona ya que el error aumenta en ocasiones para algún estado. De hecho, la que consideramos como mejor iteración observando la trayectoria del ángulo en la figura 28, es la que proporciona menor error para esta salida, así como para la velocidad angular, pero proporciona mayor error que otras iteraciones tanto para la posición longitudinal como para la velocidad longitudinal.

En la figura 31 se presentan las cuatro salidas del sistema y sus referencias. Se observa cómo todas las salidas aprenden a mantenerse relativamente cerca de sus respectivas referencias.

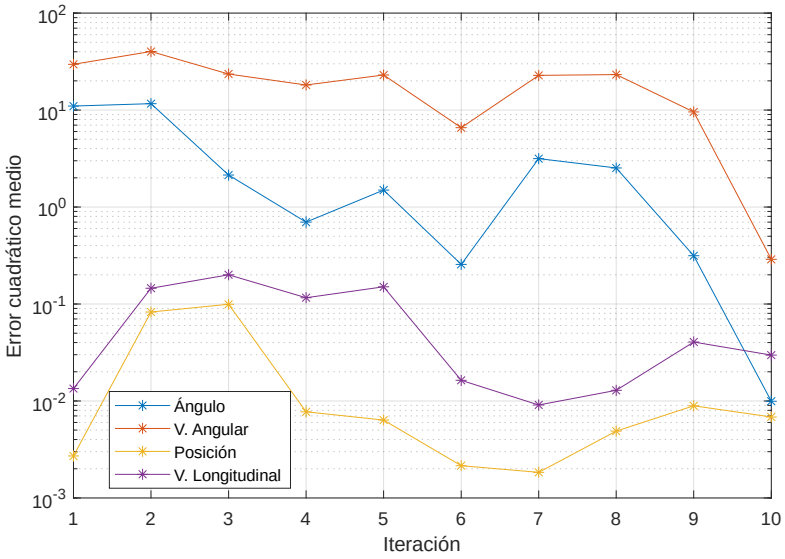

Figura 30: Evolución del error cuadrático medio de las cuatro salidas a lo largo de las iteraciones (exp. 1).

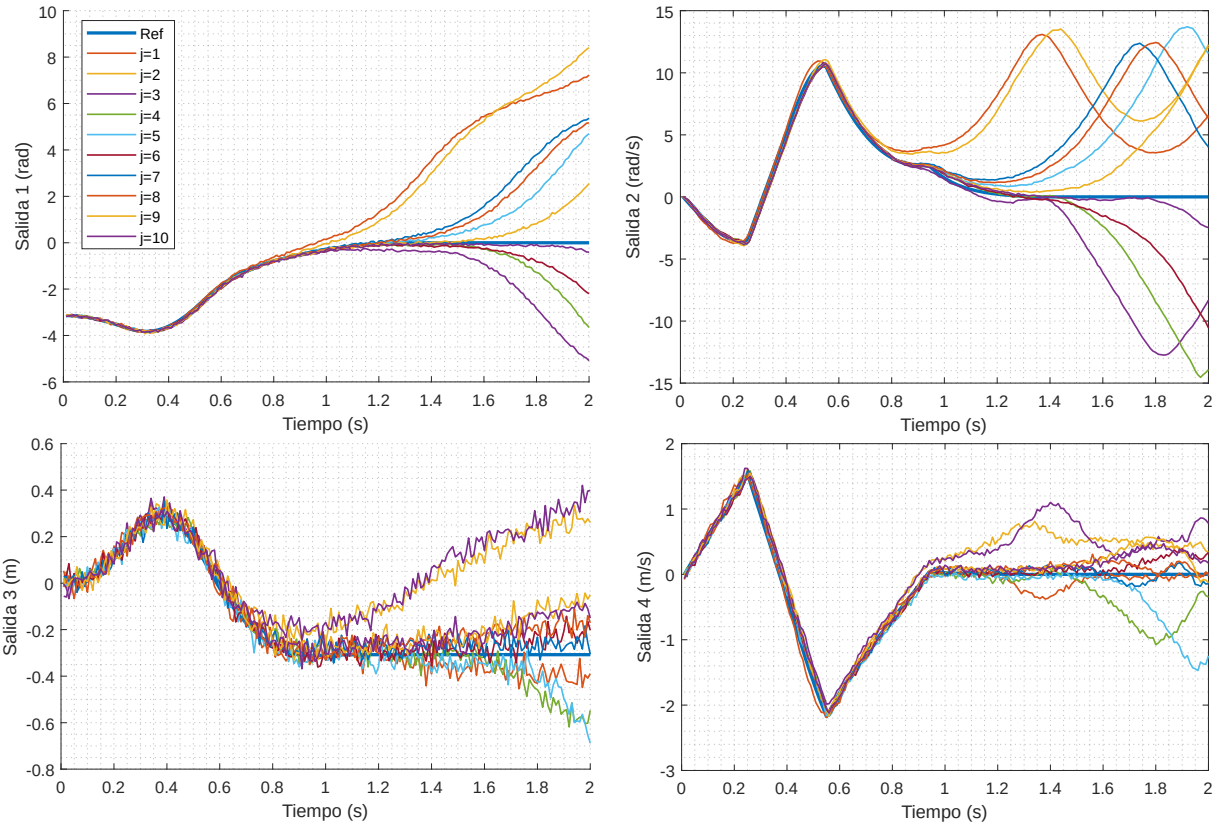

Figura 31: Evolución de las cuatro salidas a lo largo del tiempo en las diez iteraciones (exp. 1).

4.2 Control por anulación del error

Se utiliza el modelo lineal variante en el tiempo obtenido según el procedimiento descrito en el apartado 3.2.1. La entrada inicial utilizada se ha diseñado a mano, por prueba y error. Se puede ver dicha entrada representada con un grosor de línea mayor en la figura 33, y su respuesta se puede ver en la primera iteración de la figura 32.

Se aplican los mismos parámetros de la tabla 3 a los bloques de Simscape. La única excepción es la desviación típica del error en la tercera salida (posición longitudinal) que se reduce de $\sigma_{y3} = 0{,}03$ a $\sigma_{y3} = 0{,}0075$.

Se aplican los parámetros de la tabla 5 (algunos distintos de la tabla 4) al sistema de control. Se ha modificado la confianza inicial en las medidas, de forma consecuente al esquema de aprendizaje. Se han modificado los pesos de las trayectorias por valores que funcionan mejor para este segundo caso. Se han cambiado los estados iniciales, ya que ahora tanto los estados como las referencias, de cara al optimizador, son cero. Nótese también que los límites aplicados a la posición horizontal del carro son mucho más ajustados al rango máximo de movimiento x_{range} que los del experimento anterior. Se reducen esos márgenes ya que esta forma de aprendizaje los respeta con más precisión. Se aplica también la estrategia de extensión de horizonte, con una única condición de terminación aplicada sobre el error en la posición angular del péndulo.

Tabla 5: Parámetros aplicados al sistema de control para el segundo experimento del péndulo.

Descripción	Nombre	Valor
Número de iteraciones	–	10
Duración de cada iteración	t_{final}	2 s
Confianza inicial en las medidas frente al modelo	ε	0,01
Restricciones a la entrada	u_{lim}	$-30\ \text{N} < u(k) < 30\ \text{N}$
Restricciones a las salidas	y_{lim}	$-8\pi\ \text{rad} < y_1(k) < 4\pi\ \text{rad}$ $-80\ \text{rad/s} < y_2(k) < 80\ \text{rad/s}$ $-0{,}65\ \text{m} < y_3(k) < 0{,}65\ \text{m}$ $-20\ \text{m/s} < y_4(k) < 20\ \text{m/s}$
Desviación típica estimada del ruido	σ_{est}	0,054
Pesos de las trayectorias	r	[1, 1, 0,5, 1]
Peso de penalización a la entrada	q	1e–9
Estados iniciales	x(0)	[0, 0, 0, 0]
Condición de terminación	y	$e_1(k) > 1\ \text{rad}$

En la figura 32 se presentan las cuatro salidas del sistema y sus referencias. Se puede observar un comportamiento similar al del experimento anterior de la figura 31:

- En cuanto al ángulo, todas las iteraciones se ciñen a la referencia durante el primer tercio de la trayectoria. A partir del segundo 0,8, se observa cómo las primeras iteraciones se alejan más rápidamente de la referencia, y conforme avanza el número de iteración se van ajustando más tiempo a ella. En este caso la sexta iteración es la mejor, y permanece cerca de 0,6 segundos en posición vertical. En general, observando el conjunto de todas las iteraciones, el resultado es bastante parecido al observado en la figura 28.
- Las velocidades angular y lineal se comportan de manera similar a las de la figura 31: Se ajustan muy bien la primera mitad de la trayectoria, y después se alejan de manera distinta en cada iteración.
- La posición lineal se comporta de manera un poco distinta. En primer lugar, el ruido es inferior en el segundo experimento, y por tanto se pueden observar mejor las señales. En segundo lugar, todas las trayectorias son inferiores a la referencia. Esto se debe a que esta salida tiene menor importancia que las demás, y el control compromete parte del desempeño de esta entrada para mejorar las demás.

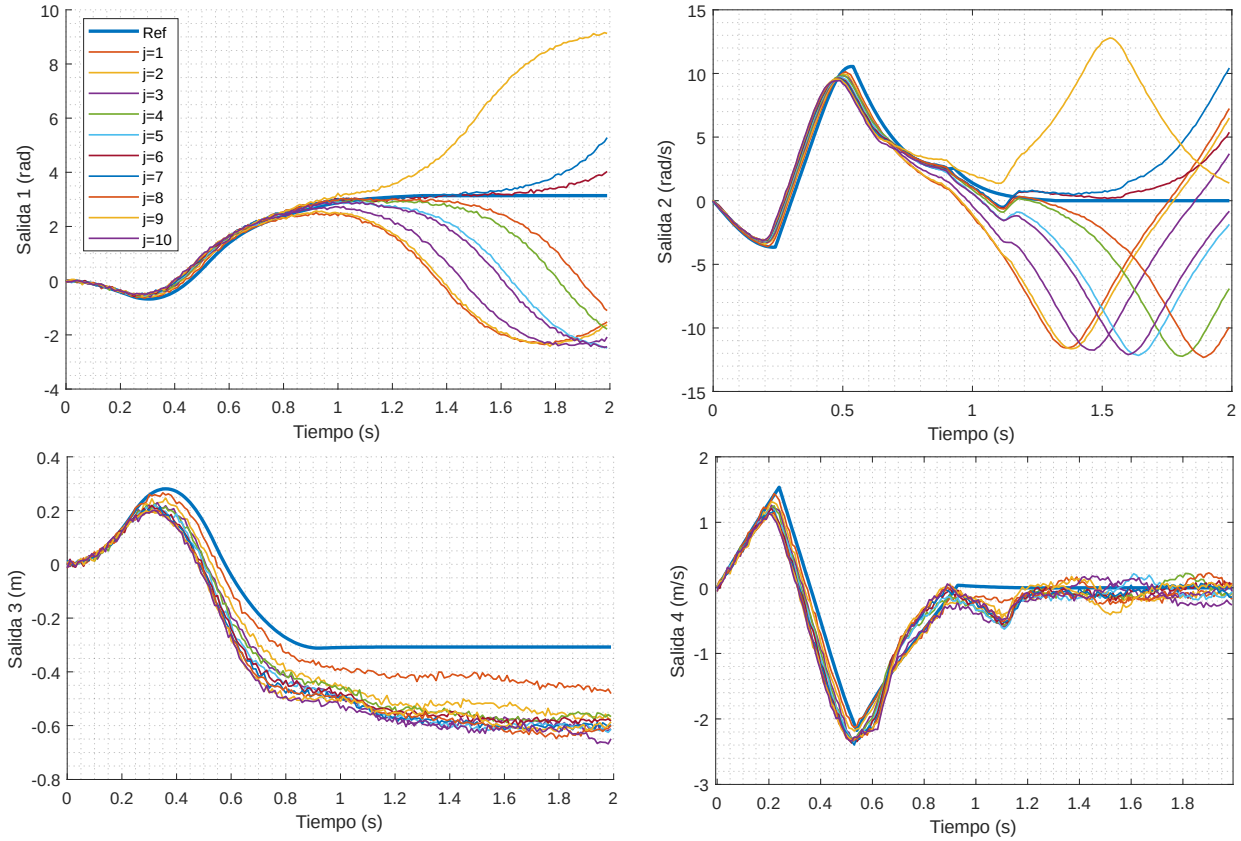

Figura 32: Evolución de las cuatro salidas a lo largo del tiempo en las diez iteraciones (exp. 2).

En la figura 33 se representan las entradas utilizadas que resultan en la evolución de las entradas de la figura 32. La entrada inicial se ha representado con un grosor de línea mayor. Observamos que la mayoría de los cambios se producen a partir del segundo 0,6. Al igual que en el experimento anterior, las entradas se aplican estrictamente en bucle abierto.

En relación a las entradas del primer experimento (figura 29), observamos que estas concentran los cambios en zonas concretas de la trayectoria, creando picos más pronunciados. Es probable que esto se deba a que la entrada inicial, diseñada a mano, es también más agresiva y menos redondeada que las que el control encuentra con el método de seguimiento de referencias absolutas. En cualquier caso, a pesar de evolucionar y converger de manera diferente, las salidas obtenidas mediante las entradas cumplen con el comportamiento esperado en ambos casos.

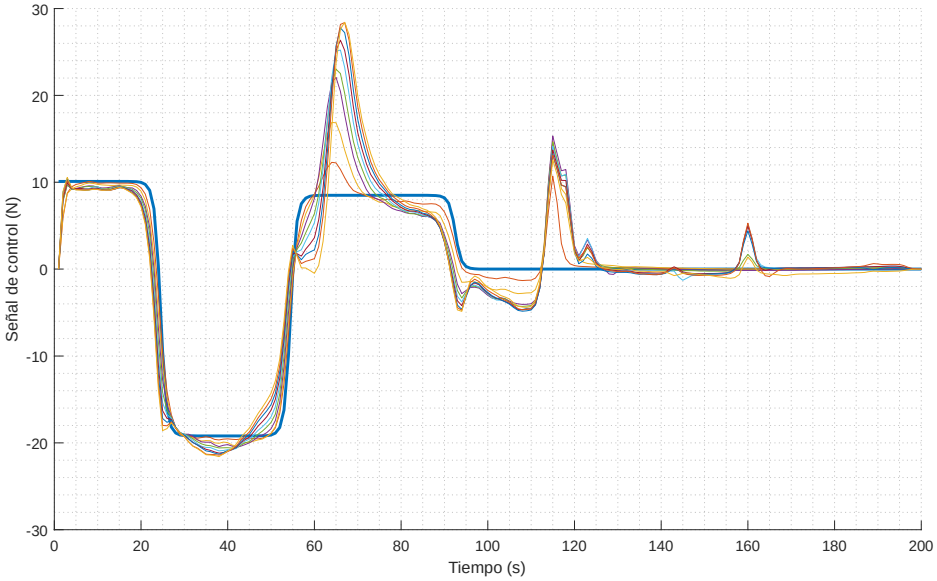

Figura 33: Evolución de las entradas a lo largo de las diez iteraciones (exp. 2).

En la figura 34 se muestra la evolución del error cuadrático medio entre las referencias y las salidas a lo largo de las iteraciones. La escala del eje de ordenadas es logarítmica. Se observa una tendencia descendente a lo largo de las iteraciones para los errores del ángulo y la velocidad angular, a pesar de que las dos últimas tienen un pico local. Sin embargo, la posición y velocidad longitudinales aumentan ligeramente de principio a fin. En las iteraciones centrales se producen los mejores resultados para el ángulo y la velocidad angular, con errores inferiores por más de un orden de magnitud a las iteraciones con mayor error.

En comparación con el error cuadrático medio del primer experimento (figura 30), la evolución del aprendizaje es diferente, ya que los mínimos se encuentran en iteraciones distintas. Sin embargo, estas diferencias se encuentran también entre diferentes ejecuciones del experimento con un mismo método. Por otro lado, los valores de las iteraciones con mayor error son bastante similares en magnitud en ambos experimentos, y análogamente los valores de las iteraciones con menor error son también del mismo orden.

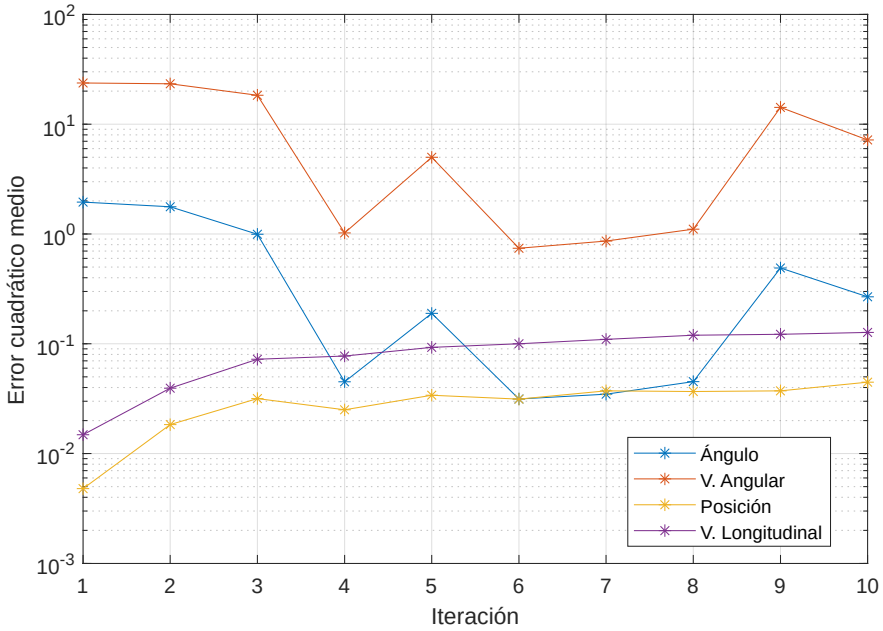

Figura 34: Evolución del error cuadrático medio de las cuatro salidas a lo largo de las iteraciones (exp. 2).

4.3 Comparación de ambos métodos

Con el fin de comparar la efectividad de ambos métodos de aprendizaje, se ha llevado a cabo el siguiente proceso:

1. Se han ejecutado los dos algoritmos de aprendizaje diez veces con diez iteraciones en cada ocasión.
2. De cada iteración se ha almacenado el error cuadrático medio entre la referencia y la trayectoria del ángulo del péndulo. Solo se ha tenido en cuenta la posición angular, y se desechan el resto de estados, ya que el objetivo final del control es colocar el péndulo en posición vertical.

   Dado que la parte final de la trayectoria describe un comportamiento caótico, se suele desviar de la trayectoria y aumenta mucho el error. Esto hace que el error acumulado a lo largo del resto de la trayectoria se diluya y tenga muy poco peso sobre el resultado final. Por tanto, consideramos las ejecuciones terminadas en el instante 1,5 segundos. Esto ayuda también a igualar los resultados en el caso de realizar los experimentos con distinta duración para cada método de aprendizaje. En cualquier caso, si el péndulo permanece en posición vertical hasta ese instante, podemos considerar el resultado como muy satisfactorio.

3. Tratando los dos métodos de control por separado, se ha calculado la media del error para todas las ejecuciones de una misma iteración. Es decir, se ha calculado la media del error para las primeras iteraciones , segundas iteraciones, terceras iteraciones, etcétera.

4. Se han representado, en la figura 35, los resultados para los dos métodos. Ambos convergen en pocas iteraciones a niveles similares de error, si bien el seguimiento de referencias absolutas proporciona un aprendizaje un poco más rápido y valores de error en las últimas iteraciones ligeramente inferiores.

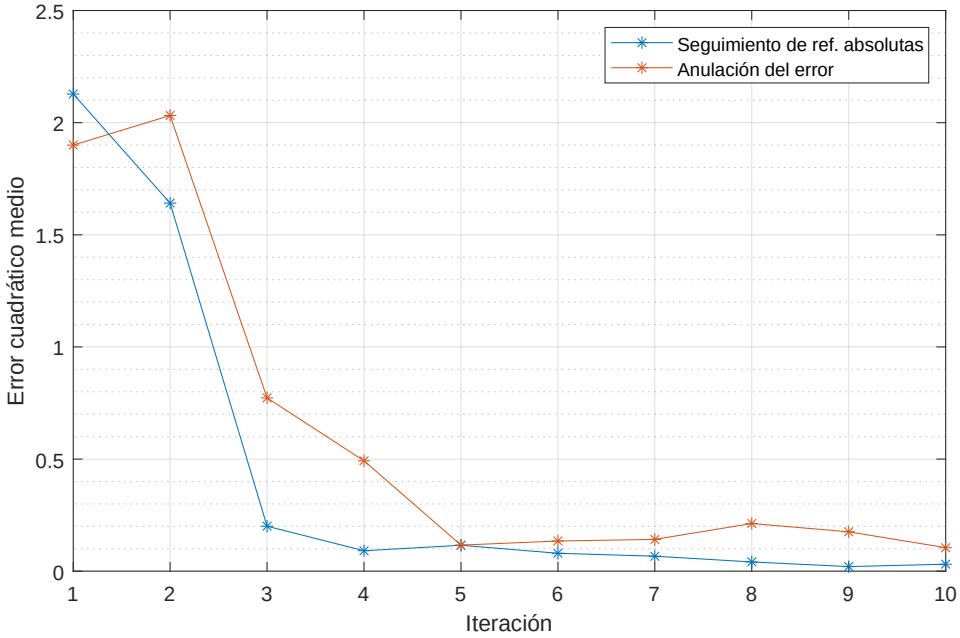

Figura 35: Media del error cuadrático medio del ángulo para cada iteración. Obtenida para los dos métodos con diez experimentos de diez iteraciones cada uno.

La precisión obtenida en el seguimiento de trayectoria es bastante buena con ambos métodos, y se encuentran señales de entrada muy cerca del comportamiento óptimo, siempre teniendo en cuenta la presencia de ruido y que cada iteración se ejecuta puramente bucle abierto.

El algoritmo de control por anulación de error aprende adecuadamente aunque converge un poco más lentamente. Si bien esto podría deberse a que no se han encontrado unos parámetros de aprendizaje tan adecuados como los que se han utilizado con el otro algoritmo.

4.4 Consideraciones comunes

Los resultados obtenidos son muy sensibles a los parámetros aplicados al sistema de control: Cambiar uno de los valores puede hacer que el sistema deje de funcionar por completo, por lo que estos deben elegirse cuidadosamente.

Por otra parte, las restricciones no sirven solo para implementar limitaciones de los sistemas reales sino que en ocasiones mejoran la eficacia del aprendizaje.

Debido a la naturaleza aleatoria del ruido introducido, la repetición del experimento con los mismos parámetros no siempre da los mismos resultados. Al respecto cabe mencionar que la iteración que mejores resultados proporciona no siempre es la última, y que hay ocasiones en las que el sistema tarda unas pocas iteraciones de más o de menos en dar resultados buenos.

También se ha comprobado que una misma señal de entrada suele dar resultados diferentes por el ruido aditivo que se aplica antes de introducirse en el sistema. Estas diferencias debidas al ruido llegan hasta el punto de que con una misma entrada el péndulo puede acabar cayendo al final del experimento hacia cualquiera de los dos lados.

Por otra parte, las referencias de las cuatro salidas se han diseñado a mano, y debido a la correlación entre algunos de los estados, es muy probable que no se puedan cumplir todas a la perfección, por lo que el control tiene que comprometer alguna y un resultado perfecto, aún sin ruido, nunca sería posible.

Utilizando únicamente Matlab en un portátil de prestaciones estándar (4 núcleos @ 2.60 GHz; 8 GB RAM), ejecutando el programa con diez iteraciones, y todas las restricciones activadas:

- El programa de control con referencias absolutas (4.1) emplea un total de 31 segundos en los procesos de optimización para calcular las nuevas entradas con trayectorias de 200 muestras (2 segundos de duración con 0,01 segundos de tiempo de muestreo), lo que implica que cada proceso de optimización tarda 3,1 segundos de media. El proceso de formación de las matrices, que se ejecuta al comienzo una sola vez, tarda aproximadamente 4,8 segundos.
- El programa de control por anulación del error (4.2), gracias a haber aplicado la estrategia de horizonte extensible, emplea un total de 15,6 segundos en los procesos de optimización para calcular las entradas con 200 muestras, dado que para diez iteraciones resuelve tan solo nueve problemas de optimización, estos tardan 1,7 segundos de media. La formación de matrices previa tarda unos 4 segundos para 200 muestras, y todo el proceso de linealización 7,6 segundos.

5 Conclusiones

Se han construido dos métodos de control por aprendizaje iterativo basados en procesos de optimización tomando como referencia los trabajos [10], [11] y [12]. Ambos sistemas de control utilizan métodos óptimos tanto en la estimación del error en las trayectorias como en el cálculo de las señales de entrada. Este segundo cálculo se deriva de la teoría de optimización matemática y se implementa mediante técnicas de optimización convexa de última generación que permiten definición de límites en las entradas y las salidas, y ponderación de salidas y partes de las trayectorias. La estimación del error se realiza mediante un filtro de Kalman modificado para adaptarse a las necesidades del problema.

El primero de los métodos es una reproducción directa del trabajo presentado en [10]. Mientras que el segundo método utiliza una forma distinta de linealización del sistema, y trata de otra forma las señales de entrada y las referencias de las trayectorias.

Se han escrito desde cero dos programas que implementan los sistemas de control al completo. Las modificaciones necesarias para trabajar con sistemas controlados distintos con diferente número de estados, entradas o salidas son mínimas.

Se ha construido un modelo simulable de un péndulo sobre un carro móvil, sobre el que se han aplicado y probado los sistemas de control.

Se ha estudiado la linealización de sistemas en espacio de estados. Se han desarrollado dos procesos de linealización sobre el péndulo móvil, que es un sistema altamente no lineal. Uno de ellos deducido en su totalidad de las ecuaciones de equilibrio dinámico del sistema, el cual se ha utilizado exitosamente con uno de los métodos de control. Y el otro basado en técnicas de identificación de sistemas, aplicadas sobre la trayectoria deseada, que también se ha utilizado, con el otro método de control.

Además de estas dos opciones, existe un algoritmo propuesto en [11], relativamente complicado, que estudia las variaciones en las salidas provocadas por pequeñas variaciones en las entradas alrededor de una entrada nominal, todo esto a través de un bucle que puede requerir cientos o miles de simulaciones. Los dos métodos de linealización utilizados son más simples y rápidos que este.

El algoritmo de control por anulación de error, que se ha tomado de [10], parte de una entrada aproximada preexistente, y el problema de optimización trabaja únicamente sobre el error, y con referencias nulas, para tratar de eliminarlo. Por su parte, el algoritmo de control con referencias absolutas (propio, si bien basado en [10], [11] y [12]) utiliza un modelo aproximado del sistema para calcular, de forma absoluta, las entradas que, aplicadas a sistema real, provocan que este se

comporte siguiendo las referencias. En las iteraciones posteriores, ambos algoritmos utilizan el error estimado de las medidas de ejecuciones previas para mejorar la precisión.

Se han ejecutado satisfactoriamente ambos algoritmos de control sobre el modelo simulable del péndulo móvil: Ambos sistemas de control aprenden satisfactoriamente y consiguen realizar el seguimiento de trayectoria en bucle abierto.

Las diferencias entre ambos métodos traen además una serie de consecuencias:

- El control propio desarrollado en este trabajo (control con referencias absolutas) puede funcionar con una linealización del sistema más simple y rápida de obtener que la usada con el control por anulación de error. Si bien para sistemas extremadamente no lineales con trayectorias largas, el control propio desarrollado en este trabajo podría complicar la obtención de una linealización utilizable.
- En sistemas donde una aproximación lineal buena puede ser obtenida analíticamente, el control propio también puede utilizarla, sin necesidad de recurrir a identificación de sistemas.
- Con el control propio desarrollado no necesario proporcionar una entrada inicial aproximada al objetivo.
- En un sistema tan no lineal como el péndulo, si se modifica mucho el sistema real, por ejemplo, triplicando el peso del péndulo, es necesario obtener un nuevo sistema lineal para el control propio (control con referencias absolutas). Mientras tanto, el control por anulación de error es capaz de aprender la trayectoria adecuadamente, con el único inconveniente de necesitar más iteraciones para ello.
- El control con referencias absolutas responde bastante bien ante trayectorias largas, como por ejemplo los dos segundos del apartado 4.1. Por el contrario, el control por anulación de error, con trayectorias demasiado largas, requiere la utilización de la estrategia de horizonte extensible para converger.

Al respecto de la estrategia de control se han probado su aplicabilidad y fiabilidad al dominar con éxito el balanceo del péndulo móvil, sistema no lineal y subactuado, usando únicamente control de bucle abierto. Se ha comprobado también que con la capacidad computacional actual, la resolución de los problemas de optimización es relativamente rápida, por lo que utilizarla es una posibilidad real, viable y barata en el aspecto material.

En términos generales se han cumplido los objetivos del trabajo.

6 Referencias

[1] UNIZAR. Historia de la Ingeniería de Control. automata.cps.unizar.es [online]. 2020. [Consultado el 14 de octubre de 2020]. Disponible en: http://automata.cps.unizar.es/Historia/Webs/primeros_ejemplos_historicos_de_.htm

[2] AUSLANDER, D. M. Evolutions In automatic control. 1971.

[3] BELLMAN, Richard. Dynamic programming. RAND CORP SANTA MONICA CA, 1956. p. 83

[4] YANES LUIS, Samuel; CASAS ROMÁN, Antón; ARIAS SÁNCHEZ, Isidro. Diseño e Implementación de un ACC mediante MPC y fusión sensorial.

[5] VARGAS LARA, José M. Control predictivo multivariable: evolución histórica y conceptos.

[6] CAMACHO, E.F. Comercial MPC. 2009.

[7] CAMACHO, E.F. Academic MPC. 2009.

[8] FERNANDEZ-CAMACHO, Eduardo; BORDONS-ALBA, Carlos. Model predictive control in the process industry. Springer London, 1995.

[9] BEHRENDT, Martin. A basic working principle of Model Predictive Control [Imagen]. Wikimedia. 2009. Disponible en https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg

[10] SCHÖLLIG, Angela; D'ANDREA, Raffaello. Optimization-based iterative learning control for trajectory tracking. En 2009 European Control Conference (ECC). IEEE, 2009. p. 1505-1510.

[11] SCHOELLIG, Angela P.; MUELLER, Fabian L.; D'ANDREA, Raffaello. Optimization-based iterative learning for precise quadrocopter trajectory tracking. Autonomous Robots, 2012, vol. 33, no 1-2, p. 103-127.

[12] MUELLER, Fabian L.; SCHOELLIG, Angela P.; D'ANDREA, Raffaello. Iterative learning of feedforward corrections for high-performance tracking. En 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012. p. 3276-3281.

[13] MOTCHENBACHER, Curtis D.; CONNELLY, Joseph Alvin. Low noise electronic system design. Wiley, 1993.

[14] CHUI, Charles K.; CHEN, Guanrong. Kalman filtering. Springer International Publishing, 2017.

[15] HUMPHERYS, Jeffrey; REDD, Preston; WEST, Jeremy. A fresh look at the Kalman filter. SIAM review, 2012, vol. 54, no 4, p. 801-823.

[16] MATHWORKS ESPAÑA. Kalman Filtering. es.mathworks.com [online]. 2020. [Consultado el 2 de octubre de 2020]. Disponible en: https://es.mathworks.com/help/control/ug/kalman-filtering.html

[17] HYNDMAN, Rob J. Moving averages. 2009.

[18] MATHWORKS ESPAÑA. Suavizado de señales. es.mathworks.com [online]. 2020. [Consultado el 8 de octubre de 2020]. Disponible en: https://es.mathworks.com/help/signal/examples/signal-smoothing.html

[19] MATHWORKS ESPAÑA. Filtrado digital de fase cero. es.mathworks.com [online]. 2020. [Consultado el 8 de octubre de 2020]. Disponible en: https://es.mathworks.com/help/signal/ref/filtfilt.html

[20] MATHWORKS ESPAÑA. Estimate state-space model using time-domain or frequency-domain data - MATLAB ssest. es.mathworks.com [online]. 2020. [Consultado el 11 de noviembre de 2020]. Disponible en: https://es.mathworks.com/help/ident/ref/ssest.html

Trabajo Fin de Máster
Máster Universitario en Ingeniería en Electrónica, Robótica y Automática

Control Mediante Aprendizaje Iterativo
Anexo A: Códigos

Autor:
Antón Casas Román

Tutor:
José Ángel Acosta Rodríguez
Profesor titular

Departamento de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Índice

1 ILC mediante optimización cuadrática básico

```
%% ILC mediante optimización cuadrática
%
% Desarrollado por Antón Casas Román.
%
% El sistema de control por aprendizaje iterativo calcula una entrada
% óptima en bucle abierto para que un sistema siga determinadas
% referencias. En base al error obtenido se actualiza la señal de entrada
% mediante optimización offline para la siguiente iteración.
%
% Con restricciones en las entradas y las salidas.
% Hay ruido aditivo a la salida y a la entrada, y se incluye un filtro de
% Kalman basado en iteraciones para reducirlo.
% Incluye también un filtro paso bajo sobre la señal de entrada.
%
% En este programa se aplica a un sistema lineal invariante en el tiempo.
% Con este programa el modelo *estimado* debe ser invariante en el tiempo.

n_iter = 5; % Número de iteraciones de optimización
t_final = 200; % Longitud del experimento (en nº de muestras). max: 240
epsilon = 0.02; % Confianza inicial en las medidas frente al modelo
x_init = [0; 1; 0]; % Estados iniciales

%% System

% Sistema estimado
A = [0.9 0.01 0.03; 0.02 0.85 0.03; 0.1 0.1 0.1];
B = [1 0.1; 0 1; 0.5 0.5];
C = [1 0 0; 0 1 0];

nx = size(A,1); %Número de estados
nu = size(B,2); %Número de entradas
ny = size(C,1); %Número de salidas

% Sistema real
Ar = A+rand(nx)*0.05;
Br = B+rand(nx,nu)*0.05;
Cr = C;

%% Referencias

ref(1,:) =
(sin(0:0.05:12-0.05)/2+0.6).*(cos((0:0.025:6-0.025)+pi)/4+0.5).*tanh(0:0.1:24-
0.1);
ref(2,:) = 1-((sin(0:0.0625:15-0.0625)/2.3+0.5).*tanh(0:4/24:40-4/24));
ref(3,:) = zeros(1,240);

if t_final > size(ref,2)
    t_final = size(ref,2);
end

long = t_final - ny;
```

```
% referencias
w = ref(1:nx*long)';

%% Restricciones

uflag = 1; % Entradas
yflag = 1; % Salidas

u_max = 1*[0.1; 0.15];
u_min = -1*[0.1; 0.1];

y_max = 1*[0.8; 1];
y_min = -1*[0; 0];

%% Ruido

m = 0.01; % Desviación típica
M = m*eye(ny); % Aditivo a la salida
M_ = 1.8*m*eye(ny*long); % Multiplicamos por 1.8 para hacer una estimación
inexacta de la covarianza del ruido a la entrada + a la salida
Omega = diag(epsilon*ones(nx*long,1));

%% Offline part

x = zeros(nx, t_final);
u = zeros(nu, t_final, n_iter);
y = zeros(ny, t_final, n_iter);
e = zeros(ny, t_final);
ecm = zeros(n_iter, ny);

x(:,1) = x_init; %Estados iniciales

% construct d
d = zeros(nx*long, n_iter);
d_ = zeros(nx,long);
d_(:,1) = x(:,1);
for i = 1:long-1
    d_(:,1+i) = A*d_(:,i);
end
d(:,1) = d_(1:end); % Primera estimación del vector d

P = d(:,1)*d(:,1)'; % Primera estimación de la matriz P (filtro de ruido)

% construct F
F = zeros(long*nx, long*nu);
for i = 1:long
    for j = 1:long
        if i > j
            F( (i-1)*nx+1:i*nx, (j-1)*nu+1:j*nu ) = A^(i-j-1)*B;
        end
    end
end

% construct G
G = zeros(long*ny, long*nx);
```

```
for i = 1:long
    for j = 1:long
        if i == j
            G( (i-1)*ny+1:i*ny, (j-1)*nx+1:j*nx ) = C;
        end
    end
end

% weighting matrices
r  = [1; 1; 0]; % Cuánto importa cada salida respecto de la otra
R_ = repmat(r, long, 1);
R_ = diag(R_);

q  = [0; 0]; % Penalización a las entradas
Q_ = repmat(q, long, 1);
Q_ = diag(Q_);

% Transformar para quadprog
Fqp = 2*((F'*R_*F)+Q_);
Fqp = (Fqp+Fqp')/2;

[num,den] = butter(1,0.3); % Filtro butterworth de paso bajo de orden 1

%% Bucle de iteraciones

for iteration = 1:n_iter

    % OPTIMIZACIÓN %

    % Restricciones
    Rc = [];
    b = [];

    if uflag == 1
        Rc = [Rc; eye(nu*long); -1*eye(nu*long)];
        b = [b; repmat(u_max, long, 1); -1*repmat(u_min, long, 1)];
    end

    if yflag == 1
        Rc = [Rc; G*F; -1*G*F];
        b = [b; repmat(y_max, long, 1) - G*d(:,iteration); -1*repmat(y_min,
long, 1) + G*d(:,iteration)];
    end

    dqp = (2*(d(:,iteration) - w)'*R_*F)';

    % Calcular señal de control:
    temp = quadprog(Fqp, dqp, Rc, b);

    % Ordenar la señal en el vector u:
    for t = 1:long
        u(:,t+1,iteration) = temp( nu*t-nu+1 : nu*t )';
    end

    % Filtrado de la señal de entrada para darle un poco de suavidad y
```

```
    % disminuir los artificios creados por el ruido.
    for i = 1:nu
        u(i,:,iteration) = filtfilt(num,den,u(i,:,iteration)); % Filtrado de
fase 0 (orden 2*1)
    end

    %% SIMULACIÓN

    for t = 2:t_final
        x(:,t) = Ar*x(:,t-1) + Br*( u(:,t-1,iteration) + M*randn(2,1) );
        y(:,t,iteration) = Cr*x(:,t) + M*randn(2,1);
    end

    %% Errores

    % Medición de errores
    for i = 1:ny
        % Cálculo del error en las trayectorias
        e(i,:) = (y(i,:,iteration) - ref(i,1:t_final));
        % Cálculo del error cuadrático medio
        ecm(iteration,i) = e(i,:)*e(i,:)'/t_final;
    end

    % Actualización de la ganancia K
    P = P + Omega;
    Theta = G*P*G' + M_;
    K = P*G'/Theta; % P*G'*inv(Theta)
    P = ( eye(nx*(long)) - K*G )*P; % (I-KG)*P
    Omega = Omega/10;

    % Actualización del vector d
    y_aux = y(:,:,iteration);
    e_ = (y_aux(1:long*ny)' - G*F*temp);
    d(:, iteration + 1) = d(:, iteration) + K*(e_ -G*d(:, iteration)); % d = d +
K(y -G*d -(GF+H)*u

end

% Trasposición de las matrices de entradas y salidas para los comandos plot
u = permute(u, [2 1 3]);
y = permute(y, [2 1 3]);

%% RESULTADOS

%Salida
figure(1)
subplot(1,2,1), hold on
plot(ref(1,:),'LineWidth',2);
for iteration = 1:n_iter
    plot(y(:,1,iteration),'LineWidth',1);
end
legend('Ref1','j=1','j=2','j=3');
xlabel('Muestras'); ylabel('Salida 1 del sistema');
grid minor; xlim([0 t_final]);
hold off;
```

```
subplot(1,2,2);
plot(ref(2,:),'LineWidth',2); hold on
for iteration = 1:n_iter
    plot(y(:,2,iteration),'LineWidth',1);
end
legend('Ref2','j=1','j=2','j=3');
xlabel('Muestras'); ylabel('Salida 2 del sistema');
grid minor;  xlim([0 t_final]);
hold off;

%Señales de control
figure(2)
subplot(1,2,1)
hold on
for iteration = 1:n_iter
    plot(u(:,1,iteration));
end
legend('j=1','j=2','j=3')
xlabel('Muestras'); ylabel('Señal de control 1');
grid minor;

subplot(1,2,2)
hold on
for iteration = 1:n_iter
    plot(u(:,2,iteration));
end
legend('j=1','j=2','j=3')
xlabel('Muestras'); ylabel('Señal de control 2');
grid minor;
```

2 ILC para el péndulo móvil con seguimiento de referencias absolutas

```
%% ILC mediante optimización cuadrática para péndulo móvil
%
% Desarrollado por Antón Casas Román.
%
% El sistema de control por aprendizaje iterativo calcula una entrada
% óptima en bucle abierto para que un sistema siga determinadas
% referencias. En base al error obtenido se actualiza la señal de entrada
% mediante optimización offline para la siguiente iteración.
%
% Con restricciones en las entradas y las salidas.
% Hay ruido aditivo a la salida y a la entrada, y se incluye un filtro de
% Kalman basado en iteraciones para reducirlo.
% Incluye también un filtro paso bajo sobre la señal de entrada.
%
% Para el péndulo implementado en Simscape 'pendulo_sm2.slx'
% Se ha linealizado el sistema a lo largo de la trayectoria.
% Aprende sobre las trayectorias de todos los estados.

load('datos_iniciales_c.mat') % Incluye el modelo linealizado y las referencias.
n_iter = 7; % Número de iteraciones de optimización
t_final = 200; % Longitud del experimento (en nº de muestras). maximo dado por
la referencia
epsilon = 1e3; % Confianza inicial en las medidas frente al modelo
x_init = [-pi; 0; 0; 0]; % Estados iniciales

%% Sistema estimado

A(:,:,1:t_final) = repmat(ss3.A,1,1,t_final);
B(:,:,1:t_final) = repmat(ss3.B,1,1,t_final);
C(:,:,1:t_final) = repmat(ss3.C,1,1,t_final);

nx = size(A,1); %Número de estados
nu = size(B,2); %Número de entradas
ny = size(C,1); %Número de salidas

%% Referencias

long = t_final - ny;

% referencias
w = ref(1:nx*long)';

%% Restricciones

uflag = 1; % Entradas
yflag = 1; % Salidas

u_max =   100;
u_min =  -100;
```

```
y_max =  1*[4*pi; 80; 0.5; 20];
y_min = -1*[8*pi; 80; 0.5; 20];

%% Ruido

m = 0.03; % Desviación típica
M = m*eye(ny); % Aditivo a la salida
M_ = 1.8*m*eye(ny*long); % Multiplicamos por 1.8 para hacer una estimación
inexacta de la covarianza del ruido a la entrada + a la salida
Omega = diag(epsilon*ones(nx*long,1));

%% Offline part

x = zeros(nx, t_final);
u = zeros(nu, t_final, n_iter);
y = zeros(ny, t_final, n_iter);
e = zeros(ny, t_final);
ecm = zeros(n_iter, ny);

x(:,1) = x_init; %Estados iniciales

% construct d
d = zeros(nx*(t_final - ny), n_iter);
d_ = zeros(nx,long);
d_(:,1) = x(:,1);
for i = 1:long-1
    d_(:,1+i) = A(:,:,i)*d_(:,i);
end
d(:,1) = d_(1:end); % Primera estimación del vector d

P = d(:,1)*d(:,1)'; % Primera estimación de la matriz P (filtro de ruido)

% construct F
F = zeros(long*nx, long*nu);
for i = 1:long
    for j = 1:long
        if i > j
            F( (i-1)*nx+1:i*nx, (j-1)*nu+1:j*nu ) = B(:,:,j);
            for k = j+1:i-1
                F( (i-1)*nx+1:i*nx, (j-1)*nu+1:j*nu ) = A(:,:,k)*F( (i-
1)*nx+1:i*nx, (j-1)*nu+1:j*nu );
            end
        end
    end
end

% construct G
G = zeros(long*ny, long*nx);
for i = 1:long
    for j = 1:long
        if i == j
            G( (i-1)*ny+1:i*ny, (j-1)*nx+1:j*nx ) = C(:,:,i);
        end
    end
end
```

```
% weighting matrices
r  = [1; 1; 1; 100]; % Cuánto importa cada salida respecto de la otra
R_ = repmat(r, long-floor(long/8), 1);
R_ = [R_; repmat(1*r, floor(long/8), 1)]; % Damos más importancia al final
R_ = diag(R_);

q  = 0; % Cuánto se confía en el modelo frente a las medidas
Q_ = repmat(q, long, 1);
Q_ = diag(Q_);

% Transformar para quadprog
Fqp = 2*((F'*R_*F)+Q_);
Fqp = (Fqp+Fqp')/2;

[num,den] = butter(1,0.3); % Filtro butterworth de paso bajo de orden 1

%% Bucle de iteraciones

for iteration = 1:n_iter

    % OPTIMIZACIÓN %

    % Restricciones
    Rc = [];
    b = [];

    if uflag == 1
        Rc = [Rc; eye(nu*long); -1*eye(nu*long)];
        b = [b; repmat(u_max, long, 1); -1*repmat(u_min, long, 1)];
    end

    if yflag == 1
        Rc = [Rc; G*F; -1*G*F];
        b = [b; repmat(y_max, long, 1) - G*d(:,iteration); -1*repmat(y_min,
long, 1) + G*d(:,iteration)];
    end

    dqp = (2*(d(:,iteration) - w)'*R_*F)';

    % Calcular señal de control:
    temp = quadprog(Fqp, dqp, Rc, b);

    % Ordenar la señal en el vector u:
    for t = 1:long
        u(:,t+1,iteration) = temp( nu*t-nu+1 : nu*t )';
    end

    % Filtrado de la señal de entrada para darle un poco de suavidad y
    % disminuir los artificios creados por el ruido.
    for i = 1:nu
        u(i,:,iteration) = filtfilt(num,den,u(i,:,iteration)); % Filtrado de
fase 0 (orden 2*1)
    end
```

```
    %% SIMULACIÓN

    out_sim = sim('pendulo_sm2');
    y(:,:,iteration) = out_sim.salida';

    %% Errores

    % Medición de errores
    for i = 1:ny
        % Cálculo del error en las trayectorias
        e(i,:) = (y(i,:,iteration) - ref(i,1:t_final));
        % Cálculo del error cuadrático medio
        ecm(iteration,i) = e(i,:)*e(i,:)'/t_final;
    end

    % Actualización de la ganancia K
    P = P + Omega;
    Theta = G*P*G' + M_;
    K = P*G'/Theta; % P*G'*inv(Theta)
    P = ( eye(nx*(long)) - K*G )*P; % (I-KG)*P
    Omega = Omega/10;

    % Actualización del vector d
    y_aux = y(:,:,iteration);
    e_ = ( y_aux(1+ny:end-12)' - G*F*temp );
    d(:, iteration + 1) = d(:, iteration) + K*(e_ -G*d(:, iteration)); % d = d +
K(y -G*d -(GF+H)*u

    pause(3) % Pausa para ver la animación

end

% Trasposición de las matrices de entradas y salidas para los comandos plot
u = permute(u, [2 1 3]);
y = permute(y, [2 1 3]);

%% RESULTADOS

%Salida
time_vector = 0:0.01:(t_final-1)*0.01;

figure(1)
subplot(2,2,1)
hold on
plot(time_vector, ref(1,1:t_final),'LineWidth',2);
for iteration = 1:n_iter
    plot(time_vector, y(:,1,iteration),'LineWidth',1);
end
legend('Ref1','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 1 (rad)');
grid minor; xlim([0 t_final*0.01]);
hold off;

subplot(2,2,2);
plot(time_vector, ref(2,1:t_final),'LineWidth',2); hold on
```

```
for iteration = 1:n_iter
    plot(time_vector, y(:,2,iteration),'LineWidth',1);
end
legend('Ref2','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 2 (rad/s)');
grid minor;  xlim([0 t_final*0.01]);
hold off;

subplot(2,2,3)
hold on
plot(time_vector, ref(3,1:t_final),'LineWidth',2);
for iteration = 1:n_iter
    plot(time_vector, y(:,3,iteration),'LineWidth',1);
end
legend('Ref1','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 3 (m)');
grid minor; xlim([0 t_final*0.01]);
hold off;

subplot(2,2,4);
plot(time_vector, ref(4,1:t_final),'LineWidth',2); hold on
for iteration = 1:n_iter
    plot(time_vector, y(:,4,iteration),'LineWidth',1);
end
legend('Ref2','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 4 (m/s)');
grid minor;  xlim([0 t_final*0.01]);
hold off;

%Señal de control
figure(2)
hold on
for iteration = 1:n_iter
    plot(u(:,1,iteration));
end
legend('j=1','j=2','j=3')
xlabel('Tiempo (s)'); ylabel('Señal de control (N)');
grid minor;
```

3 Linealización a lo largo de la trayectoria

```
%% Cálculo de las matrices linealizadas A y B del péndulo
% en distintas partes de la trayectoria. Como resultado se obtiene un
% sistema lineal variante en el tiempo que explica el comportamiento a lo
% largo de la trayectoria predefinida. El sistema tratado es un péndulo
% unido a un carro móvil a lo largo de un eje sobre el que se ejerce una
% fuerza.

%% Declaración de variables y definición de las ecuaciones

nx = 4; nu = 1;

syms phi0 phi1 x0 x1
syms F mp mc lp g

phi0A = phi1;
phi1A = ( ((mc+mp)*g/mp - lp*phi1^2*cos(phi0))*sin(phi0) ) / ( lp*(mc/mp +
(sin(phi0))^2) );
x0A = x1;
x1A = ( - g*sin(phi0)*cos(phi0) + lp*phi1^2*sin(phi0) ) / ( mc/mp +
(sin(phi0))^2 );

phi0B = 0;
phi1B = ( -cos(phi0)*F/mp ) / ( lp*(mc/mp + (sin(phi0))^2) );
x0B = 0;
x1B = (F/mp) / ( mc/mp + (sin(phi0))^2 );

%% Cálculo de las matrices jacobianas

A = jacobian([ phi0A phi1A x0A x1A ], [phi0 phi1 x0 x1]);
B = jacobian([ phi0B phi1B x0B x1B ], [F]);
C = eye(nx);

%% Sustitución de valores

A_tv = zeros(nx, nx, t_final);
B_tv = zeros(nx, nu, t_final);
C_tv = zeros(nx, nx, t_final);

% Constantes
lp = 0.30;
mp = 0.200;
mc = 1.5;
g = -9.81;

for i = 1:t_final
    % Sustitución de valores en cada instante de la trayectoria de referencia
    phi0 = ref(1, i);
    phi1 = ref(2, i);
    x0 = ref(3, i);
    x1 = ref(4, i);

    A0 = double(vpa(subs(A))); % Sustitución
```

```
    B0 = double(vpa(subs(B)));
    C0 = C;
    [A_tv(:,:,i), B_tv(:,:,i), C_tv(:,:,i), ~, ~] = ssdata(c2d( ss(A0,B0,C0,[]),
0.01)); % Discretización
end

A = A_tv;
B = B_tv;
C = C_tv;
```

4 ILC para el péndulo móvil mediante anulación del error y horizonte extensible

```
%% ILC mediante optimización cuadrática para péndulo móvil con estrategia de
anulación del error
%
% Desarrollado por Antón Casas Román.
%
% El sistema de control por aprendizaje iterativo utiliza una entrada
% inicial conocida en bucle abierto para que un sistema siga determinadas
% referencias. En base al error obtenido se actualiza la señal de entrada
% mediante optimización offline para la siguiente iteración.
% La optimización actúa únicamente sobre el vector de error. El resultado
% es la variación aditiva a aplicar sobre la señal de entrada.
%
% Con restricciones en las entradas y las salidas.
% Hay ruido aditivo a la salida y a la entrada, y se incluye un filtro de
% Kalman basado en iteraciones para reducirlo.
% Incluye también un filtro paso bajo sobre la señal de entrada.
% Utiliza la estrategia de horizonte extensible.
%
% Para el péndulo implementado en Simscape 'pendulo_sm3.slx'
% Se utiliza un sistema variante en el tiempo formado con linealizaciones
% por puntos de la trayectoria, todas centradas en cero, ya que el objetivo
% es que el error acabe centrado en cero.
% Aprende sobre las trayectorias de todos los estados.

load('datos_iniciales_f.mat') % Incluye las referencias y la entrada inicial
n_iter = 10; % Número de iteraciones de optimización
t_final = 200; % Longitud del experimento (en n° de muestras). maximo dado por
la referencia
epsilon = 1e-2; % Confianza inicial en las medidas frente al modelo
x_init = [0; 0; 0; 0]; % Estados iniciales

%% Sistema estimado

jacobiano2 %Script en el que se realiza la linealización

nx = size(A,1); %Número de estados
nu = size(B,2); %Número de entradas
ny = size(C,1); %Número de salidas

%% Referencias

long = t_final - ny;

w = ref(1:nx*long)';

%% Restricciones

uflag = 1; % Entradas
yflag = 1; % Salidas
```

```
u_max =  30;
u_min = -30;

y_max =  1*[4*pi; 80; 0.65; 20];
y_min = -1*[8*pi; 80; 0.65; 20];

%% Ruido

m = 0.03; % Desviación típica
M = m*eye(ny); % Aditivo a la salida
M_ = 1.8*m*eye(ny*long); % Multiplicamos por 1.8 para hacer una estimación
inexacta de la covarianza del ruido a la entrada + a la salida
Omega = diag(epsilon*ones(nx*long,1));

%% Offline part

x = zeros(nx, t_final);
u = zeros(nu, t_final, n_iter);
y = zeros(ny, t_final, n_iter);
e = zeros(ny, t_final);
ecm = zeros(n_iter, ny);

u(:,:,1) = u_init(1:t_final); % Entrada inicial
x(:,1) = x_init; % Estados iniciales

% construct d
d = zeros(nx*long, n_iter);
d_ = zeros(nx,long);
d_(:,1) = x(:,1);
for i = 1:long-1
    d_(:,1+i) = A(:,:,i)*d_(:,i);
end
d(:,1) = d_(1:end); % Primera estimación del vector d

P = d(:,1)*d(:,1)'; % Primera estimación de la matriz P (filtro de ruido)

% construct F
F = zeros(long*nx, long*nu);
for i = 1:long
    for j = 1:long
        if i > j
            F( (i-1)*nx+1:i*nx, (j-1)*nu+1:j*nu ) = B(:,:,j);
            for k = j+1:i-1
                F( (i-1)*nx+1:i*nx, (j-1)*nu+1:j*nu ) = A(:,:,k)*F( (i-
1)*nx+1:i*nx, (j-1)*nu+1:j*nu );
            end
        end
    end
end

% construct G
G = zeros(long*ny, long*nx);
for i = 1:long
    for j = 1:long
        if i == j
```

```
            G( (i-1)*ny+1:i*ny, (j-1)*nx+1:j*nx ) = C(:,:,i);
        end
    end
end

% weighting matrices
r  = [1; 1; 0.5; 1]; % Cuánto importa cada salida respecto de la otra
R_ = repmat(r, long-floor(long/4), 1);
R_ = [R_; repmat(1*r, floor(long/4), 1)]; % Damos más importancia al final
R_ = diag(R_);

q  = 1e-9; % Penalización a las entradas
Q_ = repmat(q, long, 1);
Q_ = diag(Q_);

% Transformar para quadprog
Fqp = 2*((F'*R_*F)+Q_);
Fqp = (Fqp+Fqp')/2;

[num,den] = butter(1,0.3); % Filtro butterworth de paso bajo de orden 1

%% Bucle de iteraciones

for iteration = 1:n_iter

    % Filtrado de la señal de entrada para darle un poco de suavidad y
    % disminuir los artificios creados por el ruido.
    for i = 1:nu
        u(i,:,iteration) = filtfilt(num,den,u(i,:,iteration)); % Filtrado de
fase 0 (orden 2*1)
    end

    % Simulación
    out_sim = sim('pendulo_sm3');
    pause(3) % Pausa para ver la animación
    y(:,:,iteration) = out_sim.salida';

    % Medición de errores
    for i = 1:ny
        % Cálculo del error en las trayectorias
        e(i,:) = (y(i,:,iteration) - ref(i,1:t_final));
        % Cálculo del error cuadrático medio
        ecm(iteration,i) = e(i,1:150)*e(i,1:150)'/150;
    end

    % Extensión de horizonte
    llimit = find( abs(e(1,:)') > 1 ,1); % Momento en que el error se hace
demasiado grande
    if isempty(llimit), llimit = long;
    elseif llimit > long, llimit = long; end
    % Re-creación de la matriz Omega
    Omega = diag( Omega(1)*[ones(nx*llimit,1); zeros(nx*(long-llimit),1)] );
%     P = diag( P(1)*[ones(nx*llimit,1); zeros(nx*(long-llimit),1)] );

    % Actualización de la ganancia K
```

```
    P = P + Omega;
    Theta = G*P*G' + M_;
    K = P*G'/Theta; % P*G'*inv(Theta)
    P = ( eye(nx*(long)) - K*G )*P; % (I-KG)*P
    Omega = Omega/2;

    % Actualización del vector d
    y_aux = y(:,:,iteration);
    e_ = ( y_aux(1:long*ny)' - w );
    d(:, iteration + 1) = d(:, iteration) + K*-(e_ +G*d(:, iteration)); % d = d
+ K(y -G*d -(GF+H)*u

    % Optimización
    if iteration < n_iter

        % Constraints
        Rc = [];
        b = [];

        if uflag == 1
            Rc = [Rc; eye(nu*long); -1*eye(nu*long)];
            b = [b; repmat(u_max, long, 1)-u(:,1:long,1)'; -1*( repmat(u_min,
long, 1)-u(:,1:long,1)' )];
        end

        if yflag == 1
            Rc = [Rc; G*F; -1*G*F];
            b = [b; repmat(y_max, long, 1)-y_aux(1:long*nx)' ; -1*(repmat(y_min,
long, 1)-y_aux(1:long*nx)')];
        end

        dqp = (2*d(:,iteration+1)'*R_*F)';

        % Calcular señal de control
        temp = quadprog(Fqp, dqp, Rc, b);

        % Ordenar la señal en el vector u:
        for t = 1:long
            u(:,t+1, iteration+1 ) = u(:,t+1,1) + temp( nu*t-nu+1 : nu*t )';
        end
    end
end

% Trasposición de las matrices de entradas y salidas para los comandos plot
u = permute(u, [2 1 3]);
y = permute(y, [2 1 3]);

%% RESULTADOS

%Salida
time_vector = 0:0.01:(t_final-1)*0.01;

figure(1)
subplot(2,2,1)
hold on
```

```
plot(time_vector, ref(1,1:t_final),'LineWidth',2);
for iteration = 1:n_iter
    plot(time_vector, y(:,1,iteration),'LineWidth',1);
end
legend('Ref1','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 1 (rad)');
grid minor; xlim([0 t_final*0.01]);
hold off;

subplot(2,2,2);
plot(time_vector, ref(2,1:t_final),'LineWidth',2); hold on
for iteration = 1:n_iter
    plot(time_vector, y(:,2,iteration),'LineWidth',1);
end
legend('Ref2','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 2 (rad/s)');
grid minor;  xlim([0 t_final*0.01]);
hold off;

subplot(2,2,3)
hold on
plot(time_vector, ref(3,1:t_final),'LineWidth',2);
for iteration = 1:n_iter
    plot(time_vector, y(:,3,iteration),'LineWidth',1);
end
legend('Ref1','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 3 (m)');
grid minor; xlim([0 t_final*0.01]);
hold off;

subplot(2,2,4);
plot(time_vector, ref(4,1:t_final),'LineWidth',2); hold on
for iteration = 1:n_iter
    plot(time_vector, y(:,4,iteration),'LineWidth',1);
end
legend('Ref2','j=1','j=2','j=3');
xlabel('Tiempo (s)'); ylabel('Salida 4 (m/s)');
grid minor;  xlim([0 t_final*0.01]);
hold off;

%Señal de control
figure(2)
hold on
for iteration = 1:n_iter
    plot(u(:,1,iteration));
end
legend('j=1','j=2','j=3')
xlabel('Tiempo (s)'); ylabel('Señal de control (N)');
grid minor;
```