Trabajo de Fin de Máster
Máster Universitario en Ingeniería Industrial

Control Basado en Datos de un proceso de laboratorio

Autor: Gloria García Marín
Tutor: Daniel Rodríguez Ramírez

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

Sevilla, 2018

Trabajo Fin de Máster

Control Basado en Datos de un proceso de laboratorio

Autor:
Gloria García Marín

Tutor:
Daniel Rodríguez Ramírez

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Autor: Gloria García Marín

Tutor: Daniel Rodríguez Ramírez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

A cuantos han hecho posible este trabajo.

Gloria García Marín

Resumen

Este trabajo ha consistido en la implementación sobre un sistema del laboratorio, concretamente un Feedback Process Control Trainer 37-100 de un control basado en datos que había sido previamente probado en simulación. Para ello se han generado varias bases de datos utilizando distintas estrategias y realizado ensayos variando los parámetros de funcionamiento. El algoritmo original también ha sido modificado, añadiéndole un término que considera en la minimización cómo una cierta acción de control reduce el error.

Abstract

In this work a data driven controller has been tested in Feedback Process Control Trainer 37-100 lab procrocess. This controller had been previously tested in simulation.

In order to test the controller, several data bases have been created using different methods, as well as some trials have been implemented using different parameters.

The original algorithm performance has also been enhanced by adding a new term which considers how an action reduces the error.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Índice **xiv**

Índice de Figuras **xvi**

Notación **xix**

1 Introducción: Estado del Arte **1**

2 Integración Software y Hardware **11**
2.1 Software *12*
2.1.1 Matlab 12
2.1.2 Quarc 12
2.2 Hardware *16*
2.2.1 Process Control Trainer 37-100 16
2.2.2 Tarjeta de adquisición de datos NI-USB 6211 16
2.3 Conexionado *16*

primera etapa **19**

1. Base de datos de las trayectorias **21**
1.1 Características *21*
1.2 Implementación de la base de datos *21*
1.2.1 Identificación del sistema y cálculo de un controlador teórico 22
1.2.2 Implementación del controlador sobre el sistema 22
1.2.3 Validez de la trayectoria. 22
1.2.4 Generación automática de la base de datos. 22
1.3 Base de datos generadas *24*

2. Implementación del algoritmo **25**
2.1 Implementación en Quarc *27*
2.2 Implementación desde MatLab. *27*
2.2.1 Data Acquisition Toolbox 28
2.2.2 Restricciones de tiempo. 28
2.2.3 Condiciones iniciales. 28
2.2.4 Cambios realizados sobre la función. 29

3. Ensayos **31**
3.1 Control del sistema por el algoritmo *32*
Referencia dentro del rango de la base de datos 32
3.1.1 Referencia fuera del rango de la base de datos 32
3.2 Comportamiento frente a un aumento del número de candidatos *33*
3.2.1 Comportamiento frente a un aumento del número de candidatos 34
3.3 Sensibilidad al tamaño de la base de datos. *35*

Segunda etapa **37**

1. Base de datos de las trayectoria **39**
1.1 Implementación *39*
1.2 Criterio de selección de las trayectorias. *41*
1.3 Base de datos de regulación *42*

2. Algoritmo modificado **45**

3. Simulaciones Algoritmo en función de gamma. **47**

4. Implementación y Ensayos **49**
4.1 Implementación de un filtro: *49*
4.2 Ajuste del cero *50*
4.3 Seguimiento de una trayectoria *50*
4.4 Evolución del Sistema al cambiar el valor de Gamma. *51*
4.5 Rechazo de perturbaciones. *52*
4.5.1 Rechazo de perturbaciones usando la base de datos de regulación. *54*
4.6 Rechazo de perturbaciones generadas por software. *54*

5. Conclusiones **57**
5.1 Líneas de mejora *58*

Anexo I: Funciones Lanzar Quarc **59**
rtwbuild *59*
set_param *59*

Anexo II: Data adquisition Toolbox **60**
Daq *60*
daq.getDevices *60*
daq.createSession *60*
addAnalogInputChannel *60*
addAnalogOutputChannel *60*
outputSingleScan *61*
inputSingleScan *61*
queueOutputData *61*
StartForeground *61*

Anexo III: Código **63**
Generación de Base de dato: Ensayos *63*
Generación de Base de dato: Composición de matriz *64*
Ensayos: Ensayo de seguimiento tipo *65*
Ensayos: Ensayo con perturbaciones generadas por software. *67*
Matlab_NI: Inicialización y configuración de la conexión *69*

ÍNDICE DE FIGURAS

Fig 1: Feedback Process Control Trainer 37-100 11
Fig 2: Tarjeta de adquisición de datos NI USB-6211 detalle conexiones. 11
Fig 3: Tarjeta de adquisición de datos de datos NI USB-6211 12
Fig 4: Configuración Quarc Simulink: Tiempo 12
Fig 5:Configuración Quarc Simulink: Configuración de la tarjeta (1) 13
Fig 6:Configuración Quarc Simulink: Configuración de la tarjeta (2) 13
Fig 7: Configuración Quarc Simulink: Configuración de los canales 14
Fig 8: Ejecutar un modelo: Botón Build Model 14
Fig 9: Ejecutar un modelo: Botón "Connect to target" 14
Fig 10: Ejecutar un modelo: Botón "Run" 14
Fig 11: Errores Quarc 15
Fig 12: Errores Quarc: ventana informativa 16
Fig 13: Conexionado de los equipos 17
Fig 14: Esquema de control para realizar las trayectorias. 22
Fig 15: Base de datos generada 24
Fig 16: Base de datos de densidad menor. 24
Fig 17: Esquema de Simulink: Entradas y salidas de la función del algoritmo 27
Fig 18:Esquema de Simulink: Esquema de control de implementación del algoritmo 27
Fig 19: funcionamiento del algoritmo con condiciones iniciales 29
Fig 20: Funcionamiento del algoritmo sin condiciones iniciales 29
Fig 21: seguimiento 31
Fig 22: Funcionamiento para una misma referencia con distintos valores de Q 32
Fig 23: Funcionamiento para referencias fuera de la base de datos(1) 33
Fig 24 : Funcionamiento para referencias fuera de la base de datos (2) 33
Fig 25: Evolución del error frente al número de candidatos con referencias dentro del rango. 34
Fig 26: Evolución del error frente al número de candidatos con referencias dentro del rango. 34
Fig 27: Evolución del error frente al tamaño de la base de datos 35
Fig 28: Esquema de Simulink: Implementación de la base de datos segunda estrategia 39
Fig 29: Esquema de Simulink: Implementación de la base de datos segunda estrategia: Entradas a la función 40
Fig 30: referencias utilizadas 40
Fig 31: Trayectorias incluidas en la base de datos 2 41
Fig 32: Un ensayo completo. La trayectoria seleccionada en rojo 42
Fig 33:Un ensayo completo, trayectoria seleccionada, parte de la trayectoria que entra en la base de datos 42

Fig 34: Esquema de Simulink: Implementación base de datos de regulación. 43
Fig 35: Trayectorias incluidas en la base de datos de regulación 43
Fig36: tiempos de ejecución del algoritmo durante un ensayo 46
Fig 37: Simulación 1 47
Fig 38: Simulación 2 48
Fig 39: Comportamiento en bucle abierto frente a entradas en escalón constante. 49
Fig 40: Trayectorias con filtro y sin filtro 50
Fig 41: Ensayo de seguimiento de la trayectoria (1) 51
Fig 42: Ensayo de seguimiento de la trayectoria (2) 51
Fig 43: Comparación del funcionamiento del sistema con distintos gamma 52
Fig 44: Comparación del funcionamiento del sistema con distintos gamma (detalle) 52
Fig 45: Situación del actuador 53
Fig 46: Valor sin perturbar 53
Fig 47:Ejemplo de perturbación 53
Fig 48: Ensayo de perturbaciones (1) 53
Fig 49: Ensayo de perturbaciones (2) 54
Fig 50: Esquema implementado para generar perturbaciones en escalón. 54
Con este método es posible dar perturbaciones mayores al sistema. En los ensayos mostrados en la Fig 51 se han dado perturbaciones de hasta 2 voltios. 55
Fig 52: perturbación de 1.5, sin offset, comparada con una perturbación de 2. Número de candidatos 1000 55
Fig 53: Perturbación de amplitud 2 para distinto número de candidatos. 55
Fig 54: Fallos de condicionamiento del problema 57
Fig 55: Evolución para distintas bases de datos 58

Notación

$x_i(t)$	Lectura del sensor en el punto t de la trayectoria i
$x(t)$	Lectura del sensor en el tiempo de muestreo
$x_i(t-1)$	Lectura del sensor el tiempo de muestreo anterior al punto t de la trayectoria i
$x(t-1)$	Lectura del sensor en tiempo de muestreo anterior.
$u_i(t)$	Acción de control llevada a cabo por el controlador en el punto t de la trayectoria i
$u(t-1)$	Acción de control calculada por el algoritmo en el instante anterior
Ref_i	Referencia de la trayectoria i
Ref	Referencia del sistema
$y(t)$	Lectura actual filtrada
$y(t-1)$	Lectura filtrada en el tiempo de muestreo anterior
$data(t)$	Lectura actual sin filtrar

1 INTRODUCCIÓN: ESTADO DEL ARTE

En la mayoría de los métodos clásicos de control en primer lugar se modela o identifica un modelo del sistema que se desea controlar y el segundo lugar se diseña el controlador a partir de dicho modelo. Sin embargo, en todos los métodos de identificación es necesario realizar hipótesis y simplificaciones de manera que en el comportamiento del sistema no está fielmente retratado en modelo. De esta forma, cuanto más preciso es modelo que se desea obtener, más coste y esfuerzos deben ser invertidos en su obtención. Además, no hay ningún método eficiente para conseguir este modelo. Todo esto, sin tener en cuenta que en algunos casos existen parámetros en el modelo que varían muy rápido o que cambian a lo largo del tiempo.

En los últimos años ciertas industrias, como la siderúrgica, química, eléctrica, electrónica etc. Han desarrollado tecnologías y equipos a gran escala y, consecuentemente la producción se ha vuelto más compleja, motivo por el que métodos tradicionales de obtención de modelos se vuelven inviables, a la vez que, en muchos casos, generan y almacenan una gran cantidad de información de proceso. Esto hace que el control basado en datos cobre relevancia [1].

Bajo el nombre de control basado en datos existen distintas formas desde las que se ha abordado el problema que van desde métodos basados en modelos en el dominio de la frecuencia hasta aplicaciones de Big Data. En la mayoría de estos métodos se consigue llegar a un funcionamiento sin offset usando distintas estrategias. Este funcionamiento (sin offset) consiste una de las principales ventajas de controladores sencillos como los PI o PID, y es un objetivo de control importante.

Este trabajo consistirá en la implementación sobre una planta conocida de una estrategia de control basada en una base de datos obtenida mediante trayectorias en bucle cerrado sin offset propuesta en el artículo Data Driven Control: An Offset free approach [2].

El planteamiento del problema en [2] es el siguiente:

Sea un sistema:

$$\begin{aligned} x(t+1) &= A \cdot x(t) + B \cdot u(t) + w(t) \\ y(t) &= C \cdot x(t) + D \cdot u(t) + v(t) \end{aligned}$$

Donde x representa el estado del sistema, u, las acciones de control y, la salida del sistema, w y v representan las perturbaciones y A, B, C, D son las matrices del sistema.

En este artículo se propone una ley de control de la forma:

$$u(t) = \sum_{j \in S} \lambda_j \cdot u_j(0)$$

Donde S es el conjunto formado por el conjunto de puntos pertenecientes a las trayectorias de las trayectorias.

De esta manera se propone una acción de control está formada por una superposición de acciones de control llevadas a cabo en el pasado.

En consecuencia, el problema principal es el cálculo de los parámetros λ_j. Esta resolución se formula como un problema de optimización con la forma:

$$\lambda_j{}^* = \arg\min \sum_{j \in S} \lambda_j{}^2$$

s.a.

$$\sum_{j \in S} \lambda_j \cdot x_j(0) = x(t)$$

$$\sum_{j \in S} \lambda_j \cdot x_j(-1) = x(t-1)$$

$$\sum_{j \in S} \lambda_j \cdot u_j(-1) = u(t-1)$$

$$\sum_{j \in S} \lambda_j \cdot y_j(0) = y(t)$$

$$\sum_{j \in S} \lambda_j \cdot r_j = r$$

$$\sum_{j \in S} \lambda_j = 1$$

Donde las λ_j aplicadas son las óptimas aplicadas. Las restricciones condicionan la evolución. La restricción $\sum_{j \in S} \lambda_j = 1$ permite el cálculo de estados exteriores a la base de datos, permitiendo que λ_j tome tanto valores negativos como positivos.

Para una explicación más a fondo, consultar la fuente [2].

En este trabajo se procederá a aplicar los algoritmos de optimización y control propuestos en el artículo ya mencionado sobre un sistema real del laboratorio, en concreto: Process Control Trainer 37-100.

[1] From model-based control to data-driven control: Survey, classification and perspective Zhong-Sheng Hou, Zhou Wang

[2] Data Driven Control: An offset Free Approach. Jose R. Salvador, D.R. Ramirez, Teodoro Alamo, D. Muñoz de la Peña, G. García-Marín

2 INTEGRACIÓN SOFTWARE Y HARDWARE

En este trabajo se ha utilizado distintas herramientas para realizar los experimentos.

En primer lugar, se ha utilizado una unidad Process Control Trainer 37-100, sobre el que se han realizado los experimentos.

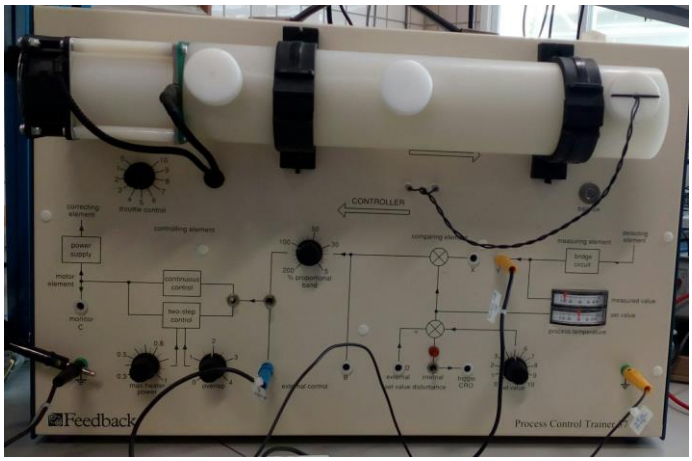

Fig 1: Feedback Process Control Trainer 37-100

Para comunicar la unidad con el ordenador del laboratorio se ha utilizad una tarjeta de adquisición de datos NI USB-6211.

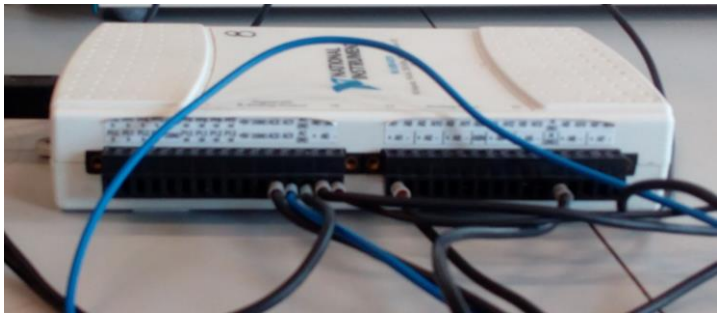

Fig 2: Tarjeta de adquisición de datos NI USB-6211 detalle conexiones.

Fig 3: Tarjeta de adquisición de datos de datos NI USB-6211

Los datos se han adquirido usando un módulo que corre sobre Matalb Simulink y se han tratado posteriormente con Matlab.

2.1 Software

2.1.1 Matlab

MatLab es una herramienta de software matemático optimizado para el trabajo con matrices y que ofrece un entorno y un lenguaje de programación propios [3]. El controlador implementado requiere de la inversión de matrices en cada tiempo de muestreo, por el que esta característica es importante.

Para realizar los ensayos se ha utilizado la versión de Matlab 2013b.

Además, ha sido necesario utilizar la herramienta Simulink, tanto para generar la base de datos como para realizar los experimentos.

Simulink es un entorno de programación de alto nivel en el que se programa utilizando distintos bloques que se van interconectando. Esto hace este entorno altamente intuitivo y fácil de programar.

2.1.2 Quarc

Quarc es un software de control en tiempo real desarrollado por Quanser, en concreto se ha utilizado la versión que corre sobre Simulink. Para las pruebas desarrolladas en este trabajo se ha utilizado su versión 2.4.

Dado que Windows no soporta tiempo real este software al compilar genera una aplicación de tiempo real que es cargada después en el equipo, en este caso, la tarjeta de adquisición de datos.

2.1.2.1 Configuración

Para realizar la conexión del equipo con la tarjeta es necesario realizar los siguientes pasos:

1- Abrir Simulink

2- Configurar el tiempo de *'Normal'* parámetro por defecto a *'External'* con esto se indica que el reloj que se debe utilizar para la simulación.

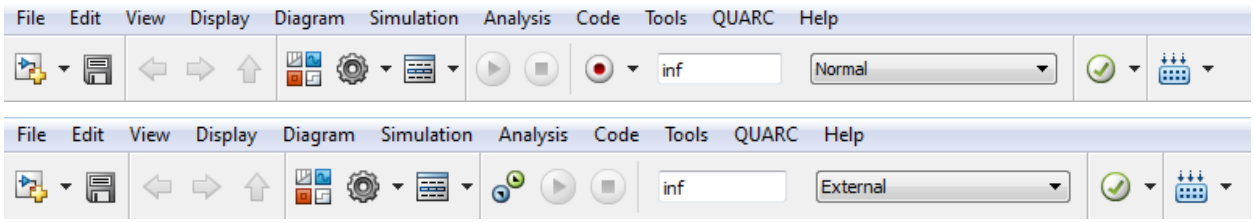

Fig 4: Configuración Quarc Simulink: Tiempo

3- Abrir la librería de Simulink. En la ruta *QUARC targets/Data Acquisition/Generic/Configuration*

seleccionar el bloque *HILL initialize*.

4- Acceder a los parámetros del bloque. En este bloque se configuran las características de la tarjeta de adquisición de datos.

5- En la pestaña *Main* Seleccionar el modelo de la tarjeta en el desplegable *Board Type.*

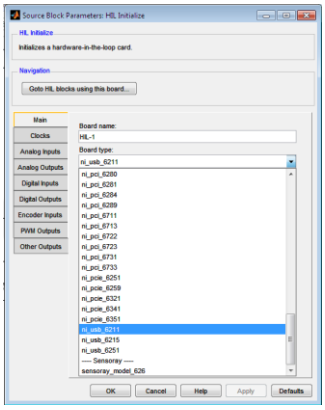

Fig 5:Configuración Quarc Simulink: Configuración de la tarjeta (1)

6- En la pestaña *Analog Inputs* se configuran los canales de entradas a utilizar. En *Analog input channels* escribir entre corchetes los nombres de los canales. En *Analog imput maximus* se introduce el máximo y en *Analog input minimun* el mínimo. Estos valores, así como los nombres de los canales son especificaciones de la tarjeta.

7- En la pestaña *Analog Output* se configuran los canales de la salida a utilizar. y se rellenan los campos de manera análoga a la pestaña *Analog inputs.*

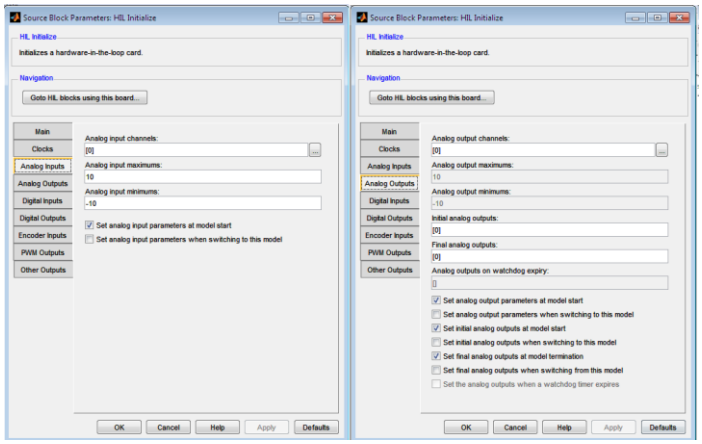

Fig 6:Configuración Quarc Simulink: Configuración de la tarjeta (2)

8- Pulsar *Apply* y *OK* [4].

Con esto se termina la configuración de la tarjeta.

Para adquirir los datos es necesario utilizar los bloques *HIL Write Analog* y *HIL Read Analog*. Que se encuentran en *QUARC targets/Data Acquisition/Generic/Inmediate I/O.*

La configuración de estos bloques es muy parecida.

Es necesario indicar el nombre del canal en *Channels* y el tiempo de muestreo. Si el tiempo de muestreo se deja por defecto el sistema utilizará el que está indicado en la configuración del solver en el caso del bloque de lectura y utilizará el tiempo de muestreo que tenga la señal de entrada en el caso del bloque de escritura. Se puede utilizar un tiempo de muestreo distinto al del solver sin problemas.

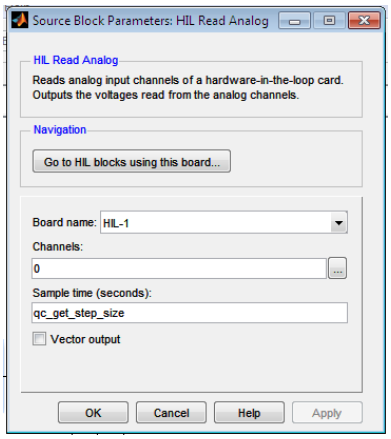

Fig 7: Configuración Quarc Simulink: Configuración de los canales

Para modificar el tiempo de muestreo en el solver:

1- Acceder al menú *model configuration parameter* clicando sobre el icono del engranaje.

2- En *Solver Options* Indicar el tiempo de muestreo en *Fixed-Step Size*.

Para este sistema se ha dejado el tiempo de muestreo *Fixed-Step Size* en el valor 0.002, pero se ha utilizado un tiempo de muestreo para los bloques de lectura y escritura de 0.07.

2.1.2.2 Ejecutar un modelo manualmente.

Una vez el modelo se ha montado y configurado, se procede a compilar, cargar y ejecutar la simulación.

Para compilar el modelo se clica sobre el botón "Build Model". Situado en la barra de menú

Fig 8: Ejecutar un modelo: Botón Build Model

Al hacer esto el compilador de Matlab genera dos carpetas auxiliares en la carpeta donde esté guardado el archivo: *"Nombre_quarc_windows"* y *"slpr"* y un archivo ejecutable de quarc en Windows d *"Nombre.rt-windows"*

Una vez compilado hay que conectar el equipo a la tarjeta pulsando sobre "Connnect To Target"

Fig 9: Ejecutar un modelo: Botón "Connect to target"

Cuando esté conectada se activará el botón de ejecutar *"Run"*.

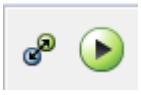

Fig 10: Ejecutar un modelo: Botón "Run"

A continuación, se ejecutará la simulación durante el tiempo especificado. Si este parámetro es *"inf"* esta habrá que pararla a mano clicando sobre el botón de stop.

2.1.2.3 Limitaciones de Quarc

Quarc va compilado, motivo por el que hace una reserva estática de memoria. Esto limita el tamaño que pueden alcanzar las variables internas por lo que algunas utilidades de Matlab Simulink pueden no funcionar correctamente. Por ejemplo: los bloques *ToWorkspace*, *FromWorkspace* o las limitaciones de punto de los *Scope*. Estas cosas a veces no se muestran como fallos de compilación, pero pueden provocar que los datos extraídos estén corruptos o que la conexión no se cierre adecuadamente, haciendo más complicado el proceso de automatización de las trayectorias.

El hecho de que esto no funcionara ha obligado a sacar las variables resultado de las simulaciones escribiendo sobre un fichero que hay que cargar posteriormente desde el espacio de trabajo.

2.1.2.4 Errores comunes:

Al darle al botón de compilar salta una pantalla con errores si la compilación falla. Esta pantalla está compuesta por dos partes, en la parte superior aparece una lista de todos los errores de compilación.

Clicando sobre ellos aparece una descripción detallada del error en la parte inferior.

Si es un error de programación haciendo doble click sobre el error nos llevará al bloque que está fallando.

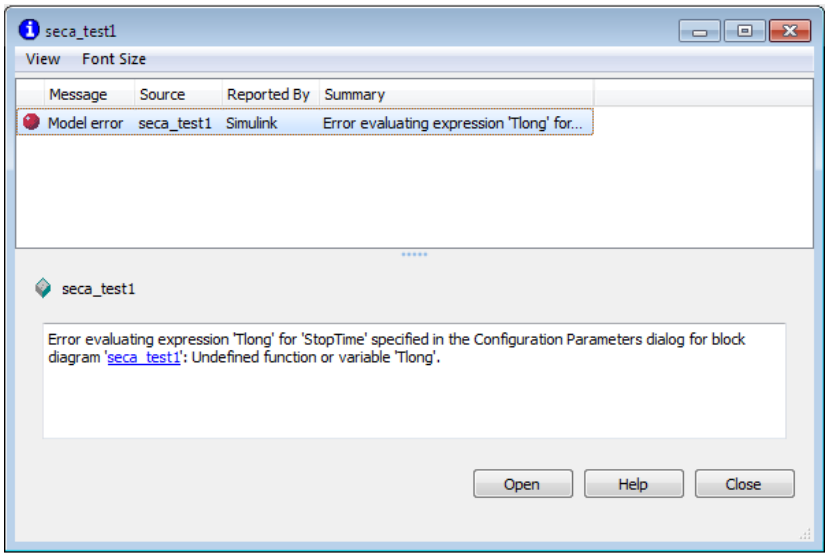

Fig 11: Errores Quarc

Algunos errores comunes

- "*Error evaluating expression 'Var1' for 'nombre del parámetro' specified in the Configuration Parameters dialog for block diagram 'modelo de Simulink': Undefined function or variable 'Var1'"*

  Este error se da cuando hay una llamada a una constante o una función que no está definida en el *workspace.*

- *"The model you are trying to run in External mode does not match the application running in your target"*

  Típicamente se da cuando no se ha realizado alguno de los pasos y el modelo no se ha compilado antes de ejecutarlo.

- *"Error occurred while executing External Mode MEX-file 'quarc-comm' DAQmx was unable to start the National Instrument device."*

  Este error ocurre cuando hay otro archivo conectado a la tarjeta o una sesión abierta desde el *workspace*. Para solucionarlo es necesario cerrar estas conexiones.

- *"Error occurred while executing External Mode MEX-file 'quarc-comm'. An operating system specific kernel-level driver for the specified card could not be found."*

  "Error occurred while executing External Mode MEX-file 'quarc-comm' DAQmx was unable to attach the specified analog output to the national Instruments analog output task."

  Estos dos errores se dan cuando la tarjeta no está físicamente bien conectada al ordenador o cuando el driver de la misma no ha terminado de instalarse.

- *"Fatal error en sbuild."*

  Este error es un fallo de compilación que requiere cerrar Matlab y reiniciar el ordenador.

Otras veces aparecen fallos o ventanas emergentes relacionados con quarc como pueden ser:

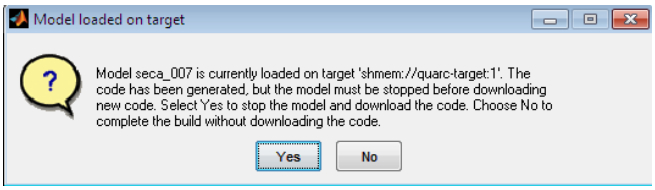

Fig 12: Errores Quarc: ventana informativa

Esta ventana sale al intentar lanzar de nuevo el modelo cuando este último no se ha parado previamente correctamente. Es necesario pulsar sí, aunque a veces al compilar con esto sale el siguiente error en la pantalla de Matlab:

"Model cannot be downloaded to target."

Otro fallo que se ha encontrado es que los datos de salida estaban corruptos y el archivo no se podía abrir. Esto puede pasar si se intenta acceder a los datos antes de que acabe la ejecución. Es un fallo que tiende a aparecer cuando hay bloques *fromWorkspace.*

2.2 Hardware

2.2.1 Process Control Trainer 37-100

La unidad Process Control Trainer 37-100 se corresponde con un sistema compuesto por un elemento calefactor controlado por un tiristor. El elemento calefactor se encuentra en medio de un flujo de aire por el que se hace circular con un ventilador axial. También cuenta con tres sensores de temperatura a lo largo del tubo.

Para simplificar la nomenclatura se referirá este equipo como secador a lo largo de este documento.

2.2.2 Tarjeta de adquisición de datos NI-USB 6211

La tarjeta de adquisición NI-USB es la tarjeta de adquisición de datos disponible en el laboratorio.

Esta tarjeta cuenta con:

- 8 canales de lectura analógica diferencial o 16 de un solo extremo con una resolución de 16 bits y una resolución temporal de 50ns y rango de lectura de ±10V.
- 2 canales de actuación analógica con una resolución de 16 bits y una resolución de 50ns.

El sistema precisa de una entrada y una salida, que se muestrean a tasas de muestreo inferiores, por lo que esta tarjeta cumple con las necesidades del sistema.

2.3 Conexionado

Para conectar la tarjeta de adquisición de datos al secador se han conectado la entrada de lectura diferencial AI0 al puerto de lectura Y del secador. Como este puerto es de lectura diferencial tiene dos conexiones, "AI0+" conectada a "Y" y "AI0-" conectada a tierra.

También se ha conectado al puerto para actuación A la salida del puerto AO0 y las tierras de ambos dispositivos.

En la Fig 13 se muestran estas conexiones mediante un código de colores.

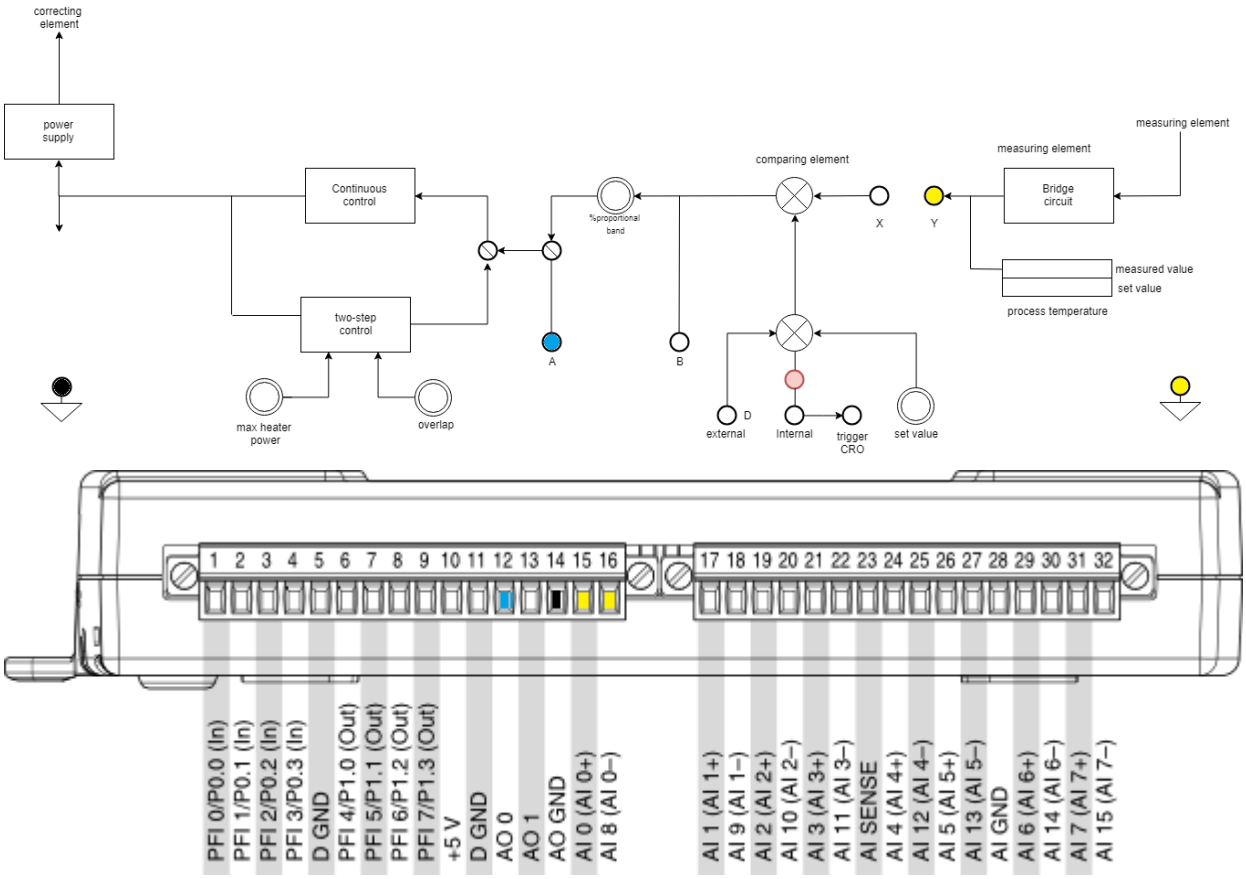

Fig 13: Conexionado de los equipos

[3]https://es.wikipedia.org/wiki/MATLAB

[4] Comunicación de tarjetas de adquisición de datos NI con Quarc e integración con prácticas de control, Adrián Rodríguez Galisteo.

PRIMERA ETAPA

En esta etapa, partiendo del algoritmo que había sido facilitado y probado en simulación, se procedió a realizar pruebas sobre el equipo real con el objetivo de determinar si funcionaba y si este funcionamiento era bueno.

Para ello se generaron varias bases de datos y se hicieron experimentos modificando los parámetros del sistema.

1. BASE DE DATOS DE LAS TRAYECTORIAS

1.1 Características

La base de datos debe tener unas ciertas características para que cumpla las hipótesis que se han considerado en el desarrollo teórico. Estas características son:

La base de datos está compuesta por trayectorias históricas del sistema en las que se alcanza en el régimen permanente el valor de referencia con un offset nulo. Cada punto de la trayectoria debe estar identificado con un ID, y una marca de tiempo.

Los datos guardados de estas trayectorias son la acción de control aplicada, el estado del sistema y la referencia.

Para este sistema, se ha considerado que el estado del sistema coincide con la con la salida del mismo. Además, todas las trayectorias tendrán la misma longitud en el tiempo. En este caso 55s.

La estructura de la base de datos es una tabla en la que cada fila representa un punto y tiene asociado un identificador de trayectoria.

1.2 Implementación de la base de datos

Para realizar la base de datos ha sido necesario implementar un controlador sobre el sistema y almacenar las variables indicadas anteriormente. Este controlador se ha implementado utilizando Matlab Simulink con Quarc.

Se han utilizado controladores de tipo PI y PID implementados sobre un bucle cerrado simple. Tal y como se muestra en la Fig 14.

En un principio se utilizaron controladores con constantes generadas de forma aleatoria, sin embargo, esto daba lugar a trayectorias que no alcanzaban la referencia en el tiempo estipulado y a trayectorias oscilantes o inestables que no cumplían los requisitos para entrar en la base de datos y en consecuencia debían ser descartadas.

Para generar una base de datos de forma que se cumpliese que: en primer lugar, todas las trayectorias pertenecientes a la base de datos cumplían las características especificadas, al mismo tiempo que se intenta

descartar el mínimo número de trayectorias ya generadas, se implementó un controlador que tuviese pequeñas variaciones en los parámetros sobre uno que funcionase bien y una vez realizada la trayectoria se realizan las comprobaciones de la validez de la trayectoria a posteriori.

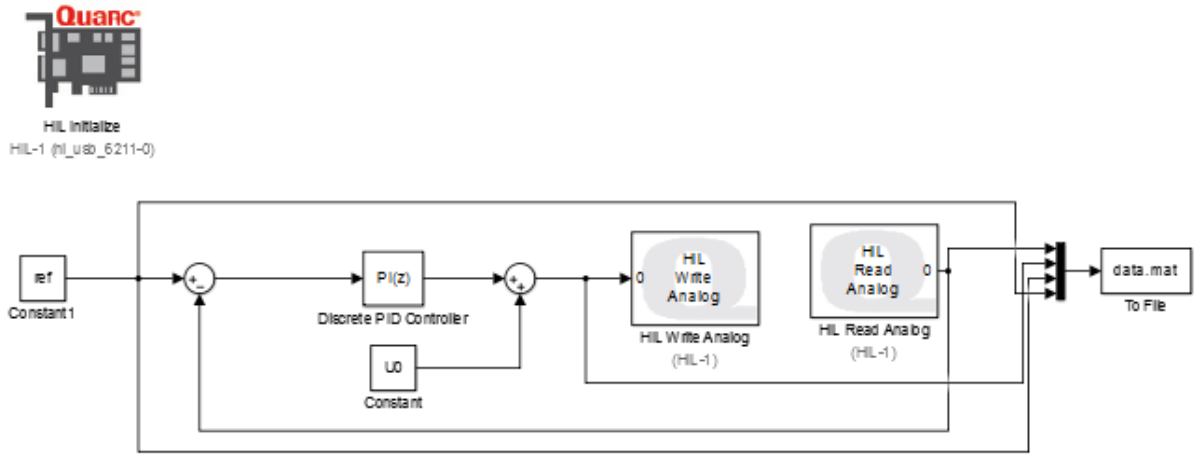

Fig 14: Esquema de control para realizar las trayectorias.

1.2.1 Identificación del sistema y cálculo de un controlador teórico

Para realizar la identificación del sistema se aproximó por un sistema de primer orden con retraso. Se han realizado ensayos en escalón y se han identificado las constantes del sistema.

Debido a que el sistema tiene una serie de parámetros que se pueden alterar y a que el dispositivo se encuentra en el laboratorio esta identificación se ha repetido varias veces a lo largo del trabajo.

Se identificó la constante de tiempo de alrededor de 0.7s, motivo por el que el tiempo de muestreo del sistema se configuró en 0.07s.

1.2.2 Implementación del controlador sobre el sistema

El controlador se ha implementado utilizando un bloque de controlador PID de la librería de Simulink.

Este bloque se ha configurado para que implemente un controlador discreto, utilizando la transformada *forward Euler*, el tiempo de muestreo que coincida con el tiempo de muestreo del sistema (0.07) y se ha aplicado un filtro para la derivada de N=10. Las constantes del controlador sobre las que se han implementado las variaciones son: Kp=0.52, Ki=0.6 y Kd=0.05.

1.2.3 Validez de la trayectoria.

Para considerar que una trayectoria es válida, y poder añadirla a la base de datos, esta debe llegar a un régimen permanente sin error y sin oscilaciones. Dado que se está trabajando con un sistema real, este está sometido a ruidos, por lo que un error en régimen permanente nulo no es posible. Para evaluar si en el régimen permanente existía un offset se ha considerado que este debe permanecer dentro de una tolerancia entorno a la referencia.

Para comprobar si la trayectoria era oscilante en el régimen permanente se ha considerado que la varianza debía estar dentro de unos límites. Este límite se ha ajustado mediante ensayo y error.

1.2.4 Generación automática de la base de datos.

Para generar la base de datos se realizó un fichero de Simulink en el que se implementaba una trayectoria en bucle cerrado.

Este fichero lee los parámetros del controlador y la referencia de constantes que existen en el *workspace* de Matlab. Y escribe los resultados sobre un fichero de extensión mat. Este archivo de Simulink debe estar abierto para poder lanzarlo desde un *scrip* de Matlab.

Para generar muchas trayectorias de manera automática se ha procedido a ejecutar el fichero Simulink desde

Matlab.

Este código de Matlab genera una referencia dentro del rango de actuación del secador y una combinación de parámetros Ki, Kd, Kp generados a partir del controlador calculado más una variación de hasta el 50% de su valor original.

A continuación, compila y lanza la simulación. Matlab no tiene ninguna manera de saber si la simulación a acabado, pues no establece una comunicación con la simulación, por lo que para que "espere" los resultados de esta, se configura un temporizado que es lanzado justo después de lanzar la simulación. Como el tiempo de simulación es conocido a priori, este temporizador se configura para su duración sea de unos segundos más que el tiempo del ensayo.

Una vez el ensayo ha concluido, salta el timer y Matlab carga los datos grabados por el fichero. Por último, se comprueba la validez de la trayectoria.

La estructura de este fichero es:

```
Inicializa variables
Para contador<N° trayectorias
     Referencia = N° Aleatorio del 0 al 10
     Constante integral= constante_teórica+porcentaje aleatorio.
     Constante proporcional= constante_teórica+porcentaje aleatorio.
     Si número aleatorio del 0 al 1 <0.5
         Constante derivativa = constante_teórica+porcentaje aleatorio.
     Si no
         Constante derivativa =0
     Fin
     Compilar modelo de Simulink
     Conectar el modelo de Simulink
     Lanzar el ensayo
     Esperar timer
     Comprobar que los datos son válidos
     Añadir a la base de datos si lo son.
Fin
```

Para evitar realizar ensayos muy largos sin realizar copias en el proceso, se han lanzado los ensayos de 100 en 100. Y se han unido las matrices a posteriori, observando que los índices de identificación no se solapasen unos con otros.

1.3 Base de datos generadas

Con este método se generó una base de datos con las siguientes trayectorias.

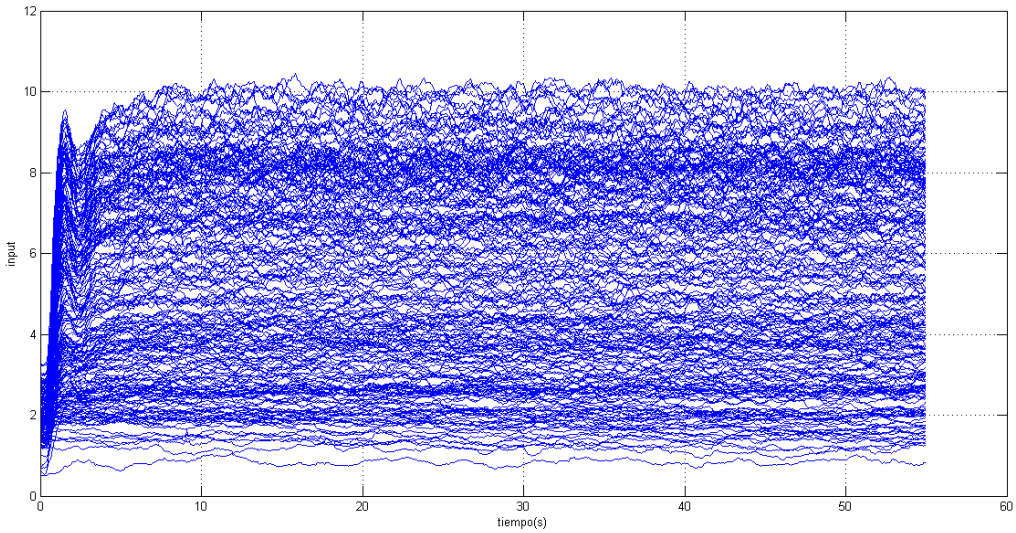

Fig 15: Base de datos generada

Se observa que las trayectorias generadas recorren todo el rango de funcionamiento del sistema.

Para realizar experimentos en los que la referencia estuviese fuera del rango de la base de datos se ha procedido a seleccionar las trayectorias con una referencia por debajo estricto de 7.

Para los ensayos en los que se utilizaban distintos tamaños de Base de datos se ha procedido a seleccionar las N primeras trayectorias de la base de datos. Como las trayectorias se generan con referencias aleatorias, al seleccionarlas así lo que se hace es reducir la densidad de trayectorias en el rango, pero sigue habiendo trayectorias en todo el rango. Tal y como se muestra en la figura

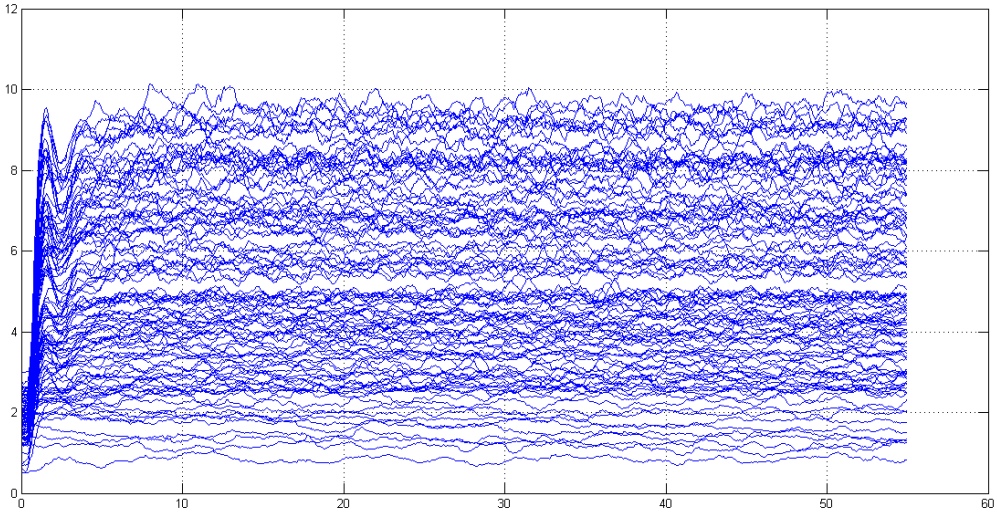

Fig 16: Base de datos de densidad menor.

2. IMPLEMENTACIÓN DEL ALGORITMO

La solución del problema de optimización planteada en [2] se puede resolver de manera explícita al estar sujeto a restricciones de igualdad. Lo que implica que el mínimo se puede calcular resolviendo el problema dual sin necesidad de utilizar un método de cálculo iterativo. Esto hace que la solución se pueda calcular más rápidamente.

El problema se resuelve de la siguiente manera:

Sea el problema genérico de optimización:

$$min\ \frac{1}{2} * \lambda^T \cdot H \cdot \lambda\ + q * \lambda$$

sa:

$$A * \lambda = b$$

Donde λ representa la variable primal, incógnita de nuestro problema.

Como las restricciones son de igualdad, este problema se puede expresar como

$$min \frac{1}{2} * \lambda^T \cdot H \cdot \lambda\ + q * \lambda + (A * \lambda - b)^T \cdot \beta$$

donde β representa las variables del problema dual asociado.

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda \\ \beta \end{bmatrix} = \begin{bmatrix} -q \\ 0 \end{bmatrix}$$

de manera que

$$\frac{\delta f(\lambda, \beta)}{\delta \lambda} = \ H \cdot \lambda + q^T + A^T \cdot \beta = 0$$

Despejando λ:

$$\lambda = H^{-1} \cdot (-q - A^T \cdot \beta)$$

Sustituyendo en las restricciones:

$$A \cdot (H^{-1}(-q - A^T \cdot \beta)) = b$$

De donde β resulta:

$$\beta = (-A \cdot H^{-1} \cdot A^T)^{-1} \cdot b\ + (-A \cdot H^{-1} \cdot A^T)^{-1} \cdot A \cdot H^{-1} \cdot q$$

Particularizando par este problema, q=0 el segundo término se anula.

Sustituyendo queda entonces:

$$\lambda = H^{-1} \cdot A^T \cdot (A \cdot H^{-1} \cdot A^T)^{-1} \cdot b$$

En este caso la matriz H es una identidad, por lo que:

$$\lambda = A^T \cdot (A \cdot A^T)^{-1} \cdot b$$

Una vez resuelto el problema, es necesario calcular cuales son las matrices que componen el problema.

Para calcular las matrices el algoritmo selecciona los puntos más cercanos y calcula las matrices con dichos puntos.

De esta manera, en cada tiempo de muestreo el algoritmo calcula la acción de control realizando los siguientes pasos:

Calcula la distancia entre todos los puntos de la base de datos y el estado del sistema. La distancia se calcula como:

$$distancia = |x_i(t) - x(t)| + |x_i(t-1) - x(t-1)| + |u_i(t) - u(t-1)| + |Ref_i - Ref|$$

Donde:

$x_i(t)$	Lectura del sensor en el punto t de la trayectoria i
$x(t)$	Lectura del sensor en el tiempo de muestreo
$x_i(t-1)$	Lectura del sensor el tiempo de muestreo anterior al punto t de la trayectoria i
$x(t-1)$	Lectura del sensor en tiempo de muestreo anterior.
$u_i(t)$	Acción de control llevada a cabo por el controlador en el punto t de la trayectoria i
$u(t-1)$	Acción de control calculada por el algoritmo en el instante anterior
Ref_i	Referencia de la trayectoria i
Ref	Referencia del sistema

A continuación, el algoritmo ordena las distancias obtenidas de menor a mayor y se queda con las Q más cercanas. Q es el número de candidatos para calcular la acción de control a partir de ellos. Cada candidato es el estado de una trayectoria en un instante de tiempo. Es decir, si el objetivo del problema es encontrar

$$\lambda_j{}^* = \arg\min \sum_{j \in S} \lambda_j{}^2$$

Q representa el tamaño del conjunto S y es uno de los parámetros que se pueden modificar.

Una vez seleccionados los Q puntos más cercanos se procede a calcular una matriz de restricciones A:

$$A = [x_Q | x_{QANT} | u_{ANT} | R_Q | 1]$$

y el vector b:

$$z = [x(t),\ x(t-1),\ u(t-1),\ Ref]$$

Se calcula la matriz auxiliar K

$$K = -A^T \cdot (-A \cdot A^T)^{-1}$$

Se resuelve el problema calculando las λ óptimas

$$\lambda^* = K \cdot z$$

Por último, se calcula la acción de control como $u(t) = \lambda^* \cdot u_{QANT}$.

2.1 Implementación en Quarc

Para implementar este algoritmo en el software de Quarc se utilizó el bloque función de Matlab de Simulink. Este bloque permite implementar una función dentro de un esquema de Simulink. Las entradas a dicho bloque son: la base de datos, la lectura actual, la lectura del tiempo de muestreo anterior, la acción de control anterior y la referencia.

Para introducir la base de datos por parámetros en la función ha sido necesario modificar la estructura de la base de datos, de manera que sea un vector compuesto por todos los parámetros que se mantiene constante entre el tiempo de inicio y el tiempo final. Consecuentemente una vez dentro de la función era necesario darle a la base de datos la estructura adecuada de nuevo.

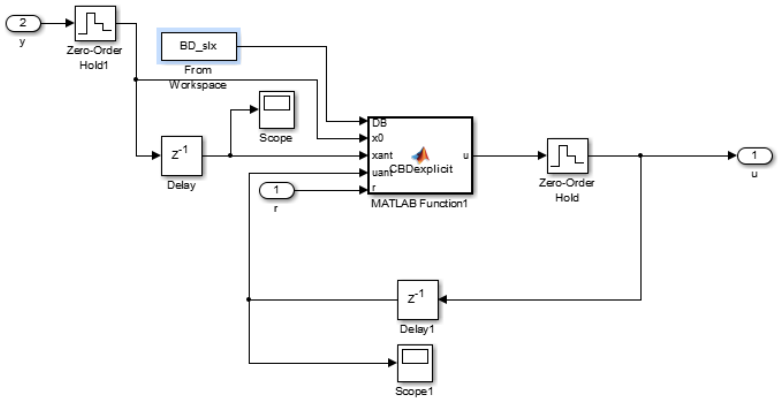

Fig 17: Esquema de Simulink: Entradas y salidas de la función del algoritmo

Esta función, a su vez, se ha introducido en otro bloque de Simulink en el lugar que correspondería al controlador en el esquema del bucle cerrado.

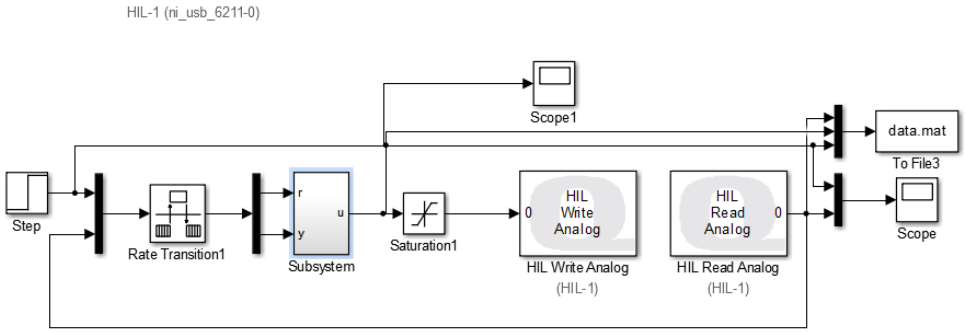

Fig 18:Esquema de Simulink: Esquema de control de implementación del algoritmo

El bloque *rate transaction* se colocó debido a que la realimentación daba problemas de sincronización del muestreo.

Con este esquema se procedió a hacer la primera tanda de experimentos. Durante la realización de los mismos se comprobó que al aumentar el tamaño de la base de datos Quarc daba problemas y la compilación falla debido a problemas de memoria.

Esto se puede deber a que darle forma de nuevo a la matriz es una función “lenta” y al aumentar el tamaño de la base de datos, Quarc no puede garantizar las restricciones de tiempo.

2.2 Implementación desde MatLab.

Debido a los problemas que estaban presentándose en Quarc, como limitaciones del tamaño de la base de datos, se optó por comunicar directamente Matlab con la tarjeta de adquisición de datos.

Se utiliza en las funciones del *Data Adquisition Toolbox* en Matlab que es compatible con las tarjetas de

National Instruments y que viene incluido en la instalación habitual del programa.

2.2.1 Data Acquisition Toolbox

Para comunicar una tarjeta con Matlab el *toolbox* presenta dos interfaces de funcionamiento, *legacy* y *sesión*. El primero es un modo que deja de estar soportado a partir de Matlab2016a.

Para conectarse con la tarjeta de adquisición de datos es necesario en primer lugar realizar un escaneo de los instrumentos conectados al ordenador.

A continuación, se inicia la sesión utilizando la identificación de la marca del elemento de adquisición de datos a utilizar.

Una vez iniciada la sesión se procede a configurar las entradas y salidas que se vayan a utilizar, así como la tasa de muestreo. Esto está explicado con más detalle tanto en el Anexo II: Data adquisition Toolbox donde se explica el uso de las funciones como en el Anexo III: Código donde se encuentra el código utilizado para esta configuración.

Es importante tener en cuenta que mientras haya una sesión iniciada, el equipo no podrá comunicarse mediante otra sesión o utilizando otro software.

Todo este proceso de configuración se ha encapsulado en un programa que se lanza cada vez que se va a utilizar la tarjeta desde Matlab.

Para escribir y leer sobre las entradas existen varias opciones. Se ha utilizado la que permite escribir y leer una sola muestra cada vez.

2.2.2 Restricciones de tiempo.

A la hora de implementar el algoritmo este se ha implementado dentro de un bucle del tipo

```
Bucle
        Lectura
        Cálculo de la acción de control
        Escritura
        Espera
Fin
```

Dado que MatLab corre sobre Windows, no soporta tiempo real. Por lo que no se puede garantizar que se estén cumpliendo los tiempos de muestreo estipulados anteriormente. Sin embargo, el tiempo de muestreo es relativamente grande comparado con los tiempos de ejecución del algoritmo, por lo que se ha optado por medir este tiempo y esperar el restante hasta completar el tiempo de muestreo de 0.07s.

De esta forma no se garantiza que se cumpla el tiempo de muestreo, sino que se comprueba que se cumple. De hecho, ocasionalmente se excede este tiempo estipulado, especialmente en el arranque pues es necesario cargar los drivers o puntualmente cuando el sistema operativo interviene con alguna tarea.

2.2.3 Condiciones iniciales.

Al dejar de usar el software de Quarc el algoritmo no tienen información de lo que está ocurriendo en el sistema al empezar a ejecutarse. Originalmente se utilizaban condiciones iniciales nulas, pero esto no se correspondía con lo que realmente estaba ocurriendo en el sistema. En consecuencia, aparecía un comportamiento indeseado en el transitorio.

Para corregir esto, antes de conectar el algoritmo se le da al sistema una entrada constante y se lee durante 20 segundos en bucle abierto. A continuación, se utilizan las dos últimas lecturas y escrituras para inicializar los vectores de acción de control y estado antes de lanzar el algoritmo. En las figuras Fig 19: funcionamiento del algoritmo con condiciones iniciales y Fig 20: Funcionamiento del algoritmo sin condiciones iniciales

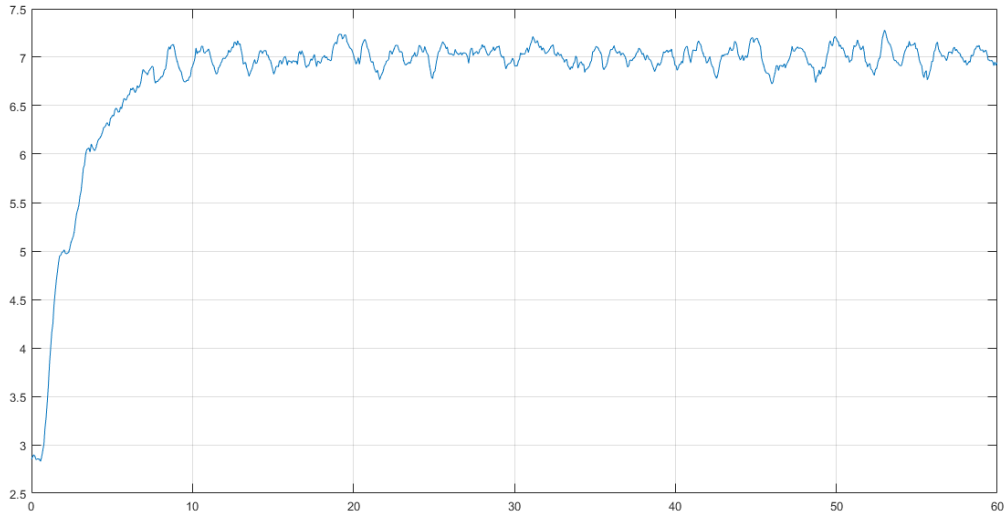

Fig 19: funcionamiento del algoritmo con condiciones iniciales

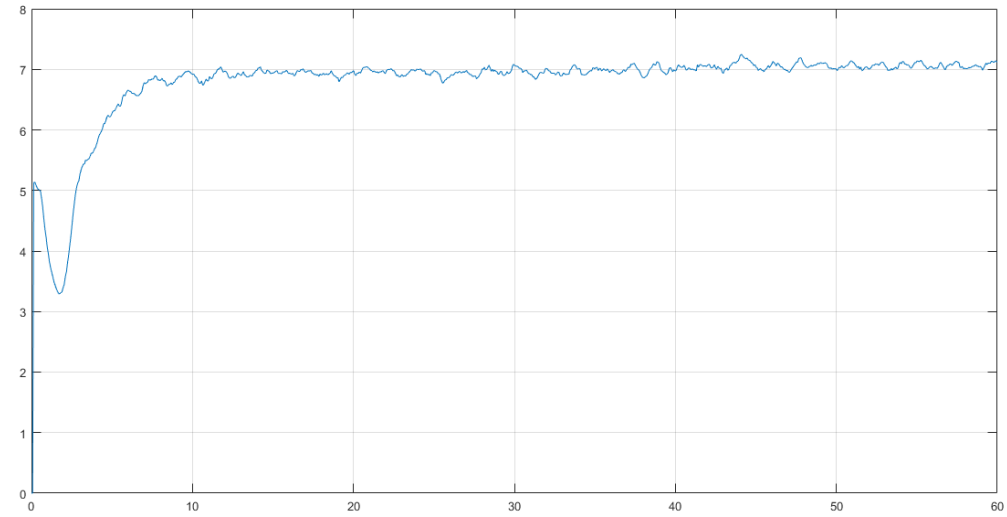

Fig 20: Funcionamiento del algoritmo sin condiciones iniciales

2.2.4 Cambios realizados sobre la función.

Para lanzar el algoritmo desde Matlab se han realizado pequeños cambios en la función utilizada en Quarc Estos cambios han sido:

- Se ha eliminado la función de reforma de la base de datos y esta se introduce en el algoritmo como una tabla. Esto hace que el tiempo de ejecución del algoritmo se reduzca, y ahorra las dos operaciones.
- Se han añadido entradas que permiten modificar los parámetros Q y tamaño de la base de datos desde fuera de la función, así como la longitud de las trayectorias. Esto hace que sea más cómoda de usar, y sea más fácil automatizar los ensayos, pudiendo modificar estos parámetros automáticamente, a diferencia de cómo hay que realizarlo en el primer caso, que para cambiar los parámetros es necesario abrir la función y cambiar el valor de las constantes

3. ENSAYOS

Para comprobar el funcionamiento del algoritmo sobre el sistema se procedió a realizar una serie de ensayos

En primer lugar, se procedió a evaluar el funcionamiento al darle una serie de escalones de referencia al sistema.

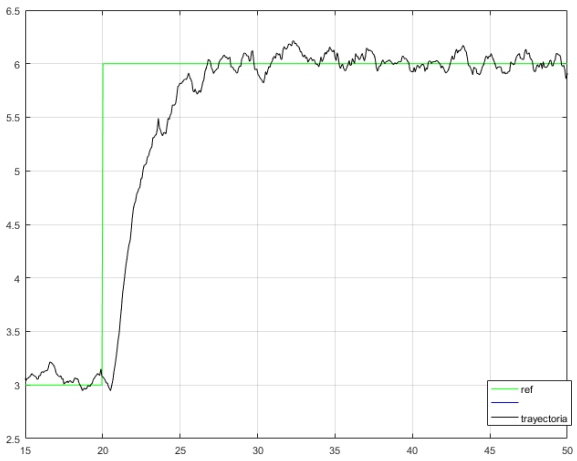

Fig 21: seguimiento

Después se realizaron una serie de ensayos para evaluar cómo funcionaba el algoritmo al variar los distintos parámetros. Los parámetros que se utilizaron fueron el tamaño de la base de datos y el número de candidatos (Q) que se seleccionaban.

3.1 Control del sistema por el algoritmo

Referencia dentro del rango de la base de datos

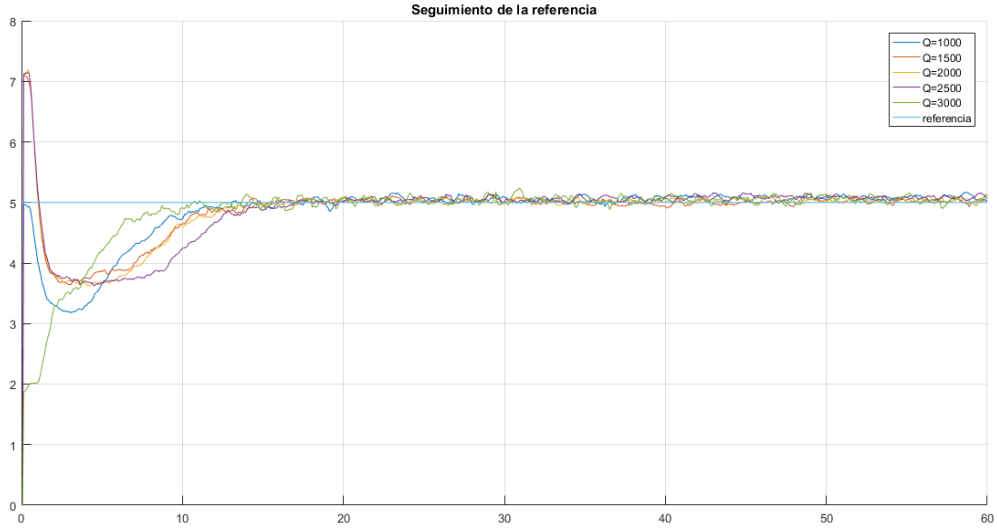

Fig 22: Funcionamiento para una misma referencia con distintos valores de Q

Para probar el funcionamiento del algoritmo se ha procedido a realizar seguimiento de una referencia en escalón modificando el número de candidatos.

Se observa que independientemente de este parámetro el sistema presenta un buen comportamiento y que llega al régimen permanente sin offset.

3.1.1 Referencia fuera del rango de la base de datos

Se probó el funcionamiento cuando se intentaban alcanzar estados fuera del rango cubierto por la base de datos.

Para ello se utilizó una base de datos con referencias por debajo de siete, y se procedió a dar referencias al sistema por encima de siete.

En estas condiciones se observa que existe un error en régimen permanente. Este error aumenta conforme la referencia se aleja de la base de datos. Como se observa en la Fig 23 donde se ha representado la evolución para una referencia de siete, justo en el límite, pero fuera y una referencia de 8, que está más lejos.

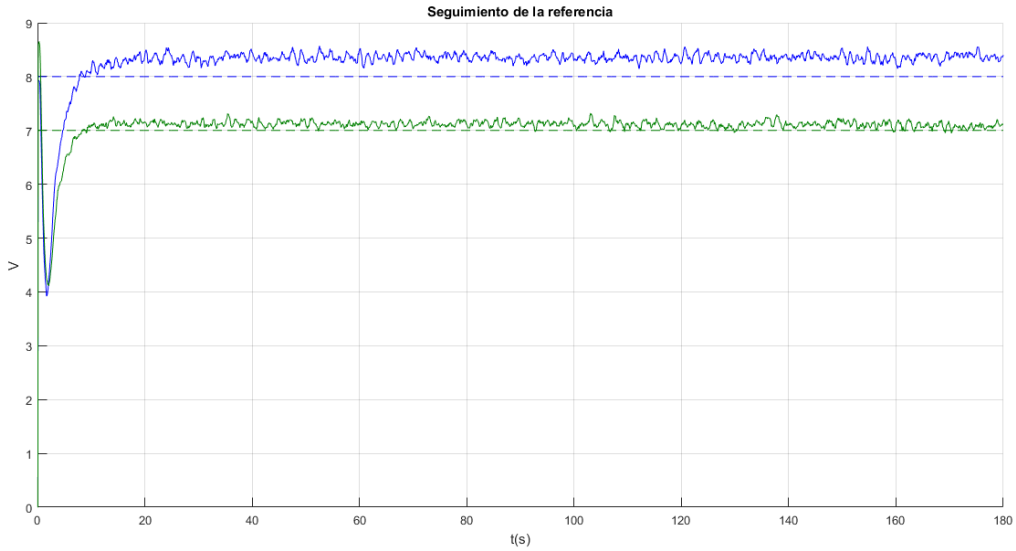

Fig 23: Funcionamiento para referencias fuera de la base de datos(1)

También se observa que este error en régimen permanente disminuye al aumentar el número de candidatos que se maneja. Tal y como se observa en la Fig 24.

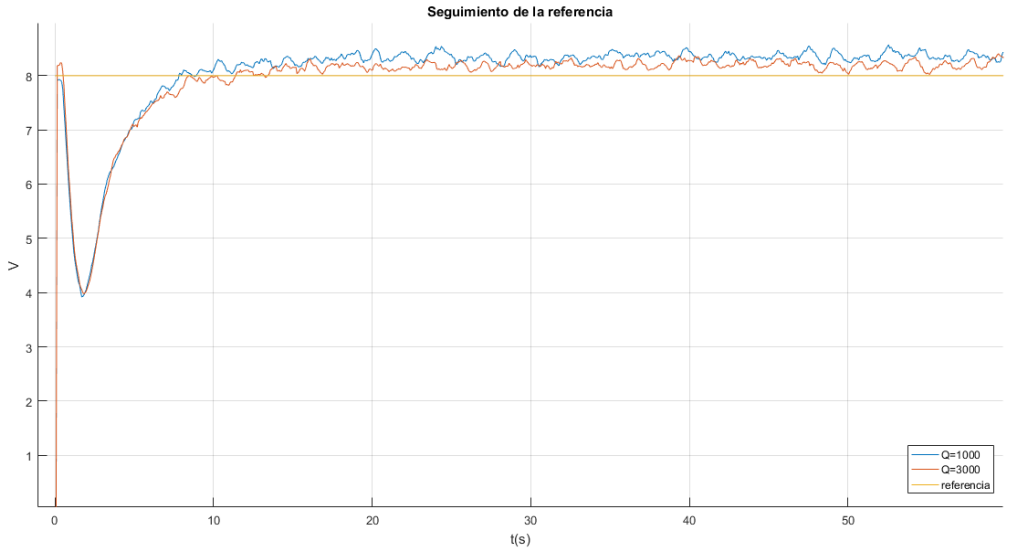

Fig 24 : Funcionamiento para referencias fuera de la base de datos (2)

La existencia de un error en régimen permanente se justifica pues el algoritmo no presenta ningún efecto integral, solo teniendo en cuenta el estado actual y el anterior para realizar la minimización.

3.2 Comportamiento frente a un aumento del número de candidatos

Para evaluar el comportamiento del sistema al modificar los parámetros se han realizado distintos ensayos. Para cada ensayo se calculó el error cuadrático medio de cada trayectoria en el régimen permanente. Este valor está representado a continuación como un punto. Para calcular las curvas se ha considerado el error medio de todas las trayectorias con el mismo juego de parámetros.

3.2.1 Comportamiento frente a un aumento del número de candidatos

Al aumentar el número de candidatos con los que trabaja el algoritmo se observa una menor dispersión del comportamiento, así como una mejora en el comportamiento, tanto dentro como fuera de la base de datos.

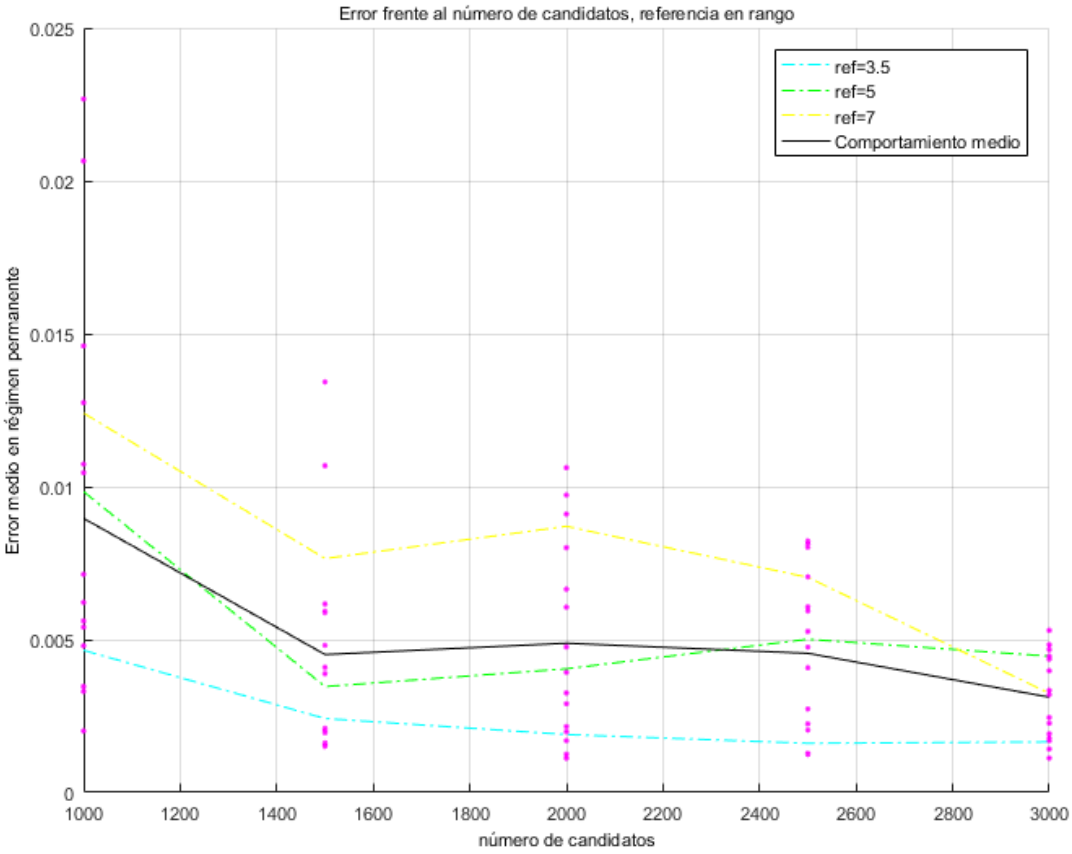

Fig 25: Evolución del error frente al número de candidatos con referencias dentro del rango.

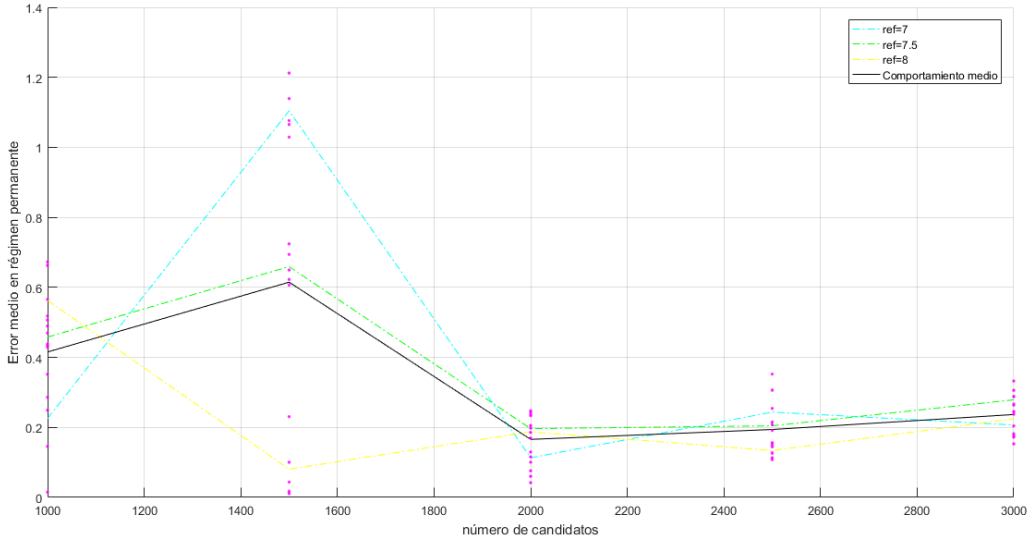

Fig 26: Evolución del error frente al número de candidatos con referencias dentro del rango.

3.3 Sensibilidad al tamaño de la base de datos.

Para obtener estos resultados se realizaron varios experimentos en los que se varió el tamaño de la base de datos y el número de candidatos. Se observó que para un aumento del tamaño de la base de datos y un Q constante el comportamiento no mejoraba. Se repitió la base de datos y los experimentos con resultados idénticos.

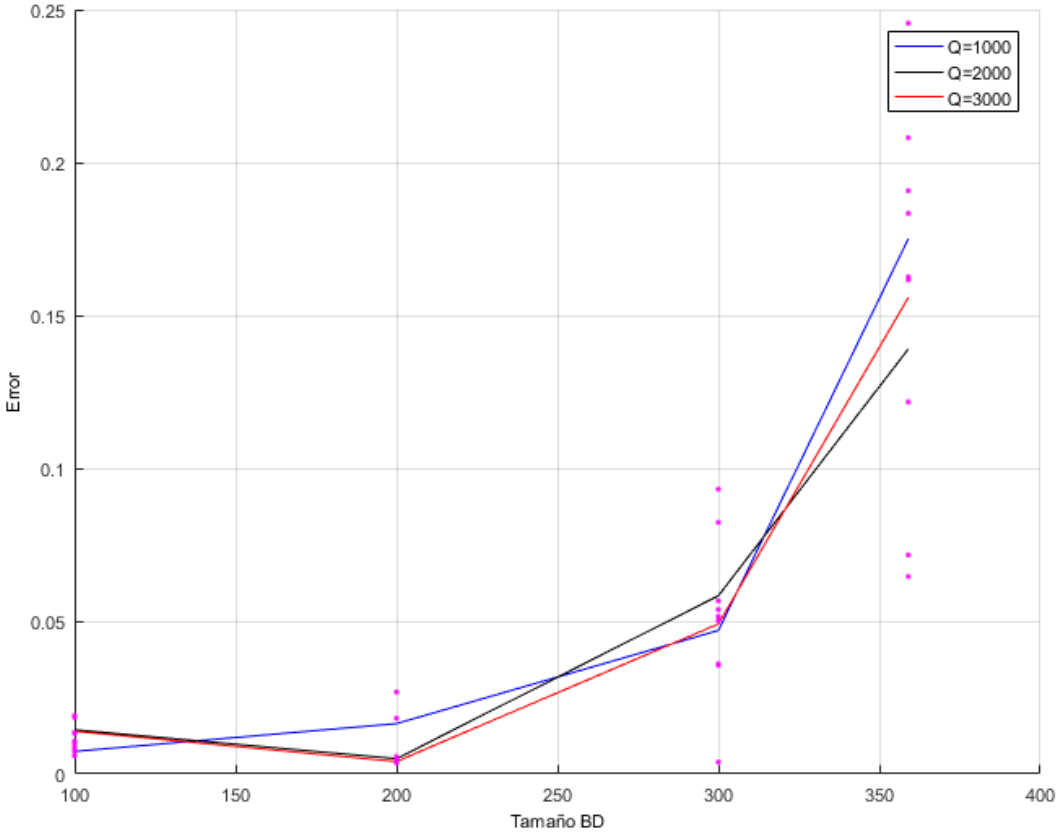

Fig 27: Evolución del error frente al tamaño de la base de datos

SEGUNDA ETAPA

En esta segunda etapa se plantearon algunas estrategias para hacer que el conjunto funcionase mejor. Para ello se introdujeron pequeñas mejoras en el algoritmo, se cambió la estrategia a la hora de hacer la base de datos y se utilizó un filtro para hacer que el sistema fuese menos sensible a los ruidos inherentes al sistema.

1. BASE DE DATOS DE LAS TRAYECTORIA

La generación de la base de datos utilizando pequeñas variaciones de un controlador para controlar las trayectorias y comprobando a posteriori si estas trayectorias cumplían los requisitos impuestos implica que no se sabe con certeza como de bueno es realmente el control que se está introduciendo en la base de datos.

Por ello se plantea una estrategia de generación de la base de datos que consiste en comparar el funcionamiento de varios controladores realizando la misma trayectoria. Dado que el sistema es un sistema real y presumiblemente no lineal, no existe un solo controlador PI que sea el mejor para todas las trayectorias. Por lo que con esto se pretende capturar este tipo de no linealidades y añadir a la base de datos la mejor trayectoria de un grupo y no cualquier trayectoria que cumpla las especificaciones.

Para ello se realizaron una serie de ensayos en los que para el mismo escalón de referencia se van variando los parámetros del controlador, y elegir de cada set de trayectoria aquella que minimizara una función de coste.

1.1 Implementación

Con el objetivo de simplificar el problema se decidió que:

- La variación de los parámetros del sistema está predeterminada a priori y no depende del comportamiento del sistema en escalones anteriores (como haría un *autotune*).
- La ganancia sería constante y con valor 1 durante todos los ensayos, siendo la constante integral la que varíe a lo largo de los ensayos.
- Se realizarán 4 trayectorias para cada escalón de referencia y estas se implementarán en un solo ensayo de Simulink para poder controlar la continuidad de las condiciones iniciales.

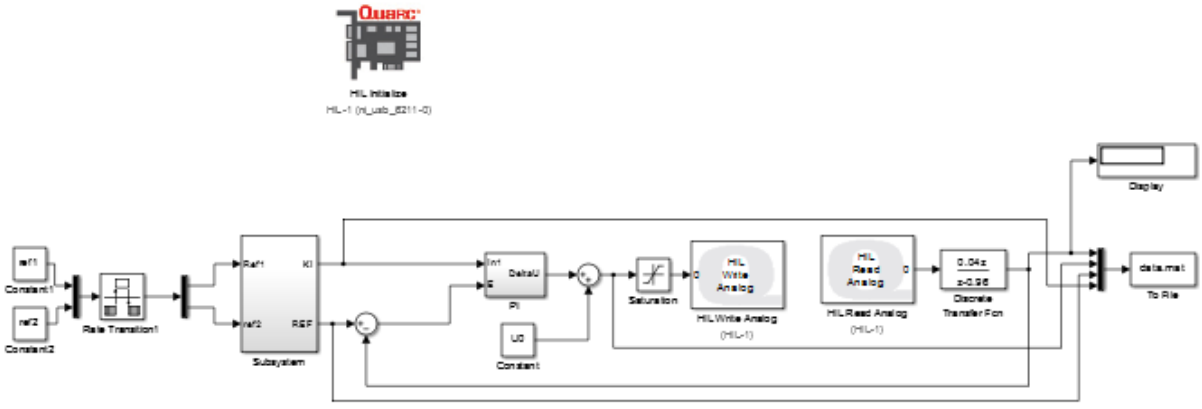

Fig 28: Esquema de Simulink: Implementación de la base de datos segunda estrategia

Dadas las limitaciones impuestas por Quarc a la hora de generar la base de datos ha sido necesario generar los escalones de referencia y los valores de la constante integral en un bloque función.

Este bloque tiene como entradas: dos constantes entre los que variará nuestra referencia, los valores actuales de la referencia y la constante integral, el tiempo de simulación y dos señales cuadradas que cambian de valor con la frecuencia y desfase necesarios para que los cambios se produzcan al detectar los cambios de flanco.

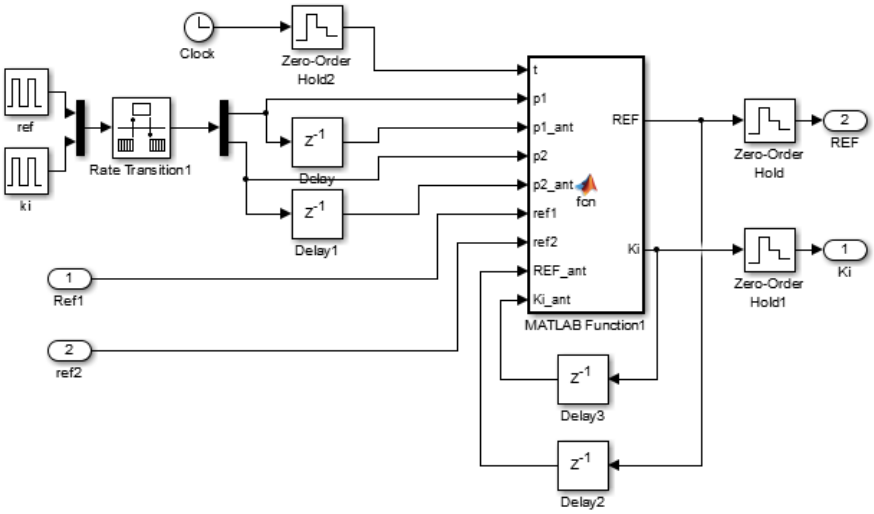

Fig 29: Esquema de Simulink: Implementación de la base de datos segunda estrategia: Entradas a la función

La salida de este bloque es una señal que está 30 segundos en el punto bajo de la referencia, después 60 segundos en el punto alto de la referencia y luego 30 segundos de nuevo en el punto bajo de la referencia. Esta secuencia se repite 4 veces. Cada ciclo se realiza con una constante integral distinta.

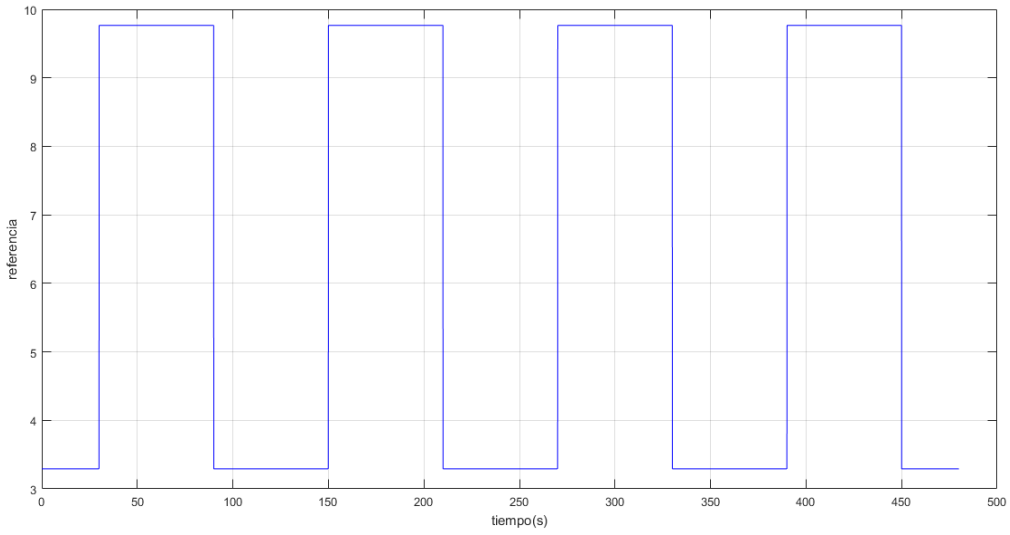

Fig 30: referencias utilizadas

Por último, ha sido necesario crear un bloque PI basado en el que hay en la librería de Simulink para poder pasar la constante integral como un parámetro. Esto se ha realizado haciendo una copia de la máscara del bloque PI de la librería.

1.1.1. Base de datos 1

Para generar esta primera base de datos se partió de un valor inicial de la constante integral (Ki) de 1.5. Para calcular los valores siguientes se multiplicó este valor por 0.4 en cada iteración.

Los valores de referencia se generan en el entorno de Matlab. El primer valor es un número aleatorio entre 2 y 5 y el segundo es un número aleatorio entre el primero más 2 y 10. Esta implementación hace que los escalones que recibe son siempre significativos y ascendentes.

Tras realizar esta base de datos se observó que el sistema solo seleccionaba trayectorias correspondientes a las primera y segunda trayectorias. Esta y otras consideraciones fueron tomadas en cuenta para ajustar ligeramente la generación de la base de datos para realizar la Base de datos 2

1.1.2. Base de datos 2

Para mejorar la base de datos generada se procedió a ajustar el método de generación de trayectorias. Para ello se recalcularon los posibles Ki para que estuviesen más repartidos en el rango que se había detectado como de buen funcionamiento. De esta manera los controladores implementados esta vez empiezan con un tiempo integral de 1.5 y van disminuyendo de forma lineal en 0.43.

También se procedió a añadir un filtro a la lectura de la trayectoria con el fin de mejorar el funcionamiento del algoritmo, esto se explicará con más detalle en el apartado 4.1.

Por último, se modificó el algoritmo de generación de referencias para que con una probabilidad del 50% se diese un escalón hacia arriba o hacia abajo. De esta manera quedaba recogida la dinámica del sistema ante un escalón hacia abajo en la base de datos.

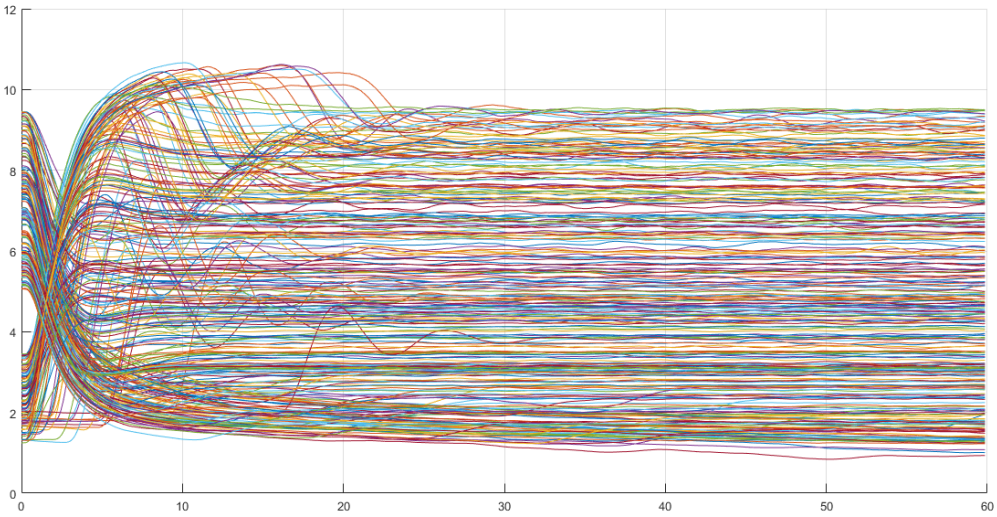

Fig 31: Trayectorias incluidas en la base de datos 2

1.2 Criterio de selección de las trayectorias.

Una vez generadas las trayectorias era necesario seleccionar la más adecuada. En este caso se ha optado por utilizar un criterio de mínimo coste.

Se ha calculado el coste de cada una de las trayectorias del ensayo como $\sum e^2$ y después se ha seleccionado la mínima.

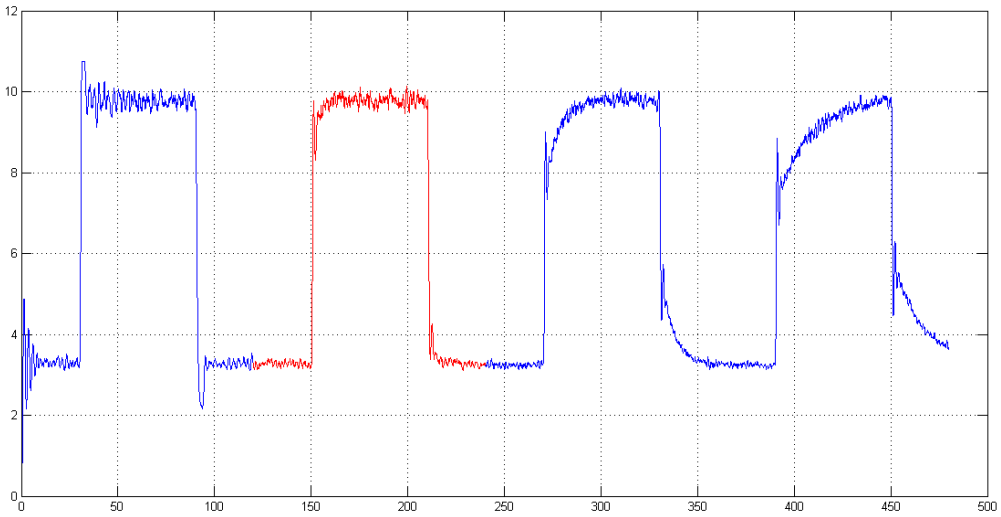

Fig 32: Un ensayo completo. La trayectoria seleccionada en rojo

Una vez seleccionada la trayectoria no puede simplemente añadirse a la base de datos. Pues la referencia varía entre dos valores a lo largo de la trayectoria y esto no está contemplado en el desarrollo teórico del algoritmo. Por ese motivo se añade a la base de datos solamente la parte de la trayectoria en la que la referencia está en su valor techo, o dicho de otra manera la parte de la trayectoria que se introduce en la base de datos captura el escalón hacia arriba de la trayectoria.

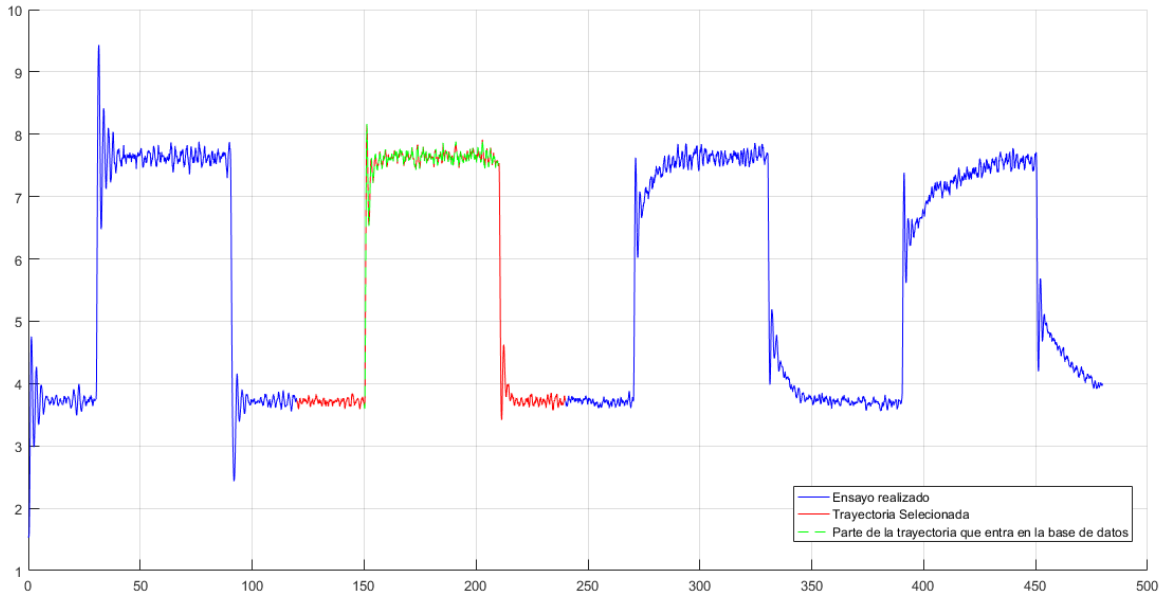

Fig 33:Un ensayo completo, trayectoria seleccionada, parte de la trayectoria que entra en la base de datos

También es importante tener en cuenta que las marcas de tiempo no empiezan en 0 para todas las trayectorias y que el origen depende de qué trayectoria sea la seleccionada como mejor. Esto se ha tenido en cuenta y se ha corregido para que todas las trayectorias empezaran en 0.

1.3 Base de datos de regulación

En todas las bases de datos generadas hasta ahora el comportamiento del sistema recogido es de seguimiento. Para intentar mejorar el funcionamiento frente a perturbaciones se generó una base de datos que capturase el funcionamiento en regulación.

Para realizar esta base de datos se mantuvo al sistema en bucle abierto y con una acción de control constante durante un minuto, de esta manera el sistema se encuentra en un punto aproximadamente estable. Pasado el minuto el sistema pasa a bucle cerrado.

A diferencia de las otras bases de datos generadas, en esta todas las trayectorias tienen la misma referencia al pasar al bucle cerrado.

Para implementarla ha sido necesario añadir a la función anterior unos switch que permitiesen abrir y cerrar el bucle.

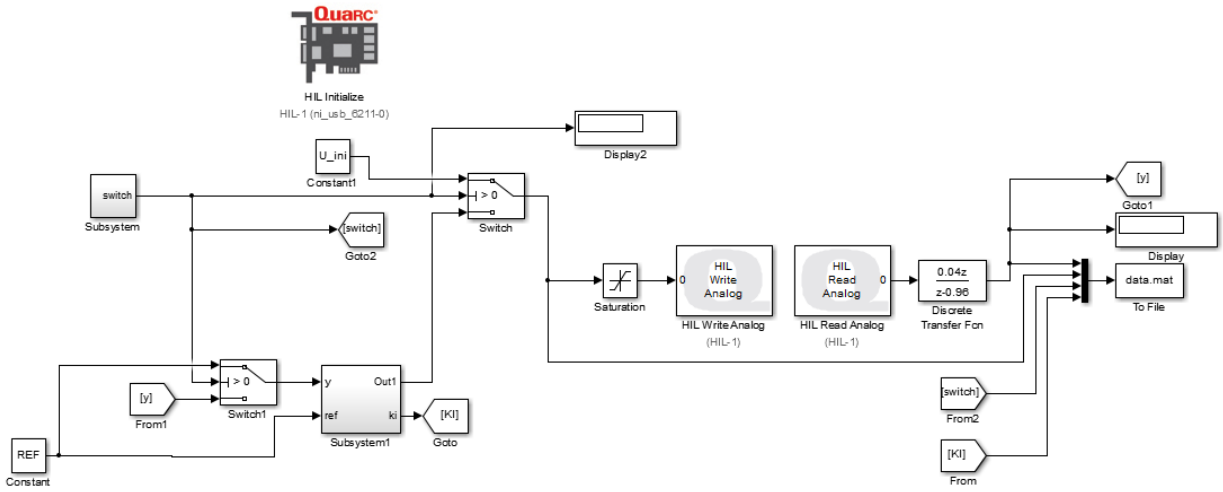

Fig 34: Esquema de Simulink: Implementación base de datos de regulación.

El bloque Switch genera una señal binaria para que se abra o cierre el buce de control.

Para evitar efecto de windup debido al bucle abierto (el controlador intenta corregir la salida, pero no puede por lo que se produce este efecto) ha sido necesario colocar un segundo interruptor de manera que la entrada del controlador entra el valor de la referencia. De esta manera, la integral del error hasta el momento en el que se pasa al bucle cerrado es nula y en consecuencia se evita este efecto.

La base de datos obtenida recoge la dinámica de un sistema desde cualquier punto hasta la referencia de valor 5. Se a elegido esta referencia pues se encuentra en el centro del rango de funcionamiento.

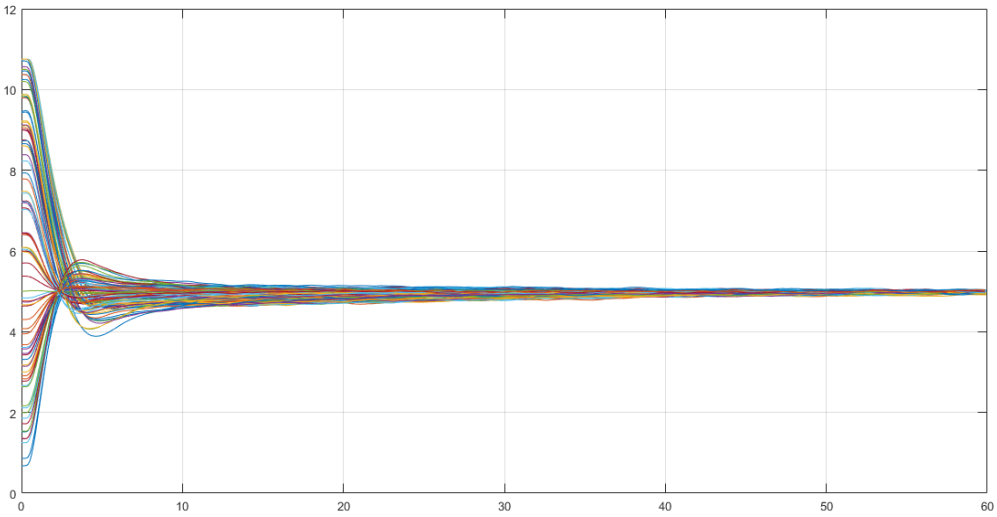

Fig 35: Trayectorias incluidas en la base de datos de regulación

2. ALGORITMO MODIFICADO

Después de realizar la implementación del algoritmo planteada en el apartado 2 de la primera parte, se añadió al planteamiento del problema un término con en el que se tiene en cuenta la evolución del error al aplicar una acción de control. Es decir, cómo una acción de control u otra hacen disminuir más o menos el error. De esta manera se formuló:

$$min\ \lambda^2 + \gamma \cdot \left\| \sum \lambda_i \cdot e(1) \right\|_p^2$$

Es decir, se busca no solo encontrar el óptimo sino el óptimo teniendo en cuenta los errores a un paso.

O expresado de otra manera:

$$min\ \frac{1}{2} \cdot \lambda^T \cdot (I + \gamma \cdot c^T \cdot p \cdot c) \cdot \lambda$$

s.a:

$$A \cdot \lambda = b$$

Donde c representa un vector con los errores a un paso de los puntos de la trayectoria.

Esto se traslada como:

$$\begin{bmatrix} I + \gamma \cdot c^T \cdot p \cdot c & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \lambda \\ \beta \end{bmatrix} = \begin{bmatrix} -q \\ 0 \end{bmatrix}$$

Es decir, ahora H ya no es una matriz identidad sino:

$$H = I + \gamma \cdot c^T \cdot p \cdot c$$

Donde γ es un parámetro y p representa la estabilidad en el sentido de Liapunov.

Resolviendo esto se llega a la conclusión de que:

$$\lambda = (I + \gamma \cdot c^T \cdot p \cdot c)^{-1} \cdot A^T \cdot (A \cdot (I + \gamma \cdot c^T \cdot p \cdot c)^{-1} \cdot A^T)^{-1} \cdot b$$

Dado que en esta ocasión la matriz H ya no es una identidad y c cambia para cada tiempo de muestreo, en cada tiempo de muestreo pasa de ser necesario realizar una inversión a ser necesario realizar tres

Las matrices a invertir son todas de dimensiones de QxQ, es decir, del orden de 1000, por lo que el tiempo en realizar estos cálculos llega a ser mayor que el tiempo de muestreo como se observa en la Fig36, motivo por el cual se ha aplicado el lema de inversión.

En primer lugar, se hace un cambio de variable $d = \sqrt{\gamma} \cdot c$.

Partiendo de la identidad de Woobury, que dice que:

$$(A + U \cdot C \cdot V)^{-1} = A^{-1} - A^{-1} \cdot U \cdot (C^{-1} + V \cdot A^{-1} \cdot U)^{-1} \cdot V \cdot A^{-1}$$

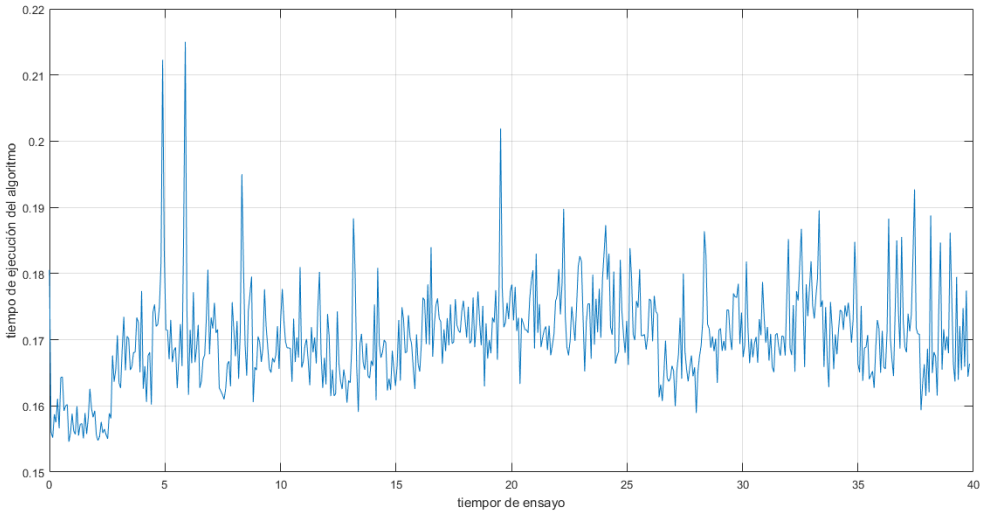

Fig36: tiempos de ejecución del algoritmo durante un ensayo

Particularizando: A=I, U=d^T, V=d y C=p

$$(I + d^T \cdot p \cdot d)^{-1} = I - d^T(p^{-1} - d \cdot d^T)^{-1} \cdot d$$

Deshaciendo el cambio:

$$(I + \gamma \cdot c^T \cdot p \cdot c)^{-1} = I - \gamma \cdot c^T(p^{-1} - \gamma \cdot c \cdot c^T)^{-1} \cdot c$$

O dicho de otro modo:

$$H^{-1} = I - \gamma \cdot c^T(p^{-1} - \gamma \cdot c \cdot c^T)^{-1} \cdot c$$

Donde I es una matriz identidad de QxQ, γ es una constante y c es un vector fila de longitud Q y p en este caso es una matriz de 1x1, por lo que la nueva matriz a invertir es de dimensión 1x1.

Con lo que el tiempo de inversión es mucho menor, y en consecuencia el tiempo que tarda el algoritmo en calcular la acción de control es más pequeño.

A continuación, se resuelve el problema dual como:

$$\beta = (-A \cdot H^{-1} \cdot A^T)^{-1} \cdot b$$

Y a continuación se resuelve el problema como:

$$\lambda^* = H^{-1} \cdot (-A^T \cdot \beta)$$

Para implementar esta solución en Matlab se añadió a la base de datos una nueva columna en la que se guardasen los errores a un paso. Los valores de esta columna son los que posteriormente formarán parte del vector c.

El método de selección de los puntos sigue siendo el mismo, es decir este algoritmo y el utilizado en la primera parte seleccionan los mismos puntos en una situación determinada, sin embargo, cada uno otorga un peso distinto a las acciones de control.

Una vez definidos los vectores se calcula la matriz H^{-1} utilizando el lema de inversión

Después se ha resuelto el problema dual y esta solución se ha utilizado el operador \ pues está más optimizado que el proceso de calcular la inversa y después multiplicarla por b.

3. SIMULACIONES ALGORITMO EN FUNCIÓN DE GAMMA.

Una vez implementado el algoritmo que tenía en cuenta el error a un paso se procedió a comprobar como afectaba el término γ al funcionamiento del sistema en simulación.

Para realizar las simulaciones se utilizó un modelo identificado mediante la herramienta ident de Matlab.

El modelo utilizado para las simulaciones ha sido:

$$A(z) \cdot y(t) = B(z) \cdot u(t) + e(t)$$

Donde:

$$A(z) = 1 - 1.6z^{-1} + 0.04825 \cdot z^{-2} + 0.16131 \cdot z^{-3} - 0.02817 \cdot z^{-4}$$

$$B(z) = 0.00307z^{-1} + 0.001297z^{-2} - 0.001808z^{-3} + 0.02922z^{-4}$$

Las simulaciones se han lanzado considerando e(t)=0. La base de datos utilizada para realizar estas simulaciones ha sido generada utilizando el modelo y métodos análogos a los explicados para el sistema real.

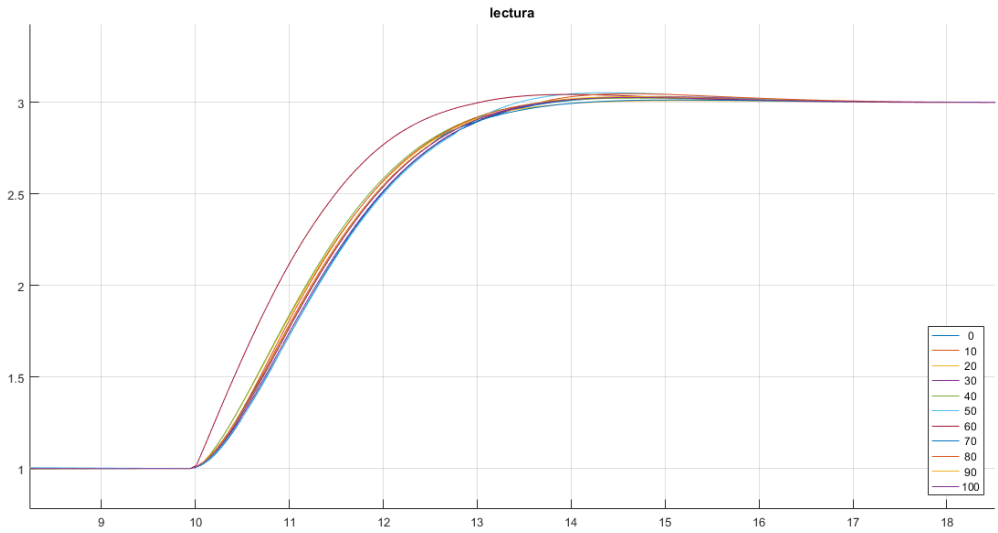

Fig 37: Simulación 1

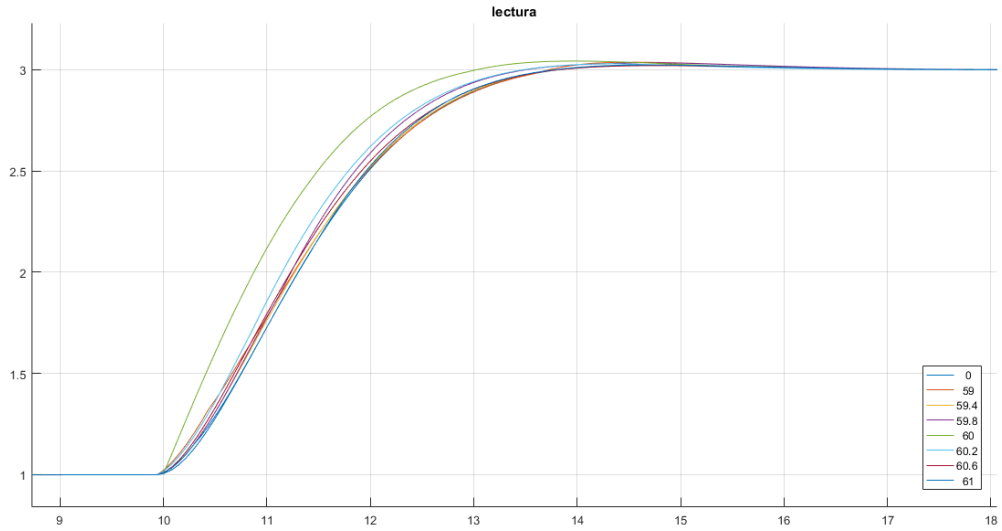

Fig 38: Simulación 2

En la Fig 37 se muestra la evolución de la lectura para una variación de γ de entre 0 y 100, aumentando de diez en diez, se observa que existe un valor de gamma que hace que el comportamiento del sistema funcione mejor.

En la Fig 38 se muestra la evolución del sistema para un valor de γ en el entorno del γ identificado como que hace mejorar el sistema.

En todo momento se muestra el resultado de $\gamma = 0$ dado que equivale con los resultados del problema sin realizar las modificaciones.

Se observa que el control es sutilmente más agresivo al aumentar la gamma y que la agresividad de este control presenta un máximo, en este caso para $\gamma = 60$.

4. IMPLEMENTACIÓN Y ENSAYOS

4.1 Implementación de un filtro:

Al someter al sistema a un ensayo en el que a bucle abierto se le somete a una actuación constante se observa que el sistema tiene mucho ruido en la lectura y una dinámica en la que, para una señal constante la lectura no se estabiliza entorno a un punto de funcionamiento, sino que subyace una dinámica más lenta, de manera que el sistema no llega realmente a estabilizarse. Esto se debe al calentamiento o enfriamiento del chasis del equipo que es más lento que la dinámica de calentamiento y enfriamiento del aire.

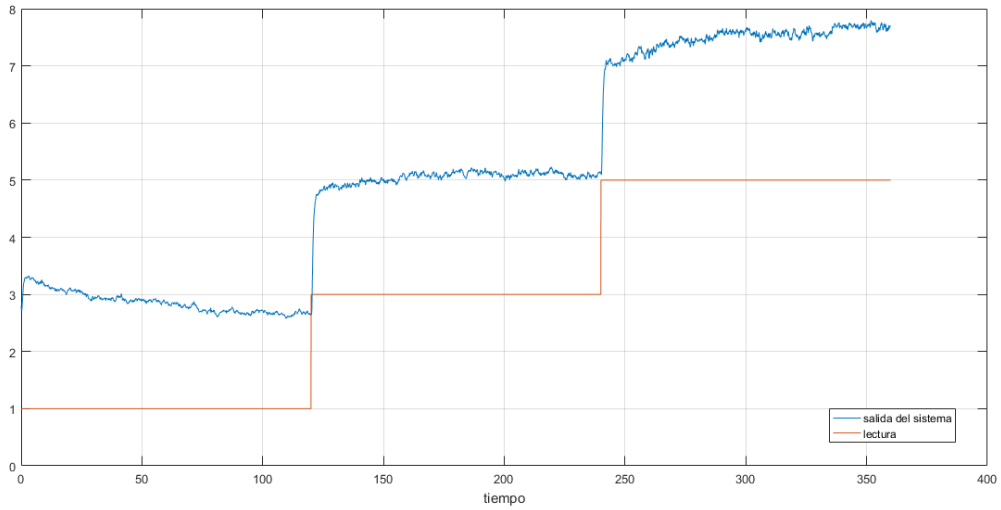

Fig 39: Comportamiento en bucle abierto frente a entradas en escalón constante.

Para disminuir el rizado de la lectura se ha implementado un filtro, tanto para generar la base de datos como ya se ha mencionado anteriormente como para realizar los cálculos en de acción de control.

Esto hace que los cálculos del algoritmo sean menos sensibles a este ruido obteniendo un mejor comportamiento del sistema

El filtro se ha ajustado mediante ensayo y error y se ha implementado en Simulink mediante una función de transferencia discreta:

$$F(z) = \frac{0.04 \cdot z}{z - 0.96}$$

También se ha utilizado en los ensayos para filtrar las lecturas y calcular la acción de control implementándose como:

$$y(t) = 0.96 \cdot y(t-1) + (1 - 0.96) \cdot data(t)$$

donde:

$y(t)$ Lectura actual filtrada

$y(t-1)$ Lectura filtrada en el tiempo de muestreo anterior

$data(t)$ Lectura actual sin filtrar

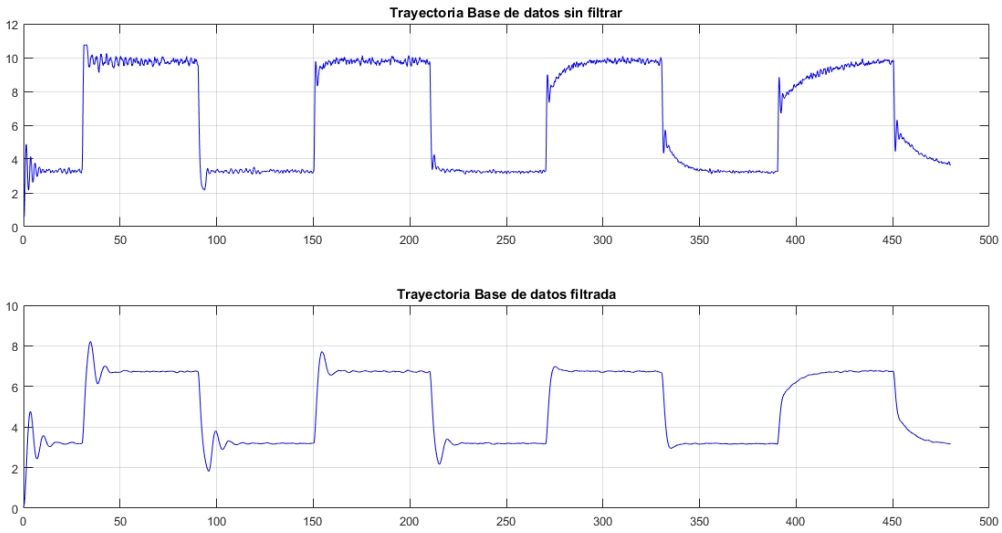

Fig 40: Trayectorias con filtro y sin filtro

4.2 Ajuste del cero

Debido a las características del sistema, la temperatura ambiente es un parámetro que no podemos controlar. Para evitar que el cambio de temperatura ambiente afectara a los experimentos se procedió a ajustar el parámetro balance.

Para ello se activaba una acción de control constante de 0 y se ajustaba el parámetro para que la lectura del sensor fuera 0.

Este ajuste se llevaba a cabo antes de empezar a tomar datos para una base de datos, y antes de empezar las pruebas.

Dado que la generación de una base de datos de 300 trayectorias tomaba unas 40 horas y bien se han hecho a lo largo de un fin de semana bien se han hecho realizando ensayos durante las noches y uniendo los resultados de varios días. Este ajuste disminuye la influencia de la temperatura ambiente sobre la base de datos, aunque no lo elimina.

4.3 Seguimiento de una trayectoria

Para realizar estos experimentos se procedió a dar una serie de escalones de referencia.

En el primer experimento se utilizó una base de datos que solo tenía escalones hacia arriba, por lo que solo se le dieron escalones hacia arriba durante el ensayo.

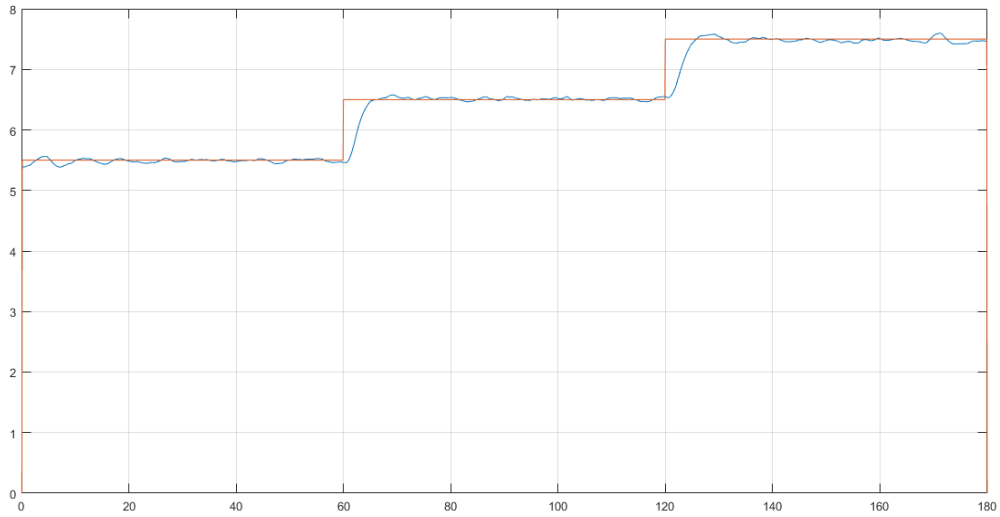

Fig 41: Ensayo de seguimiento de la trayectoria (1)

Para obtener la Fig 42 se utilizó una base de datos que contenía saltos hacia arriba y hacia abajo. Por lo que esta dinámica sí estaba recogida en la base de datos.

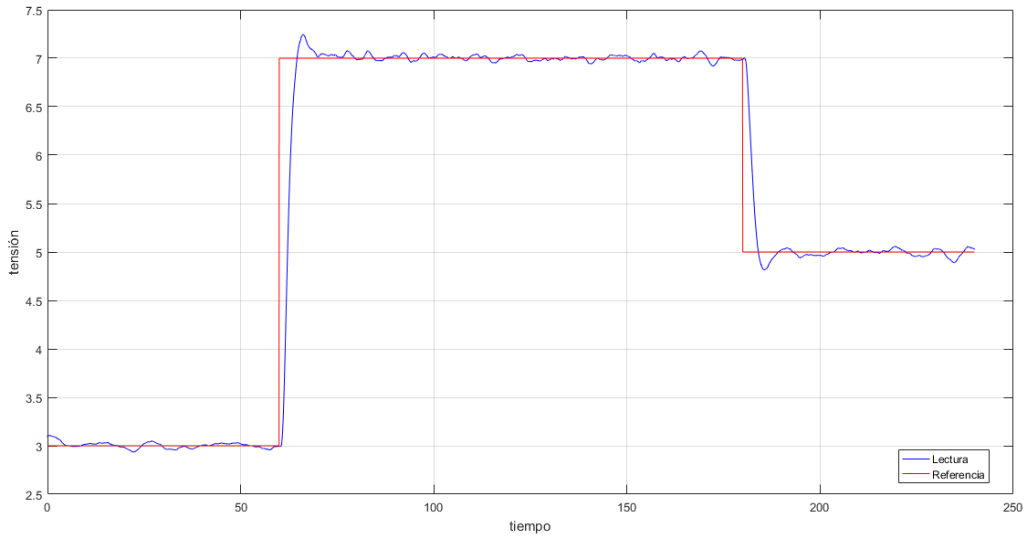

Fig 42: Ensayo de seguimiento de la trayectoria (2)

A pesar de que los ensayos se llevaron a cabo en días distintos, y con bases de datos distintas se aprecia que el sistema sigue la referencia y el seguimiento se produce sin offset.

4.4 Evolución del Sistema al cambiar el valor de Gamma.

En este experimento utilizando una misma base de datos se procedió a darle al sistema un escalón de referencia ascendente y otro descendente. Se repitió varias veces dándole a Gamma los valores 0, 1, 3, 5.

En la Fig 43 se muestra el comportamiento del sistema durante todo el ensayo en la Fig 44 se muestra el detalle del escalón ascendente en el que se observa con más claridad que al aumentar el valor de gamma la evolución del sistema es más agresiva. Esto se produce para valores relativamente pequeños gamma. Si gamma es demasiado grande el sistema puede llegar a inestabilizarse.

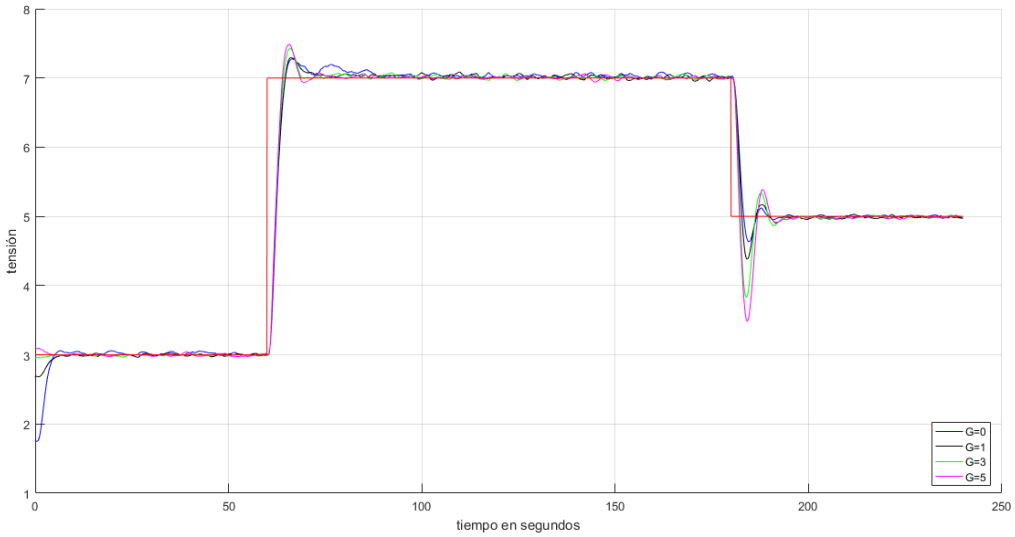

Fig 43: Comparación del funcionamiento del sistema con distintos gamma

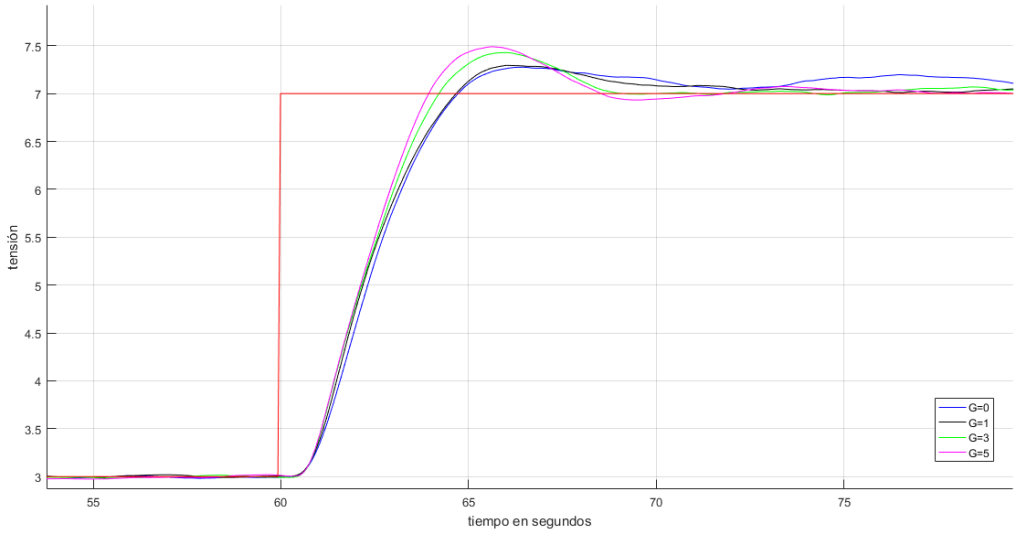

Fig 44: Comparación del funcionamiento del sistema con distintos gamma (detalle)

4.5 Rechazo de perturbaciones.

Para perturbar el sistema y realizar los ensayos se procedió a modificar el parámetro throttle control. Este parámetro representa la velocidad del ventilador. Al aumentar este parámetro un mayor flujo de aire recorre el tubo, por lo que el calefactor sobre el que tiene efecto el actuador calienta menos el aire, dicho de otro modo, al aumentar el parámetro, el actuador tiene menos efecto sobre la salida.

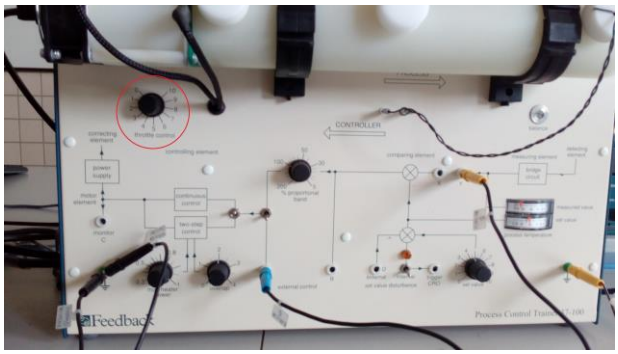

Fig 45: Situación del actuador

Fig 46: Valor sin perturbar

Fig 47:Ejemplo de perturbación

Es importante tener en cuenta que esta perturbación no es constante, sino que tienen una forma parecida a la que tienen los resultados ante una acción de control constante, es decir, que poco a poco el resultado va subiendo o bajando debido al calentamiento o enfriamiento del chasis, en consecuencia, esta perturbación es más parecida a la superposición de un escalón con una rampa. Dado que nuestro sistema es de primer orden, el rechazo de estas perturbaciones es complicado. Sin embargo, el comportamiento es bueno en variaciones pequeñas.

a) En este primer ensayo se perturbó el sistema modificando el throttle control desde de 5 a uno de 8 durante un ensayo de seguimiento de una trayectoria.

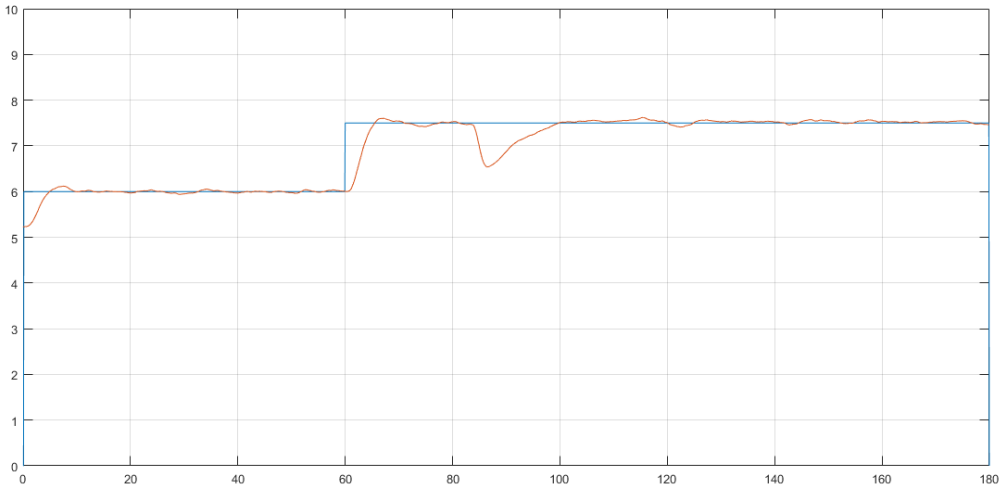

Fig 48: Ensayo de perturbaciones (1)

b) Para hacer este ensayo se mantuvo el sistema con una acción de control constante para que partiera de condiciones iniciales de equilibrio, a continuación, se procedió a controlar el sistema en una referencia cercana a estas condiciones utilizando el algoritmo. Un minuto después se modificó el valor del throtle de 5 a 7. Se dejó que se estabilizara durante dos minutos y posteriormente se volvió a modificar el valor de throtle original.

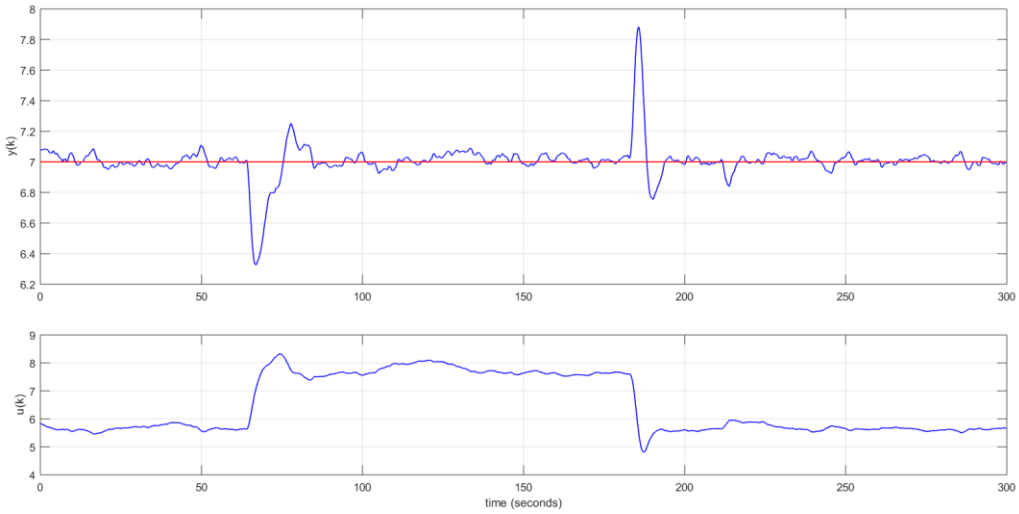

Fig 49: Ensayo de perturbaciones (2)

4.5.1 Rechazo de perturbaciones usando la base de datos de regulación.

En el proceso de realización de los ensayos anteriores se dieron perturbaciones de hasta medio voltio y se conseguía que el sistema funcionase sin offset, sin embargo, al dar perturbaciones mayores el sistema puede llegar a sobrecompensar la perturbación, es decir si la perturbación hace subir la tensión 0.5V, el régimen permanente presenta un offset por debajo de la referencia.

Para intentar mejorar este comportamiento se generó la base de datos de regulación de la que se ha hablado previamente.

Para que esta base de datos funcionase se modificó levemente el código del algoritmo ya que al ser la referencia igual para todas las trayectorias aparecía una columna de valor la referencia y la matriz perdía rango.

Sin embargo, no se observa una mejora del funcionamiento del sistema.

4.6 Rechazo de perturbaciones generadas por software.

Tal y como se ha indicado en el apartado anterior, las perturbaciones inducidas mediante el uso del *throttle* control no son constantes. Para probar el comportamiento del algoritmo ante una perturbación se procedió a sumar una entrada constante a la lectura del filtro. Esto se hizo de esta manera para evitar filtrar la perturbación y que fuese constante.

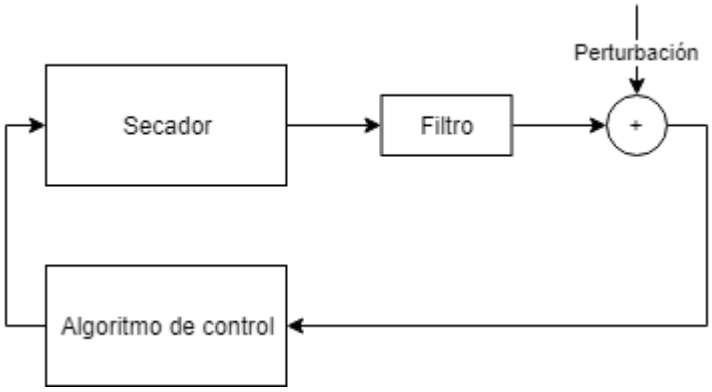

Fig 50: Esquema implementado para generar perturbaciones en escalón.

Con este método es posible dar perturbaciones mayores al sistema. En los ensayos mostrados en la Fig 51 se han dado perturbaciones de hasta 2 voltios.

Tal y como se observa en la Fig 52, para una Q de 1000 el controlador empieza a presentar un pequeño offset al corregir la perturbación. Sin embargo, este comportamiento mejora al aumentar el número de candidatos, permitiendo corregir este offset, tal y como se observa en la Fig 53.

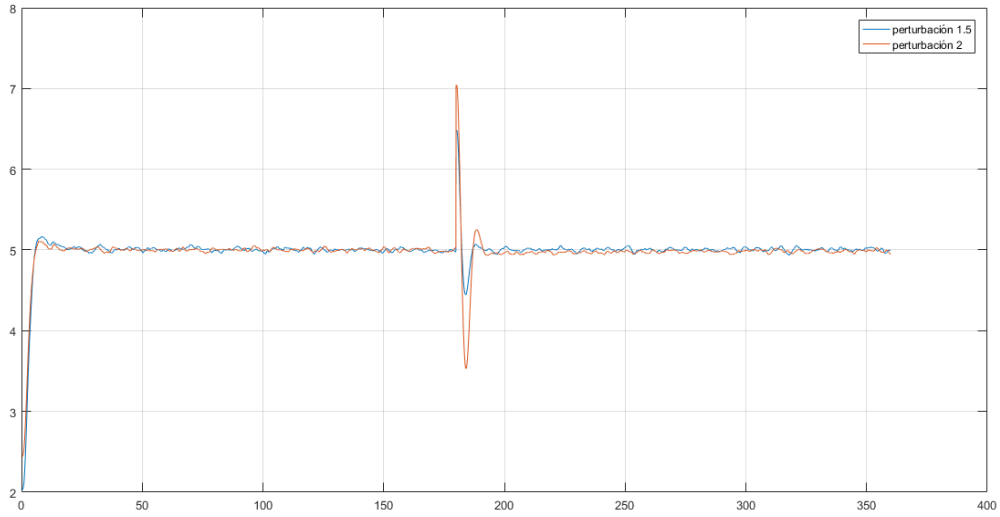

Fig 52: perturbación de 1.5, sin offset, comparada con una perturbación de 2. Número de candidatos 1000

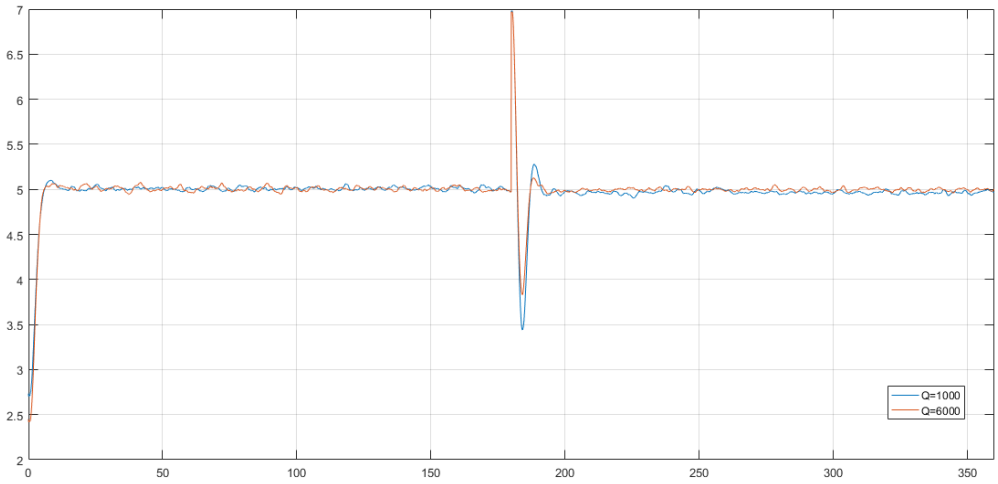

Fig 53: Perturbación de amplitud 2 para distinto número de candidatos.

5. CONCLUSIONES

En este trabajo se han hecho varias bases de datos utilizando distintas estrategias, también se ha probado el algoritmo de control basado en datos y se han realizado algunas modificaciones. Con todo esto se ha comprobado que la estrategia de control basada en datos es capaz de controlar un sistema real sin offset. Esto se ha observado tanto en ensayos de seguimiento como ensayos de rechazo de perturbaciones.

También se comprueba que el buen funcionamiento del algoritmo depende del número de candidatos que se utilice. En general, cuanto mayor sea Q mejor funcionará el algoritmo, aunque esta tendencia no tiene porqué ser creciente indefinidamente.

Con respecto este parámetro también es importante hacer notar que si es demasiado pequeño es posible que el problema deje de estar bien condicionado y que la solución no exista.

En la Fig 54 se muestra un ensayo de seguimiento realizado con 250 candidatos y se han marcado los puntos en los que el problema no estaba bien condicionado, dando lugar a soluciones erróneas.

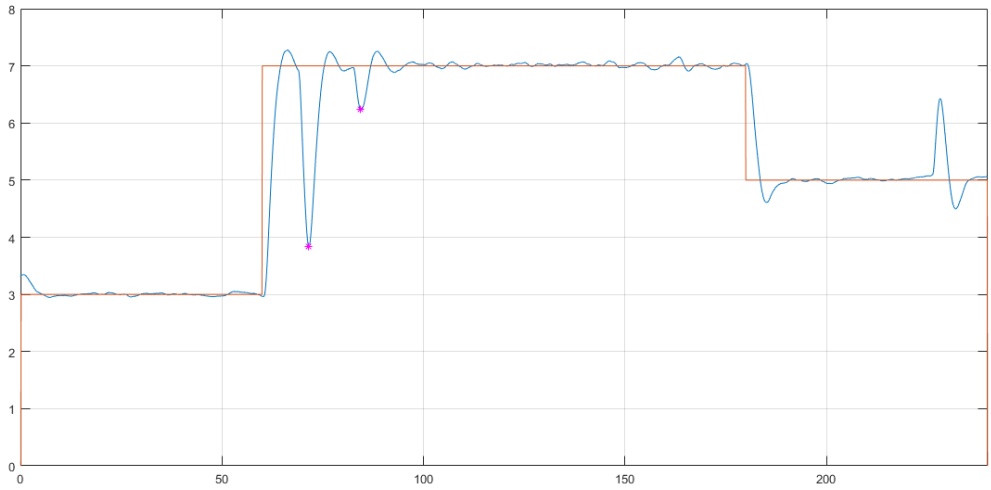

Fig 54: Fallos de condicionamiento del problema

La importancia de la información contenida en la base de datos se hace patente al observar como el sistema pasa de tener un comportamiento subamortiguado o sobreamortiguado dependiendo de la misma.

Esto queda claro al hacer ensayos utilizando una base de datos en la que se seleccionan siempre la primera trayectoria, es decir aquella con una constante integral mayor y otra en la que solo está incluida la cuarta trayectoria, es decir, aquellas trayectorias con menor constante integral.

La Fig 1Fig 55 se han generado dos trayectorias cada una con una de las bases de datos mencionadas anterior mente y se observa que, si bien el control basado en datos sigue funcionando adecuadamente, el

comportamiento en el transitorio es totalmente distinto.

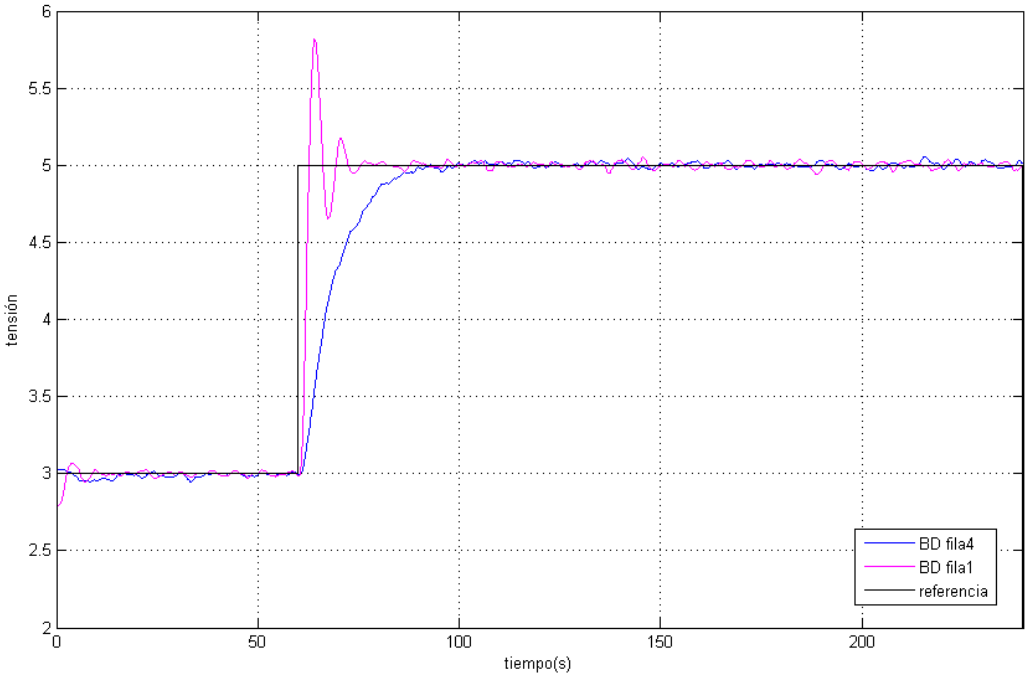

Fig 55: Evolución para distintas bases de datos

Por último, se ha observado que el parámetro gamma puede hacer que la actuación del controlador sea más o menos oscilante, sin embargo, el mejor valor de este parámetro para un sistema dado también puede depender de la base de datos y de las características de la misma.

5.1 Líneas de mejora

Algunas posibles modificaciones que podrían clarificar y mejorar el funcionamiento del control basado en datos son:

- Generar una base de datos que recoja el funcionamiento del sistema ante perturbaciones. El hecho de que este comportamiento esté recogido en la base de datos podría hacer que mejorase el rechazo de perturbaciones.
- Medir o determinar cuáles son las características que debe tener la base de datos y determinar una manera de medir como de buenas son estas características para poder determinar una mejor estrategia de selección de la información y construcción de la misma.
- Optimización del algoritmo con el fin de poder aumentar el tamaño de la base de datos, y el número de candidatos, así como para poder utilizarlo en aplicaciones con tiempos de muestreo más rápidos o incluso en aplicaciones de tiempo real.

ANEXO I: FUNCIONES LANZAR QUARC

En este anexo se explican las funciones utilizadas para lanzar los modelos de Quarc Simulink por comandos. Se explica como utilizarlas así como qué hacen.

rtwbuild

Esta función sirve para compilar el modelo. Y es equivalente a pulsar el botón compilar del modelo de Simulink.

Sintaxis: `rtwbuild('Nombre_del_modelo')`

set_param

Esta función sirve para modificar los parámetros de un objeto.

Sintaxis: `set_param('objeto','Parámetro',valor,… )`

En este caso se ha modificado el valor del parámetro 'SimulationCommand'dos veces consecutivas.
En el primer caso para conectar el modelo a la tarjeta.

```
set_param(model,'SimulationCommand', 'connect');
```

A continuación se ha vuelto a modificar este parámetro para dar comienzo a la simulación.

```
set_param(model, 'SimulationCommand', 'start');
```

ANEXO II: DATA ADQUISITION TOOLBOX

En este anexo se explican brevemente como utilizar las funciones de adquisición de datos disponibles en Matlab. También se indica brevemente cuales se han utilizado.

Daq

Es el nombre del toolbox.

daq.getDevices

Devuelve un vector con la información de todos los elementos conectados al ordenador. Cada fila del vector esa estructura de tipo (daq.DeviceInfo) los campos de dicha estructura son:
·index: Número de fila
·Vendor: Nombre del fabricante
·Device ID: Identificación del dispositivo
·Description: Modelo del dispositivo.

daq.createSession

Devuelve un objeto de tipo daq.Session que representa una sesión de comunicación con el hardware, es decir genera una sesión de comunicación identificación del fabricante.

Sintaxis: `daq.createSession(vendorID)`

Donde vendor ID es una identificación del fabricante. En este caso como el fabricante es National Instruments: `'ni'`

addAnalogInputChannel

Añade un canal de entrada analógica al elemento conectado a la sesión.

Sintaxis: `addAnalogInputChannel(sesión,deviceID,channelID,measurementType)`

Donde:

·sesión es el elemento de tipo `daq.Session` generado por daq.createSession

·deviceID es el número de identificación del elemento devuelto por `daq.getDevices`

·channelID representa el nombre del canal físico del dispositivo. Este nombre dependerá del equipo y viene habitualmente especificado en la documentación del mismo.

·measurementType: especifica el tipo de medida que va a realizar el equipo, en este caso la medida es de tensión luego el valor de este elemento es 'Voltaje'

addAnalogOutputChannel

Añade un canal de salida analógica al elemento conectado a la sesión.

La sintaxis de esta función es análoga a la función addAnalogInputChannel

outputSingleScan

Escribe sobre la salida del sistema durante un tiempo de muestreo.

Sintaxis: `outputSingleScan(sesión,data)`

Donde:

·sesión: Sesión iniciada mediante daq.createSesion

·data: información que se desea escribir sobre los canales, puede ser un número o un vector de dimensión igual al número de canales.

inputSingleScan

Devuelve una sola lectura adquirida por cada canal de entrada añadido en la sesión.
Existen dos posible sintaxis para esta función: `inputSingleScan(s)` e `s.inputSingleScan`
Donde:
·s: sesión iniciada mediante daq.createSesion

queueOutputData

Esta función se utiliza para enviar una señal de entrada al dispositivo.

Sintaxis: `queueOutputData(s,data)`

Donde

·s: Sesión iniciada

·data: matriz de dimensiones N×M donde N es el número de canales que tiene a sesión indicada y M el número de muestras.

Estas muestras se enviarán con la frecuencia de muestreo especificada en el parámetro s.Rate de la sesión.

StartForeground

Esta función lee de la sesión iniciada bloqueando la línea de comando y la ejecución de programas en Matlab, de manera que el programa espera los resultados de esta función para continuar la ejecución.

Sintaxis: `data = startForeground(s)`

Donde:

· data: información obtenida.

·s:Sesión iniciada.

La duración del escaneo está configurada por defecto para que dure 1 s o 1000 tiempos de muestreo.

Esta característica se puede cambiar usando s.DurationInSeconds = T, donde s representa la sesión y T la duración en segundos.

En el transcurso de este trabajo se han probado varias posibilidades, pero, dado que la señal dada al sistema no era un valor conocido a priori las funciones utilizadas definitivamente para la adquisición y escritura de datos han sido inputSingleScan y outputSingleScan

ANEXO III: CÓDIGO

En este anexo se explican los códigos y funciones principales utilizados durante este proyecto. Aunque se han generado más archivos casi todos los que han quedado fuera de este anexo no contienen más que pequeñas modificaciones.

Generación de Base de dato: Ensayos

```
clear all
close all
U0=0; %Acción de control de equilibrio
%% Declaración de variables
Array_select=[]; % en esta variable se guardan las trayectorias seleccionadas
Res_matrix=[];   % en esta variable se guardan todas las trayectorias del
experimento
model='seca_007_6'; %Modelo de Simulink que realiza el ensayo utilizando
Quarc
No=300 %número de ensayos a realizar.

%% Bucle con los ensayos
for l=1:No
Kp=1;
%Generación de los escalones de referencia
if rand(1)>0.5 %si un número aleatorio entre 1 y 0 es mayor que la mitad
    ref1=1+rand(1)*2.5;  %referencia inferior entre 1 3.5
    ref2=ref1+2+(7.5-ref1)*rand(1);%referencia superior entre referencia
inferior con un salto mínimo de 2 hasta 9.5
else% En otro caso salto hacia abajo
    ref1=rand(1)*4.5+5; %referencia superior desde cualquier punto entre 9.5
y 5
    ref2=(ref1-2)-(ref1-3)*rand(1);% referencia inferior desde dos por debajo
hasta 1
end
%Configuración para sacar Simulink
T=timer('TimerFcn', 'disp(''Sim end?'')', 'StartDelay', 485);%timer de la
duración del ensayo +5s
rtwbuild(model); %Compilar Modelo
set_param(model,'SimulationCommand', 'connect');%Conectar targeta
start(T) %Emperzar Timer
set_param(model,'SimulationCommand', 'start');%Ejecutar ensayo
wait(T) %Esperar Timer

%Carga de resultados
load('data.mat')
t=data(1,:)';%Vector de tiempos
y=data(2,:)';%Vector salida
u=data(3,:)';%vector acción de control
REF=data(4,:)';%vector referencia
ki=data(5,:)';%Vector con las constantes integrales

%Cálculo del segmento de tiempo de cada Ki
[N,~]=size(ki);
T_last=[1];
for i=1:N-1
    if ki(i)~=ki(i+1)% Si el valor de Ki cambia
        T_last=[T_last;i];% Guarda el instante de tiempo
    end
```

```
end
T_last=[T_last;N];%Añadimos el último tiempo de muestreo
[M,~]=size(T_last);

Res_array=[];% en esta variable auxiliar se guardan temporalmente las
trayectorias separadas
e1=[];% Vector de errores
for i=1:M-1 %Para cada trayectoria
    t_aux=t(T_last(i):T_last(i+1)); %Guardamos el valor de las variables
entre el instante inicial y el final
    y_aux=y(T_last(i):T_last(i+1));
    u_aux=u(T_last(i):T_last(i+1));
    r_aux=REF(T_last(i):T_last(i+1));
    ki_aux=ki(T_last(i));
    I=find(r_aux==ref2); %%Encuentra el trozo de trayectoria correspondiente
al escalón
    E1=sum((r_aux(I)-y_aux(I)).^2); %Cálculo del coste
    e1=[e1,E1];
    Resultados=struct('ref', r_aux,'output', y_aux,'input',u_aux,'t',t_aux,
'e', e1); %Guardamos la información en un Struct
    Res_array=[Res_array;Resultados];%Guardamos el struct en un vector
end
[~,Best]=min(e1);%Buscamos la posición del mínimo coste
Array_select=[Array_select; Res_array(Best)]; %Añadimos la mejor trayectoria
el vector de trayectoria
Res_matrix=[Res_matrix, Res_array];
 figure(1)
 plot(t,y,'b')
 hold on
 plot(Array_select(l).t,Array_select(l).output,'r')
 grid

file1=strcat('C:\Users\alumno\Desktop\20181024_3\copiseg\copi_seg',num2str(l)
,'.mat');
 save(file1);
 file2=strcat('F:\MII\TFM\copia_seguridad\copi_seg',num2str(l),'.mat');
 save(file2);
 end
```

Generación de Base de dato: Composición de matriz

```
clear all
close all
load('Array_UD_new.mat')%Cargamos el Array_select guardado del ensayo
[N,~]=size(Array_select);%número de ensayos en el vector
BD=[];%Declaración variable matriz
for i=1:N% Para cada trayectoria
    ref=Array_select(i).ref; %Guardamos el vector referencias en una variable
auxiliar
    ref_aux=ref(1);% referencia en instante de tiempo t=0
    ref_min=min(ref);% referencia mínima
    ref_max=max(ref);% referencia máxima
    if ref_aux==ref_min %si la referencia inicial coincide con la mínima
        aux2=ref_max; %es un escalón hacia arriba y entra a la BD el tramo de
referencia máxima
        display('up')
    else% en otro cas
        aux2=ref_min;% es un escalón hacia abajo y entra a la BD el tramo de
referencia mínima
        display('down')
```

```
    end
    I=find(ref==aux2);%buscamos en el vector la parte de la referencia
    time=Array_select(i).t(I)-Array_select(i).t(I(1));%Ponemos el vector de
tiempos para qeu empieze en 0.
    u=Array_select(i).input(I); %vector acción de control durante el escalón
    y=Array_select(i).output(I);%vector lectura del sistema durante el
escalón
    [n,~]=size(u);
    ID=i*ones(n,1);%Generamos unvector con el identificador de trayectoria
    ref=Array_select(i).ref(I);% modificamos el vector ref para que contenga
el tramo de referencia
    aux=zeros(n,1);
    %% adimos columna 7
    c=y(2:end)-ref(2:end);% vector de errores a un paso
    BD=[BD; ID(1:end-1),time(1:end-1),aux(1:end-1), y(1:end-1), u(1:end-1),
ref(1:end-1), c];%Añadimos los vectores a la base se datos
end
```

Ensayos: Ensayo de seguimiento tipo

```
clear all %limpieza del espacio de trabajo
close all
run('Matlab_NI.m') %ejecución del scrip que inicializa la sesión de trabajo
Array_DN=[]; %declaración de variable donde se guardan los ensayos
T_ensayo=240; %duración del ensayo en segundos
T_m=0.07;    %Tiempo de muestreo en segundos
muestras=ceil(T_ensayo/T_m);%Número de muestras que tendrá el ensayo.
datass=zeros(muestras,1);% inicialización de variables
y=zeros(muestras,1);
uss=zeros(muestras,1);
ref=zeros(muestras,1);
y=zeros(muestras,1);

%%Inicialización del bucle abierto
T_ci=20; %Tiempo del bucle abierto
muestras_0=ceil(T_ci/T_m);%muestras del bucle abierto
u0=zeros(muestras_0,1);%inicialización de variables
data0=zeros(muestras_0,1);

%%parámetros de la simulación
load('BD300_up_down.mat') %Carga de la Base de datos
DB_re=BD_300; %Cambio de nombre
%t_ambiente=input('Introduzca la Temperatura ambiente');
M=max(DB_re(:,1));%Número de trayectorias guardado en la base de datos
I=find(DB_re(:,1)==1);%localizamos una trayectoria
[LongSim,~]=size(I);%Calculamos la longitud de la trayectoria

Q=1500; %número de candidatos
G=[0,1, 3, 5]; %Valores que va a tomar Gamma
[~,G1]=size(G);
timers_h=[];% inicialización vector histórico de timers

%%Para cada trayectoria

ref2=7;
ref3=5;
for g=1:G1
    for j=1:muestras_0 %periodo de bucle abierto
        tic
        [data0(j),~]=s.inputSingleScan; %escritura
```

```
        u0(j)=3.5; %Se aplica una acción de control constante
        outputSingleScan(s,u0(j)); %lectura
        toc0=toc;
        T_0=ceil((T_m-toc0)*1000)/1000; %cálculo del tiempo de espera
        t_espera=timer('TimerFcn','nada', 'StartDelay',T_0); % timer, la
función nada está vacía
        start(t_espera)
        wait(t_espera)
    end
    datass(1:2)=data0(muestras_0-1:muestras_0);%escribimos las últimas dos
muestras en el vector de BC para inicializarlos.
    y(1:2)=datass(1:2);
    uss(1:2)=u0(muestras_0-1:muestras_0);
    %ref1=datass(1);
    ref1=3;
    for i=3:muestras %Para cada periodo de muestreo
        tic
        %Generación de los escalones de referencia
        if i<60/0.07 %si no hemos llegado al minuto
            ref(i)=ref1; %valor de referencia es ref1
        elseif i<180/0.07%si estamos en el segundo o tercer minuto
            ref(i)=ref2;% valor de referencia ref2
        elseif i<240/0.07 %tercer minuto
            ref(i)=ref3;%referencia 3
        end
        [datass(i),~]=s.inputSingleScan; %Lee
        y(i)=0.96*y(i-1)+(1-0.96)*datass(i); %filtrado de la lectura
        toc1=toc;
        tic
        uss(i)=CBDexplicit2_2(DB_re,y(i),y(i-1),uss(i-
1,1),ref(i),G(g),M,Q,longSim);%cálculo de la señal de control
        toc2=toc ;
        tic
        if uss(i)>=10 % saturación de la señal de control
            uss(i)=10;
        elseif uss(i)<=-10
            uss(i)=-10;
        end
        outputSingleScan(s,uss(i));%escritura de la señal de control
        toc3=toc;
        Taux=ceil((toc1+toc2+toc3)*1000)/1000;%tiempo empleado en el bucle
        time_rest=T_m-Taux; %el tiempo restante es el tiempo de muestreo más
        timers_h=[timers_h; toc1, toc2, toc3];%Guardamos los tiempos
        if time_rest>0%si el tiempo restante es mayor que 0
            t_espera=timer('TimerFcn','nada', 'StartDelay',time_rest);
%%sustituido display por void
            start(t_espera)%esperamos al timer
            wait(t_espera)
        else
            disp('timeout')%si ha tardado más del tiempo de muestreo avisa.
pero la simulación continua.
        end
    end
    outputSingleScan(s,3.5);
    D_N=struct('t', timers_h,'estado', datass,'filtrado',y,'control', uss,
'ref',ref, 'Q', Q, 'BD', M,'G', G(g));%generamos un struct con la información
    Array_DN=[Array_DN;D_N];% añadimos el struct al vector
    save('copi_seg');%hacemos una copia de seguridad del workspace
    figure(2)%dibujamos el ensayo
    hold on
    plot_y=[data0;y];
    plot(plot_y)
```

```
    figure(3)
    hold on
    plot_c=[u0;uss];
    plot(plot_c)

   end
outputSingleScan(s,3.5);
```

Ensayos: Ensayo con perturbaciones generadas por software.

```
%Este script es muy parecido al de un escalón de seguimiento.
clear all
close all
run('Matlab_NI.m')
Array_DN=[];
Array_DN_1=[];
T_ensayo=360; %s
T_m=0.07;    %s
muestras=ceil(T_ensayo/T_m);
datass=zeros(muestras,1);
y=zeros(muestras,1);
OUTss=ones(muestras,1);
t=zeros(muestras,6);
uss=zeros(muestras,1);
ref=zeros(muestras,1);
y=zeros(muestras,1);
pert=zeros(muestras,1);
%%variables contorno.
T_ci=20;
muestras_0=ceil(T_ci/T_m);
u0=zeros(muestras_0,1);
data0=zeros(muestras_0,1);

time_rest_h=[];
%%parámetros
load('BD_reg.mat')

DB_re=BD;
M=max(DB_re(:,1));

I=find(DB_re(:,1)==1);
[M1,~]=size(I);

longSim=M1;

Q=3000;
G=[0];
%%outputSingleScan(s,3.5);
j=1;
timers_h=[];
fl=0;
g=1;
PT=[0.5, 1, 2]; %% Amplitud del escalón de perturbación
ref2=5;

for g=1:3
    pert=zeros(muestras,1);
    for j=1:muestras_0
```

```
        tic
        [data0(j),~]=s.inputSingleScan;
        u0(j)=3.5;
        outputSingleScan(s,u0(j));
        toc0=toc;
        T_0=ceil((T_m-toc0)*1000)/1000;
        t_espera=timer('TimerFcn','nada', 'StartDelay',T_0); %%sustituido
display por void
        start(t_espera)
        wait(t_espera)
    end
    datass(1:2)=data0(muestras_0-1:muestras_0);
    y(1:2)=datass(1:2);
    uss(1:2)=u0(muestras_0-1:muestras_0);
    %ref1=datass(1);
    ref1=5;
    for i=3:muestras %
        tic
        if i<60/0.07 % Generación de un escalón de referencia
            ref(i)=ref1;
        else
            ref(i)=ref2;
        end
        [datass(i),~]=s.inputSingleScan; %LEctura
        if i<180/0.07 %Generación del escalón de perturbación
            pert(i)=0;
        else
            pert(i)=PT(g);
        end
        y(i)=0.96*y(i-1)+(1-0.96)*(datass(i)+pert(i));%Suma de la
perturbación a la entrada y filtrado
        toc1=toc;
        tic
        uss(i)=CBDexplicit1_3(DB_re,y(i),y(i-1),uss(i-
1,1),ref(i),M,Q,longSim);% Algoritmo ligeramente modificado para BD de
regulación

        toc2=toc ; % La programación a partir de aquí es idéntico al de un
ensayo de seguimiento.
        tic
        if uss(i)>=10
            uss(i)=10;
        elseif uss(i)<=-10
            uss(i)=-10;
        end
        outputSingleScan(s,uss(i));
        toc3=toc;
        Taux=ceil((toc1+toc2+toc3)*1000)/1000;
        time_rest=T_m-Taux; %el tiempo restante es el tiempo de muestreo más
        timers_h=[timers_h; toc1, toc2, toc3];
        if time_rest>0
            t_espera=timer('TimerFcn','nada', 'StartDelay',time_rest);
%%sustituido display por void
            start(t_espera)
            wait(t_espera)
        else
            disp('timeout')
        end
    end
    e=sum((datass-ref).^2);
    outputSingleScan(s,3.5);
```

```
     D_N=struct('t', timers_h,'estado', datass,'filtrado',y,'control', uss,
'ref',ref, 'Q', Q, 'BD', M,'G', G(1),'pert',pert );%,'ID',g);
    Array_DN=[Array_DN;D_N];
    save('copi_seg');
    figure(2)
    hold on
    plot_y=[data0;y];
    plot(plot_y)

    figure(3)
    hold on
    plot_c=[u0;uss];
    plot(plot_c)

   end
outputSingleScan(s,3.5);
```

Matlab_NI: Inicialización y configuración de la conexión

```
clear all% Limpiamos el workspace
devices = daq.getDevices;%Escaneamos los equipo conectados
s = daq.createSession('ni');%En este caso solo nos devuelve uno
addAnalogInputChannel(s,devices.ID, 0, 'Voltage');%Configuración del canal de
entrada
addAnalogOutputChannel(s,devices.ID,'ao0','Voltage');%Configuración del canal
de salida
R=1e5;%Tasa de muestreo.
s.Rate = R;
```